electronics

MDPI

Article

A Secure Live Signature Verification with Aho-Corasick Histogram Algorithm for Mobile Smart Pad

Kuo-Kun Tseng [1], He Chen [1], Charles Chen [2,*] and Charinrat Bansong [1]

[1] Department of Computer Science and Technology, Harbin Institute of Technology, Shenzhen Graduate School, Shenzhen 518055, China; kktseng@hit.edu.cn (K.-K.T.); ch1420@126.com (H.C.); charinrat_b@hotmail.com (C.B.)

[2] School of Computer Science and Engineering, Minnan Normal University, Zhangzhou 363000, China

* Correspondence: charles.chen@mnnu.edu.cn; Tel.: +86-159-1974-1471

Abstract: There is a long history of using handwritten signatures to verify or authenticate a "signer" of the signed document. With the development of Internet technology, many tasks can be accomplished through the document management system, such as the applications of digital contracts or important documents, and more secure signature verification is demanded. Thus, the live handwriting signatures are attracting more interest for biological human identification. In this paper, we propose a handwriting signature verification algorithm by using four live waveform elements as the verification features. A new Aho-Corasick Histogram mechanism is proposed to perform this live signature verification. The benefit of the ACH algorithm is mainly its ability to convert time-series waveforms into time-series short patterns and then perform a statistical counting on the AC machine to measure the similarity. Since AC is a linearly time complexity algorithm, our ACH method can own a deterministic processing time. According to our experiment result, the proposed algorithm has satisfying performance in terms of speed and accuracy with an average of 91% accuracy.

Keywords: live signature; signature verification; Aho-Corasick algorithm

check for updates

Citation: Tseng, K.-K.; Chen, H.; Chen, C.; Bansong, C. A Secure Live Signature Verification with Aho-Corasick Histogram Algorithm for Mobile Smart Pad. *Electronics* **2021**, *10*, 1337. https://doi.org/10.3390/electronics10111337

Academic Editors: Alfredo Arcos Jiménez, Fausto Pedro García Márquez and Caroline Leonore König

Received: 14 March 2021
Accepted: 29 May 2021
Published: 2 June 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Signature verification is a very important technology with a long history. With the development of electronic and information technology, more secure applications are requested to be verified with live handwriting signatures. Although conventional handwriting is a two-dimensional image, in this research, four live waveforms will be used for a more secure and robust verification.

As shown in Figure 1, live features of the handwriting signature are used for verification with a document management system. That signature can be obtained by writing on an edge smart pad with an electromagnetic pen. There are four live waveforms that would be obtained, the X-axis value, the Y-axis value, pressure on the board, and velocity. Then these four elements would be used to process the verification algorithm. Unlike fingerprint and human face, live waveforms are more difficult to be copied by visual sensors.

In this paper, we propose a new non-exact verification algorithm with an AC histogram (ACH) method which is an improvement from Aho-Corasick (AC) algorithm. AC is a state transition structure proposed by Alfred V. Aho and Margaret J. Corasick in 1975 [1] and is an exact string-matching algorithm for text searching. Our ACH classification algorithm is applied to user verification with handwriting signature. Since the data of live handwriting signature is a time-series waveform, the ACH algorithm mainly converts time-series waveforms into time-series short patterns, and then performs a statistical counting on an AC machine to measure the similarity. This handwriting signature has been applied to an OA system. Since AC is a linear time complexity algorithm, our ACH can own a deterministic processing time, and according to our experiment, the result

shows the total average accuracy of the ACH algorithm is 91%, which is satisfactory for signature verification.

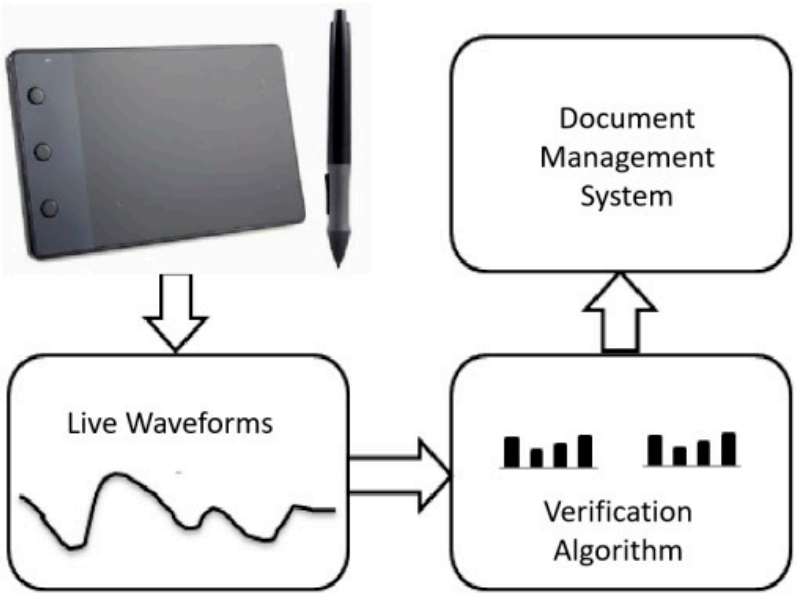

Figure 1. Live handwriting signature verification for smart pad.

The structure of this paper is as follows: Section 2 describes the proposed algorithm of the AC histogram. Section 3 is the discussion of the ACH algorithm and the parameter adjustment. Section 4 presents the analysis of the experimental results of our proposed algorithm. The conclusion is stated in the final section.

2. Related Work

In the e-commerce arena, many transactions are conducted on the Internet, such as e-banking, e-mail accounts, web shopping, credit card online payments, and electronic documents. How to prevent the considerable volume of computerized/digitized and exchanged online documents and transactions from being stolen becomes a great security concern to many organizations (Jung et al. [2]; Oppliger [3]; Ryan and Bordoloi [4]). According to the research of O'Gorman [5], authentication systems can be broadly classified into the following three categories: password-based authentication schemes, token-based authentication schemes, and biometric-based authentication schemes. Our research focuses on biometric-based authentication, which is a system that relies on physiological traits of the user (e.g., handwriting signature, fingerprint, face, palm print, voice, retina).

In this survey work, we focus on the research of handwriting signature authentication. In fact, signature verification is a difficult pattern problem because the intra-class variation could be large (Zou et al. [6]). Handwritten signature verification is the process of confirming the identity of a user based on his or her handwritten signature. A signature verification system must be able to detect forgery signatures and reduce rejection of genuine signatures at the same time. Efforts to ensure that only the right people have authorization to sign the documents have led to the development of automatic signature recognition and verification systems.

According to the research on electronic signature verification, four major aspects of handwriting signature verification have been identified: (i) signature data acquisition, (ii) preprocessing of the signature data, (iii) feature extraction, and (iv) verification. Table 1 is a summary of related works of handwriting signature.

Table 1. Related works of handwriting signature.

Subject	Phase	Methods (Key Reference)
Signature Verification	Data Acquisition	Online (Fallah et al., 2011; Wu et al., 1998) Offline (Fallah et al.; Mizukami et al. 2002)
	Preprocess	Filtering (Sabourin and Plamondon,1988) Normalization (Braut and Plamondon,1993) Noisy Remove (Dimauro et al.,1993) Localization (Plamondon et al., 1992) Skeletonization and Smooth (Ammar et al. 1990; Wirtz, 1995) Templates (Sae-Bae et al., 2018)
	Feature Extraction	Position (Fallah et al., 2011; Lee et al., 1996) Direction (Dimauro et al. 1993; Mizukami et al., 2002; Yoshimura and Yoshimura, 1992) Pressure (Crane and Ostrem,1983) Transform Based (Castellano et al., 1998; Lam and Kamins, 1989; Letjman and George, 2001) Pen up and down (Fallah et al., 2011) Acceleration and Pressure (Liu et al., 1979) Statistic Based (Nelson et al.,1994) Logarithmic Spectral (Wu et al., 1998) Wavelet Algorithm (Letjman and George,2001) Discriminative Feature Selection (Xia et al., 2018) Improved Discriminative Region Selection (Mandal et al., 2019)
	Verification	Euclidean Distance (Sabourin et al., 1997; Nelson et al., 1994; Dimauro et al., 1997) Component-Oriented (Dimauro et al., 1994) Dynamic Time Warping and Skeletal Tree (Perizeau, M. and Plamondon,1990) Split and Merge (Wu et al., 1997) Elastic (Sabourin et al., 1997) String Matching (Jaim et al., 2002) Neural Network (Lam and Kamins,1989; Tseng and Huang, 1992) Correlation (Wirtz, 1995; Zhu et al., 2010; Zhang et al., 2009; Chen et al.,2018) Hidden Markov Model (Fuentes et al., 2002) Fuzzy Set (Doroz et al.,2018) DTW with SCC (Xia et al., 2018) Recurrent Neural Networks (RNN) (Tolosana and Vera-Rodriguez, 2018) Faster R-CNN (Deng et al., 2020) Convolutional neural networks (CNN) (Altwaijry and Al-Turaiki, 2021) Dilated Temporal Convolution Network (Sharma and Jayagopi, 2021)

In data acquisition, two approaches of signature verification can be classified. They are online (dynamic) verification and offline (static) verification [7]. The main difference between these two methods is the way they obtain the signature data. Online signature verification method obtains the signature data, such as the position of handwriting trace, velocity, acceleration, pressure, and force signals, during the handwriting. On the other hand, offline handwriting obtains the signature trace by scanner or camera. In this way, it consists of static 2D image information only. Online handwriting verification processes the data in both spatial and time domain, while an offline algorithm processes the data only in the spatial domain [7]. Fallah et al. [7] used a Wacom Intuos tablet to get the coordinate, timing information, pen orientation, and pressure of each sampling period. Chen et al. [8] used a Yuntab electromagnetic tablet (10.1 inch) with size of 262 × 175 mm to capture dynamic data of signature such as x–y position, pressure of pen, and speed of each sampling period. For the data acquisition of offline signature, Ismailc and Gad [9] used a scanner to scan the signatures into the computer in black and white using 200 DPI (dot-per-inch) resolution. Paper templates of 75 signers were digitized with a scanner at 600 dpi. These binary image data are used for system development and evaluation.

After obtaining the handwriting signature data, some preprocessing may be needed. A filter may be used to remove some noise or interference information for the data coming from the online method. Then, standardized processing may be applied to the signature

data which are in spatial and time domains. For the offline method, data are captured by optical devices first, and then those digital image data are converted into bit pattern. Offline signature systems are useful in automatic verification of the signature of bank checks and documents. Online signature systems record the motion of the stylus. They use the dynamic information of the signature which is captured during the signature. Dynamic features include the number and order of strokes, velocity and acceleration of signature, and pen pressure at each point, which are more difficult to duplicate than the static shape of the signature. This makes the signature more unique, individual, sufficiently stable, and difficult to forge. Therefore, online signature verification can usually achieve better performance than offline instances [8].

