Extracting Valley-Ridge Lines from Point-Cloud-Based 3D Fingerprint Models

Xufang Pang and Zhan Song • Shenzhen Institutes of Advanced Technology
Wuyuan Xie • Chinese University of Hong Kong

Automated fingerprint identification systems (AFISs), one of the most important biometric techniques, are widely used for personal identification and security. AFISs typically employ four steps: fingerprint acquisition, preprocessing, feature extraction, and feature matching. Acquisition is often considered the most critical step because it determines the captured fingerprints’ quality and directly affects the final identification accuracy. With current AFISs, the subject presses or rolls a finger against a platen (for example, glass, silicon, or polymer), and a sensor or camera captures the fingerprint. Such touch-based operation often produces degraded fingerprint images due to inappropriate finger placement, skin deformation and smearing, or sensor noise from the wear and tear of finger surface coatings.

We’ve developed a device for touchless, efficient 3D fingerprint acquisition and an approach for directly extracting valley-ridge lines from point-cloud-based 3D fingerprint models. First, we apply the moving least-squares (MLS) method to fit a local paraboloid surface and to represent the local point cloud area. On the basis of the fitting surface, we calculate the 3D fingerprint surface’s curvature and curvature tensors. By referring to the curvatures, we detect potential valley-ridge points. Through statistical means, we project those points to the most likely valley-ridge lines. Then, by growing the polylines that approximate the projected points and removing the perturbations between the sampled points, we obtain the 3D valley-ridge lines. This approach can directly extract the features of valley-ridge lines without employing unwrapping, which converts 3D models to 2D but introduces distortions. (For more on unwrapping and touchless 3D fingerprint extraction, see the sidebar).

The 3D Fingerprint Acquisition System
To obtain the 3D fingerprint data we report on here, we used a photometric stereo 3D reconstruction system. This system comprises a camera with a resolution of 659 × 493 pixels and seven LED lamps mounted around it (see Figure 1). By synchronizing the camera and lamps, we can capture seven fingerprint images within 0.2 second. To represent the finger surface’s reflectance, we use a simplified Hanrahan-Krueger model. Using the calibrated lighting directions and image intensities in seven images, we estimate the surface normal at each image point by solving a nonlinear equation. Through surface normal integration, we obtain 3D models of fingerprints (see Figure 2).
Extracting Valley-Ridge Lines

Valley-ridge lines are one of the most distinctive features used in traditional 2D AFISs. For 3D fingerprints, these lines are usually detected from the unwrapped 3D models. How to directly extract such features from a raw point-cloud-based 3D fingerprint model remains an open issue. A fingerprint’s 3D point cloud can be viewed as a discrete model. To represent such a model, researchers usually employ the MLS surface, which lets them conveniently calculate the point set’s geometric properties.

Geometric Analysis of a 3D Fingerprint Model

We view the obtained 3D fingerprint model as an organized point set $P$, which records the fingerprint “ground truth”:

$$P = \{ p_{ij}, \, n_{ij} \},$$

where $p_{ij} \in \mathbb{R}^3$, $i \in \{1, ..., N\}$, and $j \in \{1, ..., M\}$. $P$ consists of points $p_{ij}$ and their corresponding normals $n_{ij}$. $N$ and $M$ indicate the size of $P$.

Owing to the generated 3D fingerprint data’s regularity and continuity properties, we can fit a local paraboloid surface to the local neighbors of each point in a predefined window with the size of $\delta$. For each $p_{ij}$, we define a local reference plane $H$ by its surface normal. Then, we create a local coordinate frame $(p_{ij}, x, y, n_{ij})$ on $H$, where $p_{ij}$ is the origin and $n_{ij}$ is the surface normal vector. By translating the neighborhoods of $p_{ij}$ to the local coordinate frame defined by $H$, we use a polynomial to fit these transformed points.
We do this by minimizing the sum of the weighted squared distances between the fitting points \( p_{k(i,j)} \) and \( p_{ij} \) in the original point cloud:

\[
\arg \min_{\theta} \sum_{u \in \frac{\partial}{\partial x}} \sum_{v \in \frac{\partial}{\partial y}} \left( p_{k(i,j)+u+v} - P_{g(i,j)+u+v} \right)^2 \theta(d),
\]

where \( d = \| p_{k(i,j)+u+v} - p_{ij} \| \) and \( \theta(d) \) is an exponentially decreasing weighting function—for example, \( \theta(d) = e^{-d^2/h^2} \)—that assigns larger weights to the points closer to \( p_{ij} \). The parameter \( h \) indicates the point cloud’s average distance, \( u \) and \( v \) are indexes for the neighboring points of \( p_{ij} \), and \( P_{g(i,j)+u+v} \) is the projection of \( p_{k(i,j)+u+v} \) via the polynomial fitting function \( g_{ij} \) along direction \( n_{ij} \). We define \( g \) as a parabolic surface:

\[
g(x, y) = axy + bx^2 + cy^2.
\]

Then, for any point \( q \) near \( p_{ij} \), we compute its projection on the fitting surface \( g_{ij} \) as \( p_{ij} + (x_q, y_q, g_{ij}(x_q, y_q)) \), where \( x_q, y_q \) are the local coordinates of \( q \).

On the basis of a sequence of analytic derivations, we calculate the local coordinates of \( g_{ij} \) from \( g_{ij} \). We compute the curvature tensor matrix of \( g_{ij} \) as the Hessian matrix of \( g_{ij} \):

\[
H(g_{ij}) = \begin{pmatrix} g_{ij} \\ \frac{\partial g_{ij}}{\partial x} & \frac{\partial g_{ij}}{\partial y} \\ \frac{\partial g_{ij}}{\partial x} & \frac{\partial g_{ij}}{\partial y} \end{pmatrix} = \begin{pmatrix} 2b & a \\ a & 2c \end{pmatrix},
\]

where \( \partial \) is the partial-derivative symbol and \( V \) is the vector differential operator.

With the eigenvalues and eigenvectors of \( H(g) \), we separately compute the principal curvatures \( K_{\max,\min} \) and the principal directions \( T_{\max,\min} \).

\[
K_{\max,\min} = \lambda_{1,2} = c + b \pm \sqrt{(c - b)^2 + a^2},
\]

\[
T_{\max,\min} = \frac{a}{K_{\max,\min} - 2b}.1.0
\]

For the special case of \( a = 0 \), we compute the principal curvatures and the principal directions as

\[
K_{\max} = 2b, K_{\min} = 2c,
\]

\[
T_{\max} = (1,0,0), T_{\min} = (0,0,1),
\]

\[
(b > c),
\]

and

\[
K_{\max} = 2c, K_{\min} = 2b,
\]

\[
T_{\max} = (0,0,1), T_{\min} = (1,0,0),
\]

\[
(b < c).
\]

As Figure 3 shows, the computed curvature and curvature tensor maps can match the surface shape well. For simplicity, in the following text, biggish principal curvature refers to the principal curvature of each point with the bigger absolute value.

Lambertus Hesselink and his colleagues showed how to detect degenerate points on second-order tensor fields. Inspired by them, we deduced a scheme to evaluate the confidence of the principal directions for the curvature tensors. According to Equation 1, we compute this confidence for each point on the 3D fingerprint model:

\[
w = \frac{\partial g_{ij}}{\partial x} \frac{\partial g_{ij}}{\partial y} + \frac{\partial g_{ij}}{\partial y} \frac{\partial g_{ij}}{\partial x} = (2b - 2c)^2 + a^2.
\]

As Figure 4 shows, the points with high confidence usually are in the regions with clear anisotropic features. The points with low confidence usually are in the transitional regions between the valley and ridge features. Because the principal directions in the low-confidence regions can’t align well with the surface shape, they aren’t suitable to guide generation of the following valley-ridge lines.

