Extracting Skeletons from Distance Maps

Sukmoon Chang,

Computer Science, Capital College, Penn State University, PA, USA

Summary
A skeleton is a useful shape descriptor that encodes both boundary and region information for a given object. A widely used approach for skeletonization is to use a distance transform. Although many skeletonization algorithms have been developed, most methods involve the computation of high order derivatives and the evaluation of complex expressions. In this paper, we propose an algorithm that rapidly constructs a coarse graph representation of skeletons without involving the computation of complex high order derivatives. Thus, our method can be used to quickly produce the skeleton prototype of a given image. We also show that smooth skeletons in continuous space can be obtained from the coarse graph using the snake model. Since our method processes a given image as a whole, the presence of multiple objects in an image is automatically detected and the skeletons of those objects are computed simultaneously.
Key words:
Skeletonization, distance map, ridge point detection.

1. Introduction

Given an object in an image, each point in the object is assigned a value equal to its distance to the nearest boundary of the object. This process of computing the Euclidean distance transform produces a distance map whose ridges projected back onto the image plane generates skeleton-like structures [3]. For a grayscale image, a distance transform cannot be performed directly, since the object boundary locations are not known. For some types of images, however, ridges of the intensity landscape tend to be at the center of anisotropic grayscale objects. Thus, ridges of a distance map or an intensity landscape are useful skeleton-like descriptors of a shape [4][6][8].

In an effort to obtain useful shape descriptors, many ridge detection algorithms have been proposed. For example, a direct simulation of Blum's grassfire analogy to skeletons was implemented using active contour models on the distance map of a shape [12]. Many skeleton-like shape descriptors have also been proposed, including cores [16], level sets [6][13], and shocks [11][19]. These methods usually involve the computation of curvature or high order derivatives. Several ridgeness measures based on curvature have recently been compared in [14][15].

Some of the methods previously discussed usually require explicit information about the critical points on the objects, for example, in [12], the positive curvature extrema on the object must be known to initialize and anchor a snake around the object, while others usually involve the computation of high order derivatives and evaluation of complex expressions [4][7][11][16].

In this paper, we propose an efficient skeletonization method that does not involve the computation of high order derivatives or the evaluation of complex expressions. This is achieved by not trying to compute exact locations of ridges in continuous space. Instead, our method rapidly constructs a coarse graph representation of skeletons, i.e., coarse in the sense that they are constructed in the discrete image space. Thus, our method can be used to quickly produce the skeleton prototypes of a given image. When needed, however, smooth skeletons in continuous space can be obtained by deforming the coarse graph on the distance map as in [12], but in a very different way, as we discuss later. Finally, the processing unit of our method is the entire image, not the individual objects contained in it. As result, the presence of multiple objects in an image is automatically detected and the skeletons of these objects computed simultaneously. Although we present it only in the context of distance maps, it should be noted that our method applies equally well to intensity profile maps of grayscale images filtered in scale space as demonstrated in the experimental results.

2. Ridge Point Detection

We first develop a gradient-based ridge detection algorithm, assuming that the distance map is continuously differentiable. However, ridges can be described formally as singularities of the distance function [21]. That is, a ridge corresponds to where the gradient discontinuity occurs on the distance map. Thus, the assumption of a continuously differentiable distance map is not acceptable for a ridge detection algorithm. Later in this section, we will extend the algorithm to drop the assumption.

The idea behind our ridge detection algorithm is based on the well-known fact that the gradient at any point on a

Manuscript received July 5, 2007
Manuscript revised July 25, 2007

distance map generally points towards the ridge and reverses its direction as it crosses the ridge [2][13][15]. In other words, for a point to be on a ridge, it must be a local maximum on some direction, i.e., on a line passing through the point bearing that direction. Consider a line with arbitrary orientation passing through a point of a distance map. If the point is a local maximum in the direction of the line, the distance values of the point's two opposite neighboring points must be less than that of the point, and the directions of the two opposite neighbors' gradient vectors projected onto the line must be opposite, pointing to the given point. In short, the given point generates a sign barrier between the two opposite neighbors on the line, if it is on a ridge of a distance map.

