November 2022

Random Sample Consensus Algorithm with Enhanced Latency for Fingerprint Matching

Firas Sammoura

Josie Nordrum

Random Sample Consensus Algorithm with Enhanced Latency for Fingerprint Matching

Abstract:

This publication describes a random sample consensus algorithm with enhanced latency for fingerprint matching. A fingerprint image can be represented by a fingerprint image template which includes key points (X and Y locations) and feature vectors. Key points correspond to various minutiae points of the fingerprint image. Feature vectors are rotationally invariant encodings of image blocks centered around key points. Fingerprint matching is done by comparing feature vectors from a verify fingerprint image template to feature vectors from an enrolled fingerprint image template to generate matching key point pairs. A geometric transformation between the verify fingerprint image template and the enrolled fingerprint image template is inferred by a random sample and consensus process of matching key point pairs. The geometric transformation between two matching fingerprint image templates is mainly a rigid two-dimensional (2D) transformation with a translation vector, a rotation matrix, and minimal stretch. Rather than comparing every matching key point pair to infer the geometric transformation, the disclosed fingerprint matching algorithm implements a stretch ratio check. For any two key points, the stretch ratio is the distance between the two key points from the verify fingerprint image template divided by the distance between matching key points from the enrolled fingerprint image template. If the stretch ratio falls outside an acceptable range of stretch ratios, the fingerprint matching algorithm skips the set of matching key point pairs of that stretch ratio when inferring the geometric transformation. By so doing, the fingerprint matching algorithm improves matching speed and makes matching and non-matching latencies consistent.

Keywords:

Fingerprint, enrolled fingerprint image, verify fingerprint image, minutia, minutiae, fingerprint matching algorithm, feature matching algorithm, latency, match latency, no-match latency, feature vector, biometric, authentication, machine learning, random sample consensus, M-estimator sample and consensus, MSAC, ML, artificial intelligence, AI.

Background:

Human fingerprints are detailed, nearly unique, difficult to alter, and durable over a lifetime. These characteristics make fingerprints ideal long-term markers of human identity. In modern times, a common use for fingerprint recognition is in enabling individuals to gain access to a door entrance, a vault, application software, or a computing device (e.g., smartphone). Figure 1 illustrates an example of a fingerprint image.

Figure 1

The analysis of fingerprints for matching purposes generally compares patterns and minutiae of an enrolled fingerprint image to those of a verify fingerprint image. Three main patterns of a fingerprint image include an arch, a loop, and a whorl. An arch is a fingerprint ridge that enters from one side of the image, rises in the center to form an arc, and exits the other side of the image. A loop is a fingerprint ridge that enters from one side of the image, forms a curve, and

exits the same side of the fingerprint image. A whorl is a fingerprint ridge that forms a circular shape (e.g., circle, oval) around a central point. Some of these patterns are illustrated in Figure 1. Minutiae of the fingerprint image are features of fingerprint ridges, including ridge endings, dots, spurs, deltas, bifurcations, bridges, islands, trifurcations, and so forth. Various minutiae are illustrated in Figure 2.

