

APPARATUS AND METHOD FOR COMPRESSING AND DECOMPRESSING FINGERPRINT INFORMATION

BACKGROUND OF THE INVENTION

1. Field of the Invention

- 5 The present invention pertains to techniques for representing fingerprint information efficiently. More particularly, this invention relates to techniques for compression and decompression of fingerprint information in a compact and scalable manner.

2. Description of the Relevant Art

FIG. 1 illustrates a typical fingerprint image consisting of minutiae points and thinned
10 ridges. There are two types of minutia that can occur: ending minutiae (ENDING), which represents the starting or termination point of a ridge; and bifurcation minutiae (BIFUR), which represents the branching point of a ridge into two ridges. A thinned ridge is made up of a concatenation of points on a curve or a line that has a width of one pixel and contains a binary value of either "0" or "1". Two minutiae points are connected by a ridge,
15 wherein the ridge may further be connected to another ridge until it terminates at an ending minutiae or it hits the boundary of the fingerprint image. Such an interconnection of ridges via minutiae points results in a network of ridges containing bifurcation minutia and terminating at ending minutia. From a biometrics and legal point of view, the correctness in matching a certain minimum number of minutia, both in their types and positions, can
20 constitute adequate evidence for the identification or authentication of a particular individual.

In some automated fingerprint identification systems (AFIS), having only information of the fingerprint minutia may not be considered sufficient. Additional information such as the number of ridges between two minutia, and the locations of core and delta points, can
25 be equally important. Since different AFIS solutions may require different degrees and types of fingerprint information, it becomes essential to have a means to represent fingerprint information in a more generic manner. This is possible if all the information of

the minutia, the ridges, and the relationships among themselves, can be represented with a high degree of accuracy.

For efficient storage or transmission, the resulting data file size needs to be small; hence, compression of fingerprint images is required. Furthermore, since the additional information of the ridges may be optional to some AFIS, it is very useful if the compressed data is organized in a scalable or segmented fashion. This allows the generation of only one high-detail compressed data file for storage, and, at the same time, enables different systems to decode, possibly, a lower level of details according to their respective requirements or limitations. As a result, a flexible means for representing fingerprint information in a more generic efficiently compressed manner is disclosed.

SUMMARY OF THE INVENTION

Example embodiments of the present invention relates to apparatuses and methods for encoding fingerprint information in a scalable manner, and corresponding decoder apparatuses and methods for decoding a scaled version of the fingerprint information and rendering the minutia and ridges of the fingerprint, whichever are relevant. Example embodiments of the present invention aim to exploit the intrinsic characteristics of both global and local ridge directional information present in this class of fingerprint images, and focuses on a low-complexity system for compact representation of fingerprint images comprising minutiae points and binary-valued, single-pixel width ridges.

The apparatus of the encoder in one example embodiment comprises means for establishing an equivalent representation of a local ridge direction in terms of a combination of its local ridge orientation and local ridge phase, means for partitioning an image into blocks, means for identifying the minutia, means for classifying the minutia, means for generating the minutiae positional and directional maps, means for marking the blocks, means for computing the total number of minutia, means for tracing and mapping the ridges, means for adaptive entropy encoding, means for encoding the bit stream header, means for encoding the information of minutia, means for encoding the total number of minutia, means for encoding the ridges, and means for organizing the compressed data stream into scalable segments. The method of the example embodiment for encoding

2a

comprises steps for performing the apparatus of the encoder, which, in particular, include steps for tracing and mapping the local ridge directions, choosing the predicted adaptive model using dynamic queues, predicting the local ridge phase using

global region orientation, encoding a special SWITCH bit, and cycling back to encode the next ridge of interest.

The apparatus of the decoder in the example embodiment comprises means for decoding the bit stream header, means for creating and partitioning an image into blocks, means for adaptive entropy decoding, means for scalable decompression of compressed bit stream, means for decoding the minutiae type, and the minutiae positional and directional maps, means for decoding the total number of minutia, means for marking the blocks, means for decoding the ridges, and means for rendering the ridges. The method of the example embodiment for decoding comprises steps for performing the apparatus of the decoder, which, in particular, include steps for choosing the predicted adaptive model that is in synchronous with the encoder, predicting the local ridge phase based on the SWITCH bit, determining the next ridge block for decoding, and cycling back to decode the next ridge of interest.

In accordance with one aspect of the present invention, there is provided a method of analyzing an array of pixels representing a fingerprint, the method comprising the steps of: partitioning the array of pixels into a plurality of blocks; identifying one or more of the blocks which contain a minutia, with the minutia being bifurcations of ridges and/or endings of ridges; deriving, for each block containing a minutia, directions of each of the three ridges forming the bifurcation and/or a direction of the ending ridge; and creating descriptive information representing the fingerprint based on positional data from the partitioning of the array of pixels and directional data from the derived directions of the blocks containing a minutia.

Other objects and further scope of applicability of the present invention will become apparent from the detailed description given hereinafter. However, it should be understood that the detailed description and specific examples, while indicating preferred embodiments of the invention, are given by way of illustration only, since various changes and modifications within the spirit and scope of the invention will become apparent to those skilled in the art from this detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not limitation in the figures and tables of the accompanying drawings, in which like references indicate similar elements and in which:

TABLE 1 is a chart illustrating the relationships among local block directions, block orientations, and block phases.