In the preprocessing algorithm, Dimauro et al. [10] preprocessed the handwriting trace between pen down and pen up. In their opinion, the handwriting consists of several writing units, which are the components between pen down and pen up. Plamondon et al. [11] propose a preprocessing algorithm based on the curve speed and the angular velocity. The basic elements of handwriting are sampled as vectors and strings. Brault and Plamondon [12] preprocess some important points from the handwriting trace. They firstly assign a weight value to each point of handwriting and then extract out the most important points as handwriting features. Dimauro et al. [13] applied the DP algorithm to match the test signature and assigned signature. Only the matched points in the test signature were extracted. Ammar et al. [14] used a tree structure to describe the handwriting components. Sabourin and Plamondon [15] presented a new version of the centroidal linkage region growing with a merge algorithm. Sae-Bae et al. [16] proposed an online signature template method named distinctiveness, complexity, and repeatability to assess the characteristics of signature samples.

In the feature extract algorithm, there are two common methods: parameter and function. The parameter feature described method uses a vector containing several parameters. Each element of the vector is one feature value. While using a function as a feature, the handwritten signature is characterized as a function of time. Wu et al. [17] proposed a logarithmic spectral similarity measure. They extracted features from the logarithmic spectral of handwriting trace and then matched the logarithmic spectra of test signature and assigned signature. Mizukami et al. [18] proposed a displacement feature extract method using a minimum function to shift the signature extremum value. Liu et al. [19] proposed a new handwriting signature verification that is based on the waveform of acceleration and pressure of the handwriting. Yoshimura and Yoshimura [20] used another online method to extract the feature. In their feature extract algorithm, the direction of the points on the handwriting trace is extracted. Crane and Ostrem [21] used the pressure information as the feature. Lee et al. [22] proposed a method using the X- and Y-axes; first, they described the waveforms of X and Y, and then they extracted 42 parameters from these two waveforms. Nelson et al. [23] proposed a statistical method to extract features. Castellano et al. [24] used one dimension from the handwriting trace and then processed it as a waveform. Lam and Kamins [25] firstly extracted some features from the handwriting signature such as shape, time and speed. Then, they applied some transform functions to these features. Letjman and George [26] used the wavelet algorithm to extract features and then used a neural network to classify them. Xia et al. [27] used the discriminative feature selection methodology to extract and select the feature characteristics of a live signature. The extraction process was divided into two steps: Step 1 is the consistent feature selection. Step 2 is the discriminative feature selection with two methods: feature selection based on factorial experimental design (FED) and feature selection based on orthogonal experimental design (OED). Mandal et al. [28] proposed an improved discriminative region selection methodology with two stages to extract and select the feature characteristics of a live signature. Stage 1 is the estimation of the average template of each character class. Stage 2 is the identification of DR by calculating DTW between such two templates.

The purpose of the verification algorithm is to match the features of a test signature and the features of an assigned signature. Dimauro et al. [29] combined three different

algorithms for signature verification in their paper. Sabourin et al. [30] used the Euclidean distance as the matching algorithm in their paper. It computes the distance between two signatures. Wirtz [31] and Chen et al. [8] used the correlation coefficient to calculate the relevancy. A dynamic time warping and matching skeleton tree verification method was proposed by Perizeau and Plamondon [32]. Wu et al. [33] used a split and merge matching algorithm to verify the live signature. Jaim et al. [34] used string matching to verify online signatures. Tseng and Huang [35] used a neural network as the classifier. Fuentes et al. [36] used the hidden Markov model to recognize signatures. Zhu et al. [37] used correlation test-based nonlinear adaptive noise cancellation (ANC) to recognize a live signature. Zhang et al. [38] used correlation test-based neural network validation to recognize signatures. In addition to the above signature verification algorithms, some new improved algorithms exist, such as signature stability measure method based on fuzzy set theory, which was proposed by Doroz et al. [39]; DTW with SCC for signature distance dissimilarity evaluation, which was proposed by Xia et al. [27]; a novel writer-independent online signature verification systems based on recurrent neural networks (RNNs) with a Siamese architecture, which was proposed by Tolosana and Vera-Rodriguez [40]. A faster Region-CNN (R-CNN) method to automatically locate cracks from the handwriting scripts was proposed by Deng et al. [41]. Based on convolutional neural networks (CNNs), Altwaijry and Al-Turaiki [42] also proposed an automatic handwriting recognition model to verify Arabic letters. Sharma and Jayagopi [43] presented a dilated temporal convolution network (DTCN) to recognize cursive handwriting from line-level images. Their proposed architecture combines 2-D convolutions and 1-D dilated noncausal convolutions with a connectionist temporal classification (CTC) output layer. This method can offer a high parallelism with a smaller number of parameters and is suitable for low-resource and environment-friendly deployment.

3. Proposed Basic Algorithm

Before we introduce the algorithms and equations, we present Table 2, which is an explanation of the relevant notations.

In this section, we propose a process algorithm for online signature verification. Figure 2 is the structure of the proposed signature verification process. First, we obtain the online signature data via an electromagnetic tablet (10.1 inch) with size 262 × 175 mm to capture dynamic data of signature. The data format of each signature point contains six aspects of information: (1) the ID of each point, (2) the X-axis value of the corresponding point, (3) the Y-axis value of the corresponding point, (4) the distance between pen and the pad screen, (5) pressure value, and (6) velocity (time). Figure 3 is the sample of signature points. Second, we preprocess those data and convert them into waveform style. Third, we convert those waveform data into patterns. Fourth, we build an AC state tree. Fifth, we match the AC state tree. Sixth, we complete the identification process.

3.1. Aho-Corasick Algorithm

The AC algorithm was proposed by Aho and Corasick in 1975 [1]. It is a classic multipattern matching algorithm; it was firstly used in the Bell Labs library searching system and has since been widely used in other fields. The AC algorithm is mainly composed of three functions: goto function, failure function, and output function. The time complexity of the AC pattern matching algorithm is $O(n)$. This algorithm has nothing to do with the number of pattern strings and the length of each pattern string. No matter which pattern P appears in the text string T, each character in T must be input into the state machine. So whether it is in the best case or worst case, the time complexity of the AC pattern matching algorithm is always $O(n)$. If including preprocessing, the time complexity of the AC algorithm is $O\ (M + n)$, where M is the length of string for all patterns.

Aho-Corasick (AC) automata can be defined as 5-tuple symbols $(Q, \sum, q_0, \delta, F)$ where Q is a finite set of states, $\sum$ is a finite set of input symbols (also called the alphabet), $\delta(\mathrm{Q} \times \sum \rightarrow \mathrm{Q})$ is a transition function, q_0?? is a start state, and F is a set of final states. The

AC algorithm has preprocessing and matching stages. Before performing AC matching, one needs to construct a state machine from the patterns in the preprocessing stage. In an adaptation from the example in [1], Figure 4a–c describes the AC's three major functions for patterns "TEST", "THE", and "HE". Figure 4d describes the AC table implementation.

The first *Goto* Function shown in Figure 4a starts with an empty root node and adds states to the state machine for each pattern. That *Goto* function is a tree structure that shares common prefixes with all of the patterns. During the matching, the *Goto* function is traversed from one state to the other with the text byte by byte.

Table 2. Notations of the equations.

Symbol	Description	Symbol	Description
T	Text string	$\mathcal{H}_T^{(i)}$	A vector of the outputs of ACH searching function
P	The patterns appearing in the text string	$U_p^{(i)}$	The pattern set of AC state tree
M	The length of string for all patterns	$\mathrm{p_i}$	The ith pattern
$O(n)$	Time complexity is linear to length n	$U(p_i)$	The union of pattern $\mathrm{p_i}$ added into the set
Q	A finite set of states	$\delta[P_m]_T^{(i)}$	A function to find a pattern in the tree
$\sum$	A finite set of input symbols (also called the alphabet)	$\mathcal{C}[P_m]_T^{(i)}$	A function to find which pattern
$\delta(Q \times \sum \rightarrow Q)$	A transition function of AC	H_e	An operation to establish the histogram
q_0??	A start state of AC	$d(i)$	The absolute difference of particular pattern
F	A set of final states of AC	$\mathcal{H}_S^{(i)}$	The histogram of template data.
R^d	The assigned data of ACC	$\mathcal{H}_T^{(i)}$	The histogram of one test data
T^d	The test data of ACC	TF_S^i	The total difference between two histograms
P_r	The patterns of string	ω	A parameter to describe the difference
P_T^i	The ith pattern for test	$\mathrm{U}\mathbb{N}(S^x, S^T, S^P, S^T)$	A neural network classifier to merge the four elements of signature into one vector
$\mathrm{Tree^d} = \mathfrak{T}\left(\mathrm{P_r^i}\right)$	A function to construct a state tree from P_r	$\mathrm{U}(S^x, S^T, S^P, S^T)$	Merging four elements of signature into one vector
T^d	The text data of input for searching	$\mathfrak{E}(U_j^{length} x_i)$	Performs a conversion from data to signature
$C_{S_D^{(j)}}^{(i)}$	The counted number of text data.	$\mathrm{V} = \mathfrak{X}(\mathrm{e_i} \rightarrow 1)$	The voting methods to merge these four results
$\delta[\mathrm{P_n}]_\mathrm{D}^{(\mathrm{j})}$	A function to search base of the tree of test data	r	The resolution, sample rate of input waveform
$\phi_T^{(i)}$	The reading data of test file	h_i	A height of input waveform
M_c	The function for counting	W	Waveform of signature

The second is the *Output* function shown in Figure 4b that needs a table to store the matched patterns with their corresponding states in the *Goto* tree. The *Output* function records a matched state for a matched pattern when the current state is matched during the visiting. The third is the *Failure* function, which is constructed from the *Goto* function as shown in Figure 4c. Let us define the *depth* of a state s in the *Goto* tree as the length of the shortest path from the start state to s. During the construction, the failure function for the states of depth d is computed from the failure function for the states of depth less than d. The states of depth d can be determined from the nonfail values of the *Goto* function of

the states of depth $d-1$. During the matching, the *Failure* function is used when a match fails after a partial match test. After the construction of the machine, the AC state machine is traversed from the current node to the next node according to the input byte.

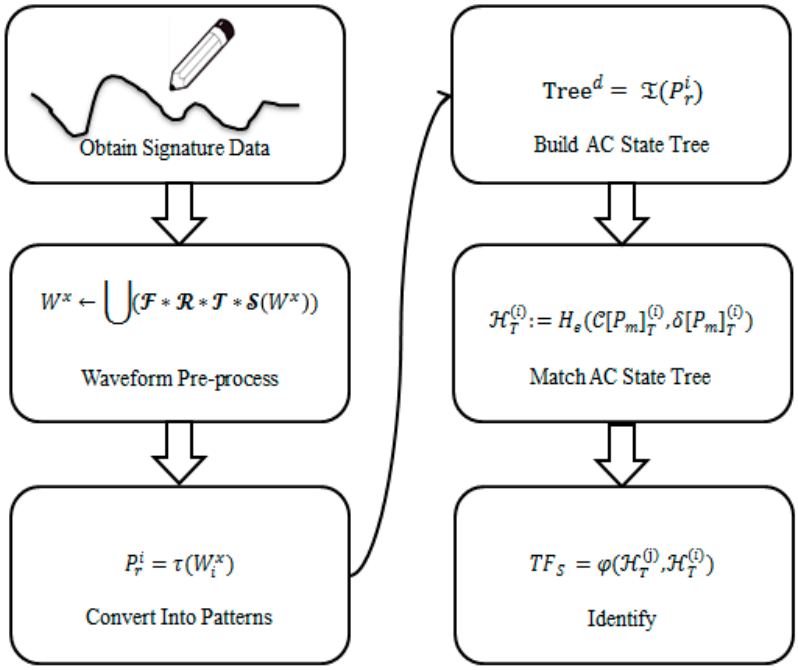

Figure 2. The structure of handwriting signature verification.

