Neural Computing and Applications
https://doi.org/10.1007/s00521-021-05854-6

S.I.: MACHINE LEARNING APPLICATIONS FOR SECURITY

Off-line signature verification using elementary combinations of directional codes from boundary pixels

Md Ajij[1] · Sanjoy Pratihar[2] · Soumya Ranjan Nayak[3] · Thomas Hanne[4] · Diptendu Sinha Roy[1]

Received: 6 December 2020 / Accepted: 19 February 2021

Abstract
Verifying the genuineness of official documents, such as bank checks, certificates, contract forms, bonds, etc., remains a challenging task when it comes to accuracy and robustness. Here, the genuineness is related to the degree of match of the signature contained in the documents relating to the original signatures of the authorized person. Signatures of authorized persons are considered known in advance. In this paper, a novel feature set is introduced based on quasi-straightness of boundary pixel runs for signature verification. We extract the quasi-straight line segments using elementary combinations of the directional codes from the signature boundary pixels and subsequently we obtain the feature set from various quasi-straight line classes. The quasi-straight line segments provide a blending of straightness and small curvatures resulting in a robust feature set for the verification of signatures. We have used Support Vector Machine (SVM) for classification and have shown results on standard signature datasets like CEDAR (Center of Excellence for Document Analysis and Recognition) and GPDS-100 (Grupo de Procesado Digital de la Senal). The results establish how the proposed method outperforms the existing state of the art.

Keywords Signature verification · Quasi-straightness · Biometrics · Person identification

1 Introduction

Today, recognition of persons by machines is an active area of research in which biometrics plays an essential role in the recognition models. The developed models are generally based on two common biometric feature types, *physical* features, such as face, fingerprint, retina, etc. [14, 23, 32], and *behavioral* features, such as voice and handwriting [22, 42]. Signature is considered as human behavioral characteristic with which individuals can be uniquely identified. Therefore, when it comes to security and fraud prevention, the signature feature can be used to design authentication systems. Bank checks, contract documents, certificates, for example, are often faked and claimed to be an original. Thus, for the verification of this type of document, we should have prior knowledge about the original signers and their original signature style. Therefore, to investigate the genuineness of a signed document, an automated signature verification can be applied. Here, we consider that we have prior knowledge of the signers and a ready-made dataset with genuine signatures of the signers (for the training of the recognition model).

✉ Thomas Hanne
thomas.hanne@fhnw.ch

Md Ajij
mdajij@nitm.ac.in

Sanjoy Pratihar
sanjoy@iiitkalyani.ac.in

Soumya Ranjan Nayak
nayak.soumya17@gmail.com

Diptendu Sinha Roy
diptendu.sr@nitm.ac.in

[1] Department of Computer Science and Engineering, National Institute of Technology Meghalaya, Shillong, India

[2] Department of Computer Science and Engineering, Indian Institute of Information Technology Kalyani, Kalyani, India

[3] Amity School of Engineering and Technology, Amity University Uttar Pradesh, Noida, India

[4] School of Business, FHNW University of Applied Sciences and Arts Northwestern Switzerland, Riggenbachstrasse 16, CH – 4600 Olten, Switzerland

Published online: 18 March 2021

There are two types of approaches to automating signature verification: *online verification* [1, 11, 12, 17, 41] and *off-line verification* [10, 13, 18, 33, 35, 44]. Off-line signature verification is considered more challenging than online verification, because dynamic information, such as pen-tip pressure, velocity, and acceleration of the pen-tip, is not available in case of off-line signature images. On the contrary, the special arrangements for the acquisition of the signatures make the online method unsuitable in practice on several occasions. Here, too, we have to go offline to verify the genuineness of existing legal documents or papers.

This paper presents a novel feature set for off-line signature verification. The objective is to detect a faked signature in relation to a particular signer if we have a ready dataset of genuine and a sample of faked signatures of the signer. There are three basic types of forgeries, namely *random forgeries*, *simple forgeries*, and *skilled forgeries*. For the first two types, the faked signature is created without knowing the name, signature shape, etc., or they are not done skillfully. But, in the case of skilled forgeries, the creator of the faked signature is assumed to be an expert in imitating the signature shape and style, and the genuine signature style is known to the imitator. It is obvious that skilled forgery detection is more challenging in the absence of dynamic features.

Related works and motivation: The performance of a verification model depends on the set of features being used for the model. A lot of work has been done related to off-line signature verification which uses various types of feature sets for the working of the model. In most of the works, the features are topology, geometric information, gradient, structural information, and concavity bases [13, 26, 28, 33, 39]. For example, Ferrer et al. [13] proposed a method that uses a set of geometric features specified in the description of the signature envelope and the distributions of the strokes. Subsequently, hidden Markov model, support vector machine and Euclidean distance classifier were used for the verification process. ZulNarnain et al. [52], in a recent work, have introduced a signature verification scheme based on geometric features like side, angle, and perimeter of the triangles which are derived after triangulation of a signature image. For the classification, they used the Euclidean classifier and voting-based classifier. Some works are reported which are based on gray value distribution [21, 24, 48], directional run of pixels [5, 35, 36], pixel surroundings [31] and curvature related features [18]. Also, graphometric feature-based works are available in the literature [4]. In [29], the authors proposed a shape feature called *chord moment* to analyze upper and lower signature envelopes. For signature verification, a support vector machine (SVM) was used with the chord moment-based features. Frequently, in a model, multiple features have been used in combination to improve the classification accuracy of the model. For example, in [35], along with the directional feature, moment information and gray value distribution have been used. The authors have used 16-directional feature obtained from the distribution of pixels in the thinned signature strokes. A combination of different types of features makes the feature extraction part costly. Clearly, the computation of moment information along with the 16-directional feature is computationally costly considering the model to be used for real-time applications. In a very recent work [44], proposed by Serdouk et al., the directional distribution is not the sole feature extraction policy. Here, a directional distribution-based longest run feature is combined with gradient local binary patterns (GLBP) to strengthen the feature set where the longest runs have been considered in horizontal, vertical, and two major diagonal directions. So, they have used a combination of topological and gradient features. As a topological feature, the *longest run of pixels* has been used. Gradient information is extracted using Local Binary Pattern (GLBP) in the neighborhood. Computing GLBP at each pixel of the signature image can be considered costly. Serdouk et al. proposed a verification system which is based on the Artificial Immune Recognition System (AIRS). A template-based verification scheme was also presented [50]. The method they provide is based on encoding the geometric structure of signatures using grid templates.

We also note that many works in the state-of-the-art use ensembles of multiple classifiers to produce best results. For example, Ooi et al. [37], in a recent work, presented a framework based on Discrete Radon Transform (DRT), Principal Component Analysis (PCA) and a Probabilistic Neural Network (PNN) to identify forgeries from genuine signatures. But in applications, the designed hardware device should act quickly for classification and decision making. A summary of the existing methods and their classification techniques is given in Table 1.

Though several methods or recognition models have been developed, the results from the existing methods corroborate that there is still plenty of scope for improvement in terms of accuracy and robustness. In addition, there is scope to propose a strong feature set that can collaborate with a low-complexity classifier to emerge as a better performer. It will be an additional advantage if the feature set can be easily extracted from the signature images. In this paper, we have proposed a novel set of features from the quasi-straight digital curve segments defining the signature strokes. The following sections describe the proposed method and experimental results in detail. The contributions in this paper are mentioned below.

Table 1 Working policies of the existing methods and the associated classifiers (listed chronologically)

Method	Features	Classifier
Kalera et al. 2004 [26]	Gradient, structural & concavity	Bayes classifier
Lv et al. 2005 [35]	Directional features	SVM
Justino et al. 2005 [25]	Pixel density, gravity center, Curvature	SVM & HMM
Hanmandlu et al. 2005 [21]	Angular distribution of pixels	Takagi-Sugeno (TS) model
Ferrer et al. 2005 [13]	Geometric features	HMM, SVM
Chen-Srihari 2006 [10]	Graph matching	TPS mapping
Larkins-Mayo 2008 [33]	Gradient direction	Similarity score heuristic
Ruiz-del Solar et al. 2008 [46]	Local interest points	Bayes classifier
Vargas et al. 2008 [48]	High pressure points& Polar distribution	k-NN and PNN
Nguyen et al. 2010 [36]	Direction feature and gradient	SVM
Kumar et al. 2010 [30]	Morphological features	SVM
Bertolini et al. 2010 [4]	Graphometric features, curvature	Genetic algorithm
Kumar et al. 2012 [31]	Pixel surroundedness	RBF-SVM, MLP
Hamadene et al. 2012 [20]	Contourlet transform	SVM
Kovari-Charaf, 2013 [28]	Height, width, area, etc.	Probabilistic model
Jiang et al. 2013 [24]	GLBP and Improved GLBP	SVM
Kumar-Puhan, 2014 [29]	Upper and lower Envelope; chord moments	SVM
Guerbai et al. 2015 [18]	Curvelet transform	RBF-SVM, MLP
Pham et al. 2015 [39]	Geometry-based features.	Likelihood ratio
Serdouk et al. 2016 [44]	Gradient Local Binary Patterns (GLBP) and LRF	k-NN
Pal et al. 2016 [38]	Uniform Local Binary Patterns (ULBP)	Nearest Neighbor
Loka et al. 2017 [34]	Long range correlation (LRC)	SVM
Zois et al. 2019 [51]	Lattice arrangements and Pixel distribution	Decision tree
Sharif et al. 2020 [45]	Local pixel distribution	GA, SVM
Batool et al. 2020 [3]	GLCM, geometric features	SVM
Proposed	Quasi-straight line segments	SVM

1. We used the idea of quasi-straightness of the pixel runs along the edges defining the signature boundaries. The orientations of the quasi-straight edge segments lead to the categorization of the edge segments based on the *singular* and *non-singular* directions. The categories or the various classes gave us a vector of features. The idea of quasi-straightness enabled us to catch longer edge segments with bends up to some extent.
2. In the proposed method, feature extraction is very fast and robust as it computes the quasi-straight edge segments from the edge contour, using Freeman's 8-N chain code. The complexity is linear with the number of edge pixels in the signature boundary.
3. The method is invariant to the presence of noises like dots, blobs, or small strokes in the signature images.
4. From results on standard signature datasets, such as CEDAR (Center of Excellence for Document Analysis and Recognition) and GPDS-100 (Grupo de Procesado Digital de la Senal), it is perceived that the feature set selected by us outperforms the existing state of the art.

