Design of Algorithm Development Interface for Fingerprint Verification Algorithms

Choonwoo Ryu, Jihyun Moon, Bongku Lee, and Hakil Kim

Biometrics Engineering Research Center (BERC),
School of Information and Communication Engineering, INHA Unversity, Incheon, Korea
{cwryu, jhmoon, bklee, hikim}@vision.inha.ac.kr

Abstract. This paper proposes a programming interface in order to standardize low-level functional modules that are commonly employed in minutiae-based fingerprint verification algorithms. The interface, called FpADI, defines the protocols, data structures and operational mechanism of the functions. The purpose of designing FpADI is to develop a minutiae-based fingerprint verification algorithm cooperatively and to evaluate the algorithm efficiently. In a preliminary experiment, fingerprint feature extraction algorithms are implemented using FpADI and an application program, called FpAnalyzer, is developed in order to evaluate the performance of the implemented algorithms by visualizing the information in the FpADI data structures.

1 Introduction

Biometrics of different modality requires different techniques of data processing, and a certain biometric technique can be implemented by various approaches. Therefore, standardization of biometric techniques is not a simple task. If biometrics modality and its technical approach are fixed, then the design of the standards is much easier. However, there are still many problems to be solved. For example, a certain fingerprint verification algorithm has a unique logical sequence of functional modules, some of which are not necessary in other verification algorithms.

The purpose of this study is to design a programming interface, so called *Fingerprint Verification Algorithm Development Interface* (FpADI) in order to standardize low-level functional modules that are commonly employed in minutia-based fingerprint verification algorithms [1]. FpADI focuses on function protocols, data structures and operational mechanism of the functional modules. In particular, FpADI must be differentiated from BioAPI [2] in the sense that it deals with low-level functions and data structures as listed in Table 1 and 2. BioAPI focuses on the interfaces between a biometric sensing device and an application program leaving the detailed algorithm for processing biometric data to algorithm developers. Meanwhile, FpADI defines the specification of the detailed algorithm for fingerprint verification in terms of the function protocols and data structures. In particular, the data structures are designed by referring to ISO standard committee's literatures [3-5].

D. Zhang and A.K. Jain (Eds.): ICB 2006, LNCS 3832, pp. 334 – 340, 2005.

Conventional methods of performance evaluation in biometrics are only able to compare the recognition results of overall algorithms consisting of numerous low-level functions such as segmentation, binarization, and thinning. They cannot compare the performance of different low-level functions for a specific data processing inside a recognition algorithm. They even fail to identify which function mainly deteriorates the performance of the recognition algorithm. The proposed standardization, however, facilitates both the comparison of the performance of different schemes for a specific low-level function and the improvement of the performance by easy modification of the algorithm. Furthermore, this standard specification will encourage several developers to invent interoperable algorithms or even a single algorithm.

2 Definition of Function Protocols and Data Structures

There are three types of data structures for FpADI as listed in Table 1. *Image* is either gray or binary while *Map* is a block-wise structure where the size of block is arbitrary. They contain the typical information produced as intermediate results by most of minutiae-based fingerprint recognition algorithms. Moreover, *Feature* contains a list of minutiae and singular points as the final result of a minutiae-based fingerprint recognition algorithm. It also has user-defined areas for algorithms generating extended features for fingerprint matching so that FpADI can cope with proprietary fingerprint verification algorithms. Table 1 describes various data for each data structure of FpADI in minutiae extraction.

Table 1. Data structure for feature extraction FpADI

Data Structure		Comments
Image	Input Image	Captured fingerprint image by a fingerprint sensor. It is the only image data provided by the FpADI calling function.
	Gray Image	Intermediate gray image output by FpADI functions.
	Binary Image	Intermediate binary image output by FpADI functions.
	Thinned Image	Binary image containing curves of one pixel width which represents fingerprint ridge or valley.
Map	Orientation	Map containing local orientation information which represents the direction of ridge flow in each block.
	Segmentation	Map containing local information of fingerprint foreground or background region.
	Frequency	Map containing local ridge frequency information representing the ridge distance between neighboring ridges in each block.
	Quality	Map containing global fingerprint image quality as well as local image quality.
Feature	Singular Points	User defined features as well as core/delta information.
	Minutiae	User defined features as well as ridge ending and bifurcation information.

Table 2 describes the functionality and typical output data type of the low-level functions employed by most of minutiae-based fingerprint recognition algorithms. Except the opening and the closing functions (*FPADI_SetInputImage* and *FPADI_FeatureFinalization*), the FpADI functions can be called at any order inside the feature extraction algorithm, which makes it possible for FpADI to develop feature extraction algorithms in different logical sequences.

