Generation and Evaluation of Brute-Force Signature Forgeries

Alain Wahl, Jean Hennebert, Andreas Humm, and Rolf Ingold

Université de Fribourg, Boulevard de Pérolles 90, 1700 Fribourg, Switzerland
alain.wahl,jean.hennebert,andreas.humm,rolf.ingold@unifr.ch

Abstract. We present a procedure to create brute-force signature forgeries. The procedure is supported by SIGN4J, a dynamic signature imitation training software that was specifically built to help people learn to imitate the dynamics of signatures. The main novelty of the procedure lies in a feedback mechanism that is provided to let the user know how good the imitation is and on what part of the signature the user has still to improve. The procedure and the software are used to generate a set of brute-force signatures on the MCYT-100 database. This set of forged signatures is used to evaluate the rejection performance of a baseline dynamic signature verification system. As expected, the brute-force forgeries generate more false acceptation in comparison to the random and low-force forgeries available in the MCYT-100 database.

1 Introduction

Most of nowadays available identification and verification systems are based on passwords or cards. Biometric systems will potentially replace or complement these traditional approaches in a near future. The main advantage of biometric systems lies in the fact that the user does not have anymore to remember passwords or keep all his different access keys. Another advantage lies in the difficulty to steal or imitate biometrics data, leading to enhanced security.

This work is fully dedicated to signature verification systems [6] [3]. Signature verification has the advantage of a very high user acceptance because people are used to sign in their daily life. Signature verification systems are said to be static (off-line) or dynamic (on-line). Static verification systems use a static digitalized image of the signature. Dynamic signature verification (DSV) systems use the dynamics of the signature including coordinates, pressure and sometimes angle of the pen as a function of time. Thanks to the extra information included in the time evolution of these features, dynamic systems are usually ranked as more accurate and more difficult to attack than static verification systems.

Signature verification systems are evaluated by analyzing their accuracy to accept genuine signatures and to reject forgeries. When considering forgeries, four categories can be defined from the lowest level of attack to the highest (as presented in [8] [9], and extended here).

– **Random forgeries**. These forgeries are simulated by using signature samples from other users as input to a specific user model. This category actually does not denote intentional forgeries, but rather accidental accesses by non-malicious users.

– **Blind forgeries**. These forgeries are signature samples generated by intentional impostors having access to a descriptive or textual knowledge of the original signature.
– **Low-force forgeries**. The impostor has here access to a visual static image of the original signature. There are then two ways to generate the forgeries. In the first way, the forger can use a blueprint to help himself copy the signature, leading to low-force **blueprint** forgeries. In the second way, the forger can train to imitate the signature, with or without a blueprint, for a limited or unlimited amount of time. The forger then generate the imitated signature, without the help of the blueprint and potentially after some time after training, leading to low-force **trained** forgeries. The so-called skilled forgeries provided with the MCYT-100 database [5] correspond here to low-force trained forgeries.
– **Brute-force forgeries**. The forger has access to a visual static image and to the whole writing process, therefore including the handwriting dynamics. The forger can analyze the writing process in the presence of the original writer or through a video-recording or also through a captured on-line version of the genuine signature. This last case is realized when genuine signature data can be intercepted, for example when the user is accessing the DSV system. In a similar way as in the previous category, the forger can then generate two types of forgeries. Brute-force **blueprint** forgeries are generated by projecting on the acquisition area a real-time pointer that the forger then needs to follow. Brute-force **trained** forgeries are produced by the forger after a training period where he or she can use dedicated tools to analyze and train to reproduce the genuine signature. In [9] and [8], tools for training to perform brute-force forgeries are presented.

We report in this article our study conducted in the area of brute-force trained forgeries. Rather than designing tools to help potential forger to imitate the dynamics of a signature, our primary objective is to understand how brute-force forgeries can be performed and to measure the impact of such forgeries on state-of-the-art DSV systems. Another objective that we will pursue in future work is to determine how DSV systems can be improved to diminish the potential risk of such brute-force forgeries.

