A General Approach to Off-line Signature Verification

BENCE KOVARI, ISTVAN ALBERT, HASSAN CHARAF
Department of Automation and Applied Informatics
Budapest University of Technology and Economics
1111. Budapest, Goldman Gyorgy ter 3.
HUNGARY
beny@aut.bme.hu http://www.aut.bme.hu/signature

Abstract: - Although automatic off-line signature verification has been extensively studied in the last three decades, there are still a huge number of open questions and even the best systems are still struggling to get better error rates than 5%. This paper targets some of the weak spots of the research area, which are comparability, measurability and interoperability of different signature verification systems. After delivering an overview of some of the main research directions, this paper proposes a generic representation of signature verifiers. In the first part of the paper it is shown how existing verification systems compare to the generic model, detailing the differences and their resolutions. In the second part a signature verification framework is introduced, which was created based on the generic model. It is demonstrated how existing algorithms and even an existing signature verifier can be modularized and modified to allow an execution through the framework. Finally the benefits of the model are outlined including the unified benchmarking, comparability of different systems and the support for distributed software architectures like SOA.

Key-Words: - signature verification; off-line; unified model; component based; loose coupling

1 Introduction

Signature recognition is probably the oldest biometrical identification method, with a high legal acceptance. Even if handwritten signature verification has been extensively studied in the past decades, and even with the best methodologies functioning at high accuracy rates, there are a lot of open questions. The most accurate systems almost always take advantage of dynamic features like acceleration, velocity and the difference between up and down strokes. This class of solutions is called on-line signature verification. However in the most common real-world scenarios, this information is not available, because it requires the observation and recording off the signing process. This is the main reason, why static signature analysis is still in focus of many researchers. Off-line methods do not require special acquisition hardware, just a pen and a paper, they are therefore less invasive and more user friendly. In the past decade a bunch of solutions has been introduced, to overcome the limitations of off-line signature verification and to compensate for the loss of accuracy. Most of these methods have one in common: they deliver acceptable results (error rates around 5-10%) but they have problems improving them. This paper presents a solution to address the problem of improvement and thereby possibly break the 5% barrier. First an overview is given of different architectures used for the purpose of signature verification. It is shown that to isolate the weaknesses of the different approaches a comparison of the verification methods would be necessary. However this is almost impossible, due to the architectural differences between signature verification systems. By combing the beneficial aspects of the different systems a new, component based architecture is presented allowing a better comparison and partial benchmarking of the employed algorithms. Finally, some applications for the model are demonstrated.

2 Architecture of Signature Verifiers

Since the first survey paper [1] (dating back to 1989) several surveys like [2] [3] [4] and journal special issues like [5] have been dedicated to the comparison and evaluation of signature verification methods. While trying to give a balanced overview of the field they all face the same problem. Namely that the evaluation of signature verification algorithms, as for many pattern recognition problems, raises several difficulties, making any objective comparison between different methods rather delicate and in many cases impossible [3]. It is behind the scope of this paper, to discuss the technical details and flaws of the unique methods.

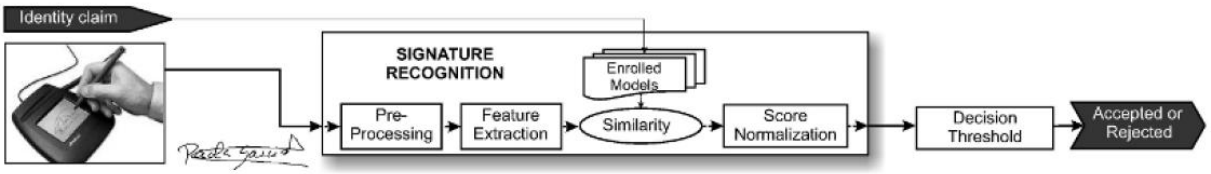

Fig.1. A typical off-line verifier with a simple threshold classifier (source: [7])

But as an alternative, the architecture of some main verification systems is examined in the following subsections.

The majority of signature verification methods can be divided into five main phases: acquisition, preprocessing and feature extraction, processing and classification (although these steps are not always separable). In the off-line case data acquisition means simply the scanning of a signature. This is followed by preprocessing whereby the images of signatures are altered (cropped, stretched, resized, normalized etc.) to create a suitable input for the next phase. The next step is feature extraction, the process of identifying characteristics, which are inherent to the particular person. The processing phase is mainly based on a single comparison algorithm, which is able to calculate the distance function between signature pairs. Using these results, the classification phase is able to make a decision, whether to accept or reject the tested signature. This coarse separation of processing phases is already an extension to [6] which does not separate feature extraction from processing and classification. In our model even further extensions will be necessary to allow a better control of the dataflow. In the following subsections these 5 steps will be explained in detail and matched to the steps of several other signature verifiers ([7] [8] [9] [10] [11] [12] [13]).

