IMAGE WARPING AMONG ARBITRARY PLANAR SHAPES

George Wolberg

Department of Computer Science
Columbia University
New York, NY 10027
wolberg@cs.columbia.edu

April 1988
Technical Report CUCS-329-88

ABSTRACT

Image warping refers to the 2D resampling of a source image onto a target image. Despite the variety of techniques proposed, a large class of image warping problems remains inadequately solved: mapping between two images which are delimited by arbitrary, closed, planar curves, e.g., hand-drawn curves.

This paper describes a novel algorithm to perform image warping among arbitrary planar shapes whose boundary correspondences are known. A generalized polar coordinate parameterization is introduced to facilitate an efficient mapping procedure. Images are treated as collections of interior layers, extracted via a thinning process. Mapping these layers between the source and target images generates the 2D resampling grid that defines the warping. The thinning operation extends the standard polar coordinate representation to deal with arbitrary shapes.

IMAGE WARPING AMONG ARBITRARY PLANAR SHAPES

George Wolberg

Department of Computer Science
Columbia University
New York, NY 10027
wolberg@cs.columbia.edu

April 1988
Technical Report CUCS-329-88

ABSTRACT

Image warping refers to the 2D resampling of a source image onto a target image. Despite the variety of techniques proposed, a large class of image warping problems remains inadequately solved: mapping between two images which are delimited by arbitrary, closed, planar curves, e.g., hand-drawn curves.

This paper describes a novel algorithm to perform image warping among arbitrary planar shapes whose boundary correspondences are known. A generalized polar coordinate parameterization is introduced to facilitate an efficient mapping procedure. Images are treated as collections of interior layers, extracted via a thinning process. Mapping these layers between the source and target images generates the 2D resampling grid that defines the warping. The thinning operation extends the standard polar coordinate representation to deal with arbitrary shapes.

1. INTRODUCTION

Image warping is a geometric transformation that maps a source image onto a target image. The mapping is defined by a 2D resampling grid. Image warping has proven to be an important tool in image processing and computer graphics. In image processing, warping is used to rectify distorted images, mapping nonrectangular patches onto rectangular ones. The distortions are usually attributed to viewing transformations and lens aberrations. In computer graphics, warping plays an opposite role: the target image is the 2D projection of a 3D surface onto which the source image had been mapped. Hence warping is a composite mapping of the reparameterization from 2D image (texture) space to 3D object space, and the subsequent projection onto 2D screen space (Heckbert 1986). Consequently, rectangular patches are mapped onto nonrectangular ones. This procedure is known as *texture mapping*.

The two roles that image warping plays in these fields has dichotomized the research in this area (Smith 1987). The common ground, however, is found in two areas — fast spatial transforms (Catmull 1980, Fant 1986, Fraser 1985, Oka 1987) and filtering (Burt 1981, Crow 1984, Heckbert 1986, Williams 1983).

Despite the considerable attention that image warping has received, a large class of image warping problems has been neglected: mapping between two images which are delimited by arbitrary closed curves, e.g., hand-drawn curves. In this instance, the mapping is driven by the correspondence of boundary points — information provided by the user. The spatial transformation of the interior points must be computed by the image warping algorithm. Unlike the problems treated in image processing or computer graphics, the stretching of an arbitrary shape onto another, and the associated mapping, is a problem not addressed in a tractable fashion in the literature.

The lack of attention to this class of problems can be easily explained. In image processing, there is a well-defined 2D rectilinear coordinate system. Correcting for distortions amounts to mapping the four corners of a nonrectangular patch onto the four corners of a rectangular patch. In computer graphics, a parameterization exists for the 2D image, the 3D object, and the 2D screen. Consequently, warping amounts to a change of coordinate system (2D to 3D) followed by a projection onto the 2D screen. The problems considered in this paper fail to meet the above properties. They are neither parameterized nor are they well suited for four-corner mapping.

The algorithm described in this paper treats an image as a collection of interior layers. Informally, the layers are extracted in a manner similar to peeling an onion. A radial path emanates from each boundary point, crossing interior layers until the innermost layer, the skeleton, is reached. Assuming correspondences may be established between the boundary points of the source and target images, the warping problem is reduced to mapping between radial paths in both images. Note that the layers and the radial paths actually comprise a sampling grid.

This algorithm uses a generalization of polar coordinates. The extension lies in that radial paths are not restricted to terminate at a single point. Rather, a fully connected skeleton obtained from a thinning operation may serve as terminators of radial paths directed from the boundary.

This permits the processing of arbitrary shapes.

Since this work is more closely tied to the image processing interpretation of warping, a background of warping for the rectification of images is given in section 2. The problem is formally stated in section 3. An overview of the algorithm is found in section 4. Section 5 describes the parameterization of arbitrary planar shapes, a central issue in the problem. The mapping procedure within the new parameter space is discussed in section 6. The inverse reparameterization, responsible for updating the target image with the resampled data, is described in section 7. Results are shown in section 8, enhancements are discussed in section 9, and a summary and conclusion is found in section 10.

2. BACKGROUND

Image warping has proven valuable for the geometric correction of digital images. It has seen extensive use in viewing geometry problems, sensor induced problems, and in the registration of similar images for comparison purposes. In all instances, a geometric correction map is required to define the necessary warping, or mapping, to invert the distortion. Determining this corrective transformation is a central problem in image warping.

The corrective transformation for mapping a distorted sampling grid onto the original grid is obtained from a distortion model. In the simplest of cases, it is possible to define the spatial transformation with an analytic expression. In most applications, however, the mapping is obtained from actual image measurements. For example, to derive the geometric correction map for the aberration of a particular lens, a rectangular grid pattern is used as a reference image. The distorted output grid is made to map onto the original grid by establishing a simple correspondence of grid intersections, or *control points*. Precise spatial mapping is defined at these sparse locations. The mapping of noncontrol points is determined by interpolation. Usually bilinear interpolation is adequate, although higher order polynomial functions are sometimes used. Since the resulting effects often take the appearance of elastic stretching, the geometric transformation associated with image warping is sometimes known as *rubber sheet transformation.*

The use of control grids and analytic expressions accounts for the majority of image warping techniques. Unfortunately, they fail to handle our problem of mapping an image, of arbitrary shape, onto a second arbitrary shape. This is a consequence of their inappropriateness for boundary value problems, the class with which we are dealing.

Boundary value problems have long been addressed in heat conduction, electrostatic potential, and fluid flow theory. Conformal mapping and relaxation techniques are commonly used to evaluate such mappings. In Fiume (1987), a conformal mapping is used to map images among arbitrary simple polygons. Conceptually, this approach can be applied to arbitrary shapes. In practice, however, the Schwarz-Christoffel transformation used to construct the conformal maps are analytically complicated and the implementation is limited to polygons with only a few vertices.

In a recent paper, Greene and Lamming (1986) describe a relaxation algorithm to propagate the boundary mapping information to interior points. The user specifies two polygons, and a mesh relaxation is used to derive a mapping from the first polygon onto the second polygon. Their low resolution 50×50 mesh is analogous to the control grids described earlier. Typically, the mesh requires 200 iterations to converge. Tailored for interactive use, there is no attempt made at antialiasing, thereby allowing artifacts to surface upon sampling irreproducibly high spatial frequencies. In addition, the image folds upon itself when radical transformations are made near the margin of the figure. Nevertheless, their system, designed for visual art applications, represents the closest effort to the work described in this paper.

In related approaches, relaxation or finite element analysis techniques may be utilized to propagate the boundary information while satisfying some constraint. A typical constraint may be the minimization of the pixels' "spring energies." Unlike such methods, which are constrained by physical models and require many iterations to converge, the technique proposed here is not tied to a physical model and allows mapping values to be computed at once. There is no need for iterative refinement.

3. STATEMENT OF THE PROBLEM

We desire to map a source image, S, onto a target image, T, each of whose boundaries are arbitrary. Both S and T are actually subimages which may be extracted with the use of a mouse and digitizing tablet, thresholding, or any other segmentation method. Pixels lying in the extracted subimage are designated as foreground. The remaining pixels are assigned a background value. In this manner, the boundaries may be trivially identified as the interface between foreground and background pixels.

The mapping will effectively treat S as if it were printed on a sheet of rubber, and stretch it to take the shape of T. The only constraint is that topological equivalence be maintained — that is, S may be stretched and squeezed but not torn. As a consequence, S and T must have the same number of interior holes. In this work, the images are assumed to have no interior holes. This can be rectified with a simple extension (see section 9).

The mapping must allow for the specification of correspondence points among both boundaries in order to clamp the warping effect. The mapping of noncontrol points along the boundaries is derived through interpolation of adjacent control points. Control point adjustments made by the user tailor the warping results.