The underlying assumptions that are taken in this work are twofold. First, the forger has access to one or more versions of a recorded on-line signature. Second, the forger trains to imitate the signature according to a specified procedure and using a dedicated software that (1) permits a precise analysis of the signature dynamics, (2) allow to train to reproduce the original signature and (3) gives feedback on "how close the forger is to break the system".

Section 2 introduces the procedure that was crafted to create brute-force trained forgeries. In Section 3, we present SIGN4J, the dynamic signature imitation training software that was specifically built to support the previous procedure. More details are given about the feedback mechanism which is a novelty in our approach. In Section 4, experiments performed using the procedure and the software are reported using the MCYT-100 database. Finally, conclusions are drawn in the last section.

2 Procedure to generate brute-force forgeries

Imitating the dynamics of a signature to perform brute-force forgeries is a difficult cognitive task considering the multiple and different pieces of information that are available. First, as for low-force forgeries, the global and local shapes of the signature need to be imitated. Second, the trajectory of the pen defining the temporal sequence of strokes need to be understood and then reproduced. For example, some users will draw the vertical bar of letter 'd' from bottom to top without a pen-up while some other users will draw it from top to bottom with a pen-up. Third, the average and local pen speed need to be reproduced. Fourth and finally, the pressure and, if available, the pen azimuth and elevation angles have also to be imitated.

Considering the difficulty of the task, we have crafted a step-by-step procedure that can be followed by the candidate forger to capture the most important pieces of dynamic information of a signature. This procedure has actually been refined through our experimentations and drove the development of our Sign4J software (see Section 3).

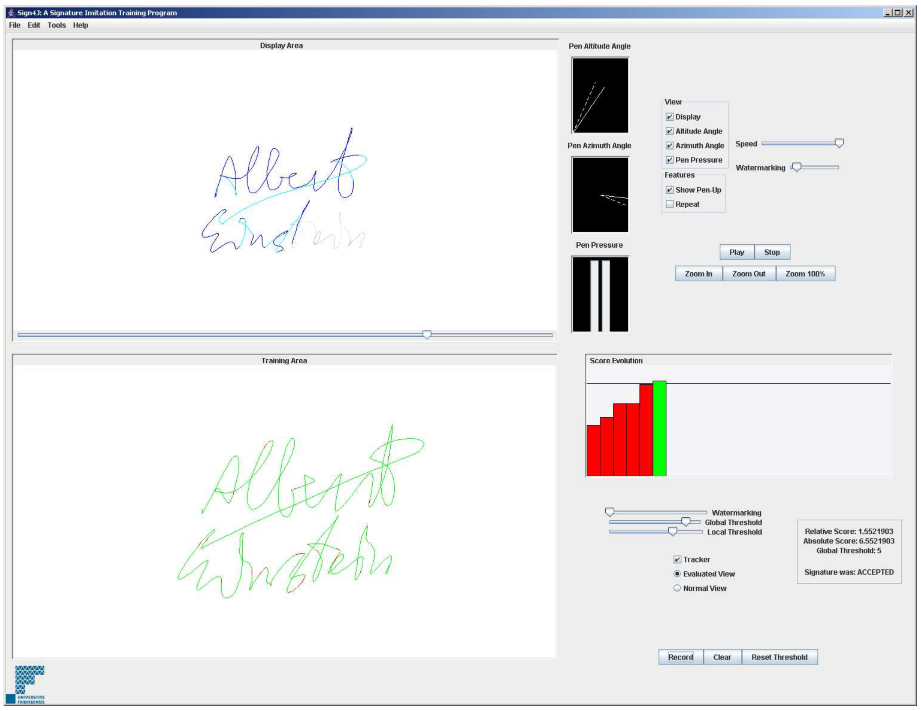
1. **Analyze and reproduce global visible features**. Analyze the global shape of the signature as well as the general sequence of letters and flourish signs. Train to reproduce at low speed the rough shape of the signature and the sequence of strokes.
2. **Reproduce the average angles**. Place hand and position pen in such a way that the angles correspond to the average angles of the genuine signature.
3. **Analyze and reproduce local features**. Analyze carefully the complex parts of the signature (flourish parts, high-speed sequence, etc.). Train on these complex parts separately then train to reproduce them in the right order, at the right speed.
4. **Retrain on different versions of the signature**. If several signatures are available, change frequently the signature on which training is performed.

