Human Authentication through Signature Recognition

Víctor Nàcher Castellet

Supervisor: **Prof. Francesc Serratosa**
Doctor of Philosophy in Automatic Control, Robotics and Computer Vision

Universitat Oberta de Catalunya
Universitat Autònoma de Barcelona
Universitat Rovira i Virgili

This dissertation is submitted for the degree of
Master's Degree in Information and Communications Security

June 2019

I would like to dedicate this thesis to my uncle, my biggest scientific inspiration...

Abstract

This dissertation describes a signature verification algorithm based on k-nearest neighbours and Levenshtein distance. It is a simple on-line method for classifying handwritten signatures into either genuines or forgeries. Forgery consists on imitating a signature, a document, a banknote or a valued work. It is considered a crime in most countries and it still remains a major problem in nowadays banking.

In first place, a brief introduction to the history of biometric systems and handwritten signatures is made. After explaining the state of the art of the field the used well-known methods are explained. Then the used datased is showed and our algorithm presented and implemented.

Resumen

Este trabajo de fin de máster describe un algoritmo de verificación de firmas basado en k-nearest neighbours y la distancia de Levenshtein. Es un método simple y on-line para clasificar firmas hechas a mano en auténticas o falsificaciones. Una falsificación consiste en imitar una firma, un documento, un billete o una obra de valor. Se considera un delito en la mayoría de países y sigue siendo un problema importante para los bancos hoy en día.

En primer lugar, se realiza una breve introducción a la historia de los sistemas biométricos y de las firmas manuscritas. Después de explicar el estado del arte del sector, se explican los métodos conocidos utilizados. Por último, se muestra la base de datos utilizada y se presenta e implementa nuestro algoritmo.

Resum

Aquest treball de fi de màster descriu un algoritme de verificació de signatures basat en k-nearest neighbours i la distància de Levenshtein. És un mètode simple i on-line per classificar signatures fetes a mà en autèntiques o falsificacions. Una falsificació consisteix en imitar una firma, un document, un bitllet o una obra de valor. Es considera un delicte en la majoria de països i segueix sent un problema important per als bancs avui dia.

En primer lloc, es realitza una breu introducció a la història dels sistemes biomètrics i de les signatures manuscrites. Després d'explicar l'estat de l'art del sector, s'expliquen els mètodes coneguts utilitzats. Finalment, es mostra la base de dades utilitzada i es presenta i implementa el nostre algoritme.

Table of contents

List of figures

List of tables

Objectives

Whilst taking the *Biometrics* course from the Master's, I discovered this area and I got amazed by the endless possibilities that this field could bring. Pursuing my studies as a telecommunication engineer I specialised in image processing so when this work was proposed I found it as the perfect junction between this two domains.

The ultimate objective during this dissertation project was to come up with an algorithm to verify handwritten signatures from images. The main constraint was to keep it as simple as possible.

The system evolved in the form keeping its essence. A handwritten verification system was developed assuming a preprocessed input of data. The SVC2004: First International Signature Verification Competition dataset was used.

With that purpose, other main objectives were set for this thesis: to set the basis for a possible future mobile application of handwritten signature verification, that is, convert the method to be able to apply it as an offline system. Last but not least, as aforementioned, keep used processing and algorithms as simple as possible while maintaining significant performance rates.

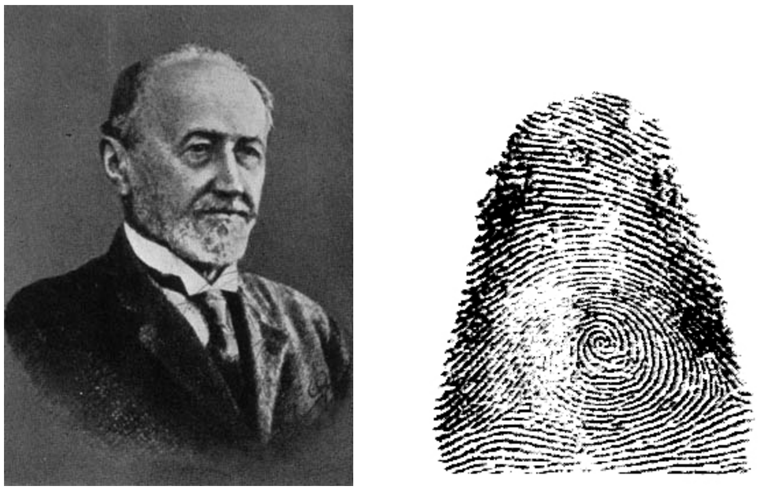
Chapter 1

Introduction

1.1 History

It was not until the last few decades that automated biometric systems became available. This does not necessarily mean that they are new concepts since most of them apply traditional techniques enhanced by the revolution of computer processing. A clear example of this is the identification of individuals through their fingerprints. The Croatian born Argentinian anthropologist and police official Juan Vucetich is considered one of the pioneers in the use of them. In 1892 he discovered the real murderer of his sons thanks to a bloody fingerprint left in the postbox [3]. Later on, Sir Francis Galton was the first man to design a fingerprint classification method in 1888 based on Sir William Herschel works in the 1860s [4]. Some of nowadays identification algorithms still use his theories.

There are many kinds of identification methods based on the used materials such as the possession of a token (card, key, etc.) or the knowledge of a secret (password, key phase,etc.). In opposition to this, biometrics refers to individual recognition based on a person's distinguishing characteristics [1]. The term derives from ancient Greek: *"bio"* which means life and *"metrics"*, to measure. This is why nothing else but our own characteristics should be used. Furthermore, it is easy to see that better discriminating characteristics will be more interesting no matter which is the purpose of the biometric system.



(a) Juan Vucetich, anthropologist and police official pioneer in the use of fingerprints.

(b) Right thumb fingerprint of Francisca Rojas.

Fig. 1.1 First known use of fingerprints for crime solving in 1892.

1.2 Biometric Systems

Biometric systems can be divided into two groups depending on its intention: verification or identification. In the first mode the goal is to authenticate the person's identity comparing it to the one he or she claims to have. On the other hand, when identifying the system is the one in charge of stating the individual's identity. There are a great variety of possible features used such as voice, iris, fingerprints, behavioural traits, face, retina, handwritten signature, etc. [5], [6], [7], [8].

1.3 Handwritten Signatures

In this project we will be focused in verification systems since the goal is to reject or accept handwritten signatures as genuine or forged ones. Forgery consists on imitating a signature, a document, a banknote or a valued work. It is considered a crime in most countries and it still remains a major problem in nowadays banking.

