International Journal of Computer Vision, 3, 51–71 (1989)

Building Surfaces of Evolution: The Weaving Wall

H. HARLYN BAKER
Artificial Intelligence Center, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025

Abstract

This paper describes a three-dimensional surface-construction process designed for the analysis of image sequences. Named the *Weaving Wall*, the process operates over images as they arrive from a sensor, knitting together, along a parallel frontier, connected descriptions of images as they evolve over time. Although the Weaving Wall was developed to support a tracking mechanism for recovering the three-dimensional structure of a scene being traversed, other applications of the surface-building process have since become apparent. These include rendering and computation on tomographic medical data, display of higher-dimensional analytic functions, edge detection on the scale-space surface, and display and analysis of material fracture data. More generally, the Weaving Wall may be of use in representing the evolution of any two-dimensional imagery varying in a nearly continuous manner along a third dimension. We are currently looking into extending the processing to higher dimensions.

1 Surfaces of Evolution

1.1 Background to the development

Images of real objects have an inherent spatial coherence that in general is quite obvious. This coherence persists as the objects are smoothly moved about or viewed from varying perspectives. Indeed clarity in the perception of a scene's structure is often enhanced through either object or viewer motion. Here, the temporal coherence in the imagery reinforces that observed spatially. The research described in this paper developed from our efforts in generalizing epipolar-plane image analysis [1] to cases where both the camera and the objects in the scene are free to move about. Although we have not yet attained all of this, the process we describe here is supporting these developments.

Coupled with the viewing generalizations was a desire to enable sequential processing of the image data, while maintaining an evolving description of the scene. Two items in particular in this effort complicated the analysis and necessitated fairly radical redesign from our earlier system: an intention of handling sequences with varying camera attitude, and the determination to produce spatially coherent results. In our earlier work, we retained the temporal continuity from sequences of images and used it quite effectively in tracking and localizing features in the scene, but, in so doing, discarded the spatial continuity apparent in the individual images. Our broader goals for this research made us realize that this was information we must not lose—our analysis will depend on both temporal and spatial image structure (see the companion paper [2] in this issue for a discussion of this point). Thus, we needed to develop a mechanism for building an explicit description of the spatial structure and temporal evolution of images acquired over time.

1.2 What Does It Mean for an Image to Evolve?

If objects move or a camera moves about a scene, the projections of scene features will move about in the image. If the motions are sufficiently fine, there will be no difficulty in following the various components of the scene in their independent

travels. Consider, as we do for the rest of this discussion, that it is the camera that is undergoing motion while viewing a static scene. If the camera slides left at right angles to its direction of view, we would expect to see the scene slide right, with the nearer objects moving most rapidly out of view at the right. If the camera is instead looking directly along its direction of motion, we would experience a looming effect from objects lying straight ahead, and see those off to the sides slowly accelerate, then fly out of sight as they approach the edges of the frame. If we had a perfect segmentation scheme (which is probably impossible), we might be able to outline an object in the individual frames in which it is seen and collect all of these together into a cone following the path it took as it moved about the image and then disappeared from view. This outline would change size and shape as the object came closer or receded, and as it presented different parts of its surface to view. This cone, a surface or two-dimensional (2D) manifold in a three-dimensional (3D) space, would define the evolution of the projection of that object over time. If we could track all such outlines for all objects in the scene, we would say we had a description of the evolution of the imagery over time. This is the description we want. The outlines encode spatial coherence, and along their axes the cones maintain the objects' temporal coherence.

Not having perfect segmentation, we will not expect to have disjoint outlines for all objects in an image. Given the variability of lighting conditions, sensor noise, reflections and texturing, etc., we might be more realistic in expecting nothing short of a hideous swirl of contours. But given a continuous deformation in the imagery, even this hideous swirl will sweep out a surface as we watch it in space-time. We have developed, and describe in this paper, a process that builds a representation of the spatial and temporal structure of these surfaces.

In the companion paper [2] we specify the computational utility of this description in the context of our spatiotemporal tracking and surface-reconstruction work. Our intention in this paper is to describe the development and operation of the surface-building algorithm, to indicate applications in which it has proved to be valuable, and to suggest others in which it may.

1.3 Is This Surface Building New?

Constructing surfaces in 3D data such as this has been done before, although not in a manner that could be effective for our needs. Artzy et al. [3] describe what has become the principal approach to surface reconstruction from sensed tomographic-style data; it is, in fact, the *only* surface constructor of note. In their approach, which was developed in the context of surface building from medical computed-tomography (CT) data, each surface is processed separately. The processing begins with the selection of a seed voxel of the desired density (measured in Hounsfield units). A sequential recursive search is then used to traverse the implied surface, forming a connected set of all those facets positioned between adjacent voxels where one has density less than and the other has density greater than or equal to the chosen value. Since surface traversal is by a connected-component search through the 3D volume, all of the data must be acquired beforehand. The surfaces traversed in this manner are guaranteed to be closed. Once one surface has been traversed, the next may be begun. This is the *cuberille model* of surface representation. Wyvill et al. [4] improved on the search efficiency by characterizing local surface structure, but maintained the technique's inherent sequential nature.

Although the description we seek is closely related to these, our requirements demanded a very different approach:

1. *All* the surfaces in the sequence are of interest to our tracker, not just selected ones, and we need to follow them all.
2. Our aim of autonomous navigation through sequential processing means that we would never expect or desire to have all the data available in advance—our data (images from a camera) are obtained as we move, and must be processed as they are acquired.
3. A recursive traversal of individual surfaces is not appropriate, nor is processing the surfaces separately in sequence—all surfaces must be

built incrementally and in parallel as each new image is acquired.

4. Integral positioning of facets is insufficiently precise—for accurate mapping we need subpixel resolution in surface definition.
5. Simple density or intensity themselves are inappropriate for surface definition—we track the evolution of image features over time, and define our features, and hence our surfaces, as being located at the zeros of Laplacians of a Gaussian.
6. We will have need of various sorts of computation on the local surface as it is being constructed, for example, feature tracking, and the organization of the process must support this.

Each of our design objectives differs in important ways from Artzy et al. [3], but the point most distinguishing of our approach is the third: Both sequential processing and a capability for parallel implementation are crucial for a realistic tracking system.

Sander and Zucker [5] introduce another approach to surface definition. Their method is directed at obtaining a globally coherent surface through the use of a relaxation process applied over local surface approximations. Surface elements are selected by a Sobel-type 3D gradient operator [6], and filtered using thresholds and nonmaximum suppression. Surface elements with large responses are categorized as elliptic, parabolic, or hyperbolic, depending upon the Gaussian and mean curvatures determined from the local fit of a quadric.

The principal advantage of their approach may be in object-matching applications where their local qualitative parameterization could enable effective surface comparison. As it is, their surfaces lack certain properties that are necessary for our work; for example, we need the surfaces to be closed. Furthermore, there could be no justification in assuming temporally evolving imagery to be locally quadric; and errors that might be smoothed by the fit of an analytic function would be better handled through robust estimation. Since their technique is an iterative refinement procedure, it would be unlikely to be amenable to sequential processing. From the perspective of sequential estimation, perhaps the greatest limitation of their approach is that establishing surface connectivity requires iteration and search in the three-space volume, whereas the approach we devised is direct and requires no search.

1.4 Surface-Building Operation

The surface constructor we have developed operates on images sequentially as they are acquired, knitting together a connected representation of the spatial structure and temporal evolution of an image sequence over time. It maintains the continuity of feature paths irrespective of changes in viewing angle, or a varying rate of image acquisition; it's sole requirement of the data is that between frames, surfaces move in some direction no more than their width.[1] The processor acts as a *loom* during surface construction, with a wall of accumulators meeting each image in sequence and weaving its elements into the mesh of surfaces it has prepared behind it. From this action we give it its name, the *Weaving Wall.*

Since the principal reason for developing the Weaving Wall was for use in motion sequence analysis, we will develop its operation in the context of Laplacians. For additional applications, which we will touch on later, other measures may be more appropriate—for example, we use Hounsfield density when we process CT data, and track not density zeros, but zeros with respect to a bias, the chosen surface density value. But the distinction is only incidental to the development, and we will work for now with the Laplacians.