4. OVERVIEW

This section illustrates the algorithm in terms of its functional units. The remainder of the paper elaborates upon the basic components listed here. The warping algorithm described herein has three stages:

1) Reparameterize S and T using a transformation function g. This yields S' and T', respectively. The new parameter space facilitates a convenient solution to our boundary value problem.
2) Apply a second transformation, h, to map (resample) S' onto T'. This stage requires filtering to avoid artifacts arising from severe compression or expansion.
3) Apply an inverse mapping, g^{-1}, to convert T' into T, the desired result.

Figure 1 depicts the homomorphism outlined above. The transformation functions g, h, and g^{-1} are elaborated in the following sections.

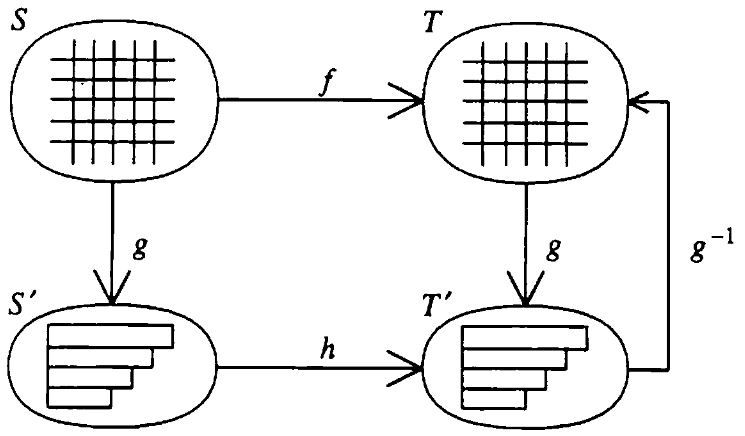


Figure 1: Chain of transformations to map S onto T.

5. REPARAMETERIZATION ONTO THE (U,V) PARAMETER SPACE

This section describes the function that transforms images *S* and *T* into a new parameter space. This reparameterization corresponds to function *g* in Fig. 1. Its purpose is to allow the mapping process to operate upon a coordinate system that is more amenable to the boundary value problem presented here.

5.1 Introduction

A function, *f*, that maps *S* onto *T* requires that a parameterization exist for both images, such that a correspondence may be established. Unfortunately, the (x,y) coordinate system imposed by the input image is not ideally suited for boundary value problems. By simply dictating spatial layout, the (x,y) coordinate system lacks the means of conveniently expressing a relation between the boundary and interior points. We seek to impose a coordinate system which more closely matches the nature of our problem.

In the absence of a suitable parametric representation for both images, the algorithm decomposes a 2D image into an alternate representation consisting of layers of interior pixels. The layers are extracted in a manner akin to peeling an onion. This imposes a (u,v) parameterization in which u runs along the boundary, and v runs radially inward across successive interior layers.

5.2 The (u,v) Coordinate System

The motivation for this formulation was derived intuitively and will therefore be expressed in a similar manner. We will begin by considering the merits of the selected coordinate system. This is followed by a discussion of its implementation.

Since the mapping information is only defined along the boundary, it is reasonable to first consider the mapping of adjacent points. These points comprise the adjacent layer. This data can then propagate further until the innermost layer is assigned a correspondence. Two problems now surface:

1) How may we derive the positions of interior layers? Note that these layers actually define a sampling grid within the shape.
2) How is the mapping information transformed from layer to layer? Clearly there are many possible solutions. However, we shall see that a convenient metric may be expressed as a result of solving the first problem.

In related approaches, relaxation or finite element analysis techniques may be utilized to propagate the boundary information while satisfying some constraint. A typical constraint may be the minimization of the pixels' ''spring energies.'' Unlike such methods, which are constrained by physical models and require many iterations to converge, the technique proposed here is not tied to a physical model and allows mapping values to be computed at once. There is no need for iterative refinement.

The algorithm poses a formulation based on the radial influence of each boundary point. This measure is defined by the path spanned between the boundary point and its corresponding point on the innermost layer. These radial paths are analogous to orthogonal grid lines (v direction) with respect to the interior layers (u direction). As we shall soon discover, the innermost layer will be referred to as the *skeleton*. Figure 2 illustrates the (u,v) grid lines superimposed upon an arbitrary shape. The skeleton is highlighted in boldface.

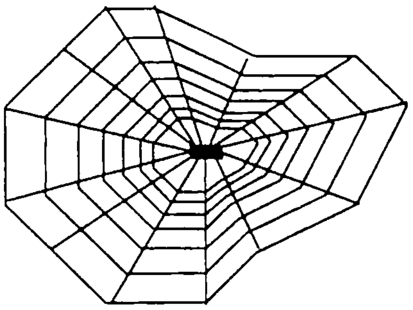
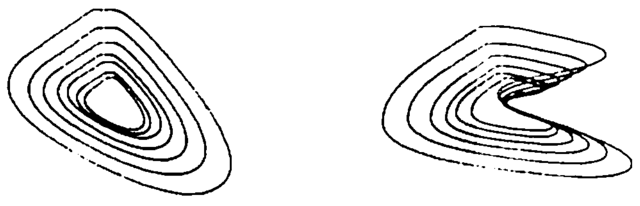


Figure 2: Arbitrary shape with superimposed sampling grid.

5.3 Extracting Interior Layers

The positions at which each layer must sample the image can be considered to be defined by an eroding boundary. In convex shapes, an eroding boundary coincides with shrinking (scaling down) the boundary positions about the centroid. Figure 3a illustrates these layers for a convex shape. Scaling cannot, however, be applied to shapes containing concavities. The problem now arises that the reduced boundary does not lie entirely within the larger adjacent boundary, and the centroid is no longer guaranteed to lie within the shape. Intuitively, the former problem is a manifestation of intervening background patches shrinking and bringing foreground pixels, lying along a concavity, closer together rather than pulling them further apart. Overlapping "interior" layers that manifest from scaling the boundary are depicted in Fig. 3b.



(a) (b)

Figure 3: Scaled boundaries for (a) convex and (b) concave shapes.

5.3.1 Thinning

The difficulty of expressing erosion analytically for shapes containing concavities is bypassed with a discrete approximation — a thinning algorithm. Thinning has long been a tool in the computer vision field for shape analysis applications. In this context, thinning, together with boundary traversal, is used to erode foreground pixels along the boundary while satisfying a necessary connectivity constraint. This helps us impose the (u,v) coordinate system upon the S and T images.

Classical thinning algorithms operate on binary raster images. They scan the image with a window (usually 3×3), labeling all foreground pixels lying along the boundary with one of two labels. The first label, *DELPXL*, is designated to deem the foreground pixel as deletable (flip to background value). This designation is issued if the foreground pixel is not found essential in preserving the mutual connectivity of neighboring foreground pixels in the window. If, however, it must be retained in order to preserve neighborhood connectivity, the pixel is said to lie on the shape's symmetric axis, or *skeleton*. Consequently, the *SKL* label is applied to all skeletal points. Skeletons typically resemble a line drawing, or stick-figure, of the image comprised of foreground pixels. They have the property of being fully connected (no gaps).

A thinning algorithm is used to assure that the shape is unraveled into closed layers. This constraint is imposed in order to avoid tearing the eroding shape, thereby mainingtain topological equivalence over all scales. That is, the deletion of boundary pixels in one layer must not introduce holes in the next layer. For example, consider a dumbbell shape. As outer layers of the shape are peeled off, the thin center bar will eventually erode away, isolating the circles at each end. No simple closed path would remain to traverse the resulting shape. Skeletons are therefore generated for the purpose of ''bridging the gaps'' between remaining boundary pixels. This translates into a guarantee that subsequent layers will remain closed. Consequently, the two ends of the dumbbell remain connected even after the center bar would have been eroded by normal means. A detailed description of the thinning algorithm used in this work is given in [Wolberg 85]. Further references may be found in [Arcelli 85], [Pavlidis 82], and [Rosenfeld 82].

5.3.2 Boundary Traversal

The boundary is traversed following each thinning pass. The preceding thinning iteration imposed a connectivity constraint on the boundary pixels and labeled them accordingly. Consequently, they are traversed while concurrently initializing the appropriate layer list and deleting those pixels labeled *DELPXL*. *SKL* pixels remain intact and are guaranteed to appear in all subsequent layers. Note that interior pixels, not having been labeled by the thinning algorithm, are not traversed in the same pass.