Plenty of methods have been proposed and they are mainly divided in two groups since Even, Goldreich and Micali established a classification criterion [9]. Depending on the availability of the data they are usually classed as offline and online verification systems. In offline verification the full finished signature is used, for example, we receive a signed bank check and we analyse the signature to decide if it has been forged or it is rather genuine. On the other hand, online verification disposes of "live" data, that is data that feeds the verification algorithm while the process is being done. An example could be when the postman arrives at your place to deliver a parcel, the machine where you digitally sign could be live-feeding data to a verification algorithm.

Online methods have shown better performance in the last years (see tables V and VI in [1]. However, they are much more difficult to implement in most cases. Let us take the example of bank checks. When a person tries to cash a check, the bank worker receives it with the signature already on it so there are no means to implement an online verification system.

1.3.1 Our approach

In this work we decided to follow the approach of an online system with the premise of making it as simple as possible without a loss of performance. Our algorithm needs to be fed with an ordered sequence of points. When people sign, they usually do it in the same order. For writing Victor I would write first the *V*, then the *i*, and so on. Even for each letter, like the *V* I will always start from top left and end at the top right.

Guided by this, the chosen data was "Sample Data", taken from the well-known SVC2004: First International Signature Verification Competition [2]. Only the *X* and *Y* data where used in order to simulate a preprocessed version of an image capturing and processing system. Any additional information available such as pressure or signing time were ignored.

Chapter 2

State of the Art

The enrolment of features in a knowledge base constitutes the first step in a verification process. After feature extraction, these features associated to genuine signatures are stored next to an identity. This allows to test against this knowledge base all new inputs to the system. It will return either true (identified) or false (not identified), meaning this last that the input is a forged signature or the person is not enrolled in the database.

In figure 2.1 we can see a non-exhaustive list of different methods grouped by principle. The input set is tested via simple matching against templates stored in the knowledge base when using template matching. They can also be tested against forgery templates. The most used approach within this family is using dynamic time warping (DWT) [10].

When template matching techniques are considered, a questioned sample is matched against templates of authentic/forgery signatures. In this case, the most common approaches use DTW for signature matching [10, 11, 12, 13].

Within statistical approaches, distance-based classifiers are a classic option and the one followed in this dissertation. Artificial neural networks (ANN) have been recently increasing their presence [14, 15], specially convolutional neural networks (CNN) thanks to their capability of classifying images [16, 17]. Even more complicated schemes like fuzzy neural networks [18]. Lately, hidden Markov models (HMM) are getting more and more attention [19, 20, 21].

Signature descriptions by their elementary elements or primitives are used in structural approaches. They are compared by matching of graphs, trees, strings, etc [22, 23].

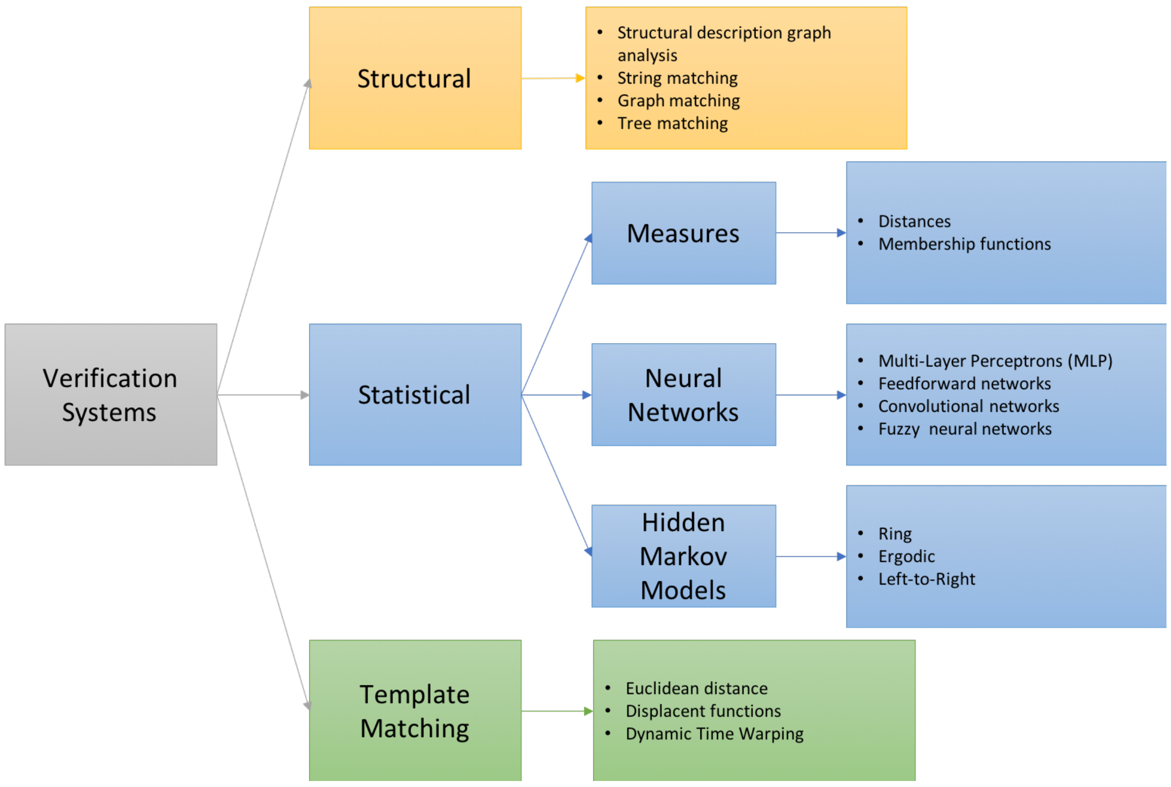


Fig. 2.1 Non-exhaustive list of verification methods grouped by principle

Chapter 3

Nomenclature and Methods

3.1 Levenshtein Distance

The Levenshtein distance is a metric used in linguistics [24] and information theory for measuring the difference between two text sequences [25]. It receives its name after the Soviet mathematician Vladimir Levenshtein, after he defined it in the sixties.

$$d_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} d_{a,b}(i-1,j) + C_{del}(b_i) \\ d_{a,b}(i,j-1) + C_{ins}(a_j) \\ d_{a,b}(i-1,j-1) + C_{sub}(b_i) \cdot \mathbb{1}_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases} \tag{3.1}$$