2 Weaving Wall Design and Implementation

2.1 Local Surface Elements from the Volumetric Data

One characteristic presented in the derivation of the Artzy et al. surface-reconstruction process, a

[1]Otherwise they would become disjoint at that point, and form two volumes, not one.

binary relationship on the voxels, holds for an important element of our processing. In CT tomographic reconstruction, voxels bear density measures, and for a chosen density a voxel is either *inside* (which includes *on*) a surface—its value is at least as great as that chosen—or *not inside*—its value is less than that chosen—and there can be no path from one to the other that does not cross the surface frontier. In our case the situation is identical: a voxel bearing a value from a Laplacian of a Gaussian is either positive or nonpositive, and a path from a positive voxel to a nonpositive one must pass through zero. This guarantees that our surfaces, being located at these zeros, will be closed. In fact, they will enclose positive or nonpositive voxel values, depending on our choice of sign. This definition ensures that each surface is a Jordan curve (Jordan surface). In essence, this means that a surface S divides R^3 into 2 volumes: that inside and that outside of S. This allows a finite case analysis of the local surface.

In 2D images, zero crossings will form closed *regions* composed of pixels having, say, positive Laplacian value. In three dimensions, the zero crossings will form closed *volumes* composed of voxels having Laplacian values of the same sign. In 2D, we define the region frontier to be a contour composed of two types of edge element (edgel): a U edgel is oriented vertically and positioned between oppositely signed, horizontally adjacent Laplacian pixels, and a V edgel is oriented horizontally and positioned between oppositely signed, vertically adjacent Laplacian pixels (see figure 1). Interpolating the edgel between the two pixels allows it to be positioned up to ± 0.5 units horizontally or vertically from the boundary between them. When working with an image sequence, we form a 3D data set by treating the collection of images as a set of arrays of voxels each one-pixel deep. Here, we define the volume zero-crossing frontier to be a surface composed of three types of voxel facets: U and V facets separate voxels horizontally and vertically, as in the 2D case, while T facets separate voxels temporally—they occur between oppositely signed voxels at identical positions in adjacent images. Again, interpolation allows us to position the surface elements with subvoxel precision.

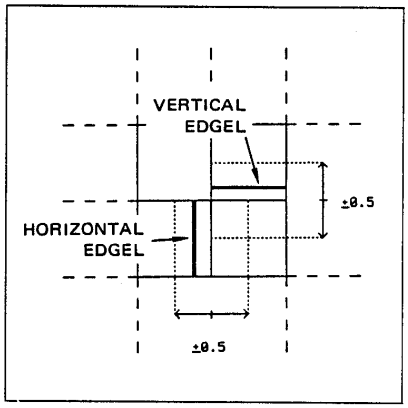

Fig. 1. Region edgels in two dimensions.

Having defined what our surface elements are, we must demonstrate that we can obtain them from the data. We will do this with reference to both the 2D and the 3D cases, and will use both monocular and crossed-eye stereo displays to indicate the geometry behind the processing.

Given the local nature of the above definition, where facets are positioned between adjacent voxels based solely on their Laplacian values, it would seem that we should be able to detect a facet with very simple tests applied to only local parts of the data set. In fact, everything necessary for determining the local surface structure is available in each 2×2×2 window of the data. Consider a 2×2×2 set of voxels containing Laplacian values. There are 2^8 different combinations of positive and nonpositive Laplacian values in these 8 voxels. If some of these voxels are of different sign, then we have a surface (or several surfaces) passing through this 2×2×2 part of the data set. Presume that we have processed at least one image, and are now processing the next in the sequence. Our scanning is bottom to top, and within that, left to right (the development for a parallel implementation differs in detail, and has not been completed). We label our facets as indicated in the stereo display of figure 2a, where the U facets are from the set (*U U-1 U-2 U-3*), the V facets are from (*V V-1 V-2 V-3*), and the T facets are from (*T T-1 T-2 T-3*). Ignoring boundary con-

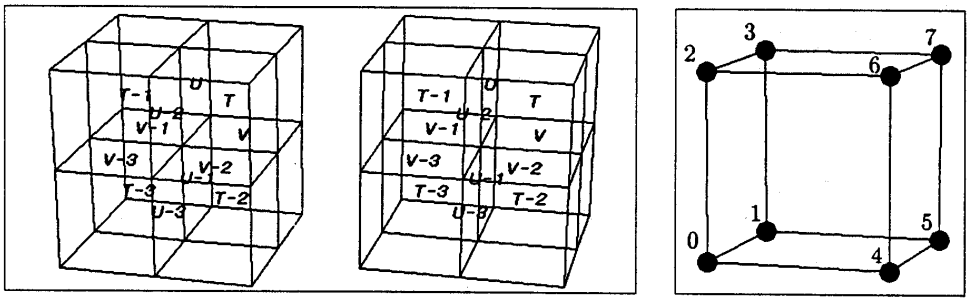

Fig. 2. (a) Facet labeling on three-space surface. (b) Voxel indexes.

ditions that occur at the edges of the frame, we can see that a simple 8- bit index based on the signs of these voxels can inform us of the local surface structure.[2] Addressing voxels by their $\{uvt|u, v, t \varepsilon (0,1)\}$ relative indexes, we can number them from 0 through 7, with 0 being the near lower left (0,0,0), 1 being the far lower left (0,0,1), 7 being the far upper right (1,1,1), etc., as shown in figure 2b. If voxel 7 is positive (termed *on*), and the rest are nonpositive (termed *off*), our 8-bit index is 200_8, and we will have a single surface having facets at *U*, *V*, and *T* (figure 3a). The interpolated position of these facets will depend on the values of voxels 3, 5, and 6 with respect to voxel 7. Figure 3b uses circles to encode voxel state (adapted from the display of Lorenson and Cline [8]), with filled meaning *on* and empty meaning *off*. If voxel 0 is also *on*, our index is 201_8 and we will have two surfaces, one as we had with 7 alone *on*, as above, and the other having facets at *U-3*, *V-3* and *T-3* (figure 4a and figure 4b). If only voxels 3, 5, and 7 are *on*, the signature is 250_8, and we will have a single surface with facets at *T-1*, *T*, *T-2*, *V-1*, and *U-1* (figure 5a and figure 5b). Sliding the 2×2×2 window one pixel in each direction will give us the adjacent local surface structure in that direction. If we do this throughout the entire volume, we will have a description of all the surfaces in the data set.

[2]This approach grew out of a 2D contour finder having a 4-bit index, developed by Marimont [7].

2.2 Surface Connectivity and Facet Adjacency

Notice in the above that we have defined neighboring voxels as being part of a particular volume if they have a face in common—edges and corners are not sufficient. This means that the complement space, made up not of enclosed

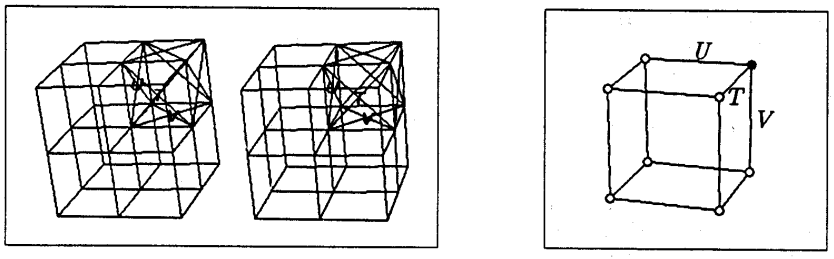

Fig. 3. (a) Local surface facets, $s = 200_8$. (b) Indexed voxels.

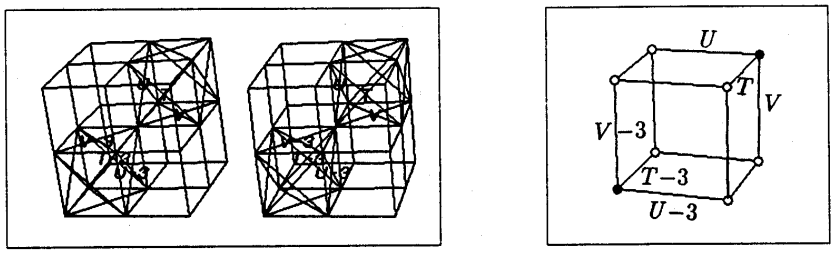

Fig. 4. (a) Local surface facets, $s = 201_8$. (b) Indexed voxels.