ID=181x=2555y=2085b=3p=0sec=0milisec=521ID=182x=2545y=2085b=3p=0sec=0milisec=528
ID=183x=2537y=2082b=3p=0sec=0milisec=539ID=184x=2537y=2082b=3p=0sec=0milisec=546
ID=185x=2541y=2075b=7p=4sec=0milisec=568ID=186x=2547y=2067b=7p=4sec=0milisec=575
ID=187x=2555y=2060b=7p=4sec=0milisec=581ID=188x=2566y=2052b=7p=4sec=0milisec=588
ID=189x=2580y=2043b=7p=5sec=0milisec=594ID=190x=2598y=2032b=7p=5sec=0milisec=601
ID=191x=2623y=2019b=7p=5sec=0milisec=608ID=192x=2651y=2004b=7p=6sec=0milisec=614
ID=193x=2683y=1988b=7p=8sec=0milisec=621ID=194x=2717y=1970b=7p=9sec=0milisec=628
ID=195x=2752y=1951b=7p=9sec=0milisec=634ID=196x=2790y=1930b=7p=9sec=0milisec=641
ID=197x=2829y=1907b=7p=10sec=0milisec=647ID=198x=2871y=1884b=7p=11sec=0milisec=654
ID=199x=2914y=1861b=7p=11sec=0milisec=661ID=200x=2957y=1837b=7p=12sec=0milisec=667
ID=201x=2998y=1812b=7p=13sec=0milisec=674ID=202x=3039y=1787b=7p=13sec=0milisec=680
ID=203x=3078y=1765b=7p=13sec=0milisec=687ID=204x=3115y=1745b=7p=13sec=0milisec=694
ID=205x=3151y=1727b=7p=13sec=0milisec=700ID=206x=3184y=1713b=7p=13sec=0milisec=707
ID=207x=3213y=1702b=7p=13sec=0milisec=713ID=208x=3237y=1696b=7p=13sec=0milisec=721
ID=209x=3257y=1694b=7p=14sec=0milisec=727ID=210x=3271y=1698b=7p=15sec=0milisec=733
ID=211x=3280y=1707b=7p=15sec=0milisec=740ID=212x=3283y=1720b=7p=15sec=0milisec=746
ID=213x=3281y=1738b=7p=15sec=0milisec=753ID=214x=3274y=1760b=7p=15sec=0milisec=760
ID=215x=3263y=1786b=7p=15sec=0milisec=767ID=216x=3248y=1815b=7p=14sec=0milisec=773I
D=217x=3232y=1845b=7p=14sec=0milisec=780ID=218x=3214y=1877b=7p=14sec=0milisec=787
ID=219x=3197y=1906b=7p=14sec=0milisec=793ID=220x=3180y=1932b=7p=14sec=0milisec=799
ID=221x=3165y=1955b=7p=14sec=0milisec=806ID=222x=3153y=1974b=7p=14sec=0milisec=813
ID=223x=3144y=1990b=7p=13sec=0milisec=820ID=224x=3139y=2002b=7p=12sec=0milisec=826
ID=225x=3137y=2013b=7p=12sec=0milisec=832ID=226x=3139y=2022b=7p=12sec=0milisec=839
ID=227x=3144y=2030b=7p=12sec=0milisec=846ID=228x=3153y=2036b=7p=12sec=0milisec=852
ID=229x=3165y=2042b=7p=13sec=0milisec=859ID=230x=3179y=2049b=7p=13sec=0milisec=865
ID=231x=3197y=2058b=7p=13sec=0milisec=872ID=232x=3214y=2068b=7p=13sec=0milisec=879
ID=233x=3233y=2079b=7p=13sec=0milisec=885ID=234x=3250y=2093b=7p=14sec=0milisec=893

Figure 3. Signature point samples.

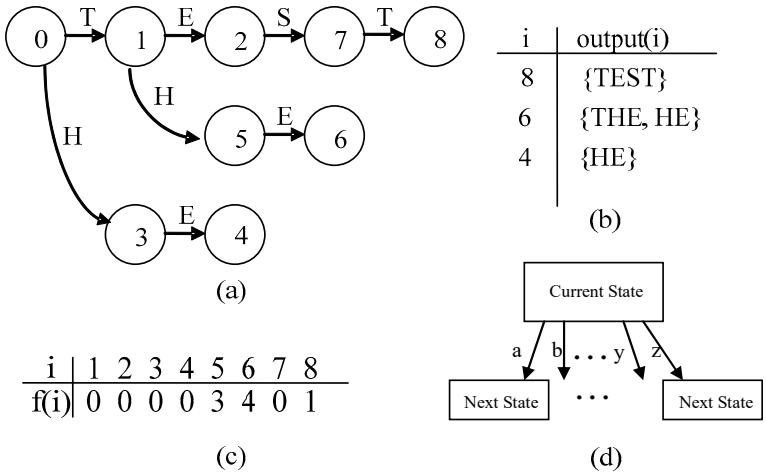

Figure 4. (**a**) *Goto* function. (**b**) *Output* function. (**c**) *Failure* function. (**d**) AC table implementation.

3.2. Basic Aho-Corasick Counting (ACC) Algorithm

AC counting (ACC) algorithm is an advanced pattern matching algorithm of Aho-Corasick multipattern matching. In the basic AC pattern matching algorithm, the aim is to determine if a particular pattern is in the constructed AC state tree. This AC pattern state tree is based on a given character text. The output of Aho-Corasick multipattern matching algorithm is a Boolean value that indicates whether the test string can be found in the text or not. However, the AC counting (ACC) algorithm is an algorithm that can output the similarity of two given data or vectors. A single Boolean value cannot stand for a similarity value. A single Boolean value can only indicate true or not true. Similarity of two data usually is in the interval of $[0\%, 100\%]$. The Aho-Corasick algorithm is a character pattern matching algorithm. The more patterns of the two given pieces of text are matching, the more similar they would be. Hence in the AC counting (ACC) algorithm, a counting value of matched patterns is obtained.

In order to calculate the similarity of two texts, one of them must be selected as the assigned data R^d, and the other one must be selected as the test data T^d. The assigned data R^d will firstly be constructed into a state tree. Firstly, the R^d data will be separated into patterns P_r. Then the AC algorithm is used to construct a state tree of P_r step by step. Equation (1) shows this processing step:

$$\text{Tree}^d = \mathfrak{T}\left(P_r^i\right) \tag{1}$$

where P_T^i is the ith pattern for the test.

After building an AC state tree, the text data T^d is input for searching. T^d is a string, and the AC state tree searches for patterns by reading the characters of T^d string one by one. In the searching function of the ACC algorithm, when a pattern in T^d is found, yhe counting value is set by Equation (2).

$$C_{S_D^{(j)}}^{(i)} := \sum_n 1\left\{M_c(\delta[P_n]_D^{(j)}, \phi_T^{(i)})\right\} \tag{2}$$

where $C_{S_D^{(j)}}^{(i)}$ is the counted number of test data. It is the base of the tree of $[P_n]_D^{(j)}$, and $\phi_T^{(i)}$ is the reading test data. The M_c function is based on Equation (3).

Equation (3) shows that if the counted number is higher than a given threshold, $\mathrm{M}_{(j)}^{(i)}$ would be 1, else if the counted number of one pattern is lower than the given threshold,

then $M_{(j)}^{(i)}$ would be 0. The value 1 means that the particular pattern is successfully matched, and value 0 means it fails to match. The given threshold would be obtained by training.

$$M_{(j)}^{(i)} = \begin{cases} 1, & C_{S_D^{(j)}}^{(i)} > \text{threshold} \\ & (i = 1, 2, 3, ...; j = 1, 2, 3, ...) \\ 0, & C_{S_D^{(j)}}^{(i)} \leq \text{threshold} \end{cases} \quad (3)$$

After searching the whole T^d test string data, a counting value will be obtained. The counting value is the number of patterns that exist in both assigned data R^d and test data T^d. The larger the counting value is, the more similar they are. It means that there are the same patterns in both data. Hence, the counting value is the similarity of two data.

In the verification algorithm, several data are tested to generate the ACC state tree. Each of the test data will be used to calculate similarity, and these similarity values are be compared to a trained threshold for identification.

3.3. Basic Aho-Corasick Algorithm with Histogram

The basic ACC algorithm can calculate the similarity of two character strings, but this similarity is not the real similarity of two strings. In the ACC algorithm, if a pattern in the test string can be found in the ACC state tree, the counting value would be increased. In an extreme case, the test string has only one pattern, but the pattern repeats hundred times. The ACC algorithm will also output a high similarity value of the assigned string and test string.

For two test strings having different patterns, every pattern has a different match number with the AC stat machine tree. The sum of all matched patterns for these two test strings may be the same. Therefore, both of them have the same counting of total time of patterns. They will be classified into the same class. However, these two strings are quite different. In order to eliminate the impact of total counting, a new counting algorithm is proposed. This new counting method not only counts the total time of patterns found in the AC state machine tree but also counts the number of every pattern in the test string found in the AC state tree. Therefore, we not only compare the total times of patterns that match the test string and AC state tree, but also the number of each pattern that matched. This method makes the similarity between two test strings more accurate. This matching method uses a histogram to show the counting of each pattern and compares every counting number one by one. Figure 5 shows the structure of the AC histogram algorithm.