The problem is then how many orientations of a projection line should be examined to determine whether or not a sign barrier exists at a point on the distance map. It has been reported that examining the four orientations of 0, 45, 90, and 135 degrees identifies all the ridge points [9][17][18]. To determine the minimum number of orientations, we need to understand when a particular orientation of the line succeeds or fails to detect certain ridges.

There are two ways that a ridge interacts with a projection line: a ridge either intersects or does not intersect the line [2]. When a ridge intersects the projection line, it generates a sign barrier between the two points on the line enclosing the ridge. In other words, a ridge is guaranteed to be detected by the sign barrier on the projection line if it intersects the line. In contrast, if a ridge is nearly parallel to the projection line and does not cross it, the ridge does not produce a sign barrier on the line. Another projection line with an orientation substantially different from the orientation of the ridge (or, equivalently, from the that of the first projection line) will detect such a ridge, since the ridge parallel to the first projection line will appear perpendicular to the second line and cross it at some point, generating a sign barrier on it. For the projection lines to have sufficiently different orientations, two orthogonal lines would be the best choice [2]. All this is illustrated in Fig. 1. A simple **H**-shaped object and its distance map are shown in Fig. 1(a) and (b). Fig. 1(c) and (d) illustrate the sign barriers on projection lines parallel to the *x* and *y*-axis, respectively. Note that the horizontal ridge in the middle of the shape is not present in Fig. 1(c), since the ridge is parallel to the projection line and does not cross it. As expected, however, the horizontal ridge is detected on the other orthogonal projection line, as can be seen in Fig. 1(d). Similarly, two vertical ridges of the shape are not detected in Fig. 1(d), but are present in Fig. 1(c). Together, the two projection lines reveal all the ridges present in the shape.

The discussion above indicates that examining two orthogonal projection lines for a sign barrier at a point on a distance map suffices to determine whether the point is located on a ridge, suggesting a simple scanline algorithm for ridge point detection (hereafter, we will refer to a *projection line* as a *scanline*).

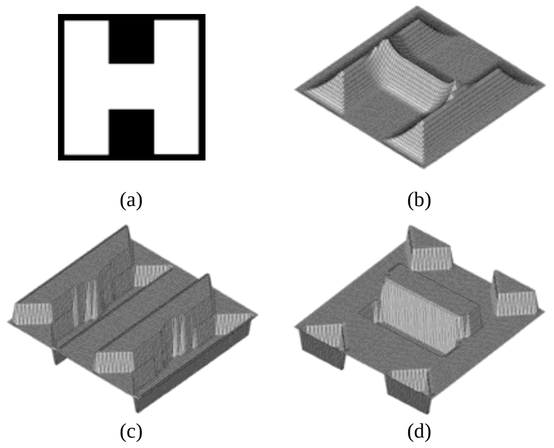


Fig. 1. Distance map and sign barriers: (a) Simple **H** shaped object, (b) Distance map of the shape, (c) and (d) Sign barriers on the scanlines parallel to *x* and *y*-axis, respectively

If we travel left to right (or top to bottom) along a scanline on a distance map, we will encounter various regions. For example, when we reach a feature point, probably after moving across some background points, we will encounter an uphill region and the projected gradient vector will change its direction from direction-less (i.e., zero vector over the background) to a positive direction (i.e., uphill). If we keep moving along the line and cross a ridge, the projected gradient vector will reverse its direction from positive to negative, and so on. Some of these sign changes we encounter on the scanline are indications of ridge existence while others are not. The complexity of sign change patterns on the line, of course, will vary depending on the topology of the distance map. However, there are only a small number of basic patterns of which the complex patterns on scanlines are composed. To identify the sign changes that indicate the existence of a ridge, we start with simple cases, i.e., the sign change pattern between two neighboring points on a scanline. There are only six possible patterns: +–, –+, +o, o+, o–, and –o, where + represents a vector with positive direction on a scanline, – with negative direction, and o a zero vector. In addition, [s…] below indicates one or more occurrences of sign s.