The list L_{Sn} or L_{Tn} is used to store the layer pixels. The subscript Sn and Tn refer to layer n in the source and target images, respectively. List C is used to store the connectivity status of each boundary pixel. Boundary traversal consists of the following cycle.

```
1.  find starting point and mark it STRT (see section 5.3.3)
2.  loop begin
3.      append the pixel's color value to list $L_{S_n}$ or $L_{T_n}$
4.      append the pixel's DELPXL/SKL label to list C
5.      IF(pixel's label is DELPXL) delete it; that is, flip it to background
6.      Move to an 8-connected neighbor labeled DELPXL, SKL, or STRT
7.      IF( pixel is marked STRT ) break loop
8.  end
```

5.3.3 Layer Alignment

Unwrapping the shape into consecutive layers presupposes that the first traversed point may be accurately mapped from layer to layer. An error in this correspondence yields misalignment problems. Therefore, the starting point for traversal is initially chosen to be the top-leftmost boundary point. This choice offers the least ambiguity for correspondence in subsequent levels. The ambiguity is diminished by the fact that only four of the eight adjacent neighbors are foreground pixels. Furthermore, since at least two of the four candidates are in direct contact with the background, they are likely to be designated as *DELPXL* by the thinning algorithm and deleted during the traversal. A better scheme consists of a preliminary search for a boundary point of high curvature.

Clearly the most reliable choice for a starting point is one which is known to be contained in all subsequent layers. Since skeletal points satisfy this property, we may choose the first encountered skeletal point to take on this role. Therefore, the top-leftmost point is used as the starting point for traversal until a skeletal point is found. All subsequent traversals will then begin from that skeletal point.

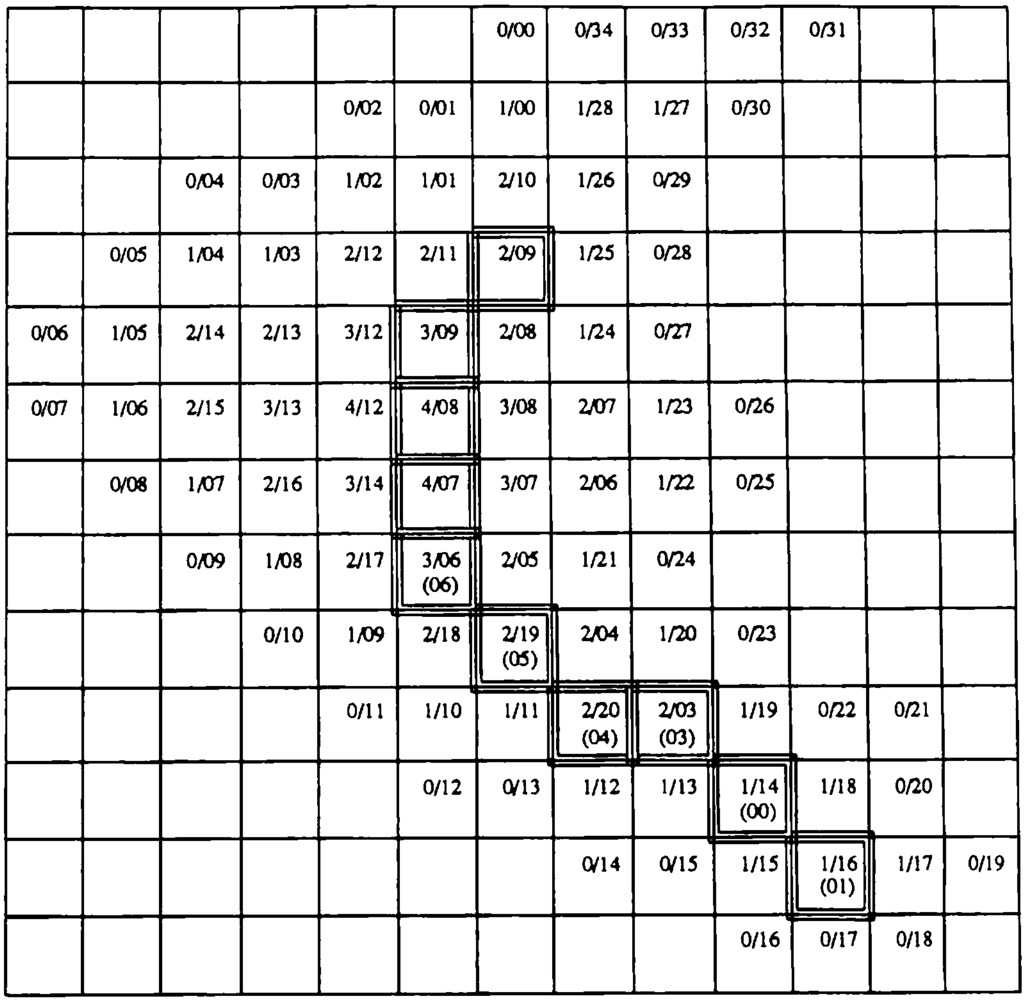
5.3.4 An Example

The discussion is now supplemented with an example. Consider the shape given in Fig. 4. All foreground pixels are labeled with a number. The format used is *layer/pos*, where *layer* denotes the layer in which that pixel is contained, and *pos* represents its position in the layer. For example, the first element in the outermost layer is pixel 0/00, the top-leftmost boundary point. The label 0/00 is interpreted as layer 0, position 00.

The shape is procesed in *p* passes, where *p* is half the maximal width. This represents the number of iterations needed to yield the entire skeleton and terminate the peeling operation. Each pass consists of the following cycle.

1) Apply one pass of a thinning process to label all boundary points as deleteable (*DELPXL*) or skeletal (*SKL*).

2) Traverse the boundary as described in section 5.3.2. This procedure serves to initialize the appropriate layer list and expose interior layers so that they may become subject to traversal in subsequent passes.



						0/00	0/34	0/33	0/32	0/31		
				0/02	0/01	1/00	1/28	1/27	0/30			
		0/04	0/03	1/02	1/01	2/10	1/26	0/29				
	0/05	1/04	1/03	2/12	2/11	2/09	1/25	0/28				
0/06	1/05	2/14	2/13	3/12	3/09	2/08	1/24	0/27				
0/07	1/06	2/15	3/13	4/12	4/08	3/08	2/07	1/23	0/26			
	0/08	1/07	2/16	3/14	4/07	3/07	2/06	1/22	0/25			
		0/09	1/08	2/17	3/06 (06)	2/05	1/21	0/24				
			0/10	1/09	2/18	2/19 (05)	2/04	1/20	0/23			
				0/11	1/10	1/11	2/20 (04)	2/03 (03)	1/19	0/22	0/21	
					0/12	0/13	1/12	1/13	1/14 (00)	1/18	0/20	
							0/14	0/15	1/15	1/16 (01)	1/17	0/19
									0/16	0/17	0/18	

Figure 4: An arbitrary shape with labeled pixels.

In Fig. 4, notice that pixels 0/00 and 1/00 lie on the top-leftmost points of the image in the first and second pass, respectively. In the second pass, however, when pixel 1/14 is traversed and is found to be labeled *SKL*, layer 1 is realigned using a circular shift. This places pixel 1/14 at the start of the list and assures proper alignment in all subsequent passes. The new role of pixel 1/14 is denoted in Fig. 4 with a (00) entry. Similarly, its neighboring skeletal points will also appear in subsequent traversals and are numbered appropriately in parentheses.

5.4. Layer Correspondence

Tracking the position of each point on the boundary as it makes its way to the skeleton is a central idea of this algorithm. By iteratively establishing correspondence across successively interior layers, we define the v grid lines in our (u,v) coordinate system. Due to the list representation that the unraveled layers have taken, the desired mapping across layers may be achieved by exploiting the 1D nature of lists.

In the absence of *SKL* points along the boundary, the 1D sampling grid of the extracted layer may be mapped onto its adjacent interior layer by uniform scaling. This means that the pixels are sampled from the adjacent layer at positions dictated by their location in the current layer and the ratio of both layers' lengths. However, the introduction of *SKL* points complicates the matter. Since they are present in all subsequent traversals, their correspondences are fixed across layers and must therefore not be imposed by scaling. Instead, *SKL* pixels partition *DELPXL* intervals into subintervals. These segments, in turn, remain subject to uniform scaling. Their corresponding subintervals in the next layer are found by identifying the same delimiting *SKL* points which originally partitioned the intervals.

An example of correspondence among layers is shown in Fig. 5. There we see the layers of Fig. 4 represented in list form. Each interior pixel is labeled with the iteration number at which it was traversed. Those pixels highlighted with a double-edged or boldface border represent skeletal points. Double-edged borders indicate skeletal pixels that first appear in the current layer. Boldface borders denote skeletal pixels which originally surfaced in a previous layer. This distinguishes them as anchor points at which correspondence is fixed. This property is used to establish the correspondence of subdivided intervals. We shall refer to such anchor points as *OLDSKL*.

