Vascular pattern recognition for finger veins using biometric graph matching

Vincent Nibbelke[1]

Abstract— **This paper aims to find out how well the biometric graph matching (BGM) method, that has shown promising results on retina vein images with equal error rates (EERs) of 0.5%, is suitable for use on finger vein images, using the best performing available vein vessel network extraction method. The biometric graph matching method takes two extracted graphs and aligns them before trying to match the edges of the graphs. The amount of matched edge pairs are a measure for the similarity of the graphs. Graphs are obtained from images from the UTwente finger vein image database. Finger vein graphs are more complex to extract than retina vein images due to the nature of the images of the finger veins where low contrast leads to very noisy graphs. However since the BGM method is a matching algorithm that is robust against noise, we expect it to perform well on our graphs. As we want to improve the performance and find out how we can make the system more specific to finger vein images, several enhancements of the initial biometric graph matching method are examined, including a new proposed line distance that has a better balance between a difference in orientation and difference in length. The adaptation in the distance score and including graph pruning leads to EERs down to 0.93% using a 10% subset of the UTwente finger vein image database. However when using a 40% subset, EERs rise to 2.89%. When compared to the state of the art work (tested on the full dataset), EERs are as low as 0.37%, so our system does not perform well enough to compete with the state of the art work. High EERs are shown to exist due to the poor quality of the graphs. Certain veins are not always detected, and when detected, might still not be matched due to small side-branches. This demonstrates that the graphs created from finger veins using the best available vein extraction method (Miura's maximum curvature method) are not well suited for biometric graph matching.**

I. Introduction

In modern day many users of electronic equipment store sensitive information on their computers. To ensure privacy of this information, a need for biometric security systems, for example finger print comparison and iris comparison, has risen. In a biometric comparison two samples are examined to see how similar they are. With this we have a similarity score, on which we can perform a threshold to determine whether or not the two samples are from the same instance. The past few years finger print authentication has become very popular for unlocking devices like computers and mobile phones. However, since copying fingerprints has proven to be possible, the need for a more secure non-invasive system for personal identification has risen.

With vascular finger vein pattern recognition we aim to develop a more secure, yet reliable alternative for biometric authentication solutions like finger print comparison. Finger vein pattern comparison for authentication has several potential advantages. Finger veins are internal features of the finger, so they will be harder to copy than external features. Also we can perform a liveliness detection as discussed by [13]. Previous research has shown several methods have been developed and succeeded in obtaining error rates below 1% (also see Section II).

A weighted Local Binary Pattern (LBP) method using a support vector machine (SVM) is used by [2], reporting equal error rates (EERs) of 0.049%. EERs as low as 0.37% using the principal curvature method with adaptive histogram equalization (AHE) are reported by [13].

We decided to investigate the biometric graph matching (BGM) method, which has been developed for retina vein images where EERs of 0.5% are reported [34]. The BGM method is a structure based method which does not look at the vein image as an image, but actually looks at the structure of the veins.

Images from finger veins are low in contrast, which is likely to give noise in our results when we try to extract graphs. An example of two of an image is given in Figure 1.

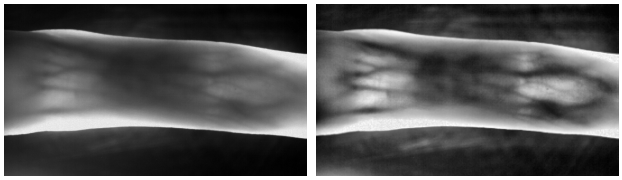

Fig. 1. Sample image of a finger before and after adaptive histogram equalization (AHE)

As the BGM method is tolerant to noise in graphs according to [34], we believe it may work well on graphs created from finger vein images as well. The purpose of this research is to find out how well biometric graph matching, that has been proven to be a reliable method for authentication on retina vein images, will perform in finger vein authentication systems. This leads to a number of research questions:

1) Is the biometric graph matching method as described by [34] a suitable method for comparing finger vascular patterns, given that we use the best performing existing vein vessel network extraction method given in [13]?
   - How does the performance of this basic system compare to the state of the art work?

[1]V. Nibbelke is the with Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, 7500 AE Enschede, The Netherlands v.nibbelke@alumni.utwente.nl

2) How can we improve the performance of our implementation of the biometric graph matching method?
   - Is there another vein vessel network extraction method which gives better results?
   - What are the best parameter settings?
   - Why are these the best settings and best vein vessel network extraction method?
3) How can we make the biometric graph matching method more specific to finger vein authentication in order to improve the recognition performance?
   - What would be the influence of including the width of the veins as a cost-label on the performance?
   - Which other improvement points are there that we can explore?

For this we will describe related work (Section II), the used method in Section III and our implementation of the system in Section IV. Results are given in Section V. We will analyze and discuss the results and determine whether or not the biometric graph matching method is a viable method for authentication using finger vein images in Section VI and say something about future work in Section VII.

II. Related Work

In the past research has been done on finger vein images using several methods. It is important to be able to compare our results with the results of other research. In order to get an idea of the different results from recent research, a short summary of the results of a few papers is given in Table I. Table II will give an overview of the results of several methods used on the Peking Database V4 and the UT Database, both with and without adaptive histogram equalization (AHE) [13].

Results in Table I are hard to compare due to the different datasets, and limited information on exact procedures regarding the image acquisition and the way the methods were implemented, etc. Therefore it is hard to directly compare results. To make our results comparable to other methods, we will perform our experiments on the same database as used by [13] for Table II.

III. Method

Biometric graph matching is a method for matching two graphs to determine whether it's from the same instance or not. It is a vein vessel network-based method that uses graphs to represent the structure of the vein vessel networks. The different image samples might vary due to noise, rotation, translation, scaling, illumination or other factors, so called capture imperfections. This leads to noisy spatial graphs. This means graphs that are similar but not exactly the same. The BGM method was chosen as it is an error- and noise tolerant matching algorithm. The BGM method is described in [20], [34], [35] consists of several steps, which are shown in Figure 2.

Firstly a skeleton of the veins must be extracted. This is then converted into a graph. After this the biometric graph

Authors	Method	Database	EER (%)
[8]	normalized cross-correlation	678 persons, 1 finger per person, 2 images per finger	0.00
[29]	repeated line tracking	678 persons, 1 finger per person, 2 images per finger	0.145
[30]	maximum curvature	678 persons, 1 finger per person, 2 images per finger	0.0009
[31]	multiscale, curvelets	400 persons, 8 fingers per person, 1 image per finger	0.128
[9]	local binary patterns	60 persons, 8 fingers per person, 10 images per finger	0.081
[2]	weighted local binary patterns	120 persons, 8 fingers per person, 10 images per finger	0.049
[32]	restoration, skeletonization	80 persons, 8 fingers per person, 10 images per finger,	0.76
[33]	principal curvature	unknown amount of persons, unknown amount of fingers per person, 29 unique fingers, unknown amount of images per finger, 118 images in total	0.36
[10]	wide line detector	5208 persons, 1-4 fingers per person, 10140 unique fingers, 5 images per finger	0.87
[26]	radon transform	10 persons, 1 finger per person, 10 images per finger	0.01
[24]	skeletonization	102 persons, 8 fingers per person, 10 images per finger	1.164
[28]	local derivative pattern	30 persons, 8 fingers per person, 10 images per finger	0.89
[3]	2D-Gabor + feature extraction	50 participants, 250 finger vein images (1 finger, 5 images)	0.79
[7]	Rotation invariant: sobel edge + SIFT vs. LBP	95 persons, 11 images per person (1 finger)	1.71-2.98

TABLE I
OVERVIEW OF DIFFERENT PAPERS AND THEIR RESULTS AS GIVEN BY [13] WITH ADDITIONAL PAPERS.