Pattern +− is the most obvious indication of ridge existence. As discussed before, when a ridge intersects a scanline, it produces a sign barrier between two neighboring points that enclose the ridge, generating +−. If the ridge crosses the scanline at an integer coordinate point, the pattern should be extended to 3 neighboring points, generating the pattern +o−. Therefore, patterns +− and +o− are *strong* indications of ridge existence.

On the other hand, pattern −+ is the most obvious indication of a valley. It can only occur in one of the two ways. If two ridges exist in such a way that they enclose two neighboring points, forming a valley between them, pattern −+ occurs at the given points. Since we are only interested in ridges, this case is ignored. Another way this pattern can occur is when two tapering subshapes (e.g., shapes such as *wedges, flares,* or *cups* defined in [1]) meet at their closing ends forming a concave boundary in such a way that the ridge of the shape is roughly aligned with the given scanline. In this case, however, the ridge will appear as the pattern +− on the other orthogonal scanline and thus, can also be ignored here, since it is handled by the other scanline. Therefore, in both cases, the pattern −+ is not an indication of ridge existence.

Pattern +o can also occur in two places: at the aliased edge of a shape due to rasterization and at the beginning of a plateau on a given distance map. The pattern occurring at an aliased edge should not be considered as an indication of ridge existence, since, if we do, a spurious skeleton branch would be triggered for each jag on the aliased edge. On the other hand, this pattern occurring on a real plateau must be considered as an indication of a ridge. Note that, when it occurs on a real plateau, it must always occur in pairs with such patterns as +o, o−, +−, or +o− on the other scanline. Therefore, in both cases, pattern +o is considered as a *weak* indication of ridge existence. However, when it is coupled with other patterns such as +o, o−, +−, or +o− on the other scanline, this pattern is promoted to being a *good* indication of ridge existence. Pattern o− is similar to pattern +o and treated the same.

If pattern o+ is the first sign change detected on a scanline, this pattern simply indicates the leftmost or topmost edge of a shape. Thus, for pattern o+ to indicate ridge existence in any way, it should not be the first sign change on a scanline, and must be preceded by others. Let us extend the pattern by including the sign change immediately preceding this pattern. There are only two possible extensions: −[o…]+ and −[o…]+. Pattern −[o…]+ suggests the existence of a valley or a basin for the same reasons that hold for pattern −+ and is not an indication of ridge existence. On the contrary, pattern −[o…]+ implies that we are at the end of an uphill, pass through a plateau, and then encounter another uphill on a distance map. In other words, the + at the end indicates the start of a new shape, while the + at the beginning indicates that we are already inside of another shape. Therefore, o+ at the end of this pattern is merely an indication of the start of a new subshape in a composite shape. The interpretation of +o at the beginning has previously been discussed. Since the extended patterns are not indications of ridge existence or can be handled by other cases, pattern o+ is ignored here.

Finally, the pattern −o can be explained in an opposing way. If it is the last sign change detected on a scanline, pattern −o simply indicates the rightmost or bottommost edge of a shape. Thus, for this pattern to indicate ridge existence in any way, it should not be the last sign change on a scanline, and must be followed by other sign changes. Extending the pattern to include the sign change immediately following it also yields only two cases: −[o…]+ and −[o…]−. Pattern −[o…]+ is not an indication of ridge existence as already discussed before. Pattern −[o…]− implies that we are at the end of a downhill, pass through a basin, and then encounter another downhill on a distance map. In other words, the − at the beginning indicates the end of a shape, while the − at the end indicates that we are still inside of another shape. Therefore, −o at the beginning of this pattern is merely an indication of the end of a subshape in a composite shape. The interpretation of o− at the end has previously been discussed. Since the extended patterns are not indications of ridge existence or can be handled by other cases, pattern −o is ignored here.