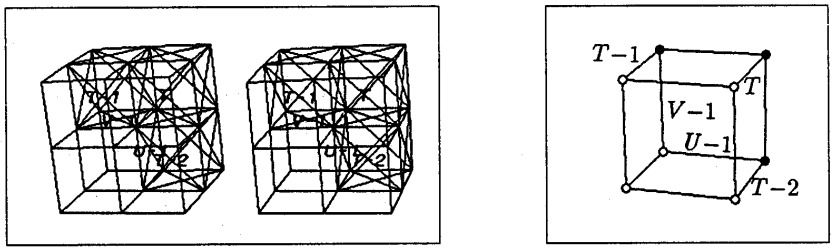

Fig. 5. (a) Local surface facets, $s = 250_8$. (b) Indexed voxels.

voxels but of *excluded* voxels, has a connectivity definition for its voxels that *includes* edges and corners. This may be visualized best in two dimensions, as demonstrated in figure 6. There, with pixels numbered 0 through 3, two cases are presented. First, with pixels 1 and 3 *on*, we have a single contour between pixels (0,2) and pixels (1,3), and it encompasses a region above that frontier (figure 6a). The complement region shares this contour, but encompasses the pixels below it. If, on the other hand, pixels 0 and 3 are *on*, we will have two contours passing through the 2×2 window, one with edgels between pixel 0 and pixels 1 and 2, the other with edgels between pixel 3 and pixels 1 and 2 (figure 6b). The first region encompasses pixel 0 and the other encompasses pixel 3. The two pixels are not judged locally to be part of the same region since they do not share an edge. The complement region, however, encompasses pixels 1 and 2, in fact squeezing itself between the two other regions. If a reverse *on* sense were used (nonpositive rather than positive), we would *not* have an identical segmentation. Rather, pixels 1 and 2 would be parts of different regions, and pixels 0 and 3 would be part of the complement region (figure 6c). This means that the choice of enclosure sign affects more than just the sense of the regions; it also affects their topology.

Being consistent in this definition of adjacency is fairly crucial to maintaining a coherent surface description in three-space. Lorensen and Cline [8] describe an independently develop algorithm for surface reconstruction based on the same use of voxel sign signatures. In their attempts to reduce the complexity of the coding, they fold the mapping about many of its symmetries. In general, *any* folding that is not undone in a later mapping would destroy the spatial coherence of our weaving. They, however, do not seek a coherent surface description, settling instead for a set of triangular facets appropriate for rendering. One of their foldings is to map the signature to its

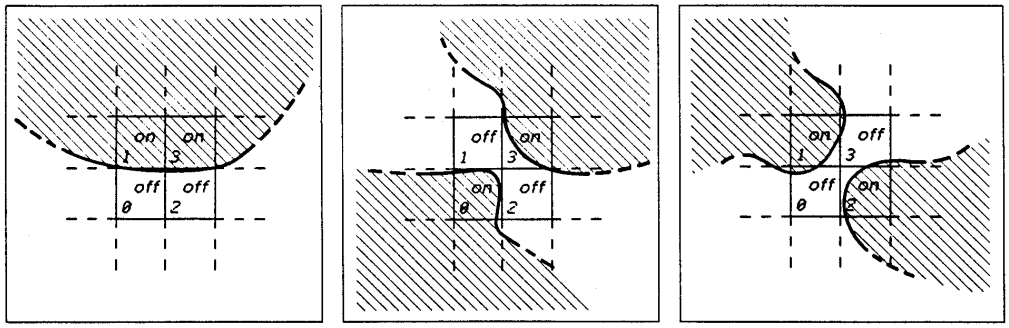

Fig. 6. (a) Pixels 1 and 3 *on*. (b) Pixels 0 and 3 *on*. (c) Pixels 1 and 2 *on*.

minimum with respect to number of voxels *on*. If five voxels are *on* with a signature of 37_8, this will be recoded as 340_8, having the previous five voxels now *off* and only three voxels *on*. The effect is to treat the local pattern of *on/off* voxels as possibly a pattern of *off/on* voxels, complementing their true sense. The 2D example of figure 6c shows that this symmetry mapping is, in fact, not valid—it destroys the coherence of the description. In 3D this is most disruptive at saddle points. The displays of Lorensen and Cline [8] are dense enough that this isn't noticed visually.

Now, individual facets are related to those on the surface around them in the same way that individual edgels in 2D are related to the edgels adjacent to them on a contour. 2D edgels have two-connectivity—they have adjacent edgels to their *left* and *right* (with respect to some sense): Adjacency of edgels requires shared endpoints. 3D facets have four-connectivity—they have adjacent facets *up*, *down*, and to their *left* and *right* (again with respect to some sense): Adjacency of facets requires a shared edge. For example, when voxel 7 alone is *on* (having a signature of 200_8), the three faces involved (*U*, *V*, and *T*) are set to indicate their neighbors; *U* and *T* are to each other's *left* and *right*, respectively, *U* and *V* are to each other's *right* and *down* respectively, and *V* and *T* are to each other's *down* and *down*, respectively. The sense is with respect to looking in at the *on* voxel from outside the face, with *up* being in the direction of increasing *V* for *T* facets, and in the direction of increasing *T* for *U* and *V* facets. Note that, in general, facet connectivity is of order exactly 4. Each of the 4 edges of a facet will be shared by just one other facet, and only when a voxel is on the boundary of the data set will there fail to be facets to share all of its edges.

2.3 Ancillary Surface Computations

The coding of a local topographic signature also enables very efficient implementation of surface property computations that would otherwise be very expensive to implement, requiring total traversal of all the surfaces once completed. First among these is the maintaining of distinction between surfaces that are disjoint. Each surface is given an identifying index when first encountered, and this index is given to each face that is later found to be part of that surface. Only certain patterns of *on/off* voxels can arise from the uncovering of a new surface; ignoring boundary conditions, the local configuration has to be such that it includes an *on* convex corner at voxel 7.[3] Not all such corners are actually new surfaces, however, as the local information is insufficient to determine whether the surface to which these facets belong has already been encountered. This

[3]All boundary conditions are taken care of by 1D or 2D versions of the 3D operations outlined here.

can only be detected when the two surfaces (the earlier and the one containing the new convex corner) come together in a 2×2×2 window. Facet linking provides for the combining of surfaces when their frontiers merge. With this approach disjoint surface computation is primed by precompilation into the signature dispatcher and finalized on the fly by the facet linker; at each stage of the processing, the surface labeling maintained is the most compact one possible.

Another measure that is similarly encoded in the voxel signature is the determination of surface Cartesian bounding volume. As was the case with the characterization of new surface arrival, the voxel configurations that contribute (or *may* contribute) to an adjustment on the enclosing Cartesian frame for a surface can be characterized by their local topography. A convex corner with voxel 7 *on*, as above, may affect the lower limits in all of u, v, and t for the surface. Similarly, voxels 2 and 3 *on* can mean an altered upper u limit and an altered lower v limit for their surface (see figure 2b), but cannot affect the t limits (again, ignoring boundary conditions). These tests are compiled into the signature dispatcher as appropriate.

This attribute coding and incorporation into the weaving process does two things: it eliminates a later (and very expensive) sequential search; and it minimizes the time spent overall by restricting the calculations to only those parts of the surface where they are necessary.

The act of creating a new facet also maintains gradient mean and variance statistics for each surface. These can be of use in filtering surfaces by their significance: a low mean indicates a subtle surface more likely to be spurious; a surface facet whose gradient is considerably lower than its mean is a candidate as a segmentation point (in the fashion of Canny [9]).

2.4 Economies in Construction

An advantage of the sequential processing—top to bottom and left to right—forced on us by using a sequential machine, is that we know when processing that certain facets in the 2×2×2 window have already been created and linked together. In fact, at any given (u, v, t), we are only creating facets of type U, V, and T, and linking facets that are adjacent across the three edges of that octant. Figure 7 sketches the local voxel configuration for a case having signature value of 267_8 (a saddle point). Only two new facets and four links were needed in constructing the fairly complex local surface for this case. All other facets and links were created earlier.

Also important to note is that the state of voxel 0 affects neither the face and surface creation nor the face linking operations. There are no U, V, or T facets in that octant, nor links to any of these facets. In effect, this removes one bit from our signatures, and lets us halve the state space. Exploiting this coding efficiency, the surface-construction code for case 267_8 of figure 7 is also appropriate for the case of signature 266_8, as shown in figure 8. Notice that this is not a saddle point, and locally involves not one but two surfaces.

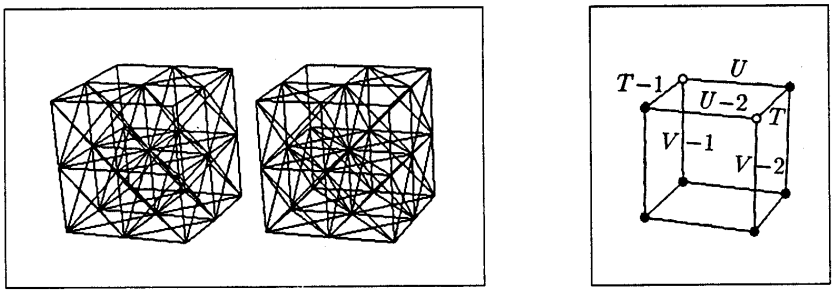

Fig. 7. (a) Voxel configuration, 267_8. (b) Indexed voxels.