TABLE 2 is a lookup-table for determining the global region orientation.

TABLE 3 is a lookup-table for determining the predicated block phase.

FIG. 1 is a typical fingerprint image illustrating the minutiae points, thinned ridges, and the interconnections among them.

FIG. 2 is a set of diagrams explaining how to determine if a minutiae is of type ENDING or BIFUR.

5 FIGS. 3, 4 and 5 are schematic illustrations of the possible local ridge directions, local ridge orientations, and global region orientations, respectively.

FIG. 6 is a block diagram of the encoder apparatus used to compactly represent fingerprint information in a scalable manner.

FIGS. 7, 8, 9 and 10 are flowcharts illustrating the encoding method of the invention.

10 FIG. 11 is a block diagram of the decoder apparatus of the invention.

FIGS. 12 and 13 are flowcharts illustrating the decoding method of the invention.

FIG. 14 is a schematic diagram illustrating the segments in a scalable compressed bit stream.

DETAILED DESCRIPTION OF THE EMBODIMENT

15 FIG. 1 is a typical fingerprint image consisting of binary-valued, single-pixel width ridges and minutia. There are a number of intrinsic characteristics of a fingerprint such as two ridges do not cross one another (though they may meet at a bifurcation point), minutia are well distributed from one another, etc.

To begin, the fingerprint image is first partitioned into a plurality of equal-sized blocks of
20 a predefined dimension, BLOCK_DIM. The block size can be chosen adaptively depending on the image content and the target compression rate. The choice of a larger block size can generally result in better compression performance but at the expense of possibly poorer approximation of the reconstructed image. Also, the upper bound on the block size should be carefully chosen such that each block contains only one minutiae
25 point for easy processing, but each block may contain more than one ridge passing through

it. In contrast, the choice of a smaller block size can provide better approximation of the reconstructed image but may result in lower compression performance. In general, the value of BLOCK_DIM can be a function of the capture resolution of the fingerprint image reader or sensor.

- 5 Each block may contain none or one minutiae point, and either none, one, or more than one ridge passing through it. Denote a block that contains a minutiae as a MINU_BLOCK, and a block that has at least one ridge passing through it as a RIDGE_BLOCK. For each RIDGE_BLOCK, the block ridge segment that passes through it is quantized and represented as one of a finite number of phases, orientations, and
- 10 directions. As illustrated in FIG. 3, there are 8 possible local block directions, denoted as angles subtending anti-clockwise with respect to the horizontal right direction: 0° (DIR0), 45° (DIR45), 90° (DIR90), 135° (DIR135), 180° (DIR180), 225° (DIR225), 270° (DIR270), and 315° (DIR315). There are 2 types of orientations, namely, local block orientation, and global region orientation. As shown in FIG. 4, there are 4 possible local
- 15 block orientations, defined with respect to the local block directions: 0° (QTR0), 45° (QTR1), 90° (QTR2), and 135° (QTR3). In FIG. 5, there are 4 possible global region orientations: northeast (NE), northwest (NW), southwest (SW), and southeast (SE). Similarly, there are 2 possible block phases, defined with respect to the definition of local block orientation: positive (POS) and negative (NEG).
- 20 It is to be noted that there are relationships between a block direction and a block orientation, and between a block direction and a block phase. Clearly, by comparing FIG. 3 and FIG. 4, a particular block orientation may correspond to 2 possible block directions; the distinction between these block directions is determined by the block phase. The convention used for the block phase in the invention is POS phase for each of the 4 block
- 25 orientations shown in FIG. 4, and NEG phase for the corresponding opposite directions (shown as dashed lines in FIG. 4). Each row in TABLE 1 provides a representation of a block direction in terms of a combination of the block orientation and the block phase.

Encoder

- FIG. 6 is a block diagram of the image encoder apparatus 30 that generates an output
- 30 compressed bit stream 260, given an input fingerprint image 200 (such as that shown in

FIG. 1). The input image 200 consists of a two-dimensional array of binary-valued points (pixels) constituting minutia and ridges with a width of one pixel. At the start of the entire encoding process, memories for working buffers are allocated, and all counters, queues, adaptive models, and buffers are initialized. The means 202 first partitions the image 200 into a two-dimensional array of equal-sized blocks with predefined dimensions. The output 204 provides the input to means 206 for generating the positional map of minutia. A list 208 of the coordinates and type of minutia is produced for future processing. Each point of the input 204 is scanned and checked to determine if it is a minutiae, which is in turn to be further classified ENDING or BIFUR, depending on the number of crossing points that the minutiae has within its 8-neighbour points. FIG. 2 (a) and (b) illustrate the methods for determining an ENDING and a BIFUR minutiae, respectively. The gray squares represent points on a ridge, and the white squares constitute the background (i.e. the non-ridge or non-minutiae points) of the fingerprint image. A crossing point is defined as a transition from a gray square to a white square, or a transition from a white square to a gray square. Consider the gray block marked A and its 8-neighbour points, marked 1, 2,...8, of FIG. 2 (a). Point A is classified as ENDING since it has a total of 2 crossing points — one from point 1 to point 2, and another from point 2 to point 3. Using a similar method, the gray block marked B of FIG. 2 (b) is classified as BIFUR since it has a total of 6 crossing points, namely, from points 1 to 2, 2 to 3, 3 to 4, 5 to 6, 6 to 7, and 8 to 1. A point on a ridge that has a total number of crossing points other than 2 or 6 is classified as a non-minutiae point.