2.1 Acquisition

In general, the acquisition step converts a number of paper sheets to a set of digital images, each of them containing one or more signatures. It is essential to note, that scanning paper sheets with written signatures is not the only way to acquire the digital images. As noted in [14] samples can also be generated from on-line databases or by altering

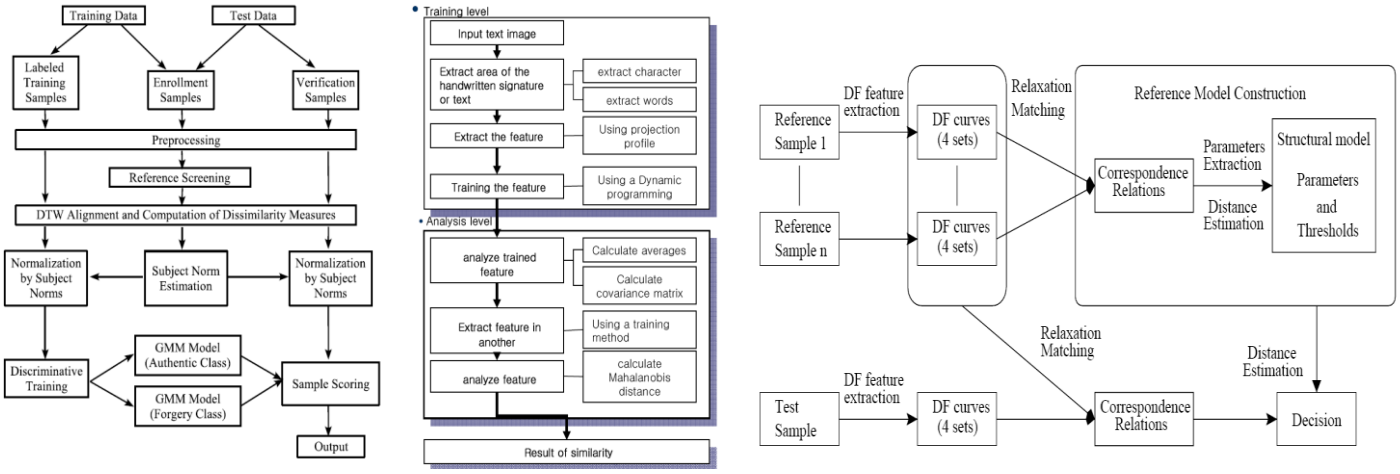

Fig. 2. Despite of the different employes algorithms and the various grouping of functions, the basic structure of a signature verifier is the same (sources: [13] left [12] middle [11] right)

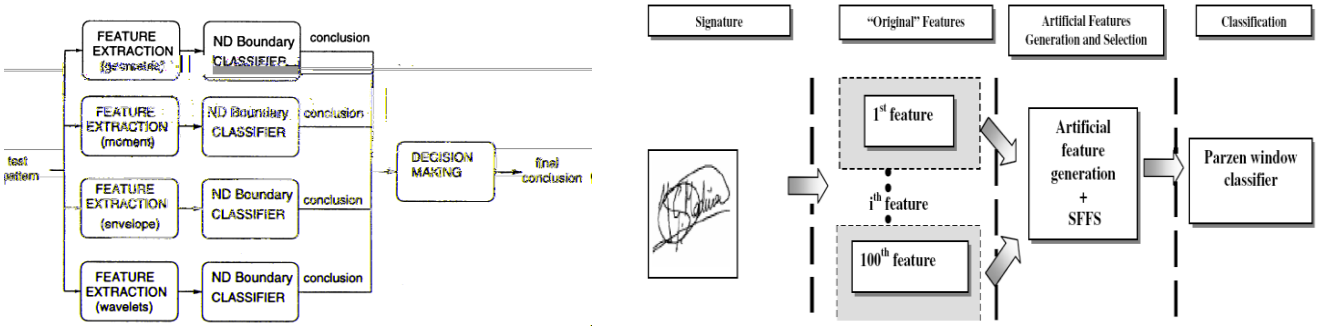

Fig. 3. Multiple features are used to allow a refined decision (sources: [8] left [9] right)

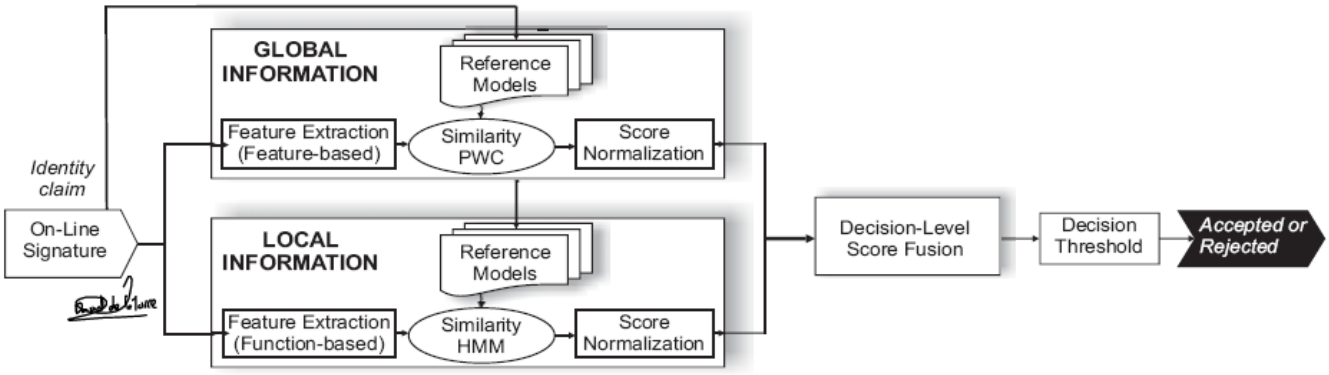

Fig. 4. Global statistics can help to interperet local information (source: [10])

existing signatures [15]. Of course these later methods cannot be used to validate the whole signature verification system; however they usually contain valuable additional information (like the correct order, direction and position of strokes) which can be used to benchmark separate parts of the system. In most of the observed systems the digitalized images of the signatures are assumed to be already present, therefore the acquisition phase is usually not a part of the system diagrams (Fig. 1-4) though they are always explained in the corresponding papers.