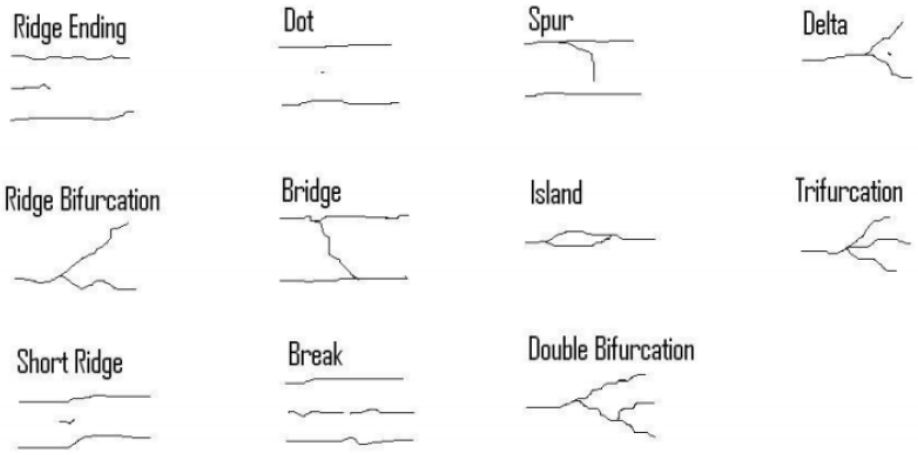

Figure 2

A smartphone may utilize a fingerprint sensor to capture a fingerprint image. If the fingerprint sensor is large, then minutia-based and pattern-based fingerprint matching approaches are achieved with high success rates. However, as the smartphone industry trends towards smaller sensors, these smaller sensors capture less of a fingerprint and thus less of the minutiae and patterns in the fingerprint image. In such implementations, the smartphone may struggle to make a positive minutia-based or pattern-based match.

Small fingerprint sensors are one reason why many smartphone manufacturers may use a vector-based fingerprint matching algorithm. The vector-based approach includes creating a template for a fingerprint image. The template includes key points (X and Y locations), which may correspond to specific minutiae, and feature vectors, which are rotationally invariant encodings of image blocks (e.g., NxN pixels) centered around key points. Matching is done by

comparing feature vectors from a verify image template to feature vectors of an enrolled image template to generate matching key point pairs between the two templates. The matching key point pairs may contain a subset of outliers, which are not consistent with a geometric transformation (e.g., rotational, translational) between two matching fingerprint images. The geometric transformation may be inferred by a random sample and consensus process, such as M-estimator random sample and consensus (MSAC). However, legacy MSAC approaches must exhaust all attempts at determining the geometric transformation based on matching key point pairs before giving up or determining a match. This process results in a difference between match and non-match latencies, as well as a higher matching latency, which is an issue from a user experience perspective.

Description:

A user of a smartphone may enroll a fingerprint during a setup process of the smartphone to generate an enrolled fingerprint image template. For example, during the setup process, a biometric authentication manager of the smartphone may prompt the user to scan a fingerprint multiple times (e.g., ten, fifteen) at various orientations and pressures. From fingerprint images produced by the scanning, the biometric authentication manager extracts key points (X and Y locations), which correspond to various minutiae (e.g., spurs, dots, bifurcations) or patterns (e.g., arches, loops, whorls) of the fingerprint images. The biometric authentication manager then creates feature vectors for the extracted key points. The feature vectors may be rotationally invariant encodings (e.g., calculated by polar shapelet-based functions) of image blocks (e.g., NxN pixels) centered around key points. Figure 3 illustrates an example enrolled fingerprint image template having two key points.

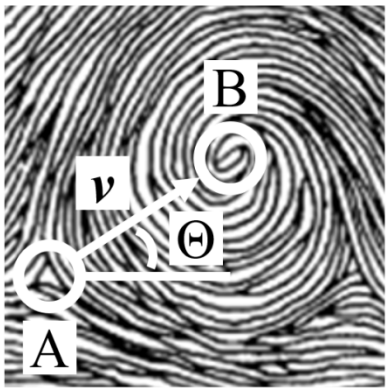

Figure 3

As illustrated in Figure 3, a first key point, labeled 'A', identifies a delta pattern in the fingerprint image at (x_0, y_0). A second key point, labeled 'B', identifies a whorl pattern in the fingerprint image at (x_1, y_1). The first and second key points A and B, respectively, may each have a respective feature vector or other rotationally invariant encoding (not shown). A vector labeled '$\boldsymbol{v}$' may be created between the two key points A and B. In the present example, the vector $\boldsymbol{v}$ has a direction Θ (e.g., 40°) and a magnitude defined in Equation 1.

$$|v| = |AB| = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} \qquad \textbf{(1)}$$

Although only two key points are shown, the enrolled image template may include many (e.g., tens, hundreds) key points. Additionally, the enrolled image template may not include all the patterns and minutiae but rather include only the key points and their associated feature vectors.

When the user wishes to access features and functions of the smartphone, the user may scan a verify fingerprint to generate a verify fingerprint image template. To do so, the user places the verify fingerprint on the fingerprint sensor, and, in response, the biometric authentication manager scans the verify fingerprint to generate the verify fingerprint image template. Figure 4 illustrates an example verify fingerprint image template having two key points.