### Detecting and Filtering Valley-Ridge Points

Once we’ve obtained the principal curvatures of each point on the 3D fingerprint model, we can determine the potential valley-ridge points with respect to their biggish principal curvature \( k_{ij} \). We add the points with \( k_{ij} < 0 \) and \( |k_{ij}| > \tau \) (where \( \tau \) is a user-defined threshold) to valley point set \( V \); we add the points with \( k_{ij} > 0 \) and \( |k_{ij}| > \tau \) to ridge point set \( R \) (see Figure 5). For convenience, in the rest of the article we use \( F(P) \) to represent \( V \) or \( R \).

To remove the noisy feature points and outliers, we apply covariance analysis and cross-correlation coefficient estimation\(^{10}\) to filter the potential valley-ridge points. For each point \( p \in F(P) \), we identify its neighboring points in a neighborhood of \( \phi \).
Related Work in Touchless 3D Fingerprint Capturing

Current automated fingerprint identification systems (AFISs) have major limitations due to the elastic skin of the subject’s finger contacting the rigid touching surface. To solve this intrinsic problem, touchless 3D fingerprint-capturing approaches have recently emerged.

Geppy Parziale and his colleagues introduced an innovative 3D fingerprint acquisition system that employed five cameras and 16 green LED lamps. By imaging and illuminating the finger surface from different angles, Wolfgang Niem and Richard Szeliski reconstructed 3D models of fingerprints from silhouettes. Yongchang Wang and his colleagues devised a structured-light system for 3D reconstruction of fingerprints that used a projector and camera. By projecting sinusoidal fringe patterns onto the finger surface and capturing it with pattern illuminations, Jielin Li and colleagues and Wang and his colleagues obtained a dense 3D point cloud of a fingerprint.

We previously presented a photometric stereo-based method for real-time 3D reconstruction of fingerprints; the system consisted of a camera and some normal LED lamps. By capturing the fingerprint images from different lighting directions, we estimated the surface normal at each image point by solving a nonlinear surface reflectance modeling function. We achieved the 3D fingerprint model through surface normal integration. All these touchless sensing technologies provide repeatable, high-quality 3D fingerprints. In addition, the 3D fingerprint models provide depth information that makes the subsequent identification and recognition more robust and accurate.

To make 3D fingerprint models interoperable with traditional 2D image-based AFISs, researchers have studied unwrapping, which translates a 3D model to a 2D space. Abhishika Fatehpuria and her colleagues employed the springs algorithm to “unroll” a 3D fingerprint to a 2D space. Wang and his colleagues presented a fit-sphere unwrapping algorithm.

Unwrapping can introduce distortions. To limit those distortions within a given range, you can constrain the distances between neighboring points. However, the transformed 2D fingerprints won’t be completely compatible with the legacy wrapped fingerprints. Wang and his colleagues used simulated finger rolling to transform 3D fingerprints into a 2D space. To minimize unwrapping distortion, they used equidistance unwrapping, and they used the unwrapped 2D fingerprints for the matching procedure. The unwrapped fingerprints were touchless, or deformation-free, whereas the legacy rolled fingerprints involved noticeable, unavoidable, and unpredictable fingerprint skin deformation.

Yi Chen and her colleagues developed a 3D descriptor for fingerprint feature points. They expressed this descriptor as \( \{x, y, z, \theta, \phi\} \), where \( x, y, \) and \( z \) are the 3D coordinates, and the angles \( \theta \) and \( \phi \) are the ridge’s orientation in 3D space. They showed how to represent a 3D fingerprint model’s valley-ridge features by mapping image pixels to a 3D space according to the pixel intensity. However, they didn’t provide a detailed algorithm for extracting the features in 3D space.

References

By computing the eigenvalues $\lambda_0, \lambda_1,$ and $\lambda_2$ ($\lambda_0 \geq \lambda_1 \geq \lambda_2$) and the corresponding eigenvectors $V_0, V_1,$ and $V_2$ of $C$, we define the principal axis of $N(p)$ by $p$ and $V_0$. Then, we obtain the filtered new point $\hat{p}$ by projecting $p$ onto the principal axis $V_0$.