Method	Original Paper	Peking Database V4		UT Database	
EER (%)		No AHE	AHE	No AHE	AHE
Normalized cross-correlation	0.00	14.67	9.81	3.15	1.99
Maximum curvature	0.00	1.22	1.32	0.63	0.49
Repeated line tracking	0.15	6.75	5.90	1.04	0.99
Principal curvature	0.36	2.72	2.20	0.89	0.37
Wide line detector	0.87	4.66	2.73	1.72	0.89

TABLE II
OVERVIEW OF RESULTS OF RESEARCH CONDUCTED BY [13]. SEVERAL METHODS WERE TESTED OM BOTH THE PEKING DATABASE V4 AND THE UT DATABASE. ALL EXPERIMENTS WERE PERFORMED BOTH WITH AND WITHOUT ADAPTIVE HISTOGRAM EQUALIZATION (AHE).

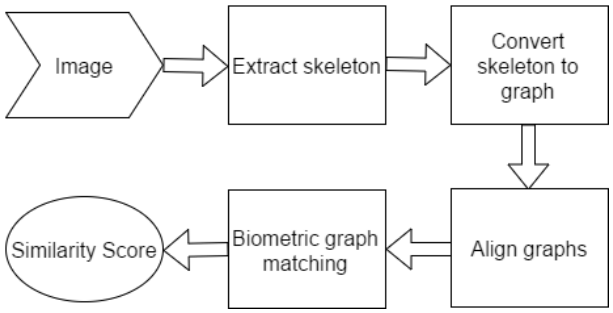

Fig. 2. Schematic overview of our method. In order to perform a biometric graph matching, skeletons must be derived from the images, which are then converted into graphs. These graphs are aligned and matched, giving a similarity score

.

matching is performed, which gives a similarity score. This similarity score is a measure for how similar the graphs are. These steps will now be explained in more detail.

A. Skeleton creation

In order to create a skeleton from the veins, several steps must be taken. An overview of this is given in Figure 3. Firstly there is some image enhancement taking place to enhance contrast of the given image. An example of this can be found in Figure 1. Now the image is enhanced in such a way that the vessels are more clearly visible, the skeleton extraction takes place. More on how this is implemented can be found in Section IV. An example of a skeleton can be found in Figure 4.

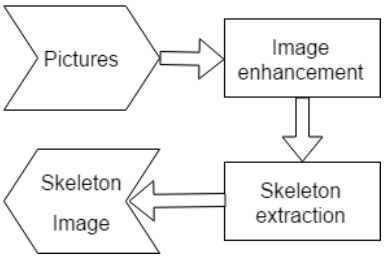

Fig. 3. Schematic overview of the high level idea of converting pictures into skeletons

The skeleton has termination points (endpoints of edges), Y-splittings (bifurcations) and X-splittings (crossovers) (the latter two being two types of branchpoints). Termination point paths might be due to noise if they are small, and can be removed if their branches are smaller than a certain amount of pixels (in [34] for example they use 15 pixels).

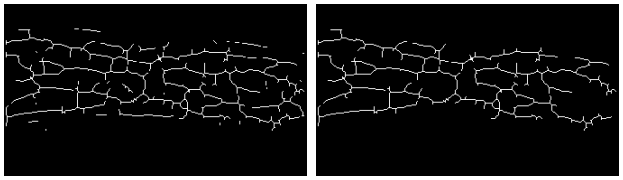

Fig. 4. Sample image of a skeleton found before and after throwing away small noise bits

.

B. Graph creation

Once the skeleton is found, it is going to be represented as a more abstract graph, where small noisy fluctuations in edges are not relevant anymore. A high-level overview of this is given in Figure 5.

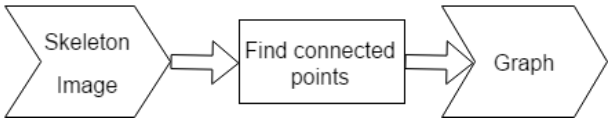

Fig. 5. Schematic overview of the high level idea of converting pictures into skeletons

The skeletons are converted to a graph by looking at the connected branch- and endpoints in the graph. In order to match two graphs, we need to define graphs in a uniform way. The graph representation is defined as:

$$g = (V, E, \mu, \nu) \tag{1}$$

In this V is our set of feature points (bifurcation-, end- and branchpoints) and E is the set of connected pairs of vertices, which form an edge (so which points are connected by a edge). Bifurcation points are a special type of branchpoints, specifically where 1 vein splits to two veins. $\mu : V \to \mathbb{R}^2$ maps each vertex (point in a graph) v to the corresponding Cartesian coordinates (q_1, q_2) (this is to be able to show the point in the image space). $\nu : E \to \mathbb{R}^2$ maps each edge (also called line) e to (l, θ) where the straight line between the connected pair of feature points has a length l and a slope θ. An example of this can be seen in Figure 6.

Fig. 6. Sample image of a graph

.

The different image samples might vary due to noise, rotation, translation, illumination or other factors, so called capture imperfections. This leads to noisy spatial graphs. This means graphs that are similar but not exactly the same. The Biometric Graph Matching algorithm is a noisy spatial graph matching technique involving 2 parts: graph alignment and error-tolerant graph matching. A schematic overview can be found in Figure 7.

Two graphs are input, and will be aligned for the Ni best matching edge pairs. Every time a similarity score is calculated, and the score representing the best matching will be our similarity score. We will briefly discuss the alignment and similarity score now.

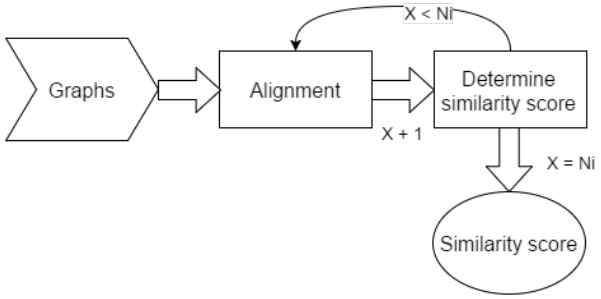

Fig. 7. Schematic overview of the high level idea of finding the similarity score between two graphs

C. Graph alignment and matching

Graph registration is done to ensure that both graphs are on same spatial frame, free from rotation and translation. This is done using a greedy RANSAC algorithm (this is described in detail in [34], Algorithm 1).

Every pair of edges is scored using a dissimilarity score based on the edge labels (length and orientation are used, but not the exact location, as this may be different due to the translation of the graphs). This dissimilarity score s_{ab}, as used by [34]:

$$s_{ab} = \frac{1}{0.5 * (l_a + l_b)} * \sqrt{(l_a - l_b)^2 + (\theta_a - \theta_b)^2} \qquad (2)$$

In which l_a and l_b are the lengths of the respective edges in graph a and graph b and θ_a and θ_b are their respective angles. When the lengths are equal and the angles are equal, the square root will go to zero, giving a distance between the edges of 0.The distance is normalized for the length by dividing by the sum of the lengths. The angle difference is not weighted compared to the length difference, which is why we have doubts on how this distance measure will reflect the reality. Therefore we will later introduce a new dissimilarity score, which we will explain in subsection III-D.

With this dissimilarity score s_{ab} we aim to find most similar edge pairs, these are considered to give the best alignment of the graphs we want to compare. For each of these edge pairs, the compared graphs are translated and rotated to make these two edges lay horizontal, starting in the origin.

The vertices of the translated and rotated graphs are compared using an Euclidean distance. If this distance is less than certain tolerance ε, we have a match. The number of matched edges (C) is used to calculate the similarity score (also called distance score) d_k:

$$d_k = 1 - \frac{C}{a_{max} * b_{max}} \qquad (3)$$

Here d_k is the similarity score between two graphs (0 being a total match, 1 being no match at all), C being the number of matching edges, a_{max} and b_{max} being the total number of edges in graph a and b. This distance score is calculated for the first Ni best matching edge pairs, as these are most likely to give a good alignment of the graph. The minimum distance score that is found among these matches is used as the distance score between graphs. From all Ni possible rotations that are tested, the one with smallest distance score is kept and used.