Both the output 204 of means 202 and the output 208 of means 206 provide the inputs to means 210 for marking the blocks that contain minutia. Each predefined block that contains a plurality of points is checked to determine if it is a MINU_BLOCK. If so, the block is marked, and the block type 212 is used to facilitate future processing. The output 208 of means 206 and the input image 200 provide the inputs to means 214 for classifying each minutiae as either a true minutiae (TRUE) or a guiding minutiae (GUIDING), depending on its position in relation to the entire image 200. A TRUE minutiae has significant importance and is actually used for the purpose of matching or identification in an AFIS. On the other hand, a GUIDING minutiae is located at the boundary of the image, and merely serves as a starting or termination point of a ridge. Unlike TRUE minutia, GUIDING minutia are not used for the purpose of matching, as they are almost always not genuine minutia of a fingerprint.

The following steps describe how the minutiae are classified as TRUE or GUIDING. Since there are 4 boundaries in an image, consider the processing of one boundary at a time. When processing the top boundary, each column (vertical array) of points on the image is scanned from top to bottom, and the first point that is found to be a minutiae is marked as

5 GUIDING. It is to be noted that a GUIDING minutiae must be of type ENDING. Otherwise, if a ridge point that is not an ENDING is encountered first during the scanning process, the current column of points is skipped (that is, no GUIDING is found), and the process continues with the next unprocessed column of the image. Using a similar method when processing the left boundary, each row (horizontal array) of points on the image is

10 scanned from left to right, and the first ENDING that is found to be a minutiae is marked as GUIDING. Otherwise, if a ridge point that is not an ENDING is encountered first during the scanning process, the current row of points is skipped and the process continues with the next unprocessed row of the image. Similarly, when processing the bottom boundary (or the right boundary), each column (or row) of points on the image is scanned

15 from bottom to top (or right to left), and the same method for marking the GUIDING minutiae is employed. It is important to note that in identifying the GUIDING minutiae, only the boundary blocks of the image are processed since these minutiae are located at the boundary of the image. The result of the classification 216 is used to guide future processing.

20 The output 208 of means 206 and the input image 200 provide the inputs to means 218 for generating a directional map 220 of minutiae. The direction (or directions) of each minutiae with respect to the ridge (or ridges) with which it is connected to is computed. An ENDING has 1 direction; a BIFUR has 3 directions, each along one of the 3 ridges that meet at that minutiae. The following steps describe how a minutiae direction with respect

25 to a particular ridge is computed. Starting from a minutiae point, the particular ridge of interest is traced one point at a time until a certain predefined number of points, NUM_POINTS, on the ridge have been traversed. This stopping point and the starting (minutiae) point now constitute two points between which a straight line can be formed. The minutiae direction is defined as the angle that subtends between the straight line

30 connecting the two points and the horizontal line extending to the right side from the minutiae point. The computed angle is then quantized to another predefined number of discrete minutiae directions, MINU_DIR. The directional map 220 is then used in future processing.

The output 208 of means 206 also provides the input to means 222 for computing the total numbers of TRUE (output 224) and GUIDING (output 226) minutia. The output 204 of means 202, the output 208 of means 206, the output 212 of means 210, the output 216 of means 214, and the output 220 of means 218, provide inputs to means 228 for tracing and
5 mapping of all ridges of the input image 200. This step traces all the ridges, one at a time, starting from a minutiae and terminating at another minutiae. During tracing, the block ridge directions along a ridge are mapped and stored for encoding later (shown in detail in FIG. 7).

Means 238 for encoding the bit stream header accepts as inputs the total number of TRUE
10 minutia 224, the total number of GUIDING minutia 226, the fingerprint compression tag 232, the height and width of the image 234, and some other optional parameters 236 of the image. The options may include the value of BLOCK_DIM (if it is not fixed), the fingerprint classification index, and the core and delta points of the fingerprint. The bit stream header constitutes the first portion of the compressed bit stream 260 representative
15 of the input image 200.

The means 240 accepts as inputs the directional map 220 of means 218, the classification result 216 of means 214, and the positional map with minutiae type information 208 of means 206, for encoding the information of all TRUE minutia. For each TRUE minutiae, the horizontal and vertical coordinates, the minutiae type, and the direction (or 3 directions
20 for a BIFUR) are encoded. The output 242 of means 240 is entropy encoded by means 258 to generate another portion of the compressed bit stream 260. Means 244 also accepts the same three inputs as means 240 but it encodes both the positional and directional information of all GUIDING minutia. It is to be noted that the minutiae type information is not required, as all GUIDING minutia must be of type ENDING. The output 246 of
25 means 244 is then entropy encoded by means 258 as another portion of the compressed bit stream 260.

