TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol.12, No.5, May 2014, pp. 4079 ~ 4090
DOI: http://dx.doi.org/10.11591/telkomnika.v12i5.5208

A Multi-Threaded Fingerprint Direct-Access Strategy Using Local-Star-Structure-based Discriminator Features

G. Indrawan*[1], B. Sitohang[2], S. Akbar[3], A. S. Nugroho[4]
[1,2,3]Data & Software Engineering Research Division, STEI – ITB,
Jl. Ganesha 10, Bandung, Indonesia
[4]Agency for The Assessment and Application of Technology (BPPT),
Jl. MH Thamrin 8, Jakarta, Indonesia
*Corresponding author, e-mail: gindrawan@live.com[1], benhard@stei.itb.ac.id[2], saiful@informatika.org[3], asnugroho@ieee.org[4]

Abstract

Taking advantage of multi-thread technology provided by multi-core in single processor, this paper describes multi-thread implementation on fingerprint direct-access strategy using local-star-topology-based discriminator features. Multi-thread was applied for data partitioning on hashing add-lookup process, and for work partitioning on similarity score computation. Efficiency of the implementation was achieved, as confirmed by the experiments results. Using quad-thread of dual-core single processor on last tests on experiment, multi-thread implementation can reduce average search time about 17% compare to its single-thread implementation. This result was achieved with relatively same average accuracy trend and has significant improvement by considering millisecond order importance of searching process on large scale data.

Keywords*: core, direct-access, fingerprint, local-star-structure, thread*

Copyright © 2014 Institute of Advanced Engineering and Science. All rights reserved.

1. Introduction

This paper is about parallel process implementation as an improvement of our research on fingerprint direct-access strategy [1]. The improvement accommodates efficiency aspect, i.e. higher searching speed without sacrifying accuracy, related to taking advantage of running algorithm on multi-thread[1] environment of multi-core of single processor.

Our research proposed a new method in area of fingerprint direct-access, whose result confirmed it as a promising method because it can compared favorably with and even outperformed other competing state-of-the-art techniques over three publicly available data sets [2, 3]. For the accuracy aspect [4], up to certain small Penetration Rate, this method gave smaller Error Rate compared to other methods.

Related to other competing state-of-the-art techniques, in the last decade, several proposed fingerprint direct-access strategies has roughly classified by [5].

a) Global features, e.g., average ridge-line frequency. They are usually collaborated with other features for obtaining smaller search space [6].
b) Local ridge-line orientations, e.g., [6-9], other features derived from the orientation image, e.g., [10, 11].
c) Minutiae, e.g., [12, 15]: most indexing approaches based on minutiae derive robust geometric features from triplets of minutiae points and use hashing techniques to perform the search.
d) Other features obtained from the fingerprint pattern: such as Finger Code [15], ridge curvature [16], and SIFT features [17].
e) Matching scores to build index keys [18, 19].

[1] Based on [32], a thread, or thread of execution, is a software term for the basic ordered sequence of instructions that can be passed through or processed by a single central processing unit (CPU) core. At the other side, a core is a hardware term that describes the number of independent CPUs in a single computing component (die or chip).

Received November 8, 2013; Revised December 29, 2013; Accepted January 19, 2014

Proposed new method is close to the third approach above, but instead of using triplets of minutiae points, it used local-star-structure associated with the minutiae, and then its derived features were used by hashing techniques [20] to perform the search. Sequential access strategy (one-to-one comparison) using this local-star-structure was introduced first by Ratha et al. [21] and was followed by its variants and evolutions [22]. In the area of direct-access strategy, as far as author's knowledge, there is still no method utilizing Ratha's approach.

2. Research Method

Working on parallel process improvement of this new method, the rest of this paper is organized as follows. Chapter 2.1 introduces the proposed direct-access strategy, comprised of the underlying features extraction (Chapter 2.2), its mathematical perspective for indexing process (Chapter 2.3), retrieval process (Chapter 2.4-2.5), and candidate-list reduction mechanism (Chapter 2.6). Chapter 3 explains design for parallel process (multi-thread) implementation. Chapter 4 describes experiments on public data set, compare multi-thread implementation to its single-thread implementation. It describes the performance evaluation (Chapter 4.1), parameters calibration (Chapter 4.2), and comparison result (Chapter 4.3). Finally, Chapter 5 draws the conclusion.

2.1. Direct-Access Strategy

Figure 1 shows block diagram of direct-access strategy for fingerprint data.

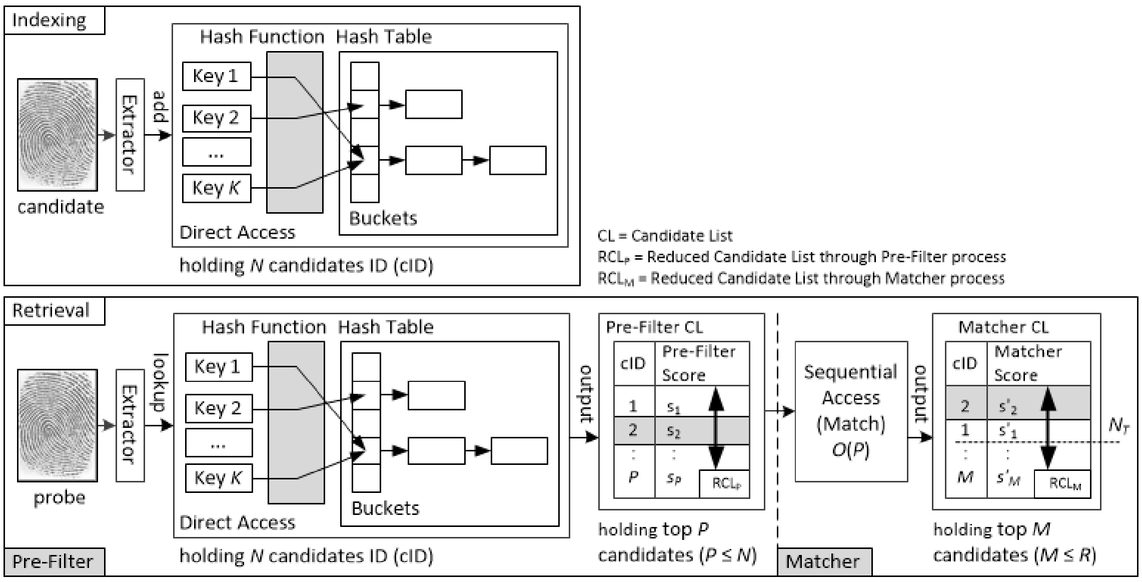


Figure 1. Proposed block diagram of direct-access strategy for fingerprint data.

