Graph Embedding for Offline Handwritten Signature Verification

Michael Stauffer
University of Applied Sciences and Arts Northwestern Switzerland
Riggenbachstrasse 16
4600 Olten, Switzerland
michael.stauffer@fhnw.ch

Paul Maergner
University of Fribourg
Boulevard de Pérolles 90
1700 Fribourg, Switzerland
paul.maergner@unifr.ch

Andreas Fischer
University of Fribourg and University of Applied Sciences and Arts Western Switzerland
Boulevard de Pérolles 80
1700 Fribourg, Switzerland
andreas.fischer@hefr.ch

Kaspar Riesen
University of Applied Sciences and Arts Northwestern Switzerland
Riggenbachstrasse 16
4600 Olten, Switzerland
kaspar.riesen@fhnw.ch

ABSTRACT

Due to the high availability and applicability, handwritten signatures are an eminent biometric authentication measure in our life. To mitigate the risk of a potential misuse, automatic signature verification tries to distinguish between genuine and forged signatures. Most of the available signature verification approaches make use of vectorial rather than graph-based representations of the handwriting. This is rather surprising as graphs offer some inherent advantages. Graphs are, for instance, able to directly adapt their size and structure to the size and complexity of the respective handwritten entities. Moreover, several fast graph matching algorithms have been proposed recently that allow to employ graphs also in domains with large amounts of data. The present paper proposes to use different graph embedding approaches in conjunction with a recent graph-based signature verification framework. That is, signature graphs are not directly matched with each other, but first compared with a set of predefined prototype graphs, in order to obtain a dissimilarity representation. In an experimental evaluation, we employ the proposed method on two widely used benchmark datasets. On both datasets, we empirically confirm that the learning-free graph embedding outperforms state-of-the-art methods with respect to both accuracy and runtime.

ICBEA 2019, May 29–31, 2019, Stockholm, Sweden
© 2019 Association for Computing Machinery.
ACM. ISBN 978-1-4503-6305-1/19/05...$15.00
DOI: 10.1145/3345336.3345346

CCS Concepts

•**Security and privacy → Biometrics; Graphical / visual passwords; Authorization;** •**Computing methodologies → Biometrics;** •**Applied computing →** *Document capture; Document analysis;*

Keywords

Signature Verification; Graph Matching; Graph Representation; Graph Embedding; Ensemble Methods;

1. INTRODUCTION

Since decades, handwritten signatures are used as a biometrical authentication measure in business and legal transactions around the world [1, 2]. For this reason, handwritten signatures are omnipresent in our everyday life, and thus, expose a large risk for possible misuse. To mitigate this risk, automatic signature verification systems can be employed. That is, a questioned signature is compared with a set of reference signatures in order to distinguish between genuine and forged signatures [3–5]. The accuracy of most signature verification approaches rely on the amount of available reference signatures. In most cases, however, the acquisition of reference signatures is expensive and/or limited, and thus, signature verification is generally regarded as a challenging task (even for human experts).

Signature verification systems can be distinguished with respect to their input device [1, 2]. In case of *online* (also termed *dynamic*) signature verification, signatures are acquired by means of an electronic input device, such as for example, digitised pens or tablets, or via input on a touch screen. As a result, dynamic temporal information of the handwriting process (e.g. acceleration, speed, and pressure) can be acquired during the signing process. In contrast to that, *offline* (also termed *static*) signature verification is based on scanned signatures, and thus, limited to the (x, y)-positions of the handwritten strokes. For this reason, offline signature verification is commonly regarded as the more challenging task. The present paper is focusing on offline signature verification.

In addition to the type of input, signature verification can also be distinguished with respect to the type of signature representation, viz. statistical and structural representation. In case of *statistical* (i.e. vectorial) representations, signatures are represented by means of feature vectors or sequences of feature vectors that are extracted from scanned images of handwritten signatures [1, 2]. In early works, features are used to represent characteristics of the handwriting like for example projection profiles [6], slant direction [7, 8], outline [3], or the contour [4, 9]. Yet, also more generic feature descriptors have been employed such as for example *Local Binary Patterns (LBP)* [10, 11] and *Histogram of oriented Gradients (HoG)* [10]. In the last years, features of handwriting signatures have been derived by means of *Convolutional Neural Networks (CNN)* [5] and other Deep Learning techniques [12]. Regardless of the employed type of features, different classification and/or matching approaches have been employed such as for example *Dynamic Time Warping (DTW)* [6, 9], *Support Vector Machines (SVM)* [3, 10, 11], or *Hidden Markov Models (HMM)* [3, 8], to mention just a few.

A number of *structural* (i.e. string, tree, or graph-based) signature verification frameworks have been proposed in the last decade [13–17]. Graphs, actually the most general data structure, allow to represent the inherent topological handwriting characteristics of signatures in a natural and comprehensive way. Moreover, graphs are able to directly adapt both the structure and the size to the size and complexity of the underlying handwriting signature. However, this representational advantage is accompanied with an increased computational complexity of most mathematical operations. In fact, the computation of a graph similarity or dissimilarity measure is of much higher complexity when compared to the same operation on vectorial representations. To overcome this limitation, several fast *graph matching* algorithms have been proposed in the last decade [18, 19].