2 Proposed method

The directional distribution of edge pixels is often considered an important feature. Here, in this paper, we proposed a set of features using the run of pixels along the signature edge boundary. To do so, we introduced classes of almost straight line segments following some straightness criteria. In the beginning, the set of edge pixels E, coming from the signature boundary, is obtained by

applying a 3×3 filter on the binarized signature image [15]. We make the boundary edge 1-pixel thick, through the removal of redundant pixels, using a morphological thinning procedure [15]. For binarization, we used the method given in [43]. The edge boundary for a sample signature image (cropped and zoomed portion) is shown in Fig. 1. The edge pixels coming from E can be understood as a set of digital straight line segments. The neighbors of a pixel are represented using the values coming from the range $\{0, 1, 2, \ldots, 7\}$ (considering 8-N). The pixel in the east with respect to the center pixel is represented by 0 and the other pixels by 1, 2, 3, 4, 5, 6, and 7 in counter clockwise order. So, the southeast pixel gets the code 7. This scheme is referred to as Freeman's chain code. The straight-line segments are represented as a sequence of pixels which follow some regularity properties (in terms of the chain code values) [27, 40]. A curve segment is said to be *digitally straight* if and only if there are at most two values, differing by ± 1 mod 8, in the chain code of the segment, and for one of these two values, the run-length must be 1 (Property–1). The code always with run-length 1 is referred to as *singular code (s)* and the code other than the singular code is referred to as *non-singular code (n)*. Also, the runs of the non-singular code n can have only two lengths, which are consecutive integers (Property–2). Here, only Property-1 was used for the detection of the straight line segments and subsequent feature extraction. Henceforth, as we only use Property–1, the extracted straight line segments will be referred to as *quasi-straight line segments*. We considered twelve quasi-straight line classes, using all possible combinations of singular and non-singular codes. The features used in our proposed method are based on the distribution of the boundary edge pixels among the twelve different quasi-straight line classes. The flowchart shown in Fig. 2 represents the proposed method.

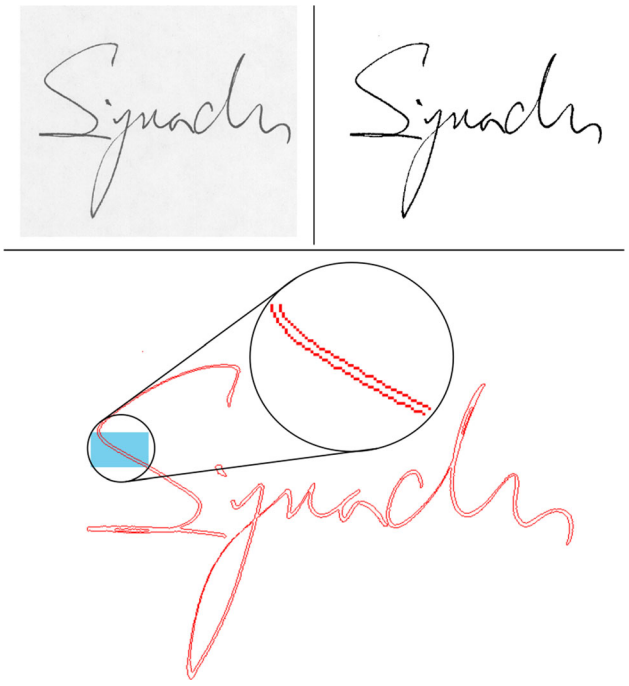

Fig. 1 Sample signature from CEDAR dataset, the binarized image (top row); the set of boundary edge pixels, E (shown as cropped and zoomed for clarity; bottom row)

2.1 Quasi-straight line segment classes

Twelve different classes $C_1, C_2, C_3, \ldots, C_{12}$ of quasi-straight line segments are considered in connection with Property-1 of digital straightness. Class-wise singular (s) and non-singular (n) code values are: C_1: $n = 0$, $s = \epsilon$; C_2: $n = 0$, $s = 1$; C_3: $n = 1$, $s = 0$; C_4: $n = 1$, $s = \epsilon$; C_5: $n = 1$, $s = 2$; C_6: $n = 2$, $s = 1$; C_7: $n = 2$, $s = \epsilon$; C_8: $n = 2$, $s = 3$; C_9: $n = 3$, $s = 2$; C_{10}: $n = 3$, $s = \epsilon$; C_{11}: $n = 3$, $s = 4$; C_{12}: $n = 4$, $s = 3$. Figure 4 demonstrates the classes concerning an example digital curve. Here, the code value ϵ denotes that the code is absent or not defined. In understanding the quasi-straight line segments present in the signature stroke boundaries, no restriction was applied on the run lengths of the non-singular code (n). The number of classes could be further increased considering the variations in non-singular code run length, but it would be over-sensitive to detect the genuine signatures.

The method used for the detection of quasi-straight line segments is given in Algorithm 1. The algorithm takes the edge map E, non-singular code n and singular code s as input. It selects an initial run of two consecutive pixels p_1 and p_2, where the direction code from p_1 to p_2 is n, i.e., $\text{dir}(p_1,p_2) = n$. Then, this run, initiated from the seed (p_1, p_2), extends along the curve, if possible, in two prospective directions n and $(n + 4)$ mod 8 (the non-singular direction n and its opposite direction). The singular direction s is an important part of this extension which is another input to the algorithm. To extend the current run, the Procedure Extend-Segment is invoked as shown in Line 6 and Line 7 of Algorithm 1. The maximal quasi-straight line segment is finally reported, which started from the seed (p_1, p_2) (see Fig. 3). The process is continued to report all quasi-straight line segments that suit non-singular code n and singular code s, thereby giving a set (class) of quasi-straight line segments. Different sets (classes) are reported by varying the code values n and s. The algorithm shown here considers that $s \neq \epsilon$. In case of $s = \epsilon$ (e.g., C_1: $n = 0$, $s = \epsilon$), the segment from the seed (p_1, p_2) is extended in direction n and $(n + 4)$ mod 8 just by checking the neighborhood of the pixels. The algorithm halts at a point if either its neighbor is not in direction n (or $(n + 4)$ mod 8) or there exists no unvisited neighbor pixel. It must be noted that the Procedure Extend-Segment returns *Status*= TRUE only if the singular code s appears at

least once in the run when $s \neq \epsilon$. In our proposed method, a quasi-straight line segment is considered for inclusion into a class if its length is at least a predefined threshold l (see Line 8 of Algorithm 1). The first-hand selection of l is made on the basis of experimentations so that very small segments are removed from consideration. Necessary discussions are provided in Sect. 3.3.

Algorithm 1: Detect-Quasi-Straight-Segments

Input: Thin boundary edge E, non-singular code n, singular code s.
Output: The set of quasi-straight segments, $Q = \{q_1, q_2, q_3, \ldots, q_k\}$ made of n and s.

1 $Q \leftarrow \{\phi\}$, $i \leftarrow 1$, $q_i \leftarrow \{\phi\}$
2 *Continue* $\leftarrow$ FALSE, VISITED $\leftarrow \{\phi\}$
3 **if** $\exists(p_1, p_2) \in E$ *s.t.* $dir(p_1, p_2) = n$ AND $\{p_1, p_2\} \notin$ VISITED **then**
4 *Continue* $\leftarrow$ TRUE, $q_i \leftarrow \{p_1, p_2\}$, VISITED $\leftarrow \{p_1, p_2\}$
5 **while** *Continue* **do**
6 $q_i \leftarrow$ Extend-Segment(p_2, n, s, q_i, VISITED)
7 $q_i \leftarrow$ Extend-Segment(p_1, $(n+4) \mod 8$, $(s+4) \mod 8$, q_i, VISITED)
8 **if** $|q_i| \geq l$ **then**
9 $Q \leftarrow Q \cup \{q_i\}$, $i \leftarrow i+1$, $q_i \leftarrow \{\phi\}$
10 Search for the next seed-pair $(p_1, p_2) \in E$
11 **if** $\exists(p_1, p_2)$ *s.t.* $dir(p_1, p_2) = n$ AND $\{p_1, p_2\} \notin$ VISITED **then**
12 $q_i \leftarrow \{p_1, p_2\}$, VISITED $\leftarrow \{p_1, p_2\}$
13 **else**
14 *Continue* $\leftarrow$ FALSE
15 **return** Q

Procedure Extend-Segment(p, n, s, q_i, VISITED)

1 *Cond* $\leftarrow$ TRUE, $d \leftarrow n$
2 **while** *Cond* **do**
3 **if** $\exists p'$ *in* 8-*N of* p AND $p' \notin$ VISITED **then**
4 **if** $d = n$ AND $(dir(p, p') = n$ OR $dir(p, p') = s)$ **then**
5 **if** $dir(p, p') = n$ **then**
6 $p \leftarrow p'$, $q_i \leftarrow q_i \cup \{p\}$, VISITED $\leftarrow \{p\}$
7 $d \leftarrow n$
8 **else**
9 $p \leftarrow p'$, $q_i \leftarrow q_i \cup \{p\}$, VISITED $\leftarrow \{p\}$
10 $d \leftarrow s$
11 **else if** $d = s$ **and** $dir(p, p') = n$ **then**
12 $p \leftarrow p'$, $q_i \leftarrow q_i \cup \{p\}$, VISITED $\leftarrow \{p\}$
13 $d \leftarrow n$
14 **else**
15 *Cond* $\leftarrow$ FALSE
16 **else**
17 *Cond* $\leftarrow$ FALSE
18 **return** q_i