a) The construction was comprised of indexing and retrieval process. It considered general, which means it can also be applied for other kind of data that try to accomplish search in direct-access fashion.
b) Indexing process was based on hashing function with its mathematical perspective (Chapter 2.3) constructed by three discriminator attributes (Figure 2).
c) Retrieval process will output subset N_T of N candidates from database, where N_T are candidates with top similarity score in descending-ordered fashion, and ideally N_T << N and it includes right searched candidate.
d) Retrieval process was constructed by pre-filter stage and matcher stage. Pre-filter stage was based on relative fast-inaccurate pre-filter-score computation (Chapter 2.4), while matcher stage was based on relative slow-accurate matcher-score computation (Chapter 2.5).
e) Retrieval process can be considered as multi-stage score computation, where pre-filter stage is its first stage and matcher stage is its second stage. First stage will output descending-

ordered-score candidate-list, while second stage will output final updated candidate-list, which has new updated score (in descending-ordered fashion too) that come from weighted sum of pre-filter score (Chapter 2.4) and matcher score (Chapter 2.5). Weighting value was optimally determined through learning process on training data (Chapter 4.2).

f) Retrieval process uses multi-stage candidate-list reduction mechanism to finally outputs N_T candidates (Chapter 2.6).

Figure 2 illustrates a structure of a local-star (circle with dashed-line) associated with a minutiae of two different fingerprint impressions, respectively. So, if a fingerprint has *n* minutiae, it will have *n* local-star structures. Zoomed rectangle area shows an edge, which constructs a local-star structure, with its three discriminator attributes (*sd*( ), β_i, β_j), used by proposed direct-access strategy. *sd*( ) is edge length, β_i is relative orientation of minutia-reference, and β_j is relative orientation of minutia-neighbour. Relative orientation is the angle difference between minutia's absolute angle to the edge's absolute angle. Absolute angle is the angle with reference to the horizontal line.

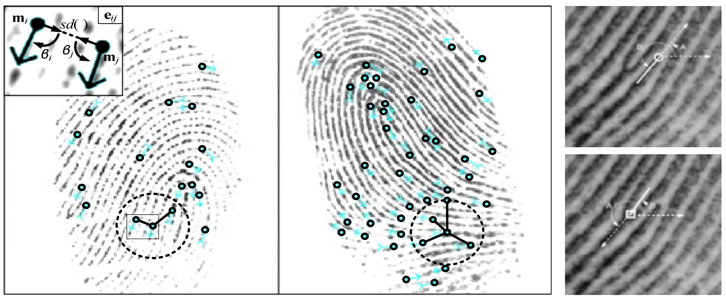


Figure 2. Left: structure of a local star (circle with dashed-line) associated with a minutia, each belongs to left-probe and right-candidate (a matching pair). Right: minutiae orientation from features extraction for: ending type (top); bifurcation type (bottom)

Based on Figure 1 and Figure 2, mathematical perspective of proposed direct-access strategy, would be applied on: indexing process (Chapter 2.3), pre-filter stage of retrieval process (Chapter 2.4), matcher stage of retrieval process (Chapter 2.5), and candidate-list reduction mechanism applied on retrieval process (Chapter 2.6).

2.2. Feature Extraction

Proposed direct-access strategy was run underlying preceded feature extraction process (block Extractor in Figure 1) based on minutiae detection algorithm [23, 24], utilizing algorithm of smoothing, binarization, and thinning [25], where the type, the coordinate location and the orientation of the ridge at each minutiae point are recorded. A minutia orientation is an absolute angle (refer to horizontal line and counter-clockwise increased angle). A ridge-ending orientation is computed by measuring an absolute angle of the line starting at the minutia point to the middle of the ridge. A ridge-bifurcation orientation is computed by measuring an absolute angle of the line starting at the minutia point to the middle of the valley between the bifurcating ridges.

The minutiae plotted in previous Figure 2 illustrate the line to which the angle of orientation is measured. Each minutia point is represented by a circle or square, and the line from the circle or square is either the ridge ending's ridge, or the ridge bifurcation's valley. In the illustration, the orientation angle marked as "A" is the ANSI/NIST standard. Angle "B" as FBI/IAFIS standard angle was used for this proposed direct-access strategy.

2.3. Indexing Process

Based on Figure 2, local-star structure can be represented by using graph notation. The *star* associated with the minutia $\mathbf{m}_i$ for a given maximum distance $d_{\max}$ and maximum neighbour $n_{\max}$ is the graph S_i

$$S_i = (V_i, E_i) \quad (1)$$

Consisting of:

a) The set of vertices V_i containing less than or equal to $n_{\max}$ minutiae $\mathbf{m}_j$ whose spatial distance *sd*( ) from $\mathbf{m}_i$ is less than or equal to $d_{\max}$.

$$V_i = \{\mathbf{m}_j \mid n(\mathbf{m}_j) \leq n_{\max} \text{ and } sd(\mathbf{m}_i, \mathbf{m}_j) \leq d_{\max}\} \quad (2)$$

b) The set of edges E_i

$$E_i = \{\mathbf{e}_{ij}\} \quad (3)$$

Where $\mathbf{e}_{ij}$ is the edge connecting minutia $\mathbf{m}_i$ with minutia $\mathbf{m}_j$ in V_i; $\mathbf{e}_{ij}$ is labeled with a 6-tuple (*id*, *i*, *j*, $sd(\mathbf{m}_i, \mathbf{m}_j)$, β_i, β_j), where *id* is a fingerprint *identifier* in database, β_i is relative orientation of $\mathbf{m}_i$ to $\mathbf{e}_{ij}$, and β_j is relative orientation of $\mathbf{m}_j$ to $\mathbf{e}_{ji}$.