To bridge the gap between statistical and structural approaches in pattern recognition, a number of different methods for *graph embedding* as well as *graph kernels* have been proposed in the last decade [18–20]. Both approaches aim at mapping individual graphs or pairs of graphs into (an implicit) feature space. Consequently, several statistical measures and mathematical operations can be applied to the graphs maps [21]. To the best of our knowledge, neither graph embeddings nor graph kernels have been employed in the field of structural signature verification. In the present paper, we thus focus on different graph embedding approaches for signature verification. In particular, we make use of the dissimilarity based graph embedding framework originally presented in [21]. By means of this procedure, we can derive n-dimensional feature representations for graphs that can eventually be used for the classification of signatures. In an experimental evaluation on two widely used benchmark datasets, we show that the proposed approach can keep up or even outperform state-of-the-art statistical and structural signature verification frameworks with respect to both accuracy and runtime. This is particular interesting as the present approach is not depending on large amounts of training data.

The remainder of this paper is organised as follows. In Section 2, we formally introduce the employed signature verification framework including the graph embedding approaches. In Section 3, we present the results of a thorough empirical investigation. Finally, we draw conclusions and discuss possible future lines of research in Section 4.

2. SIGNATURE VERIFICATION PROCESS

In the present paper, we use different graph embedding approaches in conjunction with a recently proposed graph-based signature verification framework [15].

Hence, we first review the existing framework and discuss how handwritten signatures can be represented by means of graphs. Next, we show how graph embedding approaches can be used to extend this framework. Moreover, we propose to combine the embedding approaches with direct graph matchings in order to build an ensemble. Finally, we review a normalisation method that allows to reduce intrapersonal variations in the signature data.

2.1 Graph Representation

To represent handwritten signatures by means of graphs, scanned artefacts are first preprocessed in three subsequent processing steps [15]. First, *Difference of Gaussians* enhancement is employed to reduce the influence of noisy background. Next, signature images are binarised by means of global thresholding. Finally, preprocessed signature images are skeletonised by means of thinning.

In general, a graph g is defined as a four-tuple $g = (V, E, \mu, \nu)$ where V and E are finite sets of nodes and edges, and $\mu : V \to L_V$ and $\nu : E \to L_E$ are labelling functions for nodes and edges, respectively. In our specific scenario described below, nodes are labelled with two-dimensional numerical labels, while edges remain unlabelled, i.e. $L_V = \mathbb{R}^2$ and $L_E = \emptyset$.

We make use of so-called keypoint graphs [22, 23]. That is, graphs are extracted based on the detection of characteristic points (keypoints), in the preprocessed and skeletonised signature images. In particular, nodes are used to represent keypoints (as well as intermediate points between the keypoints), while edges are used to represent strokes between keypoints. In Figure 1, two exemplary signatures of two different datasets (see Sect. 3.1 for details) as well as their corresponding graph representations are shown.

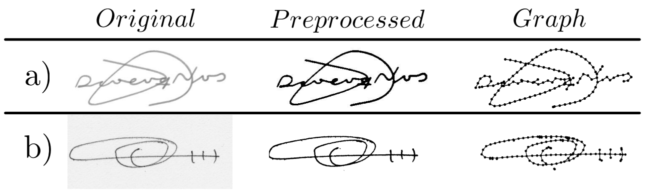


Figure 1. Exemplary offline signature (original and preprocessed) as well as the corresponding keypoint graph representation: (a) GPDSsynthetic, (b) MCYT-75.

2.2 Graph Embedding

The actual signature verification is based on two subsequent processing steps as shown in Figure 2. First, a *prototype-based graph embedding* [21] is employed. In particular, all graphs $g \in \mathcal{G}$ — actually representing signatures — are mapped to a feature vector space by means of a set of prototype graphs $E = \{e_1, \dots e_m\}$. Formally, the mapping $\varphi : \mathcal{G} \to \mathbb{R}^m$ is defined by

$$\varphi(g) = (d(g,e_1), d(g,e_2), \dots, d(g,e_m)) \quad ,$$

where $d(g, e_i)$ is equal to a graph dissimilarity between g and $e_i \in E$. Note that, the vector space embeddings obtained via φ are normalised by a z-score to reduce variations between different embedded signature graphs.
In order to compute the dissimilarities between a given graph g and the prototype graphs $e_i \in E$, we make use of a recently proposed linear time graph dissimilarity measure, viz. the so-called *Polar Graph Distance (PGD)* [24]. For each pair of graphs (g, e_i), we employ

$$d(g, e_i) = PGD(q, e_i) \quad .$$

PGD is based on three different processing steps. First, both graphs are segmented into a polar coordinate system (defined by the number of different radii P_r and the number of different angles P_ϕ). Second, histograms are derived based on the node and edge distributions of each polar graph segment (defined by the radial range P per histogram bin). Third, the distance between the resulting histograms is computed by means of the χ^2-distance. For more details we refer to [24].