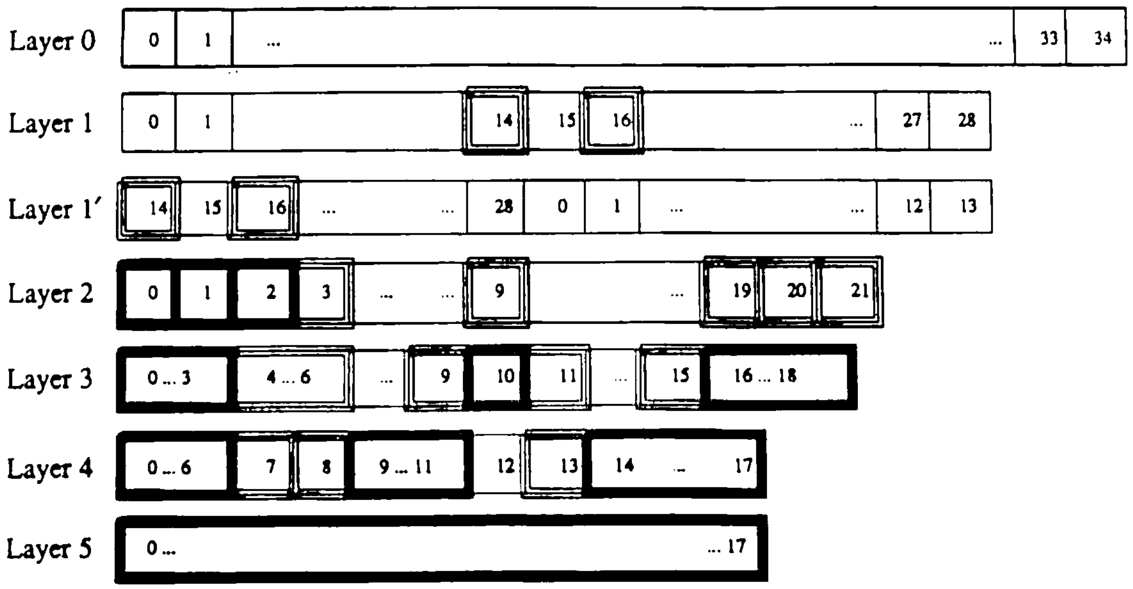


Figure 5: Correspondence among layer lists.

Initially, layer 0 is initialized with the pixels lying along the outermost layer. Since no skeletal pixels are present, positions within this layer are mapped to layer 1 by uniform scaling. That is, a scale factor of 28/34 is applied to each index of layer 0. In layer 1, skeletal pixels emerge at positions 14 and 16. This partitions layer 1 into two intervals and requires realignment of the layer. Layer 1′ illustrates the circular shift that positions pixel 14 as the first entry in the list. The two intervals are now mapped onto layer 2 by identifying corresponding *OLDSKL* points and their intervening segments. Observe that pixel 15 in layer 1 vanishes in layer 2 — it maps onto a skeletal interval. This is the first instance in which a subdivided interval has no descendants. Notice that the three *OLDSKL* pixels in layer 2 arise from the traversal of (00), (01), and (00) again. This explains why (02) is not explicitly labeled in Fig. 4.

Under most circumstances, the first few layers extracted will be free of skeletal pixels. Subsequently, some layer pixels will be labeled *SKL*. These pixels must, by definition, appear in all subsequent iterations. With each iteration, more pixels are labeled *SKL*. This is a consequence of the property that skeletons are fully connected, and thus preceding skeletal points must themselves be connected to *SKL* pixels. Also, the lengths of the lists diminish as the skeleton is approached. This follows from the fact that fewer pixels are extracted from a continuously eroding boundary. Finally, when all the pixels in the outer layer are found to be skeletal, the innermost layer has been reached, and the peeling procedure terminates.

5.5 Data Structure: L-trees

The recursive subdivision of layers gives rise to a tree representation. We shall refer to the resulting data structure as an L-tree, for layer tree. An L-tree has the following properties.

1) Each level of the tree coincides with an interior layer of the shape. Beginning with the outermost layer stored in the root, successive layers are represented in consecutive tree levels.
2) The vertices in each level represent the non-*OLDSKL* strips that have been subdivided by intervening *OLDSKL* pixels. Non-*OLDSKL* pixels include *DELPXL* and first-generation *SKL* pixels.
3) Leaves denote strips which have no descendants — that is, the entire strip maps onto an *OLDSKL* segment in the next layer.

Figure 6 illustrates a tree with nodes consisting of the non-*OLDSKL* strips from Fig. 5. The root, denoting the outermost layer, has only one child since it contains no *SKL* pixels. The child node represents the second exposed layer. It gives rise to two children as a result of the two *SKL* pixels appearing in that layer. The subdivision continues until a node's strip maps onto a skeletal point and thereby vanishes.

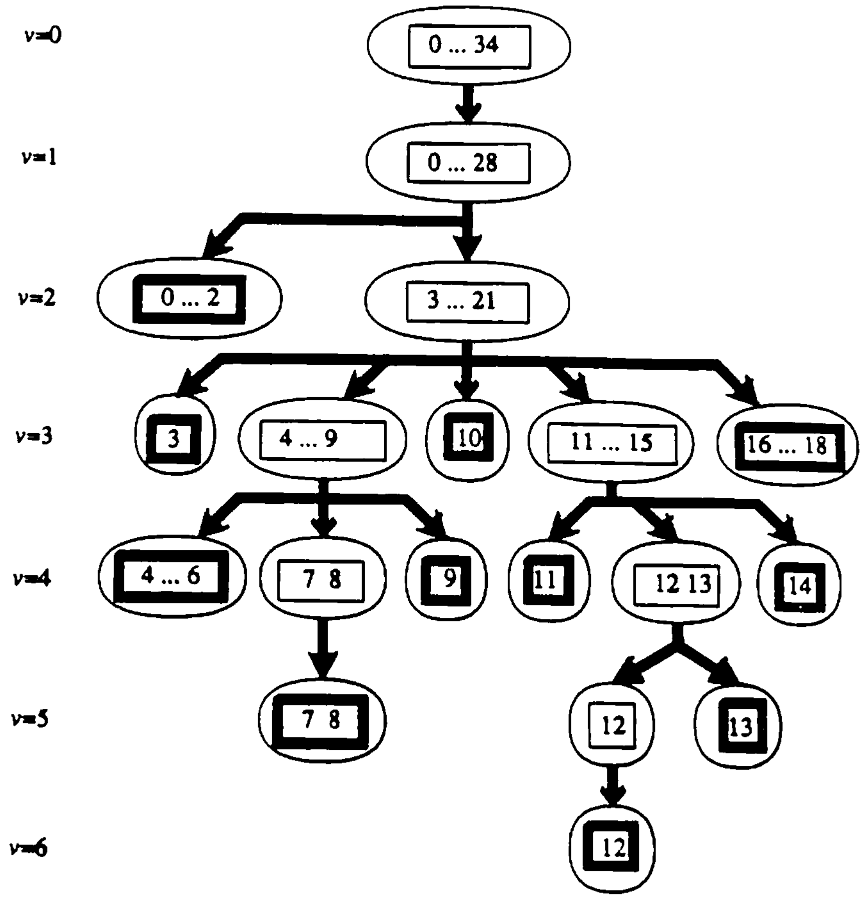


Figure 6: An L-tree representation.

Internally, an L-tree is encoded as an array of vertices, each represented by an instance of the data structure shown below.

```
struct {
        unsigned char *imgbf;   /* pointer to list memory */
        int len;                /* length of list */
        int *links;             /* pointer to auxiliary data */
}
```

The first member contains a pointer to the non-*OLDSKL* list whose length is specified by the second member. Finally, the third entry is a pointer to auxiliary data, given as a list of 3-tuples. Each 3-tuple specifies the endpoint indices of a *DELPXL* strip in the vertex interval, and a pointer to the child vertex containing the corresponding subinterval in the next layer. Note that *DELPXL* strips are delimited by *SKL*, *OLDSKL*, or boundary endpoints.

5.6 The (u,v) Parameters

Our (u,v) parameterization is conveniently represented by L-trees. The (u,v) coordinates are indices into the layer lists stored in the tree vertices.

The v coordinate runs from 0 to v_{max}, where v_{max} is the height of the tree. Beginning at the root, where $v=0$, the v coordinate is incremented at each successive tree level. This establishes a one-to-one correspondence between interior layers and tree levels. Figure 6 shows that all pixels on a given level share the same v coordinate.