Once the similarity scores are calculated for all genuine matches and the impostor matches, we can threshold the distance scores. If the score is below the threshold value we decide that we have a match. With this we can find the equal error rate (EER).

D. System Improvements

Besides the initial implementation we also studied our second and third research question: How can we improve the performance of our initial implementation and how can we make it more specific to finger vein images?

One thing to investigate is pre-aligning the graphs in order to make the angle differences as small as possible before starting. If this would work well, we could perhaps omit looking at the rotation in the step of aligning the graphs. Together with this, besides sorting the edge pairs by s_{ab}, sorting by angle difference ($\Delta\theta$) might be a good idea, especially after pre-rotation. Also an increase in Ni, the maximum number of attempts to align the graphs, might have a positive effect on the performance.

We will also investigate different vein extraction methods to see if this brings improvements in the system performance. Besides this we will also investigate the parameter values and how we might gain some performance improvements by tuning those some more. Also graph simplification by throwing out small edges will be investigated.

As shown in the initial implementation, we have a point where we sort the edge pairs to find the best matching edge pairs for the alignment. It was sorted by s_{ab}, but we could also sort by the angle difference $\Delta\theta$. For the dissimilarity score we do not only want to look at the length of the edges, but also the orientation is important. As the current dissimilarity score (see Equation 2) does not balance well between angle difference and edge difference, we introduce a new dissimilarity score that balances better between angle and edge length, which is defined as:

$$d_{norm}(s_1, s_2) = \frac{|l_1 e^{J\theta_1} - l_2 e^{J\theta_2}|}{\sqrt{l_1^2 + l_2^2}} \qquad (4)$$

This equation can be simplified to:

$$d_{norm}(s_1, s_2) = \sqrt{1 - 2\rho_1\rho_2\cos(\theta_1 - \theta_2)} \qquad (5)$$

In this equation, ρ is defined as:

$$\rho_i = \frac{l_i}{\sqrt{l_1^2 + l_2^2}} \qquad (6)$$

In this θ_1 and θ_2 are the angles of the edge in respectively graph 1 and 2. l_1 and l_2 are the lengths of the edge in the first and second graph. When both lengths and both angles are equal, the distance is 0. However when there is a difference in angle or length, the dissimilarity score will get larger, up

to 1. In this equation both lengths and angles are equally important and make a normalized and balanced dissimilarity score that will allow better edge matching.

Besides this we can also try to enhance the results by giving more weight to certain edges that are matched that come from wider or longer veins in the original image. For example, a vein with a length of 10 pixels would get a weight of 10, while a vein with a length of 2 pixels would get a weight of 2. In this way we could make the long and/or wide main veins more important than some sub-veins or noise.

As stated before, the graphs created with the maximum curvature method give quite noisy graphs with lots of side-branches. Therefore we want to simplify the graphs a bit, such that noisy edges will have less influence on the matching. One way is to remove all edges between branch-points and end-points smaller than N pixels. This process is called pruning.

Besides this, it is also possible to combine several smaller edges into one bigger edge. This is important because due to the noisy nature of the graphs, you might not get a proper match while they are in fact the same finger. A simplified example is given in Figure 8.

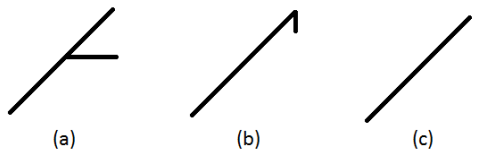

Fig. 8. Sample image of 3 fictional graphs of the same finger. In (a) a noisy side-branch is found in the middle, (b) shows an clear match with a noisy side-branch at the end, (c) shows a clean graph.

Three fictional graphs of the same instance are shown. In Figure 8(a) a noisy side-branch is found in the middle, (b) shows a clear match with a noisy side-branch at the end and (c) shows a clean graph. When we want to match the clean graph (c) to the graph with a noise edge at the end (b), we will have 1 edge that matches, which is a big edge, so the graphs are probably a match. However when we have a noise edge in the middle, like with (a), the big edge is split into two smaller edges, which implicates they are not the same edge (e.g. two small edges is not the same as the one big edge), and we thus have 0 matching edges, and therefore it's not a matching graph. Therefore we want to apply a edge-combination after the pruning to combine smaller edge-parts that were separated by the noisy side-branch before. An example of this is shown in Figure 9.

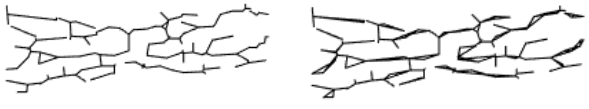

Fig. 9. Sample image where on the right the edge combinations are added to the graph of the left to enhance the matching performance.

Finally we want to see what the effect is of using the edge length as a weight-factor. For this, the way of calculating the similarity between graphs is slightly altered. This is now defined as:

$$d_k = \frac{C}{(\sum_{i=1}^{a_{max}} l_{a,i} + \sum_{j=1}^{b_{max}} l_{b,j})/2} \tag{7}$$

where C is defined as:

$$C = \sum_{n=1}^{m} \frac{l_{an} + l_{bn}}{2} \tag{8}$$

In this, l_{an} is the edge of the n'th matching pair of the first graph, l_{bn} is the edge of the n'th matching pair of the second graph and m is the total number of all matching edge pairs. With this we take the average length of the two edge, as they might not be exactly the same length, and both are equally likely as the real length. This total matching length C is now divided by the total average length of the edges in the graphs in order to determine a ratio of how much of the edges are matching. This way we make longer edges more important to match than smaller edges.

IV. Implementation

For the implementation of the Biometric Graph Matching method, several design choices were made. We made use of work previously done by [13]. Parts of the already available code is re-used for this research.

The system has been subdivided into three parts, such that we can investigate some intermediate results which we can save for use in the later parts. In this way we can reduce the calculation time by not re-computing the same intermediate results over and over again. These three parts are getting the skeleton of the veins from a finger vein image, creating a graph from the skeleton, and aligning and matching two graphs to get a similarity score.

A. Converting input images to skeletons

In Figure 10 a schematic overview of converting pictures to skeletons is shown. This section will explain the work-flow and give some example images to clarify the process.

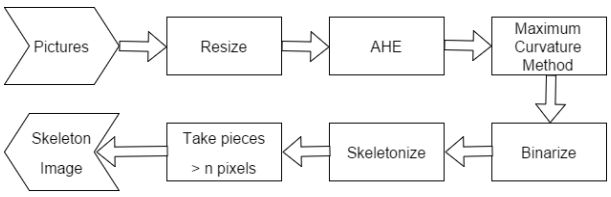

Fig. 10. The work-flow of the conversion of images into skeletons, where the input image is processed such that the finger vein skeleton is found

The original images are resized and an adaptive histogram equalization (AHE) is done to enhance the contrast of the image. An example result of this is shown in Figure 1.

These images are then input for Miura's Maximum Curvature Method as developed by [30] and implemented by [13]. This method is chosen as it showed the most promising EERs according to [12], [13], which is an indication for good finger vein extraction. Visual inspection confirms that this

method provides a proper finger vein vessel extraction. Next a threshold is applied to the result to obtain a binary version of the resulting image. This threshold (0.005 for the UTwente database, which will be described in more detail later) is experimentally determined by visual inspection, such that as little detail of the veins would disappear as possible. An example of the result of this method can be seen in Figure 11.

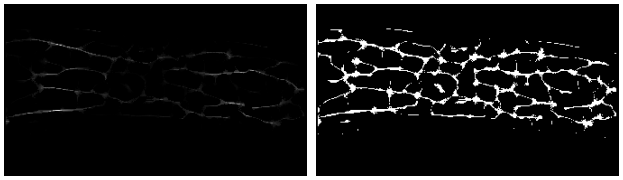

Fig. 11. Sample image of unbinarized and binarized detected veins

.