2.3 Definition of the Prototypes

Clearly, the employed graph embedding crucially depends on the definition of adequate sets of prototypes. In the present paper, the prototype graphs E are based on two different sets, i.e. the *Reference Embedding (RE)* and the *Prototype Embedding (PE)*.
In case of RE, the set of prototype signature graphs E is equal to the set of reference signature graphs R of a specific user u, and thus, varying for every user (and dataset). In particular, the prototype set E is either composed of five or ten reference signatures for a given user (that is, we use two different scenarios in our evaluation – see below).
In case of PE, the set of prototype graphs E is defined on a global set of graphs, and thus, not adapted for every user (and dataset). In particular, E is based on ten different genuine signatures of 100 different users of an independent dataset (viz. GPDSlast-100, see Sect. 3.1 for details). That is, in case of $|E| = 25$, for instance, we use 25 different signatures from the 25 first users, in case of $|E| = 125$ we use 125 different signatures from 100 different users (i.e. two prototype signatures are used from the first 25 users), etc. In contrast with RE, the optimal size of E is not *a priori* defined and needs to be optimised separately.
Clearly, there are more sophisticated methods for the selection of prototype graphs available (e.g. [21, 25]). However, the chosen approach can be seen as a baseline for future investigations.

2.4 Ensemble Methods

Rather than conducting the signature verification with a single graph embedding approach (i.e. RE or PE), we combine both graph embeddings with different graph matching approaches in order to build a verification *ensemble*. We combine the graph embeddings with themself as well as with different graph matching algorithms, i.e. *Bipartite Graph Edit Distance (BP)* [26] and *Polar Graph Distance (PGD)* [24]. Formally, we make use of the following combinations

$$\alpha\, RE(q,r) + (1-\alpha)\, PE(q,r) \quad ,$$

$$\alpha\, BP(q,r) + (1-\alpha)\, RE(q,r) \quad ,$$
$$\alpha\, BP(q,r) + (1-\alpha)\, PE(q,r) \quad ,$$
$$\alpha\, BP(q,r) + (1-\alpha)\, RE(q,r) + \beta\, PE(q,r) \quad ,$$

$$\alpha\, PGD(q,r) + (1-\alpha)\, RE(q,r) \quad ,$$
$$\alpha\, PGD(q,r) + (1-\alpha)\, PE(q,r) \quad ,$$
$$\alpha\, PGD(q,r) + (1-\alpha)\, RE(q,r) + \beta\, PE(q,r) \quad ,$$

$$\alpha\, BP(q,r) + (1-\alpha)\, PGD(q,r) + \beta\, RE(q,r) \quad ,$$
$$\alpha\, BP(q,r) + (1-\alpha)\, PGD(q,r) + \beta\, PE(q,r) \quad ,$$
$$\alpha\, BP(q,r) + (1-\alpha)\, PGD(q,r) + \beta\, RE(q,r) + (1-\beta)\, PE(q,r) \quad ,$$

where $\alpha, \beta \in [0,1]$ are weighting factors that can be adjusted.

2.5 Signature Verification

Our signature verification process is relying on reference signatures r_i from a reference set of signatures R_u per user u. That is, we measure the pairwise *Euclidean* distance $d(\cdot,\cdot)$ between the embedding $\varphi(q)$ of a questioned signature graph q and all embeddings $\varphi(r_i)$ of all reference signature graphs $r_i \in R_u$.
To reduce interpersonal variations, we make use of a user-based score normalisation [27]. In particular, we calculate a normalisation score μ_{R_u} based on the average of all minima between all reference signatures R_u of user u

$$\mu_{R_u} = \frac{\sum\limits_{s \in R_u} \min\limits_{r \in R_u \setminus \{s\}} d(\varphi(s), \varphi(r))}{|R_u|} \quad ,$$

where the embedding $\varphi(\cdot)$ is either based on RE or PE.
Note that the mean of the minimal distances μ_{R_u} can be regarded as the expected dissimilarity score for user u. For this reason, we make use of μ_{R_u} to derive a signature score s for a questioned signature q based on the sum of the k smallest normalised distances for user u. Formally, for user u we first sort the reference signatures $r_i \in R_u$ with respect to $d(q, r_i)$ such that $d(q, r_1) \le d(q, r_2) \le \dots$ and then define

$$s(q, R_u) = \sum_{i=1}^{k} \frac{d(q, r_i)}{\mu_{R_u}} \quad . \tag{1}$$

A similar normalisation process is also employed on combined distances (using both BP and PGD).
If the normalised score $s(q, R_u)$ is below a certain threshold, the questioned signature q is regarded as genuine, otherwise q is regarded as forged.