Table 2. FpADI functions for feature extraction

Function	Comments
FPADI_SetInputImage	Input a fingerprint image to the feature extraction algorithm. This function is the first function to be called in the extraction algorithm.
FPADI_Preprocessing	Pre-process an *Input Image*. Typical output data: *Gray Image* in *Image*
FPADI_LocalOrientation	Compute local orientation. Typical output data: *Orientation* in *Map*
FPADI_QualityEvaluation	Compute global and local fingerprint quality. Typical output data: *Quality* in *Map*
FPADI_Segmentation	Segment an image into foreground and background regions. Typical output data: *Segmentation* in *Map*
FPADI_RidgeFrequency	Compute local ridge frequency. Typical output data: *Frequency* in *Map*
FPADI_Enhancement	Enhance a gray or binary image by noise removal. Typical output data: *Gray image* or *Binary image* in *Image*
FPADI_Binarization	Produce a binary image from a gray image. Typical output data: *Binary image* in *Image*
FPADI_Skeletonization	Generate a thinned image Typical output data: *Thinned image* in *Image*
FPADI_MinutiaeDetection	Generate minutiae and their extended features Typical output data: *Minutiae* in *Feature*
FPADI_MinutiaeFiltering	Post-process to eliminate noise in minutiae information Typical output data: *Minutiae* in *Feature*
FPADI_SingularityDetection	Generate singular points and their extended features Typical output data: *Singular Points* in *Feature*
FPADI_FeatureFinalization	Release all internal memory blocks in the feature extraction. This is the last function to be called by the request of either user or the algorithm itself.

As shown in Fig. 1, the FpADI manipulation module in an application calls all FpADI functions. FpADI functions are not allowed to call any other FpADI functions. However, the FpADI compliant algorithm, called FpADI SDK, specifies the order of FpADI function calls. In detail, the FpADI manipulation module calls the opening function (*FPADI_SetInputImage*) by providing a fingerprint image as the *Input image*. *FPADI_SetInputImage* mainly performs initializations of the feature extraction algorithm, and its return value indicates the next function to be called by the FpADI manipu-

lation module. In the same fashion, the FpADI manipulation module calls all the FpADI functions in the SDK until the closing function (*FPADI_FeatureFinalization*) is called.

FPADI_FeatureFinalization resets the internal memory blocks and prepares for the next feature extraction. Normally, *FPADI_FeatureFinalization* is called by the FpADI manipulation module according to the request from a certain FpADI function in the SDK. However, it also can be called directly from the application-specific module in the middle of the feature extraction process. In this case, it has to clean up all unnecessary memory blocks and prepares for the next feature extraction.

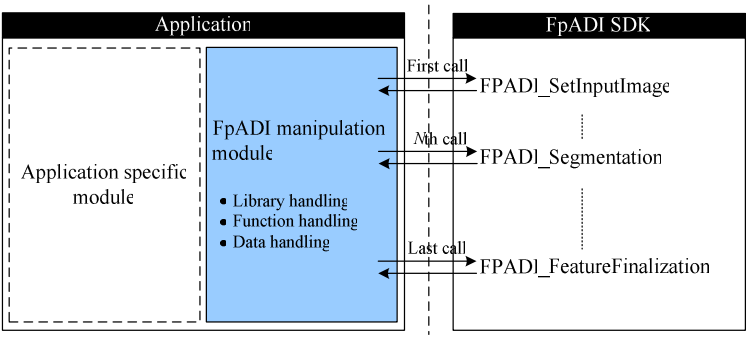


Fig. 1. Mechanism of FpADI function call

Except *FPADI_FeatureFinalization*, each FpADI function has four input parameters which correspond to *Image*, *Map*, *Feature* and *Calling order*, respectively. The data corresponding to the first three parameters are generated and referred to by FpADI functions themselves, while *Calling order* is a number starting from one and increases by one as a next function is called. Therefore, *Calling order* is a unique number associated with each FpADI function called. It distinguishes the functions especially when a certain function is called multiple times and each time performs different tasks. Fig. 2 shows an example of the FpADI function protocol.