This image is used to make a skeleton, which is implemented by Matlab's skeletonize function. Components with less then n pixels are likely due to noise and are therefore removed. An example of this is shown in Figure 4. In our experiments we have chosen for n = 100 pixels, which, by visual inspection, has shown to keep the relevant parts of the vein vessel network. A part is relevant if it gives viable information about the structure of the vein vessel network. This gives us the skeleton image, which is an intermediate result.

B. Converting skeletons to graphs

Now the skeleton of the vessels is extracted, we can find the feature points; termination points, Y-splittings (bifurcations) and X-splittings (crossovers). We can find these points by scanning the skeleton with a 3x3 window for points where we only have 2 white pixels (end point) or more than 3 white pixels (branchpoint). Endpoint paths can be removed if their branches are smaller than a certain number of pixels (in [34] a minimum length of 15 pixels is used). A simplified schematic overview of this part is given in Figure 12. In this overview, together with the skeleton image, a list of the branchpoints (bifurcations and crossovers) and endpoints (termination points) is used for the conversion of the skeletons to graphs.

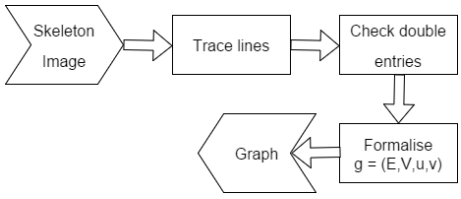

Fig. 12. The work-flow of the second part of the implementation, where the skeleton image is processed into a graph

.

In order to create a graph from a skeleton, we need to know which branch- and endpoints are connected and need to be an edge in the graph. For this we made a line tracer. The line tracer starts at a given branch- or endpoint and traces the line to the next branch- or endpoint. In this way the edges of the graph are found. All these edges are saved in a large list, which afterwards will be checked for double entries (tracing from A to B or from B to A gives the same edge). These will then be formalized into the graph as defined in Equation 1. An example of such a graph can be seen in Figure 6.

In practice we save our graph as the x- and y-positions of the vertices, and we save l and θ to make sure we are able to sort the edges more easy, for example when we need the have the edge pairs with small angle differences, like we will need for the experiments (more about this later).

For debugging purposes it is also possible to plot a graphical representation of the graph onto the original image. An example of this is shown in Figure 13. We see here that the skeleton fits the veins by visual inspection, but there are some edges that might be due to noise. The resulting graphs will be saved and will be the input for the third part of our implementation.

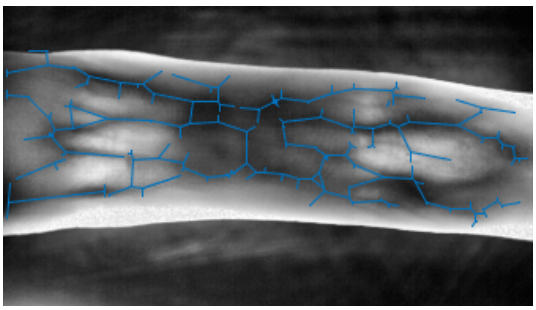

Fig. 13. An example of a graph displayed on the input image (after AHE to make the veins more visible).

.

C. From graphs to a similarity score

The third and last part of implementation is to compare two graphs and find a similarity score. A schematic overview is given in Figure 7. When we want to do a comparison, we take two graphs and send them into the graph matching code, which calculates the similarity score (see Equation 3) between the two graphs. The pseudo-code of this can be found in [34], Algorithm 1. This score is normalized between 0 and 1, where 0 is two graphs exactly matching, and 1 is not any similarity at all.

V. Results

Several experiments were performed to get an idea of how well the system will perform with different conditions. For this we will first introduce the dataset that was used, after which we will explain something about our experiments and give the results.

A. Dataset

There are several datasets available with finger vein images. In order to make our results as comparable as possible, we use the UTwente Database, containing finger vein images of 60 participants, with 4 pictures of 6 different fingers per participant. From those 4 images, there are 2 weeks between

two sets of 2 images. Previous results using this database were given by [13], and can also be found in Table II. Also the Peking V4 database, that was also used by [13] was considered, but when trying to create initial graphs, it showed the image quality was not sufficient to create useful graphs.

B. Overview of the experiments

We decided to use the maximum curvature method as this one gave the best results when looking at both with and without adaptive histogram equalization, and therefore we expect it to be the most stable option. When only looking at the results with adaptive histogram equalization, the principal curvature method shows slightly better results, and might be considered as well.

There are several ways in which we can sort the edge pairs in matching algorithm as described in [34]. The initial way is to find the Ni smallest dissimilarity scores between edge pairs and use those for rotation. However, as we can assume that the images are oriented in the same way, we can also only take a look at the smallest Ni angle differences.

Also the distance between two edges (so how far two edges are away from each other) in the distance-matching (where we find how many edges in the graphs are matching in total after rotation and translation) should be less than a certain value ε. With some experimental matching of the first few images we found that $\varepsilon = 10$ was a reasonable value to obtain good results. For matching two edges, we decided that the minimum length of a edge should be 10 pixels, to prevent random small pieces from matching.

For the experiments we made two sets of graphs. One set that was taken directly from the input pictures (normal graphs), and one set that was pre-aligned using rotation based on the finger contour (pre-aligned graphs). For this, the middle line of the finger was determined using the average of the top-line of the finger and the bottom-line of the finger. The idea is that if a rough pre-rotation based on the contour is good enough to give proper results, we can save calculation time on the rotation in the algorithm.

To calculate the equal error rates (EER), distance scores for N genuine matches and N impostor matches are calculated. Then we step by step increase the threshold value to determine the false acceptance rate (FAR) and false rejection rate (FRR). The EER is the point where the FRR and FAR are equal.

The first experiment is done to get a baseline result to which we can compare the results of our changes to the initial implementation. In our second experiment we orient all the edges in the same direction to examine the orientation-dependence of the algorithm. The third experiment removes all small edges from the graph to see if these small edges, that may be considered to be noise, will improve the performance. The fourth experiment studies the amount of edge pairs that need to be tried for alignment in order to get good results. Our fifth experiment investigates a the new balanced dissimilarity score d_{norm}. Experiment 5a attempts image blurring as noise reduction and Experiment 5b investigates different vein vessel network extraction methods to see whether this could lead to an improved performance. Experiment 6 investigates graph pruning as a more subtle way of edge removal. Experiment 7 investigates edge combining to see if reconstructing longer vessels again will improve the number of matching edge pairs. Experiment 8 investigates the influence of using the lengths as a weight for how important an edge match is.

C. Experiment 1: Initial implementation

The first experiment was done on 10% of the dataset to reduce long calculation times. This 10% is, in all experiments, selected using the images of the first 6 persons from the dataset. This gives us 36 unique fingers with 4 images each. With this we have 216 genuine matches. We randomly select 216 impostor matches to compare against. Extrapolating the time of the small experiment, matching all 1440 images with their corresponding images, which is 2160 genuine comparisons, and calculate 2160 impostor pairs would take well over 4 hours, given $Ni = 120$, which means we try to rotate and match the two graphs with the first 120 possible edge pair matches. The results of the first experiment can be found in Table III.

EER (%)	Normal graphs	pre-aligned graphs
Sorted by s_{ab}	10.2	10.2
sorted by $\Delta\theta$	4.63	8.33

TABLE III

RESULTS OF EXPERIMENT 1: 10% OF THE DATASET IS USED, OF WHICH BOTH PRE-ALIGNED AND NOT PRE-ALIGNED GRAPHS ARE MADE. THE BEST MATCHING EDGE PAIRS ARE FOUND BY SORTING ON EITHER DISTANCE OR ANGLE DIFFERENCE.