2.2 Preprocessing

The preprocessing phase is a sequence of image transformations creating the best possible input for the feature extraction algorithms. In on-line signature verifiers the acquired data is usually already in an optimal form for further processing, therefore this phase is superfluous [16] [17] (see also Fig. 4). In the off-line preprocessing it is usually necessary to eliminate the noise introduced during the acquisition phase. Some of the preprocessing steps, like noise filtering, rotation normalization, position normalization induce minimal information loss, while others, like binarization, morphological closing or size normalization can cause the loss of valuable information. Thus the second class of preprocessing steps is only applied where the feature extraction algorithm can gain direct advantages from them. Although preprocessing is used in all examined off-line verifiers, only three of them ([7] [12] [13]) display it on their system diagram.

2.3 Feature extraction

Feature extraction is with great certainty the most ambiguous processing phase. First let us make some definitions clear. "An image feature is a distinguishing primitive characteristic or attribute of an image. Some features are natural in the sense that such features are defined by the visual appearance of an image, while other, artificial features result from specific manipulations of an image […] Image features are of major importance in the isolation of regions of common property within an image (image segmentation) and subsequent identification or labeling of such regions (image classification)." [18]. Therefore feature extraction is the location and characterization of features, and generally it should not be confused with the later processing phases. Contrary to preprocessing which is defined sequence of transformation steps altering the original images, feature extraction is a set of (usually) independent functions returning a characteristic feature set for their input image. Several systems (Fig. 2) take advantage of multiple features to improve the quality of the input provided for distance calculations and classifiers.

Feature extraction is correctly isolated in Fig. 1 and Fig. 2, but the system representations shown in Fig. 3 are ambiguous.

2.4 Processing

The processing phase differs from the previous ones, in that it can (and has to) work with multiple images. First a matching is defined between the features of the different images, then a distance (or similarity) measure is calculated based on the corresponding features and finally this measure is normalized to make it a suitable input for a classifier. All of the three steps can be identified in Fig. 1 while the other systems do not mention all of them. This is only a side effect of the interpretation of phases, for example the feature extraction boxes usually also represent the processing phase (Fig. 3) and score normalization is sometimes considered to be part of the classification phase.

2.5 Classification

In the classification phase a single classifier is trained with a set of original signatures. Based on the training, the classifier can make decisions about the acceptance or rejection of a single test signature.

It should be noted, that several simplifications were used to get a uniform view of the different approaches. This "single classifier" can of course represent a composite system consisting of different local (Fig. 3) and global (Fig. 4) experts allowing the decision to be made with a deep understanding of the context. In some other cases there are classifiers used, to improve the feature extraction or distance calculation phase (Fig. 4). These should not be confused with the classification phase which in that scenario is a single threshold decision.

While it is uncommon in literature, the decision of the classifier is not limited to a binary decision. Beside the values "accept" and "reject" a third value "uncertain" is introduced in some works, usually combined with a confidence value.

3 Proposed model

Based on the observations of the previous section generalized model for off-line signature verifiers is proposed (See Fig. 5).

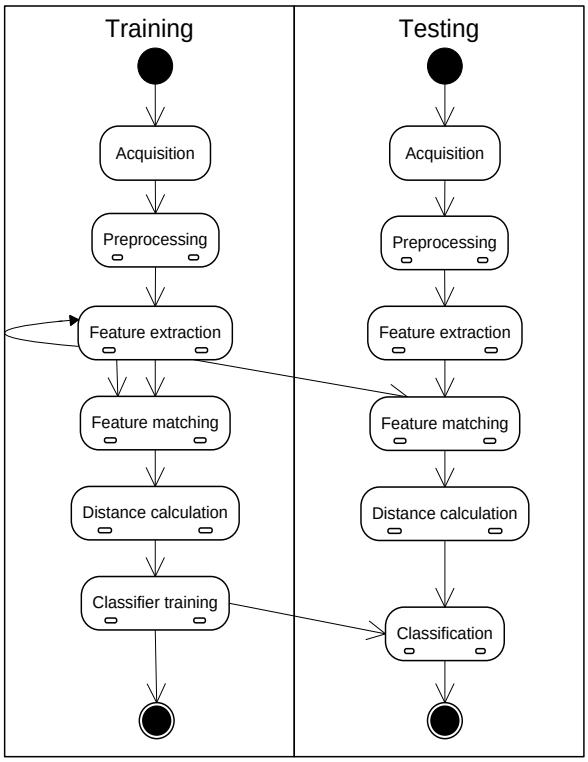

Fig. 5 Generalized view of an off-line signature verification system

The model combines the advantages of the previously introduced systems and their representations. Similarly to [11] and [13] it clearly isolates the path of the training set (original signatures) from the path of test signatures while traversing the same modules of the system. It incorporates the 5 phases of verification, where processing was further broken into feature matching and distance calculation steps to improve support for modularity. It is also interesting to note, that the two steps of the classifier were also partitioned: the classifier is trained during the training phase, only using the original signatures from the reference database, and classification decisions about test signatures will be made based on the training during the testing phase.