Based on Equation (1), for hashing system of indexing process (Figure 1), there is a set:

$$H_i = \{\mathbf{h}_{ij}\} \quad (4)$$

a) $\mathbf{h}_{ij}$ is labeled with a 1-tuple (*h*), where *h* is a positif number, which is an output of hashing function $h(S_i)$ and considered as a *key* for data structure hash-table. *Value* associated with this *key*, represents S_i.

b) Hashing function $h(S_i)$ defined by:

$$h_k(S_i) = x_i \cdot 2^0 + y_i \cdot 2^{16} + z_i \cdot 2^{24} \quad (5)$$

k is a positif number that represents number of generated *buckets* (*keys*) associated with input data that is going into *hashing* function, where there is a set:

$$K = \{k \mid k \in P, k \leq |A| + |B| + |C|\} \quad (6)$$

x_i is an integer number that represents *bucket* (*key*) associated with input data, i.e. edge length, *sd*( ), that is going into *hashing* function, where there is a set:

$$A = \{ x_i \mid x_i \in Z, (sd(\mathbf{m}_i, \mathbf{m}_j) - \Delta_d)/\Delta_d \leq x_i \leq (sd(\mathbf{m}_i, \mathbf{m}_j) + \Delta_d)/\Delta_d\} \quad (7)$$

y_i is an integer number that represents *bucket* (*key*) associated with input data, i.e. relative orientation of minutia-reference, β_i, that is going into *hashing* function, where there is a set:

$$B = \{y_i \mid y_i \in Z, (\beta_i - \Delta_\beta)/\Delta_\beta \leq y_i \leq (\beta_i + \Delta_\beta)/\Delta_\beta\} \quad (8)$$

z_i is an integer number that represents *bucket* (*key*) associated with input data, i.e. relative orientation of minutia-neighbour, β_j, that is going into *hashing* function, where there is a set:

$$C = \{z_i \mid z_i \in Z, (\beta_j - \Delta_\beta)/\Delta_\beta \leq z_i \leq (\beta_j + \Delta_\beta)/\Delta_\beta\} \quad (9)$$

Δ_d is an integer number, represents maximum tolerance of $sd(\mathbf{m}_i, \mathbf{m}_j)$

Δ_β is an integer number that represents maximum tolerance of β_i, β_j

c) *Collision* in hashing system was accomodated by using data structure *list* containing input data S_i with the same *bucket* (*key*).

2.4. Pre-filter Stage of Retrieval Process

Based on Equation (1), Figure 3 shows a graph-pair (probe- and candidate- graph) which was used to compute similarity score [23, 26, 27] of matching-pair. This score computation algorithm was considered relatively slow but accurate, and was used by the next

Matcher Stage of Retrieval Process (Chapter 2.5). It needs to mention earlier, to show how relatively fast but inaccurate score computation algorithm was built on it.

Relatively slow-accurate score computation algorithm will work on graph-pair, i.e. probe-graph and candidate-graph, each has its longest-edge by number of (possibly discontinued) edges. Through this graph, the maximum score was computed (Chapter 2.5). The construction of the-longest-edge-pair of graph-pair passed through vertex- or minutia-pair, i.e. one probe minutia and one candidate minutia. Selected minutia-pair was based on certain similarity range of comparison of its three discriminator attributes (Figure 2).

On the other side, relatively fast-inaccurate score computation algorithm just accumulate its score from number of correspondence connected-edges-pair (probe connected-edge and candidate connected-edge). Connected-edge-pair was constructed by two or three probe's edges for probe connected-edge, and by two or three candidate's edges for candidate connected-edge. Counted connected-edge-pair was based on certain similarity range of comparison of its three discriminator attributes (Figure 2). Equation (10) describes this relatively fast-inaccurate score computation of a matching pair (probe and candidate), which is constructed by number of its connected-edge-pair from two edges, s_{R_2}, and by number of its connected-edge-pair from three edges, s_{R_3}.

$$s_R = s_{R_2} + w_{R_3} \cdot s_{R_3} \tag{10}$$

Where:

a) s_{R_2} is equal to $|\mathrm{E}_{R_2}|$ (cardinality of the set of edges E_{R_2}), where $\mathrm{E}_{R_2}$was computed on the set of edges E*ᵢ* (Equation 3), such that:

$$\begin{aligned} E_{R_2} = \; & \{(\mathbf{e}_{p_{ij}}, \mathbf{e}_{p_{jk}}) \mid i \neq j \text{ AND } i \neq k \text{ AND } j \neq k \text{ AND} \\ & \left| sd\left(\mathbf{m}_{p_i}, \mathbf{m}_{p_j}\right) - sd\left(\mathbf{m}_{c_{i'}}, \mathbf{m}_{c_{j'}}\right) \right| \leq \Delta_d \text{ AND } |\beta_{p_i} - \beta_{c_{i'}}| \leq \Delta_\beta \text{ AND} \\ & |\beta_{p_j} - \beta_{c_{j'}}| \leq \Delta_\beta \text{ AND } |sd(\mathbf{m}_{p_j}, \mathbf{m}_{p_k}) - sd(\mathbf{m}_{c_{j'}}, \mathbf{m}_{c_{k'}})| \leq \Delta_d \text{ AND} \\ & |\beta_{p_j} - \beta_{c_{j'}}| \leq \Delta_\beta \text{ AND } |\beta_{p_k} - \beta_{c_{k'}}| \leq \Delta_\beta\} \end{aligned} \tag{11}$$

b) s_{R_3} is equal to $|\mathrm{E}_{R_3}|$ (cardinality of the set of edges E_{R_3}), where $\mathrm{E}_{R_3}$was computed on the set of edges E*ᵢ* (Equation (3)), such that:

$$\begin{aligned} E_{R_3} = \; & \{(\mathbf{e}_{p_{ij}}, \mathbf{e}_{p_{jk}}, \mathbf{e}_{p_{kl}}) \mid i \neq j \text{ AND } i \neq k \text{ AND } i \neq l \text{ AND } j \neq k \text{ AND} \\ & j \neq l \text{ AND } |sd(\mathbf{m}_{p_i}, \mathbf{m}_{p_j}) - sd(\mathbf{m}_{c_{i'}}, \mathbf{m}_{c_{j'}})| \leq \Delta_d \text{ AND} \\ & |\beta_{p_i} - \beta_{c_{i'}}| \leq \Delta_\beta \text{ AND } |\beta_{p_j} - \beta_{c_{j'}}| \leq \Delta_\beta \text{ AND} \\ & |sd(\mathbf{m}_{p_j}, \mathbf{m}_{p_k}) - sd(\mathbf{m}_{c_{j'}}, \mathbf{m}_{c_{k'}})| \leq \Delta_d \text{ AND } |\beta_{p_j} - \beta_{c_{j'}}| \leq \Delta_\beta \text{ AND} \\ & |\beta_{p_k} - \beta_{c_{k'}}| \leq \Delta_\beta \text{ AND } ((|sd(\mathbf{m}_{p_j}, \mathbf{m}_{p_l}) - sd(\mathbf{m}_{c_{j'}}, \mathbf{m}_{c_{l'}})| \leq \Delta_d \text{ AND} \\ & |\beta_{p_j} - \beta_{c_{j'}}| \leq \Delta_\beta \;\&\; |\beta_{p_l} - \beta_{c_{l'}}| \leq \Delta_\beta) \text{ OR} \\ & (|sd(\mathbf{m}_{p_k}, \mathbf{m}_{p_l}) - sd(\mathbf{m}_{c_{k'}}, \mathbf{m}_{c_{l'}})| \leq \Delta_d \text{ AND } |\beta_{p_k} - \beta_{c_{k'}}| \leq \Delta_\beta \text{ AND} \\ & |\beta_{p_l} - \beta_{c_{l'}}| \leq \Delta_\beta))\} \end{aligned} \tag{12}$$

c) For point 2 and 3, *p* represents probe, *c* represents candidate, Δ_d and Δ_β represents maximum tolerance for related variable (Chapter 2.3).

d) $w_{R_3}$is weighting of s_{R_3} which is determined through the learning process on training data (Chapter 4.2).

2.5. Matcher Stage of Retrieval Process

Equation (13) describes relatively slow-accurate score computation, which was applied on matching graph-pair (Figure 3).

$$s_M = \max(S_M) \tag{13}$$

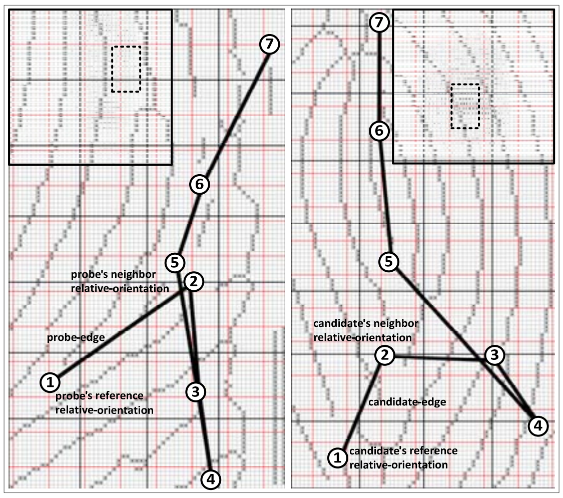

7
6
5
2
probe's neighbor relative-orientation
probe-edge
1
3
probe's reference relative-orientation
4
7
6
5
candidate's neighbor relative-orientation
2
3
candidate-edge
4
1
candidate's reference relative-orientation

Figure 3. Case of False Reject (FR) of a matching pair, i.e. probe (left) and candidate (right), where the algorithm computes similarity score on wrong fingerprint area (square area with dash-line). Nevertheless, it shows how the algorithm works to produce the longest-edge-pair (each belongs to probe and candidate) that will be used for relatively slow-accurate score computation.

Where s_M is equal to maximum value of s_k on set $S_{M.}$

$$S_M = \{s_k \mid k \in P,\ s_k = \sum_{i=1}^{N} w_i \cdot v_i\} \quad (14)$$

k is positive number, *N* is number of involved matching variables, v_i is individual score of a matching variable, and w_i is weighting of v_i, which is determined by learning process on training data (see Chapter 4.2). As shown by Figure 3, involved matching variables, *v*, are:

a) Number of minutia-pair, one minutia from probe-graph and one corresponding minutia from candidate-graph (seven minutia-pairs as shown by Figure 3).
b) Number of repetition found of minutia-pair that construct the longest-edge of graph-pair (not visible by Figure 3).
c) Number of edge-pair that construct the longest-edge of graph-pair (six edge-pairs as shown by Figure 3).
d) Number of minutia-pairs that have same type of minutia (as shown by Figure 3, minutia-pair number 3 was not counted since its probe-minutia of left graph is type Ending and its corresponding candidate-minutia of right graph is type Bifurcation).
e) Accumulation of the difference of edge-length between probe-edge of probe-graph and its corresponding candidate-edge of candidate-graph associated with the longest-edge of graph-pair (not visible by Figure 3).
f) Accumulation of the difference of relative orientation between minutia-reference of probe-graph and its corresponding minutia-reference of candidate-graph associate with the longest-edge of graph-pair (not visible by Figure 3).
g) Accumulation of the difference of relative orientation between minutia-neighbour of probe-graph and its corresponding minutia- neighbour of candidate-graph associate with the longest-edge of graph-pair (cannot shown by Figure 3).
h) Fraction of minutia-pairs equals to average value of fraction of probe-minutiae and fraction of candidate-minutiae. Fraction of probe-minutiae equals to ratio of number of probe-minutiae (that construct the longest-edge of probe-graph) to the total number of probe-minutia. Fraction of candidate-minutiae equals to ratio of number of candidate-minutiae (that construct the longest-edge of candidate-graph) to the total number of candidate-minutia. Respectively, number of probe-minutiae and number of candidate-minutiae that construct the longest edge of graph pair, is equals to number of minutia-pair, i.e. seven, as shown by Figure 3.