3. EXPERIMENTAL EVALUATION

3.1 Datasets

In the following experimental evaluation, the proposed signature verification framework is evaluated on two offline signature benchmark datasets, i.e. *GPDSsynthetic-Offline* and *MCYT-75*.
In case of GPDSsynthetic-Offline, offline signatures are synthetically generated [28]. This novel dataset replaces the widely used GPDS-960 dataset that is no longer publicly

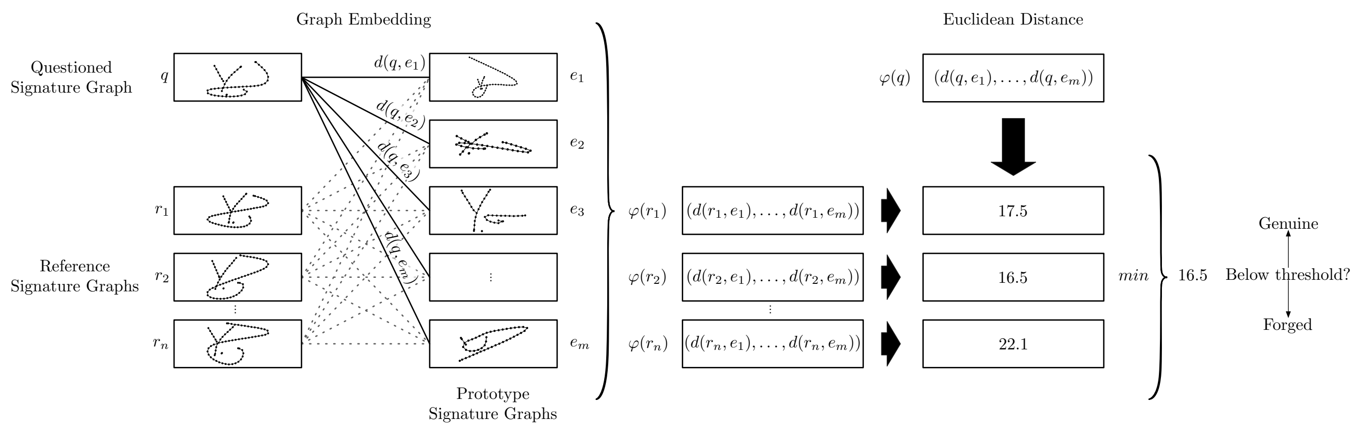


Figure 2. Graph-based signature verification process including graph embedding (left) and dissimilarity measure by means of Euclidean distance (right).

Table 1. The number of users, the number of genuine and forged signatures per user, and the resolution of the images.

Name	Users	Genuine	Forgeries	dpi
GPDSsynthetic	4000	24	30	600
MCYT-75	75	15	15	600

available[1] and contains 4,000 synthetic users with 24 genuine and 30 forged signatures per user. Note that every signature is generated by modelling different pens. In case of MCYT-75, offline signatures are based on the MCYT baseline corpus [8,29]. This well-known dataset contains 75 users with 15 genuine and 15 forged signatures per user. All users have signed on a 127 mm × 97 mm field. In Table 1, an overview of the main characteristics of the two datasets is given.

3.2 Experimental Setup

In the domain of signature verification, frameworks are generally evaluated using two kind of forgeries, i.e. *Random Forgeries (RF)* and *Skilled Forgeries (SF)*. In case of RF, the forger has no prior knowledge about the signature to be forged[2], while in case of SF, the forger has commonly knowledge of one (or several) genuine signatures as well as time to train to forge the signature. Hence, skilled forgeries are often visually very similar when compared with the genuine template, and thus, more difficult to detect.

To verify whether a questioned signature q is genuine or forged, q is commonly compared with a set of reference signatures R. In general, there is a strong positive correlation between the size of the reference set R and the accuracy of the signature verification system. However, the acquisition of arbitrary large reference sets is often not possible and/or expensive in practice, and thus, restricted to few reference signatures per user. In the present paper, we make use of $|R| = 5$ (denoted by $R5$) and $|R| = 10$ (denoted by $R10$). In particular, we make use of the first five or ten genuine signatures as reference set R for each user, while the remaining genuine signatures are used as the positive signing attempts for the evaluation.

The accuracy of the signature verification framework is generally measured by two kind of errors, i.e. the *False Rejection Rate (FRR)* and the *False Acceptance Rate (FAR)*. In case of FRR, the accuracy is measured by the percentage of genuine signatures that are falsely rejected by the system, while in case of FAR, the accuracy is measured by the percentage of forgeries that are falsely accepted by the system. Finally, both measures can be used to detect the *Equal Error Rate (EER)*, which refers to the error rate when the FRR is equal to the FAR.

In the present paper, the experimental evaluation is conducted in two subsequent steps. In the first step, we optimise the proposed systems on a single and independent dataset, i.e. a subset of GPDSsynthetic containing the last 100 users of the dataset (denoted by *GPDS-last100* from now on), using a skilled forgery scenario and ten reference signatures (R10). In the second step, we test the proposed framework (using optimised parameters) on an independent subset of GPDSsynthetic containing the first 75 users of the dataset (denoted by *GPDS-75*) as well as on *MCYT-75*.

3.3 Parameter Optimisation

For both graph embedding approaches RE and PE, we first optimise different polar segmentations for the polar graph distance PGD (defined via $P_r = \{1, 2, \dots, 6\}$ and $P_\phi = \{6, 7, \dots, 12\}$) as well as different radial ranges (defined by $P = \{26, 28, \dots, 34\}$) in combination with different number of k reference signatures, i.e. $k = \{1, 2, \dots, 5\}$ (see Eq. 1).