The return value of all FpADI functions contains three types of information, function status, data-updating indicator, and the next calling function. The function status indicates the function's completion status, *success*, *failure*, or *bad parameter*. The data-updating indicator informs which input data have been updated by the function itself. And, the next calling function contains the name of the FpADI function which must be called in the next step.

```
UINT32 FPADI_QualityEvaluation(LPIMAGE Image, LPMAP Map, LPFEATURE Feature, UINT32 CallingOrder);
```

Fig. 2. Example of the FpADI function protocol

In summary, FpADI has following characteristics to encompass minutiae-based fingerprint verification algorithms of various logical sequences and data:

- Data structure for both pre-defined and algorithm-defined (extendable) fingerprint features
- Algorithm-defined sequence of calling functions
- Omission or multiple calls of a function

3 Implementations

3.1 Common Visual Analyzer: FpAnalyzer

For the purpose of demonstrating the effectiveness of FpADI, SDKs for fingerprint feature extraction, FpADI manipulation module (implemented in C++ class), and a visual algorithm analysis tool called *FpAnalyzer* are implemented. Firstly, the SDKs implemented in this study are fingerprint local orientation estimation, image quality estimation and fingerprint feature extraction algorithm. They observe the proposed FpADI specification. The first two algorithms contain partial functionality compared to the third algorithm which consists of most of the data and the functions listed in Table 1 and 2, respectively.

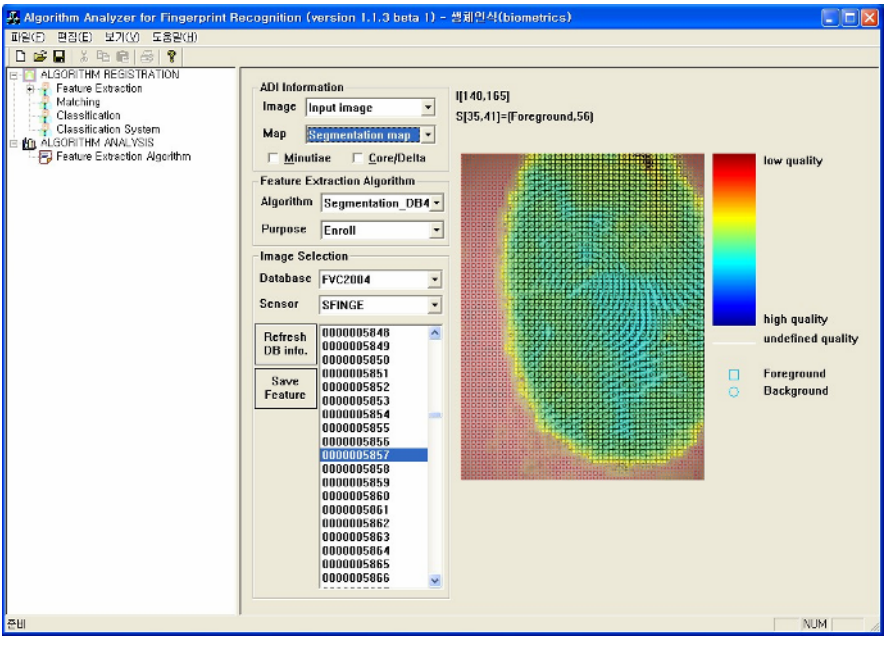


Fig. 3. *FpAnalyzer* - Visual algorithm analysis tool for fingerprint minutiae extraction

Secondly, the FpADI manipulation class, called *CFeatureADI*, can load and execute any FpADI compliant algorithms. It calls FpADI functions in the FpADI compliant SDK and performs data management such as memory allocation according to the requests of the called FpADI functions.

Finally, *FpAnalyzer* is an application tool for analyzing the algorithms under MS-Windows as shown in Fig. 3. It utilizes the *CFeatureADI* class for handling any FpADI compliant algorithms and displays all the data in the FpADI data structures listed in Table 1. It also provides a linkage between FpADI compliant algorithms and fingerprint databases.

3.2 FpADI Compliant Fingerprint Feature Extraction Algorithms

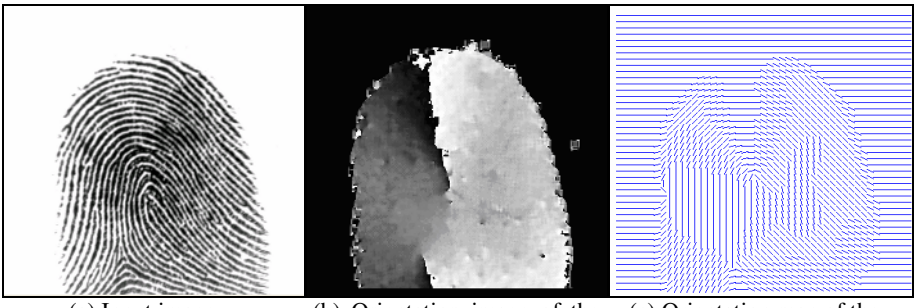
As mentioned in the previous section, three FpADI compliant algorithms have been implemented, fingerprint local orientation estimation, image quality estimation and fingerprint feature extraction algorithm, in order to show FpADI's characteristics under various programming requirements such as various block sizes and different sequences of FpADI function calls. Technical analysis of these algorithms is out of the scope of this study. Therefore, this paper will describe only the structural features of the algorithms.