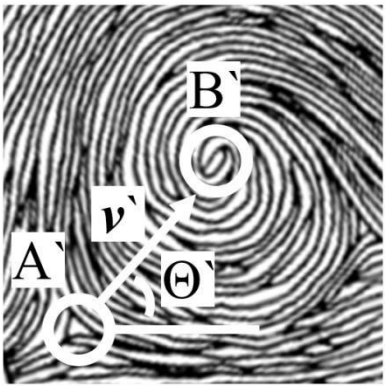

Figure 4

As illustrated in Figure 4, a first key point, labeled 'A`', identifies a delta pattern in the fingerprint image at ($x`_0$, $y`_0$). A second key point, labeled 'B`', identifies a whorl pattern in the fingerprint image at ($x`_1$, $y`_1$). The first and second key points A` and B`, respectively, may each have a respective feature vector or other rotationally invariant encoding (not shown). A vector labeled '$\boldsymbol{v}$`' may be created between the two key points A` and B`, which has a direction Θ` (e.g., 60°) and a magnitude defined in Equation 2.

$$|v`| = |A`B`| = \sqrt{(x`_1 - x`_0)^2 + (y`_1 - y`_0)^2} \qquad \textbf{(2)}$$

In the present example, the verify fingerprint and the enrolled fingerprint are a same fingerprint. However, the user may scan the verify fingerprint at a different orientation or pressure compared to those of the enrolled fingerprint. If scanned at a different pressure, the vectors $\boldsymbol{v}$ and $\boldsymbol{v}$` do not share the same magnitude. In other words, a distance between two key points may stretch or compress based on the pressure applied during fingerprint scanning. If scanned at a different orientation, the vectors $\boldsymbol{v}$ and $\boldsymbol{v}$` do not share a same direction. In other words, Θ and Θ` are not equal. Figure 5 illustrates the enrolled fingerprint image template and the verify fingerprint image template scanned at a different orientation.

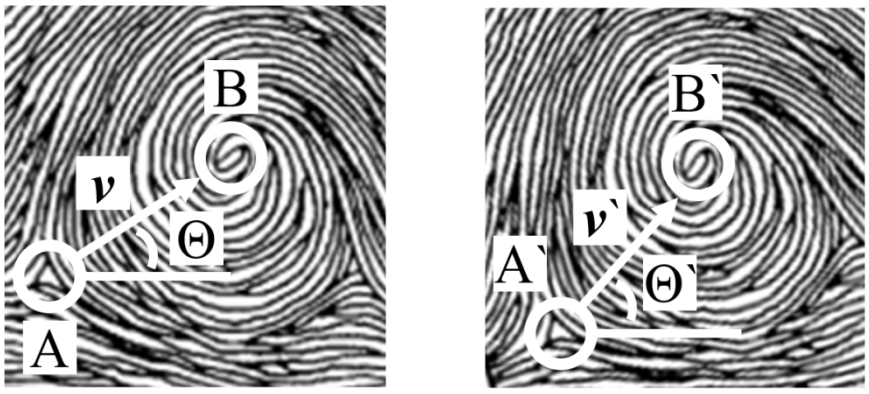

Figure 5

As illustrated in Figure 5, Θ is less than Θ`, resulting from the user scanning the verify fingerprint (right) at the different orientation. However, the biometric authentication manager may infer a geometric transformation between the two fingerprint image templates. The geometric transformation, especially for small-area fingerprint sensors, is mainly a rigid two-dimensional (2D) transformation with a translation vector ***T***, a rotation matrix ***R***, and minimal stretch (no affine 2D). The biometric authentication manager determines the geometric transformation using the MSAC process outlined below:

Initialize

1) Threshold: *error*

2) Confidence level: *confidence_level*

3) Maximum iterations: *max_iterations*

4) Maximum cost C_m = *error* * *number of key point pairs*

5) Final transformation matrix = *identity_matrix*

6) Inliers = *{}*

Loop while *i* < *max_iterations*

1) *i*++

2) Randomly pick two matching key point pairs (A, A`) and (B, B`)