2.6. Candidate List Reduction

Cascade candidate-list (CL) reduction mechanism through two step process, was applied for retrieval process (Figure 1). The first step was applied to pre-filter stage and the second one for matcher stage.

a) The first step, by applying variable threshold on score ratio [28]. Given a database of *N* candidate fingerprints and a query (probe) fingerprint, let CL from point 1, i.e. $\{(\mathrm{id}_k, s_{R_k})\}$, $k = 1, \ldots, n$, be the sorted list of *n* initial candidates with $0 < n \le N$. Each CL element consists of a database fingerprint identifier id_k and a score s_{R_k}, where $s_{R_k} \ge s_{R_{k+1}}$. A reduction criterion retains the first n_R candidates of CL, where the number n_R ($0 \le n_R \le n$) is determined on the basis of the scores in CL itself. Variable threshold on score ratio given by the equation below:

$$n_R = \begin{cases} \min(K_\rho) - 1, & if\ K_\rho \neq 0 \\ n, & otherwise \end{cases} \tag{15}$$

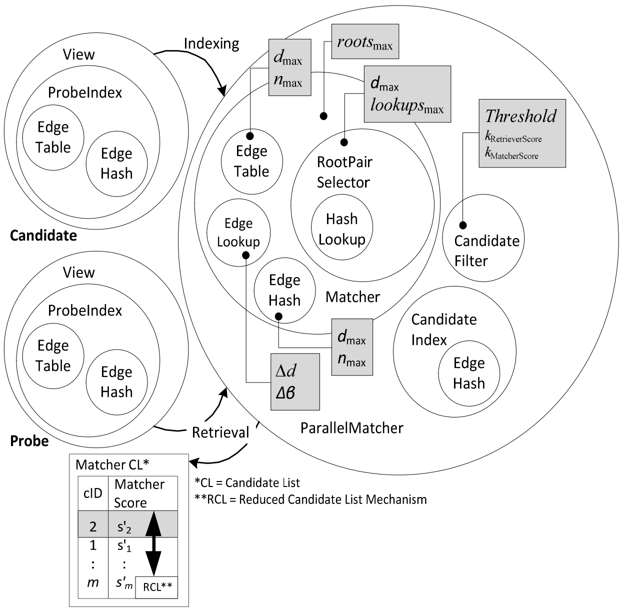
Where:

$$K_\rho = \{k \mid \frac{s_{k'}}{s_k} > \rho\}, \tag{16}$$
$$s_{k'} = \max(1, k - p.n), \frac{1}{n} \le p \le 1, and\ 0 \le \rho \le 1$$

b) The second step, by simply truncating CL from point 2 to become N_T if CL length longer than N_T. If CL length shorter or same length than N_T, nothing will be truncated.

3. Multi-Thread Design

To be able to design parallel (multi-thread) process [29], we need to understand memory objects operation in process. Figure 4 illustrated several improved memory objects, preliminary studied by [23, 30], that constructs proposed direct-access process blocks (Figure 1) and implement several supporting data structures [31].



cID	Matcher Score
2	s'$_2$
1	s'$_1$
:	:
m	s'$_m$

Figure 4. Memory Objects Construction in Proposed Direct-access Strategy

Figure 5 illustrates multi-thread design based on those memory objects functionality.

a) Multi-threading was applied both to indexing and retrieval process that construct proposed direct-access strategy (Figure 1).

b) There are two types of thread involved, i.e. thread for data partitioning (applied to indexing and retrieval), and thread for work partitioning (applied to retrieval only).
c) Step 1, at indexing process and retrieval process, uses multi-thread for construction of 2D data-array (Equation (1)), related to computation and ordering of relatively slow-accurate similarity score (Equation (13)) by Step 4 at retrieval process.
d) Step 2 using multi-thread for process at hash-table (Equation (4)) with data (Equation (5)) addition and lookup activity related to indexing process and retrieval process, respectively.
e) Step 3 at retrieval process using multi-thread related to computation and ordering relatively fast-inaccurate similarity score (Equation (10)).

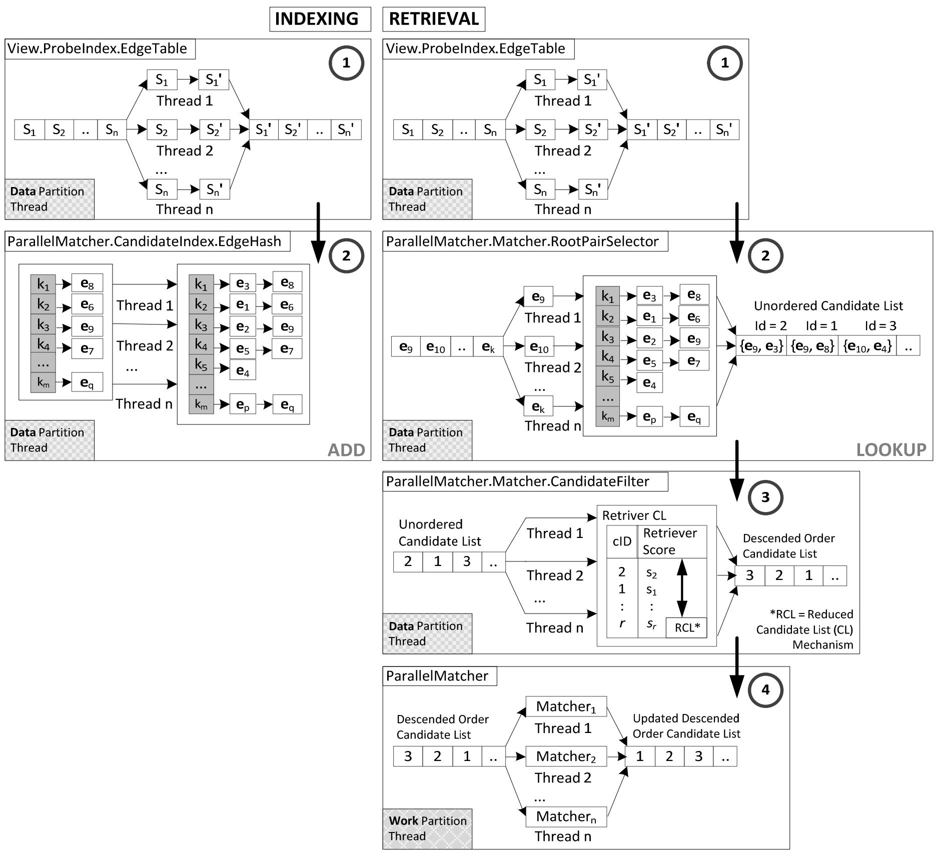

INDEXING
RETRIEVAL
View.ProbeIndex.EdgeTable
ParallelMatcher.CandidateIndex.EdgeHash
ParallelMatcher.Matcher.RootPairSelector
ParallelMatcher.Matcher.CandidateFilter
ParallelMatcher
Thread 1
Thread 2
Thread n
Data Partition Thread
Work Partition Thread
ADD
LOOKUP
Unordered Candidate List
Retriver CL
cID
Retriever Score
RCL*
Descended Order Candidate List
*RCL = Reduced Candidate List (CL) Mechanism
Updated Descended Order Candidate List
Matcher₁
Matcher₂
Matcherₙ