However, the filtering’s performance is sensitive to $\phi$. To achieve stable filtering, we use an adaptive neighborhood-growing scheme. First, we project the neighboring points $N(p)$ onto the plane defined by $V_0, V_1,$ and $p$. Next, we translate the projected points to the local coordinate frame $(\overline{p}, x, y, V_0)$, where $\overline{p}$ is the origin and $V_0$ is the $z$-axis. Then, we implement statistical analysis:

$$\rho_{XY} = \frac{\text{cov}(X, Y)}{\sqrt{\text{D}(X) \sqrt{\text{D}(Y)}}},$$

where

- $\rho_{XY}$ is the computed cross-correlation coefficient, which ranges from $-1$ to 1 and indicates the degree of linear dependence between random variables $X$ and $Y$,
- $\text{cov}(X, Y)$ is the covariance of $X$ and $Y$, and
- $\sqrt{\text{D}(\overline{p})}$ is the standard deviation.

If $|\rho_{XY}|$ is small, $\phi$ will be large; in this article, we use $|\rho_{XY}| \geq 0.5$. If $\phi$ is bigger than a predefined threshold, we abandon $p$ as an outlier or noise. Otherwise, we calculate the principal axis of the new $N(p)$. We project $p$ to the new $N(p)$ to create a new $\hat{p}$, which we add to the point set of $F(S)$, the smoothed feature points. Figure 6 shows a filtering example.

**Extracting Valley-Ridge Lines**

We generate the valley-ridge lines by growing the polylines along the directions of the maximal principal curvatures of $V(S)$ for valleys or $R(S)$ for ridges. Sampled points on the valley-ridge curves are separately added to the point sets $V(L)$ and $R(L)$, where $L$ means the valley or ridge lines. The ridges and valleys turn into each other while the surface orientation changes. So, without loss of generality, we consider only the lines of ridge curves.
Figure 4. Evaluating the confidence of the principal directions for the curvature tensors. (a) The calculated confidence map of principal directions under different thresholds. (b) The distribution of the surface points with lower confidence values on the 3D fingerprint model. The points with high confidence usually are in the regions with clear anisotropic features. The points with low confidence usually are in the transitional regions between the valley and ridge features.

Figure 5. Detecting potential valley-ridge points. (a) The potential valley points. (b) The potential ridge points. (c) A combination map of the potential valley and ridge points. Unlike 2D traditional fingerprinting methods, our algorithm extracts not only the ridge features but also the valley features.

Figure 6. Valley-ridge points after filtering. (a) The filtered valley points. (b) The filtered ridge points. (c) A combination map of the filtered valley-ridge points. The smoothed valley and ridge points can be obtained separately.
To extract the valley-ridge lines from the detected feature points, we first add all the points in \( R(S) \) to a priority queue. The points with higher confidence receive the higher priority. Then, we pop a new seed point \( p_i \) from the queue and start a tracing procedure from \( p_i \) in directions \( T_{\text{min}} \) and \(-T_{\text{min}}\). Suppose the current generated curve is \( L = \{ l_{-m}, \ldots, l_{-1}, l_0, l_1, \ldots, l_n \} \), where \( l_k \) or \( l_{-k} \) is the \( k \)th sample point in \(-T_{\text{min}}\) or \( T_{\text{min}} \). Tracing then involves the following three steps.

First, we initially add \( p_i \) to \( L \) and call it \( l_0 \). Then, we generate a new sample point on the current curve:

\[
l_k = l_{k-1} + s \cdot d_{k-1},
\]

where \( s \) is a user-defined step size and \( d_{k-1} \) is the principal direction of \( l_{k-1} \). Because \( l_k \) probably doesn’t exactly belong to the point set surface, we must optimize it. We collect the neighbor points of \( l_k \) within radius \( r \) and compute the neighbor points’ barycenter as \( p_c \). By adjusting the affected principal directions of the neighbor points with the same sign as the direction vector of \( l_{k-1} \), we can compute the principal direction of \( p_c \) as the normalized vector \( T_c \). So, we set \( l_k \) as \( p_c \) and set its principal direction as \( T_c \).