From this test it seems that sorting by angle rather than by distance between the edges is a promising option. However it should be noted that the sorting by angle is slower in generating results than sorting by distance. This might be due to the fact that most small distances are caused matches between small edges, which will not be calculated all the way through as they get filtered ([34], Algorithm 1, line 12, an additional length threshold is performed to prevent these small edges from matching) because they would generate unreliable results, while with angle differences that is not necessarily the case as long edges can also have a small angle difference if they're oriented in the same direction (most veins are oriented horizontally), which is likely with the fingers in our images.

The EERs are still very high in Experiment 1. When analyzing the data, it seemed that seemingly similar edges were rejected. When we look at the way the distance measure is constructed, it doesn't take the direction of the edge into account. As we traced from all points, the edge in one images might have been detected from left to right while in the other image it was detected right to left. This would lead to a non-match while it would be a proper match.

D. Experiment 2 - Orientation of the edges

Because of the directional dependency, we decided to orient all edges in such a way that the smallest X-value is on the begin-coordinate and the largest X-value on the end-coordinate. In this way, all edges would be oriented from left to right. And as the veins are generally oriented in a horizontal way, we do not have to add a sorting in the Y-direction for these rare cases where there might be a small vein with two same X-value for both the begin-coordinate and end-coordinate of the edge. With these oriented edges we performed our second experiment. The rest of the system is unchanged compared to our first experiment. The results can be seen in Table IV.

EER (%)	Normal graphs	pre-aligned graphs
Sorted by s_{ab}	4.63	4.17
sorted by $\Delta\theta$	1.85	5.09

TABLE IV

RESULTS OF EXPERIMENT 2: THE CONDITIONS ARE THE SAME AS IN EXPERIMENT 1, ONLY THE ORIENTATION OF THE EDGE IS CORRECTED SUCH THAT THE SMALLEST X-VALUE IS ALWAYS IN THE BEGIN-COORDINATE.

With this change there was an instant improvement of all EERs. It is interesting to see that the EERs are nearly equal for pre-aligned and non pre-aligned graphs (1 match more or less is a 0.46% difference in EER given the 10% of the dataset) when we sort by the edge distances, while when we look at the smallest angles for matching, the pre-alignment clearly has a negative effect on the EER.

When we analyzed the errors with the pre-rotation, it seemed that these images were not properly rotated due to deformation of the finger in one of the two images. An example of two images that give a mismatch is given in Figure 14.

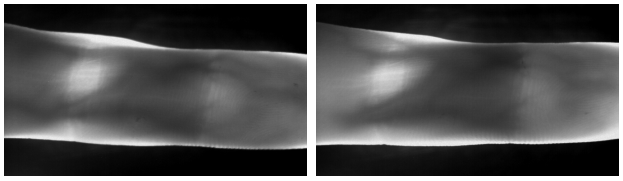

Fig. 14. An example of two images that give worse results due to pre-alignment.

One can see that the left image is rotated clockwise very slightly compared to the right image when you look at the structure of the veins. However when you look at the contour the rotation seems to be much larger, mostly due to the fact that in the bottom contour of the right image it goes slightly upwards, therefore giving a larger pre-rotation than needed when we want to do a good angle comparison. The fact that the pre-alignment gives worse results shows that the alignment that is used is not very robust. This is probably due to the dissimilarity score s_{ab}, which we will investigate further in Experiment 5.

E. Experiment 3 - Removing small edges

In our third experiment the aim was to find out whether our hypothesis that the small edges caused many errors was in fact true. We decided to see what would happen if we remove all edges from the graph that are smaller than 10 pixels, as these are not valid for matching either way for the distance measure. This would allow for more useful matches to be in the 120 matching edge pairs that we try for rotation. Also it would make the total number of edges smaller, which should make Equation 3 more robust, as there is less influence of a random number of small edges in our graphs. The results of this experiment can be found in Table V.

EER (%)	Normal graphs	pre-aligned graphs
Sorted by s_{ab}	6.02	5.56
sorted by $\Delta\theta$	1.85	3.24

TABLE V

RESULTS OF EXPERIMENT 3: THE CONDITIONS ARE THE SAME AS IN EXPERIMENT 2, ONLY THE SMALL EDGES ARE REMOVED FROM THE GRAPHS.

In case of the distance matching by s_{ab} this removing seems to have a negative effect on the matching results. However when we look at the results with the angle-sorting, results stayed the same or improved as expected. This suggests that in a few cases either the number of longer edges in the graphs is more random than we expected, and thus making the distance score more random rather than more reliable, or that we accidentally remove edges that we used to rotate our graphs properly, and we need to search further in our list of Ni edge matches for a good match. To see if the latter is indeed the case, we investigated the influence of the value Ni.

F. Experiment 4 - Influence of Ni

In our fourth experiment the influence of the choice of Ni is studied. As we do not know for sure whether our best rotation will be found when looking at the first 120 matching edge pairs (out of at least several thousands of edge pairs) for rotation. Ni was increased in 5-fold to 600 to see whether the results would improve. The results of this experiment can be found in Table VI.

EER (%)	Normal graphs	pre-aligned graphs
Sorted by s_{ab}	1.39	2.78
sorted by $\Delta\theta$	0.93	1.85

TABLE VI

RESULTS OF EXPERIMENT 4: THE CONDITIONS ARE THE SAME AS IN EXPERIMENT 3, ONLY NOW WITH $Ni = 600$ INSTEAD OF $Ni = 120$.

As we can see, the increase of Ni has a great effect on the results, as all EER's are decreasing. On one hand that tells us that we may need to increase our Ni value further to get better

results. On the other hand it also tells us that both sorting methods are still not ideal, as the best matching rotated graphs don't come from the edge pairs that are necessarily on the top of the list. The threshold value when using the angle difference an non-rotated graphs is found to be 0.786.

To get some information about reproducibility of the EERs, the 0,93% EER of Experiment 4 was validated by running the experiment 5 more times. This gave EERs of 1.39%, 0.93%, 0.93%, 1.39% and 0.93%. As 0.46% corresponds to 1 image more or less matched, we can conclude that the results are reproducible within a small error-range (0.46%, 1 image). To get an idea of the distribution of distance values we plotted them against their number to get a useful density plot. This plot can be seen in Figure 15.

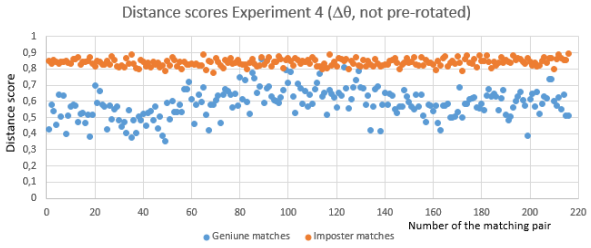

Fig. 15. The density plot of the distance scores. It can be clearly seen that the band in which the impostor match distance scores are is quite narrow, while the values of the genuine matches are quite spread.

As can be seen, the density of impostor matches is very high in the band with a distance score of 0.8-0.9 (0 being an absolute match, 1 being no match at all). On the other hand we can see that the band of the genuine matches is much wider and spans a much wider range of 0.4-0.8. This can be explained by the fact that when you try to match two images that are not exactly the same, there will always be some disturbances and veins that are detected in one image but not in the other. What we can see however is that in general the quality of the images, and with that the graphs, can differ much, which is an explanation for the wider band of genuine matches. From this result we can conclude that the impostor matches are usually well-defined such that the scores are very similar. So if two images are not a match, we are very well able to detect that they are not a match. However when it comes to matching two genuine graphs, it seems to be more difficult. Therefore in order to improve the performance of the system it is suggested to improve the matching for the genuine matches and try to find out why certain graphs have more difficulty matching than others.