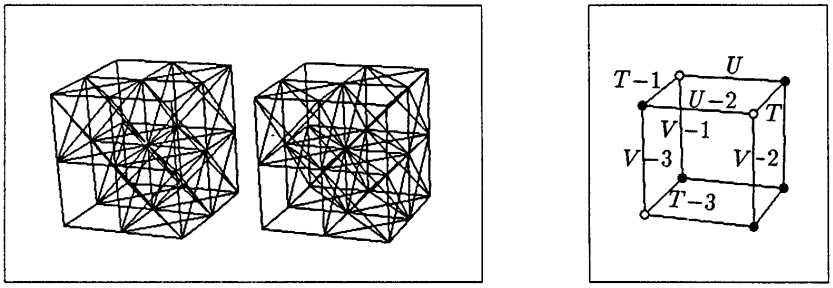

Fig. 8. (a) Voxel configuration, 266_8. (b) Indexed voxels.

Figure 9 shows a local surface configuration differing from that of figure 7 by, again, a single sign bit—voxel 5 here is *off*—and this gives it a signature of 227_8. This is identical topologically to the local surface of figure 8, but is oriented differently. This slight change leads to a rather different set of instructions for the weaver, including the creation of a new surface.

2.5 Representational Duality

A surface representation composed of the facets separating *on* and *off* voxels, as indicated in figure 2a, is one way of viewing the results of the Weaving Wall. This is the view in the *cuberille* model, although without the interpolation (which in effect allows our facets to be of varying size). Each planar facet has integral coordinates along 2 axes, an interpolated position along the third, and an estimate of surface normal computed by the 3D gradient convolutions. An enriched view of the surfaces is obtained if one chooses to consider them in a dual sense, the principal elements of which are the patches bounded by arcs joining adjacent facets. The shape of the patches can be estimated by interpolation of their vertex (facet) normals. Since the boundaries are oriented along the three principal axes, this gives a Cartesian ruled-surface representation, as indicated in figure 10, showing, in crossed-eye stereo form, a dart-shaped cone of circular cross-sections. Patches can have from three to seven linear arcs on their boundaries: three is a triangle (see figures 12a,b in section 2.8); four, five, and most sixes are simple nonplanar shapes; some sixes and all sevens are saddle points.

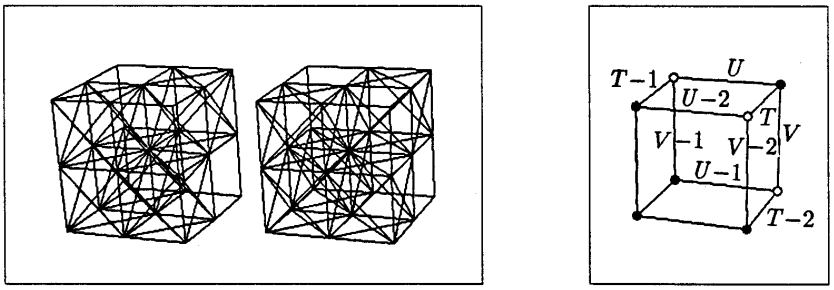

Fig. 9. (a) Voxel configuration, 227_8. (b) Indexed voxels.

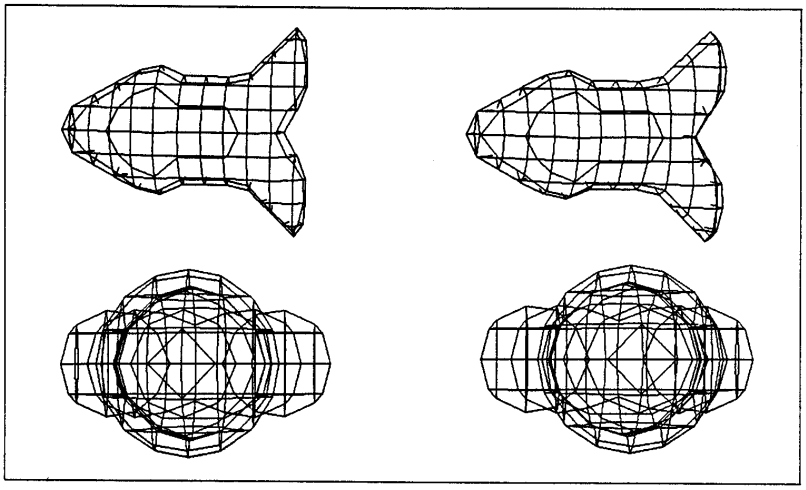

Fig. 10. Cartesian grid surface: Top view (above), front view (below).

2.6 Locating Facets

We have discussed the structure of our surfaces and their construction from local surface elements, but have not described how we locate and parameterize those individual elements—the facets. Our data is three-dimensional, two being spatial with the third being temporal; in determining the facet positions and orientations, we convolve this data with a battery of three-dimensional filters, treating the three dimensions as isotropic. Facet *orientation* is determined by computing the three partial directional derivatives about each voxel. We compute these derivatives by forming a partial derivative of a chosen 3D Gaussian and convolving it in the appropriate direction at each voxel. Computation of facet *position* is accomplished in a similar manner: we compute the sum of the second partial directional derivatives of the Gaussians at each voxel, and position the facets by linear interpolation at zeros of these values.

The Gaussian is a separable filter, so we can obtain the three-directional derivative components as

$$\partial_u G(I) = G'_u \otimes G_v \otimes G_t \otimes I$$

$$\partial_v G(I) = G_u \otimes G'_v \otimes G_t \otimes I$$

$$\partial_t G(I) = G_u \otimes G_v \otimes G'_t \otimes I$$

where $\otimes$ denotes convolution, G_u is a $1 \times k$ vector of Gaussian weights, G_v is a $k \times 1$ vector of Gaussian weights, and G_t is a $1 \times k$ vector of Gaussian weights, and is applied in the third dimension over a set of k images, centered at image I_t. The G's are derivatives of the appropriate Gs.

The Laplacian is decomposable (as developed by Huertas and Medioni [10] and Marimont—who developed this independently, and did it for higher dimensions) and can be computed as the sum of the second partial directional derivatives:

$$\begin{aligned} L(G(I)) &= \partial_u^2 G(I) + \partial_v^2 G(I) + \partial_t^2 G(I) \\ &= G''_u \otimes G_v \otimes G_t \otimes I \\ &\quad + G_u \otimes G''_v \otimes G_t \otimes I \\ &\quad + G_u \otimes G_v \otimes G''_t \otimes I \end{aligned}$$

where the G''s are second derivatives of the appropriate Gs.

We can use separability to recompose these convolutions economically as follows:

$$G_u(I) = G_u \otimes I \tag{1}$$

$$G_v(I) = G_v \otimes I \tag{2}$$

$$G_{uv}(I) = G_u \otimes G_v(I) = G_v \otimes G_u(I) \tag{3}$$

$$G_{ut}(I) = G_u \otimes G_t(I) = G_t \otimes G_u(I) \quad (4)$$

$$G_{vt}(I) = G_v \otimes G_t(I) = G_t \otimes G_v(I) \quad (5)$$

$$\underline{\partial_u G(I)} = G'_u \otimes G_v \otimes G_t \otimes I$$
$$= \underline{G'_u \otimes G_{vt}(I)} \quad (6)$$

$$\underline{\partial_v G(I)} = G_u \otimes G'_v \otimes G_t \otimes I$$
$$= \underline{G'_v \otimes G_{ut}(I)} \quad (7)$$

$$\underline{\partial_t G(I)} = G_u \otimes G_v \otimes G'_t \otimes I$$
$$= \underline{G'_t \otimes G_{uv}(I)} \quad (8)$$

$$\partial_u^2 G(I) = G''_u \otimes G_v \otimes G_t \otimes I$$
$$= G''_u \otimes G_{vt}(I) \quad (9)$$

$$\partial_v^2 G(I) = G_u \otimes G''_v \otimes G_t \otimes I$$
$$= G''_v \otimes G_{ut}(I) \quad (10)$$

$$\partial_t^2 G(I) = G_u \otimes G_v \otimes G''_t \otimes I$$
$$= G''_t \otimes G_{uv}(I) \quad (11)$$

$$\underline{L(G(I))} = \underline{\partial_u^2 G(I) + \partial_v^2 G(I) + \partial_t^2 G(I)}$$

giving us eleven one-dimensional convolutions (as numbered) for the full set of required gradient and Laplacian three-dimensional convolutions (those underlined).