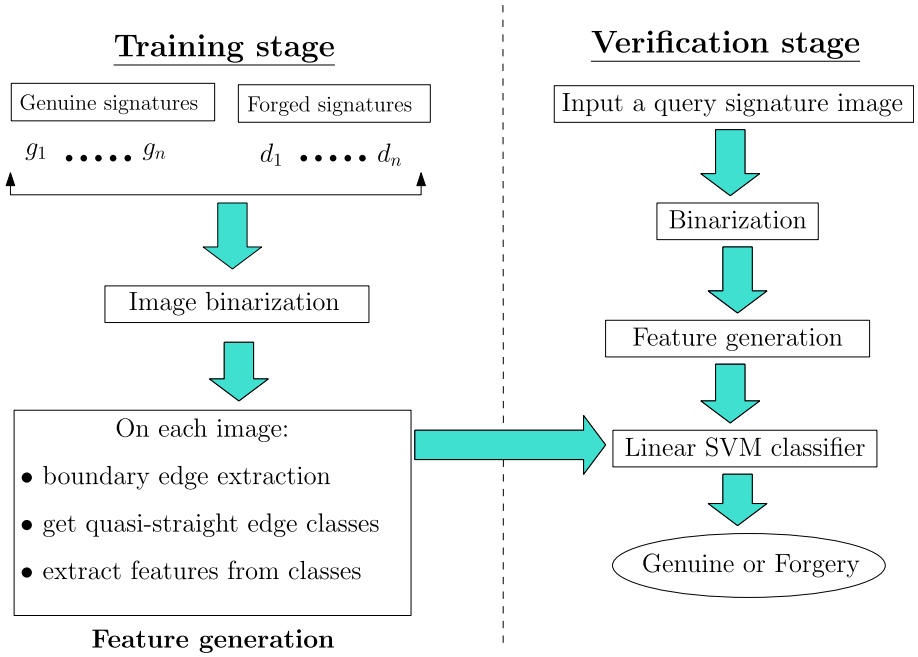

Fig. 2 Flowchart showing the proposed scheme for signature verification

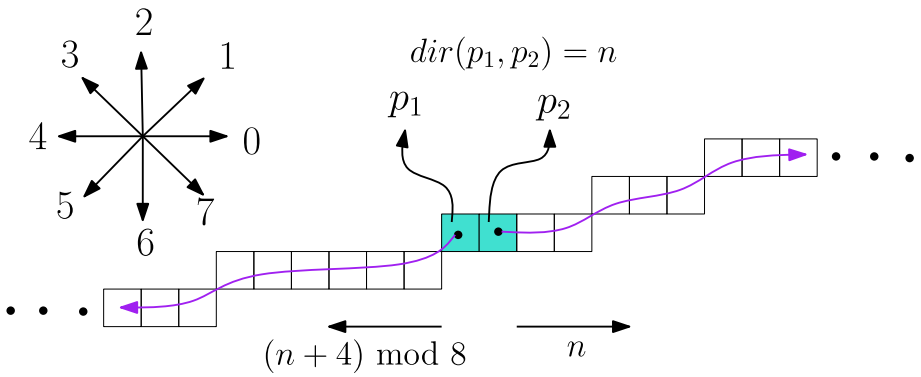

Fig. 3 Extending the segment from a seed (p_1, p_2) in direction n and $(n+4)$ mod 8

2.2 Features from the classes

Of each of the twelve different quasi-straight line classes, we have the following three features.

- The number of quasi-straight line segments in the class C_i, denoted as n_i.
- The pixel density of the class C_i, i.e., $\frac{p_i}{P}$, where p_i is the number of edge pixels in the class and P is total edge pixels from the signature boundary E.
- Average edge length, i.e., $\frac{p_i}{n_i}$ (in terms of pixels) in the class.

Hence, from the twelve classes of line segments, we have the following 36 feature values. So, from this part, we have a feature vector $\langle f_1, f_2, f_3, \ldots, f_{36} \rangle$, directly using the distribution of edge pixels among classes. Here, as an example, the distribution of edge pixels in class C_5 and class C_6 are shown in Fig. 5 with respect to the signature image shown in Fig. 1.

Further, the count of common pixels (c_p) in two neighboring classes C_i and C_j where $j = (i+1)$ mod 12 is considered as a feature of C_i. The existence of common pixels in two neighboring classes defines the smoothness of the boundary curves. We consider the ratio $\frac{c_p}{P}$ for every class and this adds twelve more features to the feature vector, i.e., we get $\langle f_{37}, f_{38}, f_{39}, \ldots, f_{48} \rangle$.

In addition to these 48 features, we added 30 more features. The rectangular signature area is first divided into six equal-sized rectangular regions $R_1, R_2, \ldots, R_6$, as shown in Fig. 6. Then, we ask the two questions as given below.

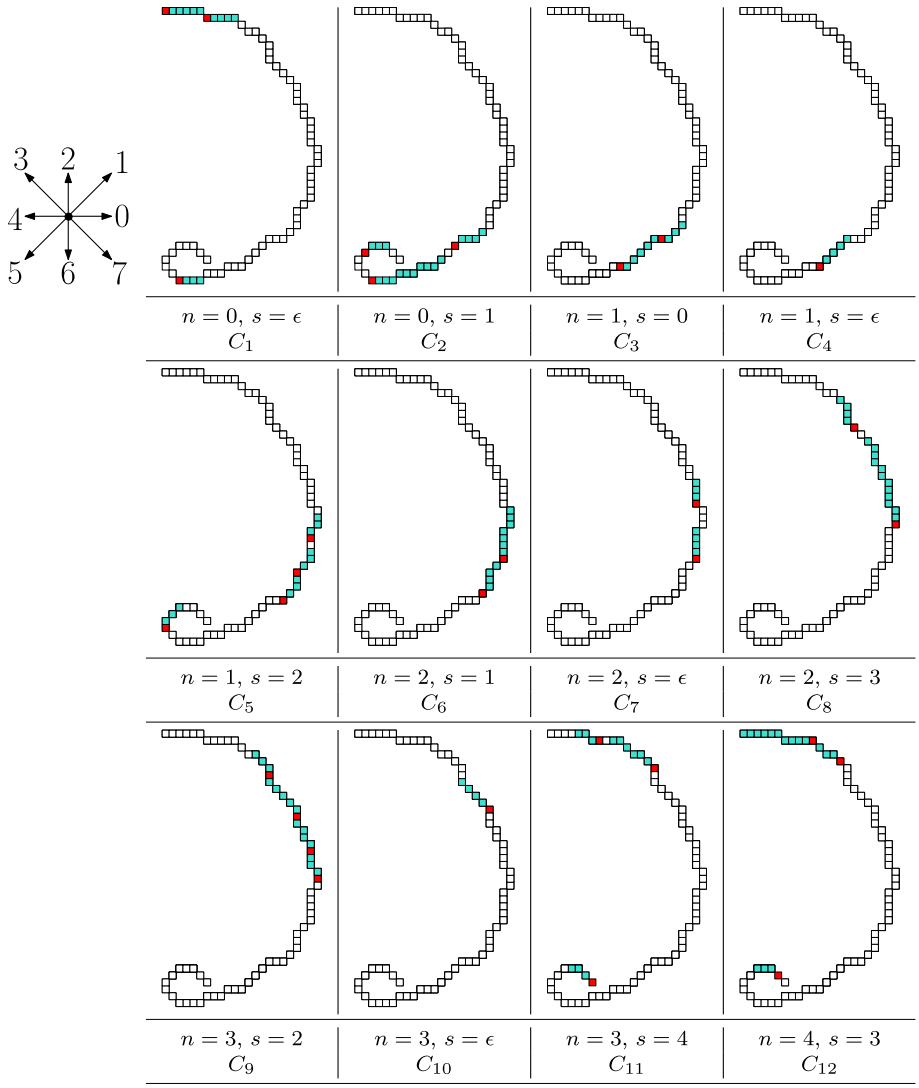

Fig. 4 The classes of quasi-straight line segments for a digital curve

1. Considering a region R_i, which class C_j has maximum contribution in R_i, i.e., which class is the leader in a given region?
2. Given the class C_j, in which region does it have the maximum contribution and what is the contribution (pixel density)?

These pieces of information help us to measure the degree of presence of the classes within the signature image area. The first question provides us with 6 values, i.e., the six class numbers, which are region-wise leaders. On the other hand, the second question provides us with 24 values. If the class C_i (one of the 12 classes) is the leader in region R_j $(j = 1, 2, \ldots, 6)$, we have feature values j and $\frac{r_{ij}}{P}$, where, r_{ij} is the count of pixels in region R_j concerning class C_i. Hence, from this part, we obtain a total of 30 feature values, i.e., we have the feature vector, $\langle f_{49}, f_{50}, f_{51}, \ldots, f_{78} \rangle$.

2.3 Sample feature vector

We have a feature vector of length 78 for every signature image. As an example, the extracted feature vector for the signature image shown in Fig. 1 is presented in Table 2. The first three columns represent the feature values $\langle f_1, f_2, f_3, \ldots, f_{36} \rangle$ providing the number of quasi-straight line segments (n_i), pixel density (p_i/P), and average edge length (p_i/n_i) of all the twelve classes. The fourth column lists 12 values corresponding to 12 classes giving common pixel densities (c_p/P) with the neighboring classes. The fifth column shows 12 values (one against each class), mentioning the region number where the class C_i has a maximum contribution, denoted by m_{C_i}. The sixth column contains another 12 values showing the class-wise pixel densities indicated by r_{ij}/P (for all $i = 1, 2, \ldots, 12$, as discussed in Sect. 2.2). The last column shows which class

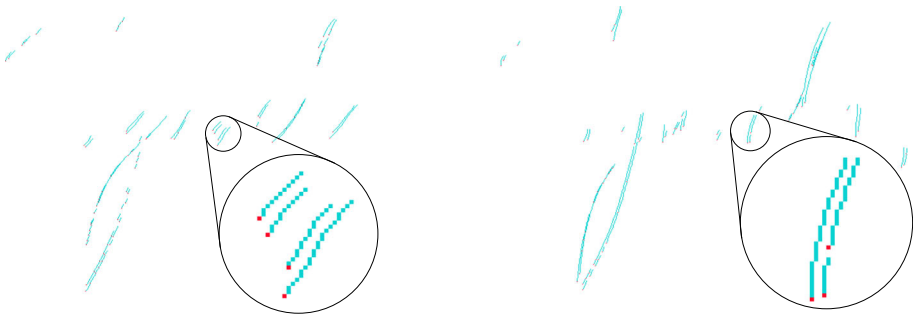

Fig. 5 Edge pixels in Class C_5, $n = 1$, $s = 2$ (left) and Class C_6, $n = 2$, $s = 1$ (right) for the signature image shown in Fig. 1

C_i has a maximum contribution in a region R_j i.e., the region-wise leaders, denoted as l_{R_j}. Here, for example, C_6 is the leader in R_2, R_3, R_4, R_5; C_3 is the leader in R_6; C_{11} is the leader in R_1. We obtain 6 values from this part. Thereby, in total, we have a 78-dimensional feature vector. Representative comparisons of the feature values representing objects from the same class, and objects from different classes are shown in Figs. 7, 8, 9, 10, 11, and 12. The plots clearly show that our features can be used to differentiate the genuine signatures from the faked.