In order to see how the system would change when using a larger piece of the dataset, we also did an experiment where we increased the part of the dataset we use from 144 images to 600 images. As this drastically increases the calculation times, it was decided to only calculate the value for the situation that already gives us the lowest EER, so using the angle difference for sorting and the normal non pre-rotated graphs. This gave us an EER of 3.67%, which is significantly higher than the initial equal error rate of 0.93% on 10% of the dataset. The corresponding threshold value is 0.795.

While our initial testing, given the conditions as used in Experiment 4 ($Ni = 600$, small edges removed, sorted edges and 10% of the dataset used) showed error rates around 0.93%, which is in edge with the results produced by the methods tested by [13], using a larger part of the dataset has a drastic effect on the performance, with an EER around 3,67%. As shown in the density plot of Figure 15, all impostor scores are well-defined in a small bandwidth. With this we can assume that in the larger part of the dataset, there are probably more low-quality graphs that are categorized as impostors. A further increase of Ni was done to see where the optimum would be, but this did not make a difference for the EER, and therefore was not investigated further.

G. Experiment 5 - Changing the dissimilarity score

As said before, the number of edge pairs used for the alignment of the graphs had to be increased to $Ni = 600$ to obtain good results. As we want to find a good match as early as possible, we decided to have a look at the way the dissimilarity score between two edges is defined for sorting before the rotation. We replace the original dissimilarity score s_{ab} as given in Equation 2 by the new balanced and normalized dissimilarity score d_{norm} as shown in Equation 5 as explained in the previous section. This gave the following results (given $Ni = 600$):

EER (%)	Using first 144 images of the dataset	using first 600 images of the dataset
Sorted by $\Delta\theta$	0.93	3.67
sorted by d_{norm}	1.39	3.89

TABLE VII

RESULTS WHEN COMPARING THE OLD DISTANCE MEASURE s_{ab} FOR SORTING EDGES COMPARED TO THE NEW DISTANCE MEASURE d_{norm} FOR BOTH A SMALL AND LARGER PART OF THE DATASET.

The results, as can be seen, are quite similar. Again, the 0.46% difference in EER at 10% of the dataset comes down to one more match or mismatch, which is most likely due to the random selection of impostors. The 0.22% difference in EER at the larger part of the dataset comes down to two matches difference. These are again very small deviations in EER (1 image in 10% of the dataset or 2 images in 40% of the dataset), which leads to believe that the new dissimilarity score, even though results look slightly worse at first sight, be a good idea.

To see if the new dissimilarity score indeed helps the sorting, we investigate a decrease of Ni to $Ni = 120$ again. The EER given 10% of the dataset was again 1.39%. This leads to the conclusion the new dissimilarity score d_{norm} indeed sorts better than the old dissimilarity score s_{ab} and gets the right edge pairs for rotation sorted in the first 120 edge pairs. Therefore we decided to keep $Ni = 120$ with the new dissimilarity score.

H. Experiment 5a - Blurring images

The graphs that were created show noisy results as can be seen in Figure 6. Therefore we decided to try to reduce the noise by pre-filtering. We applied a Gaussian blur to the AHE image in order to see if it helps to remove the small noisy bits of the graphs. When doing an initial test with a Gaussian blur ($\sigma = 25$), error rates went up from 0.93% to 4.17%. An example image of a blurred image with the found graph is shown in Figure 16.

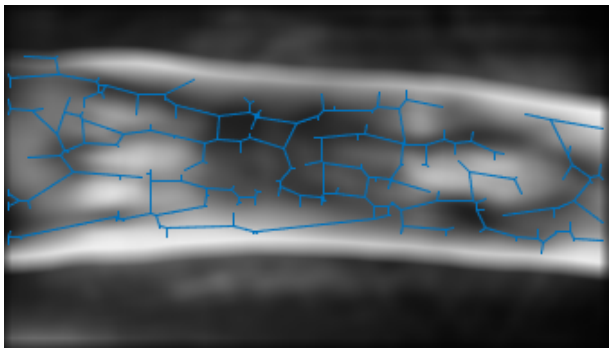

Fig. 16. An example of a graph created using a Gaussian blurred image, displayed on the blurred image.

The image is the same as used in Figure 13. The graph created with the blurred image as input clearly gives a less accurate graph of the veins and still contains many noisy small edges. As this initial test shows that blurring does not improve our graphs, we decided to focus on other possible improvements.

I. Experiment 5b - Different vein vessel network extraction methods

Besides blurring our image to reduce the amount of noise in our images, we also wanted to answer part of our second research question: What is the influence of the vein vessel network extraction method. We decided to take a look at the different methods as explored by [13]. An example image of the different methods is given in Figure 17.

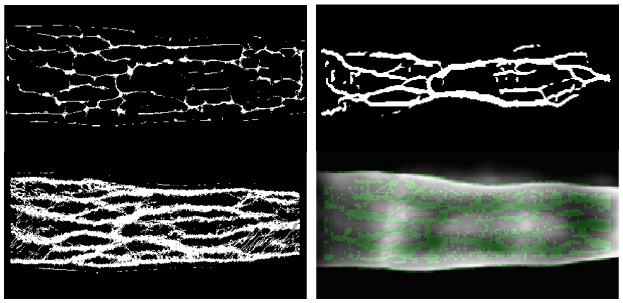

Fig. 17. An example of four different vein vessel network extraction methods. From left to right, top to bottom: Maximum curvature method, principal curvature method, repeated line tracking, wide line detector.

We decided to create some skeletons with the principal curvature method as well to see what the difference in resulting graphs is. To demonstrate this, Figure 18 shows the found skeleton with both methods.

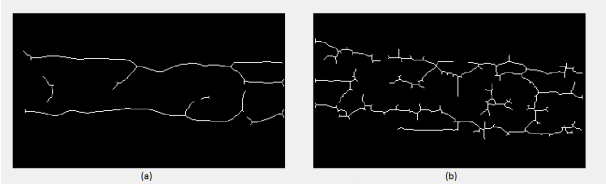

Fig. 18. Two skeleton images of different vein extraction methods. (a) shows the skeleton as extracted by the principal curvature method, (b) shows the skeleton as extracted by the maximum curvature method. There is a clear difference in noise levels visible in the skeleton images.

When we look at the different methods, we see that the principal curvature method (right top) [13], [33] gives nice and clear veins, but with very little detail when compared to the other three methods. The repeated line tracking method (left bottom) seems to give a more detailed overview of the veins, but is filled with lots of noise that connects veins that should not be connected. The wide line detector (right bottom) also shows a noisy result that is not that useful when trying to create a graph. This means that the only other candidate besides the maximum curvature method (left top), which gives a proper result in terms of detecting veins, would be the principal curvature method. However when we look at Figure 18, the lack of detail in this vein vessel network would lead to very simplistic graphs with a very limited number of edges. This in turn would lead to poor matching results as there are just very few edges that may randomly be interrupted by a small side-branch, and matching one edge more or less in the matching will have a larger influence then when this would happen in the maximum curvature method. Therefore we decided to stick with the maximum curvature method, even though this method's finger vein vessel networks are not perfect either.

J. Experiment 6 - Graph pruning

Another possible improvement is the pruning of the graphs. This is a method to remove small edges, where we remove all edges that have an end point, and are smaller than N pixels. In our experiment we have chosen for $N = 10$, as this will remove small noise edges, but keep the larger edges that are probably genuine veins. It may be noted that this is similar to Experiment 3, where we removed all edges smaller than 10 pixels. However this time it is only the small edges with at least one endpoint. The disadvantage of this method is that it does not remove small edges between branch points (for example two branch points that are most likely 1 branchpoint in reality), but it may solve the problem we had before, where the number of edges kept became more random.

