Towards a flexible framework for open source software for handwritten signature analysis

Richard Guest[1], Mike Fairhurst[1] and Claus Vielhauer[2]

[1] University of Kent,Canterbury, CT2 7NT, UK
[2] Otto-von-Guericke Magdeburg University, 39016 Magdeburg, Germany

Abstract. The human signature is still the most widely used and accepted form of personal authorisation and verification applied to documents and transactions. In this paper we describe the design and implementation of a flexible and highly configurable framework for the experimental construction and performance investigation of biometric signature verification systems. We focus on a design approach which encompasses a general process model for automatic signature processing, reference instances relating to specific data parsing, feature extraction and classification algorithms and detail the provision of a framework whereby unique signature systems can be easily constructed, trialed and assessed.

1 Introduction

The human signature has a long history of usage for personal authentication. Indeed, it is still the most widely used legally admissible technique for transaction and document authorisation (Jain et al.). Despite this long history of usage, conventional visual methods of authenticity assessment are prone to forgery and fraud - a situation recognised by credit card companies and banks as they move to alternative systems for consumer transitions (such as chip-and-pin).

Modern automatic biometric signature systems assess both the constructional (”on-line”) aspects of the signature (for example timing, velocity and pen rhythms) alongside the conventional (”off-line”) assessment of the drawn signature image utilising data captured on a graphics tablet or dedicated signature device (Plamondon and Srihari). These devices capture data in a pen position (X/Y) and pressure format at a constant sampling frequency. Over the last five years, signature verification systems have also found a use in securing mobile devices such as PDAs, Tablet PCs and mobile phones. Other recent device developments include a number of systems which collect data from accelerometers mounted in a pen which remove the need for a fixed capture surface. Also, additional signal types for online signature verification such as pen altitude and pen azimuth signals have been explored for signature verification (Hangai et al.). As mobile and ’novel’ devices become more prevalent, addressing the issues of verification on these platforms will become an even greater focus of research as will the applications to which they are used.

Research in the field of automatic signature verification can be seen to be concentrated, at the technological level, on two main strands: (a) the development of signature measurement features and novel methods of assessing the physical and constructional properties of a signature. Reported techniques within this first strand include pen direction and distance encoding, velocity and dynamic profiling, signature shape features, force and pressure characteristics and spectral and wavelet analysis (Nakanishi et al.). Studies have also analysed features derived from classical assessment of signatures from the forensic community. (b) The development of methods for selecting and combining feature measurements and verifying/identifying signature ownership. Techniques within this strand of research include multiple classifier structures and decision fusion algorithms, PCA, neural networks, probabilistic classification, dynamic time warping/matching and Hidden Markov Modeling (Rabiner). Important studies have also focused on issues such as enrolment strategies, template storage and update, forgery assessment, and so on (see, for example, Plamondon and Srihari). This diversity of work illustrates the need for appropriate software tools to support and facilitate effective future research in automatic signature verification.

2 An Experimental Framework

The motivation for the development of a flexible framework for the implementation, investigation and evaluation of signature systems is born out of two major issues within the research community. As further research is carried out into new systems and algorithms, so the diversification of standards and reference systems increases. Until recently no standard reference signature database was available making it impossible to accurately compare system performance - due to the sensitivity of data, many research groups and institutions keep signature test sets private, storing data in a range of proprietary formats. Likewise, due to proprietary requirements, there yet are no standard reference systems to provide a baseline for performance comparison.

A general purpose framework is currently being developed within which researchers can efficiently implement, investigate and evaluate techniques and system components for signature verification. Its key feature is the implementation of a software toolbox to facilitate investigation of both on-line and off-line signature processing, providing open source reference software to the research community and thereby enabling the system to be contributed to by third-party developers. The key design ethos of the system is the simple configuration of system modules (pre-processing, feature extraction and selection, classifiers and storage) for performance evaluation with normalised performance metrics thereby providing a standard against which other systems can be measured. The principal characteristics of the experimental framework include: modular design of system components with respect to handwriting/signature analysis, to provide the means to collect/import test data from various diverse sources, to provide the means to evaluate semantics beyond signatures (e.g. hand-written passwords, pass-phrases or personal identification numbers) and allow reproducible evaluation and exchange (and reuse) of module instances.