Figure 5. Multi-threading in Proposed Direct-access Strategy: indexing (left) and retrieval (right). Step 1 at both side, constructs 2D data-array for computing & ordering relatively slow-accurate score by Step 4 at retrieval side. Step 2 at both side related to candidates hash table add-lookup activity. Step 3 at retrieval side related to computing and ordering relatively fast-inaccurate score

4. Results and Analysis

This chapter describes experiment carried out to compare multi-thread to its single-thread implementation of proposed direct-access strategy (Chapter 4.3). It is a C# implementation, running on a 2.40-GHz Intel Core 2 Quad CPU [32]. The experiment was based on direct-access performance evaluation (Chapter 4.1) using optimized algorithm's parameters through learning process on separated training data (Chapter 4.2).

4.1. Performance Evaluation

4.1.1. Accuracy

The accuracy performance of fingerprint direct-access [4] approaches is typically evaluated by reporting the trade-off between Error Rate (ER) and Penetration Rate (PR). The

ER is defined as the percentage of searched fingerprints that are not found. The PR is the portion of the database that the system has to search on the average (corresponding to the average length of the candidate lists).

In this benchmark area, the ER/PR trade-off depends on a parameter (Max_{PR}) controlling the maximum portion of database to be considered: the ER/PR is calculated for Max_{PR} values in the range [1%, …, 100%]. Each value of Max_{PR} corresponds to a maximum number of candidates $Max_C = Max_{PR} \cdot N_{DB}$, where N_{DB} is the total number of database fingerprints: if a candidate list is longer than Max_C, only the first Max_C candidates are considered to calculate the corresponding PR and ER values.

Figure 6 shows the pseudo-code describes the procedure in detail. Note that the length of the candidate list produced by an algorithm can be different for each query. A shorter candidate list will results in lower PR values, but may produce more errors, hence this aspect should be carefully considered [33, 28].

```
Let $C_i = \{ID_{i,1}, ID_{i,2}, \ldots, ID_{i,Ni}\}$ be the list of candidates returned by the algorithm for the $i^{th}$ query (containing $N_i$ candidates)
For $Max_{PR}$ = 1 to 100
    // Average PR considering at most $Max_{PR}$ % candidates
    $PR(Max_{PR}) = \underset{i}{Avg}\left(\frac{L(i,Max_{PR})}{N_{DB}}\right)$
    // Average ER considering at most $Max_{PR}$ % candidates
    $ER(Max_{PR}) = \underset{i}{Avg}(Err(i, L(i, Max_{PR})))$
End For
where:
// Actual length of the i^th candidate list considering at most $Max_{PR}$ % candidates
$L(i, Max_{PR}) = min\left(\frac{Max_{PR}, N_{DB})}{100}, N_i\right)$
$Err(i, L) = \begin{cases} 1 & if\ the\ i^{th}\ mate\ is\ not\ among\ the\ first\ L\ candidates\ in\ C_i \\ 0 & otherwise \end{cases}$
```

Figure 6. Pseudo-code of Performance Evaluation Generation for Proposed Direct Access.

4.1.2. Efficiency

Efficiency includes speed and scalability aspects. Based on [4, 5], speed performance of direct access strategy is evaluated by the flatness trend of average query search time as a function of number of candidates in the database. Meanwhile, scalability performance of direct access strategy is evaluated with an average ER graphs on a certain PR as a function of number of candidates in the database.

4.2. Parameters Calibration

Our direct-access strategy uses hill climbing optimization [34] to adjust parameter values that construct overall algorithm. Hill climbing is a mathematical optimization technique which belongs to the family of local search. Local search algorithms search several solutions in the of candidate-solutions space by incrementally changing a single factor of the solution, until a solution deemed optimal is found or a time bound is elapsed.

It attempts to maximize (or minimize) a target function $f(\mathbf{X})$, where $\mathbf{X}$ is a vector of continuous and/or discrete values and then $\mathbf{X}$ was said to be "locally optimal". This proposed direct-access strategy attempts to minimize a target function $f(\mathbf{X})$, where $\mathbf{X}$ is a vector of discrete values.

a) A separate data set has been used during some preliminary experiments to optimize various parameters of the proposed approach.In the following, this data set will be referred to as Calibration DB. It is FVC2000 DB1 A, the first FVC2000 database (2) which contains 800 fingerprints from 100 fingers, eight impressions per finger, 300x300 pixels, 500 DPI, and captured using low-cost optical sensor “Secure Desktop Scanner” by KeyTronic.

b) Both the parameters of feature extraction process (Chapter 2.2) and the indexing-retrieval processes (Chapter 2.3 - 2.5) have been tuned on the Calibration DB and have been used for all of the experiments described in this Chapter 4.
c) Optimizing various parameters based on target function.

$$f(\mathbf{X}) = \sum_{i=1}^{i_{max}} w_{m_i} \cdot ER_{m_i}(\mathbf{X}) \tag{17}$$

Where **X** is a vector of discrete parameters (50 parameters for features extraction process and 21 parameters for indexing-retrieval process). $ER_{m_i}(\mathbf{X})$ is the lowest PR for ER ≤ n_i, with $i_{max} = 20$ then m_i (in %) = 100 / n_i, and n_i (in %) = {15, 13, 11, 10, 9, 7, 5, 4, 3, 2, 1, 0.9, 0.8. 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1}. So we have m_i = {7, 8, 9, 10, 11, 14, 20, 25, 33, 50, 100, 111, 125, 143, 167, 200, 250, 333, 500, 1000}. Different weighting applied for w_{m_i} since lower $ER_{m_i}(\mathbf{X})$ for bigger *i* is more important (4).

$$w_{m_i} = \frac{i_{max} - (n_i - n_{i_{max}})}{i_{max}} \tag{18}$$

Figure 7 shows the optimized minimum *f*(**X**) which is the trade-off between the PR and ER (Chapter 4.1) on the Calibration DB.