With the 10% of the dataset we get an EER of 0.93%, which is the same as we had before, but now with $Ni = 120$. However on the larger part of the dataset (600 images in stead of the 144 images), error rates went down from 3.67% to 2.89%. This is 7 out of 900 genuine image pairs that are properly classified. This is more than the 1 or 2 images difference that were the deviations in Experiment 5, and therefore this is a significant improvement. This is

probably because the pruning decreases the opportunity for small edges to make random matches that are treated equally important to real matches of longer edges, yet it does not influence the total number of edges so much.

An interesting thing to note is the change in threshold value at which the EER is found. When increasing the size of the dataset, the threshold value to determine whether it's a genuine match went from 0.754 to 0.768. This could indicate that the quality of the first 144 images is higher than of the following 456 images, as the threshold moves closer to the impostor scores.

K. Experiment 7 - Edge combining

This experiment consists of combining edges as described in the previous section. We combine the edges, but since we are not sure whether the uncombined or combined edges are better for matching, we decided to keep both in the same graph. An example of this can be seen in Figure 19. To make sure this adding of edges does not influence the results by just having a larger number of edges to divide by in Equation 3, we decided to not increase a_{max} and b_{max}, which is the number of edges in the graph.

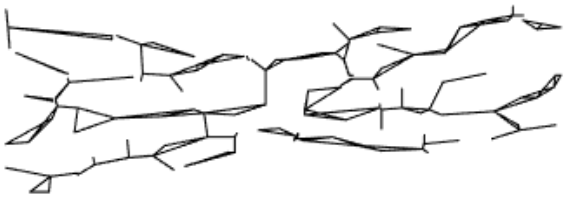

Fig. 19. An example of a graph where combined edges are added to improve the performance.

EERs were 2.78% at the 10% of the database, or 3.56% when taking the first 600 images, given $Ni = 120$. Investigating the results has shown that in genuine matches only one or two extra matching edge pairs were found in the examined cases. When we look at the threshold values, they were respectively 0.787 and 0.791. Compared to the threshold values of 0.786 and 0.795 respectively in Experiment 4, the threshold values don't really change, while it is expected to have a lower threshold value if more edges on average were matched (see Equation 3). This shows that combining edges is not an effective way to improve the performance of our system.

L. Experiment 8 - Adaptation in distance scores

Finally an experiment was performed with the new similarity score using the edge length as a weight factor in the matching process. In this, besides the normalized version as described in Equations 7-8, also a non-normalized version was made, where no division by the average total length of both graphs was performed. In the not normalized condition the EER was 5.09%, while in the normalized condition the EER was 3.70%, given that we used 10% of the database and $Ni = 120$. This proves that the normalization is working, however the results themselves are not as well as in some of the previous experiments. Apparently edges that are matched in genuine comparisons are roughly the same length on average as the matched edges in impostor comparisons. This still leaves an option to investigate using the vein vessel width as a weight for the matching.

M. Results overview & comparison to state of the art work

As we now have results, we want to compare them to the state of the art work. In Table II an overview of EERs (on the same database) is given as found by [13]. Our results are briefly summarized in Table VIII.

EER (%)	without pre-rotation		with pre-rotation	
Experiment	s_{ab}	$\Delta\theta$	s_{ab}	$\Delta\theta$
Experiment 1	10.2	4.63	10.2	8.33
Experiment 2	4.63	1.85	4.17	5.09
Experiment 3	6.02	1.85	5.56	3.24
Experiment 4	1.39	0.93	2.78	1.85
EER (%)	10% of the dataset		40% the dataset	
Experiment	$\Delta\theta$	d_{norm}	$\Delta\theta$	d_{norm}
Experiment 5	0.93	1.39	3.67	3.89
Experiment 6		0.93		2.89
Experiment 7		2.78		3.56
Experiment 8		3.70		

TABLE VIII
OVERVIEW OF RESULTS FROM THE EXPERIMENTS. A SHORT EXPLANATION OF THE EXPERIMENTS CAN BE FOUND IN TABLE IX.

Experiment	What was done
1	initial implementation
2	sorted edges (smallest X-value being the first coordinate)
3	removing all small edges
4	increasing the value of Ni from 120 to 600
5	changing the dissimilarity score measure s_{ab} to d_{norm}
6	graph pruning
7	combining edges
8	using edge lengths as a measure for the distance scores

TABLE IX
OVERVIEW OF THE EXPERIMENTS.

As can be seen, our EERs on a 10% of the dataset are still in the range of the results found by [13] when we made some changes to our implementation. However when we started analyzing a larger part of the dataset, EERs went up to at least 2.89%, which is not very good when compared to the state of the art, where on the entire dataset EERs are obtained below 1% (see Table I and II). This leads us to conclude that the biometric graph matching method as it is now developed and adapted to the finger veins, is not a suitable solution for finger vein authentication when compared to some of the other methods developed before.

In the future our system might be improved by taking the width of the veins into account as a measure of how important a vein is, and different ways of extraction the vein vessel networks may be explored.

VI. Discussion & Conclusions

Our goal was to find out how well the biometric graph matching (BGM) method would perform on finger veins compared to the state of the art work. This gave us three research questions:

1) Is the biometric graph matching method as described by [34] a suitable method for comparing finger vascular patterns, given that we use the best performing existing vein vessel network extraction method given in [13]?
2) How can we improve the performance of our implementation of the biometric graph matching method?
3) How can we make the biometric graph matching method more specific to finger vein authentication in order to improve the recognition performance?

Answering our first question, implementing the biometric graph matching system as described by [20], [34] was performed as described in Section IV. The EERs were up to 10% on a 10% subset of the dataset, and therefore the performance is worse than the 0.37% EER given in literature before (see Table II) for finger vein images, of the 0.5% EER for retina vein images using the BGM method [34]. Also a small implementation change to assure the orientation of the veins is the same did decrease the EERs down to 1.85%, but this is still not near the performance of state of the art work. This leads us to conclude that the biometric graph matching method in its original form is not well-suited for comparing finger vascular patterns.

To answer the question how we could improve our basic method, a number of options were explored. Removing all small edges gave varying results, but in case of sorting the edge pairs for rotation by the angle, it did give an improvement. However when sorting by s_{ab} the performance went down.

When we wanted to optimize our parameters, the most relevant choice was to change amount of edges pairs that are attempted for alignment to see if the best graph rotation and matching would happen within those first 120 possible edge pairs. Therefore we increased this value to an experimental value of 600, which decreased the EERs down to around 1%. Increasing the value further did not enhance the performance.

Besides this, we also took a look at a new dissimilarity score as found in Equation 5. This gave similar results as before, so it was not a big improvement, though matching edge pairs were generally found earlier so we could go back to 120 matching edge pairs, saving computational time. As we tune these parameters manually on the dataset we use, it might give somewhat rosy results.

Other vein vessel network extraction methods were explored such as the principal curvature method. This method showed vein vessel networks that were less noisy, but also had less detail, making the graphs too generic to compare. The repeated line tracking method and wide line detector both showed detailed finger vein vessel network images, but were more noisy than the maximum curvature method, and were therefore not further investigated.

Simplifying graphs by pruning small end-edges did make an improvement in EERs from 3.67% to 2.89%. This is probably due to the fact that the number of edges in the graph a_{max} and b_{max} got smaller, making the matching more about the genuine veins rather than the noisy bits. This leads to the conclusion that in order to improve the performance of our implementation, increasing the number of attempts for alignment could help. Our new dissimilarity score helps to find the proper edge pairs for alignment earlier. Pruning our graphs also helps to obtain better results, as this removes some of the noise.

Our third research question was how we can improve the system performance by making the method more specific for finger vein authentication. We attempted to combine smaller edge pieces into bigger edges, but this had a negative effect on the error rates. When analyzing several images, it was found that only one or two additional edges were matched. As the threshold values were staying the same, this leads to think that on average there are no extra edges matched, as more edges matching would mean a smaller distance between graphs, and therefore a smaller threshold value.