The u coordinate runs from 0 to u_{max}, where u_{max} is the length of the outermost layer. Since interior layers have fewer pixels than u_{max}, the successive levels must necessarily be supersampled in order to evaluate the pixels along the radial paths. The (u_i, v_j) value is determined by performing a recursive descent from the i^{th} boundary pixel, stored in the root, onto the j^{th} level. Clearly, since the distances between boundary points and their corresponding skeletal points vary, (u,v) is defined only over a finite irregular range.

The subsequent mapping stage requires the image pixels, currently stored in the layers of the tree vertices, to be restored into an image of conventional format. Therefore, the (u,v) and (x,y) axes are aligned and the (u_i, v_j) points are collected from the distinct tree vertices into a single reparameterized image, S'. Figure 7 illustrates the structure of S'.

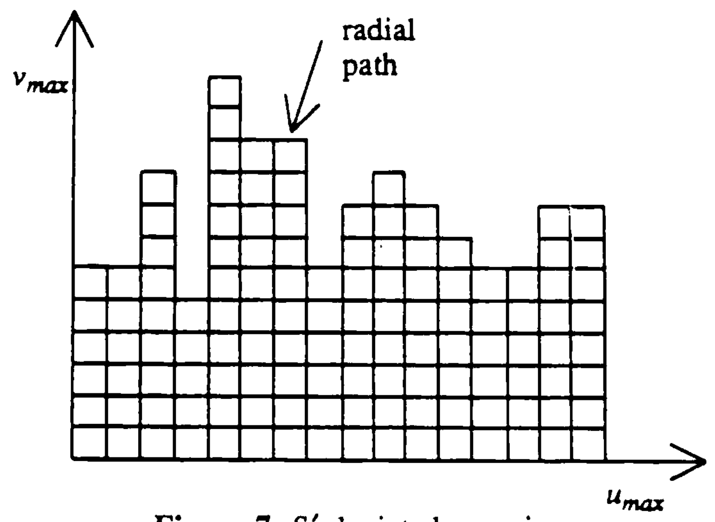


Figure 7: S' depicted as an image.

Notice that each column represents a radial path. Since the upper rows coincide with shorter layers, pixels along these rows are supersampled. In assembling S', we have discarded scale information along the rows in return for a structure amenable to standard spatial transformations. This will prove valuable in the mapping stage that follows. Furthermore, the scaling information will be restored later when the resampled S' is mapped onto T' which is then subdivided and scaled to fit back into corresponding layers. Moreover, no information is lost in this procedure since the transformations are applied over supersampled pixels.

6. MAPPING WITHIN THE (U,V) PARAMETER SPACE

This section describes the function that resamples S' onto T'. This mapping corresponds to function h in Fig. 1. The mapping solution in the (u,v) space is now more tractable than its counterpart in the rectilinear coordinate system.

6.1 Introduction

The L-tree representation has allowed us to impose a convenient (u,v) coordinate system on the image. The primary benefit of this reparameterization is that the image may now be considered as a collection of radial paths defined between each boundary point and its corresponding skeletal point. Furthermore, decomposing the image in this manner facilitates efficient referencing of interior image information using an orthogonal coordinate system. Since the range of valid (u,v) values for S' and T' are generally different, the problem becomes one of resampling S' so that its dimensions match that of T'.

There are a number of techniques available to us in performing the mapping. The most straightforward approach involves projecting each pixel in T' onto S', integrating over that area, and assigning the normalized value to the appropriate pixel in T'. However, this inverse pixel mapping technique is cumbersome, especially for applications in which the parameterized images must be referenced several times, as in multi-dimensional data and animation applications. As a result, we describe a resampling scheme which is decomposed into three simple 1D transforms: the first pass in the v direction, the second in the u direction, and the third pass in the v direction again. Therein lies the advantage of the reparameterization stage.

6.2 First Pass: Normalizing the v-axis

Unlike standard rectangular images lying on the (x,y) plane, the (u,v) space is defined over an irregular domain. The first pass of the mapping function is responsible for normalizing the v-axis in S' so that the (u,v) space is defined over a rectangular domain. This serves to establish correspondence between radial paths and facilitates a straightforward 1D scaling operation along the u-axis, a property required for the second pass.

The normalization is achieved as follows. For each u, resample the column of pixels along the v-axis so that v_{max} samples are used for the corresponding radial path. v_{max} is the height of the L-tree used to store the layers for S'. This is an appropriate choice since it coincides with the number of samples used to represent the longest radial path. This forces all columns to be supersampled at a rate dictated by v_{max} and the height of the respective column. The resulting sampling rate properly exceeds the Nyquist rate below which the highest frequency (the samples in the longest radial path) would be irreproducible.

6.3 Second Pass: Resampling the u-axis

The second pass of the mapping function scales the u-axis in S' so that its dimension matches that of T'. This serves to equate the number of radial paths in both images. Due to the first pass, a simple 1D scaling operation may be applied to each row in S'. This is a consequence of the fact that the (u,v) space is now defined over a rectangular domain. Since the boundaries of S and T are also their longest layers, respectively, the scale factor used is simply the ratio of their boundary lengths.

6.4 Third Pass: Resampling the v-axis

Now that the (u,v) parameter spaces for S and T have identical dimensions in the u direction, the information in the v direction must be made identical as well. For each u, resample the column of pixels along the v-axis in S' so that its resulting dimension matches that of the corresponding column in T'. The sampling rates are dictated by the heights of the corresponding columns in S' and T'.

The three passes are depicted pictorially in Fig. 8. Notice that the first pass supersamples all columns so that they have dimensions v_1, the height of the tree. This now allows us to apply a 1D scale operation to match the number of radial paths between S and T. As a result, the u_1 columns in S' are resampled to map onto the u_2 columns in T'. Finally, each of the u_2 columns are resampled to take on the dimensions given by T'. The intensity data in S' has now been fully mapped onto T'.

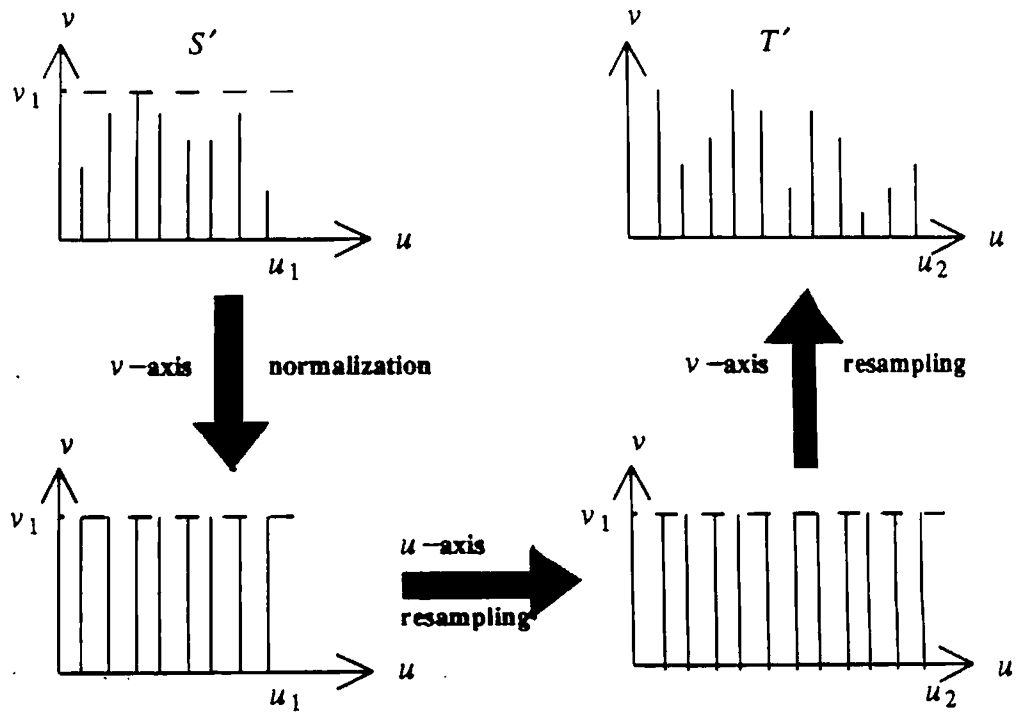


Figure 8: Three passes are required to map S' onto T'.

7. REPARAMETERIZATION FROM (U,V) TO (X,Y)

Having already initialized the content of T' with the resampled data of S', we must now reapply it onto T. This coincides with function g^{-1} in Fig. 1. Not surprisingly, this stage is the reverse sequence of operations described in section 5 that reparameterized (x,y) into (u,v). The following two steps are required.

1) Scale appropriate intervals along the rows of T' to update the layers in T's L-tree.