The model allows a direct top down data flow without ever referencing a previous module. This makes a loose coupling and individual testing of the components possible.

As noted in the previous section, individual phases can consist of multiple sequentially or parallel coupled subcomponents, therefore five of the six states are marked as composite states.

4 Model validation

To validate our model we created a signature verifier framework and implemented several different modules. Although some of the modules correspond to our main research line, many of them were chosen randomly, based on other signature verification methods. Our thesis is that all of these methods can be implemented by strictly using our pipelined architecture. The next subsections first describe the framework itself then some of the modules are explained in detail.

4.1 Signature verification framework

The framework itself is a direct implementation of the model presented in Fig. 5. It coordinates the training and testing phases and provides benchmarking capabilities. Individual modules interact with the framework through the implementation of well defined interfaces.

8 different interfaces are provided; each of them corresponds to one of the levels in Fig. 5.

4.1.1 Interfaces and architecture

By implementing the ITransformation (Fig 6. a.) interface, a module can provide one directional

image transformation capabilities, required by the preprocessing step of the model. Input and output are simple images.

Unlike the preprocessing phase, the feature extraction consists of two separate interfaces. This allows separating the process of extracting feature information from a single signature (IFeatureExtraction, Fig. 6.b.) and to identify signer specific aspects, for which all signatures are required. This is not limited to graphical features. Features can also include any kind of general measures extracted from an image, like [19] or [20]. This is called the "model" of a signer, modules extracting these kind of information shuld implement the IModelFeatureExtraction interface.

Similarly, when examining a signature, it can be tested both against a single original signature, and against the model of the signer. Another important aspect is the separation of the feature matching phase from the distance calculation. This means that corresponding features, like the same accent in different signatures has to be identified only once, thereby allowing the definition and calculation of several different similarity measures (distance functions) based on the same input. These can be performed both on signature and model level, which requires the introduction of 4 new interfaces (IFeatureMatching, IModelFeatureMatching, IComparison, IModelFeatureComparison).

The framework calculates all possible matches and distance measures. For example if we have 10 original signatures and one sample to examine, the framework creates 10*9=90 feature matches (every original signature is compared to the others) and 10 model matches (every original signature is compared to the model) in the training phase. In the testing phase the sample signature is compared with each of the originals (10 feature matches) and with the model of the signer (1 model match). Distance measures are calculated for all of these matches and used as an input for the classifier. The classifier is the first component which has different behaviors in the training and the testing phases (the functionality of the other components is the same in both scenarios). All the distance measures calculated in the testing phase are used as an a input for the IClassifier.Train() method. This allows the classifier to analyze the data and (for example calculate the weights corresponding to the different features). This step could be implemented by training a neural network or a hidden Markov model based classifier. After the training, the test results are feeded to the component by using the IClassifier.Test() method. The response should be a real number in the [0..1] interval, where 1 corresponds to a definitely valid, and 0 corresponds to a definitely forged signature. Values between the two boundaries represent the uncertainty of the response.