Equation 3.1 defines the Levenshtein distance between the first i characters in the a sequence and the first j characters in sequence b, where $\mathbb{1}_{(a_i \neq b_j)}$ is the indicator function defined as in equation 3.2:

$$\mathbb{1}_{(a_i \neq b_j)} = \begin{cases} 0 & \text{if } a = b \\ 1 & \text{if } a \neq b \end{cases} \tag{3.2}$$

From these expressions we can see that the Levenshtein distance corresponds to the minimum number possible of editions of a single character to transform one sequence into the other. These editions are deletions, insertions and substitutions. C_{del}, C_{ins} and C_{sub} indicate respectively the associated costs for each operation. In our work 1 was considered for all of them.

$$C_{del} = C_{ins} = C_{sub} = 1$$

For example, $d_{care,cast} = 2$:

1. care $\rightarrow$ case (substitution of "r" for "s")
2. case $\rightarrow$ cast (substitution of "e" for "t")

3.1.1 Iterative with full matrix

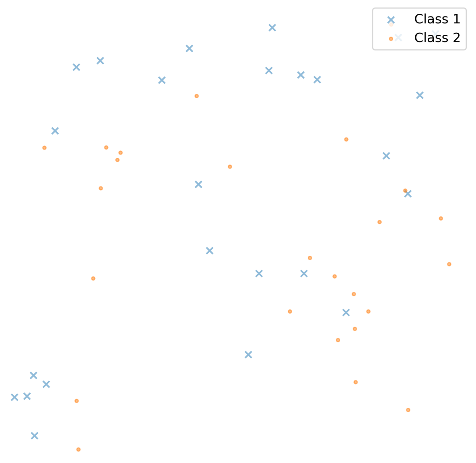
The way of computing the Levenshtein distance is in a full iterative way, following Wagner and Fischer implementation [26]. It can be dynamically programmed thanks to the fact that a matrix is built to maintain all the previously calculated distances (from the first index up until the current one). The last computed value will be the final distance between the two sequences.

3.2 K-nearest Neighbours Approach

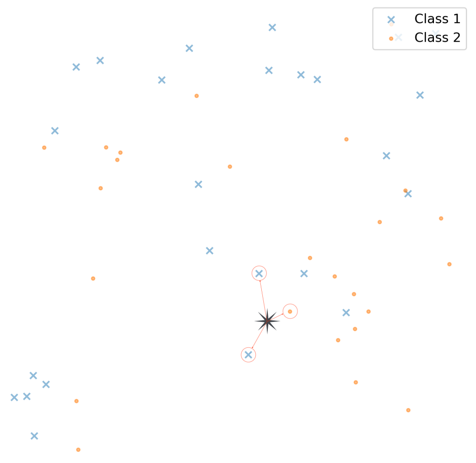
K-nearest neighbours or k-NN classification is a non-parametric method for splitting data into classes. The k nearest training samples are the input of the algorithm, who decides output's class by voting these neighbours' classes. k is an integer and usually small [27].

From this definition one can see that a distance metric must be defined since we need to take the closest samples. In this work we used the Levenshtein distance defined in section 3.1.

In figure 3.1 we can see an example of 3-nearest neighbour algorithm applied to a two classes dataset. In subfigure 3.1a we observe the available training set. In subfigure 3.1b a new point has appeared and needs to be classified. The 3 nearest points from the training set are taken into account, resulting in two votes for class 1 and one vote for class 2. Hence, the new point will be classified as class 1.



(a) Available training dataset



(b) New point and its 3 nearest neighbours

Fig. 3.1 Example of 3-nearest neighbour algorithm applied to a two classes dataset

Chapter 4

Our Algorithm

4.1 Data Preprocessing

In order to be able to use K-Nearest neighbours with Levenshtein distance as a metric (see chapter 3), we first need to manipulate the raw input data. We need text sequences so the followed approach was to convert signatures to text following the following algorithm:

We go through the whole signature point by point performing the following steps:

1. If it is the first point, skip following steps and move to the next one
2. Compare (x_2, y_2) coordinates of next point with actual point (x_1, y_1).
    - if $x_1 > x_2$ and $y_1 > y_2 \rightarrow$ substitute point with 'b'.
    - if $x_1 > x_2$ and $y_1 < y_2 \rightarrow$ substitute point with 'h'.
    - if $x_1 < x_2$ and $y_1 > y_2 \rightarrow$ substitute point with 'd'.
    - if $x_1 < x_2$ and $y_1 < y_2 \rightarrow$ substitute point with 'f'.
    - if $x_1 > x_2$ and $y_1 = y_2 \rightarrow$ substitute point with 'a'.
    - if $x_1 < x_2$ and $y_1 = y_2 \rightarrow$ substitute point with 'e'.
    - if $x_1 = x_2$ and $y_1 > y_2 \rightarrow$ substitute point with 'c'.
    - if $x_1 = x_2$ and $y_1 < y_2 \rightarrow$ substitute point with 'g'.
3. If it is not the last point, move to the next one and go back to first step.

Following this steps we end up with a text sequence generated from the original signature in accordance with figure 4.1.

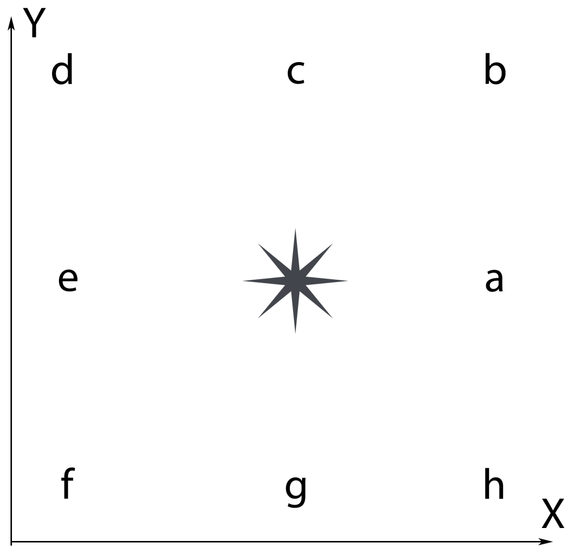


Fig. 4.1 Character allocation in function of relative position of next point with respect to the actual point.

4.2 Algorithm

Once we have the text sequences generated from the signatures we just need to use the well-known k-Neighbours algorithm (see section 3.2) using the Levenshtein's distance as the metric (see section 3.1). In the next chapter we will see the details of the implementation.

Chapter 5

Implementation

5.1 Data splitting

We split all the data present in the dataset [2] into a training set and a test set. These are of size 66% and 33% of the original one. Code is available in listing A.3. Once this is done for the 5 users, we proceed to compute the Levenshtein distances.