For the optimisation of the dissimilarity measure PGD, we set $m = 10$ and $m = 100$ for RE and PE, respectively. In our final evaluation, for RE the number of prototypes m is set to 5 or 10 with respect to the actual scenario (R5 or R10). For PE we separately optimise $m \in \{25, 50, \dots, 1000\}$. It turns out that the EER is not (or only marginally) improved with m larger than 325. In Table 2, the best performing parameters are shown for both RE and PE.

In a final step, we optimise the weighting parameters $\alpha, \beta \in \{0.1, 0.2, \dots, 0.9\}$ for the ensemble methods in combination with $k = \{1, 2, \dots, 5\}$. In Table 3, the best performing weights are shown for all combinations.

[1]For details, http://www.gpds.ulpgc.es/downloadnew/download.htm (March 30, 2018).

[2]In the present paper, we make use of the first genuine signature from each other user to build a set of random forgeries.

Table 2. Optimal graph embedding parameter settings.

Method	P_r	P_ϕ	P	k	m
RE	3	8	26	1	5/10
PE	4	9	30	2	325

Table 3. Optimal ensemble method parameter settings.

Ensemble	α	β	k
RE + PE	0.7	-	3
BP + RE	0.4	-	2
BP + PE	0.4	-	2
BP + RE + PE	0.4	0.1	2
PGD + RE	0.3	-	1
PGD + PE	0.6	-	3
PGD + RE + PE	0.6	0.4	3
BP + PGD + RE	0.8	0.6	1
BP + PGD + PE	0.4	0.3	4
BP + PGD + RE + PE	0.6	0.5	4

3.4 Results and Discussion

In Table 4, we compare the proposed graph embedding approaches RE and PE in a skilled forgery scenario (SF) with two graph matching algorithms, i.e. BP and PGD. We observe that the embeddings using RE lead in general to improvements on GPDS-75, while PGD performs better than the embedding using RE on MCYT-75.

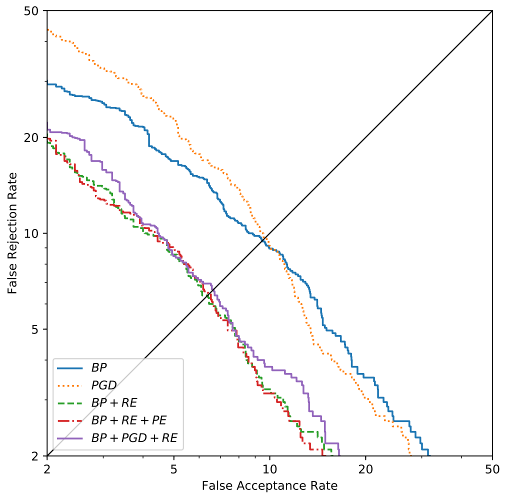
Regarding the graph embedding based on PE, we observe lower accuracy rates when compared with both BP and PGD. However, if we combine RE and PE (i.e. RE + PE), we observe substantial improvements when compared with the single graph embedding methods. Further improvements can be observed in case of the ensemble methods BP + RE, BP + RE + PE, as well as BP + PGD + RE. In fact, these ensemble methods are in twelve out of twelve cases better than both individual systems BP and PGD. The same observation can also be made in the *Detection Error Tradeoff (DET)* curves in Figure 3.

Table 4. Equal error rates in a skilled forgeries scenario with 5 and 10 references on GPDS-75 and MCYT-75.

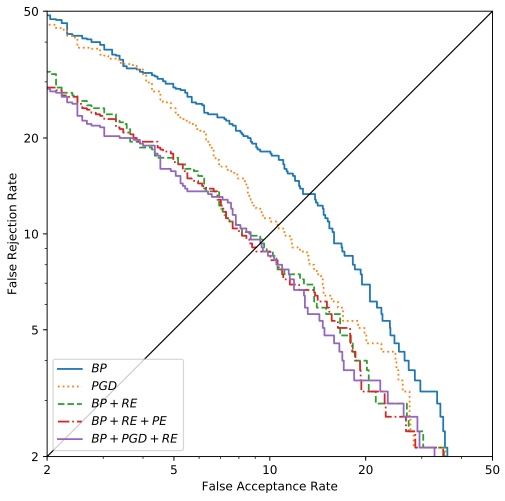
System	GPDS-75		MCYT-75	
	R5	R10	R5	R10
BP	12.31	9.47	19.73	13.24
PGD	12.40	9.64	16.53	10.67
RE	11.02	8.00	17.60	11.82
PE	14.80	11.24	23.02	16.98
RE + PE	10.58	8.04	16.89	11.11
BP + RE	9.02 (2)	6.36 (1)	15.82 (2)	9.51 (3)
BP + PE	11.82	8.84	18.93	12.71
BP + RE + PE	8.76 (1)	6.44 (2)	15.11 (1)	9.07 (1)
PGD + RE	11.24	8.44	17.24	10.84
PGD + PE	10.80	8.31	16.98	11.11
PGD + RE + PE	10.31	7.78	16.80	10.31
BP + PGD + RE	9.24 (3)	6.62 (3)	15.91 (3)	9.33 (2)
BP + PGD + PE	9.60	6.80	15.91 (3)	9.69
BP + PGD + RE + PE	9.38	7.07	16.36	9.51 (3)

Similar observations can also be made in the random forgery scenario (RF) in Table 5. That is, the graph embedding using RE leads to similar EER rates when compared with BP and PGD, while a certain decline can be observed in the case of graph embedding based on PE. Moreover, we observe that some ensemble methods (viz. BP+RE, BP+RE+PE, and BP + PGD + RE) lead to clear improvements when compared with the graph-based reference systems. In particular, the ensemble method BP + RE + PE leads in three out of four cases to the overall lowest EER rates.