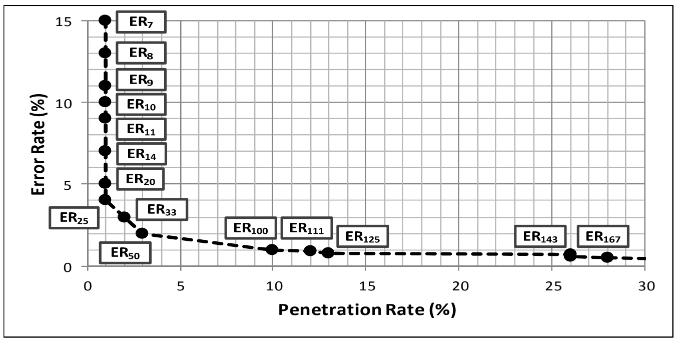


Figure 7. The Optimized Minimum *f*(**X**) on the Calibration DB.

4.3. Result: Comparison

In order to evaluate comparison of multi-thread implementation to its single-thread implementation of proposed direct-access strategy, a large public fingerprint data set provided by CASIA (35) was used. The experiments have been carried out as follows:
a) 20.000 impressions from 4.000 fingers (five impressions per finger) were used. First impression for index creation, and other four for searching (as query impressions).
b) Four searching tests have been performed, increasing the database size *N* from 1000 to 4.000 impressions (adding *N* = 1000 per test).
c) For each query, $N_T = N/10$ candidates have been retrieved.

For each of the fifteen tests, the following performance indicators have been measured:
a) Average total search time, i.e. time of extractor, pre-filter, and matcher (Figure 1).
b) Average ER at an average PR of 10% (Chapter 4.1).

Figure 8 shows measured performance indicators above as functions of database size *N*. Several aspects about comparison:
a) Single-thread implementation seems slight faster on average search time than its multi-thread implementation for first and second test. There is turning point where at third and fourth test, multi-thread implementation faster on average search time than its single-thread implementation.
b) For the fourth test, multi-thread implementation can reduce average search time about 17% compared to its single-thread implementation. For further bigger database size, we predict multi-thread implementation stil can reduce average search time at certain vaule more than 17% until saturation status (maximum work of threads) was reached.

c) Analyzing previous point, it seems for smaller database size (first and second test), multi-thread implementation has relatively constant value of additional overhead time (for maintaining its parallel process) contribute to total average search time. This additional overhead time can not compensate to give faster total average search time compared to total average search time of its single-thread implementation.
d) There was slight difference in average accuracy between multi-thread and its single-thread implementation, as shown by graph at right side of Figure 8. The slight difference in accuracy was caused by internal random process and slight difference in template data generation during extraction process.

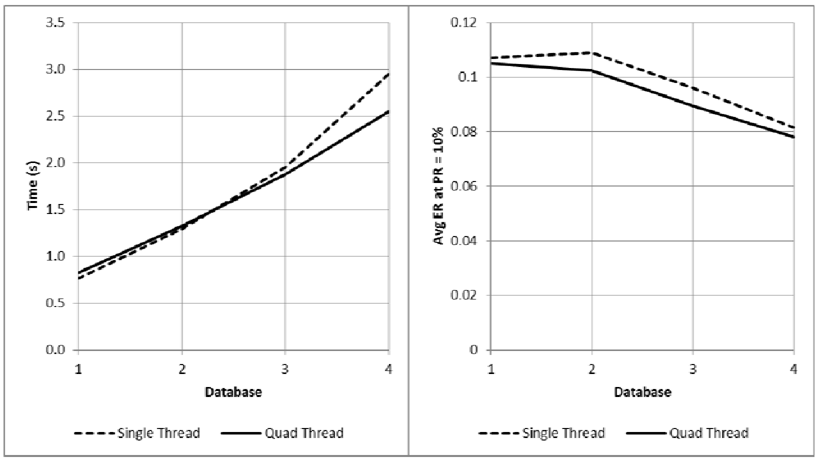


Figure 8. Average total search time (left) and average accuracy indicators (right) of proposed direct-access strategy as functions of the database size *N*.

5. Conclusion

Multi-thread (parallel process) implementation for new method of fingerprint direct-access strategy has already been proposed. Mainly, during query (Figure 5), multi-thread applied on lookup process of hashing data structure (considered as data partitioning) and on similarity score computation (considered as work partitioning). With careful design of multi-thread process, start on certain database size, its implementation confirmed faster average search time result compare to its single-thread implementation. Using quad-thread of dual-core single processor on last two tests on experiment, multi-thread implementation can reduce average search time. More specific at last tests, it can reduce about 17% average search time compare to its single-thread implementation with relatively same accuracy trend (Figure 8).

Acknowledgements

Portions of the research in this paper use the CASIA-FingerprintV5 (http://biometrics.idealtest.org/) collected by the Chinese Academy of Sciences' Institute of Automation (CASIA).

References

[1] Indrawan G. Direct Access Strategy for Fingeprint Data. Bandung Institute of Technology; 2013.
[2] Maio D, Maltoni D, Cappelli R, Wayman JL, Jain AK. FVC2000: Fingerprint Verification Competition. *IEEE Trans Pattern Anal Mach Intell.* 2002; 24(3): 402–12.
[3] Maio D, Maltoni D, Cappelli R, Wayman JL, Jain AK. *FVC2002: Second Fingerprint Verification Competition*. Proc 16th Int'l Conf Pattern Recognit. 2002; 811–4.
[4] BioLab - UniBo. FVC-onGoing | Benchmark area: Fingerprint Indexing [Internet]. 2013 [cited 2013 Oct 25]. https://biolab.csr.unibo.it/fvcongoing/UI/Form/ BenchmarkAreas/BenchmarkAreaFIDX.aspx
[5] Cappelli R, Ferrara M, Maltoni D. Fingerprint Indexing Based on Minutia Cylinder-Code. IEEE Trans Pattern Anal Mach Intell. 2011;33(5):1051–7.