3 Framework Implementation

The design of the open source framework allows for flexibility in system implementation and restructuring alongside the addition of new modules and techniques. The latter is achieved through the release into the public domain of data structures containing the input and output formats for modules. The framework can be defined as consisting of six software subsections; the relationship between each subsection can be seen in Figure 1. Each of the subsections has a defined input and output data class structure enabling the development and integration of third party routines for use within the framework.

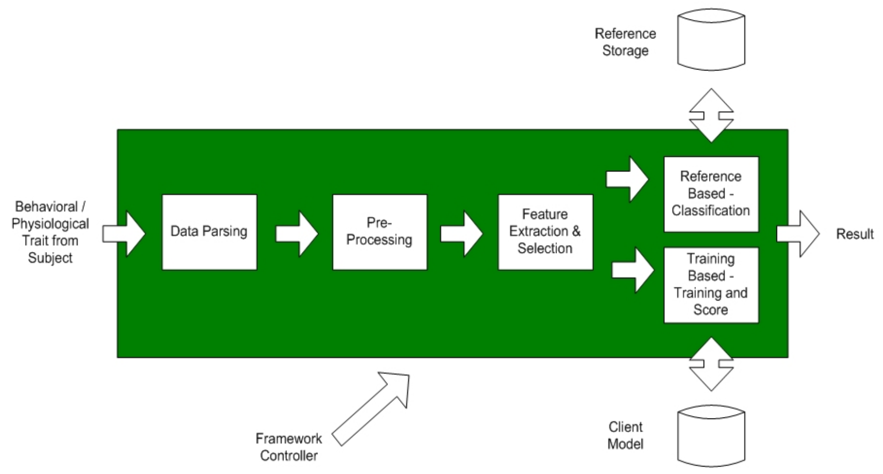
Reference Storage
Behavioral / Physiological Trait from Subject
Data Parsing
Pre-Processing
Feature Extraction & Selection
Reference Based - Classification
Training Based - Training and Score
Result
Framework Controller
Client Model

Fig. 1. Framework Subsections

To fully embrace the open source nature of the project the framework is constructed using Gnu CC. Each module instance is implemented as a Linux dynamic library which allows the framework controller to perform dynamic binding thereby producing an optimally compiled system for each configuration.

3.1 Data Parsing

Typically, a single signature sample captured from a subject is stored in an individual text file. Currently, there are no standard methods for storing data captured from a signature/writing device, with most research groups and commercial software companies use their own proprietary data format. This is one of the primary motivations for the development of a modality interchange format currently being undertaken by ISO/IEC (NIST). This subsection of this framework parses data adhering to a particular format into the internal data structure for the framework. For experimentation purposes and for the conceptual proof of our design, a parser has been constructed to read files in the SVC 2004 format, one of the most publicly available and used signature database in recent years (Yeung et al.).

This format stores the signature as a series of timestamped sample points comprising x and y position and pen pressure values. The data structure comprises five data classes; the relationship between these can be seen in Figure 2. CDataSet is the parent class containing such information as the size of the signature capture file, the date and time of the sample capture and the semantic class (signature, drawing, other, etc.). CPData contains information about the test subject while CDevice details the technical details of the capture device. Definitions of these classes can be reused across sample instances. CSourceData contains the raw sample data parsed from the input file while CData contains normalised values.

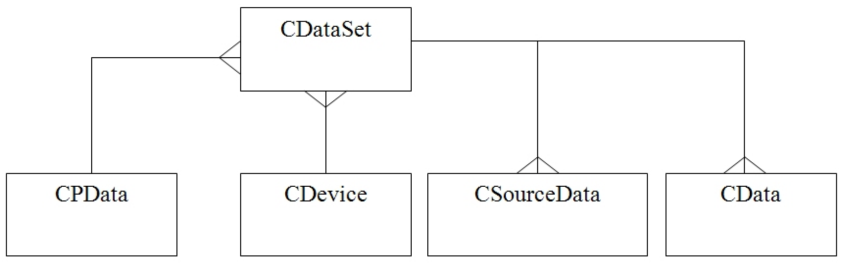

Fig. 2. Data parsing class structure

3.2 Preprocessing

Due to the wide range of capture devices providing the signature data, pre-processing prior to feature extraction is often necessary. In this subsection, data, stored in the standard framework structure as defined above is pre-processed and then stored back into the same structure. Common pre-processing routines such as low-pass filtering and special and temporal interpolation are implemented in the initial module set.