3 Experiments

Our presented signature verification scheme is signer dependent. For performance evaluation, we have carried out the tests for each signer individually. Also, we have used standard available performance measurement methods. One of them is, *False Rejection Rate (FRR)*, is defined by the percentage of genuine signatures rejected by the system. The other is the *False Acceptance Rate (FAR)*, which gives the percentage of faked signatures accepted as genuine. Also, FRR and FAR are combined to report the total verification error and termed as *Average Error Rate (AER)*. The AER is a simple average of FAR and FRR when the CEDAR dataset is considered, but for the GPDS-100, the weighted mean of FAR and FRR are taken, because test counts for genuine signatures and faked signatures differ. Effectively, AER reflects the total verification error. For various set-wise results (different training and test sets of images), we have shown an equal error rate (EER) also. Here, EER is computed as the mean (weighted mean in case of GPDS-100) of respective FAR and FRR of the set, which shows the minimum difference in FAR and FRR values. Experiments were conducted on two standard datasets, as discussed in the following sections.

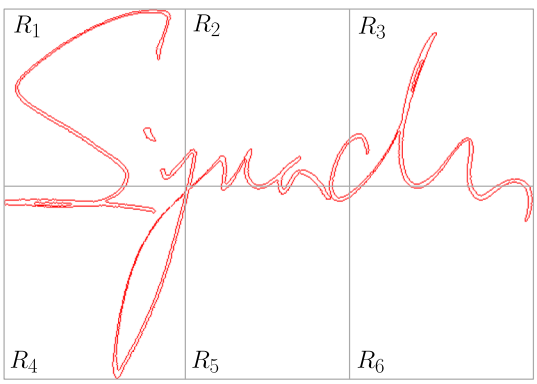

Fig. 6 The six regions R_1, R_2, R_3, R_4, R_5 and R_6 of the signature content area

3.1 Test datasets

The **CEDAR** dataset contains signatures of 55 different signers. For each signer, there are 24 genuine signatures and 24 skillfully faked signatures. So, the CEDAR dataset has a total of 1320 genuine signatures and 1320 skilled forgeries. The signature images have 8-bit gray value levels and they were scanned at 300 dpi. The dataset is created by the *Center of Excellence for Document Analysis and Recognition* [8].

The **GPDS** corpus was prepared by Grupo de Procesado Digital de Senals [16]. The collection consisting of the first

Table 2 Extracted feature set used in the proposed method in relation to the signature image shown in Fig. 1

Class (C_i)	n_i	p_i/P	p_i/n_i	c_p/P	m_{C_i}	r_{ij}/P	l_{R_j}
C_1	36	0.10	9.92	0.06	4	0.07	–
C_2	45	0.16	12.60	0.05	4	0.05	–
C_3	41	0.16	14.63	0.06	1	0.05	R_6
C_4	32	0.08	8.78	0.06	2	0.02	–
C_5	85	0.28	12.09	0.16	4	0.10	–
C_6	75	0.41	19.81	0.13	4	0.15	R_2, R_3, R_4, R_5
C_7	69	0.14	7.62	0.08	3	0.05	–
C_8	53	0.17	11.45	0.03	3	0.06	–
C_9	24	0.06	9.75	0.02	1	0.02	–
C_{10}	11	0.02	7.55	0.02	1	0.01	–
C_{11}	22	0.09	15.18	0.04	1	0.06	R_1
C_{12}	48	0.15	11.52	0.08	4	0.08	–

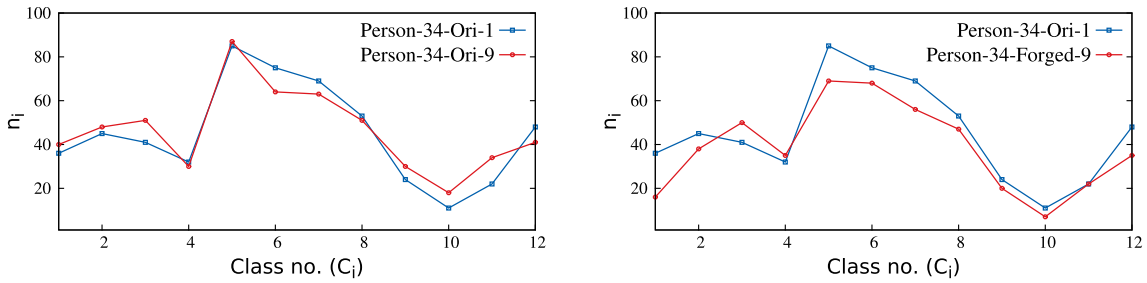

Fig. 7 Comparison of 12 feature values showing n_i class-wise: between original sample-1 and original sample-9 related to Person-34 in CEDAR (left); between original sample-1 and faked sample-9 related to Person-34 in CEDAR (right)

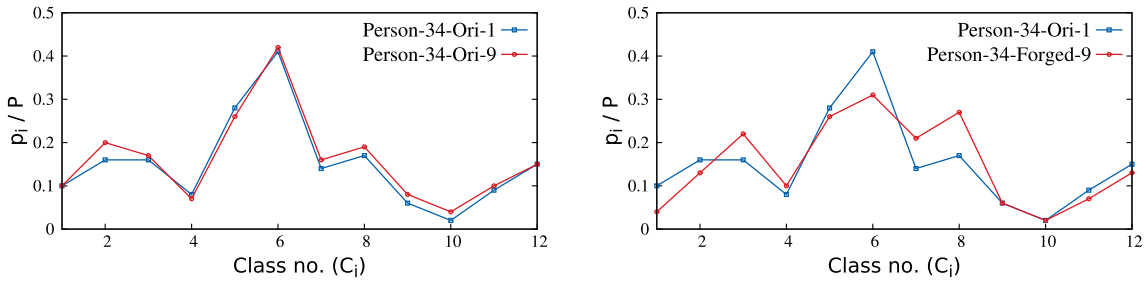

Fig. 8 Comparison of 12 feature values showing p_i/P class-wise: between original sample-1 and original sample-9 related to Person-34 in CEDAR (left); between original sample-1 and faked sample-9 related to Person-34 in CEDAR (right)

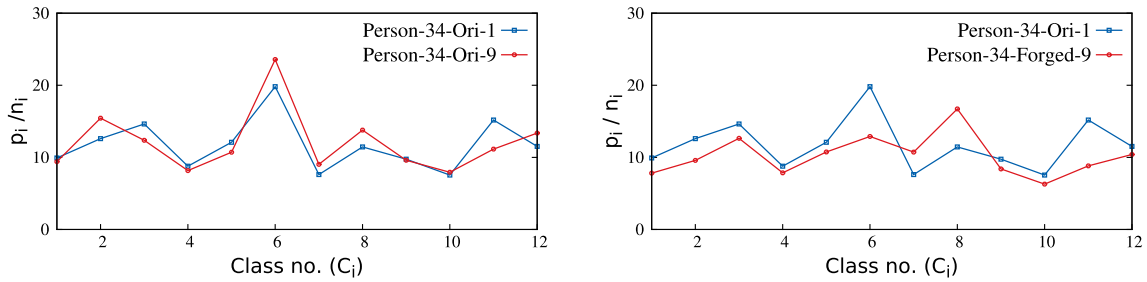

Fig. 9 Comparison of 12 feature values showing p_i/n_i class-wise: between original sample-1 and original sample-9 related to Person-34 in CEDAR (left); between original sample-1 and faked sample-9 related to Person-34 in CEDAR (right)

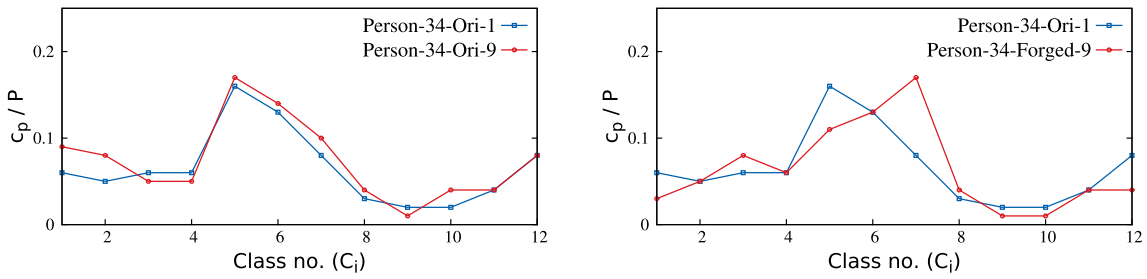

Fig. 10 Comparison of 12 feature values showing c_P/P class-wise: between original sample-1 and original sample-9 related to Person-34 in CEDAR (left); between original sample-1 and faked sample-9 related to Person-34 in CEDAR (right)

100 signers, referred to GPDS-100, was used by us. With respect to each person, there are 24 genuine signatures and 30 skilled forgeries. This gives a total of 2400 genuine signatures and 3000 skilled forgeries. The signature images are obtained as binary images.

Sample signature images, both genuine and faked, from CEDAR and GPDS-100 datasets are shown in Figs. 15 and 16, respectively. In many cases, it is difficult to find out differences between the faked signatures and the genuine signatures of the respective person. Hence, we consider these types of forgeries as skilled forgeries.

