Skeletonization

L. Gursoz, A. Sudhalkar, F. Prinz

EDRC 24-59-91

Skeletonization

Levent Gursoz Atul Sudhalkar Fritz Prinz

May 20, 1991

This report describes preliminary work in the automatic generation of the Medial Axis Transform from discretized objects. The method is similar in spirit to grassfire algorithms described by various authors in the past 20 years in the 2D domain. However, the ideas presented here work in both 2D and 3D, and appear to be easily extensible to n-dimensions. The algorithm presented here is also a significant improvement over previous grassfire algorithms in that it directly yields a skeleton of lower dimensionality than the initial model.

1 Introduction

In design, a means of hiding irrelevant details is often useful. Shape being an important aspect of design, such an abstraction process could also be employed in describing the shape of the designed object. This work describes one means of abstracting shape information.

One's perspective determines what features of a model are interesting. This is especially true when crossing domains: what is relevant information to a machinist need not be so to a blacksmith. Therefore, it is not to be expected that this work can be all things to all people. The assumptions made in defining the abstraction process will limit the domains where the process is useful, as will be seen.

The abstraction process in geometry will always reduce the dimensionality of a model. For obvious reasons, such an abstraction has been called a *skeletal* or *thinned* object in the literature, and these terms are also used here. The skeleton of an entity is similar to the *Medial Axis Transform* (MAT) [Blum 67], [Chern 89]. This results in a reduction of the model to a representative set of skeletal surfaces and/or axes from which shape information is readily elicited. This has strong implications for several design-related tasks, such as shape feature recognition, the automatic generation of finite-element meshes, etc.

For a two-dimensional object, the medial axes are the locus of the centers of maximal discs which can be fit into the object. Figure 1 shows how medial axes are determined by fitting maximal discs into a polygon. A radius function gives the distance from each point on the medial axes to the original object boundary. Figure 2 shows a polygon and its medial axes.

The MAT definition is similarly extended to three-dimensional objects. The MAT then consists of a set of medial surfaces (instead of axes) and a radius function. The medial surfaces are identified from the boundary of the solid using a set of maximal spheres (instead of discs). Figure 3 shows a rectangular plate, and it's MAT, which consists of four triangles, eight trapezia, and a rectangle.

2 Grassfire Thinning

The process of skeletonization (sic) presented here is related to an approach called Grassfire Thinning, which works like a prairie fire [Blum 67] [Xia 89], Consider a patch of grass with its entire boundary set on fire simultaneously. All parts of the flame front propagate inward at the same speed, burning away the patch of grass. Points at which the flame fronts meet, called quench points (after [Xia 89]), yield a skeletal form of the object.

Several researchers have described grassfire algorithms for the 2D domain, motivated mainly by the needs of character recognition in computer vision. The algorithm of [Xia 89] was developed toward these ends. All such algorithms are described on a *discretized* model. Thus, in 2D the model consists of quanta called *pixels* (essentially squares of fixed size), while in 3D the model consists of cubes, called *voxels*[1]. All grassfire algorithms begin with such a discrete model, and progressively strip away outer layers of such discrete elements. The outermost layer is called the flame-front, from the analogy to a prairie fire. Wherever two flame-fronts meet, those voxels are considered part of the skeleton.

The objective of the thinning process has traditionally been set at reducing the discrete object to a set of elements (pixels/voxels) which have unit thickness. An additional consideration has been the preservation of components of the skeleton, so that they do not get shortened.

One weakness of traditional grassfire algorithms has been the somewhat fuzzy nature of the above terms. In 2D, it suffices to exhaustively enumerate templates covering important configurations of a pixel's neighborhood. However, this is practically impossible in three-dimensions. Recently, attempts have been made to formalize the process [Jang 90], but this is focussed only on the 2D problem.

There is another fundamental shortcoming in the traditional algorithms, at least from the perspective of the design community. These algorithms consider the job complete when pixels/voxels are a single layer thick. Note, however, that these are still pixels/voxels. In other words, there is no reduction in the dimension in going from the original object to it's skeleton.