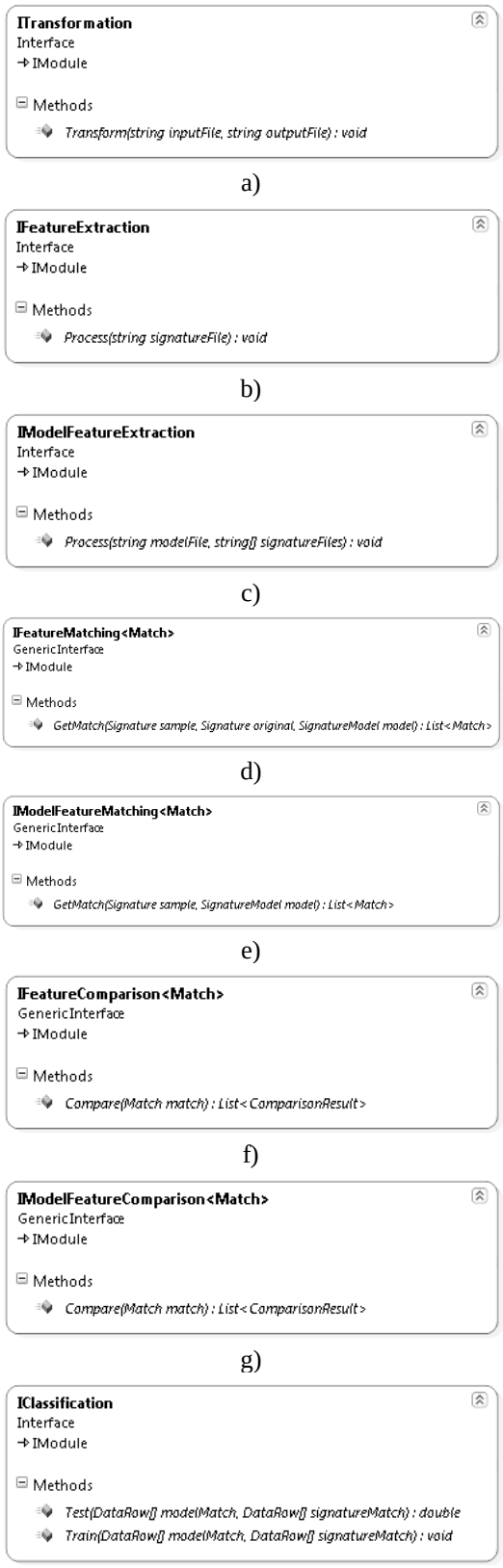

Fig. 6. Main interfaces in the signature verifier framework

4.1.2 Evaluation of the framework

One of the main goals of modern software design is to keep the cohesion (within a component) high and the coupling (between components) low [21]. As stated above, state of the art signature verifiers mostly lack these characteristics. Although they usually isolate some of the main modules (like preprocessing), the core is often a highly coupled composition of different components, often incorporating loops and other complex constructs in the information flow. Although this is not necessarily a design flaw, this makes the objective evaluation and the partial comparison of different systems hard and the simultaneous development of modules almost impossible.

Our model reduces coupling to a minimal level. Only the signature files and their correspondent metadata information are passed between components (later these are replaced by the matching and the distance information). The input and output parameters of the interface functions are restricted to the minimum necessary level without limiting the functionality of the modules.

The data flow is well defined and strictly follows Fig. 5. That means a straight flow, without any loops among the processing phases.

This structure allows the independent development and testing of the different modules. Although dependencies are possible (e.g. a distance calculation module may depend on a specified matching module) the components implementing the same interfaces are mainly interchangeable. In addition, several components (e.g. several distance calculation algorithms) can be used simultaneously to create a more sophisticated input for the classification module.

In next subsections examples are shown, how this model can be applied to aid the implementation and testing of various signature verification tasks.

4.1.3 A signature verification system based on stroke end directions

In [22] the relative direction and location of stroke ends are used to compare signatures. This example demonstrates how our framework can be applied to implement the system. There is no change in the acquisition phase. Signatures are scanned with 600dpi resolution, resulting in an average image size of 1000*250 pixels. Similarly to the resolution used in the original system.

Signature preprocessing is performed, by implementing the ITransformation interface. 4 different modules are created and used in the system in a pipeline.

1. Noise filtering: A noise filter is applied to remove the noise caused by the scanner.
2. Cropping: The image is cropped, to the bounding rectangle of the signature.
3. Binarization: Transformation from color to grayscale, and finally to black and white.
4. Thinning: Thinning the black and white image results always in a huge information loss. Therefore it is essential to select a thinning algorithm which gives a good abstraction of the original signature, with a low noise level. We selected an algorithm, which removes pixels so that an object without holes shrinks to a minimally connected stroke, and an object with holes shrinks to a connected ring halfway between each hole and the outer boundary [23].

Once the preprocessing has finished, feature extraction can begin. The IFeatureExtraction implementation performs the identification of the endpoints of strokes. Because of the thinning, these can be defined as pixels with one single neighbor connected. To help improve the robustness of the later processing algorithms, endpoints are also characterized by the direction of their corresponding strokes. Direction vectors are calculated from the first 10 stroke pixels next to the attached endpoint (Figure 7).

There are two side effects of the thinning algorithm which should be noted here.

- Rapid changes in pen angle often result in falsely detected endpoints. Although this reduces the quality of our “abstraction” but it does not reduce the quality of our matching algorithm because such changes in angle are as characteristic as real stroke endpoints.
- Connection points will always have three branches. This property only simplifies our mathematical model.

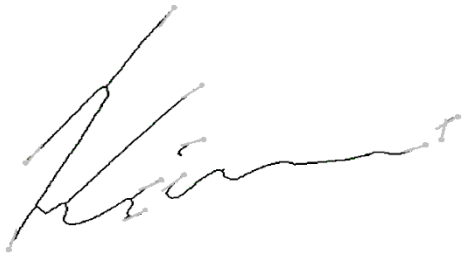

Fig. 7. Endpoints are represented by their location and direction

Next, an algorithm must be defined to identify the corresponding endpoints in two signatures. This is done through the implementation of the IFeatureMatching interface.

The simplest procedure for giving the correspondence is the Iterative Closest Point algorithm [24], which matches two clouds of points. It estimates the transformation (translation, rotation) between the coordinates of points on two images iteratively.