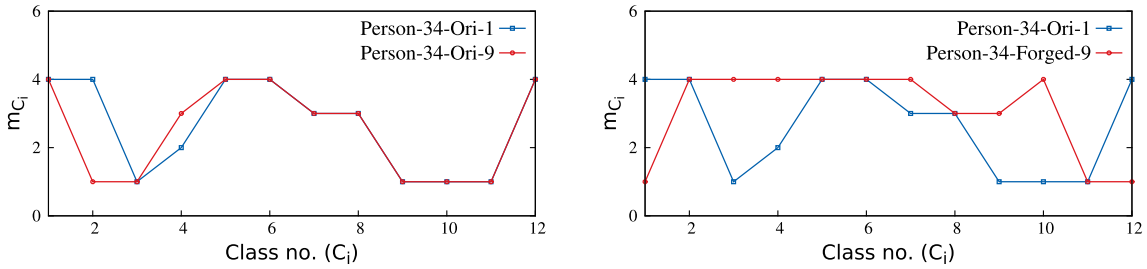

Fig. 11 Comparison of 12 feature values showing m_{C_i} class-wise: between original sample-1 and original sample-9 related to Person-34 in CEDAR (left); between original sample-1 and faked sample-9 related to Person-34 in CEDAR (right)

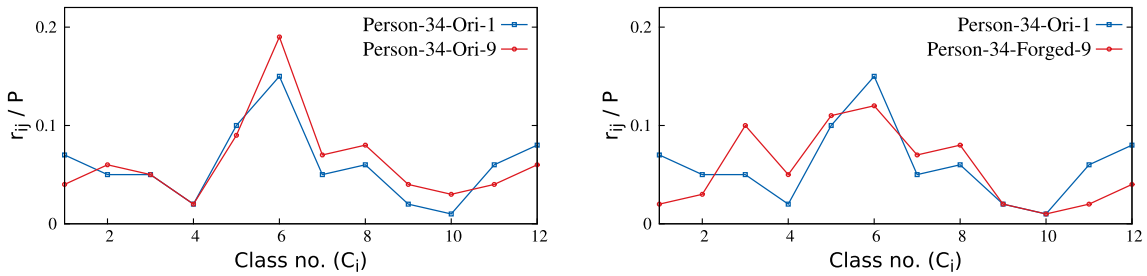

Fig. 12 Comparison of 12 feature values showing r_{ij}/P class-wise: between original sample-1 and original sample-9 related to Person-34 in CEDAR (left); between original sample-1 and faked sample-9 related to Person-34 in CEDAR (right)

3.2 Support vector machine (SVM)

The support vector machine (SVM) is a well-accepted classifier in two-class classification problems. It is observed that in the signature verification problem, the SVM was used extensively [2, 5, 6, 29, 47, 50]. In our proposed method, performance evaluation is done for each individual signer, and our problem is considered a two-class classification problem. We considered *Class-0* and *Class-1* to correspond with the genuine signature class and the faked signature class, respectively.

For both CEDAR and GPDS-100, signatures from the genuine category and signatures from the faked category were taken to train the model. How many signatures are to be taken from each category depends on the training size in use. For example, we refer to this training size as $\langle 16{+}16 \rangle$ when 16 signatures from the genuine category and 16 signatures from the faked category were taken to train the model. Other than the size $\langle 16{+}16 \rangle$, we used training sizes $\langle 12{+}12 \rangle$ and $\langle 8{+}8 \rangle$ for experimentation. We found that the LINEAR kernel shows better results than POLY, RBF, and SIGMOID on our feature set. So, we used the LINEAR kernel SVM for the classification work.

We used 8-fold cross-validation to set the value of parameter C. For every person, a set of 48 random signatures (equal contribution from genuine and faked signatures of that person) was considered, and they were divided into 8 equal groups (6 images per group). During cross-validation, 7 groups, i.e., 42 images were used for training, and one group (6 images) was tested. The selection of the training and testing groups was made in all possible combinations, thereby giving a total of 48 test results (considering all different training samples). Again, the C value was updated/incremented in a loop with a small variation (0.0001), starting at 0.0001, and we checked up to 1. The corresponding C was selected for the person for whom we have reached a maximum accuracy rate during cross-validation. Accuracy means how many times Class-0 objects are reported as Class-0 objects and how many Class-1 objects are reported as Class-1 objects. It must be noted that 48 tests were carried out for every C value in the loop.

3.3 Results and discussions

For the implementation of the system, we have used C++ in Ubuntu 12.04, Kernel Linux 3.2.0-54-generic 64-bit, Intel® Core™ i5-2310 CPU 2.90GHz. SVM class of OpenCV has been used for classification. Our system takes *edge length threshold*, l, as user input, and this is the only parameter used as input. This parameter is used to restrict the small quasi-straight line segments from taking part in the feature extraction phase. A quasi-straight segment is considered if its length is at least l. We experimented with various l values empirically and observed that the system works well for $l = 4$. The average error rates, as found by the proposed method taking various l, for the CEDAR dataset, are shown in Fig. 13.

As an initial setup, for every individual, we had a training size of $\langle 16 + 16 \rangle$, i.e., 16 genuine signatures and 16 faked signatures were used for training, as mentioned in Sect. 3.2. Further, to report the error rates (FAR, FRR, and AER), we carried out the experiments four times by changing the training set randomly (test set consisting of the remaining signatures of that individual). The four types of different random arrangements are referred to as Set-1, Set-2, Set-3, and Set-4. Such a test result for Set-4 from the CEDAR dataset is shown in Table 3.

During each test, the corresponding error rates are recorded, and finally, the resultant FAR, FRR, and AER are reported. Other than the training size $\langle 16{+}16 \rangle$, as mentioned in Sect. 3.2, we have tested the performance of our algorithm on training sizes $\langle 8{+}8 \rangle$ and $\langle 12{+}12 \rangle$ also. The corresponding set-wise results are shown in Tables 4 and 5 with respect to CEDAR and GPDS-100 datasets. The EER values are reported in Tables 4 and 5 against each training size considering the various sets of training-test arrangements.

Comparative results with existing methods are shown in Tables 6 and 7. Only those existing methods that use CEDAR and GPDS-100/160/200/300 datasets to report their results are shown. The corresponding results are taken from the respective papers where the works are reported. First, we will discuss the results we obtained in the CEDAR dataset and then we will discuss GPDS-100. We notice an average error rate below 3% (FAR = 3.35 and FRR = 2.39) for the proposed method when applied to the CEDAR dataset, considering training size $\langle 16{+}16 \rangle$. We now discuss the training policies of the other models, and the corresponding FAR and FRR values. Four methods corresponding to the four best results, as shown in Table 6, other than the proposed method, are considered for the discussion. In their method, Bharathi and Shekar [5] used chain-code (4-directional)-based directional features from the contour of the signature. They have also used SVM as the classifier and obtained FAR = 7.84 and FRR = 9.36. In their model, they used 12 genuine samples of a person, and 108 random forgeries from other persons for training (taking 2 from the remaining other persons; $54 \times 2 = 108$). So, the test size against each person was 12 genuine signatures and 24 forgeries of the same person. Results of the proposed method using training size $\langle 12{+}12 \rangle$ are better than their results. Larkins and Mayo [33] used 16 genuine signatures for training. Then, eight genuine signatures along with 24 forgeries were used for testing in relation to the same person. They used the adaptive feature thresholding (AFT)-based similarity score finding method, and the automatic verification method resulted in FAR = 10.96 and FRR = 8.16. The training policy of these methods is not directly comparable with ours; still, these pieces of information give us an idea about the performance of our proposed method. Kumar and Puhan [29] used the same training size as we, i.e., 16 genuine and 16 forgeries for each person. As a classifier, they also used SVM. Corresponding FAR and FRR values are 5.68 and 6.36,

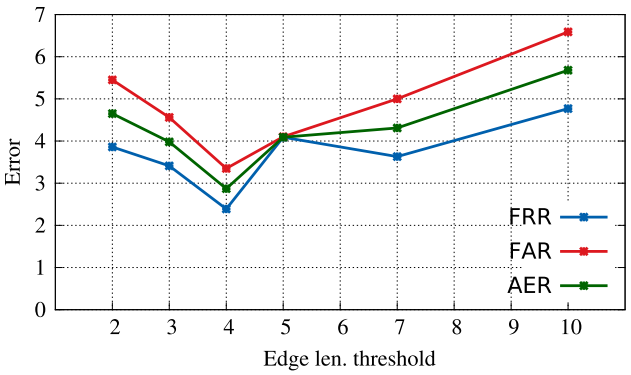

Fig. 13 The error rates (FRR, FAR and AER) are shown with respect to various edge length threshold l when applied on CEDAR dataset

Table 3 Sample test results on CEDAR dataset; FR: false rejection, FA: false acceptance

Persons	#FR	#FA	Persons	#FR	#FA
1	0	1	29	0	1
2	0	0	30	0	0
3	1	0	31	0	1
4	0	0	32	0	0
5	0	0	33	0	0
6	1	1	34	0	0
7	0	0	35	1	0
8	0	1	36	0	0
9	0	0	37	0	0
10	0	1	38	0	1
11	0	0	39	0	0
12	0	1	40	0	0
13	0	0	41	0	0
14	0	0	42	0	0
15	0	0	43	0	0
16	1	1	44	0	0
17	0	0	45	1	0
18	0	0	46	0	1
19	0	0	47	0	0
20	0	0	48	1	1
21	0	0	49	0	0
22	0	1	50	0	0
23	3	2	51	0	0
24	0	0	52	0	0
25	0	0	53	0	0
26	0	0	54	1	0
27	1	1	55	0	0
28	0	0	Total	11	15

Table 4 Set-wise verification results according to the proposed method on CEDAR dataset