5.2 Distances Calculation

In order to implement the proposed algorithm in chapter 4 we firstly compute all distances between the signatures. That is, the Levenshtein distance (see 3.1) between the first signature to all the other 19 from user one. Then, the second one from the 18 remaining and so on. See listing A.4.

5.3 Execution

We are then ready to execute the whole algorithm, calculating training and test errors. In listing A.5 we see how we obtain the best parameter k for the algorithm minimising the training error and then apply it to classify the test set.

Chapter 6

Experimental Validation

6.1 Dataset Structure

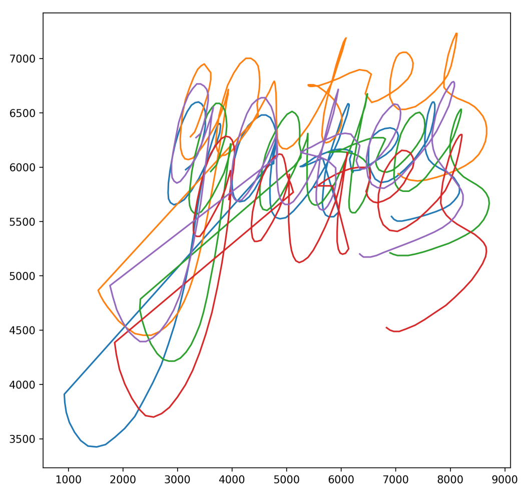
As aforementioned, the used data was "Sample Data", taken from the well-known SVC2004: First International Signature Verification Competition [2]. This dataset is composed of 5 users with 40 signatures each: 20 original ones and 20 skilled forgeries.

For each signature, we dispose of a text representation of a sequence of points. The first line is an integer indicating the total number of points conforming the signature and the rest corresponds to one point per line. Each line has 7 features:

Table 6.1 Dataset signature features

Feature	Details
X-coordinate	scaled cursor position along the x-axis
Y-coordinate	scaled cursor position along the y-axis
Time stamp	system time at which the event was posted
Button status	current button status (0 for pen-up and 1 for pen-down)
Azimuth	clockwise rotation of cursor about the z-axis
Altitude	angle upward toward the positive z-axis
Pressure	adjusted state of the normal pressure

Only the X and Y data where used in order to simulate a preprocessed version of an image capturing and processing system. Any additional information available such as pressure or signing time were ignored.


(a) 5 original signatures from user 3

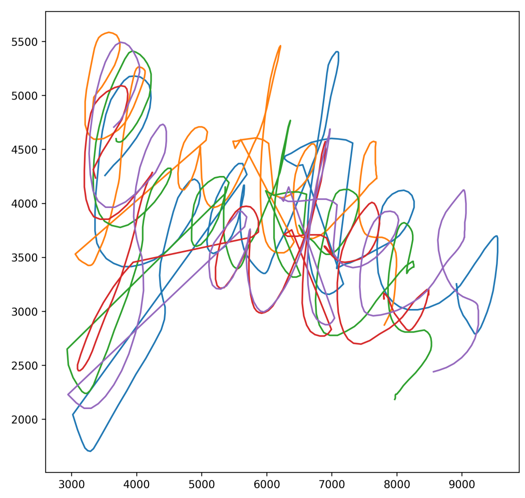

(b) 5 forged signatures from user 3

Fig. 6.1 Example of original and forged signatures contained in the dataset [2].

6.2 Obtained Results

As seen in section 5.3 we obtain the best parameter k for the algorithm minimising the training error and then apply it to classify the test set.

Figure 6.2 shows the optimisation of parameter k of the algorithm. We obtained 2 as the number of neighbours with a training error of 9.2308%.

Applying this parameter to the test set we obtained a test error of 5.7143%. This values are in concordance with other methods from the state of the art (See Table V from [1] included in appendix B).

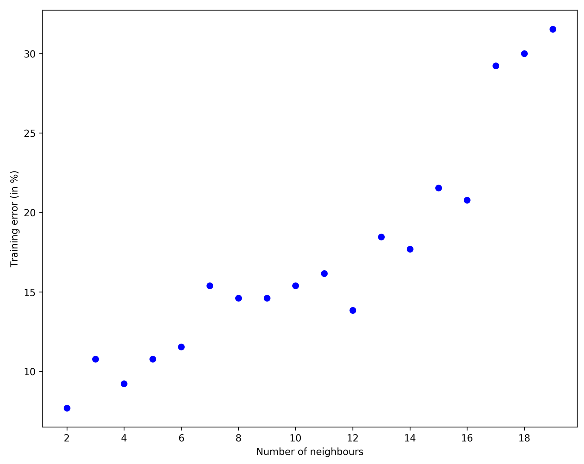


Fig. 6.2 K-neighbour algorithm parameter optimisation

Conclusions and Future Work

Signature verification remains an important field of study in security. With every step of evolution in forgeries quality, verification methods must do as well.

Here we demonstrated a very simple method, yet significant, to verify signatures. The main limitation is the need to know the points' order, which doesn't yet enable off-line verification schemes.

Solving this last inconvenient could enable further projects such as a mobile signature scanning app, installed on portable battery-constrained devices that classify signatures into original or forgeries.

References

[1] Donato Impedovo and Giuseppe Pirlo. Automatic signature verification: The state of the art. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(5):609–635, 2008.

[2] First international signature verification competition, svc2004. http://www.cse.ust.hk/svc2004/download.html, 2004.

[3] Juan Vucetich. *Dactiloscopía comparada: el nuevo sistema argentino*. Establecimiento Tipográfico Jacobo Peuser, 1904.

[4] Francis Galton. *Finger Prints*. Macmillan, 1892.

[5] Kevin W Boyer, Venu Govindaraju, and Nalini K Ratha. Introduction to the special issue on recent advances in biometric systems [guest editorial]. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(5):1091–1095, 2007.

[6] Kalyan Veeramachaneni, Lisa Ann Osadciw, and Pramod K Varshney. An adaptive multimodal biometric management algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(3):344–356, 2005.

[7] Anil K Jain, Patrick Flynn, and Arun A Ross. *Handbook of biometrics*. Springer Science & Business Media, 2007.

[8] David D Zhang. *Automated biometrics: Technologies and systems*, volume 7. Springer Science & Business Media, 2013.

[9] Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures. In *Conference on the Theory and Application of Cryptology*, pages 263–275. Springer, 1989.

[10] Marc Parizeau and Rejean Plamondon. A comparative analysis of regional correlation, dynamic time warping, and skeletal tree matching for signature verification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):710–717, 1990.