1. Initialize:
    a. Define P point set as P_0 and set k iteration variable to 0
2. Iteration:
    a. Find the new point set Q_k as the closest points on the image from the P_k
    b. Find the optimal rigid transformation T from P_k to Q_k
    c. Calculate $P_{k+1} = T(P_k)$
3. If the distance metric $d_k - d_{k+1} < \tau$, then terminate (τ is a predefined threshold value)

As an extension of the spatial relations methods, the correspondence of the features can be estimated using the properties above. It can be done through the following metrics:

- absolute distance from an appropriate reference point
- orientation of the above distance vector
- and the own properties of the features (at the endpoints this is an angle compared to horizontal axe, at junctions there are three angles compared to the horizontal direction, etc.)

The algorithm runs cyclically (like the Iterative Closest Point algorithm), it calculates the weighted distances of above metrics and changes the reference point in every loop. For the first iteration, the appropriate reference point can be simply the center of gravity (COG) in both images. In the next loops new reference points are selected, as the pair of points with the smallest distance.

Fig. 8. Correspondence between "a" letters

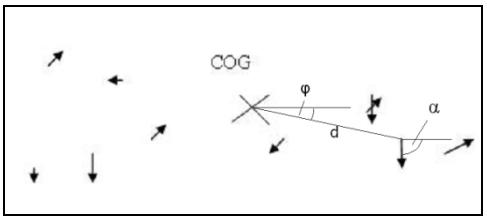

Fig. 9. Properties of the features (the arrows represent the position and orientation of endpoints in a signature).

A simplified version of the above algorithm can also be used in IFeatureComparison to calculate the similarity between two signatures e.g. by summing the distances (shown as d in Fig. 9.) of the 10 closest points.

To fully reconstruct the original scenario, the IClassification interface should be realized by a simple threshold classifier. The comparison and classification process is similar to the process described in [25].

At this point we were able to reconstruct the whole signature verifier only by implementing the appropriate modules in our verification framework.

4.1.4 Stroke extraction

A main research direction of signature verification is the extraction of the original strokes from the image and thereby the reconstruction of the writing process of the original signature, as this will probably increase the accuracy of the system [26].

Or model allows the chaining of different modules. This means that the stroke extraction could be implemented as an IFeatureExtraction step thus upcoming feature extraction modules in the pipeline could take usage of the extracted stroke information.

Normal thinning algorithms perform poorly on detecting strokes in signatures; therefore a specialized algorithm is used.

First, some definitions must be introduced:

Signature point: a pixel in the image, which belongs to the signature.

Stroke point: strokes are in fact polylines represented by the endpoints of their segments. These endpoints are called here stroke points

Constant values

p: the average pen width (in pixels)

r: the radius of the scanning circle (in pixels) usually $r = p * 2$

d: constant value, must be smaller than r usually $d = r - 1$

Weighted middle point: given a series of points $P_1, P_2 \ldots P_n$, with corresponding intensity values of $i_1, i_2 \ldots$in, the weighted middle point Pm is the first point in the series where

$$\sum_{k=1}^{m} i_k > \frac{1}{2}\sum_{k=1}^{n} i_k \qquad (1)$$

Free point: any pixel which has a minimum distance of d from any previously detected stroke point.

Connected points are pixels which can be connected with a straight path consisting only of signature points

Using the above definitions, the algorithm is defined as follows:

Step 1. – Locate a signature point

Going from the bottom to the top and from the left to the right, locate the first free signature point (*P*). This point is going to be the starting point of our next stroke. *S:=P*

If there is no free point, then exit.

Step 2. – Find connected stroke points

Examine all points of a circle with a center S and radius r. More than *p/2* adjacent signature points on the circle which are connected with S define arcs. Select the weighted middle points (A_1, $A_2 \ldots A_n$,) of these arks as possible stroke points.

Deselect all A_i where A_i is not free, or not connected with *S*. (see Fig. 10.)

Step 3. – Finish the stroke (if needed) and repeat

If *count(selected A)* is

0: Finish the current stroke and go to Step 1

1: Take the ark point as the next stroke point (*S:=A*) and go to Step 2

>1: Finish the stroke and begin two (or more) new strokes with starting point *S*, and take *A1*, *A2*… as the second point. Now follow Step 2 for each of them.

Fig. 11. shows this logic in a form of a flow chart, a sample run of the algorithm is illustrated in Fig 12.

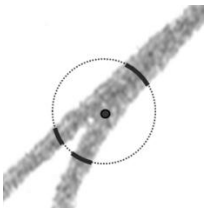

Fig. 10: Demonstration of step 2 from the stroke extraction algorithm. The dark circle in the middle represents S, the dotted line is the scanning circle around S with radius r, the located arcs are shown in black around the circle.