Set	Training-size: 8 + 8			Training-size: 12 + 12			Training-size: 16 + 16		
	FAR (%)	*FRR* (%)	*AER* (%)	*FAR* (%)	*FRR* (%)	*AER* (%)	*FAR* (%)	*FRR* (%)	*AER* (%)
1	3.73	2.73	3.23	4.70	2.12	3.41	1.81	2.05	1.93
2	5.00	3.64	4.32	2.88	2.72	2.80	3.86	2.27	3.07
3	4.32	2.39	3.36	4.39	2.57	3.48	4.32	2.73	3.53
4	5.23	2.73	3.98	3.79	2.12	2.96	3.41	2.50	2.96
Mean	4.57	2.87	3.72	3.94	2.38	3.16	3.35	2.39	2.87
EER		3.23			2.80			1.93	

Table 5 Set-wise verification results according to the proposed method on GPDS-100 dataset

Set	Training-size: 8 + 8			Training-size: 12 + 12			Training-size: 16 + 16		
	FAR (%)	*FRR* (%)	*AER* (%)	*FAR* (%)	*FRR* (%)	*AER* (%)	*FAR* (%)	*FRR* (%)	*AER* (%)
1	19.73	13.19	16.97	16.73	10.34	14.16	14.86	7.00	12.0
2	19.50	15.69	17.89	14.45	13.84	14.2	15.29	8.13	12.68
3	21.77	13.38	18.23	15.73	12.67	14.5	15.43	8.00	12.72
4	19.68	13.69	17.15	12.45	12.75	12.57	14.57	8.25	12.28
Mean	20.17	13.99	17.56	14.84	12.40	13.85	15.04	7.85	12.42
EER		17.89			12.57			12.28	

Table 6 Results of the proposed method (considering training size ⟨16+16⟩) on the CEDAR dataset in comparison with other methods

Dataset	Method	*FAR*(%)	*FRR*(%)	*AER*(%)
CEDAR	Surroundedness [31]	08.33	08.33	08.33
	Morphological features [30]	11.23	12.39	11.59
	Curvelet transform [18]	08.25	07.41	07.83
	Gradient, structural and concavity [10]	08.20	07.70	07.90
	Gradient direction [33]	07.42	07.75	07.58
	Chain code histogram [5]	07.84	09.36	08.60
	Chord moments [29]	05.68	06.36	06.02
	GLBP and LRF [44]	04.93	02.12	03.54
	GSC [26]	19.50	22.45	21.50
	Zernike moments [9]	16.30	16.60	16.40
	Proposed	3.35	2.39	2.87

respectively. In their recent work [44], Serdouk et al. used the same training size as we. So, 16 genuine signatures and 16 forgeries are used in the training stage, and the remaining signatures (eight genuine + eight forgeries for CEDAR and eight genuine + 14 forgeries for GPDS-100) are used to test the verification performance. Indeed, we find this method perfectly comparable with our proposed method. It is also noted that the work has achieved a comparable and sometimes better performance than other systems. Now, in comparison with Serdouk et al., we see that our proposed method exhibits better FAR=3.35 (Serdouk et al.: 4.93) and comparable FRR=2.39 (Serdouk et al.: 2.12). Loka et al. [34] used two types of training strategies. In the first strategy, they used genuine and simulated forgeries, whereas in the second, they used genuine and only random forgeries. In the testing phase, the remaining genuine and forgery samples were tested using a binary SVM. The second strategy is found to be

Table 7 Results of the proposed method (considering training size ⟨16+16⟩) on the GPDS-100 dataset in comparison with other methods

Dataset	Method	*FAR*(%)	*FRR*(%)	*AER*(%)
GPDS-100/160/300	Surroundedness [31]	13.76	13.76	13.76
	Curvelet transform [18]	19.40	12.50	15.95
	Gradient direction [33]	14.21	10.48	12.34
	Chain code histogram [5]	09.64	13.16	11.40
	GLBP and LRF [44]	13.16	11.38	12.52
	Geometric features [13]	15.50	16.39	15.94
	High pressure points [48]	14.66	10.01	12.33
	Local interest points [46]	14.20	16.40	15.30
	MDF and the gradient [36]	16.54	13.51	15.03
	DCNN [19]	05.99	19.81	12.9
	Proposed	15.04	7.85	12.42

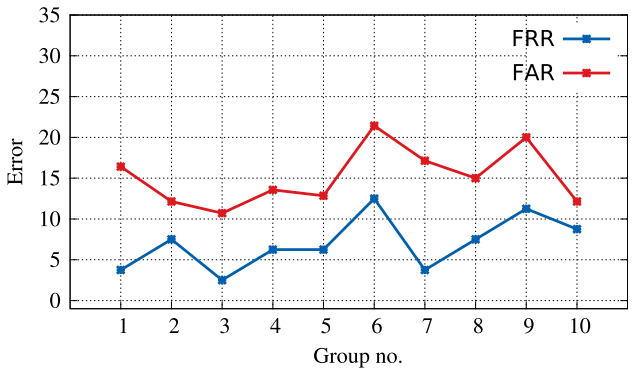

Fig. 14 Group-wise error rates FRR and FAR are plotted with respect to our Set-1 test on GPDS-100 dataset

comparable with our proposed method, as we are also employing random forgeries in training. Taking training size 16, they obtained FRR=6.22 and FAR=5.33. In conclusion, on the CEDAR dataset, the performance of our proposed method is either better than or comparable to other methods.

The GPDS dataset is available in various sizes. GPDS-100 consists of the samples of the first 100 persons, GPDS-160 of the first 160 persons, GPDS-200 of the first 200 persons and GPDS-300 refers to the set for all 300 persons. Our test results shown here refer to the GPDS-100 dataset. For the GPDS-100 dataset, the average error rate according to our method is 12.42, which is comparable to other methods (see Table 7). We obtained this result when the training size ⟨16+16⟩ was used. Serdouk et al. also provided results for GPDS-100, and the corresponding FAR and FRR are reported as 13.16 and 11.38 (AER = 12.52) taking training size = 16, which are comparable to our results. Our proposed method leads to FAR = 15.04 and FRR = 7.85, respectively. Good results are obtained for many individuals (100 signers), but poor results for some persons downgrade the overall average accuracy. We have shown group-wise error rates of GPDS-100 in Fig. 14, where each group contains 10 signers. Group-1 is from signer 1 to 10, Group-2 is from signer 11 to 20, and so on. If we consider the best four groups (40% best results) with a lower average error rate, we have an AER close to 10%.

Fig. 15 Sample signature images (binarized) from CEDAR dataset; genuine signatures in the top row and faked signatures in the bottom row

Fig. 16 Sample signature images (binarized) from GPDS-100 dataset; genuine signatures in the top row and faked signatures in the bottom row

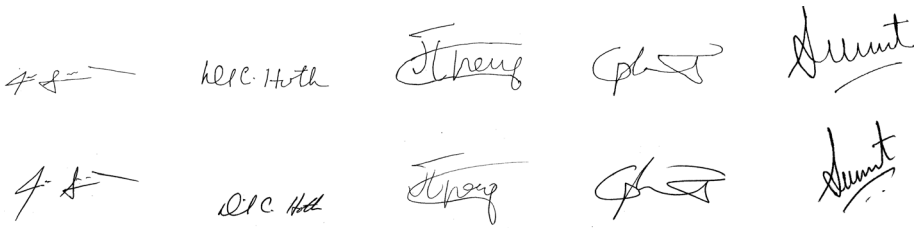

Fig. 17 Sample signature images from MCSFC dataset (top row); genuine CEDAR samples (bottom row)

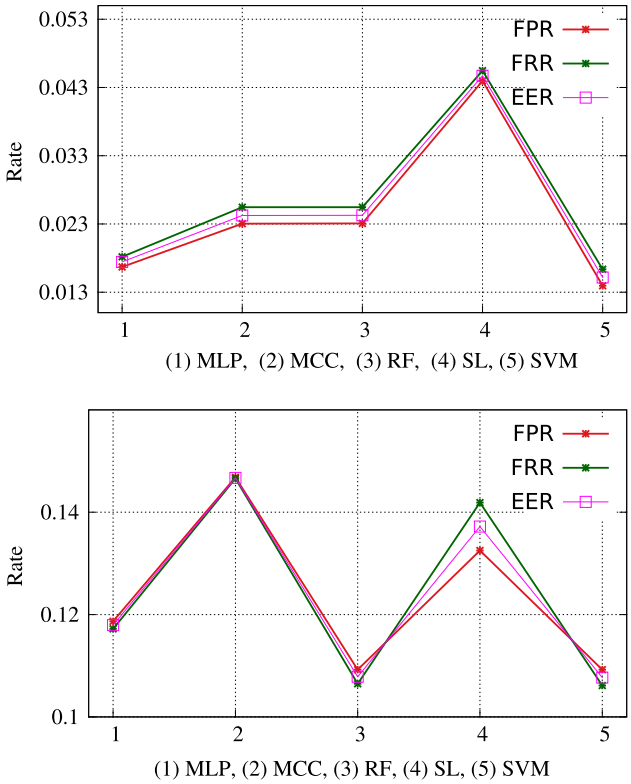

Fig. 18 Error rates FPR, FRR and EER by all classifiers with respect to CEDAR (top) and GPDS (bottom) dataset

Zois et al. reported the same behavior using the GPDS-300 dataset in their recent paper [50].

We notice three other works in the literature that have been published very recently which employ pixel distribution-based features and geometric features. Zois et al. [51] presented a method based on lattice structure arrangements and pixel distribution. They used random training and testing sets and obtained the FAR = 12.35%, FRR = 12.21%, AER = 12.28% for the CEDAR dataset and FAR = 9.11%, FRR = 5.05%, AER = 7.08% for the GPDS. In the work proposed by Sharif et al. [45], the authors have used geometric features and features generated from the study of the local distribution of pixels. They have used a genetic algorithm-based selection of features and then finally SVM for classification work. The training and testing sizes ratio is observed as 70:30 in their

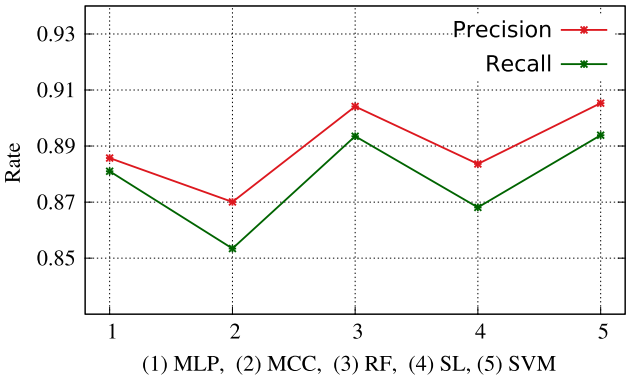

Fig. 19 Precision and Recall values by all classifiers related to the GPDS dataset

experiments, and they obtained the FAR = 4.17%, FRR = 4.17%, AER = 4.17% for the CEDAR dataset and FAR = 6.67%, FRR = 4.16%, AER = 5.42% for the GPDS. Batool et al. [3] presented a method that generates the features by finding the distribution of pixels in regions of the signatures. They have used SVM for classification. The training and testing sizes ratio is noted as 70:30 in their experiments, and they obtained the FAR = 3.34%, FRR = 3.75% for the CEDAR dataset and FAR = 9.17%, FRR = 10.0% for the GPDS.