[11] Brigitte Wirtz. Stroke-based time warping for signature verification. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 179–182. IEEE, 1995.

[12] A Piyush Shanker and AN Rajagopalan. Off-line signature verification using dtw. *Pattern recognition letters*, 28(12):1407–1414, 2007.

[13] Oscar Miguel-Hurtado, Luis Mengibar-Pozo, Michael G Lorenz, and Judith Liu-Jimenez. On-line signature verification by dynamic time warping and gaussian mixture models. In *2007 41st Annual IEEE International Carnahan Conference on Security Technology*, pages 23–29. IEEE, 2007.

[14] Amir Soleimani, Babak N Araabi, and Kazim Fouladi. Deep multitask metric learning for offline signature verification. *Pattern Recognition Letters*, 80:84–90, 2016.

[15] Ruben Tolosana, Ruben Vera-Rodriguez, Julian Fierrez, and Javier Ortega-Garcia. Exploring recurrent neural networks for on-line handwritten signature biometrics. *IEEE Access*, 6:5128–5138, 2018.

[16] Luiz G Hafemann, Robert Sabourin, and Luiz S Oliveira. Learning features for offline handwritten signature verification using deep convolutional neural networks. *Pattern Recognition*, 70:163–176, 2017.

[17] Bernardete Ribeiro, Ivo Gonçalves, Sérgio Santos, and Alexander Kovacec. Deep learning networks for off-line handwritten signature recognition. In *Iberoamerican Congress on Pattern Recognition*, pages 523–532. Springer, 2011.

[18] Nabeel A Murshed, Flavio Bortolozzi, and Robert Sabourin. Off-line signature verification using fuzzy artmap neural network. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 2179–2184. IEEE, 1995.

[19] Bao Ly Van, Sonia Garcia-Salicetti, and Bernadette Dorizzi. On using the viterbi path along with hmm likelihood information for online signature verification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(5):1237–1247, 2007.

[20] Julian Fierrez, Javier Ortega-Garcia, Daniel Ramos, and Joaquin Gonzalez-Rodriguez. Hmm-based on-line signature verification: Feature extraction and signature modeling. *Pattern recognition letters*, 28(16):2325–2334, 2007.

[21] Edson JR Justino, Flávio Bortolozzi, and Robert Sabourin. Off-line signature verification using hmm for random, simple and skilled forgeries. In *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pages 1031–1034. IEEE, 2001.

[22] Ibrahim SI Abuhaiba. Offline signature verification using graph matching. *Turkish Journal of Electrical Engineering & Computer Sciences*, 15(1):89–104, 2007.

[23] Siyuan Chen and Sargur Srihari. A new off-line signature verification method based on graph. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 2, pages 869–872. IEEE, 2006.

[24] Wilbert Jan Heeringa. *Measuring dialect pronunciation differences using Levenshtein distance*. PhD thesis, Citeseer, 2004.

[25] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, number 8, pages 707–710, 1966.

[26] Robert A Wagner and Michael J Fischer. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173, 1974.

[27] Thomas M Cover, Peter E Hart, et al. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

Appendix A

Pyhton implementations

Listing A.1 Levesnshtein distance implementation

```
def levenshtein(seq1, seq2, print_full_matrix=False):
    size_x = len(seq1) + 1
    size_y = len(seq2) + 1
    matrix = np.zeros ((size_x, size_y))
    for x in range(size_x):
        matrix [x, 0] = x
    for y in range(size_y):
        matrix [0, y] = y

    for x in range(1, size_x):
        for y in range(1, size_y):
            if seq1[x-1] == seq2[y-1]:
                matrix [x,y] = min(
                    matrix[x-1, y] + 1,
                    matrix[x-1, y-1],
                    matrix[x, y-1] + 1
                )
            else:
                matrix [x,y] = min(
                    matrix[x-1,y] + 1,
                    matrix[x-1,y-1] + 1,
                    matrix[x,y-1] + 1
                )
    if print_full_matrix:
        print (matrix)
return (matrix[size_x - 1, size_y - 1])
```

Listing A.2 k neighbours implementation

```
def my_k_neighbours(distances,y,n_neighbours = 3, debug=False):
    errors=0
    for j in range(1,6):
        if debug:
            print("User "+str(j))
        for i,val in enumerate(distances[j]):
            idx = np.argpartition(distances[j][i], n_neighbours+1)
            k_neighbours_idx = idx[0:n_neighbours+1][idx[0:n_neighbours
                ↪ +1]!=i]
            neighbours_y = [y[j][a] for a in k_neighbours_idx]
            if (sum(d for d in neighbours_y if d))/n_neighbours <= 0.5:
            # Majority of 0s in the n_neighbours of the i-th signature
                if y[j][i]==0:
                    # estimation OK
                    pass
                else:
                    if debug:
                        print(str(k_neighbours_idx)+" --> ", end="")
                        print(neighbours_y, end="")
                        print("****** "+str(y[j][i]),end="")
                        print(" --> ESTIMATION ERROR")
                    errors+=1
            else:
            # Majority of 1s in the n_neighbours of the i-th signature
                if y[j][i]==1:
                    # estimation OK
                    pass
                else:
                    if debug:
                        print(i,end="")
                        print(str(k_neighbours_idx)+" --> ", end="")
                        print(neighbours_y, end="")
                        print("****** "+str(y[j][i]),end="")
                        print(" --> ESTIMATION ERROR")
                    errors+=1

    error = 100*errors/(n_users*len(distances[1]))
    if debug:
        print("\n\nTraining error = ",end="")
        print("%.4f" % round(error,4),end="")
        print("%")
return error
```

Listing A.3 Dataset splitting into train and test sets

```
X_train_all_users = {}
X_test_all_users = {}
y_train_all_users = {}
y_test_all_users = {}
for i in range(n_users):
    X_temp = []
    for key, value in X.items(): # iter on both keys and values
        if key.startswith(str(i+1)+'-'):
            X_temp.append(X[key])
    X_train, X_test, y_train, y_test = train_test_split(X_temp, yp
        ↪ [:40], test_size=0.33, random_state=42)

    X_train_all_users[i+1] = X_train
    X_test_all_users[i+1] = X_test
    y_train_all_users[i+1] = y_train
    y_test_all_users[i+1] = y_test
```