The AC histogram (ACH) algorithm is an advanced version of the basic ACC (AC counting) algorithm. The ACC algorithm just outputs one value, the sum of all patterns matched. ACH algorithm not only outputs the sum number of all patterns but also, more importantly, outputs a histogram. This histogram describes every particularly matching pattern case. In the ACC algorithm, the similarity of two raw handwriting signature data is described by one single value. However, the ACH describes the similarity by a histogram, a vector. This means the information of similarity is enhanced from one dimension to multiple dimensions. In order to obtain the histogram information of similarity, a new structure is added to the AC state machine. When a test string is input, the ACH algorithm needs to record every pattern in the test string matched with the AC state machine tree. That means in the AC state machine tree searching step, a vector $\mathcal{H}_T^{(i)}$ is set up.

$$\mathcal{H}_T^{(i)} = his\_count[PATTERN\_NUM] \quad (4)$$

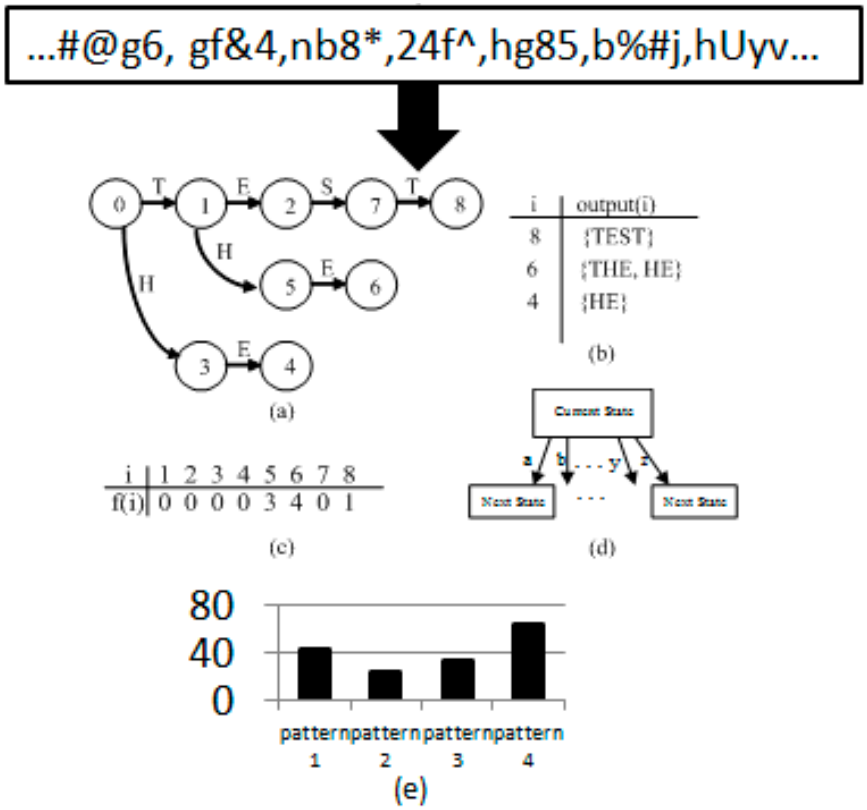

Figure 5. The structure of the AC histogram algorithm.

This vector counts every pattern found in the test string. PATTERN_NUM is the number of patterns in the AC state machine tree. Every position in the *his_count* vector indicates the number of a particular pattern found in the test string. Therefore, after searching the AC state machine tree, the test string will obtain a histogram of similarity to the template handwriting signature data. The method of counting the number of matched patterns in a particular position of the vector and obtaining the histogram information of similarity is stated below.

In order to record the number of particular patterns found in the test string after matching, the algorithm should know which pattern has been matched and where this matched pattern is stored. As shown in Figure 5, two new data structures are added to the AC state machine tree: one is the pattern set, and the other one is the pattern number index. At the construction step of the AC state machine tree, when a pattern is input, the add pattern step of ACH will search the existing state tree. If the processing pattern is a new pattern to the state tree, then pushes this new pattern into the pattern set. At the same time, ACH records the number ranked in the pattern set.

$$U_p^{(i)} = \mathrm{U}(p_i) \tag{5}$$

$U_p^{(i)}$ denotes the pattern set of the AC state tree. $\mathrm{U}(p_i)$ is the union of pattern p_i added into the set. That is $U_p^{(i)} = \{n_1, n_2, n_3, n_4, \ldots .n_i\}$, where n_i is the pattern p_i with *pattern_no i*. The pattern set belongs to the whole AC state machine tree; however, *pattern_no* belongs to every pattern node in the tree. Figure 6 shows how the ACH algorithm works.

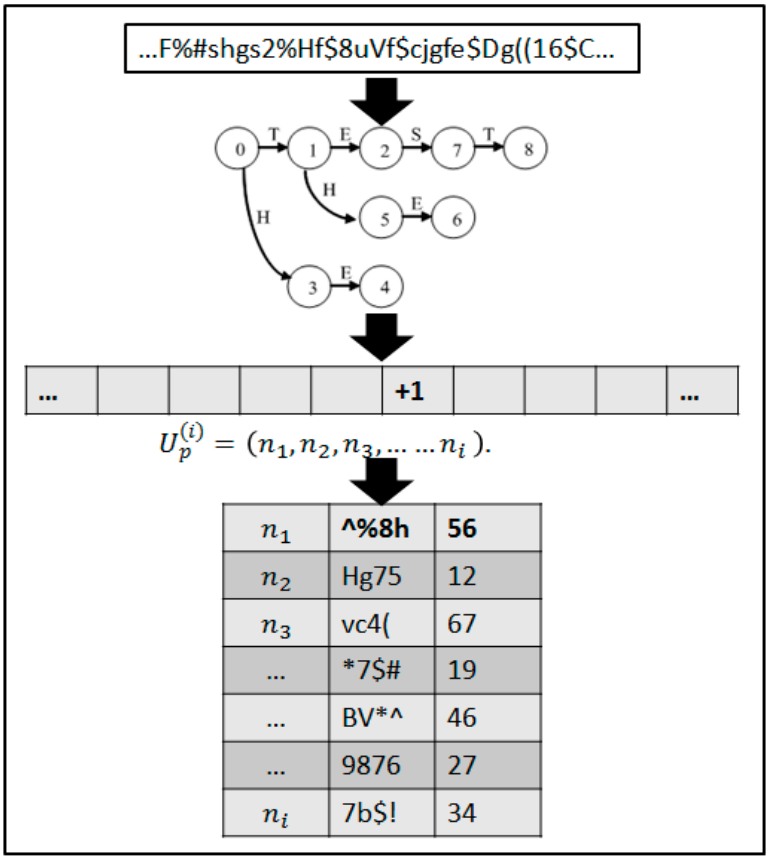

Figure 6. ACH histogram calculation.

When inputting a test string, the ACH search function obtains characters one by one and searches the ACH state machine tree. Test string characters go along the state tree; if a pattern is found out, the node of the character will be a finished node. When a substring in the test string matches a pattern in the state tree, in the basic ACC algorithm, it means the total time of pattern counting will increase by one, while in the ACH algorithm, the number of counting a particular pattern will increase by one. Thus a vector is needed to record this histogram information. This vector $\mathcal{H}_T^{(i)}$ is one of the outputs of the ACH searching function. Because every pattern node has a *pattern_no*, when matching a pattern through the value of *pattern_no*, we know which pattern is matched. Then we calculate the histogram by Equation (6).

$$\mathcal{H}_T^{(i)} := H_e\left(\mathcal{C}[P_m]_T^{(i)}, \delta[P_m]_T^{(i)}\right) \tag{6}$$

$\delta[P_m]_T^{(i)}$ is used to find pattern in the tree; $\mathcal{C}[P_m]_T^{(i)}$ is used to find which pattern. Operation H_e is to establish the histogram. The total histogram is Equation (7).

$$\mathcal{H}_T = \mathrm{U}\left(\left(\sum_{m=1}^{n} P_m\right) * \delta[P_m]_T\right) \tag{7}$$

After searching the whole test string, a histogram vector $his\_count[PATTERN\_NUM]$ is worked out. The sum of the elements in the histogram vector equals the ACC algorithm's total time counting. Figure 7 shows the pseudo-code of the ACH algorithm. This algorithm first initializes the initial *hisgtogram hi_countset*[], pattern number *patno*[], and sets the *current* node as the *root* node. Then it starts to read the test string of handwriting signature from the *testfile* into *buf*[] and obtain its *size*. In the main processing, it read out the *buf*[*i*] as a character one by one and then travels to the automata of ACH with *goto[ch]*. If the same

pattern is found, *out*[*current*] = *0*, and *his_countset*[*current- > pattern_no*] will be accumulated to increase the similarity between two signature strings. After reading all the *buf*[] for ACH, we can obtain the template of the histogram vector for later use.

```
his_countset[] = 0;
patno[] =-1;
current=root;
Readfile(testfilebuf[]);
size = strlen(buf);
for i = 0 to size
        ch = buf[i];
        if current->goto[ch]==NULL
                do
                        current = current->failure;
                while current->goto[ch]==NULL&&(current!=root);
        else
                current = current->goto[ch];
        if out(current)!=O
                his_countset[current->pattern_no]++;
                if current->goto[ch]==NULL
                        current =current->failure;
```

Figure 7. Pseudo-code of ACH for histogram computation.

After obtaining histogram information of matching, the similarity of two handwriting signature data would be calculated. Firstly, a section of raw handwriting signature data is selected for the template file. These handwriting signature data are processed to build the ACH state machine tree. At the same time, the histogram information of these template data is recorded. This template histogram vector records the numbers of every pattern that appeared in the ACH state tree. The similarity of the two compared histograms is calculated by following Equation (8):

$$di = |template(i) - test(i)|,\ i = 1, 2, 3, \ ... \tag{8}$$

$template(i)$ and $test(i)$ are the values of ith pattern in the histogram vector of state tree and test string. $d(i)$ is the absolute difference of a particular pattern. A parameter ω is set to describe the difference. If $d(i)$ is larger than ω, then the particular patterns in two handwriting signature data are quite different, so the total difference of these two handwriting signature data is enlarged.

$$\text{If } (i) > \omega, \text{ Let } TF_S^i = \varphi\left(\mathcal{H}_S^{(i)}, \mathcal{H}_T^{(i)}\right) \tag{9}$$

where TF_S^i is the total difference of two histograms with $d(i) > \omega$. $\mathcal{H}_T^{(i)}$ is a histogram of one test data. $\mathcal{H}_S^{(i)}$ is the histogram of template data.

After searching the whole test string and comparing the two histogram vectors, TF_S^i is the difference between two histogram vectors. It is the number of different elements in these two vectors, based on parameter γ. Finally, set a threshold to separate two different handwriting signature data based on TF_S^i. Figure 8 shows how ACH classifies two histograms.

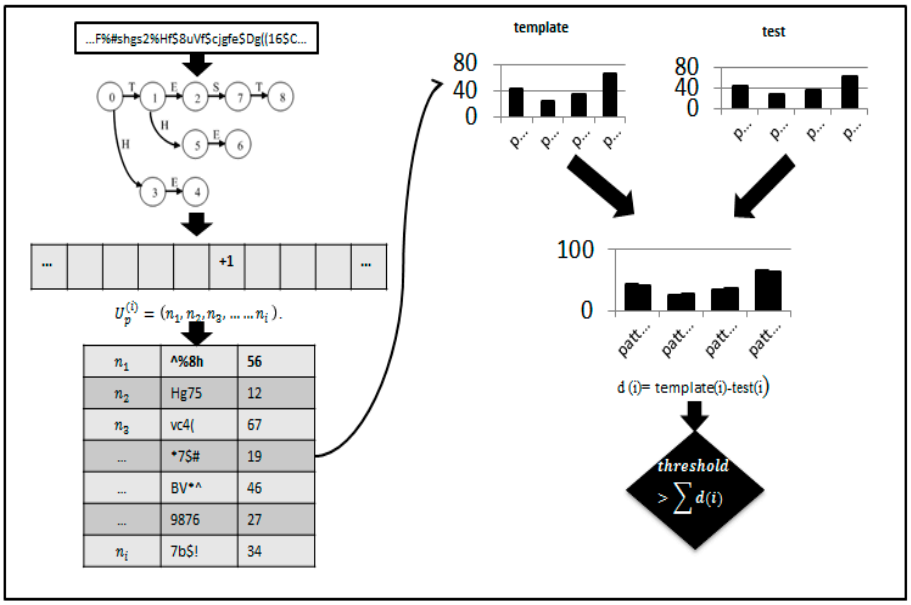

Figure 8. Advanced AC algorithm with histograms.