3.3 Feature Extraction and Selection

Feature modules individually extract the on-line and off-line performance characteristics from a signature sample (for example width of signature, time taken to produce signature). Separate features are implemented in individual modules increasing the flexibility in system construction. Selection of which features form templates or are presented to classifiers is also performed in this section. For experimentation, routines to perform a variety of statistical features have been implemented alongside basic selection configuration. Output from the feature extraction and selection module is stored in a hierarchical class structure represented in Figure 3. At the lowest level, a collection of separate Feature Data (from different feature extraction modules) can be grouped into a Feature Vector. The defined class structure allows for a complete signature capture file to be segmented as a collection of feature vectors and also allows for multiple vectors per segment. A collection of these Feature Segments is brought together to form a Feature Set. Under this scheme Feature Extraction modules have the freedom to ignore one or more segments of the original (for example, an investigator may only be interested in the first n seconds of all files).

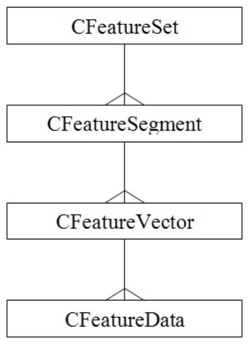

Fig. 3. Feature extraction class structure

Selection of features, vectors and segments is defined in the configuration file and managed by the framework controller. Two types of classification systems are allowed within the framework design:

In the context of signature biometrics, Reference Storage Systems extract a feature set from a series of training signatures and store them in either feature space and/or template form (stored in the Reference Storage subsection).

A comparison can then be made between test and training data with the output being an (optionally normalised) matching score based on the distance between a testing and training set. Training and testing data is formed using the same Data Parsing, Pre-processing and Feature Extraction chain. For initial experimentation a Levenshtein distance metric has been implemented for this category (Schimke et al.). The second subset, Training-based Systems, relies upon the use of a training set to configure the internal parameters of a classifier (for example a neural network system). These internal parameters are stored (often in proprietary form) for later classification of testing data. The framework provides a structure for the training of a classifier system and recognition through testing with the trained system. Again, the output is an optionally normalised matching score. The Client Model provides storage for these systems. For initial experimentation an HMM-based (Hidden Markov Model) classification system has been implemented.

3.4 Framework Controller

The Framework Controller provides the control, configuration and reporting mechanisms for the software subsections, or modules. Prerequisite and compatibility issues for each of the modules within the framework are defined and are verified before dynamic binding by the controller during configuration thereby ensuring that incompatible configurations are not selected. Key functions within the framework controller include the selections of modules to form an experimental system following a check on the validity of selected routines, presenting a list of available routines to the experimenter, selection and management of enrolment and training data and output matching score calculation signifying a matching between enrolment and verification data. Systems are configured using a text file script system which allows for ease and flexibility in system construction. The script uses a series of keywords to enable the definition of each subsection which is parsed by the framework controller.

3.5 Example System Configurations

Two examples are shown below of typical framework implementations. The first (Figure 4) details a signature system utilising many of the standard routines initially implemented within the framework to assess on-line features from the SVC 2004 database. Following parsing, filtration and interpolation, 15 user-implemented features are extracted and selected from each sample and either used to create a template (training) or test the system. The framework controller checks to see if the selected configuration (as defined by an external script) is valid and handles the presentation of training and testing data.

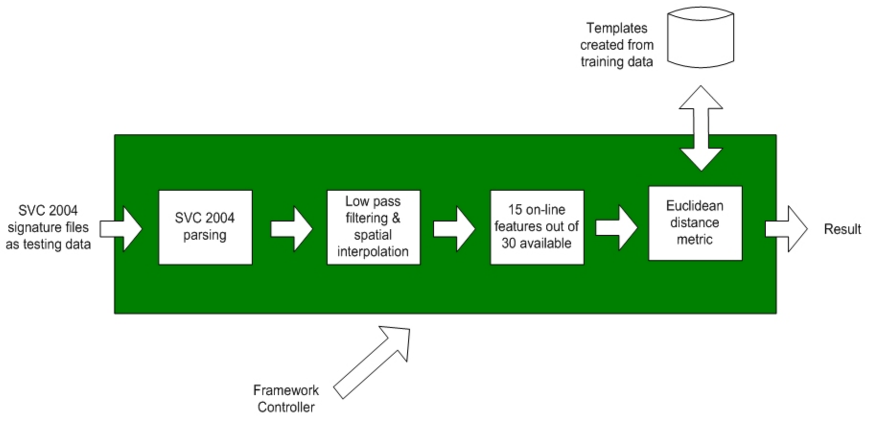
Templates created from training data
SVC 2004 signature files as testing data
SVC 2004 parsing
Low pass filtering & spatial interpolation
15 on-line features out of 30 available
Euclidean distance metric
Result
Framework Controller