The proposed method for extracting features is significantly fast. For example, the CPU time required to generate all features for the image shown in Fig. 1 (size = 582×486, #-edge-pixels=3638) is 0.028 Sec (excluding binarization). On average, the classification time for a single image is less than 0.2 Sec (excluding cross-validation tests) which shows the fitness of the feature set to be used in real-time applications.

3.3.1 Other datasets and skilled forgeries

In addition to the two datasets (CEDAR and GPDS-100), we have tested our method in two other datasets. The first of them is the dataset created at the Netherlands Forensic Institute (NFI) [7]. The dataset consists of authentic signatures from 100 newly introduced writers and faked signatures from 33 writers (the writers are NFI employees). In

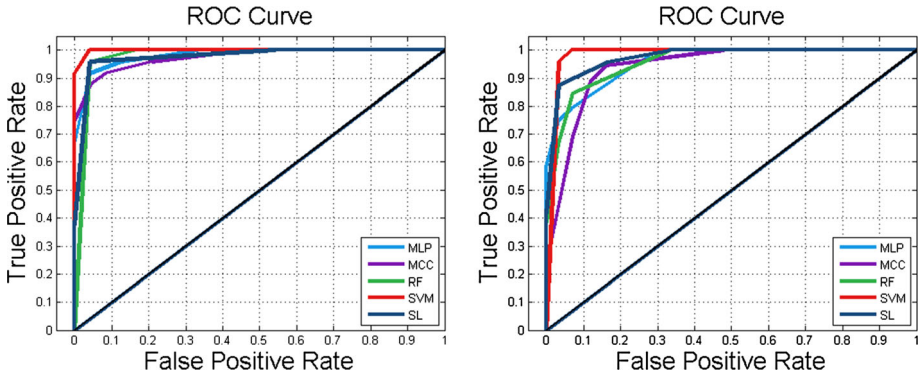

Fig. 20 ROC plots for all classifiers related to the CEDAR (left) and GPDS dataset (right)

total, there are 1953 signatures. The dataset is referred to as SigComp2009.

Another dataset, consisting of faked signatures imitating the genuine signatures in the CEDAR dataset is created by us. A genuine signature is selected for each person, and we imitate the signature style. The copied signatures were written on plain paper and captured by a mobile camera. We refer to this dataset as *mobile captured skilled forgeries of CEDAR genuine signatures* or *MCSFC* in short. For preparing the MCSFC, special care has been taken to ensure that our faked signatures look as good as genuine signatures. Hence, these signatures are highly skilled forgeries. Sample signature images from MCSFC are shown in Fig. 17.

In addition to SVM, we have experimented with four different classification schemes, namely Multi-Layer Perceptron (MLP), Multi-class Classifier (MCC), Random Forest (RF), and Simple Logistic (SL) Regression. Multi-Layer Perceptrons (MLP) are used as a type of neural network, and are employed to perform computational tasks, such as predictive modeling tasks. The Multi-Class Classifier (MCC) is used for classification purposes, and normally consists of more than two classes or outputs. But variants have been designed for binary classification problems as well. The Random Forest (RF) classifier consists of several decision trees on various subsets, and it takes the prediction from each of the trees based on a majority, voting scheme and finally predicts the final prediction output. Simple Logistic (SL) or Logistic Regression

Table 8 Results on scaled down sample images with respect to `Person-12` in CEDAR dataset

Classification method	Scaled-down factor	Training-testing size	Accuracy
MLP	2	10-Fold CV	91.66
	2	80:20	94.7
	3	10-Fold CV	90.08
	3	80:20	92.0
SL	2	10-Fold CV	88.0
	2	80:20	84.0
	3	10-Fold CV	85.08
	3	80:20	82.0
SVM	2	10-Fold CV	93.75
	2	80:20	90.0
	3	10-Fold CV	88.0
	3	80:20	92.2
MCC	2	10-Fold CV	85.42
	2	80:20	90.0
	3	10-Fold CV	75.0
	3	80:20	78.5
RF	2	10-Fold CV	83.33
	2	80:20	90.0
	3	10-Fold CV	80.0
	3	80:20	87.08

is one of the most popular supervised machine learning techniques. In application purposes, it predicts output as probabilistic values in [0, 1] [49].

For MLP, we have the following parameter settings: number of hidden layers $= 40$, learning rate $= 0.3$, momentum $= 0.2$, batch size $= 100$, the number of epochs to train through is 500, and the validation threshold is equal to 20. For MCC, the batch size is set to 100, the classifier model is *logistic regression* with a ridge estimator, and the random width factor is set to 2.0. For RF, the bag sizes are the same as the training set size; the batch size is set to 100, the maximum depth of the trees are unlimited, the number of iterations to be performed is set to 100. For SL, the maximum boosting iterations are set to 500, and the batch size is considered as 100.

To evaluate our proposed method on the SigComp2009 dataset, we have used the training and testing size ratio as 80:20. The detection accuracies are reported as 91.48%, 90.73%, 89.43%, 80.60% and 86.53% for SVM, Random Forest (RF), Multi-Layer Perceptron (MLP), Multi-class classifier (MCC) and Simple Logistic (SL), respectively. The SVM shows the best result, and the proposed method shows an EER $= 8.51\%$ in comparison with 9.15% as reported by the method given by Blankers et al. [7].

A separate test is conducted for testing on MCSFC using the training size $\langle 16{+}16 \rangle$, as mentioned earlier. Three faked signatures from MCSFC are taken in relation to each person for the test. Even if there are maximum similarities with the genuine signatures, we observe notable accuracies in rejecting them as faked. The FAR value with respect to MCSFC is found to be 24.5%, i.e., 75.5% of skilled faked signatures are rejected as faked.

The plots in Fig. 18 show the corresponding FPR, FRR, EER values for CEDAR and GPDS dataset, respectively. The ROC plots concerning all the classification methods are shown in Fig. 20 for both CEDAR and GPDS datasets. Also, the *Precision* and *Recall* values are shown in Fig. 19 with respect to GPDS dataset. We notice that the SVM shows the best results and RF performs almost similarly.

3.3.2 Scale invariance

We tested our proposed feature set for the identification of scaled-down signature images. We used scaling factors of 2 and 3 and created images from the CEDAR dataset. Scaling factor k indicates that both height and width are reduced by a factor k. As representative data, we present the recognition accuracy when all the classification methods are used on `Person-12` of the CEDAR dataset in Table 8. Here, the results related to 10-fold cross-validation and training-testing size ratio 80:20 are shown in the table. For the full dataset, we found that MLP and SVM shows an accuracy percentage just above 90.0 for both 10-fold cross-validation and the training-testing size ratio 80:20 when $k = 2$, which is very promising because of the extracted feature values' normalized nature. Our observation is that when k increases to 3, the accuracy of the detection decreases.

3.3.3 Impactful features

For the CEDAR dataset, we consider one test loop for each of the 55 persons. In one loop there are 16 tests (as there are 24 genuine and 24 faked signatures and we use $\langle 16 + 16 \rangle$ training). The WEKA-attribute-selection method reports the impactful features at the end of each person's test-loop, i.e., after 16 tests. So, 55 times we report the 78 features in descending order of their impact ranks. We have the following parameters settings for SVM: *the number of xval folds to be used when estimating subset accuracy* $= 5$, *the number of attributes to evaluate in parallel* $= 1$, *seed*(Seed to use for randomly generating xval splits)$= 1$, *threshold* $= 0.01$. We observe that the WEKA-attribute selection method picks more frequently the first 36 features, i.e., $\langle f_1, f_2, \ldots, f_{36} \rangle$, than the comparatively impactful features from the set of 78 features. For example, on CEDAR, one random test loop (for all 55 persons) shows the features $\langle f_{17}, f_{18}, f_{43}, f_{26}, f_{19}, f_4, f_7, f_{20}, f_{23}, f_1 \rangle$ as the most impactful 10 features with maximum number of presence in 55 tests.

4 Conclusion

In this paper, we have proposed a novel feature set for verification of signatures based on the distribution of boundary-edge pixels of the signature-image. Previously, in some other works, directional features in terms of the longest pixel run or chain code have been used in combination with additional features. In this work, newly proposed classes of quasi-straight line segments have been used to define the discriminating features. We conducted experiments on standard datasets like CEDAR and GPDS-100. Experimental results corroborate that the proposed feature set shows significantly good results and may be of use in real-time applications. Also, there are scopes to combine with other geometric features like convex hull shape, count and locations of endpoints, count of closed loops, etc., to improve the accuracy level. We must mention that the parameter, edge length threshold, plays a vital role in the process of feature generation and classification. Hence, the automatic selection of the edge length threshold will be worth investigating. This threshold may be related to the resolution of the signature images. Further, considering any two consecutive quasi-straight line segments from the boundary, an investigation may be worth to check

whether their non-singular code values, singular code values, and non-singular run lengths together can perceive the local curvature information when joining the two segments. The nature of curvature and its amount can be important features in the recognition process. It is to be noted that no extra computation is needed to extract the curvature information. This curvature information may also improve the accuracy level of the model.