The output 230 of means 228 provides the input to means 248 for encoding the number of block ridge directions of each MINU_BLOCK. For easy processing, it is useful that each MINU_BLOCK contains only one minutiae but it may contain more than one block ridge
30 direction. However, it is to be noted that the number of block ridge directions of each MINU_BLOCK represents the actual number of ridges that originate from the minutiae

during the tracing and mapping process by means 228. In other words, the actual number to be encoded is the number of ridges of which the minutiae acts as a starting minutiae, and not as a terminating minutiae. The output 250 of means 248 is then entropy encoded by means 258 as another portion of the compressed bit stream 260.

- 5 At this stage, all the information pertaining to the minutia has been encoded. Means 254 accepts as inputs the output 230 of means 228, the output 252 of means 248, the output 208 of means 206, the output 212 of means 210, and the output 220 of means 218 for encoding all the ridge information of the input image 200. Each block that contains a block ridge direction, as determined by means 228, is encoded. The output 256 of means
10 254 is further entropy encoded by means 258 to improve the compression performance. Details of means 254 are described with the help of a flowchart in FIG. 8.

- FIG. 7 details the steps in means 228 of the encoder 30. All the ridges are traced and mapped into corresponding block ridge directions for encoding later. At the start of the process, each minutiae is initialized as untraced. The process starts with the first untraced
15 minutiae, and continues tracing and mapping the ridges until all minutia are found in step 500 to have been traced before the process ends at step 534. The image 204 of means 202 is scanned in a raster manner, and the next untraced minutiae is identified in step 502. The 8-neighbour points centered at the minutiae of interest are checked for any untraced ridges in step 504. If the answer is NO, the process cycles back via 528 to step 500 and processes
20 the next untraced minutiae, if any. Otherwise, if the answer is YES at step 504, the next untraced neighbor on the ridge of interest is identified in step 506, and is marked as traced in step 508 to avoid redundant processing. The ridge point of interest is checked to determine if it is a terminating minutiae in a MINU_BLOCK in step 510.

- If the answer is NO, the point of interest is checked if it is the first point of the ridge of
25 interest in step 514. At the beginning of tracing a ridge, the 8 counters are reset to zero in step 516. The process then continues in step 518 to define the first region of interest and its boundaries. In order to determine the current region of interest, the current block of interest is first defined. At the beginning of tracing a ridge, the current block of interest is defined as the block that contains the starting minutiae of the ridge of interest. The 8-
30 neighbor blocks centered at the current block of interest then define the current region of interest, which boundaries are defined as the perimeter enclosing the region. The current

boundaries are stored and used for mapping the ridge of interest into corresponding block ridge directions. The 8 counters that were reset in step 516 represent the corresponding 8-neighbor blocks within the current region of interest. The number of ridge points that lie within each of the 8-neighbor blocks are kept tracked by incrementing the corresponding
 5 counters in step 520.

The step 522 checks if the current ridge point of interest lies on a region boundary. If the answer is NO, the process cycles back via 532 to step 506, and continues to trace the next unprocessed neighbor point along the same ridge of interest. If the answer is YES at step 522, the ridge has reached the boundary of the current region of interest. The block ridge
 10 direction for the current block of interest is then determined. To do so, the counter that has the maximum count is identified in step 524. The block that corresponds to this counter will become the next block of interest in step 512 to continue the tracing process of the current ridge of interest. The block ridge direction for the current block of interest is defined as one of the 8 possible local block directions that points from the current block of
 15 interest to the next block of interest. The next block of interest will become the current block of interest when processing the next ridge point of interest. The next region of interest and its boundaries are defined in step 526 using the same method as described in step 518. The 8 counters are reset to zero in step 516 before the process cycles back via 532 to step 506, and continues processing the next ridge point of interest.

20 If the answer is YES at step 510, the current ridge of interest has been completely traced and mapped. The block ridge direction of the last block on the ridge of interest is then determined in step 512. The process cycles back via 530 to step 504 to process the next untraced ridge connected to the minutiae, if any. The process of tracing each ridge and mapping the ridge of interest into corresponding block ridge directions will continue until
 25 all minutia and ridges are processed. The list of block ridge directions representative of the ridges will then constitute the output 230 of means 228.

FIG. 8 details the steps in means 254 of the encoder 30. All the block ridge directions that were mapped by means 228 are to be encoded. The process of means 254 starts with the first MINU_BLOCK in step 800, and continues until all the block ridge directions are
 30 encoded before it ends at step 840. In step 802, each of the total number of block ridge directions originating from the current MINU_BLOCK of interest (as determined by

means 248) is processed one at a time. Upon completion, the process cycles back via 838 to step 800 to process the next MINU_BLOCK, if any.

For each of the new block ridge directions identified in step 802, the steps 832 will encode implicitly all the pertinent block ridge directions corresponding to one entire ridge by means of encoding the block orientations and block phases representative of the block ridge directions. In order to encode the current block ridge direction of interest, a prediction of the current block orientation is first made in step 804 based on previously encoded blocks that are stored in a buffer while processing step 810.