Next, we compare the matching times of the different graph matching algorithms (i.e. BP and PGD) on graphs with different sizes, as shown in Table 6[3]. BP has cubic time complexity with respect to the number of nodes [26], while PGD offers linear time complexity [24]. Note that the pro-



(a) GPDS-75



(b) MCYT-75

Figure 3. Detection error tradeoff curves for skilled forgeries with R=10. For the sake of readability, we show only a subset of all curves.

[3]Note that all performance-related experiments have been measured on the same machine (iMac 5K, 4GHz Intel Core i7, 32GB DDR3) in a single thread scenario.

Table 5. Equal error rates in a random forgeries scenario with 5 and 10 references on GPDS-75 and MCYT-75.

System	GPDS-75		MCYT-75	
	R5	R10	R5	R10
BP	5.75	3.80	5.73	2.67
PGD	4.43	3.12	5.19	2.13
RE	4.67	2.88	6.00	2.88
PE	8.41	5.80	8.25	4.50
RE + PE	4.56	2.65	5.46	2.23
BP + RE	3.44 (3)	2.09 (1)	3.77 (2)	1.75 (2)
BP + PE	5.82	3.60	5.46	2.65
BP + RE + PE	3.30 (1)	2.09 (1)	3.73 (1)	1.80 (3)
PGD + RE	4.29	2.79	5.33	2.40
PGD + PE	4.97	3.21	5.73	2.04
PGD + RE + PE	4.27	2.90	5.30	1.86
BP + PGD + RE	3.35 (2)	2.18 (2)	3.73 (1)	1.62 (1)
BP + PGD + PE	4.00	2.56	4.13 (3)	1.86
BP + PGD + RE + PE	4.00	2.47 (3)	4.52	1.87

posed graph embedding approaches make use of PGD as basic graph dissimilarity measure. That is, for the RE graph embedding a questioned signature graph is compared with ten or five reference signature graphs, while in case of embedding via PE a questioned signature graph is compared with 325 prototype graphs. Hence, even in the worst case (i.e. 325×2.0 ms $= 650$ ms) the matching times are still clearly lower than with BP.

Table 6. Matching time (ms) using different sizes of graphs. With $|\overline{V}|$ we denote the mean number of nodes of the graphs.

System					Average
GPDS-75	$\lvert\overline{V}\rvert = 315$	$\lvert\overline{V}\rvert = 218$	$\lvert\overline{V}\rvert = 168$	$\lvert\overline{V}\rvert = 139$	
BP	10,698.0	2,694.5	1,000.2	492.2	3,721.3
PGD	1.6	1.1	0.8	0.7	1.1
MCYT-75	$\lvert\overline{V}\rvert = 330$	$\lvert\overline{V}\rvert = 236$	$\lvert\overline{V}\rvert = 189$	$\lvert\overline{V}\rvert = 160$	
BP	12,597.6	3,423.8	1,389.1	805.3	4,554.0
PGD	2.0	1.5	1.4	1.1	1.5

Finally, we compare the proposed systems (including the ensemble methods) with five state-of-the-art signature verification systems in Table 7. The first reference systems [7] uses the Mahalanobis distance to compare slant and variability features, while in [8] similar features are used in conjunction with a HMM. The third reference systems [4] is based on contour-based features that are matched by means of the χ^2 distance. In a recent paper [12], Deep Multitask Metric Learning (DMML) is used in combination with Histogram of oriented Gradients (HoG) and Discrete Radon Transform (DRT) features. Lately, a CNN has been combined with a Spatial Pyramid Pooling (SPP) in [5].
In this evaluation, we make use of two threshold scenarios to measure the EER, i.e. a global and a local scenario. In the former case, no user-dependent threshold adoption is employed, while in the latter case, a *posteriori* user-dependent score normalisation is employed [8]. Generally, local thresholds lead to lower EER rates, whereas global thresholds are regarded as the more realistic case.
If we compare the RE graph embedding method with the non learning-based reference systems (i.e. [4, 7]), we observe an improvement in three out of four cases. Especially for random forgeries (using a local threshold), we observe substantially lower error rates. Moreover, we observe that the ensemble BP + PGD + RE is in each case among the three best performing systems. The CNN-based reference system [5] leads to the overall best performance in each test case. However, we have to keep in mind that our proposed framework is not depending on a learning procedure. In fact, the proposed method is optimised on a small and independent training set (i.e. GPDS-last100), while Deep Learning approaches (i.e. DMML and CNN) are known for their thorough training phases on very large datasets. Yet, especially in signature verification applications, the acquisition of large sets of signatures is often legally restricted and/or expensive.