3) Form transformation matrix *matrix_i*

4) Transform all points from *view_1* to *view_2* using *matrix_i*

5) Calculate distance d_j between A_j and A_i^T

a) If $d_j > error$, $d_j = error$

6) Calculate Cost as sum of all distances

7) If Cost < C_m

a) C_m = Cost

b) Inliers: $d_j < error$

c) Final transformation matrix = *matrix_i*

d) *inlier_ratio* = number of inliers / total number of key point pairs

e) *max_iterations* = log(1 – *confidence_level*) / log(1 – *inlier_ratio*^{2})

f) if *i* < *max_iterations*, continue at Step 1 of the loop

g) if *i* > *max_iterations*

g.1) *Matrix_final* = transformation matrix using all inliers

g.2) Recalculate inliers

g.3) Break out of the loop

As outlined, the MSAC process includes an initialization sequence that defines various parameters. The various parameters include a threshold (*error*), a confidence level (*confidence_level*), a maximum number of iterations (*max_iterations*), a maximum cost (C_m), a final transformation matrix (*identity_matrix*), and an inliers list (*{}*). These parameters are utilized to make decisions (described below) throughout the MSAC process.

The biometric authentication manager then, for all key point pairs, randomly selects two matching key point pairs. The biometric authentication manager may, for example, use polar

shapelet-based functions to produce feature vectors for key points and, based on the feature vectors, identify that A and A` are the same key point and that B and B` are the same key point. After identifying the matching key point pairs (A, A`) and (B, B`), the biometric authentication manager forms a transformation matrix (*matrix_i*). Using the transformation matrix (*matrix_i*), the biometric authentication manager may transform all points from a first view (*view_1*) to a second view (*view_2*). The first view (*view_1*) may be the verify fingerprint image, and the second view (*view_2*) may be a fingerprint image generated after all key points from the verify fingerprint image are transformed using the transformation matrix (*matrix_i*).

Next, a distance (d_j) is calculated between a key point (A_j) and a transformed key point (A_i^T). If the distance (d_j) is larger than the threshold (*error*) defined in the initialization sequence, then the distance (d_j) is set to the threshold (*error*). All distances are summed to calculate a cost for the transformation. If the cost is less than the maximum cost (C_m) defined in the initialization sequence, then the cost is set to the maximum cost (C_m). Next, the biometric authentication manager populates the inliers list ($\{\}$) with all transformed key points whose distance (d_j) is less than the threshold (*error*) and sets the final transformation matrix (*identity_matrix*) to the transformation matrix (*matrix_i*). The biometric authentication manager calculates an inlier ratio (*inlier_ratio*) as the quotient of the number of inliers and the total number of key point pairs. The biometric authentication manager then sets the maximum number of iterations (*max_iterations*) using Equation 3.

$$max\_iterations = \frac{log\,(1-confidence\_level)}{log\,(1-inlier\_ratio^2)} \qquad \textbf{(3)}$$

If the current iteration (i) is less than the maximum number of iterations (*max_iterations*), then the biometric authentication manager continues at Step 1 of the loop. Otherwise, the biometric authentication manager calculates a geometric transformation matrix (*matrix_final*) as

the final transformation matrix (*identity_matrix*) using all the inliers, recalculates the inliers, and exits the loop.

As described above, the biometric authentication manager completes all iterations of the MSAC process loop before determining the geometric transformation matrix (*matrix_final*). This can result in a higher matching latency as well as a larger difference between match and non-match latencies. For example, suppose the user scans a different fingerprint from the fingerprint used to generate the enrolled fingerprint image template. The result is an incorrect geometric transformation matrix because the verify fingerprint and the enrolled fingerprint are different fingerprints. However, in the present example, the biometric authentication manager must exhaust all iterations of the MSAC process loop before arriving at the incorrect geometric transformation matrix. As another example, suppose the user scans a same fingerprint as the enrolled fingerprint but with a different pressure. This can result in a different magnitude of a same vector connecting two key points due to non-linear distortion of the fingerprint as it touches the fingerprint sensor. In this example, even though the fingerprint is the same fingerprint as the enrolled one, the biometric authentication manager may arrive at an incorrect geometric transformation matrix because the magnitude of the same vector connecting two key points is different from that of the enrolled fingerprint image template vector. Again, however, the biometric authentication manager must exhaust all iterations of the MSAC process loop before arriving at the incorrect geometric transformation matrix.