The details of the prediction are shown in FIG. 9. The result of the prediction decides the choice of the adaptive model used in entropy encoding by means 258. In step 806, the current block ridge direction that is to be encoded is retrieved from the input 230. The current block orientation can then be determined uniquely in step 808 using a simple table-lookup for the conversion, as given in TABLE 1. The step 810 stores the current block orientation in a dynamic buffer that is used for the prediction in step 804.

The steps 812 perform the actual encoding, and the output 256 is entropy encoded with the adaptive model that was predicted for the current block in step 804. A SWITCH bit may or may not be encoded in step 814; it is encoded only if the predicted block phase made while encoding the previous block ridge direction was incorrect (that is, the SWITCH bit was set in step 824). It is to be noted that the SWITCH bit that is encoded while encoding the current block actually conveys additional information of the direction of the previous block, which can be the previous block on the current ridge of interest, or the last block on the previous ridge of interest. The step 816 encodes the current block orientation, as determined in step 808, with the same predicted adaptive model for the current block of interest.

In order to convey the complete information of the current block ridge direction, the current block phase has to be transmitted in addition to the current block orientation. Step 820 performs a simple table-lookup using TABLE 1 for the conversion from block direction to block phase. For efficient encoding, the current block phase is not directly encoded as a good prediction of it can be obtained from information of previously encoded blocks within a neighboring region. The step 822 makes a prediction of the current block

phase (shown in detail in FIG. 10). If the prediction was incorrect, additional information has to be encoded so that the decoder can recover the actual current block ridge direction accurately; the SWITCH bit is set in step 824. Otherwise, no extra overhead is needed as the block orientation can provide sufficient information for the actual block ridge direction. Before completing the processing for the current ridge block, information of the current block is stored in step 826 for later use in the prediction in step 822. The next block for processing can be determined in step 828 from the current block ridge direction. The process checks in step 830 if the next ridge block is a MINU_BLOCK that contains a terminating minutiae, thus signifying the end of encoding the current ridge of interest. If the answer is NO, the process cycles back via 836 to step 804 and continues encoding the next RIDGE_BLOCK on the same current ridge of interest. If the answer is YES at step 830, the process cycles back via 834 to step 802 and encodes the next ridge. The whole process repeats until all the block ridge directions representative of all the ridges of the input image 200 are encoded.

In FIG. 9, the method for predicting the current block orientation in step 804, or more specifically, for determining the choice of adaptive model for entropy encoding of the current block information, is described. At the start, the 4 counters representing the 4 possible predicted block orientations (or, equivalently, 4 possible adaptive models) are initialized to zero in step 1000. In order to exploit the strong correlation of ridge information within a neighborhood, a block region with dimension PRED_DIM and centered at the current block of interest is defined in step 1002. Each block within the defined block region is checked in step 1004. As a backward adaptive prediction, only information that has been conveyed to the decoder up to the time at which the current prediction is made will be considered. The step 1006 ensures that only causal blocks (that is, RIDGE_BLOCK that have been encoded earlier) are used in the prediction. Also, the step 1008 ensures that only valid blocks within the image are considered. If any of the answers is NO at step 1006 or 1008, the process cycles back via 1014 to step 1004 to evaluate the next block within the defined block region, if any. For each valid causal block within the defined block region, the block orientation that was stored in the buffer at step 810 is retrieved in step 1010. The appropriate counter corresponding to the stored block orientation is incremented in step 1012. The prediction process repeats until all valid block within the defined block region have been evaluated. The counter that has the maximum count is determined in step 1016, and is recorded as the predicted block

orientation in step 1018. In step 1020, the prediction is completed and the adaptive model used for encoding the current block of interest is determined. In the implementation, the value of PRED_DIM is set to 7 as a good practical trade-off between speed and reliability of the prediction.

5 In FIG. 10, the method for predicting the current block phase in step 822 is described. The prediction process exploits the fact that the dominant global ridge orientation within a local region causal to the current block of interest is likely to have a smooth (stable) transition in block phase information. The 4 possible global region orientations are illustrated in FIG. 5. In order to keep track of local block ridge orientations, a cyclic queue of size
10 QUEUE_DIM is used. The steps in 1110 record the local behavior of the current ridge of interest by means of a cyclic queue. In step 1100, the current index of the queue is first obtained. The step 1102 then records the block ridge direction of the previous block (that is, the block that was processed just prior to the current block of interest) into the current queue position. The queue index is then incremented later in step 1104. If the
15 incremented queue index exceeds the queue size in step 1106, the queue index is reset to zero in step 1108 to simulate a cyclic queue. With this process, the local block ridge directions can be recorded and analyzed later. The step 1112 computes the 2 out of 8 possible block directions that have the highest probabilities of occurrence using information in the queue. In step 1114, the dominant global region orientation is estimated
20 by means of a fast table-lookup using information of the top two block directions. TABLE 2 summarizes the lookup results, with the first column and first row representing the block directions with the first and second most probabilities of occurrence, respectively. In step 1116, both the estimated dominant global region orientation, and the actual current block orientation that was determined in step 808, are used to obtain the predicted block phase
25 for the current block of interest. Another fast table-lookup is used, as summarized in TABLE 3. The predicted block phase is then stored in step 1118, before the process ends in step 1120.