The previous procedure was crafted to reach, on average and in a quite reduced training time, good quality of brute-force signatures. We removed on purpose from this procedure the analysis of local instantaneous angles, mainly because they are not easy to analyze and learn. For the same reason, we also removed the analysis of the local dynamics of the pressure with the further argument that the pressure value is pretty much dependent to the settings of the acquisition device. Training to reproduce instantaneous values of angles and pressure is probably possible but it would have increased dramatically the requested training time.

3 Design of Sign4J

Sign4J is a software that has been developed to support the procedure presented in Section 2. We describe here the most important features of Sign4J and give more details about the graphical user interface. Sign4J has been written in Java to benefit from the wide, already existing, graphical and utility libraries. This choice allowed us to reduce significantly the development time and make the software available on any operating system supporting Java. Sign4J currently supports the family of Wacom Cintiq devices integrating tablet and screen.

Figure 1 shows a screenshot of the interface of SIGN4J. The interface has been organized into different areas with, as principle, the top part of the view dedicated to the analysis of a genuine signature and the bottom part dedicated to forgery training.



Relative Score: 1.5521903
Absolute Score: 6.5521903
Global Threshold: 5

Signature was: ACCEPTED

Fig. 1. Screen Shot of SIGN4J Graphical User Interface

1. **Signature analysis**
   – In the top part, the **display area** gives a view of the original signature. Pens up corresponding to zero pressure values and pens down are displayed in two different colors, respectively cyan and blue. The signature

is actually drawn point by point on top of a static watermarking version. The watermarking can be set with a custom transparency level. The *play* button starts the display of a signature in real-time, i.e. reproducing the real velocity of the signer. Zooming functions allow to analyze in more details some specific parts of the signature trajectory.
   - The user can adjust the speed of the signature between 0% and 100% of the real-time speed with a slider. The slider below the display area can be used to go forward or backward onto some specific parts of the signature, in a similar manner as for a movie player.
   - The instantaneous elevation and azimuth angles are displayed as moving needles in two different windows. The average values of these angles are also displayed as fixed dashed needles.
   - The instantaneous pressure is displayed as a bar where the level represents the pressure value. The left bar indicates the pressure of the original signature and the right one shows the pressure of the forger.
   - Display of the angles, pressure or signature trajectory can be turned on or off with check boxes to allow a separate analysis of the different features.
2. **Forgery training**
   - In the bottom part, the **training area** is used to let the forger train to reproduce the original signature. An imitation can then be replayed in a similar manner as in the top analysis area. To ease the training, a blueprint of the genuine signature can be displayed. A tracking mode is also available where the genuine signature is drawn in real-time so that the forger can track the trajectory with the pen.
   - After an imitation has been performed, the signature is automatically sent to the DSV system that outputs a global score and a sequence of local scores. The global score has to reach a given threshold for the forgery to be accepted by the system. The global absolute score is displayed together with the global relative score that is computed by subtracting the absolute score from the global threshold. The global scores are kept in memory in order to plot a sequence of bars showing the progress of the training session. The global threshold value can be set using a slider.
   - By comparing the local scores to a local threshold value, regions of the signature where the user still has to improve are detected. The forger can then train more specifically on these regions. Figure 2 gives an example of such a local feedback with a clear indication that the first letter of the signature needs to be improved. We have to note here that when the forger performs equally well (or bad) on the signature, the color feedback is less precise and difficult to interpret. The local threshold can also be set with a slider.