List of symbols Symbol: Meaning; E: Set of boundary edge pixels; C_i: The i-th ($i = 1 \ldots 12$) class of quasi-straight line segments; n: Non-singular code; s: Singular code; l: The straight edge length threshold; Q: The set of quasi-straight line segments; q_i: The i-th segment in the set Q; p: A pixel in E; $d(p_1, p_2)$: Chain code direction from pixel p_1 to pixel p_2 where p_2 is in 8-N of p_1; f_i: The i-th feature; R_i: The i-th region in the signature area; $i = 1 \ldots 6$; p_i: The number of edge pixels in E belonging to i-th class; P: Total edge pixels in E; n_i: Number of edge segments in the i-th class; c_p: Number of common pixels in two neighboring classes C_i and C_j, $j = (i + 1) \bmod 12$; r_{ij}: Number of pixels in the region R_j concerning class C_i; m_{c_i}: The region number (from 1 to 6) where the class C_i has the maximum contribution; l_{R_j}: The class number (from 1 to 12) which has the maximum presence in region R_j

Funding Open Access funding provided by Fachhochschule Nordwestschweiz FHNW..

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Ansari AQ, Hanmandlu M, Kour J, Singh AK (2014) Online signature verification using segment-level fuzzy modelling. IET Biom 3(3):113–127
2. Batista L, Granger E, Sabourin R (2012) Dynamic selection of generative-discriminative ensembles for off-line signature verification. Pattern Recognit 45(4):1326–1340
3. Batool FE, Attique M, Sharif M, Javed K, Nazir M, Abbasi AA, Iqbal Z, Riaz N (2020) Offline signature verification system: a novel technique of fusion of GLCM and geometric features using SVM. Multimed Tools Appl 1–20
4. Bertolini D, Oliveira LS, Justino E, Sabourin R (2010) Reducing forgeries in writer-independent off-line signature verification through ensemble of classifiers. Pattern Recognit 43(1):387–396
5. Bharathi RK, Shekar BH (2013) Off-line signature verification based on chain code histogram and support vector machine. In: International conference on advances in computing, communications and informatics (ICACCI), pp 2063–2068
6. Bhattacharya I, Ghosh P, Biswas S (2013) Offline signature verification using pixel matching technique. Procedia Technol 10:970–977
7. Blankers VL, van den HCE, Franke KY, Vuurpijl LG (2009) Icdar 2009 signature verification competition. In: International conference on document analysis and recognition, pp 1403–1407
8. CEDAR (Center of Excellence for Document Analysis and Recognition) Dataset. http://www.cedar.buffalo.edu/NIJ/publications.html. Last accessed: 2017-06-12
9. Chen S, Srihari S (2005) Use of exterior contours and shape features in off-line signature verification. In: International conference on document analysis and recognition (ICDAR), pp 1280–1284
10. Chen S, Srihari S (2006) A new off-line signature verification method based on graph matching. In: International conference on pattern recognition (ICPR), pp 869–872
11. Cpalka K, Zalasinski M (2014) On-line signature verification using vertical signature partitioning. Expert Syst Appl 41(9):4170–4180
12. Cpalka K, Zalasinski M, Rutkowski L (2016) A new algorithm for identity verification based on the analysis of a handwritten dynamic signature. Appl Soft Comput 43:47–56
13. Ferrer MA, Alonso JB, Travieso CM (2005) Offline geometric parameters for automatic signature verification using fixed-point arithmetic. IEEE Trans Pattern Anal Mach Intell 27(6):993–997
14. Galbally J, Marcel S, Fiérrez J (2014) Image quality assessment for fake biometric detection: application to iris, fingerprint, and face recognition. IEEE Trans Image Process 23(2):710–724
15. Gonzalez RC, Woods RE (2006) Digital Image Processing (3rd Edition). Prentice-Hall, Inc
16. GPDS-100 (Grupo de Procesado Digital de la Senal) Dataset. http://www.gpds.ulpgc.es/download/. Last accessed: 2017-06-12
17. Griechisch E, Malik MI, Liwicki M (2014) Online signature verification based on Kolmogorov–Smirnov distribution distance. In: International conference on frontiers in handwriting recognition (ICFHR), pp 738–742
18. Guerbai Y, Chibani Y, Hadjadji B (2015) The effective use of the one-class SVM classifier for handwritten signature verification based on writer-independent parameters. Pattern Recognit 48(1):103–113
19. Hafemann LG, Sabourin R, Oliveira LS (2016) Writer-independent feature learning for offline signature verification using deep convolutional neural networks. In: International joint conference on neural networks, pp 2576–2583
20. Hamadene A, Chibani Y, Nemmour H (2012) Off-line handwritten signature verification using contourlet transform and co-occurrence matrix. In: 2012 International conference on frontiers in handwriting recognition (ICFHR), pp 343–347
21. Hanmandlu M, Yusof MHM, Madasu VK (2005) Off-line signature verification and forgery detection using fuzzy modeling. Pattern Recognit 38(3):341–356
22. He Z, You X, Tang YY, Fang B, Du J (2006) Handwriting-based personal identification. Int J Pattern Recognit Artif Intell 20(2):209–225

23. Jain A, Hong L, Bolle R (1997) On-line fingerprint verification. IEEE Trans Pattern Anal Mach Intell 19(4):302–314
24. Jiang N, Xu J, Yu W, Goto S (2013) Gradient local binary patterns for human detection. In: International symposium on circuits and systems, pp 978–981
25. Justino EJ, Bortolozzi F, Sabourin R (2005) A comparison of SVM and HMM classifiers in the off-line signature verification. Pattern Recognit Lett 26(9):1377–1385
26. Kalera MK, Srihari S, Xu A (2004) Offline signature verification and identification using distance statistics. Int J Pattern Recognit Artif Intell 18(07):1339–1360
27. Klette R, Rosenfeld A (2004) Digital straightness: a review. Discrete Appl Math 139(1–3):197–230
28. Kovari B, Charaf H (2013) A study on the consistency and significance of local features in off-line signature verification. Pattern Recognit Lett 34(3):247–255
29. Kumar MM, Puhan NB (2014) Off-line signature verification: upper and lower envelope shape analysis using chord moments. IET Biom 3(4):347–354
30. Kumar R, Kundu L, Chanda B, Sharma J (2010) A writer-independent off-line signature verification system based on signature morphology. In: International conference on intelligent interactive technologies and multimedia, pp 261–265
31. Kumar R, Sharma J, Chanda B (2012) Writer-independent off-line signature verification using surroundedness feature. Pattern Recognit Lett 33(3):301–308
32. Lajevardi SM, Arakala A, Davis SA, Horadam KJ (2013) Retina verification system based on biometric graph matching. IEEE Trans Image Process 22(9):3625–3635
33. Larkins R, Mayo M (2008) Adaptive feature thresholding for off-line signature verification. In: International conference on image and vision computing, pp 1–6
34. Loka H, Zois EN, Economou G (2017) Long range correlation of preceded pixels relations and application to off-line signature verification. IET Biom 6(2):70–78
35. Lv H, Wang W, Wang C, Zhuo Q (2005) Off-line chinese signature verification based on support vector machines. Pattern Recognit Lett 26(15):2390–2399
36. Nguyen V, Kawazoe Y, Wakabayashi T, Pal U, Blumenstein M (2010) Performance analysis of the gradient feature and the modified direction feature for off-line signature verification. In: International conference on frontiers in handwriting recognition (ICFHR), pp 303–307
37. Ooi SY, Teoh ABJ, Pang YH, Hiew BY (2016) Image-based handwritten signature verification using hybrid methods of discrete radon transform, principal component analysis and probabilistic neural network. Appl Soft Comput 40:274–282
38. Pal S, Alaei A, Pal U, Blumenstein M (2016) Performance of an off-line signature verification method based on texture features on a large indic-script signature dataset. In: Workshop on document analysis systems (DAS), pp 72–77
39. Pham TA, Le H, Do N (2015) Offline handwritten signature verification using local and global features. Ann Math Artif Intell 75(1–2):231–247
40. Rosenfeld A (1974) Digital straight line segments. IEEE Trans Comput 23(12):1264–1269
41. Sae-Bae N, Memon ND (2014) Online signature verification on mobile devices. IEEE Trans Inf Forens Secur 9(6):933–947
42. Said HES, Tan TN, Baker KD (2000) Personal identification based on handwriting. Pattern Recognit 33(1):149–160
43. Sauvola JJ, Pietikäinen M (2000) Adaptive document image binarization. Pattern Recognit 33(2):225–236
44. Serdouk Y, Nemmour H, Chibani Y (2016) New off-line handwritten signature verification method based on artificial immune recognition system. Expert Syst Appl 51:186–194
45. Muhammad SK, Muhammad AF, Muhammad Y, Mussarat F, Steven L (2020) A framework for offline signature verification system: best features selection approach. Pattern Recognit Lett 139:50–59
46. Ruiz-del Solar J, Devia C, Loncomilla P, Concha F (2008) Off-line signature verification using local interest points and descriptors. In: Iberoamerican congress on pattern recognition, pp 22–29
47. Vargas JF, Ferrer MA, Travieso C, Alonso JB (2011) Off-line signature verification based on grey level information using texture features. Pattern Recognit 44(2):375–385
48. Vargas JF, Ferrer MA, Travieso CM, Alonso JB (2008) Off-line signature verification based on high pressure polar distribution. In: International conference on frontiers in handwriting recognition (ICFHR), pp 373–378
49. WEKA—The Workbench for Machine Learning. https://www.cs.waikato.ac.nz/ml/weka/. Last accessed: 2021-01-18
50. Zois EN, Alewijnse L, Economou G (2016) Offline signature verification and quality characterization using poset-oriented grid features. Pattern Recognit 54:162–177
51. Zois Elias N, Alexandridis A, Economou G (2019) Writer independent offline signature verification based on asymmetric pixel relations and unrelated training-testing datasets. Expert Syst Appl 125:14–32
52. ZulNarnain Z, Rahim MSM, Ismail NAF, Arsad MAM (2016) Triangular geometric feature for offline signature verification. World Acad Sci Eng Technol Int J Comput Electr Autom Control Inf Eng 10(3):485–488