Decoder

30 The decoder 80, which operates in the converse manner of the encoder 30 of FIG. 6, is described below. FIG. 11 is a block diagram of the image decoder apparatus 80 that decodes and reconstructs an approximated version 1366 of the original image 200 (that is,

the input image to the encoder 30), given an input compressed bit stream 260. As pointed out earlier, the compressed bit stream 260 is scalable, or it comprises a segmented stream. Therefore, the decoder 80 can have the flexibility to choose and to decode only a suitable subset of the input compressed bit stream 260, and recover only the pertinent information of the fingerprint that is needed for a particular AFIS solution. As an example, the ridge information is usually not needed for the purpose of identification in most AFIS; hence, only the segments of the bit stream that correspond to the positional and directional information of TRUE minutia is decoded.

In step 1300, the bit stream header is first decoded to retrieve information of fingerprint compression tag 1314, the total number of TRUE minutia 1318, the total number of GUIDING minutia 1320, the height and width of the image 1316, and some other optional information 1312 of the image. The output 1316 of means 1300 provides the dimensions to means 1344 for creating a null image, which is then partitioned into blocks of dimension BLOCK_DIM. Before proceeding further with the entire decoding process, memories for the working buffers are allocated, and all counters, queues, adaptive models, and buffers are initialized. It is to be noted that, from the way the various portions of the compressed bit stream are organized during the encoding process, information of GUIDING minutia can only be recovered after restoring information of TRUE minutia.

The means 1332 recovers the information of all TRUE minutia by decoding the appropriate portion of the compressed bit stream 260 using the adaptive entropy decoder 1322. The decoded data 1324 as well as the total number of TRUE minutia information 1318 will be the inputs to means 1332 for restoring the positional map with minutiae type information 1334, and the directional map 1336 of all TRUE minutia. Similarly, the means 1338 will entropy decode the appropriate portion of the compressed bit stream, and the data 1320 and 1326 will be the inputs to means 1338 for recovering the positional map 1340, and the directional map 1342 of all GUIDING minutia. It is prior knowledge to the decoder that all GUIDING minutia are of type ENDING.

Using the positional information 1334 and 1340 of TRUE and GUIDING minutia, respectively, as well as the partitioned image 1346 as inputs to means 1348, the corresponding MINU_BLOCK of the image are marked to produce the output 1350 for future processing. The data 1334 and 1340 will be the inputs to means 1352 for recovering

the number of block ridge directions of each TRUE or GUIDING minutiae. This is achieved by means of entropy decoding the appropriate portion of the compressed bit stream to obtain the input 1328.

The output 1354 of means 1352, the output 1350 of means 1348, and the appropriate
5 entropy decoded data 1330, will be the inputs to means 1356 for generating all the block
ridge directions of the fingerprint image (shown in detail in FIG. 12). The output 1358,
which consists of information of the minutia and the reconstructed ridges, as well as the
data 1334, 1336, 1340, 1342, and 1350, will be the inputs to means 1360 for post-
processing and smooth rendering of the ridges (shown in detail in FIG. 13). In fact, the
10 means 1364 comprises both the means 1356 and the means 1360 that operate concurrently
and synchronously via 1362 for each RIDGE_BLOCK that is being processed. The final
approximated version of the original fingerprint image 200 is then recovered as the output
1366 of decoder 80.

FIG. 12 details the steps in means 1356 of the decoder 80. All the block ridge directions
15 are recovered by entropy decoding the appropriate portion of the compressed bit stream
260. The process of means 1356 starts with the first MINU_BLOCK in step 1500, and
continues until all the block ridge directions are decoded before it ends at step 1540. In
step 1502, each of the total number of block ridge directions originating from the current
MINU_BLOCK of interest (as determined by means 1352) is processed one at a time.
20 Upon completion, the process cycles back via 1538 to step 1500 to process the next
MINU_BLOCK, if any.

For each of the new block ridge directions identified in step 1502, the steps 1536 will
recover all the pertinent block ridge directions corresponding to one entire ridge by means
of decoding the block orientations and block phases representative of the block ridge
25 directions. In order to recover the current block ridge direction of interest, a prediction of
the current block orientation is first made in step 1504, based on previously decoded
blocks that are stored in a buffer while processing step 1528. In order to synchronize with
the encoder 30, the prediction in step 1504 must be an exact replica of step 804 (shown in
detail in FIG. 9) so that the adaptive model that is used in entropy decoding is the same as
30 that was used in entropy encoding the same RIDGE_BLOCK. It is to be noted that the

actual coordinates of the current RIDGE_BLOCK of interest are not known at this point of decoding; only the coordinates of the previous RIDGE_BLOCK are known.