Listing A.4 Distance calculation

```
# We will build features as the distance with all the rest
#print(" 1 --> Original | 0 --> Fake ")
all_users_distances_train={}
for j in range(1,6):
    distances = np.zeros((len(X_train_all_users[j]),len(
        ↪ X_train_all_users[j])))

    for i in range(len(X_train_all_users[j])):
        for k in range(len(X_train_all_users[j])):
            distances[i,k] = levenshtein(X_train_all_users[j][k],
                ↪ X_train_all_users[j][i])

    all_users_distances_train[j] = distances



all_users_distances_test = {}
for j in range(1,6):
    distances = np.zeros((len(X_test_all_users[j]),len(X_test_all_users
        ↪ [j])))

    for i in range(len(X_test_all_users[j])):
        for k in range(len(X_test_all_users[j])):
            distances[i,k] = levenshtein(X_test_all_users[j][k],
                ↪ X_test_all_users[j][i])

    all_users_distances_test[j] = distances
```

Listing A.5 Training and test error calculations

```
tr_err = []
bias = 1 # To prevent overfitting from which k we start. (1-neighbour
    ↪ is dangerous, variance too high).
for k in range(1+bias,20):
    tr_err.append(my_k_neighbours(all_users_distances_train,
        ↪ y_train_all_users,k))
best_k = np.argpartition(tr_err,1)[0]+1+bias
print("Best parameter: k = "+str(best_k))

ax = plt.figure().gca()
ax.plot(np.arange(1+bias,len(tr_err)+1+bias),tr_err,'bo')
ax.xaxis.set_major_locator(MaxNLocator(integer=True))
```

```
plt.xlabel("Number of neighbours")
plt.ylabel("Training error (in %)")
plt.title("K-Neighbour algorithm parameter minimisation")
plt.show()

print("\n\nTraining error = ",end="")
print("%.4f" % round(tr_err[best_k],4),end="")
print("%")

############

test_err = my_k_neighbours(all_users_distances_test,y_test_all_users,
    ↪ best_k)
print("Test error = ",end="")
print("%.4f" % round(test_err,4),end="")
print("%")
```

Appendix B

Tables V and VI from Impedovo and Pirlo [1]