3.4. Voting Mechanism

The previous section introduced the ACH algorithm for calculating the similarity of two string data. In this section, we introduce the methodology for the verification of handwriting signatures. In the data acquisition stage, four elements of handwriting signature are used for verification. The four elements are X-axis value, Y-axis value, pressure value, and velocity. For the assigned signature string, each element will be used to construct an ACH state tree. Then in the training stage, several individual signature data are used to search for the state tree. Then for each element, the training stage would output a threshold. In the test stage, for every test signature string data, the search for the ACH state trees of the four elements will output four results. These four results stand for whether the special element is accepted as the assigned signature or not. So there are four judgments for deciding the identification of the test signature data. One of the methods to merge these four results is voting. Equation (10) shows this voting method.

$$\mathrm{V} = \mathfrak{X}(e_i \rightarrow 1) \tag{10}$$

This means, in these four results, one acceptant result gives one vote. We can set the number of acceptant ballots, such as more than two ballots. Another voting method is giving different weights for different elements. As X- and Y-axes may have more contribution than pressure or velocity, one result of X- or Y-axis would vote for two ballots, and the result of pressure or velocity remains for one ballot, setting more than 3 ballots as the acceptance. In this way, if X- or Y-axis accepts the test signature, the final result will have more chance to accept.

3.5. Merging Four Elements

The voting mechanism algorithm processes the four elements of signature data individually and works out the final result by voting. In this way, the ACH algorithm will be run four times for these four elements. From the introduction of the ACH algorithm, the ACH algorithm is used to match two character strings. These two strings are expressed in vectors instead of building four ACH state trees and then searching these four state trees separately. Therefore, we propose a merging algorithm that puts four elements of signature into one vector. Equation (11) shows this procedure.

$$S^m \Leftarrow \mathrm{U}\left(S^x, S^T, S^P, S^T\right) \tag{11}$$

After the merging procedure, the handwriting signature will have only one vector which represents the original four elements. In this way, only one state tree will be built, searched, and used to generate the final result.

3.6. ACH with Neural Network

The previous sections introduce two decision criteria. Both of them calculate the pass rate by comparing two handwriting signature data and use the Euclidean distance of the histograms of two signature strings for identification. Another way for classification and identification is to apply a classifier. In our experiment, a neural network classifier is applied. The previous steps are the same as the voting mechanism. The input of the neural network is the histogram of four elements of signature data. Every bar value of the histogram is a feature of the signature element. Then, the neural network classifier is used to generate the voting result. We call this neural network voting, which classifies the four elements separately.

Another method is to use the merging algorithm to merge the four elements of signature into one vector. Then, the merged histogram (vector) is inputted into the neural network classifier and generates the result. Equation (12) shows this procedure.

$$S^m = \mathbb{UN}(S^x, S^T, S^P, S^T) \tag{12}$$

3.7. Preprocessing of Waveforms of Four Elements

Before applying the ACH algorithm, the raw signature data should undergo some preprocessing. The preprocessing not only converts the signature data for ACH algorithm input but also processes the signature for better matching. The preprocessing steps are as follows:

(a) Shift all the handwriting signature words, placing the left of the signature words on the left of the screen and the top of the signature on the top of the screen. This is done because it cannot be guaranteed that the signature will in the same position on the screen each time people write their signature. In order to reduce the error caused by different signature positions, before verifying the signature, it needs to be shifted to the same position in the screen. In our handwriting signature verification algorithm, we shift all signatures to the top-left side of the screen. That is, the left-most point of the signature will be shifted to the left-most point of the screen, the 0 of the X-axis of the screen; the top-most point of the signature is shifted to the top-most point of the screen, to the 0 of Y-axis of the screen.

(b) Scale the number of signature points. The number of signature points changes each time a person writes their signature. In order to match the test signature data and the assigned signature data, the numbers of points should be the same. In this paper, both the test signature data and the assigned signature data are scaled to 1000 points. If the original signature data has less than 1000 points, it will be enlarged to 1000 points by inserting some mean value of the two original neighbor point values. If the original signature data has more than 1000 points, the redundant points will be removed. After this step of the process, both the test *signature* (*dst*) and assign *signature* (*src*) will have the same number of points, facilitating the processing of signature verification.

(c) Remove some points in the signature data that are protruding. When people are writing words, they may draw some stroke much longer or more protruding than the whole word. These protruded strokes influence the whole font style, although the number of these strokes is very low in the signature. Especially when scaling the size of signature words, these protruded strokes would cause a large error. When scaling all the words into the same size, these protruded strokes will occupy the main area of the whole word. However, these strokes have no meaning. The method to remove these protruded strokes is to apply the median height of every stroke. Both the X-axis and Y-axis would be used to calculate the median height. Twice the median height is used for the height limit. Because the number of protruded strokes is very few, the median height of the whole word depends little on the protruded strokes. After this step, the protruded strokes are removed.

(d) Scale the size of signature words. Each time a person writes the same word, the size of the word will be different. Even the same word written by the same person will have a similar shape but different size. Therefore, the signature words must be scaled to the same size. In our experiment, the size of the X-axis is scaled 2 times bigger than the Y-axis, because people usually write words from left to right.

4. Discussion of ACH Parameter Adjustment

The ACH algorithm has several parameters that can be adjusted. Different parameter values impact deeply on the classification result. In the experiment of this paper, three main parameters are adjusted; some of them impact the result greatly, some of them have little impact on the result. In the adjustment of different parameters, they are varied separately. That means when one of the parameters is adjusted, the other two parameters stay constant.

4.1. Sample Rate of Waveforms

The sample rate γ adjustment is taken before preprocessing. It is the first step, immediately after input the string data. The input string data are the whole data of signature, and we can take samples. Different samples represent the different levels of precision of data. In our experiment, the result shows that high precision does not result in high accuracy.

4.2. Resolution of Character Transform

Resolution of character transform is the height of waveform that converts into characters. The input data of the ACH algorithm are vectors that store the height of each point in the waveform. In order to apply the ACH algorithm, the integer values must be transformed into characters. Different heights of waveforms would be transformed into different characters. The resolution is the range of a height that converts into one character. If a point of a waveform is in the height which is $h_i = [i,\ i+1] * r$, where the r is the resolution, then all points that have a height of h_i would be converted into the same character. The adjustment of resolution is to vary the resolution and test the accuracy.

4.3. Length of ACH Pattern

In the ACH algorithm, an ACH state tree must be built first. In building the ACH state tree, patterns must be converted from data to signature. The patterns consist of the characters. As previous sections mentioned, after converting the signature data into characters according to the resolution, the continuing several characters would be picked up, and these characters would compose a pattern with a specific *length*. Then, shift one character and pick another group of characters to compose another pattern. Equation (13) shows a conversion function of fixed *length* of characters x_i that composes a pattern.

$$P_i = \mathfrak{E}(\mathrm{U}_j^{length} x_i\Big) \tag{13}$$

4.4. Parameter Selection

In the adjustment of different parameters, different parameters are varied and tested separately. Then, the best parameter value of the result is selected. In this situation, the best assembled three parameters may not be the global optimization. It may be a local optimization. Different people's signature data will have quite different features, so the selected parameters for different people may be different. In the experiment, different individuals will be trained and tested separately in order to obtain the best parameters for the particular people.

5. Experimental Results and Comparisons

The signature data for our experiment was obtained from ten persons. Each person wrote 19 handwriting signatures. Ten of the signatures each person wrote were their own name. This means that we obtained a total of 100 genuine signature files for testing. The other nine signatures were the names of the other nine persons. This means that we

obtained 9 forgery signature data files for each person and a total of 90 forgery signature data files for testing. Firstly, during the training and testing, 10 self signatures were used as template data to build an ACH state tree. After building an ACH state tree, the other nine signatures were used to search the state tree.

Table 3 is a summary of the parameter tuning; detailed explanation is provided in the following paragraphs.

Table 3. A summary of the parameter tuning.

Figure	Parameter	Result
Figure 9	For different sample rate with four waveforms, fixed $L = 4$, $r = 18$	The average accuracy is stable and around 70%. The highest accuracy is 75% at sample rate, $\gamma = 5$ or, $\gamma = 12$
Figure 10	For different sample rates with X-axis and Y waveform, fixed $L = 4$, $r = 18$	The average accuracy is about 65%. The highest accuracy is 71% at $\gamma = 14$.
Figure 11	For different pattern lengths with four waveforms, fixed $r = 18$, $\gamma = 5$	The accuracy rate is from 68% to 75.56%. A better accuracy and cost occur when length $L = 4$.
Figure 12	For different resolutions with four waveforms, fixed $L = 4$, $\gamma = 4$	The accuracy is not stable, and it varies from 62.22% to 74.44%. When setting resolution $r = 18$, the best accuracy is obtained.
Figure 13	$L = 4$, $\gamma = 4$, $r = 18$	For X-axis, the average accuracy is 60%
Figure 14	$L = 4$, $\gamma = 4$, $r = 18$	For Y-axis, the average accuracy is 54.7%.
Figure 15	$L = 4$, $\gamma = 4$, $r = 18$	For pressure waveform, the average accuracy is 47%.
Figure 16	$L = 4$, $\gamma = 4$, and $r = 18$	For pressure velocity waveform, the average accuracy is 55.3%.
Figure 17	$L = 4$, $r = 18$, $\gamma = 4$	For merged four waveforms, the average accuracy is 81%.
Figure 18	$L = 4$, $r = 18$, $\gamma = 4$	For merged four waveforms, the average accuracy is 91%.

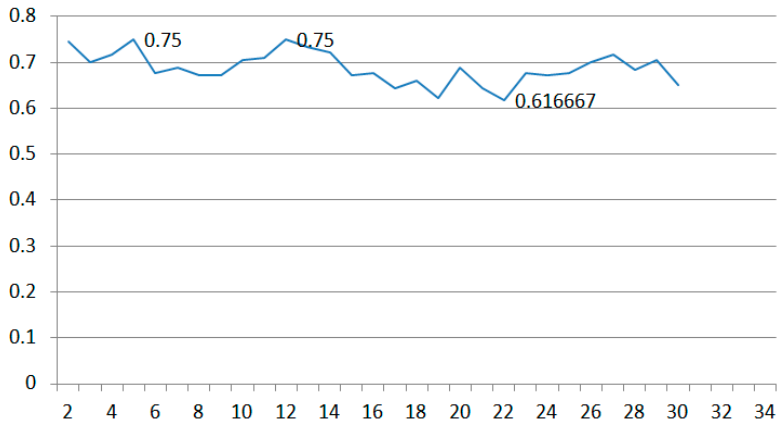

Figure 9. Accuracy under different sample rates for four elements.