Using the predicted adaptive model, the decoder will recover the block ridge direction of a RIDGE_BLOCK. It is to be noted that, in order for the decoder to synchronize with the encoder, the RIDGE_BLOCK of which the block ridge direction is being recovered has to
5 be delayed by one block. In other words, when the decoder is processing the current block of interest, the block ridge direction of the previous block will be recovered; only the block orientation (but not the block ridge direction) of the current block of interest is restored at the present point of decoding. While processing the current RIDGE_BLOCK of interest,
10 the step 1506 will first decode a symbol 1330 using the predicted adaptive model. If the symbol is found to be not a SWITCH bit in step 1508, the decoded symbol is set as the current block orientation in step 1512. Otherwise, if the decoded symbol is a SWITCH bit, the decoder will decode another symbol in step 1510 using the same predicted adaptive model. This new decoded symbol will be the current block orientation. The first decoded
15 symbol (that is the SWITCH bit) will provide information to toggle the block phase of the previous RIDGE_BLOCK, in which the block orientation has been recovered in step 1514. Toggling the block phase means switching a POS block phase to NEG, and vice versa – hence restoring the actual block phase of the previous RIDGE_BLOCK.

The step 1516 stores the block ridge direction of the block prior to the previous
20 RIDGE_BLOCK in a cyclic queue for the processing in step 1518, which predicts the block phase of the current block of interest. Similarly, in order for the decoder to synchronize with the encoder, the method of prediction is an exact replica of step 822. The detailed steps involved are similar to those shown in FIG. 10, except that the step 1102 should refer to recording the block ridge direction of the block prior to the previous
25 RIDGE_BLOCK.

In step 1520, the actual block ridge direction of the previous RIDGE_BLOCK can be uniquely determined from its block orientation and block phase using a simple table-lookup for the conversion, as given in TABLE 1. Step 1522 then stores the block ridge
direction of the previous block for future use when the decoder processes the next
30 RIDGE_BLOCK of interest. In step 1522, the block coordinates of the previous block are also recorded. With the actual block ridge direction of the previous block, the actual

coordinates of the current RIDGE_BLOCK of interest can be determined in step 1524. Both the coordinates of the previous and current blocks will then be used when reconstructing and rendering the block ridge segment in step 1526 (the details will be described in FIG. 13).

- 5 Before proceeding to decode the next RIDGE_BLOCK, the block orientation of the current block, which was found in step 1510 or 1512, is stored in a buffer for subsequent predictions in step 1504. The process then checks in step 1530 if the next block is a MINU_BLOCK that contains a terminating minutiae, thus signifying the end of decoding the current ridge of interest. If the answer is NO, the process cycles back via 1532 to step
- 10 1504 and continues decoding the next ridge block on the same current ridge of interest. If the answer is YES at step 1530, the process cycles back via 1534 to step 1502 and decodes the next ridge. The whole process repeats until all the block ridge directions representative of all the ridges of the input image 200 are recovered.

FIG. 13 details the steps in means 1360 of the decoder 80. Since the means 1360 operates

- 15 concurrently in synchronous with means 1356, a block ridge segment will be rendered in step 1526 of means 1356 when a new block ridge direction is decoded. However, it is to be noted that the block ridge segment being rendered is not necessarily the current block which block ridge direction is being decoded; in fact, the rendered RIDGE_BLOCK is two blocks preceding the current decoded RIDGE_BLOCK. Denote the current and previous
- 20 decoded blocks as CURR and PREV, respectively, and the current rendered block as PREV2. In particular, the steps in FIG. 13 will render one block ridge segment that begins at a point in PREV2 and ends at another point in PREV.

The rendering process begins with step 1600, which ensures that the CURR and PREV blocks (that is, the first and second blocks) on a new ridge of interest are not rendered

- 25 immediately. If the answer is YES at step 1600, the rendering process ends in step 1616. Otherwise, the process continues in step 1602 to check if PREV2 is a MINU_BLOCK. For each new ridge being rendered, the starting minutiae must produce an answer YES at step 1602. In step 1604, the actual coordinates of the minutiae in MINU_BLOCK are determined and recorded as the current reconstruction coordinates (a point in PREV2)
- 30 representative of the starting point of the block ridge segment being rendered. The step 1606 determines the target reconstruction coordinates (a point in PREV) representative of

the end point of the block ridge segment being rendered. As the current reconstruction point (for the first block of a new ridge) coincide with the minutiae point, the direction of the block ridge segment is rendered along the direction of the minutiae (decoded as either output 1336 of means 1332, or output 1342 of means 1338) in step 1608. To complete
5 rendering of the current block ridge segment, step 1610 connects the reconstructed ridge segment of interest to the target reconstruction point using a linear line segment. It is noted that higher-order of curvature approximation (e.g., second-order, or third-order curve fitting), or spline-based techniques can be employed.

In step 1612, CURR is checked if it is a MINU_BLOCK; if the answer is YES, a
10 terminating minutiae of a short ridge is reached. The actual coordinates of the minutiae are determined in step 1604, and the target reconstruction point is then connected to the minutiae point using a linear line segment in step 1610. Otherwise, if the answer is NO at step 1612, the step 1614 updates the current reconstruction point as the target reconstruction point at which the rendered ridge ends so far. The rendering process for the
15 current block ridge segment terminates at step 1616.

If the answer is NO at step 1602, the step 1612 checks if CURR is a MINU_BLOCK; if the answer is YES, a terminating minutiae of the current ridge being rendered is reached. The steps 1606 and 1610 then connect the current and target reconstruction points with a linear line segment. The coordinates of the terminating minutiae in CURR are determined
20 in step 1604, and finally step 1610 renders the final block ridge segment of the current ridge of interest before the process terminates at step 1616.