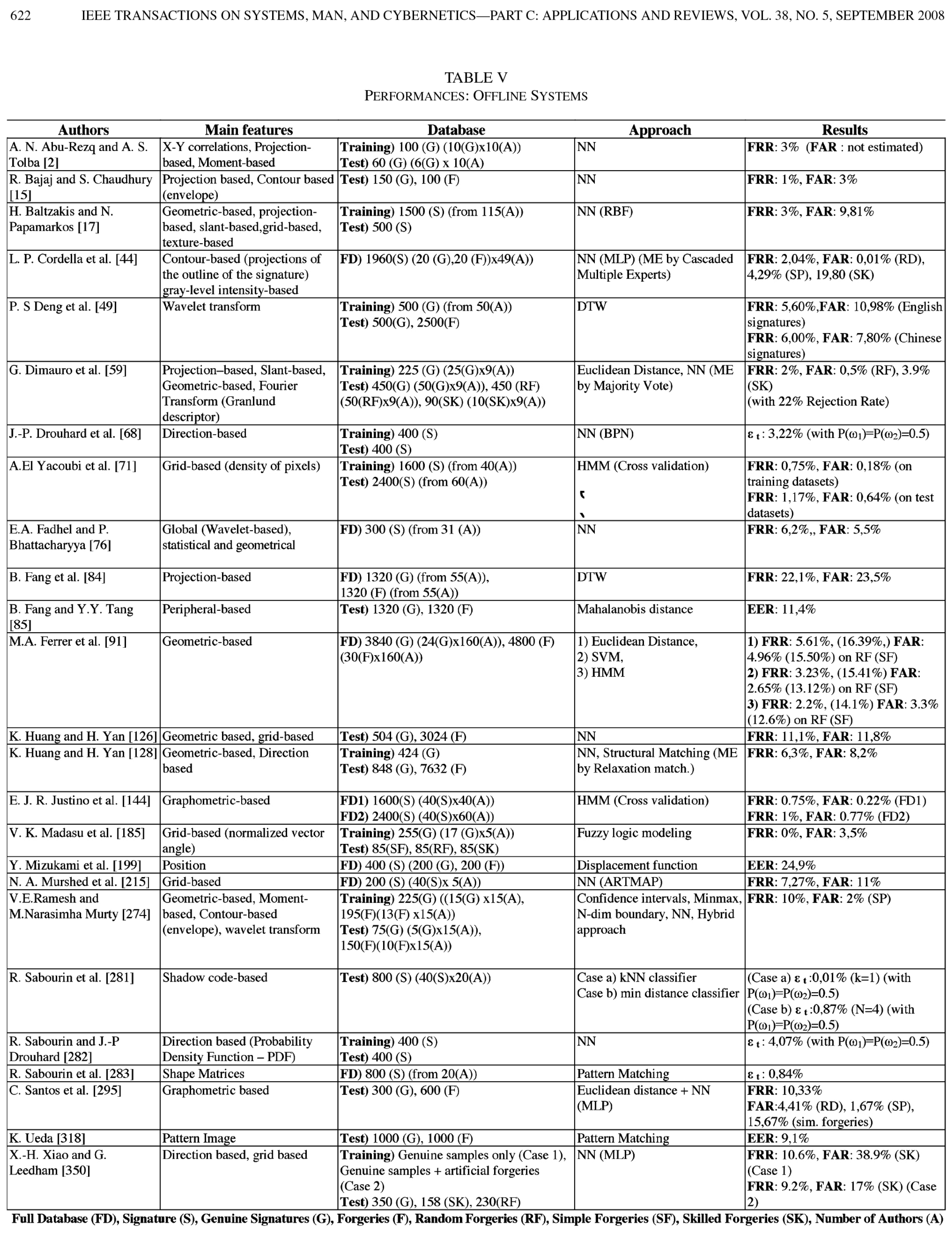


TABLE V
PERFORMANCES: OFFLINE SYSTEMS

Authors	Main features	Database	Approach	Results
A. N. Abu-Rezq and A. S. Tolba [2]	X-Y correlations, Projection-based, Moment-based	**Training**) 100 (G) (10(G)x10(A)) **Test**) 60 (G) (6(G) x 10(A)	NN	**FRR**: 3% (**FAR** : not estimated)
R. Bajaj and S. Chaudhury [15]	Projection based, Contour based (envelope)	**Test**) 150 (G), 100 (F)	NN	**FRR**: 1%, **FAR**: 3%
H. Baltzakis and N. Papamarkos [17]	Geometric-based, projection-based, slant-based,grid-based, texture-based	**Training**) 1500 (S) (from 115(A)) **Test**) 500 (S)	NN (RBF)	**FRR**: 3%, **FAR**: 9,81%
L. P. Cordella et al. [44]	Contour-based (projections of the outline of the signature) gray-level intensity-based	**FD**) 1960(S) (20 (G),20 (F))x49(A))	NN (MLP) (ME by Cascaded Multiple Experts)	**FRR**: 2,04%, **FAR**: 0,01% (RD), 4,29% (SP), 19,80 (SK)
P. S Deng et al. [49]	Wavelet transform	**Training**) 500 (G) (from 50(A)) **Test**) 500(G), 2500(F)	DTW	**FRR**: 5,60%,**FAR**: 10,98% (English signatures) **FRR**: 6,00%, **FAR**: 7,80% (Chinese signatures)
G. Dimauro et al. [59]	Projection–based, Slant-based, Geometric-based, Fourier Transform (Granlund descriptor)	**Training**) 225 (G) (25(G)x9(A)) **Test**) 450(G) (50(G)x9(A)), 450 (RF) (50(RF)x9(A)), 90(SK) (10(SK)x9(A))	Euclidean Distance, NN (ME by Majority Vote)	**FRR**: 2%, **FAR**: 0,5% (RF), 3.9% (SK) (with 22% Rejection Rate)
J.-P. Drouhard et al. [68]	Direction-based	**Training**) 400 (S) **Test**) 400 (S)	NN (BPN)	ε_t: 3,22% (with $P(\omega_1)=P(\omega_2)=0.5$)
A.El Yacoubi et al. [71]	Grid-based (density of pixels)	**Training**) 1600 (S) (from 40(A)) **Test**) 2400(S) (from 60(A))	HMM (Cross validation)	**FRR**: 0,75%, **FAR**: 0,18% (on training datasets) **FRR**: 1,17%, **FAR**: 0,64% (on test datasets)
E.A. Fadhel and P. Bhattacharyya [76]	Global (Wavelet-based), statistical and geometrical	**FD**) 300 (S) (from 31 (A))	NN	**FRR**: 6,2%,, **FAR**: 5,5%
B. Fang et al. [84]	Projection-based	**FD**) 1320 (G) (from 55(A)), 1320 (F) (from 55(A))	DTW	**FRR**: 22,1%, **FAR**: 23,5%
B. Fang and Y.Y. Tang [85]	Peripheral-based	**Test**) 1320 (G), 1320 (F)	Mahalanobis distance	**EER**: 11,4%
M.A. Ferrer et al. [91]	Geometric-based	**FD**) 3840 (G) (24(G)x160(A)), 4800 (F) (30(F)x160(A))	1) Euclidean Distance, 2) SVM, 3) HMM	**1) FRR**: 5.61%, (16.39%,) **FAR**: 4.96% (15.50%) on RF (SF) **2) FRR**: 3.23%, (15.41%) **FAR**: 2.65% (13.12%) on RF (SF) **3) FRR**: 2.2%, (14.1%) **FAR**: 3.3% (12.6%) on RF (SF)
K. Huang and H. Yan [126]	Geometric based, grid-based	**Test**) 504 (G), 3024 (F)	NN	**FRR**: 11,1%, **FAR**: 11,8%
K. Huang and H. Yan [128]	Geometric-based, Direction based	**Training**) 424 (G) **Test**) 848 (G), 7632 (F)	NN, Structural Matching (ME by Relaxation match.)	**FRR**: 6,3%, **FAR**: 8,2%
E. J. R. Justino et al. [144]	Graphometric-based	**FD1**) 1600(S) (40(S)x40(A)) **FD2**) 2400(S) (40(S)x60(A))	HMM (Cross validation)	**FRR**: 0.75%, **FAR**: 0.22% (FD1) **FRR**: 1%, **FAR**: 0.77% (FD2)
V. K. Madasu et al. [185]	Grid-based (normalized vector angle)	**Training**) 255(G) (17 (G)x5(A)) **Test**) 85(SF), 85(RF), 85(SK)	Fuzzy logic modeling	**FRR**: 0%, **FAR**: 3,5%
Y. Mizukami et al. [199]	Position	**FD**) 400 (S) (200 (G), 200 (F))	Displacement function	**EER**: 24,9%
N. A. Murshed et al. [215]	Grid-based	**FD**) 200 (S) (40(S)x 5(A))	NN (ARTMAP)	**FRR**: 7,27%, **FAR**: 11%
V.E.Ramesh and M.Narasimha Murty [274]	Geometric-based, Moment-based, Contour-based (envelope), wavelet transform	**Training**) 225(G) ((15(G) x15(A), 195(F)(13(F) x15(A)) **Test**) 75(G) (5(G)x15(A)), 150(F)(10(F)x15(A))	Confidence intervals, Minmax, N-dim boundary, NN, Hybrid approach	**FRR**: 10%, **FAR**: 2% (SP)
R. Sabourin et al. [281]	Shadow code-based	**Test**) 800 (S) (40(S)x20(A))	Case a) kNN classifier Case b) min distance classifier	(Case a) ε_t:0,01% (k=1) (with $P(\omega_1)=P(\omega_2)=0.5$) (Case b) ε_t:0,87% (N=4) (with $P(\omega_1)=P(\omega_2)=0.5$)
R. Sabourin and J.-P Drouhard [282]	Direction based (Probability Density Function – PDF)	**Training**) 400 (S) **Test**) 400 (S)	NN	ε_t: 4,07% (with $P(\omega_1)=P(\omega_2)=0.5$)
R. Sabourin et al. [283]	Shape Matrices	**FD**) 800 (S) (from 20(A))	Pattern Matching	ε_t: 0,84%
C. Santos et al. [295]	Graphometric based	**Test**) 300 (G), 600 (F)	Euclidean distance + NN (MLP)	**FRR**: 10,33% **FAR**:4,41% (RD), 1,67% (SP), 15,67% (sim. forgeries)
K. Ueda [318]	Pattern Image	**Test**) 1000 (G), 1000 (F)	Pattern Matching	**EER**: 9,1%
X.-H. Xiao and G. Leedham [350]	Direction based, grid based	**Training**) Genuine samples only (Case 1), Genuine samples + artificial forgeries (Case 2) **Test**) 350 (G), 158 (SK), 230(RF)	NN (MLP)	**FRR**: 10.6%, **FAR**: 38.9% (SK) (Case 1) **FRR**: 9.2%, **FAR**: 17% (SK) (Case 2)

Full Database (FD), Signature (S), Genuine Signatures (G), Forgeries (F), Random Forgeries (RF), Simple Forgeries (SF), Skilled Forgeries (SK), Number of Authors (A)