Second, once we’ve obtained \( l_k \), we label its neighboring points within radius \( s \) of \( l_{k-1} \) and remove them from the priority queue. For \( l_0 \), we label and remove only the points in the current tracing direction’s hemisphere (see Figure 7).

Third, the current curve stops growing if the current sample point has no neighbor point within radius \( r \) or if all its neighbors have been labeled.

The parameters \( s \) and \( r \) depend on the point cloud’s average distance, \( \kappa \). Our experiments indicated that \( s = 1.5 \ast \kappa \) and \( r = 0.8 \ast \kappa \) can lead to better results.

By repeating the three steps until the priority queue is empty, we obtain the ridges’ curves. We can use the same scheme to extract the valley’s curves. After that, the tracing procedure visits all the endpoints of valley or ridge lines to search for the gaps between multiple polylines. Then, we complete the original sampled polylines by connecting each endpoint to another sample point, which is in a cone defined by the tangent vector \( T \) of the endpoint and a predefined aperture angle \( \mu \). Finally, we remove the perturbations on the valley-ridge lines by fixing each polyline’s endpoints and applying Laplacian smoothing on the internal vertices.

This entire process produces high-quality valley-ridge lines (see Figure 8).

**Testing the Approach**

For our experiments, we captured three 3D fingerprint models from different subjects: Data-1, Data-2, and Data-3. We implemented our experiments with Visual Studio C++ 2008 on a PC with a 2.7-GHz CPU and 2 Gbytes of memory. Table 1 shows the total execution time and maximum memory cost for our algorithm. Because the algorithm and
implemented code still lack optimization, the algorithm needed seconds to complete the computation.

Figure 9 shows the results for extracting valley-ridge lines from the Data-2 3D fingerprint model. Compared with Data-1 (see Figure 2a), the depth information between the valley and ridge features is relatively shallow, especially near the borders of Figure 9a. This makes 3D feature detection difficult. This is similar to traditional 2D image-based fingerprinting approaches, in which capturing such shallow valley-ridge features is difficult owing to uneven finger pressure and distortions. However, in the central areas, we can extract the exact valley-ridge lines from the 3D model (see Figures 9h, 9i, and 9j).

Figure 10 compares the valley-ridge features in the 3D model and the 2D images for Data-3. Figure 10a shows the reconstructed 3D model. Figures 10b and 10c present the 3D model rendered with the original fingerprint image’s color and gray levels, respectively. Some detailed valley-ridge features are difficult to recognize, especially near the image borders. However, our approach can still extract the features (see Figures 10d and 10e).

A future task could be to optimize our algorithm’s efficiency to meet real-time requirements via GPUs or some embedded device. However, it’s more important and urgent to establish an extensive 3D fingerprint database and perform fingerprint identification and recognition based on 3D valley-ridge features.

Acknowledgments
The National Natural Science Foundation of China (grant 61002040), the Introduced Innovative R&D
Team of Guangdong Province (Robot and Intelligent Information Technology), and the Shenzhen Key Laboratory of Machine Vision and Pattern Recognition partly supported this research.

References


Xufang Pang is a research assistant at the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. Her research interests include computer graphics and digital geometry processing. Pang received an MS in educational technology from Nanjing Normal University. Contact her at xpang4@city.edu.cn.

Zhan Song is an associate researcher at the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. His research interests include 3D reconstruction, human-computer interaction, and pattern recognition. Song received a PhD in mechanical and automation engineering from the Chinese University of Hong Kong. He’s a member of IEEE. He’s the corresponding author; contact him at zhan.song@siat.ac.cn.

Wuyuan Xie is a PhD student in the Department of Mechanical and Automation Engineering at the Chinese University of Hong Kong. Her research interests include photometric stereo and stereo vision. Xie received an MS in information engineering from the South China University of Technology. Contact her at wy.xie@mae.cuhk.edu.hk.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.