By evaluating all the patterns of sign changes between two neighboring points and their extensions, we have identified four patterns that indicate ridge existence: +−, +o−, +o, and o−. These patterns are called *prominent sign barriers*. Note that the patterns +o and o− must be paired with one of these four patterns on the other scanline to be considered an indication of a ridge. Having identified prominent sign barriers, we now describe the ridge detection algorithm. Before the algorithm is applied, the given image is distance transformed. Then, two normalized vector fields, N_x and N_y, are computed by projecting gradient vectors onto two scanlines, S_x and S_y, respectively, where S_x is a line parallel to the *x*-axis and S_y to the *y*-axis. The ridge detection algorithm operates directly on the vector fields, N_x and N_y.

The algorithm scans the given image from top to bottom and from left to right, searching for the prominent sign barriers. More precisely, it scans N_x with S_x from top to bottom and N_y with S_y from left to right. For each scan, it searches for prominent sign barriers from left to right for S_x and from top to bottom for S_y. When it detects prominent sign barriers at points on a scanline, the algorithm labels the points appropriately based on the prominency, i.e., *strong*, *good*, *weak*, and *none*. After the scanning process is done in both directions, each point in the image has an appropriate label and the algorithm returns these labels for further processing, i.e., building a graph representation.

The algorithm as described so far, however, still has the problem of its assumption of a differentiable distance map. This assumption was necessary since our algorithm was based on the gradient vectors of the distance map and, to compute them, the distance map must be differentiable everywhere, including ridges. However, this contradicts the description of ridges as singularities of the distance function. We can rid our algorithm of this contradictory assumption with a simple modification. Recall that the algorithm operates on two vector fields, N_x and N_y. More precisely, the algorithm examines only the directions or signs of vectors in N_x and N_y, to identify the prominent sign barriers. These signs can be interpreted in a slightly different context: a point with $+$ means that the point is at an uphill on the distance map, a point with $-$ at a downhill, and a point with 0 at a flat region or a plateau. Then, all the algorithm needs to know is where on the distance map a point is located, i.e., uphill, downhill, or plateau. Therefore, instead of computing gradients under the assumption of the differentiable distance map, we can simply use the relative location of points on a scanline without the assumption. That is, instead of using N_x and N_y computed from gradient vectors, we can use:

$$N'_x(x,y) = \operatorname{sign}\big(D(x+1,y) - D(x,y)\big)$$
$$N'_y(x,y) = \operatorname{sign}\big(D(x,y+1) - D(x,y)\big)$$

where, D is a given distance function, to compute the relative location of a point on a scanline. See Fig. 2(a) for the result of the algorithm applied to a stylized shape image.

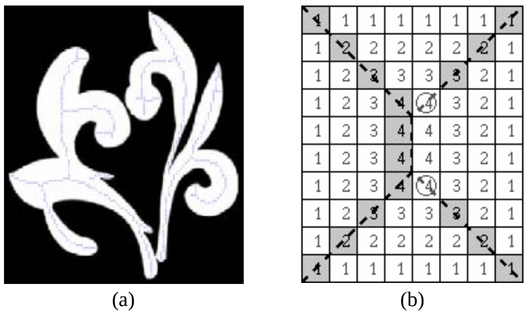


Fig. 2. Ridge points and gaps: (a) Ridge points detected by the algorithm. (b) Distance map of 8 by 10 rectangle. Dashed lines are the actual skeletons gray boxes present the pixels detected. Each circled pixel forms a sign barrier along with the pixels at its left. Our algorithm labels these pixels as week ridge points.

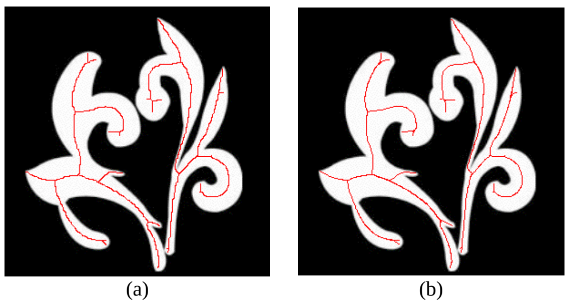


Fig. 3. Linking and deformation: (a) Result of linking process applied to Fig. 2(a). (b) Result after 50 iterations of deformation steps.