[6] Jiang X, Liu M, Kot AC. Fingerprint Retrieval for Identification. *IEEE Trans Inf Forensics Secur*. 2006; 1(4): 532–42.
[7] Cappelli R, Maio D, Maltoni D. A Multi-Classifier Approach to Fingerprint Classification. *Pattern Anal Appl.* 2002; 5(2): 136–44.
[8] Cappelli R, Lumini A, Maio D, Maltoni D. Fingerprint Classification by Directional Image Partitioning. *IEEE Trans Pattern Anal Mach Intell.* 1999; 21(5): 402–21.
[9] Lee SO, Kim YG, Park GT. *A Feature Map Consisting of Orientation and Inter-Ridge Spacing for Fingerprint Retrieval.* Proc 5th Int Conf AVBPA. 2005: 184.
[10] Li J, Yau WY, Wang H. *Fingerprint Indexing Based on Symmetrical Measurement.* Proc 18th ICPR. IEEE; 2006: 1038–41.
[11] Liu M, Jiang X, Kot AC. *Fingerprint Retrieval by Complex Filter Responses.* Proc 18th ICPR. IEEE; 2006. p. 1042.
[12] Germain RS, Califano A, Colville S. Fingerprint Matching Using Transformation Parameter Clustering. *IEEE Comput Sci Eng*. 1997; 4(4): 42–9.
[13] Bhanu B, Tan X. Fingerprint Indexing Based on Novel Features of Minutiae Triplets. IEEE Trans *Pattern Anal Mach Intell.* 2003; 25(5): 616–22.
[14] Liang X, Asano T, Bishnu A. *Distorted Fingerprint Indexing Using Minutia Detail and Delaunay Triangle.* Proc 3rd ISVD. 2006; 217–23.
[15] De Boer J, M Bazen A, H Gerez S. *Indexing Fingerprint Databases based on Multiple Features*. Proc Work Circuits, Syst Signal Process. 2001: 300–6.
[16] Biswas S, K Ratha N, Aggarwal G, Connell J. *Exploring Ridge Curvature for Fingerprint Indexing.* Proc IEEE 2nd Int Conf BTAS. 2008; 1–6.
[17] Shuai X, Zhang C, Hao P. *Fingerprint Indexing Based on Composite Set of Reduced SIFT Features.* Proc 19th ICPR. IEEE; 2008; 1–4.
[18] Gyaourova A, Ross A. A Novel Coding Scheme for Indexing Fingerprint Patterns. Proc 7th Int'l Work S+SSPR. Orlando, FL; 2008; 765–74.
[19] Maeda T, Matsushita M, Sasakawa K. Identification Algorithm Using A Matching Score Matrix. *IEICE Trans Inf Syst - Spec Issue Biometric Pers Authentication.* 2001; E84-D(7): 819–24.
[20] Silberschatz A, F Korth H, Sudarshan S. Database System Concepts. 6th ed. New York: McGraw-Hill; 2011.
[21] K Ratha N, D Pandit V, M Bolle R, Vaish V. *Robust Fingerprint Authentication Using Local Structural Similarity*. Proc Work Appl Comput Vis. 2000; 29–34.
[22] Maltoni D, Maio D, Jain AK, Prabhakar S. Handbook of Fingerprint Recognition. 2nd ed. London: Springer-Verlag. 2009.
[23] Vazan R. SourceAFIS | Fingerprint Recognition Library for .NET and Experimentally for Java. 2009 [cited 2013 Oct 25]. Available from: http://sourceforge.net/projects/sourceafis/
[24] Watson CI, Garris MD, Tabassi E, Wilson CL, McCabe RM, Janet S, et al. User's Guide to NIST Biometric Image Software. Gaithersburg. 2004.
[25] Indrawan G, Sitohang B, Akbar S. *Parallel Processing for Fingerprint Feature Extraction.* Int Conf Electr Eng Informatics. Bandung. 2011.
[26] Watson CI, Garris MD, Tabassi E, Wilson CL, McCabe RM, Janet S, et al. User's Guide to Export Controlled Distribution of NIST Biometric Image Software. *Gaithersburg.* 2004.
[27] Indrawan G, Sitohang B, Akbar S. *EER Distribution Consideration for Optimal Matching Threshold of Multi-Finger Fingerprint Recognition.* Int Conf Comput Sci Electron Instrum. Yogyakarta. 2012.
[28] Cappelli R, Ferrara M, Maio D. Candidate List Reduction Based on The Analysis of Fingerprint Indexing Scores. *IEEE Trans Inf Forensics Secur.* 2011; 6(3): 1160–4.
[29] Albahari J. Threading in C#. C# 40 a Nutshell. O'Reilly; 2013.
[30] Indrawan G, Sitohang B, Akbar S. Review of Sequential Access Method for Fingerprint Identification. *TELKOMNIKA Indonesian Journal of Electrical Engineering.* 2012; 10(2).
[31] Mitchell S. Microsoft Developer Network | An Extensive Examination of Data Structures Using C# 2.0. 2005 [cited 2013 Oct 25]. Available from: http://msdn.microsoft.com/en-us/library/hh830851(v=vs.80).aspx
[32] Intel. ARK | Intel® Core^{TM} i5-2430M Processor (3M Cache, up to 3.00 GHz). [cited 2013 Oct 25]. Available from: http://ark.intel.com/products/53450/
[33] Cappelli R. Fast and Accurate Fingerprint Indexing based on Ridge Orientation and Frequency. IEEE Trans Sys, Man, Cybern B, Cybern. 2011 Dec 23;41(6):1511–21.
[34] Russell SJ, Norvig P. Artificial Intelligence: A Modern Approach. 2nd ed. New Jersey: Upper Saddle River. 2003; 111–4.
[35] CASIA. Biometrics Ideal Test. 2010 [cited 2013 Oct 25]. Available from: http://biometrics.idealtest.org/