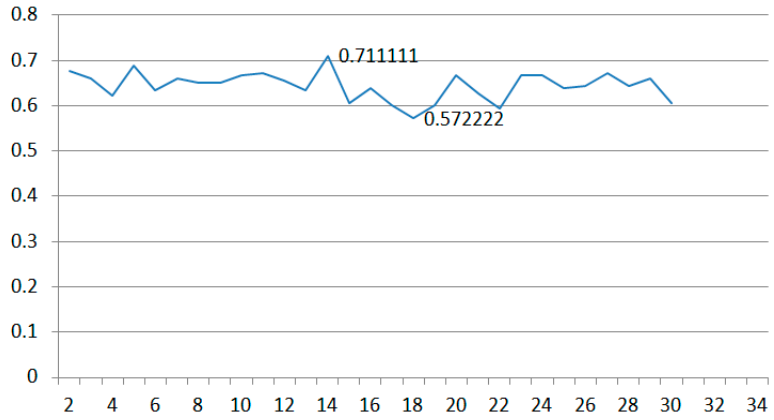

Figure 10. Accuracy under different sample rates for X-axis and Y-axis only.

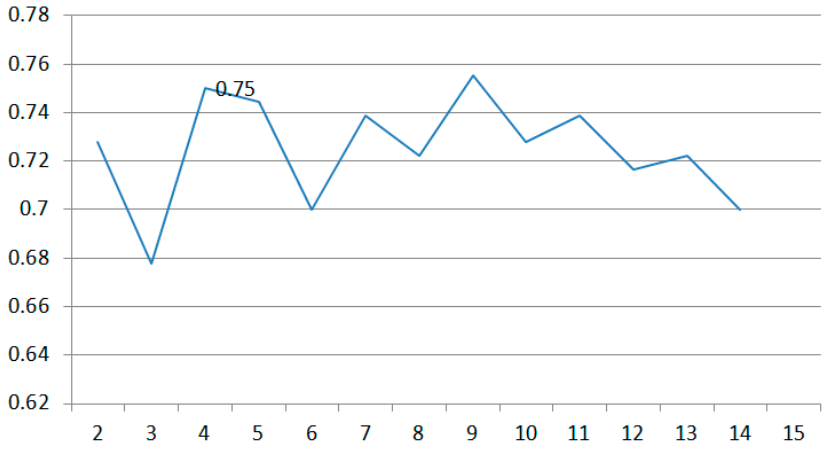

Figure 11. Accuracy under different pattern lengths.

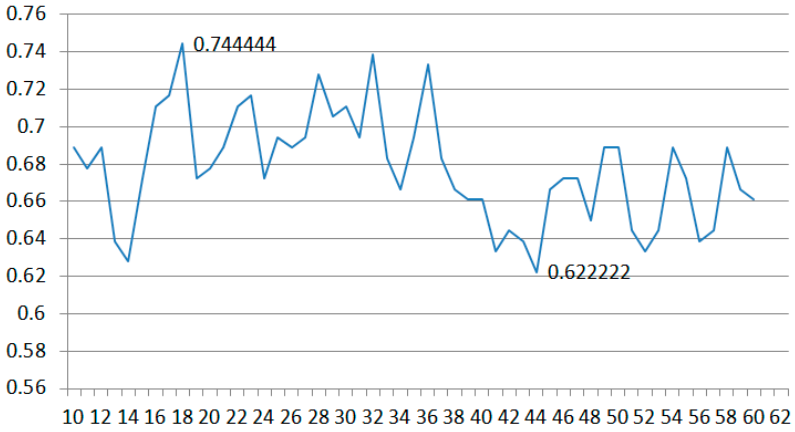

Figure 12. Accuracy under different resolutions.

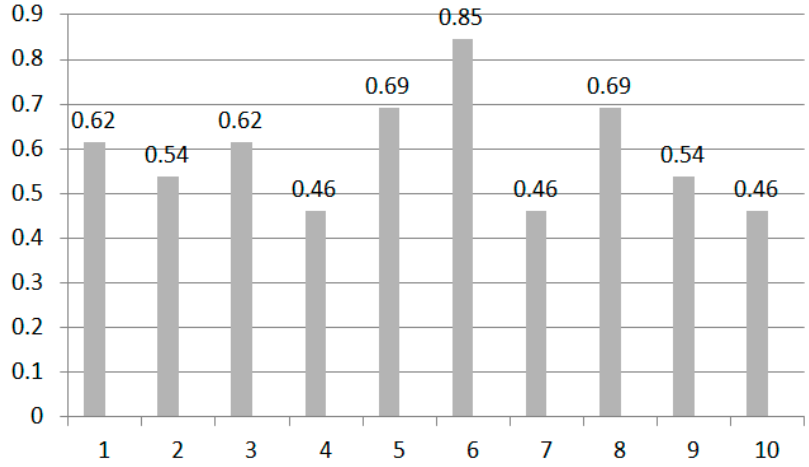

Figure 13. Accuracy of X-axis with ACH neural network.

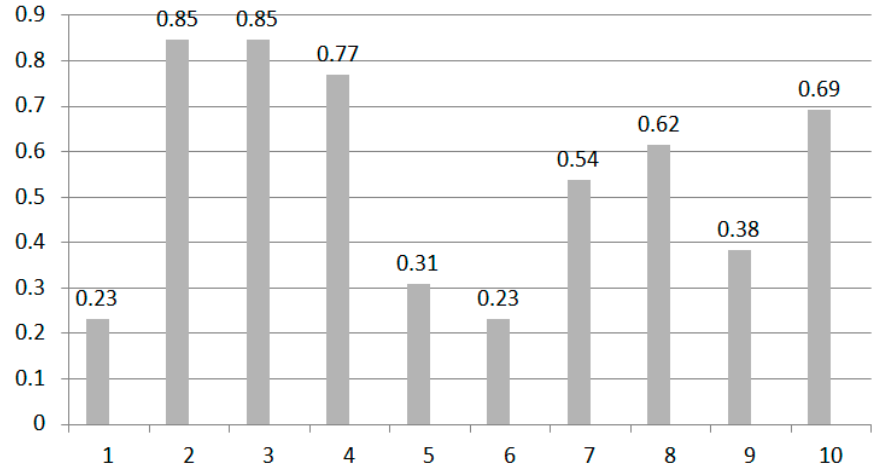

Figure 14. Accuracy of Y-axis with ACH neural network.

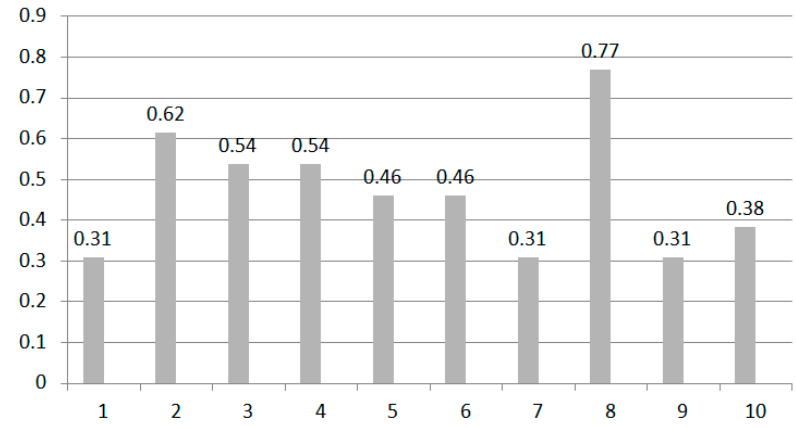

Figure 15. Accuracy of pressure with ACH neural network.

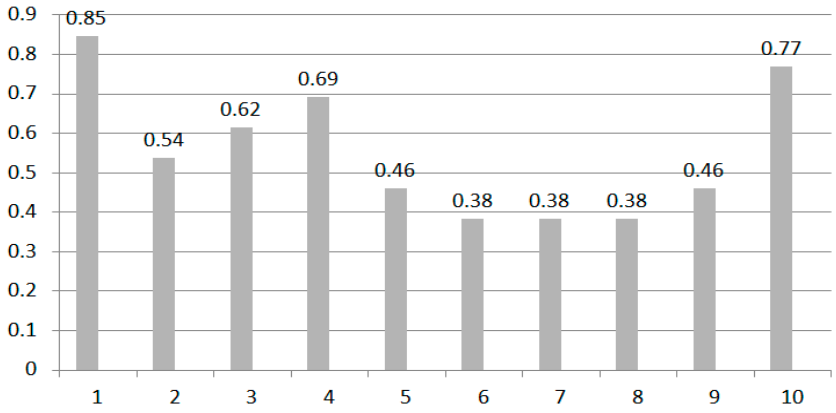

Figure 16. Accuracy of velocity with ACH neural network.

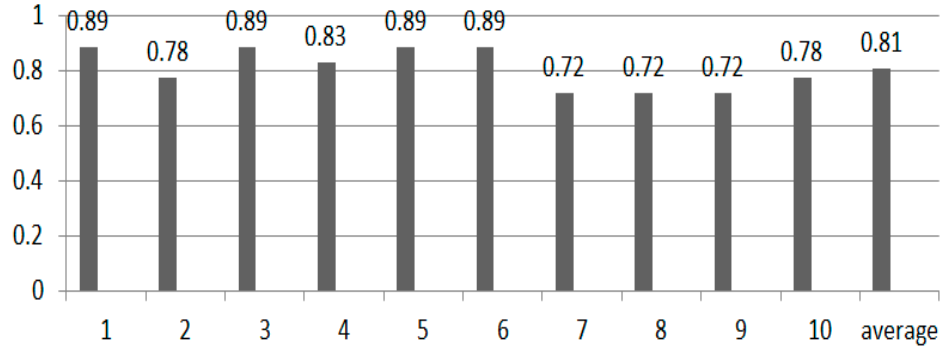

Figure 17. Accuracy of different people with same condition and voting.

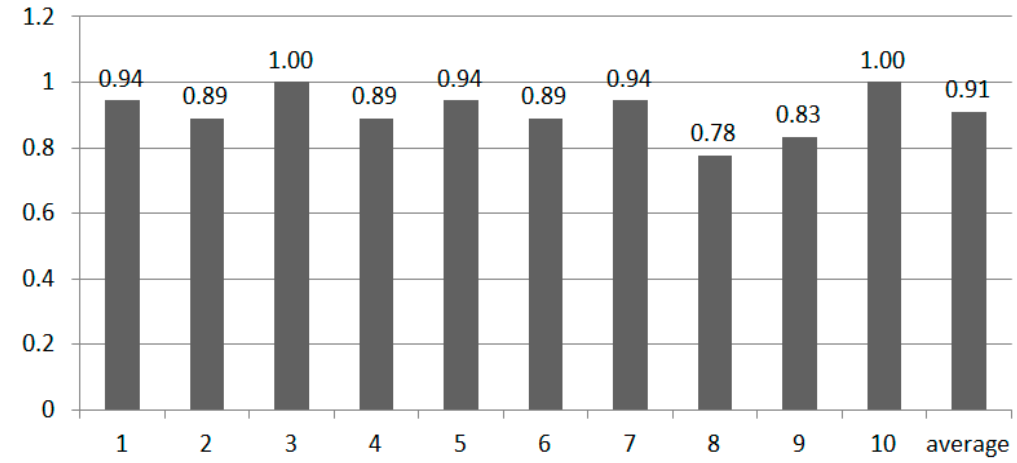

Figure 18. Accuracy of different people with different conditions and voting.