In our experimental development we precompute these derivative images for a sequence by loading in all images at once and doing a massive, optimized set of convolutions. Another version of the processor, meant for true sequential operation, maintains a pipeline of k images, centered on the *current* frame t, carrying out the appropriate U and V convolutions on the newest image in the pipeline, and appropriate T convolutions on the *current* centered image t. The Weaving Wall, at frame t, requires the gradients for frame t and the Laplacians for both frame t and frame $t-1$.

2.7 Propagating Sequence Constraints

Returning for a moment to our sequence analysis work, we see that an indexing strategy similar to that discussed in section 2.1 can be used to turn a complex operation on the spatiotemporal surface into a fairly simple operation when carried out in parallel locally at the facet level. The task is the tracking of features between frames; it is carried out locally by intersecting a pencil of planes with the evolving spatiotemporal surfaces and propagating tracker information along these paths (see the companion paper [2] for a discussion of this tracking process). Figure 11a shows a set of tracker paths on the local surface for the signature case 267_8 shown in figure 7. Tracker information is passed from the trailing to the leading edge of the patch along the lines drawn, each, in effect, tracking a feature through time on the spatiotemporal surface. Here, no folding of the signature is possible, and all 256 cases must be handled explicitly. Figure 11b shows the situation for the signature case 266_8, dealing with two surface patches.

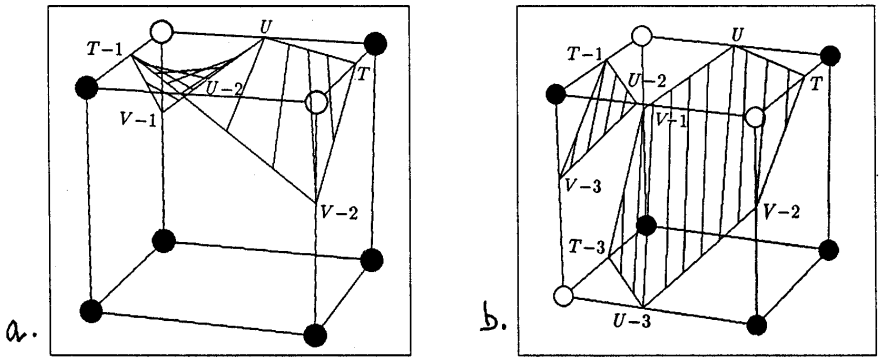

Fig. 11. Epipolar intersections (a) 267_8, (b) 266_8.

One very important point to note when considering real-time implementation of this tracking mechanism is that *none* of the above intersections would be necessary if the data could be acquired by a sensor having a geometry correct for the task. A spherical sensor, providing this geometry, would remove the conflict between the space of the *sampling* and the epipolar space required for the *analysis,* resulting in surface construction directly in epipolar space. A further advantage of this sampling space is that our *tracking* resolution would be matched to the *sampling* resolution. With the intersection process described here, the voxel at the epipole (focus of expansion, or FOE) will be multiply-interpolated to provide observations for every epipolar plane that passes through it (*all* the planes do!)[4] while voxels more orthogonal to the path may only be observed on single epipolar planes. This variation in effective resampling density leads to nonuniformity in the treatment of pixel resolution, and enormous nonuniformity in the computation required at each voxel. The correct sensor would simplify the tracker, and, by making the operations at each voxel more similar, would make the process even more amenable to parallel implementation.

[4]In fact, we disable tracking in a small disc at the FOE for just this reason, in effect giving the system a blind spot there.

2.8 Rendering Surfaces

Rendering surfaces for display is generally a simpler procedure than the weaving of coherent surface descriptions, and can be seen as an adaptation of the surface patch dual representation mentioned in section 2.5. For rendering, we must tesselate the patches into triangles. Given our signature mapping technique, this is quite straightforward, even for saddle points. Figure 12a shows the triangulation produced for the case of a signature of 267_8. The order of node traversal about the patches uses a right-handed rule with the thumb pointing out of the surface (refer to figure 2). Figure 12b shows the triangulation produced for a surface patch having signature of 266_8, where two surface patches are involved.

Actual rendering is done later, as desired, using the computed facet coordinates for spatial position, and the facet gradients as vertex normals. Triangle shading is based on bilinear interpolation of these gradients. Since disjoint surfaces are represented as distinct objects, color coding in the display and independent manipulation of the surfaces is readily attainable.

2.9 Considerations for Parallel Implementation

Alterations in the assumptions underlying these processes would have to be made in producing

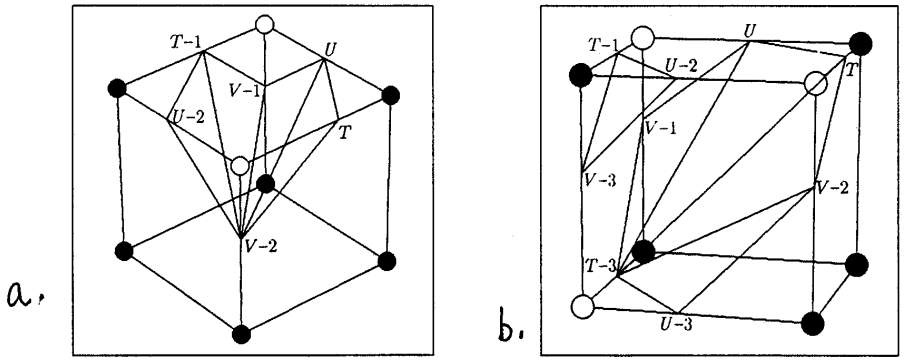

Fig. 12. Rendering patches, (a) 267_8, (b) 266_8.

code for a parallel implementation, but these would be minor rather than radical changes. The advantages of parallel implementation are quite apparent: We could divide our processing time by a large fraction of the number of pixels in an image. Currently the weaver operates on a Symbolics 3600 at about a 1-KHz voxel rate, and this suggests that a parallel implementation might operate at hundreds of frames per second.

3 Weaving Wall Applications

Some exciting secondary applications of this research arose directly from the development of the Weaving Wall spatiotemporal surface-building process. This process was designed to satisfy our needs for spatially coherent incremental tracking over an image sequence. These characteristics make it particularly useful for other applications where coherent description of nearly continuous 3D data is sought. Most obvious among these is the construction of surface models from CT and other tomographic (e.g., magnetic resonance, ultrasound) medical data. We have also been applying it to visualization problems in studying the behavior both of images with respect to spatial resolution and of analytic functions of dimension higher than three.

3.1 Medical Tomographic Data

Although we developed the algorithm for spatiotemporal analysis, we have applied it to surface reconstruction from CT slice data—data that has no temporal component, but is totally spatial. In this we use tissue density rather than Laplacian values, maintaining the 3D gradients for surface normal calculations. Figure 13 shows the evolution of surfaces judged to be 'bone' in a 70×30 window of a 52-image CT data set. You can see the incremental nature of the surface development.

Figure 14 shows rear and front stereo-pair displays of the spine depicted in figure 13. Figure 15 shows a stereo pair of the bone from the full data set of which the spine in figure 14 is a part. The tomographic images used in producing this figure were of size 256×256 pixels, and covered the head area from the base of the chin to the upper teeth.

Figure 16 shows front and side views of the skin from another CT case study. This data set of 46 slices covers the area from the neck to almost the top of the skull. Each slice is roughly 120×140 pixels, with interslice spacing about five times the slice resolution. The slices are more narrow than their separation, and this leads to a rather chunky appearance vertically, particularly noticeable at the ear. The patient had been undergoing cranial reconstruction, and slice sampling was increased in the area of the orbits. It would appear from the horizontal band there that the patient moved slightly as the scan adjustments were being made. There was also motion in the area of the jaw, and X-ray reflections from metal teeth fillings. Figure 17 shows several stereo views (top, side, and front) of the denser bone tissue.

Although the surface display aspects of this technique are evidently quite worthwhile, bear in mind that the primary representation is a surface model, with all the connectivity appropriate for full model-based computation (e.g., finite element analysis, symmetry mappings, elastic deformation operations, etc.). Figure 18 indicates some of the potential of this representation: it shows a simulation of jaw kinematics for the tissue displayed in figure 15. Although this particular display does not mirror the true behavior of the temporomandibular joint, a little more effort could have made this the case. One of our intentions in this research is to provide the capability for defining such kinematic relationships, and, in fact, performing any manipulation on the representation, including simulation of surgical operations.