The method to be described here yields skeletons which are always of a lower dimensionality than the original model. Indeed, this is the criterion for terminating: while any of the original elements (pixels or voxels) remain, the process cannot be complete.

3 Formalization of the Discrete World

The discretization process divides a model into a collection of uniform-sized cubical elements, called *voxels.* Since all thinning operations will occur on the discretized model, it is necessary to put forth a clear formalism to be used here.

A discrete model is a *chunked* version of a solid model. Concepts such as connectivity, genus, etc. all apply to it. However, difficulties crop up as voxels are deleted in the process of skeletonization. To illustrate, consider the rectangular (2D) object in figure 5a. Skeletonization might yield an object as in figure 5b. However, if the original rectangle were to be *tilted,* as shown

[1]This term is derived via a tortuous path: 2D discrete representations consist of picture elements, or *pixels.* The 3D version, then, consists of "volumetric elements", or voxels.

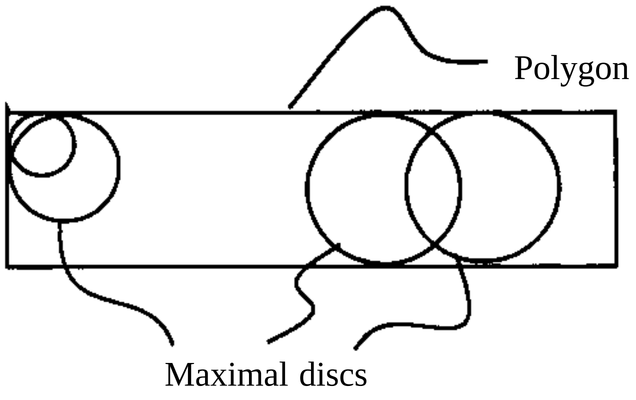


Figure 1: Definition of 2D MAT

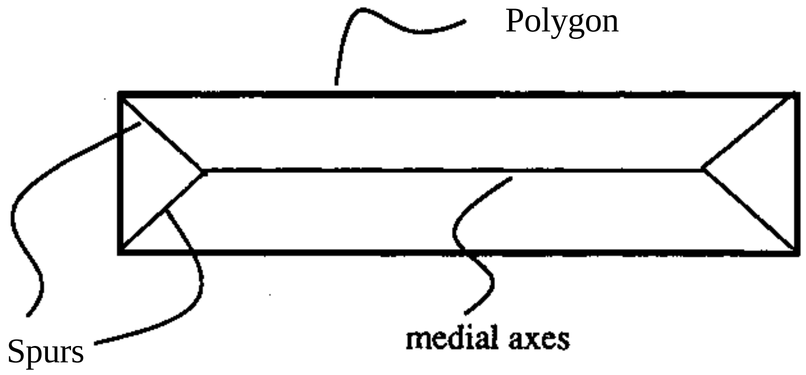


Figure 2: Example of MAT for a simple polygon

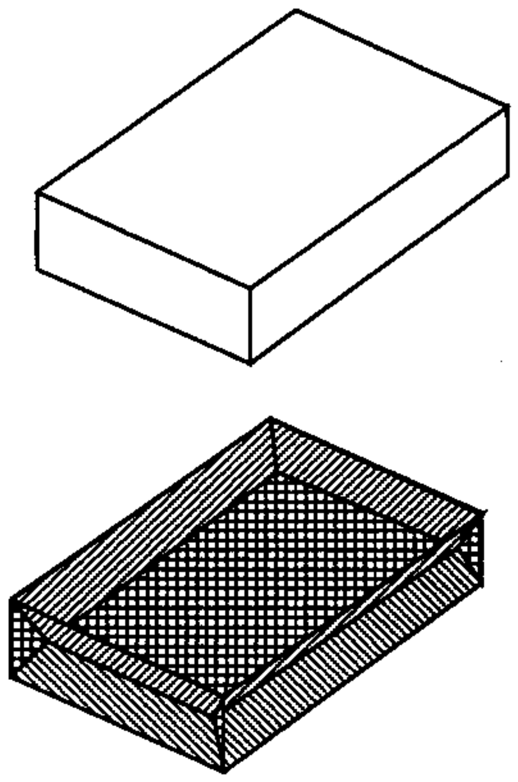


Figure 3: A rectangular plate and the 3D MAT

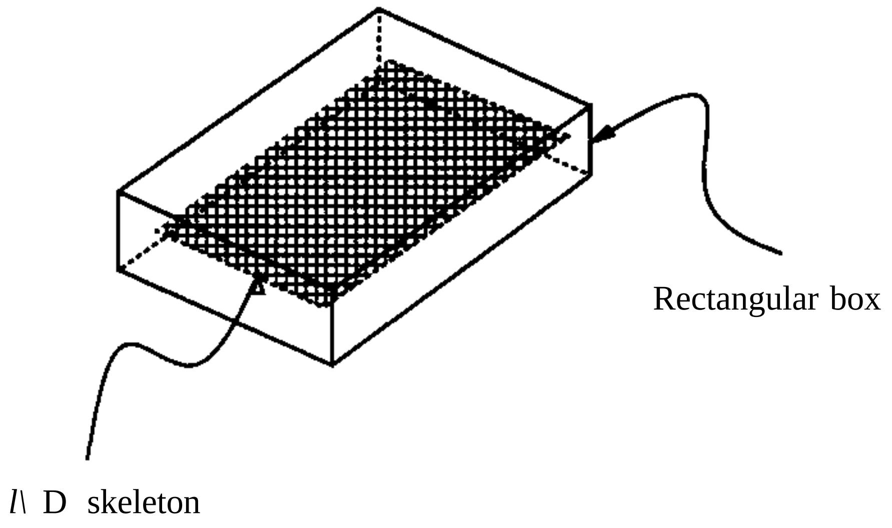
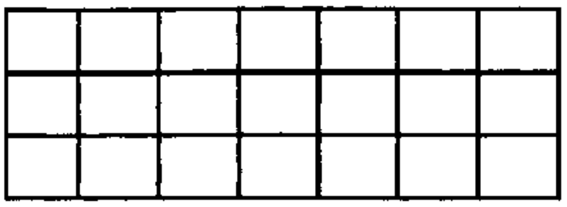


Figure 4: A rectangular plate and it's 2.5D skeleton.


(a)

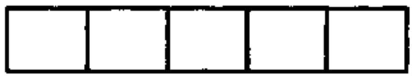
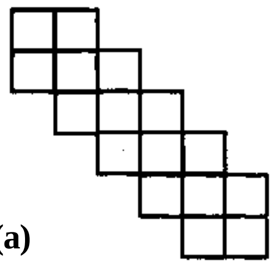