In this first stage of the experiment, all of the 10 people's signatures were trained and tested under the same parameters. In the second stage experiment, the signature data of each person were trained and tested separately in order to obtain a suitable parameter for each person.

(1) The first stage of the experiment.

Figure 9 is the accuracy of the experiment under different sample rates. The length of pattern was set as $L = 4$, and the resolution of the waveform was $r = 18$. This experiment tested all the four elements (X, Y, pressure, and velocity) of the signature.

Figure 9 shows that the average accuracy was stable and around 70% under different sample rate γ values. The highest accuracy was 75% at sample rate $\gamma = 5$ or $\gamma = 12$.

Figure 10 shows the results of the experiment of sample rate test for only processing the information of X-axis and Y-axis. The average accuracy was about 65%. The highest accuracy was 71%.

From Figures 9 and 10, it can be seen that the process of all four elements has higher accuracy than the process including only X-axis and Y-axis information. This means that the pressure and velocity of handwriting also have some contribution to recognition. However, the main contribution is from X-axis and Y-axis information. Pressure and velocity can assist to improve the accuracy of identification.

Figure 11 shows the results of the test of different pattern lengths. The resolution was set as $r = 18$, and the sample rate was $\gamma = 5$. Four elements were used for testing.

From Figure 11, we can see that the length of the pattern has a great influence on the identification accuracy. The accuracy rate ranges from 68 to 75.56%. Only a small change in the length may make a big difference in identification accuracy. Better accuracy and cost are found when length $L = 4$.

Figure 12 shows the results of the experiment on resolution. The condition of the experiment was $L = 4$, and the sample rate was $\gamma = 4$.

Figure 12 shows that change in pattern resolution also has a big influence on the result. The accuracy is not stable, and it varies from 62.22 to 74.44%. The curve of accuracy changes with different resolutions. When setting resolution $r = 18$, the best accuracy is obtained.

(2) The second stage of the experiment, in which all people were tested separately.

Figures 13–16 show the results of testing with the ACH neural network. Four elements were tested under $L = 4$, $\gamma = 4$, and $r = 18$ individually.

Figure 13 shows that when only testing the information of the X-axis, the average accuracy is 60%. It is too low to identify.

Figure 14 shows the accuracy of the Y-axis with the ACH neural network. The result is not stable. Its average accuracy is 54.7%. This means that in terms of Y-axis information, some people are quite different, but some people are similar.

Figure 15 shows the accuracy of pressure with ACH Neural Network. The pressure information is not good information for primary signature identification. The average accuracy is 47%. Therefore, pressure can only help improve accuracy.

From Figure 16, we can see that the accuracy of velocity is similar to that of the X-axis and ranges from 38% to 85%. The average rate is 55.3%. It is too low to be used in primary signature identification, so velocity can only help improve accuracy.

Figures 13–16 show that the accuracy is not good enough for identification if the four elements are processed separately. Therefore, we used the merging algorithm, which is discussed in Section 3.5, to merge these four elements into one vector in an attempt to obtain better accuracy.

Figure 17 is the accuracy for testing different persons' signatures under the same condition with $L = 4$, $\gamma = 4$, and $r = 18$. The average accuracy is 81%. If the experiment is processed under different conditions for different people, we find the accuracy is higher. Therefore, in the real application, it is better to use different parameters to identify different persons' signatures.

From Figure 18, we can see that the average accuracy is 91%. The signatures of different persons were tested under different experiments. We obtained a different parameter combination, as shown in Table 4. We can see that the most common sample rate for testing different persons' signatures is 5. The most common resolution is 18, and the pattern length is always 4. Thus, we can set the resolution r = 18, sample rate $\gamma = 5$, and pattern length $L = 4$ for general use in real-world application.

Table 4. Parameters for the identification of different persons' signatures.

ID	Resolution	Sample Rate	Length
1	18	5	4
2	18	5	4
3	40	5	4
4	10	5	4
5	21	5	4
6	18	5	4
7	18	13	4
8	18	11	4
9	18	3	4
10	12	5	4

5.1. Comparison with Euclidean Distance

The Euclidean distance algorithm is used to calculate the distance of two data. The input signature data, which are the same as in ACH algorithm, are two vectors. These two data also need to be pre-processed first. Equation (14) is the method for calculating the distance between two signature data. Figure 19 is the accuracy of the Euclidean distance algorithm. We can see the result is not good. Most of the accuracy is below 50%; the average accuracy is only 46%.

$$Sim = \sum ||x(i) - y(i)|| \tag{14}$$

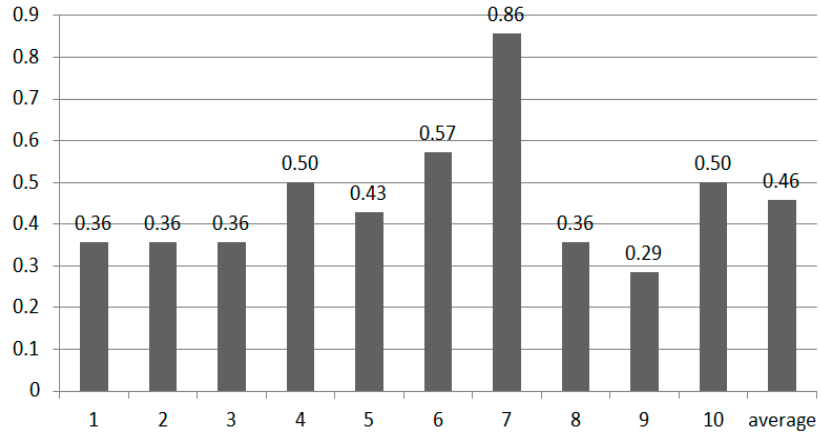

Figure 19. Accuracy of Euclidean distance algorithm.

5.2. Comparison with Correlation Classification Algorithm

The correlation algorithm uses the statistical correlation concept to calculate the similarity of two vector data. The correlation algorithm is calculated by following Equation (15). Figure 20 is the accuracy of the correlation algorithm. We can see the result is better than the result obtained from the Euclidean distance algorithm. The average accuracy is only 57%, but it is still far below normal accuracy.

$$Sim = \frac{\sum_i [(x(i) - mx) * (y(i-d) - my)]}{\sqrt{\sum_i (x(i) - mx)^2} \sqrt{\sum_i (y(i-d) - my)^2}} \tag{15}$$

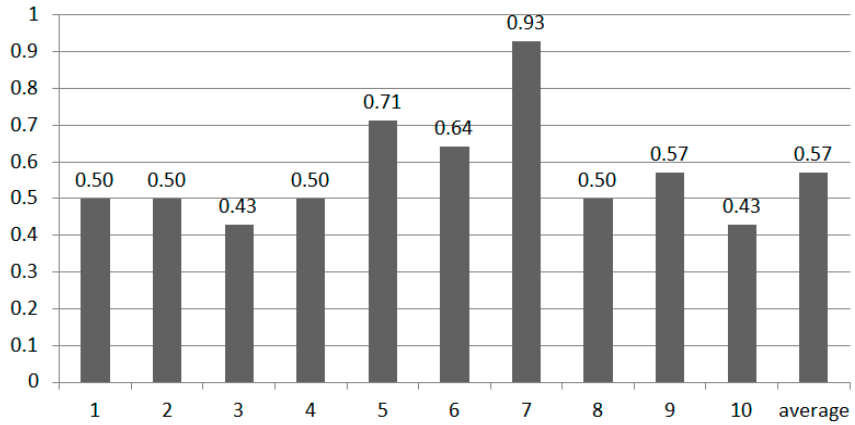

Figure 20. Accuracy of correlation algorithm.

5.3. Comparison with Convolutional Neural Network Algorithm

A convolutional neural network (CNN) [44] is used to differentiate between signatures. Before the verification part, the input data need to be subjected to feature extraction and converted to waveforms to obtain a pattern for each signature data; then, the verification task is performed with 50 epochs, and the whole training uses binary cross-entropy as the loss function, as shown in Equation (16).

$$H_y0(y) = -1/N^X \left(y_i^0 \log(y_i) + (1 - y_i^0) \right) \log(1 - y_i)) \tag{16}$$

Another VGG-16 is an advanced convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford [45]. Table 5 is the accuracy of CNN and VGG16. The results generated from CNN and VGG16 are better than the result obtained from the Euclidean distance algorithm but not better than that of the correlation algorithm. The average accuracy is 48% for CNN and 55% for VGG16.

Table 5. Accuracy of CNN and VGG16.

Compared Algorithms/Person ID	1	2	3	4	5	6	7	8	9	10	Average
CNN	65	46	48	48	47	40	52	43	47	52	48%
VGG16	84	60	29	67	71	26	78	39	60	43	55%

We infer that there are two reasons why the deep learning algorithm does not work well in this experiment. One is that we are doing signature identification, which is mainly the classification of two categories, and deep learning does not have much advantage for it. In addition, the benefit of the deep learning algorithm for a small number of data sets is not very good. Therefore, we suggest that the deep learning algorithm should not be used in the application of signature identification.

6. Conclusions

Aho-Corasick is a pattern matching algorithm that can find matching patterns among strings. However, for inaccurate biometric verification, traditional AC cannot be used for handwriting verification. Therefore, we designed a new ACH algorithm to be applied for live signature verification. The main new idea of the ACH algorithm is to employ a counting technique for handwriting features, which is represented by a histogram.

In our experiment, the accuracy of the ACH algorithm was 91%, which is higher than the total average accuracy of Euclidean distance method (46%), correlation algorithm (57%), CNN (48%), and VGG16 (55%). In the tuning experiment, we also found a suitable set of parameters for a mobile smart pad in real-world applications; i.e., one can set the resolution $r = 18$, sample rate $\gamma = 5$, and pattern length $L = 4$ for general use. In the future,

researchers can also use public datasets such as the MCYT, SUSIG, and NYU datasets to test the ACH algorithm.

In the near future, our proposed algorithm can be used by companies located in China, and we will continue to extend the number of Chinese signatures and share our results with other researchers. We also apply ACH to the watermarking [46] and digital fingerprint [47] applications.

In the design of an application system using the mobile smart pad in the real world, researchers may also need to consider the quality of experience (QoE), quality of service (QoS), architecture, and protocols for multimedia transmission over mobile smart pads [48].