If the answer is NO at step 1612, the process is rendering a RIDGE_BLOCK in the middle of the current ridge of interest. To do so, the step 1606 first determines the target reconstruction point in PREV, and then step 1610 connects the current reconstruction point
25 to the target reconstruction point using a linear line segment. Finally, step 1614 updates the current reconstruction point as the target reconstruction point for rendering the next RIDGE_BLOCK, before the process terminates at step 1616.

Organization of Scalable Compressed Bit Stream

FIG. 14 illustrates the organization of the segmented compressed bit stream that is generated by the encoder 30. The 5 main segments (labeled A to E), which are arranged according to their order of generation, are described as follows:

- 5 • A: Compressed bit stream header, which consists of the basic information and some optional parameters of the fingerprint image;
- B: Positional, directional, and minutiae type information of all TRUE minutia;
- C: Positional and directional information of all GUIDING minutia;
- D: Information of the number of block ridge directions in each MINU_BLOCK;
- 10 and
- E: Compressed data of all block ridge directions representative of the directional and positional information of all ridges of the fingerprint image.

The above organization of the bit stream allows 2 basic levels of abstraction of the compressed bit stream. Such abstractions can be performed either at the decoder end
15 during decoding, or at the encoder end during bit stream parsing and transmission.

The first level of abstraction decodes only segments A and B of the compressed bit stream. This level of abstraction should enable the decoder to recover the positional, directional, and minutiae type information of all TRUE minutia. Choosing the first level of abstraction also indirectly allows the decoder to scale for a lower level of computational complexity.
20 As pointed out earlier, this set of fingerprint information is sufficient for unique identification or authentication of a particular individual by most AFIS solutions.

The second level of abstraction decodes all the segments A to E of the compressed bit stream. This level of abstraction allows the decoder to recover the positional, directional, and minutiae type information of all TRUE minutia, as well as to reconstruct all the ridges
25 of a fingerprint. This set of complete information of a fingerprint provides a generic

framework for the decoder to extract different types and levels of details to suit the specific requirements of a particular AFIS solution.

The invention being thus described, it will be obvious that the same may be varied in many ways. Such variations are not to be regarded as a departure from the spirit and scope of the invention, and all such modifications as would be obvious to one skilled in the art are intended to be included within the scope of the following claims.

What is Claimed:

1. A method of analysing an array of pixels representing a fingerprint, the method comprising the steps of:
 - partitioning the array of pixels into a plurality of blocks;
 - identifying one or more of the blocks which contain a minutia, with the minutia being bifurcations of ridges and/or endings of ridges;
 - deriving, for each block containing a minutia, directions of each of the three ridges forming the bifurcation and/or a direction of the ending ridge; and
 - creating descriptive information representing the fingerprint based on positional data from the partitioning of the array of pixels and directional data from the derived directions of the blocks containing a minutia.
2. The method according to claim 1, further comprising:
 - encoding the descriptive information.
3. The method according to claims 1 or 2, wherein the blocks are sized so that only one minutia resides inside a single block.
4. The method according to any one of the preceding claims, wherein the blocks which contain a minutia comprise one or more boundary blocks, each boundary block being located at one of the boundaries of the array of pixels, and the method further comprises:
 - identifying one or more of the boundary blocks containing endings of ridges;
 - deriving, for each boundary block containing an ending of a ridge, a direction of the ending ridge; and
 - wherein the descriptive information further comprises information based on positional data from the partitioning of the array of pixels and directional data from the derived directions of the boundary blocks containing an ending of a ridge.
5. The method according to any one of the preceding claims, wherein the directions comprise extension directions of the respective ridges.

6. The method according to any one of the preceding claims, wherein the blocks are all equal in size.

7. The method according to any one of the preceding claims, wherein a size of the blocks is chosen based on a desired resolution of a representation of the fingerprint by the descriptive information.

8. The method according to any one of the preceding claims, further comprising the step of deriving block ridge directions for one or more blocks not containing a minutia, the block ridge directions being representative of a direction of ridges in the respective blocks not containing a minutia.

9. The method according to claim 8, wherein the step of deriving the block ridge directions for the blocks not containing a minutia comprises:

tracing and mapping the ridges in the array of pixels; and

the descriptive information further comprises information based on the traced and mapped ridges.

10. The method according to claim 9, wherein said step of tracing and mapping entails tracing an untraced ridge from one ending of the untraced ridge to another ending of the untraced ridge.

11. The method according to claims 9 or 10, wherein the information based on the traced and mapped ridges includes at least one of a ridge direction, orientation, and phase.

12. The method according to any one of the preceding claims, further comprising the step of decoding an encoded form of the descriptive information for reconstructing a representation of the fingerprint.

13. The method according to claim 12, wherein the step of decoding the encoded form of the descriptive information comprises selecting an abstraction level relating to a desired level of resolution in a reconstruction of the fingerprint, and, based on the selected abstraction level, decoding a portion only or the entire encoded form of the descriptive information.

14. The method according to claims 12 or 13, further comprising generating a compressed bitstream representing the descriptive information in the encoded form.

15. The method according to claim 14, wherein the compressed bitstream further comprises an encoded height and width of the array of pixels.