4 DSV System Description and Experiments

The choice of the DSV system embedded in Sign4J has been driven by the necessity to provide local scores, i.e. scores for each point of the signature sample.

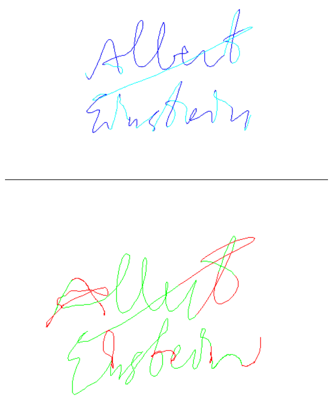


Fig. 2. Example of the local feedback mechanism. The top part is the original signature and the bottom part is the forgery where the red (dark) parts corresponds to region having produced scores below a given local threshold.

We have then chosen to implement a system based on local feature extraction and Gaussian Mixture Models (GMMs) in a similar way as in [7] and [2]. GMMs are also well-known flexible modelling tools able to approximate any probability density function. For each point of the signature, a frontend extracts 25 dynamic features as described in [4]. The frontend extracts features related to the speed and acceleration of the pen, the angles and angles variations, the pressure and variation of pressure, and some other derived features. The features are mean and standard deviation normalized on a per signature basis. GMMs estimates the probability density function $p(x_n|M_{client})$ or *likelihood* of a D-dimensional feature vector x_n given the model of the client M_{client} as a weighted sum of multivariate gaussian densities :

$$p(x_n|M_{client}) = \sum_{i=1}^{I} w_i \mathcal{N}(x_n, \mu_i, \Sigma_i) \tag{1}$$

in which I is the number of mixtures, w_i is the weight for mixture i and the gaussian densities $\mathcal{N}$ are parameterized by a mean $D \times 1$ vector μ_i, and a $D \times D$ covariance matrix, Σ_i. In our case, we make the hypothesis that the features are uncorrelated so that diagonal covariance matrices can be used. By making the hypothesis of observation independence, the global *likelihood* score for the sequence of feature vectors, $X = \{x_1, x_2, ..., x_N\}$ is computed with :

$$S_c = p(X|M_{client}) = \prod_{n=1}^{N} p(x_n|M_{client}) \tag{2}$$

The likelihood score S_w of the hypothesis that X is **not** from the given client is here estimated using a world model M_{world} or *universal background model* trained by pooling the data of many other users. The likelihood S_w is computed in a similar way, by using a weighted sum of gaussian mixtures. The global score is the log-likelihood ration $R_c = \log(S_c) - \log(S_w)$. The local score at time n is the log-likelihood ratio $L_c(x_n) = log(p(x_n|M_{client})) - log(p(x_n|M_{world}))$. The training of the client and world models is performed with the Expectation-Maximization (EM) algorithm [1]. The client and world model are trained independently by applying iteratively the EM procedure until convergence is reach, typically after few iterations. In our setting, we apply a simple binary splitting procedure to increase the number of gaussian mixtures to a predefined value. For the results reported here, we have used 64 mixtures in the world model and 16 in the client models.

Experiments have been done with online signatures of the public MCYT-100 database [5]. This mono-session database contains signatures of 100 users. Each user has produced 25 genuine signatures, and 25 low-force trained forgeries are also available for each user (named as *skilled* forgeries in the database). These forgeries are produced by 5 other users by observing the static images and training to copy them.

We have used Sign4J and the procedure described earlier to produce brute-force trained forgeries for 50 users of MCYT-100. The training time to train on one user was on purpose limited to 20 to 30 minutes. After the training phase, 5 imitation samples were produced by the forgers. We have to note here that our acquisition device (Wacom Cintiq 21UX) is different to the MCYT-100 signature acquisition device (Wacom A6 tablet). We had to uniform the ranges and resolutions of the records to be able to perform our tests. Better brute-force forgeries could potentially be obtained by using strictly the same devices.