Author Contributions: Supervision, K.-K.T.; conceptualization and methodology, K.-K.T. and C.C; software, data curation, and writing—original draft preparation, H.C.; validation and visualization, H.C. and C.B.; formal analysis and investigation, K.-K.T. and C.C.; resources and writing—review and editing, C.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Aho, A.V.; Corasick, M.J. Efficient string matching: An aid to bibliographic search. *Commun. ACM* **1975**, *18*, 333–340. [CrossRef]
2. Jung, B.; Han, I.; Lee, S. Security threats to Internet: A Korean multi-industry investigation. *Inf. Manag.* **2001**, *38*, 487–498. [CrossRef]
3. Oppliger, R. Internet security: Firewalls and beyond. *Commun. ACM* **1997**, *40*, 92–102. [CrossRef]
4. Ryan, S.; Bordoloi, B. Evaluating security threats in mainframe and client/server environments. *Inf. Manag.* **1997**, *32*, 137–146. [CrossRef]
5. O'Gorman, L. Comparing passwords, tokens, and biometrics for user authentication. *Proc. IEEE* **2003**, *91*, 2021–2040. [CrossRef]
6. Zou, M.; Tong, J.; Liu, C.; Lou, Z. On-line signature verification using local shape analysis. In Proceedings of the Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings, Edinburgh, UK, 6 August 2003; pp. 314–318.
7. Fallah, A.; Jamaati, M.; Soleamani, A. A new online signature verification system based on combining Mellin transform, MFCC and neural network. *Digit. Signal. Process.* **2011**, *21*, 404–416. [CrossRef]
8. Chen, C.; Tseng, K.K.; Zhang, X.F. A Real World On-line Signature Verification System Based on Correlation Algorithm. *Int. J. Comput. Appl. Technol. IJCAT* **2018**, *58*, 321–339. [CrossRef]
9. Ismail, M.; Gad, S. Off-line arabic signature recognition and verification. *Pattern Recognit.* **2000**, *33*, 1727–1740. [CrossRef]
10. DiMauro, G.; Impedovo, S.; Pirlo, G. Component-oriented algorithms for signature verification. *Int. J. Pattern Recognit. Artif. Intell.* **1994**, *8*, 771–793. [CrossRef]
11. Plamondon, R.; Yergeau, P.; Brault, J.J. A multi-level signature verification system. In *From Pixels to Features III-Frontiers in Handwriting Recognition*; Impedovo, S., Simon, J.C., Eds.; Elsevier: Amsterdam, The Netherlands, 1992; pp. 363–370.
12. Brault, J.-J.; Plamondon, R. Segmenting handwritten signatures at their perceptually important points. *IEEE Trans. Pattern Anal. Mach. Intell.* **1993**, *15*, 953–957. [CrossRef]
13. Dimauro, G.; Impedovo, S.; Pirlo, G. On-line Signature Verification by a Dynamic Segmentation Technique. In Proceedings of the 3th IWFHR, Buffalo, NY, USA, 25–27 May 1993; pp. 262–271.
14. Ammar, M.; Yoshida, Y.; Fukumura, T. Structural description and classification of signature images. *Pattern Recognit.* **1990**, *23*, 697–710. [CrossRef]
15. Sabourin, R.; Plamondon, R. Segmentation of handwritten signature images using the statistics of directional data. In *[1988 Proceedings] 9th International Conference on Pattern Recognition*; Institute of Electrical and Electronics Engineers (IEEE) Computer Science: Washington, DC, USA, 2003; pp. 282–285.
16. Sae-Bae, N.; Memon, N.; Sooraksa, P. Distinctiveness, complexity, and repeatability of online signature templates. *Pattern Recognit.* **2018**, *84*, 332–344. [CrossRef]
17. Wu, Q.-Z.; Lee, S.-Y.; Jou, I.-C. On-line signature verification based on logarithmic spectrum. *Pattern Recognit.* **1998**, *31*, 1865–1871. [CrossRef]
18. Mizukami, Y.; Yoshimura, M.; Miike, H.; Yoshimura, I. An off-line signature verification system using an extracted displacement function. *Pattern Recognit. Lett.* **2002**, *23*, 1569–1577. [CrossRef]
19. Liu, C.N.; Herbst, N.M.; Anthony, N.J. *Automatic Signature Verification: System Description and Field Test Results*; IEEE Transactions on Systems, Man, and Cybernetics (IEEE T-SMC): Washington, DC, USA, 1979; Volume 9, pp. 35–38.
20. Yoshimura, I.; Yoshimura, M. On-line signature verification incorporating the direction of pen movement–An experi-mental examination of the effectiveness. In *From Pixels to Features III- Frontiers in Handwriting Recognition*; Impedovo, S., Simon, J.C., Eds.; Elsevier Publ.: Amsterdam, The Netherlands, 1992; pp. 353–362.
21. Crane, H.D.; Ostrem, J.S. Automatic signature verification using a three-axis force-sensitive pen. *IEEE Trans. Syst. Man Cybern.* **1983**, *SMC-13*, 329–337. [CrossRef]

22. Lee, L.L.; Berger, T.; Aviczer, E. *Reliable On-Line Human Signature Verification Systems*; IEEE T-PAMI: Washington, DC, USA, 1996; Volume 18, pp. 643–647.
23. Nelson, W.; Turin, W.; Hastie, T. STATISTICAL METHODS FOR ON-LINE SIGNATURE VERIFICATION. *Int. J. Pattern Recognit. Artif. Intell.* **1994**, *8*, 749–770. [CrossRef]
24. Impedovo, S.; Castellano, M.; Pirlo, G.; Mingolla, A. A spectral analysis-based signature verification system. In *Transactions on Petri Nets and Other Models of Concurrency XV*; Springer Science and Business Media LLC: Berlin/Heidelberg, Germany, 1989; Volume 399, pp. 316–323.
25. Lam, C.F.; Kamins, D. Signature recognition through spectral analysis. *Pattern Recognit.* **1989**, *22*, 39–44. [CrossRef]
26. Letjman, D.; George, S. On-line handwritten signature verification using wavelets and back-propagation neural networks. In *Proceedings of the Sixth International Conference on Document Analysis and Recognition*; IEEE: Seattle, DC, USA, 2001; pp. 596–598.
27. Xia, X.; Song, X.; Luan, F.; Zheng, J.; Chen, Z.; Ma, X. Discriminative feature selection for on-line signature verification. *Pattern Recognit.* **2018**, *74*, 422–433. [CrossRef]
28. Mandal, S.; Prasanna, S.R.M.; Sundaram, S. An improved discriminative region selection methodology for online handwriting recognition. *Int. J. Doc. Anal. Recognit. (IJDAR)* **2018**, *22*, 1–14. [CrossRef]
29. Dimauro, G.; Impedovo, S.; Pirlo, G.; Salzo, A. A multi-expert signature verification system for bank check processing. *Int. J. Pattern Recognit. Artif. Intell.* **1997**, *11*, 827–844. [CrossRef]
30. Sabourin, R.; Genest, G.; Preteux, F. Off-line signature verification by local granulometric size distributions. *IEEE Trans. Pattern Anal. Mach. Intell.* **1997**, *19*, 976–988. [CrossRef]
31. Wirtz, B. Stroke-based time warping for signature verification. In Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, Canada, 14–16 August 1995; Volume 1, pp. 179–182.
32. Perizeau, M.; Plamondon, R. *A Comparative Analysis of Reginal Correlation, Dynamic Time Warping and Skeletal Tree Matching for Signature Verification*; IEEE T-PAMI: Washington, DC, USA, 1990; Volume 12, pp. 710–717.
33. Wu, Q.-Z.; Lee, S.-Y.; Jou, I.-C. On-line signature verification based on split-and-merge matching mechanism. *Pattern Recognit. Lett.* **1997**, *18*, 665–673. [CrossRef]
34. Jaim, A.K.; Griess, F.D.; Connell, S.D. On-line signature verification. *Pattern Recognit.* **2002**, *35*, 2963–2972.
35. Tseng, L.Y.; Huang, T.H. An on-line Chinese signature verification scheme based on the ART neural network. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*; IEEE: Baltimore, MD, USA, 1992; pp. 624–630.
36. Fuentes, M.; Gaci, S.; Dorizzi, B. On line signature Verification: Fusion of a Hidden Markov Model and a Neural Network via a Support Machine. In Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition, Niagra-on-the-Lake, ON, Canada, 6–8 August 2002; pp. 253–258.
37. Zhu, Q.; Zhang, L.; Longden, A. A correlation test-based validity monitoring procedure for online detecting the quality of nonlinear adaptive noise cancellation. *Int. J. Syst. Sci.* **2010**, *41*, 1043–1055. [CrossRef]
38. Zhang, L.F.; Zhu, Q.M.; Longden, A. A Correlation-Test-Based Validation Procedure for Identified Neural Networks. *IEEE Trans. Neural Netw.* **2008**, *20*, 1–13. [CrossRef]
39. Doroz, R.; Kudlacik, P.; Porwik, P. Online signature verification modeled by stability oriented reference signatures. *Inf. Sci.* **2018**, *460–461*, 151–171. [CrossRef]
40. Tolosana, R.; Vera-Rodriguez, R.; Fierrez, J.; Ortega-Garcia, J. Exploring Recurrent Neural Networks for On-Line Handwrit-ten Signature Biometrics. *IEEE Access* **2018**, *6*, 5128–5138. [CrossRef]
41. Deng, J.H.; Lu, Y.; Cheng, V.; Lee, S. Concrete crack detection with handwriting script interferences using faster re-gion-based convolutional neural network. *Comput. Aided Civ. Infrastruct. Eng.* **2020**, *35*, 373–388. [CrossRef]
42. Altwaijry, N.; Al-Turaiki, I. Arabic handwriting recognition system using convolutional neural network. *Neural Comput. Appl.* **2021**, *33*, 2249–2261. [CrossRef]
43. Sharma, A.; Jayagopi, D.B. Towards efficient unconstrained handwriting recognition using Dilated Temporal Convolution Network. *Expert Syst. Appl.* **2021**, *164*, 114004. [CrossRef]
44. Kim, T.-Y.; Cho, S.-B. Predicting residential energy consumption using CNN-LSTM neural networks. *Energy* **2019**, *182*, 72–81. [CrossRef]
45. Guan, Q.; Wang, Y.; Ping, B.; Li, D.; Du, J.; Qin, Y.; Lu, H.; Wan, X.; Xiang, J. Deep convolutional neural network VGG-16 model. *J. Cancer* **2019**, *10*, 4876–4882. [CrossRef]
46. Li, D.; Deng, L.; Gupta, B.B.; Wang, H.; Choi, C. A novel CNN based security guaranteed image watermarking generation scenario for smart city applications. *Inf. Sci.* **2019**, *479*, 432–447. [CrossRef]
47. Alsmirat, M.A.; Al-Alem, F.; Al-Ayyoub, M.; Jararweh, Y.; Gupta, B. Impact of digital fingerprint image quality on the fin-gerprint recognition accuracy. *Multimed. Tools Appl.* **2019**, *78*, 3649–3688. [CrossRef]
48. Nauman, A.; Qadri, Y.A.; Amjad, M.; Bin Zikria, Y.; Afzal, M.K.; Kim, S.W. Multimedia Internet of Things: A Comprehensive Survey. *IEEE Access* **2020**, *8*, 8202–8250. [CrossRef]