4. CONCLUSION AND OUTLOOK

In the last years, a number of promising graph-based signature verification approaches have been proposed. These approaches make use of the inherent representational advantages of graphs when compared with statistical (i.e. vectorial) representations. That is, graphs are able to directly represent the topological characteristic of the handwriting and adapt both structure and size to the size and complexity of the underlying signature.
The present paper follows this line of research and proposes different graph embedding approaches for signature verification. Graph embedding tries to bridge the gap between structural and statistical approaches by means of explicit mappings of graphs into a feature space. In the present paper, we use the dissimilarity based graph embedding where a given graph is represented by its distances to m prototype graphs. We make use of two strategies in order to define these prototypes, viz. Reference Embedding (RE) and Prototype Embedding (PE). In case of RE, a graph is mapped into a feature space by comparing it with a user-specific set of reference graphs. In case of PE, the mapping is based on an independent set of prototype graphs.
In an experimental evaluation on two datasets, i.e. GPDS-75 and MCYT-75, we confirm that the proposed graph embedding approaches can keep up with graph-based state-of-the-art reference systems with respect to both runtime and accuracy. Moreover, the proposed graph embedding approaches can be combined with direct graph matching approaches to form an *ensemble*. This combinatorial strategy allows to keep up with recent Deep Learning approaches without the need of an *a priori* learning step. In contrast with learning-based approach, the proposed graph embedding is distinguished by not depending on large training data. The high generalisability of our approach is a clear advantage in case of signature verification, where the number of training data is often rather small and/or expensive to acquire.
In future research, we plan to employ the proposed graph embedding approaches in combination with more elaborated learning-based statistical classification methods (i.e. SVM, HMM, or CNN). Moreover, we plan to employ more sophisticated methods for the selection of the prototype graphs [21].

5. ACKNOWLEDGMENTS

This work has been supported by the Swiss National Science Foundation project 200021_162852.

6. REFERENCES

[1] Impedovo, D. and Pirlo, G. 2008. Automatic signature verification: The state of the art. *IEEE Transactions*

Table 7. Equal error rates compared to state-of-the-art methods on MCYT-75 using R10.

System	Global				Local			
	RF		SF		RF		SF	
Mahalanobis (Slant,Variability) [7]	-		-		7.26		22.13	
HMM (Slant,Envelope) [8]	-		-		1.14		9.28	
χ^2 (Contour) [4]	-		-		1.18		6.44	
DMML (HoG,DRT) [12]	0.37	(2)	9.86		-		-	
CNN (SPP) [5]	0.19	(1)	3.64	(1)	0.03	(1)	-	
RE	2.88		11.82		0.63		6.67	
PE	4.50		16.98		2.16		10.31	
RE + PE	2.23		11.11		0.70		5.87	
BP + RE	1.75		9.51		0.31	(3)	4.09	(1)
BP + PE	2.65		12.71		0.70		7.02	
BP + RE + PE	1.80		9.07	(2)	0.36		4.09	(1)
PGD + RE	2.40		10.84		0.40		5.87	
PGD + PE	2.04		11.11		0.76		5.87	
PGD + RE + PE	1.86		10.31		0.70		5.51	
BP + PGD + RE	1.62	(3)	9.33	(3)	0.29	(2)	4.09	(1)
BP + PGD + PE	1.86		9.69		0.40		4.36	(2)
BP + PGD + RE + PE	1.87		9.51		0.41		4.62	(3)

on Systems, Man and Cybernetics Part C: Applications and Reviews. 38, 5, 609 – 635.

[2] Hafemann, L. G., Sabourin, R., and Oliveira, L. S. 2017. Offline handwritten signature verification — Literature review. In *International Conference on Image Processing Theory, Tools and Applications.* IEEE 1–8.

[3] Ferrer, M. A., Alonso, J., and Travieso, C. 2005. Offline geometric parameters for automatic signature verification using fixed-point arithmetic. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 27, 6, 993–997.

[4] Gilperez, A., Alonso-Fernandez, F., Pecharroman, S., Fierrez, J., and Ortega-Garcia, J. 2008. Off-line signature verification using contour features. In *International Conference on Frontiers in Handwriting Recognition.* Concordia University.

[5] Hafemann, L. G., Oliveira, L. S., and Sabourin, R. 2018. Fixed-sized representation learning from offline handwritten signatures of different sizes. *International Journal on Document Analysis and Recognition.* 21, 3, 219–232.

[6] Piyush Shanker, A. and Rajagopalan, A. 2007. Off-line signature verification using DTW. *Pattern Recognition Letters.* 28, 12, 1407–1414.

[7] Alonso-Fernandez, F., Fairhurst, M., Fierrez, J., and Ortega-Garcia, J. 2007. Automatic Measures for Predicting Performance in Off-Line Signature. In *IEEE International Conference on Image Processing.* IEEE I–369–I–372.