3. Skeletons with Graph Representation

Our algorithm guarantees the detection of all ridge points. However, gaps may be generated due to the discrete nature of the image space. Note that the algorithm does not compute the exact location of ridge points in continuous space but the approximate location in discrete space. For example, when a sign barrier is detected between two neighboring points, the algorithm labels the point with a higher value as a ridge point. This process may cause gaps that usually occur around forking points of skeletons located at non-integer coordinates. Fig. 2(b) illustrates a representative example. The points in the gaps, however, are usually labeled as *weak* by the algorithm and can be recovered during the linking process described below. Because of the gaps, the linking process consists of two passes. Initially, only the points with labels of *strong* and *good* are considered as ridge points.

In the first pass, each point is connected to at most 2 neighboring ridge points. The first link is made to the nearest ridge point among its 8 neighbors and the second to a ridge point among the remaining neighbors that forms an angle close to 180 degrees to the first link. This pass converts a set of ridge points into a set of ridgelines. The ridgelines are the links of the skeleton graph, but are not well connected yet because of the gaps discussed above. The second pass tries to fill the gaps by tracing a maximum gradient path. For each end point of a ridge line, either a point with label *weak* or a point with maximum gradient (if no point with label *weak* exists) among its neighbors that forms an angle close to 180 degree to its previous link is selected as the next candidate point. The process repeats from the new candidate point until the trace reaches one of the ridge points, in which case the trace is accepted, or until the trace crosses the boundary of the given object, in which case the trace is ignored. After the second pass, we have a graph representation of the skeleton, in which each object in the image is represented by a connected subgraph. Fig. 3(a) illustrates the graph representation after the linking process applied to the detected ridge points in Fig. 2(a).

4. Smooth Skeletons via Graph Deformation

The graph representation computed in the previous section is a coarse discrete approximation to the skeleton in continuous space. To obtain a continuous counterpart, the graph is regarded as a collection of deformable splines and the snake model [10] is applied to it.

For each edge in the graph, a snake is initialized on the distance map. Then, both ends of the snake are anchored to the corresponding branching or branch end points with spring forces. After initialization, each snake is deformed independently from the others on the distance map. Since they are used as anchor points, the positions of branching and branch end points are fixed on the distance map. To obtain their accurate locations, each anchor point's position is recomputed by averaging its position and the positions of the snaxels anchored to it after a predefined number of snake deformation steps. Note that a snake initialized on a steep ridge tends to move toward the bottom of the ridge where potential energy is the lowest in order to reach an equilibrium state. The steeper the ridge, the faster the snake moves to the bottom of the ridge. This causes a snake initialized on the steepest ridge to pull the anchor point towards it rapidly when averaged, causing the anchor point to pass through its optimal position. To avoid this effect, we use weighted averaging when recomputing the position of an anchor point, giving more weight to its original position. Another way to circumvent this effect is to resample the snaxels on each snake so that the intersnaxel distances are all the same over all snakes and then perform simple averaging of the involved snaxels. Fig. 3(b) illustrates the result after 50 deformation steps applied to the coarse graph in Fig. 3(a). Note how some coarse ridgelines are smoothed by the deformation.

5. Results

The results of our method to simple binary images are shown in Fig. 4. The initial coarse graphs obtained with the method were refined with 50 iterations of deformation steps to get smooth skeletons. The results show that the objects in the figure are correctly skeletonized. Also note in Fig. 4(b) that all disconnected subshapes have been automatically detected and represented by separate subgraphs.