3.2 Geometry from Surfaces

Surface descriptions such as these have an important role in medical imaging applications: They provide a *manipulable* representation of the 3D data geometry. Simple volume display devices such as the Pixar II [11] and the Voxel Processor of Goldwasser et al. [12] provide important diagnostic information, but their utility is limited to viewing, and there is no possibility of geometric

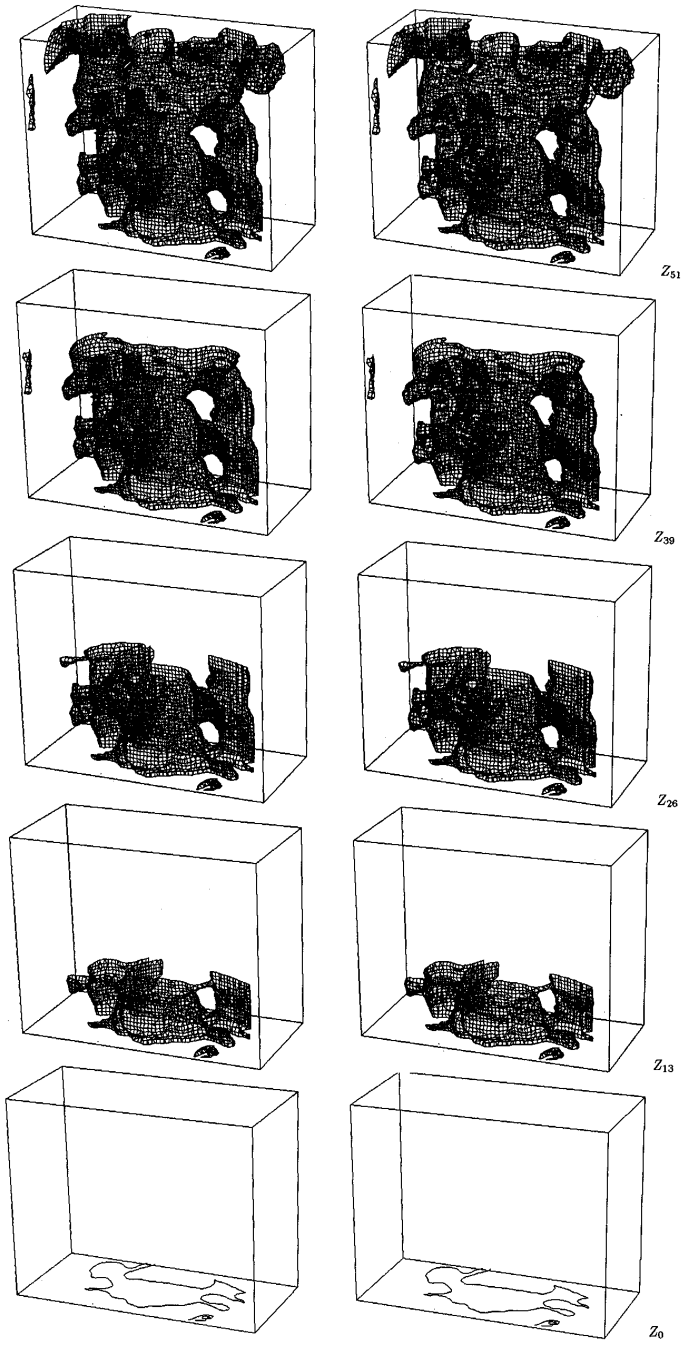

Fig. 13. Slice evolution of spine.

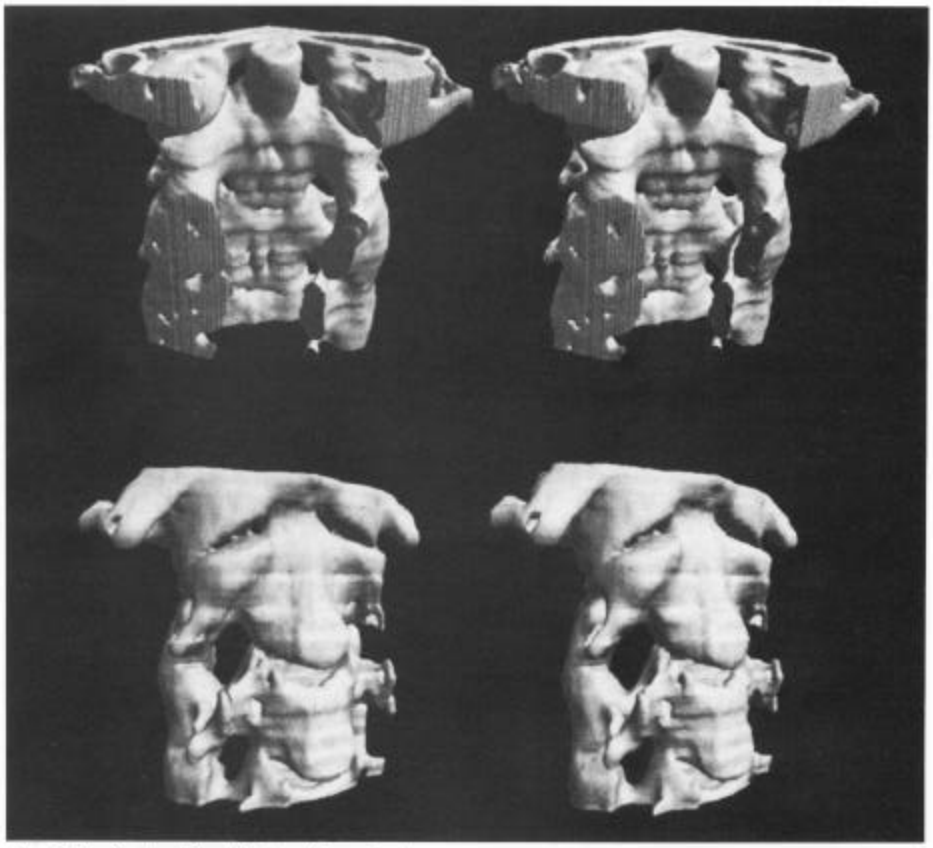

Fig. 14. Rear (top) and front (bottom) views of spine.

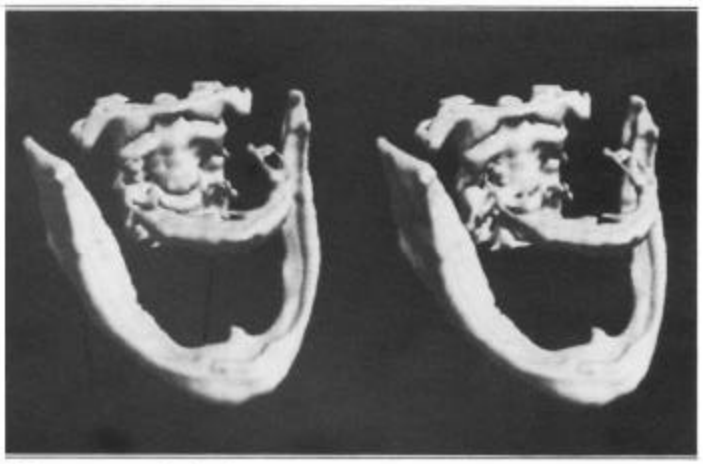

Fig. 15. Jaw, upper teeth, and spine.

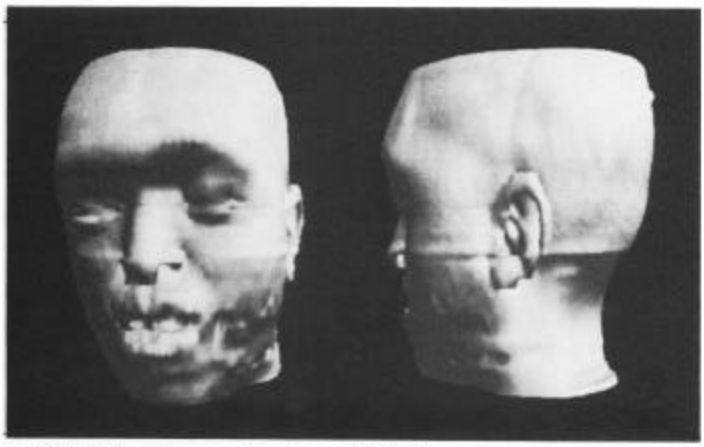

Fig. 16. Soft-tissue reconstruction—front and side views.