The performances of a baseline DSV system, similar to the one embedded in Sign4J, were then evaluated using three sets of signatures: a set of random forgeries (RF), the set of low-force forgeries (LF) included in MCYT-100 and the brute-force forgeries (BF) generated with Sign4J. Equal Error Rates (EER) of 1.3%, 3.0% and 5.4% are obtained respectively for RF, LF and BF forgeries. As expected, low-force forgeries are more easily rejected than brute-force forgeries, with a significant relative difference of 80%.

5 Conclusions and Future Work

We have introduced a procedure to generate brute-force signature forgeries that is supported by Sign4J, a dedicated software. The main novel feature of Sign4J lies in a link with an embedded DSV system. The DSV system allows to implement a feedback mechanism that let the forger see how close he or she was to break the system. Sign4J also exploit the local scores of the DSV system to indicate to the forger what are the potential parts of the signature where improvements are needed. A set of forgeries has been generated on the MCYT-100 database, by following our forgery procedure and by using Sign4J. These forgeries have been compared to the low-force forgeries available in MCYT-100, measuring Equal Error Rates obtained with our baseline verification system. Although the training time has been limited to 20 to 30 minutes per signature, the brute-force forgeries are measured to be significantly more difficult to reject than the low-force forgeries.

In potential future work, we would like to investigate better rendering of the local feedback that reveals noisy when the forger performs equally well in all areas of a signature. Also, more precise feedback about the features to improve could be possible, i.e. not only answer the question "where to improve", but also "how to improve". Another possible amelioration of SIGN4J is in the play-back of the angles and pressure which are currently difficult to analyze and reproduce. Finally, an important area of research would be to leverage on the knowledge acquired in this project and to investigate how DSV systems can be improved in order to diminish the potential risks of such brute-force forgeries.

References

1. A.P. Dempster, N.M. Laird, and Rubin D.B. Maximum likelihood from incomplete data via the em algorithm. *Journal of Royal Statistical Society*, 39(1):1–38, 1977.
2. A. Humm, J. Hennebert, and R. Ingold. Gaussian mixture models for chasm signature verification. In *Accepted for publication in 3rd Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms*, Washington, 2006.
3. F. Leclerc and R. Plamondon. Automatic signature verification: the state of the art–1989-1993. *Int'l J. Pattern Recognition and Artificial Intelligence*, 8(3):643–660, 1994.
4. B. Ly Van, S. Garcia-Salicetti, and B. Dorizzi. Fusion of hmm's likelihood and viterbi path for on-line signature verification. In *Biometrics Authentication Workshop*, May 15th 2004. Prague.
5. J. Ortega-Garcia, J. Fierrez-Aguilar, D. Simon, J. Gonzalez, M. Faundez-Zanuy, V. Espinosa, A. Satue, I. Hernaez, J.-J. Igarza, C. Vivaracho, D. Escudero, and Q.-I. Moro. Mcyt baseline corpus: a bimodal biometric database. *IEE Proc.-Vis. Image Signal Process.*, 150(6):395–401, December 2003.
6. R. Plamondon and G. Lorette. Automatic signature verification and writer identification - the state of the art. *Pattern Recognition*, 22(2):107–131, 1989.
7. J. Richiardi and A. Drygajlo. Gaussian mixture models for on-line signature verification. In *Proc. 2003 ACM SIGMM workshop on Biometrics methods and applications*, pages 115–122, 2003.
8. Claus Vielhauer. *Biometric User Authentication for IT Security.* Springer, 2006.
9. F. Zoebisch and C. Vielhauer. A test tool to support brut-force online and offline signature forgery tests on mobile devices. In *Proceedings of the IEEE International Conference on Multimedia and Expo 2003 (ICME)*, volume 3, pages 225–228, Baltimore, USA, 2006.