[8] Fierrez-Aguilar, J., Alonso-Hermira, N., Moreno-Marquez, G., and Ortega-Garcia, J. 2004. An Off-line Signature Verification System Based on Fusion of Local and Global Information. In *International Workshop on Biometric Authentication.* Springer 295–306.

[9] Deng, P. S., Liao, H.-Y. M., Ho, C. W., and Tyan, H.-R. 1999. Wavelet-Based Off-Line Handwritten Signature Verification. *Computer Vision and Image Understanding.* 76, 3, 173–190.

[10] Yilmaz, M. B., Yanikoglu, B., Tirkaz, C., and Kholmatov, A. 2011. Offline signature verification using classifier combination of HOG and LBP features. In *International Joint Conference on Biometrics.* IEEE 1–7.

[11] Ferrer, M. A., Vargas, J. F., Morales, A., and Ordonez, A. 2012. Robustness of Offline Signature Verification Based on Gray Level Features. *IEEE Transactions on Information Forensics and Security.* 7, 3, 966–977.

[12] Soleimani, A., Araabi, B. N., and Fouladi, K. 2016. Deep Multitask Metric Learning for Offline Signature Verification. *Pattern Recognition Letters.* 80, 84–90.

[13] Bansal, A., Nemmikanti, P., and Kumar, P. 2008. Offline Signature Verification Using Critical Region Matching. In *International Conference on Future Generation Communication and Networking Symposia.* IEEE 115–120.

[14] Fotak, T., Bača, M., and Koruga, P. 2011. Handwritten signature identification using basic concepts of graph theory. *WSEAS Transactions on Signal Processing.* 7, 4, 117–129.

[15] Maergner, P., Riesen, K., Ingold, R., and Fischer, A. 2017. A Structural Approach to Offline Signature Verification Using Graph Edit Distance. In *International Conference on Document Analysis and Recognition.* IEEE 1216–1222.

[16] Maergner, P., Howe, N., Riesen, K., Ingold, R., and Fischer, A. 2018. Offline signature verification via structural methods: Graph edit distance and inkball models. In *International Conference on Frontiers in Handwriting Recognition.* IEEE 163–168.

[17] Maergner, P., Pondenkandath, V., Alberti, M.,

Liwicki, M., Riesen, K., Ingold, R., and Fischer, A. 2018. Offline signature verification by combining graph edit distance and triplet networks. In *International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*. Springer 470–480.

[18] Foggia, P., Percannella, G., and Vento, M. 2014. Graph Matching and Learning in Pattern Recognition in the last 10 Years. *International Journal of Pattern Recognition and Artificial Intelligence.* 28, 01, 1450001.

[19] Vento, M. 2015. A long trip in the charming world of graphs for Pattern Recognition. *Pattern Recognition.* 48, 2, 291–301.

[20] Conte, D., Ramel, J.-Y., Sidère, N., Luqman, M. M., Gaüzère, B., Gibert, J., Brun, L., and Vento, M. 2013. A Comparison of Explicit and Implicit Graph Embedding Methods for Pattern Recognition. In *International Workshop on Graph-Based Representations in Pattern Recognition*. Springer 81–90.

[21] Riesen, K. and Bunke, H. 2010. Graph Classification and Clustering Based on Vector Space Embedding, Series in Machine Perception and Artificial Intelligence, World Scientific.

[22] Fischer, A., Riesen, K., and Bunke, H. 2010. Graph similarity features for HMM-based handwriting recognition in historical documents. In *International Conference on Frontiers in Handwriting Recognition.* IEEE 253–258.

[23] Stauffer, M., Fischer, A., and Riesen, K. 2018. Keyword Spotting in Historical Handwritten Documents based on Graph Matching. *Pattern Recognition.* 81, 240–253.

[24] Stauffer, M., Maergner, P., Fischer, A., and Riesen, K. 2019. Polar Graph Embedding for Handwriting Applications. *International Journal on Document Analysis and Recognition.* Submitted.

[25] Ferrer, M., Valveny, E., Serratosa, F., Riesen, K., and Bunke, H. 2010. Generalized median graph computation by means of graph embedding in vector spaces. *Pattern Recognition.* 43, 4, 1642–1655.

[26] Riesen, K. and Bunke, H. 2009. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing.* 27, 7, 950–959.

[27] Fischer, A., Diaz, M., Plamondon, R., and Ferrer, M. A. 2015. Robust score normalization for DTW-based on-line signature verification. In *International Conference on Document Analysis and Recognition*. IEEE 241–245.

[28] Ferrer, M. A., Diaz-Cabrera, M., and Morales, A. 2015. Static signature synthesis: A neuromotor inspired approach for biometrics. *Transactions on Pattern Analysis and Machine Intelligence.* 37, 3, 667–680.

[29] Ortega-Garcia, J., Fierrez-Aguilar, J., Simon, D., Gonzalez, J., Faundez-Zanuy, M., Espinosa, V., Satue, A., Hernaez, I., Igarza, J.-J., Vivaracho, C., Escudero, D., and Moro, Q.-I. 2003. MCYT baseline corpus: a bimodal biometric database. *Vision, Image, and Signal Processing.* 150, 6, 395.