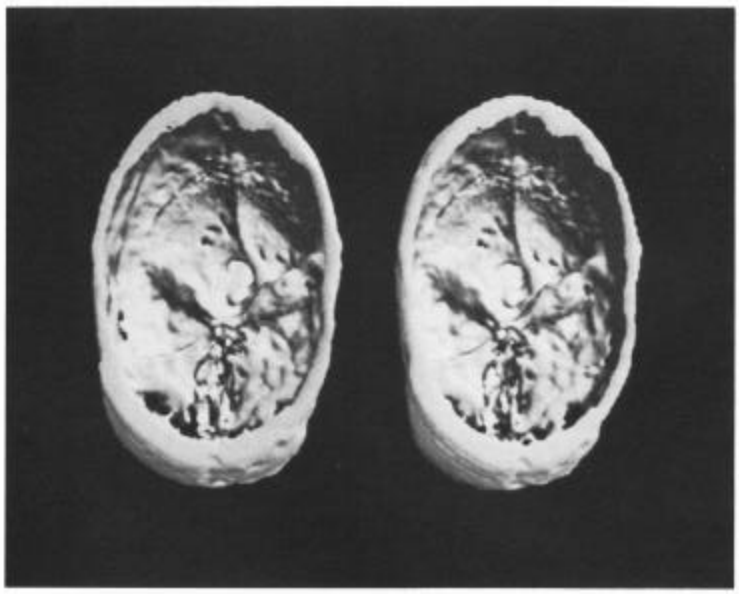

Fig. 17. Skull reconstruction—top view (skull interior).

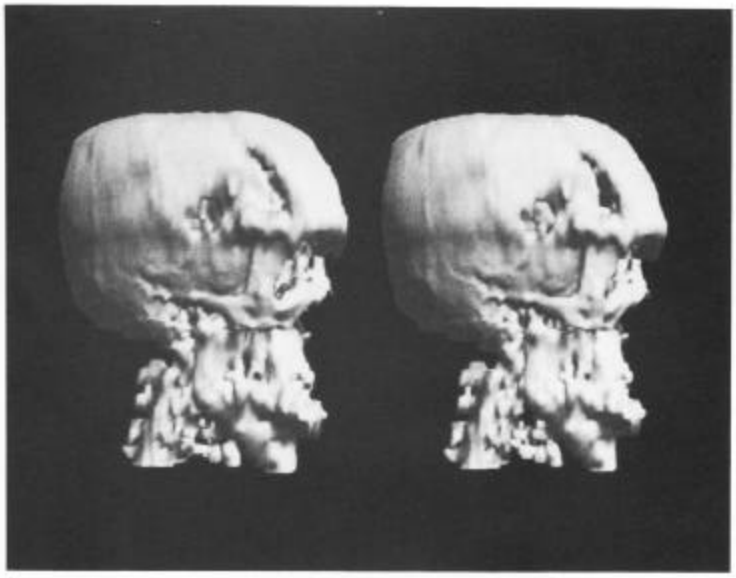

Fig. 17. Side view.

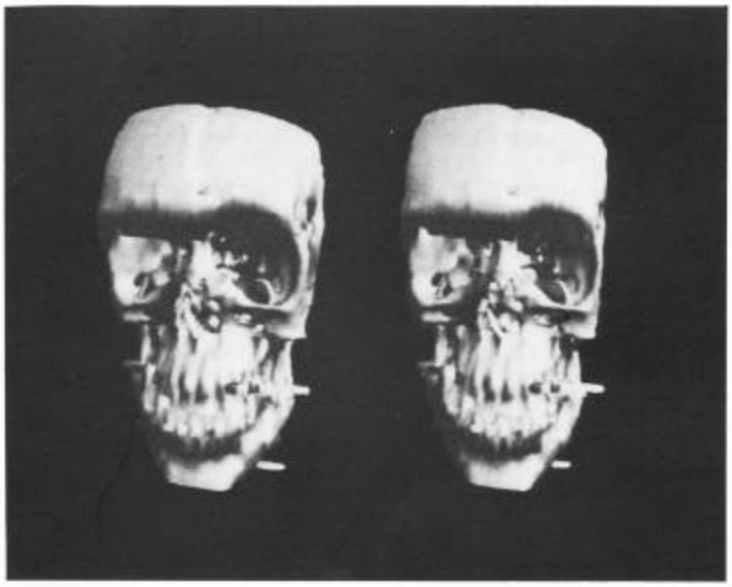

Fig. 17. Front view.

Fig. 18. Simulation of jaw kinematics.

manipulation of any of the objects present. Even simple computations, such as volume determination, are not available. If the desire is to do anything more with a three-dimensional data set than just view it, it is necessary to construct a description that is explicit in terms of the data set's geometry. This is clearly the case for surgical simulation—cutting and moving tissue under physics-based constraints. Another application of our surface construction process supports this argument for manipulable descriptions. In that, we represent the structure of chromosomes, imaged with a tomographic technique during mitosis. The computational nature of our descriptions enables analysis on the data that could not have been obtained otherwise. Given the complementary capabilities of these two approaches to data presentation, the eventual clinical scenario may well involve both: using volumetric display for initial assessment of pathology, and then constructing selected surfaces for intensive study and computational evaluation.

3.3 Images in Scale-Space

One of the long-standing issues in computer vision is the selection of a Gaussian to be the basis for feature-detection operations; in effect, selecting the scale of analysis. In researching this issue, we have done some limited experimentation with our surfaces where the third dimension is Gaussian scale (σ). Building a surface of the evolution of an image as its resolution varies provides a vivid picture of how Witkin's 1-D scale-space studies [13] can extend to 2D images. Figure 21 shows a surface constructed at the 3D Laplacian zeros obtained by applying a battery of increasingly larger Gaussians to the image of figure 19. The scale dimension holds eight images, each differing in σ by 0.5 from the one before. The left of figure 20 shows the zero crossings of the smallest Gaussian, and the right shows the zero crossings of the largest Gaussian—these are the extremes of which the surfaces in figure 21 represent the continuum. The most stable representation of a feature in this space may be at that part of its evolution exhibiting minimum spatial velocity with respect to $\Delta\sigma$. The connected scale-space surface makes this stability explicit.

In the near term we will be modifying the Weaving Wall to produce four-dimensional surfaces, where the first three dimensions are the spatial and temporal as before, and the fourth is Gaussian scale. Our intention is to use the most stable representation of a feature as its instantiation to be tracked. The linear estimators will then use these more appropriate σ values in determining observation weights and in estimating the resulting spatial precisions. In effect, tracking will be occurring at all scales at once. In other applications, such as the processing of magnetic resonance images, we will be attempting to use the stability of a contour in this scale dimension as a measure to combine with gradient strength in estimating both contour scale and significance.

Another motivation for four-dimensional surface reconstruction, still lying ahead in our work,

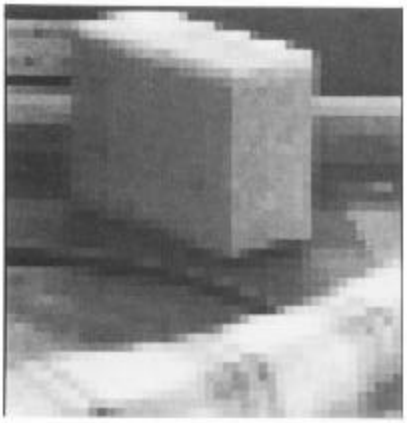

Fig. 19. First image in scale-space hierarchy.

is its applicability to describing the evolution of three-dimensional (rather than two-dimensional) events over time. Rather than having a fourth dimension being *scale*, we could make it *time*, as we have in our tracking work. Our first application of this will be in a collaborative effort, we are now beginning, to represent the time-dependent geometry of materials as they fracture. An equally exciting use lies in capturing the temporal relationships in 3D tomographic data—building dynamic models of, for example, beating hearts or rotating vertebrae.

3.4 Analytic Functions

The Weaving Wall has proved useful in object representation studies. Our research group ex-

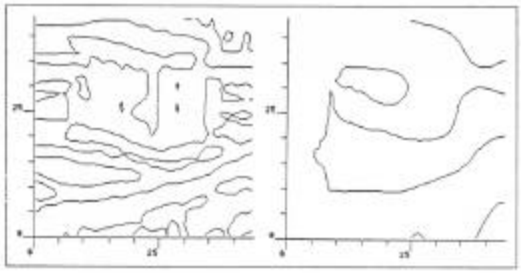

Fig. 20. First and eighth zeros.