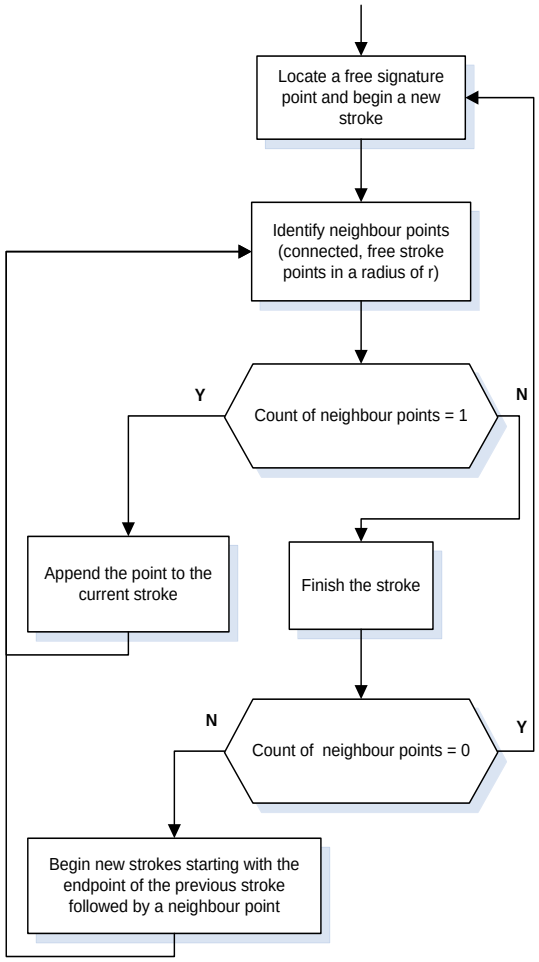

Fig. 11: Flowchart of the stroke extraction algorithm

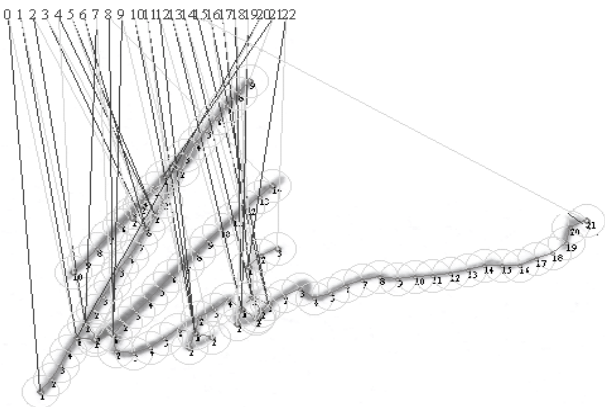

Fig. 12.: Sample run of the stroke extraction algorithm. Strokes are numbered and marked by the dark lines. Scanning circles are illustrated in light gray color. 22 strokes have been located in the signature

5 Experimental results

Although some of the above examples served only demonstration purposes, a framework, taking advantage of the generic signature verification model has already been realized and has been used for two years. It allowed students, to join our signature verification research without the need, to implement their own full featured signature verification systems. All they had to do was to implement a predefined interface. Currently the system supports the addition of
custom preprocessing steps,
custom feature extraction functions,
custom feature matching functions,
custom distance calculation functions,
custom classifiers.

The architecture allowed us to create benchmarks for each of the implemented components and thereby not only measure their effects on the global verification results, but to compare them to each other. An example of such a benchmark can be seen on Fig. 13. Currently, we have working modules performing the following tasks:

- cropping of signatures
- removing underlines from signatures
- removing noise from signatures
- removing
- skeletonization
- stroke extraction based on original signature
- stroke extraction based on skeleton
- extracting intensity information from strokes
- extracting slopes
- extracting baseline information
- matching strokes
- matching baselines
- matching slopes
- comparing baselines
- comparing slopes
- threshold classifier
- ellipsoid classifier
- quartile classifier

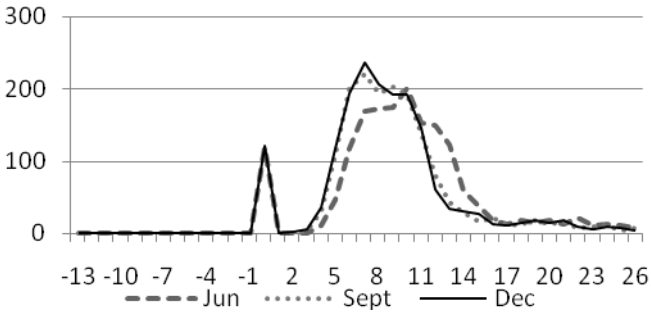

Fig. 13. Improvements of a stroke extraction algorithm [14] [27] [28]

6 Conclusion

Several problems have been presented related to the architecture of off-line signature verification systems. A new highly modularized model was introduced to overcome the identified limitations. It has been demonstrated, that existing signature verifiers can be analyzed and interpreted according to the generic model, and can even be used in a generic framework with slight modifications. Designed with loose coupling between the components the architecture is ideal for team development, load balancing or for application in a service oriented environment, which is subject to our ongoing researches and will be targeted in our future works.

7 ACKNOWLEDGMENT

This project was supported in part by the Innovation and Knowledge Centre of Information Technology BME(IT^2), the National Office for Research and Technology (NKTH) and the Agency for Research Fund Management and Research Exploitation (KPI)

References:

[1] R. Plamondon and G. Lorette, "Automatic Signature Verification and Writer Identification - The State of the Art," 1989, *Pattern Recognition*, Vol. 22, no. 2, pp. 107-131.

[2] F. Leerle and R. Palmond. "Automatic Signature Verification - The State of the Art 1989-1993," 1994, *Int'l Pattern Recognition and Artificial Intelligence, special issue signature verification*, Vol. 8, no. 3, pp. 643-660.

[3] Réjean Plamondon and Srihari N. Sargur. "On-Line and Off-Line Handwriting REcognition: A Comprehensive Survey," 2000, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, no. 1, pp. 63-80.