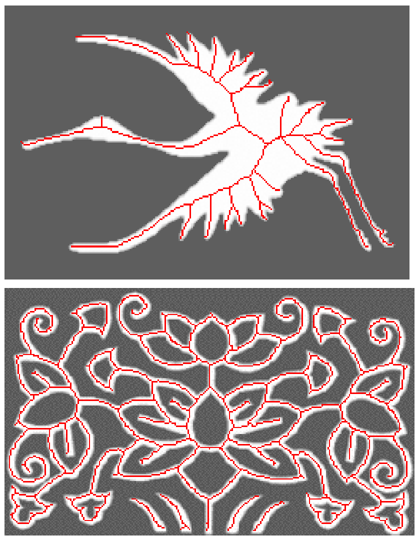
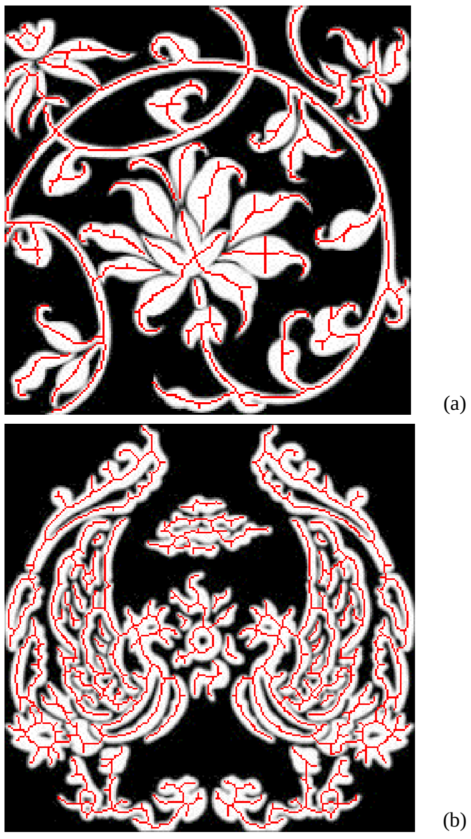


Fig. 4. Skeletons of simple binary objects.

Next, we applied our method to more complex binary images and the results are illustrated in Fig. 5. The initial coarse graphs obtained with the proposed method were smoothed with 50 iterations of deformation steps. Again, note in the figure that all disconnected subshapes have been detected and represented by separate subgraphs.

Finally, we demonstrate that our method applies equally well to the skeletonization of grayscale images. In grayscale images, however, the distance maps cannot be obtained easily since the object boundary required to perform the distance transformation is not readily available. Fortunately, for some types of grayscale images such as medical images, ridges of the intensity landscape tend to be at the center of anisotropic grayscale objects. Thus, ridges of an intensity landscape profile maps are useful skeleton-like descriptors of a shape. Therefore, we applied our method to the intensity profile maps of grayscale images filtered in scale space [5][15][20]. Fig. 6

shows the results of our method applied to grayscale images. As in the case of binary images, the initial coarse graphs obtained with the proposed method were refined with 50 iterations of deformation steps. From the figure, we can see that all the objects present in each image are automatically detected and properly skeletonized.


(a)

(b)

Fig. 5. Skeletons of complex binary objects.

6. Conclusions

We have presented a skeletonization method that rapidly constructs a coarse graph representation of skeletons. We defined and identified four patterns of prominent sign barriers by exhaustive review of all possible sign change patterns on a scanline. Ridge points are extracted from the distance map by detecting prominent sign changes on two orthogonal scanlines and connected based on the topology of the distance map. Finally, the graph is then converted to a set of snakes and deformed on the distance map. The net result is smooth connected skeletons.

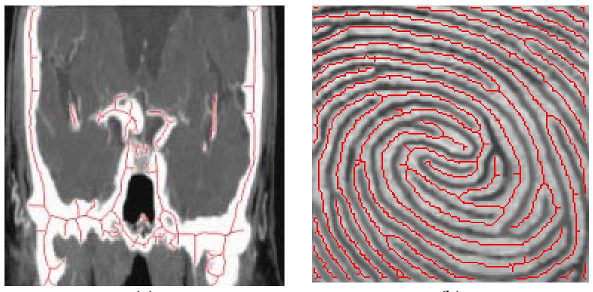

(a) (b)

Fig. 6. Skeletons of grayscale images: (a) CT image of a skull. (b) Finger print

Our algorithm differs significantly from the previous approaches in several respects. First, it does not require explicit information about object boundaries (i.e., for grayscale images) or critical points on them. Second, our algorithm operates on the entire image, not on individual objects contained in the image. That is, our algorithm can automatically detect the presence of multiple objects and computes their skeletons simultaneously, representing each object with a separate subgraph. Finally, without involving any complex computations, our algorithm rapidly constructs a coarse graph representation of skeletons, which can be further refined using a snake model. The algorithm proposed here can be used to quickly generate skeleton prototypes of a given image.