2) Traverse shape T while concurrently updating the traversed pixels with the values stored in the L-tree. The traversal consists of the same cycle of thinning and boundary traversal described in section 5.

The first step of resampling T' to update T's L-tree requires some discussion since it is not entirely symmetric to the forward process. When supersampling values between adjacent pixels, it is only necessary to perform interpolation. However, the inverse process of integrating supersampled values into one pixel requires two accumulators to sum the weighted values and weights. Fortunately, since row subintervals corresponding to the same layer share the same scale (weight), we can apply simple 1D scaling operations to map them back onto the appropriate layer in the L-tree. The problem now reduces to establishing a correspondence between row subintervals and L-tree vertices.

Beginning at the bottommost row in T', the pixels along the $v = 0$ row map directly to the root of the L-tree since they share the same dimensions. Conveniently, each child corresponds to a row subinterval delimited by undefined (u,v) points or the T' image border. Thus, the s descendants in the r^{th} generation correspond to the s subintervals in the r^{th} row. Since the pixels have been supersampled, the row subintervals will be subsampled to match the dimensions of their counterpart vertices in the L-tree. This completes the first phase of g^{-1}. The second phase, as outlined above, is the initialization of the pixel values along T by traversing the shape while copying the L-tree entries onto the traversed pixels.

8. RESULTS

The algorithm is written in C and runs on an EDGE 1200 super-minicomputer under the UNIX† operating system. The execution time for the 256×256 images shown was approximately 30 seconds. Note, however, that segments of the code are subject to large speedups. For instance, the thinning passes are subject to parallel processing. In addition, the 1D scaling operations are ideally suited for simple hardware implementation.

Several warping examples are given below. Figure 9 shows four images. S and T are displayed in the upper left and lower left quadrants, respectively. The upper right quadrant shows S mapped onto the shape defined by the foreground pixels of T. The mapping of T onto S is shown in the lower right quadrant. In both cases, only one boundary correspondence point is used: the upper leftmost point of S maps onto that of T.

In Fig. 10, the images in the left column are shown with superimposed skeletons. The distance between each boundary point and its corresponding skeletal point is depicted in the right column. In the upper right quadrant, the flat curve indicates that all boundary points of the box are equidistant to the skeleton, a single point at the center. The lower right quadrant shows the irregular domain of its counterpart in the lower left quad. Note that the u-axis is shown running counterclockwise along the boundary, starting from the top-leftmost point. The peaks denote large radial paths, stemming from boundary points that lie farthest from the skeleton. Similarly, valleys correspond to short radial paths emanating from boundary points lying close to the skeleton.

Figure 11 shows the effect of adding a boundary correspondence point to limit the effect of the uniform perturbation along the boundary. In the upper right quadrant only the top-leftmost points of S and T are used. In the lower right quadrant the central topmost and bottommost points are used. Notice that the checkerboard pattern is less severely skewed along the boundary in the latter case.

The effect of warping a checkerboard into a box with a spike is given in Fig. 12. Only one boundary correspondence point is supplied at the top-leftmost point. Notice that a minor boundary perturbance has spurred a long skeleton that grossly distorts the checkerboard pattern. This issue is addressed in section 9.