In contrast, the disclosed techniques and systems, which check for a stretch ratio before building a geometric transformation matrix, enhance the latency of the fingerprint matching algorithm and improve the non-match experience. To demonstrate the enhanced latency and improved non-match experience, refer to Figure 5. For the enrolled fingerprint image template

(left) and the verify fingerprint image template (right), the stretch ratio is calculated using Equation 4.

$$stretch\_ratio = \frac{|v|}{|v`|} = \frac{|AB|}{|A`B`|} \qquad \textbf{(4)}$$

The biometric authentication manager may also determine upper (S_u) and lower (S_l) stretch ratio thresholds, for example, through experimentation. The biometric authentication manager implements this stretch ratio check at Step 3 of the loop in the MSAC process outlined below:

Initialize

1) Threshold: *error*

2) Confidence level: *confidence_level*

3) Maximum iterations: *max_iterations*

4) Maximum cost C_m = *error* * *number of key point pairs*

5) Final transformation matrix = *identity_matrix*

6) Inliers = *{}*

Loop while *i* < *max_iterations*

1) *i*++

2) Randomly pick two matching key point pairs (A, A`) and (B, B`)

3) If *stretch_ratio* > S_u or *stretch_ratio* < S_l, continue at Step 1 of the loop

4) Form transformation matrix *matrix_i*

5) Transform all points from *view_1* to *view_2* using *matrix_i*

6) Calculate distance d_j between A_j and A_i^T

a) If d_j > *error*, d_j = *error*

7) Calculate Cost as sum of all distances

8) If Cost < C_m

a) C_m = Cost

b) Inliers: $d_j < error$

c) Final transformation matrix = *matrix_i*

d) *inlier_ratio* = number of inliers / total number of key point pairs

e) *max_iterations* = log(1 – *confidence_level*) / log(1 – *inlier_ratio*^{2})

f) if $i < max\_iterations$, continue at Step 1 of the loop

g) if $i > max\_iterations$

g.1) *Matrix_final* = transformation matrix using all inliers

g.2) Recalculate inliers

g.3) Break out of the loop

Although a 2D example was used to demonstrate an improvement in the biometric authentication manager using the MSAC process with the stretch ratio check, the process can be applied to three-dimensional (3D) rigid transformation problems as well. By so doing the stretch ratio check at Step 3 of the loop, the biometric authentication manager eliminates selecting bad transformation models that do not satisfy fingerprint rigidity. Additionally, the biometric authentication manager improves matching speed and makes matching and non-matching latencies consistent, resulting in an improved user experience.

References:

[1] Patent Publication: US20200265211A1. Fingerprint Distortion Rectification Using Deep Convolutional Neural Networks. Priority Date: February 14, 2019.

[2] Patent Publication: US20020126883A1. System and Method for Transforming Fingerprints to Improve Recognition. Priority Date: January 7, 1998.

[3] Zhou, Jun, Wang, Chao, Wang, Fang, and Ma, Jinhai. "A Fingerprint Image Enhancement Algorithm Based on Stretching Transfer Function." Journal of Physics Conference Series. August 2019.

https://www.researchgate.net/publication/335575684_A_Fingerprint_Image_Enhancement_Algorithm_Based_on_Stretching_Transfer_Function.

[4] Ghafoor, Mubeen, Taj, Imtiaz Ahmad, and Jafri, Mohammad Noman. "Fingerprint Frequency Normalisation and Enhancement Using Two-Dimensional Short-Time Fourier Transform Analysis." The Institution of Engineering and Technology. July 4, 2016. https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/iet-cvi.2016.0005.

[5] Patent Publication: US20160132715A1. Method and System for Rectifying Distorted Fingerprint. Priority Date: May 20, 2014.

[6] Patent Publication: US20130094724A1. Method of Performing Fingerprint Matching. Priority Date: October 18, 2011.

[7] Patent Publication: TW202034210A. Noise Elimination Method for Fingerprint Identification, Control Device, and Information Processing Device Eliminates Interference of Common-Mode Noise in Fingerprint Recognition Using a Software. Priority Date: March 5, 2019.