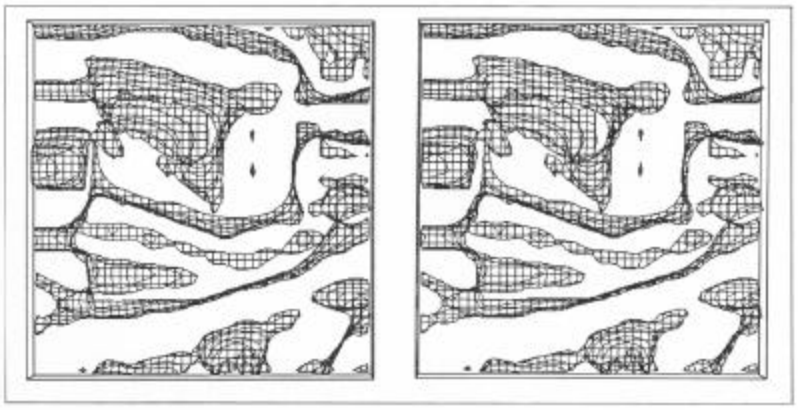

Fig. 21. Scale-Space surface.

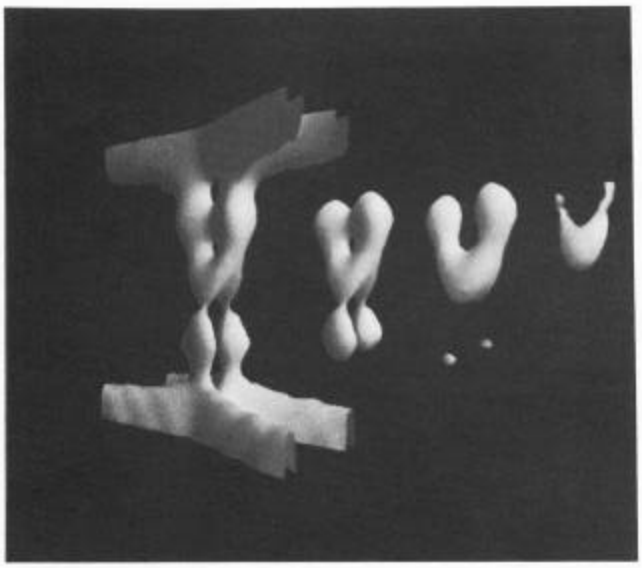

Fig. 22. Four 3D projections through 4D surface.

plores issues of representation for object modeling, and has developed a representational facility based on superquadrics [14]. Hanson [15] has developed a hyperquadric generalization permitting the description of shapes with arbitrary polyhedral bounds; superquadrics, in contrast, permit only shapes having the three orthogonal Cartesian bounds. Use of these hyperquadrics enriches the modeling by yielding a superset of the superquadric primitives. Experimenting with these higher-dimension objects is complicated by the difficulty in visualizing them. To facilitate this, we display sequences of three-dimensional projections of these n-dimensional objects ($n > 3$), using the Weaving Wall in this rendering. Figure 22 shows a selection of such projections through a four-dimensional surface. Viewing such frames as a sequence gives us insight into the structure of these objects. Banchoff [16] discusses similar issues in assessing the structure of higher-dimension functions.

Other possible applications of the surface-building process employed here include representation of surfaces from such two-dimensional sensing domains as ultrasound and geology, display of fracture analysis data (2D over time), analysis of tomographic data from nondestructive testing, and the colorization of black-and-white film. In general, this process can be used in any application where a description is desired of the evolution of a two-dimensional pattern varying gradually in a third dimension—be that dimension time, space, viewing position, resolution, or any other.

4 Conclusions

The Weaving Wall was a necessary development for our continuing research in sequence analysis. The structures it produces provide for our current needs in tracking features on the spatiotemporal surface, and seem also to be most appropriate for camera solving and for our intended extension of the analysis to nonlinear camera paths. These latter points are discussed in the companion paper [2]. The manner in which we implemented the surface builder has enabled us to introduce many

complex geometric three-dimensional image operations at a very local level, and to maintain important surface properties at minimal cost. The wide range of current applications, including 3D medical imaging, display of higher-dimensioned analytic functions, and computations on the scale-space surface, suggests that we have captured, in our surface-building process, a powerful and general tool. We will be developing it further in these and other directions.

Acknowledgements

David Marimont, Lynn Quam, and Bob Bolles have been crucial in the development of this work. Alex Pentland, with his Supersketch modeling and graphics system, was very helpful in tasks of generating simulated data and rendering surface images. Andy Hanson provided the data and rationale for the hyperquadric analytic surface displays. The medical CT data is courtesy of Dr. C. Cutting, New York University, and CEMAX Corporation, Santa Clara, California. This research was supported by DARPA Contracts MDA 903-86-C-0084 and DACA 76-85-C-0004, and SRI International.

References

1. R.C. Bolles, H.H. Baker, and D.H. Marimont, "Epipolar-plane image analysis: An approach to determining structure from motion," *Intern. J. Computer Vision* 1:7-55, June 1987.
2. H.H. Baker and R.C. Bolles, "Generalizing epipolar-plane image analysis on the spatiotemporal surface," *Intern. J. Computer Vision,* (this issue), December 1988.
3. E. Artzy, G. Frieder, G.T. Herman, "The Theory, design, implementation, and evaluation of a three-dimensional surface detection algorithm," *Computer Graphics and Image Processing* 15:1-24, January 1981.
4. G. Wyvill, C. McPheeters, and B. Wyvill, "Data structure for soft objects," *Visual Computer* 2:227-234, 1986.
5. P.T. Sander and S.W. Zucker, "Tracing surfaces for surfacing traces," *1st Intern. Conf. Computer Vision,* London, pp. 241-249, June 1987.
6. S.W. Zucker and R.A. Hummel, "A three-dimensional edge operator," *IEEE Trans. PAMI* 3:324-331, May 1981.
7. D.H. Marimont, "Segmentation in Acronym," *Proc. DARPA Image Understanding Workshop,* Stanford, CA, pp. 223-229, September 1982.
8. W.E. Lorensen and H.E. Cline, "Marching Cubes: A high resolution 3D surface construction algorithm," *Computer Graphics* 21: 163-169, July 1987.
9. J. Canny, "A computational approach to edge detection," *IEEE Trans.* PAMI 8:679-698, November 1986.
10. A. Huertas and G. Medioni, "Detection of intensity changes with subpixel accuracy using laplacian-gaussian masks," *IEEE Trans.* PAMI 8:651-664, September 1986.
11. "Pixar unveils new entry-level image computer," *Datamation,* pp. 85-88, February 1988.
12. S.M. Goldwasser and R.A. Reynolds, "Real-time display and manipulation of 3-D medical objects: The voxel processor architecture," *Computer Vision, Graphics, and Image Processing* 39:1-27, July 1987.
13. A.P. Witkin, "Scale space filtering," *Proc. 8th Intern. Joint Conf. Artif. Intell.,* Karlsruhe, West Germany, pp. 1019-1021, August 1983.
14. A.P. Pentland, "Perceptual organization and the representation of natural form," *Artificial Intelligence,* 28:293-331, 1986.
15. A.J. Hanson, "Hyperquadrics: Smoothly deformable shapes with convex polyhedral bounds," *Computer Vision, Graphics, and Image Processing,* 44:191-210, September 1988.
16. T.F. Banchoff, "Visualizing two-dimensional phenomena in four-dimensional space: A computer graphics approach." In *Statistical Image Processing and Computer Graphics,* E. Wegman and D. Priest (ed.), Marcel Dekker, New York, 1986.

International Journal of Computer Vision, 3, 5 (1989)

Editorial

This issue is our second special issue devoted to the work of a particular group of vision researchers, one of the unique forums that the International Journal of Computer Vision provides. This time we feature the work of the vision group in the AI center at SRI International. The SRI's vision group has produced pioneering work in a number of areas including photointerpretation, three-dimensional model-based object recognition, mobile robot vision, and image sequence analysis.

We asked Martin Fischler, the leader of the group, to solicit a number of original research papers and submit them to the journal. We then subjected the papers to our normal rigorous review process. On the basis of the received reviews, we made a final selection of the four papers presented in this issue, together with an overview of SRI's vision work by Fischler.

We believe that these special issues devoted to particular vision research groups provide a unique opportunity to showcase a coherent view of their philosophy, approaches, and achievements. We welcome your opinion on these issues, and new proposals.

Takeo Kanade
Computer Science Department
Carnegie Mellon University

Rodney Brooks
Artificial Intelligence Laboratory
Massachusetts Institute of Technology

International Journal of Computer Vision

Volume 3, No. 1, 1989