† UNIX is a trademark of AT&T Bell Laboratories

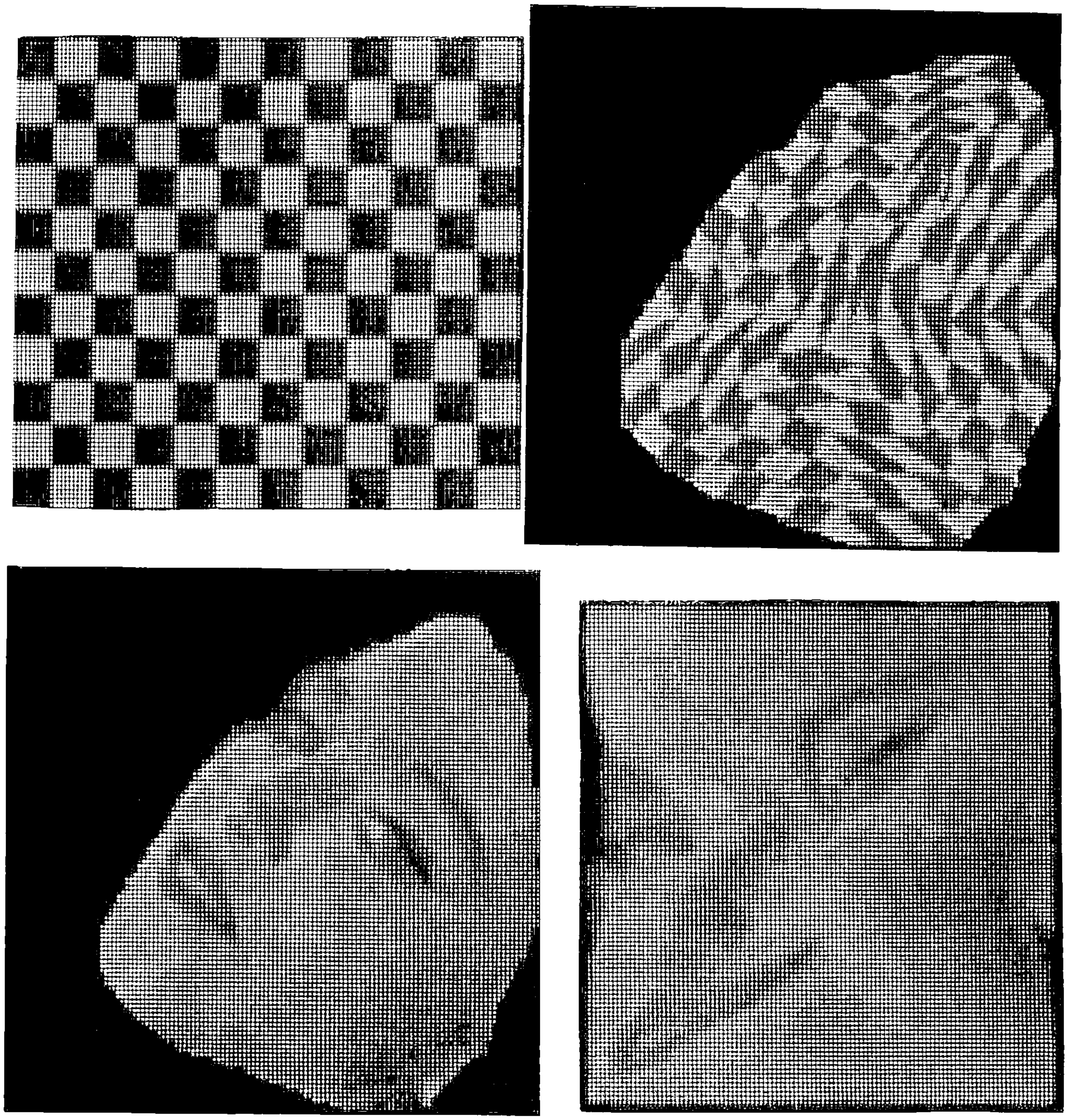


Figure 9

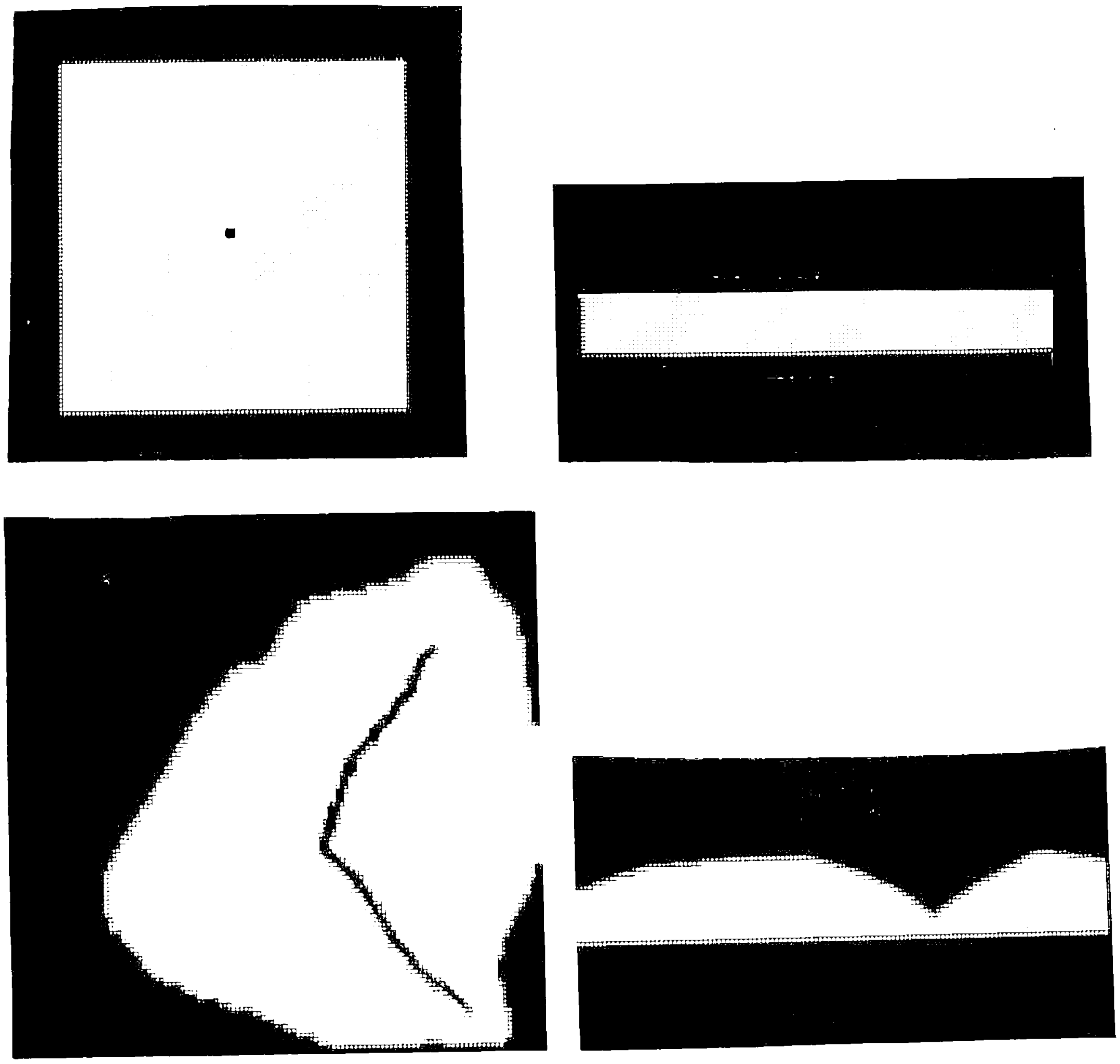


Figure 10

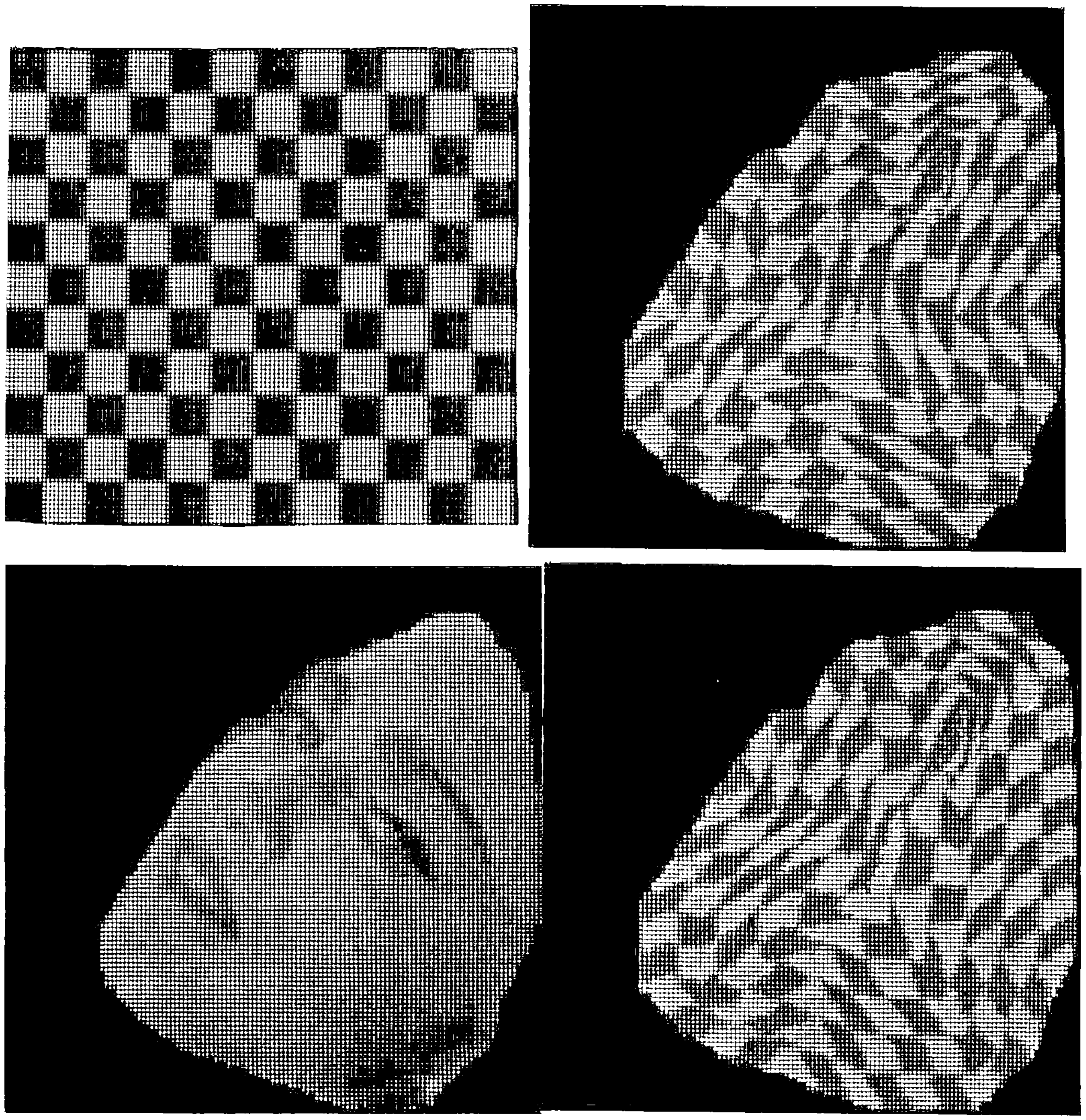


Figure 11

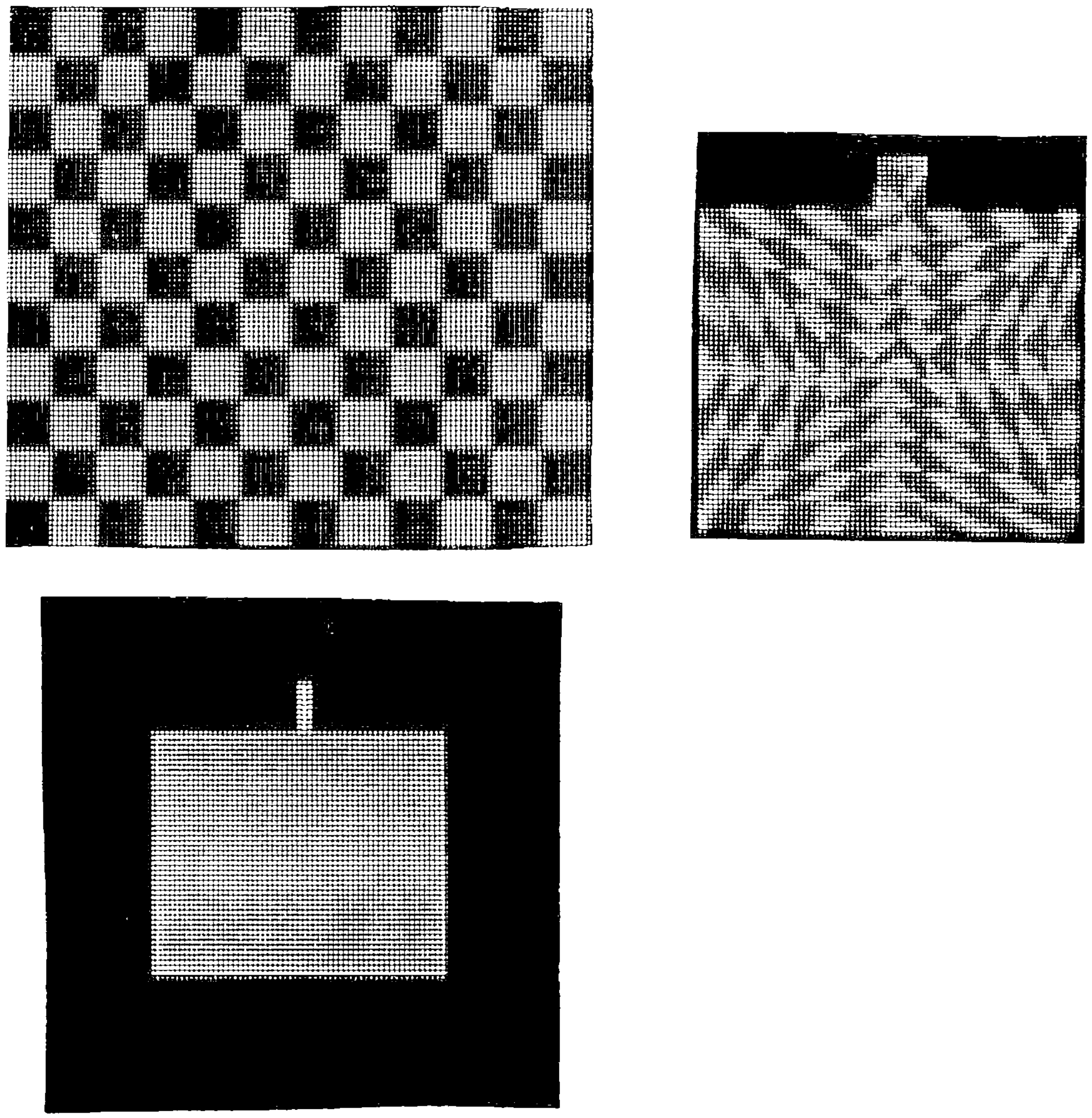


Figure 12

9. ENHANCEMENTS

This section lists some enhancements and issues which can be used to improve the results and versatility of the described algorithm.

9.1 Skeleton Editing

The thinning algorithm used to assure the closure of extracted layers is well known for its sensitivity to noise. This manifests itself as extraneous branches that are spurred by noise along the boundary, i.e. spikes. These branches may then be responsible for undesired compression or expansion of interior pixels. To counterbalance this problem, there is a facility to edit the skeleton. That is, the user may pick a subset of the displayed skeleton as those skeletal points onto which the boundary ultimate maps. In this manner, a more desired skeleton may be used for warping. Note that another arbitrary skeleton that does not entirely lie on the original skeleton cannot be used.

9.2 Use of a Better Distance Transformation

In addition to editing a skeleton, it is also worthwhile to use a more accurate distance transformation to compute the skeleton. Some recent work is described in Borgefors (1986) and Dorst (1986) in which large neighborhoods are applied in a 2-pass algorithm to determine the distance of each foreground pixel from the boundary. The skeleton is taken as the peaks of the resulting distance map. While it might appear natural to use this algorithm to evaluate the radial paths as well, there is the issue of multiple equi-distant paths to address. Nevertheless, the use of skeletons with improved noise sensitivity is a valuable asset.

9.3 Non-uniform Scaling

The algorithm described here used uniform scaling exclusively. This is inappropriate near corners where the Manhattan distance is a poor approximation to the Euclidean distance metric. Non-uniform scaling, together with additional boundary correspondence points, can be used to offset the distortions (skew) that manifests near corners and other points of high curvature.

9.4 Interior Holes

Adding interior holes changes the topology of the shape. Recall that the target image must be topologically equivalent to that of the source image. Therefore, S and T must have the same number of holes. With each hole, there is an additional set of boundary correspondence points to specify.

9.5 Applications

Warping an arbitrary shape onto, say, a rectangle, has potential value for fast convolution. The Fast Fourier Transform, for instance, can only be applied upon rectangular images. Applying it to an arbitrary shape would include unwanted neighboring pixels. With the proposed warping algorithm, it is possible to warp shape X into rectangle Y, apply the FFT to Y, and then warp Y back into X, its original form. In order to avoid information loss, X must be entirely contained in Y so that the convolution is applied upon supersampled pixels.

Another application may be to use the domain of S', as given in Fig. 10, as a shape measurement that can be used for quantifying shape deformation. This presupposes that the skeleton, which is the basis of the measurement, remains fairly consistent among the sampled images and the reference model.

Finally, the role of this warping algorithm for visual effects is obvious. If an image is partitioned into jigsaw pieces, it is now possible to arbitrarily distort the jigsaw boundaries, warp the interior, and create sophisticated special effects.

10. SUMMARY AND CONCLUSIONS

This paper describes an efficient algorithm to perform image warping among arbitrary shapes. The resulting spatial transformation is derived using boundary correspondence specified by control points. These points serve to clamp the effect of the warp perturbation to specified intervals.