References

[1] H. Blum and R. Nagel, “Shape Description Using Weighted Symmetric Axis Features,” *Pattern Recognition*, 10: 167-180, 1978.

[2] S. Chang, D. Metaxas, and L. Axel, “Scan-Conversion Algorithm for Ridge Point Detection on Tubular Objects,” *MICCAI*, LNCS 2879:158-165, 2003.

[3] P. E. Danielsson, “Euclidean Distance Mapping,” *Computer Graphics and Image Processing*, 14:227-248, 1980.

[4] D. Eberly, R. Gardner, B. Morse, S. Pizer, and C. Scharlach, “Ridges for Image Analysis,” *Journal of Mathematical Imaging and Vision*, 4(4):353-373, 1994.

[5] L. Florack, B. ter Haar Romeny, J. Koenderink, and M. Viergever, “Scale and the Differential Structure of Images,” *Image and Vision Computing*, 10(6):376-388, 1992.

[6] J. Gauch and S. Pizer, “Multiresolution Analysis of Ridges and Valleys in Grey-Scale Images,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):635-646, 1993.

[7] J. Gauch and S. Pizer, “The Intensity Axis of Symmetry and Its Application to Image Segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(8):753-770.

[8] R. Haralick, “Ridges and Valleys on Digital Images,” CVGIP, 22:28-38, 1983.

[9] S. Ho and C. Dyer, "Shape Smoothing Using Medial Axis Properties," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(4):512-520, 1986.
[10] M. Kass, A. Witkin, and D. Terzopoulos, "Snake: Active Contour Models," *International Journal of Computer Vision*, 15:189-224, 1995.
[11] B. Kimia, R. Tannenbaum, and S. Zucker, "Shapes, Shocks, and Deformations I: The Components of Two Dimensional Shape and the Reaction-Diffusion Space," International Journal of Computer Vision, 15:189-224.
[12] F. Leymarie and M. Levine, "Simulating the Grassfire Transform Using an Active Contour Model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):56-75, 1992.
[13] A. Lopez, F. Lumbreras, and J. Serrat, "Creaseness from Level Set Extrinsic Curvature," *ECCV*, 156-169, 1998.
[14] A. Lopez, F. Lumbreras, J. Serrat, and J. Villanueva, "Evaluation of Methods for Ridge and Valley Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):327-335, 1999.
[15] J. Maintz, P. van den Elsen, and M. Viergever, "Evaluation of Ridge Seeking Operators for Multimodality Medial Image Matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):353-365, 1996.
[16] S. Pizer, D. Eberly, D. Fritsch, and B. Morse, "Zoom-Invariant Vision of Figural Shape: The Mathematics of Cores," Computer Vision and Image Understanding, 69(1):55-71, 1998.
[17] W. Seemuller, "The Extraction of Ordered Vector Drainage Networks from Elevation Data," Computer Vision, Graphics, and Image Processing, 47:45-58, 1989.
[18] F. Shih and C. Pu, "A Skeletonization Algorithm by Maxima Tracking on Euclidean Distance Transform," Pattern Recognition, 28(3):331-341, 1995.
[19] H. Tek and B. Kimia, "Symmetry Maps of Free-Form Curve Segments via Wave Propagation," ICCV, 362-269, 1999.
[20] A. Witkin, "Scale-Space Filtering," IJCAI, 367-375, 1995.
[21] M. Wright, R. Cipolla, and P. Giblin, "Skeletonization Using an Extended Euclidean Distance Transform," *Image and Vision Computing*, 13(5):367-375, 1995.

Sukmoon Chang received the Ph.D. degree in Computer Science from Rutgers University in 2002. During 2002-2004, he stayed in Computational Biomedicine and Modeling (CBIM) Center at Rutgers University as a Research Associate to study medical image analysis. He is now with Capital College, Pennsylvania State University.