Including the edge length as a weight factor for the graph matching is one other thing we tried. Longer edges would get more important than smaller edges in this way. So rather than counting the number of matching edges, one would match on the total length of matched edges. With the initial experiment, EER's went up to 3.70% when normalized to the length and 5.09% in the unnormalized case. Even though normalizing for the total average length of the edges in the graphs is a good idea, it does not get the EERs down to a point where it is an improvement. This is probably because edges that are matched in genuine comparisons have roughly the same length on average as the matched edges in impostor comparisons, so our initial assumption that in genuine matches the matched edges are longer than in the randomized case is rejected. This leads to the conclusion that our attempts to make the authentication more specific to finger veins did not work as intended.

In general, our conclusion is that the system is not suitable for finger vein graph matching due to the extremely noisy vein patterns when compared to retina or hand vein patterns. Compared to the state of the art work our system doesn't perform well enough.

VII. Future Work

As our system does not perform well enough due to the noisy nature of the graphs we get, the first thing that would need improvement is the graph extraction process. One could look into other vein vessel network extraction methods, or try to optimize one of the existing methods in order to get more detail and less noise.

Besides this it would be interesting to investigate a weight-factor for the matched edges based on the width of the veins in the extracted vein vessel network image. In this way, big veins would become more important than smaller veins (that could also be noise). In this way the system may become more robust against noise.

ACKNOWLEDGMENT

The author would like to thank his supervisors, dr. M. Poel, prof.dr.ir. R.N.J. Veldhuis and dr.ir. L.J. Spreeuwers for their guidance and advise during this Master thesis project.

REFERENCES

[1] J. Yang et al, *Finger-vein network enhancement and segmentation*, Pattern Analysis and Applications (2014) 17:783-797

[2] H.C. Lee et al, *Finger vein recognition using weighted local binary pattern code based on a support vector machine*, Journal of Zhejiang University-SCIENCE C, (2010) 11(7) 514-524

[3] H. Zhang et al, *Finger Vein Recognition Based on Gabor Filter*, Intelligence Science and Big Data Engineering (Lecture Notes), (2013) pp. 827-834

[4] J. Peng et al, *An Effective Proprocessing Method for Finger Vein Recognition*, Fifth International Conference on Digital Image Processing, Proceedings of SPIE (2013), Vol. 8878, 887808

[5] R. Xiao et al, *A Novel Matching Strategy for Finger Vein Recognition*, Intelligence Science and Big Data Engineering (Lecture Notes), (2012) pp. 364-371

[6] D. Wu et al, *NSCT-based Fusion Enhancement for Multispectral Finger-vein Images*, Sixth International Conference on Digital Image Processing, Proceedings of SPIE (2014), Vol. 9159, 91590U-1

[7] S. Pang et al, *Rotation Invariant Finger Vein Recognition*, Biometric Recognition (Lecture Notes), (2012) pp. 151-156

[8] M. Kono et al, *Near-infrared finger vein patterns for personal identification*, Applied Optics, (2002) Vol. 41, No. 35, pp. 7429-7436

[9] E.C. Lee et al, *Finger Vein Recognition Using Minutia-Based Alignment and Local Binary Pattern-Based Feature Extraction*, International Journal of Imaging Systems and Technology, (2009) 19(3) 179-186

[10] B. Huang et al, *Finger-Vein Authentication Based on Wide Line Detector and Pattern Normalization*, International Conference on Pattern Recognition, (2012) pp. 1269-1272

[11] F. Hillerström et al, *Generating and Analyzing Synthetic Finger Vein Images*, International Conference of the Biometrics Special Interest Group (BIOSIG), (2014) pp. 111-120

[12] B. Ton et al, *A High Quality Finger Vascular Pattern Dataset Collected Using a Custom Designed Capturing Device*, International Conference on Biometrics Compendium, (2013) pp. 1-5

[13] B. Ton, *Vascular pattern of the finger: Biometric of the future?*, (2012)

[14] Hitachi Ltd., *How Finger Vein Works: Finger Vein Authentication Technology*, http://www.hitachi.eu/veinid/howfvworks.html, (2009)

[15] *The Hong Kong Polytechnic University Finger Image Database (Version 1.0)*, http://www4.comp.polyu.edu.hk/ csajaykr/fvdatabase.htm

[16] A. Kumar et al, *Human Identification using Finger Images*, IEEE Transactions on Image Processing (2012) Vol. 21 pp. 2228-2244

[17] Y. Lu et al, *An available database for the research of finger vein recognition* , 6th International Congress on Image and Signal Processing (CISP), (2013) Vol. 1 pp. 410-415

[18] International Commission of Illumination, *Koschmieder's Law*, http://eilv.cie.co.at/term/629

[19] X. Wen et al, *Research on Enhancing Human Finger Vein Pattern Characteristics*, Asia-Pacific Conference on Power Electronics and Design (APED), (2010) pp. 97-100

[20] S.M. Lajevardi et al, *Hand vein authentication using biometric graph matching*, Biometrics IET, (2014) Vol. 3, Iss. 4, pp. 302–313

[21] N. Poh et al, *Biometrics Evaluation and Testing, D3.6: Description of Biometric Databases and Protocols*, https://www.beat-eu.org/project/deliverables-public/d3.6-description-of-biometric-databases-and-protocols (2012)

[22] J. Yang et al, *Finger-Vein Recognition Based on Gabor Features*, ed. by Z. Riaz. InTech, (2011), Chap. 2, pp. 17-32, http://www.intechopen.com/books/show/title/biometric-systems-design-and-applications.

[23] E.C. Lee et al, *Image restoration of skin scattering and optical blurring for finger vein recognition*, Optics and Lasers in Engineering 49.7, (2011), pp. 816-828

[24] B.J. Kang et al, *Multimodal biometric method based on vein and geometry of a single finger*, IET Computer Vision 4.3 (2010), pp. 209-217

[25] X. Yu et al, *A Novel Finger Vein Pattern Extraction Approach for Near-Infrared Image"*, 2nd International Congress on Image and Signal Processing, (2009), pp. 1-5

[26] D. Wang et al, *User identication based on finger-vein patterns for consumer electronics devices*, IEEE Transactions on Consumer Electronics 56.2, (2010), pp. 799-804

[27] K.J. Wang et al, *Finger vein recognition by improved filtering and correction of Hausdorff distance*, Journal of Computer-Aided Design & Computer Graphics 23, (2011), pp. 385–391

[28] E.C. Lee et al, *New finger biometric method using near infrared imaging*, Sensors (Basel) 11, (2011), pp. 2319–2333

[29] N. Miura et al, *Feature extraction of finger vein patterns based on iterative line tracking and its application to personal identication*, Syst. Comput. Japan 35, (2004), pp. 61-71

[30] N. Miura et al, *Extraction of finger-vein patterns using maximum curvature points in image proles*, IAPR conference on machine vision applications, (2005), pp. 347-350

[31] Z. Zhang et al, *Multiscale Feature Extraction of Finger-Vein Patterns Based on Curvelets and Local Interconnection Structure Neural Network*, The 18th International Conference on Pattern Recognition Vol. 4., (2006), pp. 145-148

[32] E.C. Lee et al, *Restoration method of skin scattering blurred vein image for finger vein recognition*, Electronics Letters 45.21, (2009), pp. 1074-1076

[33] J.H. Choi et al, *Finger vein extraction using gradient normalization and principal curvature*, Proc. SPIE 7251 - Image Processing: Machine Vision Applications II, (2009), p. 725111

[34] S.M. Lajevardi et al, *Retina Verification System Based on Biometric Graph Matching*, IEEE Transactions on Image Processing Vol. 22, No. 9, (2013), pp. 3625-3635

[35] K. Riesen et al, *Approximate graph edit distance computation by means of bipartite graph matching*, Image Vision Computations, vol. 27, no. 7, (2009), pp. 950–959