TABLE VI
PERFORMANCES: ONLINE SYSTEMS

Authors	Main features	Database	Approach	Results
L. Bovino et al. [18]	Position, Velocity, Acceleration	**Training)** 45(G) (3(G) x 15(A)) **Test)** 750(G) (50(G)x15(A)), 750 (F) (50(F)x15(A))	DTW (ME by simple averaging)	**EER**: 0,4%
V. Di Lecce et al. [50]	Shape-based features (segmentation dependent), Velocity	**Training)** 45(G) (3(G) x 15(A)) **Test)** 750 (G) (50(G)x15(A)), 750 (F) (50(F)x15(A))	DTW (ME by majority voting)	**FRR**: 3,2%, **FAR**: 0.55%
K. Huang and H. Yan [129]	Velocity, Pressure	**FD)** 4600 (S)	DTW	**EER**: 4%
J. J. Igarza et al. [130]	Direction of pen movement, …	**FD)** 3750 (G) (25(G)x150(A)), 3750 (F)	HMM	**EER**: 9,253%
A. K. Jain et al. [138]	Velocity, Curvature based.	**FD)** 1232 (S) (from 102 (A))	String matching	**FRR**: 3,3%, **FAR**: 2,7% (common threshoold) **FRR**: 2,8%, **FAR**: 1,6% (writer dependent threshold)
R. S. Kashi et al. [146]	Total signature time duration, X-Y (speed) correlation, RMS speed, Moment-based, direction-based, etc.	**Test)** 542 (G), 325 (F)	HMM	**EER**: 2,5%
J. Lee et al. [166]	Position (geometric extrema), AVE velocity, number of pen-ups, time duration of neg. /pos. velocity, total signing time, direction-based, …	**FD)** 6790 (S) (from 271(A))	NN + DP	**EER**: 0.98%
B. Li et al. [180]	Position, Velocity	**Training)** 405 (G) (5(G) x 81(A)) **Test)** 405 (G) (5(G) x 81(A)), 405(F) (5(F) x 81(A))	PCA, MCA	**EER**: 5,00%
H. Morita et al. [205]	Position, Pressure, Pen Inclination	**Training)** 205 (S) **Test)** 861 (G), 1921 (F)	DTW	**EER**: 3%
D. Muramatsu and T. Matsumoto [213]	Direction of pen movements	**Training)** 165 (G) **Test)** 1683 (G), 3170 (SK)	HMM	**EER**: 2,78%
I. Nakanishi et al. [220]	Wavelet Transform.	**Training)** 20(G) from(4(A)) **Test)** 98 (G) (from 4(A)), 200(F) (from 5(A))	Dynamic Programming	**EER**: 4%
J. Ortega-Garcia et al. [238]	Position, Velocity, Pressure, Pen Inclination (Azimuth), Direction of Pen Movement, …	**Training)** 300(G) (from 50(A)) **Test)** (450 (G) (from 50(A)), 750 (SK) (from 50(A))	HMM	**EER**: 1,21% (global threshold) **EER**: 0,35% (user-specific threshold)
T. Qu et al. [266]	Total signature time, AVE/RMS speed, pressure, direction-based, number of pen-ups/pen downs, …	**Test)** 60 (G), 60 (F)	Membership function	**FRR**: 6,67%, **FAR**: 1,67%
M. M. Shafiei and H. R. Rabiee [299]	AVE Speed, acceleration, pressure, Direction of Pen movement,…	**FD)** 622 (G) (from 69(A)), 1010 (SK)	HMM	**FRR**: 12%, **FAR**: 4%
T. Wessels and C.W. Omlin [333]	Position, Pressure, Direction of pen movements, Pen inclination.	**Training)** 750 (G) (15(G) x 50(A)) **Test)** 750 (G) (15(G) x 50(A))	HMM	**FAR**: 13%
W. S. Wijesoma et al. [335]	RMS / MAX speed, acceleration, Time duration of Positive /Negative Velocity, Pen-down time ratio, Direction-based, …	**Training)** 410(G) (10(G) x 41(A)) **Test)** 820(G) (20(G) x 41(A)), 410 (F) (from 6 (A))	Fuzzy Logic	**EER**: 4,82%
Q. Z. Wu et al. [347]	Fourier transform (cepstrum coefficients)	**Training)** 270(G) (from 27(A)) **Test)** 560 (G) (from 27(A)), 650 (F)	Dynamic similarity measure	**FRR**: 1,4%, **FAR**: 2,8%
Y. Xuhua et al. [354]	Geometric-based, Curvature-based,,…	**Training)** 45(G) (45(G) x 1(A)), 45(F) (from 19 (A)) **Test)** 75(G) (75(G) x 1(A)), 90 (F) (from 19 (A))	Fuzzy Logic	**FRR**: 8,5%, **FAR**: 1,8%
L. Yang et al. [357]	Direction of Pen movements	**FD)** 496 (S) (from 31 (A)	HMM	**FRR**: 1,75%, **FAR**: 4,44%
D.-Y. Yeung et al. [360] (1st Int. Signature Verification Competition)	**Task 1:** Position; **Task 2:** Position, Pen Inclination (azimuth), Pressure,…	**Training)** 800(G)(20(G)x40(A)),800(SK)(20(SK)x40(A)) **Test 1)** 600(G)(10(G)x60(A)), 1200(SK)(20(SK)x60(A)) **Test 2)** 600(G)(10(G)x60(A)), 1200(RF)(20(RF)x60(A))		**(Test 1) EER:** 2,84% (Task 1), **EER:** 2,89% (Task 2) **(Test 2) EER:** 2,79% (Task 1), **EER:** 2,51% (Task 2)
H.S. Yoon et al. [362]	Position	**Training)** 1500 (S) ((15 (S) x 100 (A)) **Test)** 500(S) (5 (S) x 100 (A))	HMM	**EER**: 2,2%
K. Zhang et al. [371]	Geometric-based, Curvature–based	**FD)** 306 (G), 302 (F)	Mahalanobis distance, Euclidean Distance,DTW	**FRR**: 5,8%, **FAR**: 0%
M. Zou et al. [379]	Speed, Pressure, Direction-based, Fourier transform	**FD)** 1000 (G), 10000 (F)	Membership function	**FRR**: 11,30%, **FAR**: 2,00%

Full Database (FD), Signature (S), Genuine Signatures (G), Forgeries (F), Random Forgeries (RF), Simple Forgeries (SF), Skilled Forgeries (SK), Number of Authors (A)