[4] Bence Kovari. "The development of off-line signature verification methods, comparative study," 2007. microCAD 2007 International Scientific Conference.

[5] "Pattern Recognition, special issue on automatic signature verification," June 1994, Vol. 8, no. 3.

[6] K. Anil Jain. Handwritten Signature Recognition. *Michigan State University - Biometrics.* http://www.cse.msu.edu/~cse891/Sect601/SignatureRcg.pdf.

[7] Julian Fierrez and Javier Ortega-Garcia. On-Line Signature Verification. [book auth.]

Anil K. Jain, Patrick Flynn and Arun A. Ross. *Handbook of Biometrics.* s.l. : Springer US, 2007, pp. 189-209.

[8] V. E. Ramesh and M. Narasimha Murty, "Off-line signature verification using genetically optimized weighted features," 1999,*Pattern Recognition*,no32,pp.217-233.

[9] Alessandra Lumini and Loris Nanni. "Over-complete feature generation and feature selection for biometry," 2007, *Expert Systems with Applications*.

[10] Julian Fierrez-Aguilar, et al. "Fusion of Local and Regional Approaches for On-Line Signature Verification," 2005, *IWBRS 2005, LNCS 3781*, pp. 188–196.

[11] Kai Huang and Hong Yan. "Off-line signature verification using structural feature correspondence," 2002, *Pattern Recognition*, no. 35, pp. 2467 – 2477.

[12] Se-Hoon Kim, Kie-Sung Oh and Hyung-Il Choi. "Off-line Verification System of the Handwrite, Signature or Text using a Dynamic Programming," 2007, *ICCSA 2007, LNCS 4705*, no. 1, pp. 1014-1023.

[13] Gregory F. Russell and Alain Biem Jianying Hu. "Dynamic Signature Verification Using Discriminative Training," 2005. Proceedings of the 2005 Eight International Conference on Document Analysis and Recognition (ICDAR'05). 1520-5263/05 IEEE.

[14] Bence Kovari. "Time-Efficient Stroke Extraction Method for Handwritten Signatures," 2007. ACS07, The 7th WSEAS International Conference on Applied Computer Science. pp. 157-161. ISBN 978-960-6766-15-2, ISSN 1790-5117.

[15] E. Frias-Martinez, A. Sanchez and J. Velez. "Support vector machines versus multi-layer perceptrons for efficient off-line signature recognition," 2006, *Engineering Applications of Artificial Intelligence*,no. 19

[16] S. Hangai, S. Yamanaka and T. Hamamoto. "Writer Verification using Altitude and Direction of Pen Movement," 2000. IEEE, Proceedings of the International Conference on Pattern Recognition (ICPR'00).

[17] M. E. Munich and P. Perona. "Visual Identification by Signature Tracking," February 2003, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, no. 2.

[18] William K. Pratt. *Digital Image Processing: PIKS Scientific Inside.* s.l. : Wiley-Interscience, 2007. ISBN 978-0-471-76777-0.

[19] S. Akle, M.-E. Algorri and A. Salcedo, "Use of wavelet-based basis functions to extract rotation invariant features for automatic image recognition ," 2008, *WSEAS Transactions on Information Science and Applications*, Vol. 5, no. 5, pp. 664-673. ISSN: 1790-0832.

[20] X.D. Zhuang and N.E. Mastorakis. "The curling vector field transform of gray-scale images: A magneto-static inspired approach," 2008, *WSEAS Transactions on Computers*, Vol. 7, no. 3, pp. 147-153 .

[21] Larman. *Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design and Iterative Development.* s.l. : Prentice-Hall, 2004. ISBN: 978-0137488803.

[22] Bence Kovari, Zsolt Kertesz and Attila Major, "Off-Line Signature Verification Based on Feature Matching," Budapest : s.n., 2007. IEEE, 11th International Conference on Intelligent Engineering Systems (INES). pp. 93-97.

[23] L. Lam, Seong-Whan Lee and Ching Y. Suen. "Thinning Methodologies-A Comprehensive Survey," September 1992, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, no. 9, p. 879.

[24] Z. Zhang. "Iterative point matching for registration of free-form curves and surfaces," 1994, *International Journal of Computer Vision, vol.13, no.2*, pp. 119-152.

[25] R. Nerino. "Automatic registration of point-based surfaces ," 2006, *WSEAS Transactions on Computers*, Vol. 5, no. 12, pp. 2984-2991.

[26] N.L. Othman, J. Shin and W.-D. Chang. "Chain code distance: A global feature for on-line signature verification ," 2006, *WSEAS Transactions on Computers* , Vol. 5, no. 9, pp. 2037-2042. ISSSN 1109-2750 .

[27] Csaba Illes and Bence Kovari. "Robust signature stroke extraction for use in off-line signature verification," 2007. microCAD 2007 International Scientific Conference.

[28] Bence Kovari, et al. "Off-Line Signature Verification - Comparison of Stroke Extraction Methods," Barcelona : s.n., 2007. ICSOFT, 2nd International Conference on Software and Data Technologies. pp. 270-276. ISBN 978-989-8111-09-8.