The algorithm formulates a convenient homomorphism to yield a tractable solution to this problem. The chain of transformations begins with the reparameterization of the source and target images. This consists of initializing two L-trees with values that are peeled off both shapes, one layer at a time. A thinning algorithm supplements this procedure to assure that *closed* layers are extracted, thereby guaranteeing that all interior pixels are considered. Once the L-trees are initialized for the source and target images, the warping problem becomes one of mapping one L-tree onto another. Since the pixels are now scattered in a tree, we supersample the tree and collect the data into a standard image format. This facilitates the application of three 1D transformations to map the reparameterized source image onto that of the target. Having done this, we simply restore the target's L-tree and reapply the updated pixel values onto the target image. This sequence yields the desired result.

Facilities are available for establishing boundary correspondence points. This enables the warping to be clamped to specified intervals. Furthermore, the skeleton may be edited to remove extraneous branches that would otherwise give rise to unwanted compression or expansion. This feature is an effort to address the noise sensitivity intrinsic to skeletons. An improved distance measure would prove more robust against noise and yield improved skeletons.

Aside from the obvious visual effects application, this algorithm is well-suited for warping an arbitrary shape into a rectangle, a shape suited for FFT filtering. This enables fast convolution without incorporating neighboring background pixels. More studies remain to be done on the effectiveness of this method for this purpose. Finally, an extension of this algorithm from the current discrete implementation to a continous domain offers promising possibilities for increased accuracy and control. This would prove valuable for the mapping, analysis, and registration of 2D and 3D data useful in a variety of applications.

10. REFERENCES

Arcelli C, di Baja GS (1985) A Width-Independent Thinning Algorithm. *IEEE Trans. Pattern Anal. Machine Intell.* 7(4) : 463-474

Borgefors G. (1986) Distance Transformations in Digital Images. *Computer Vision, Graphics, and Image Processing* 34(3) : 344-371

Burt PJ (1981) Fast Filter Transforms for Image Processing. *Computer Graphics and Image Processing* 16 : 20-51

Crow FC (1984) Summed-Area Tables for Texture Mapping. *Computer Graphics*, (SIGGRAPH '84 Proceedings) 18(3) : 207-212.

Catmull E, Smith AR (1980) 3-D Transformations of Images in Scanline Order. *Computer Graphics*, (SIGGRAPH '80 Proceedings) 14(3) : 279-285

Dorst L (1986) Pseudo-Euclidean Skeletons. *Proceedings* 8th *Intl. Conf. Pattern Recognition*, pp. 286-288.

Fant KM (1986) A Nonaliasing, Real-Time Spatial Transform Technique. *IEEE Computer Graphics and Applications* 6(1) : 71-80

Fiume E, Fournier A, Canale V (1987) Conformal Texture Mapping. Proceedings of *Eurographics* 1987, pp. 53-64.

Fraser D, Schowengerdt RA, Briggs I (1985) Rectification of Multichannel Images in Mass Storage Using Image Transposition. *Computer Vision, Graphics, and Image Processing* 29(1) : 23-36

Greene D, Lamming M (1986) Interactive Distortion of Images. Xerox Palo Alto Research Center, 1986.

Heckbert P (1986) Survey of Texture Mapping. *IEEE Computer Graphics and Applications* 6(11) : 56-67

Oka M, Tsutsui K, Ohba A, Kurauchi Y, Tagao T (1987) Real-Time Manipulation of Texture-Mapped Surfaces. *Computer Graphics*, (SIGGRAPH '87 Proceedings), 21(4) : 181-188

Pavlidis T (1982) An Asynchronous Thinning Algorithm. *Comput. Graph. Image Processing* vol. 20(2) : 133-157

Rosenfeld A, Kak A (1982) *Digital Picture Processing*, Volume 2, Academic Press, NY

Smith AR (1987) Planar 2-Pass Texture Mapping and Warping. *Computer Graphics*, (SIGGRAPH '87 Proceedings), 21(4) : 263-272

Williams L (1983) Pyramidal Parametrics. *Computer Graphics*, (SIGGRAPH '83 Proceedings), 17(3) : 1-11

Wolberg G (1985) An Omni-font Character Recognition System. M.E.E thesis, Cooper Union School of Engineering, Oct. 1985. (Available from UMI, Ann Arbor, Michigan.) Also appears in Proceedings of *IEEE Computer Vision and Pattern Recognition*, June 1986.