Fig. 4. On-line System Configuration Example

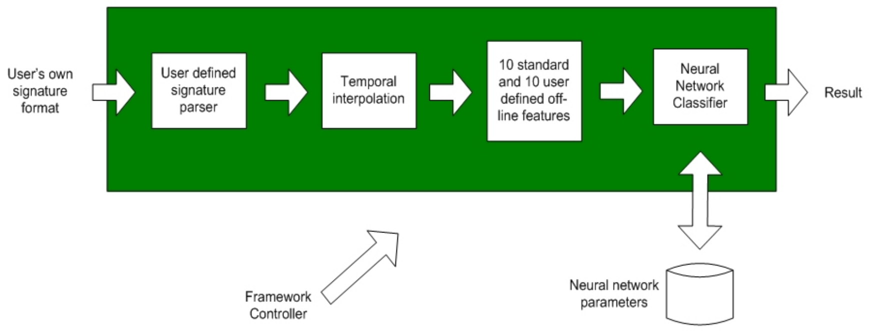
User's own signature format
User defined signature parser
Temporal interpolation
10 standard and 10 user defined off-line features
Neural Network Classifier
Result
Framework Controller
Neural network parameters

Fig. 5. Off-line System Configuration Example

The second example (Figure 5) shows an off-line evaluation system using a neural network training-based classifier. In this configuration, the experimenter has defined and implemented a number of features and parser instances for their own proprietary data format according to the open source data framework. Again the framework controller assesses compatibility prior to dynamic binding as well as handing the division of training and testing data.

3.6 Future Work and Usage

In this paper, we have introduced a novel design and implementation of an open and flexible framework for evaluation of online signature verification modules. We have further introduced to an initial set of reference modules for data parsing and feature extraction, and have shown two exemplary system configurations. It is envisaged that the framework will be of benefit to the signature verification community through the provision of both an experimental system for development and investigation and also, through a standardised framework configuration, a reference system for performance comparison. The flexibility in configuration and open source nature of the specification mean that additional feature routines and classifiers adhering to system standard are easy to implement. This widens the scope of the system's use beyond signature verification to other handwritten forms (such as drawings and forensic writing investigations) and even to other time-based measurement systems. In the short term, experimentation with the developed framework will be conducted as part of the EU BioSecure activities (BIOSECURE) focussing on an investigation of optimum system configuration and authoring of additional features and pre-processing modules.

3.7 Acknowledgements

The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507634 BIOSECURE.

References

BIOSECURE: BioSecure Network of Excellence, http://www.biosecure.info

HANGAI, S et al.(2000): On-Line Signature Verification based on Altitude and Direction of Pen Movement. In: *Proceedings of the IEEE International Conference on Multimedia and Expo, 1, 489–492.*

JAIN, A.K. et al (1999): Biometrics: Personal Identification in Networked Society, *The Kluwer International Series in Engineering and Computer Science, Vol.479, Springer, New York*

NAKANISHI, I et al. (2004): On-line signature verification based on discrete wavelet domain adaptive signal processing. *Proc. Biometric Authentication, LNCS 3072: 584–591*

NIST: The National Institute of Standards and Technology, Common Biometric Exchange File Format (CBEFF), http://www.itl.nist.gov/div895/isis/bc/cbeff/

MARTENS, R. and CLAESEN, L.(1996): On-Line Signature Verification by Dynamic Time Warping. In: *Proceedings of the 13th IEEE International Conference on Pattern Recognition, Vienna, Austria, 1, 38–42.*

PLAMONDON, R. and SRIHARI, S.N.(2000): On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. *IEEE Trans. PAMI, 22(1), 63–84.*

RABINER, L.R. (1989): A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE, 77(2), 257–286.*

SCHIMKE, S.et al. (2004): Using Adapted Levensthein Distance for Online Signature Verification, *Proc. IEEE International Conference on Pattern Recognition (ICPR), Vol. 2, 931–934*

YEUNG, D.Y.et al. (2004): SVC2004: First International Signature Verification Competition, *Proc. International Conference on Biometric Authentication (ICBA), 16–2.*