The fingerprint local orientation estimation produces an orientation map in pixel, *i.e.*, 1×1 pixel block, where the orientation angle is in degree from 0 to 179. As shown in Table 3, this algorithm is the simplest one consisting of only three FpADI functions: *FPADI_SetInputImage*, *FPADI_LocalOrientation* and *FPADI_FeatureFinalization*. The second algorithm, image quality estimation, has six FpADI functions. Unlike in the first algorithm, *FPADI_LocalOrientation* is called at the fourth and *FPADI_QualityEvaluation* produces a map of 32×32 pixel blocks.

The third algorithm is a typical fingerprint feature extraction algorithm. Therefore, it generates minutiae information from the input image. Further, the block size of the orientation map in this algorithm is 8×8 pixels and its angle is represented in 8-directions. As listed in Table 3, this algorithm has 11 out of 13 FpADI functions. The function *FPADI_RidgeFrequency* and *FPADI_SingularityDetection* are not implemented because the algorithm does not utilize the information of local ridge frequency and singular points. Figure 4 shows an experimental example of local orientation of the first and third algorithm for the same input image.

Table 3. Calling functions of the implemented algorithms

Calling order	Local orientation estimation	Image quality estimation	Feature extraction
1	FPADI_SetInputImage	FPADI_SetInputImage	FPADI_SetInputImage
2	FPADI_LocalOrientation	FPADI_Segmentation	FPADI_Preprocessing
3	FPADI_FeatureFinalization	FPADI_Preprocessing	FPADI_LocalOrientation
4		FPADI_LocalOrientation	FPADI_Segmentation
5		FPADI_QualityEvaluation	FPADI_QualityEvaluation
6		FPADI_FeatureFinalization	FPADI_Enhancement
7			FPADI_Binarization
8			FPADI_Skeletonization
9			FPADI_MinutiaeDetection
10			FPADI_MinutiaeFiltering
11			FPADI_FeatureFinalization



(a) Input image (b) Orientation image of the first algorithm (c) Orientation map of the third algorithm

Fig. 4. Input and output data of the implemented algorithms

4 Conclusions and Future Works

We designed and implemented FpADI, a programming interface for development of minutiae-based fingerprint feature extraction algorithms. The function protocols and the data structures are defined in order to be able to cope with flexibility in various minutiae-based feature extraction algorithms. FpADI can provide technical benefits, for example, easy co-working with several algorithm developers and easy modification of an algorithm. In the near future, the implemented products including the sample SDK, *CFeatureADI* and *FpAnalyzer* will be available to public with the FpADI specification. One of our future works includes the design of FpADI specification for fingerprint matching algorithms.

Acknowledgement

This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the Biometrics Engineering Research Center (BERC) at Yonsei University.

References

1. D. Maltoni, D. Maio, A. K. Jain and S. Prabhakar, *Handbook of Fingerprint Recognition*, Springer, 2003.
2. Biometric Consortium, *BioAPI Specificaion Version 1.1*, March 2001.
3. ISO/IEC FDIS 19794-2:2004, *Information Technology - Biometric data interchange Formats-Part 2: Finger minutiae data*, ISO/IEC JTC 1/SC 37 N954, January 2005.
4. ISO/IEC FDIS 19794-4:2004, *Information Technology - Biometric data interchange Formats-Part 4: Finger image data*, ISO/IEC JTC 1/SC 37 N927, November 2004.
5. ISO/IEC FCD 19785-1:2004, *Information Technology - Common Biometric Exchange Formats Framework - Part 1: Data Element Specification*, ISO/IEC JTC 1/SC 37 N628, October 2004.
6. B.M. Mehtre and B.Chatterjee, "Segmentation of fingerprint images-a composite method," *Pattern Recognition*, vol.22, no.4, pp.381-385, 1989.
7. A.M. Bazen and S.H. Gerez, "Segmentation of Fingerprint Images," in Proc. Workshop on Circuits Systems and Signal Processing(ProRISC2001), pp.276-280, 2001.