(b)

Figure 5: End view of a plate and it's skeleton


(a)

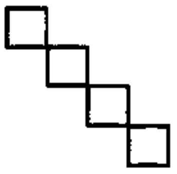

(b)

Figure 6: End view of a plate at 45 degrees to the axis, and it's skeleton

in figure 6a, the skeleton would look like figure 6b. Here it is no longer clear whether the skeleton is a line, or a set of disconnected of points.

Each voxel has three types of neighbors: those that share a face with it, those that share an edge but no face, and those that share a vertex alone. Figure 7 illustrates each of these. Each voxel is thus has six face-neighbors, twelve edge neighbors, and eight vertex neighbors — twenty-six in all. Correspondingly, three types of connectivity can be defined for three-dimensional discrete objects: face connectivity, edge connectivity, and vertex connectivity.

A consistent set of conventions in this regard [Tsao 81] is to consider the *object* to have all three types of connectivity, and the *complement* to have only face connectivity [Tsao 81].

Two distance metrics can be defined in the discrete domain: *block* distance, and *chess-board* distance [Xia 89]. Block distance is the *maximum* of the differences between the coordinates of two points, while the chess-board distance is the *sum* of these.

4 Description of the Process (2D)

As stated earlier, the goal of skeletonization (at least for use in engineering design/analysis) should be to produce an entity of lower dimensionality than the original model. Consider Figure 8. Part (b) of that figure is a skeletal form of the object in part (a). Note that this is one of the criteria

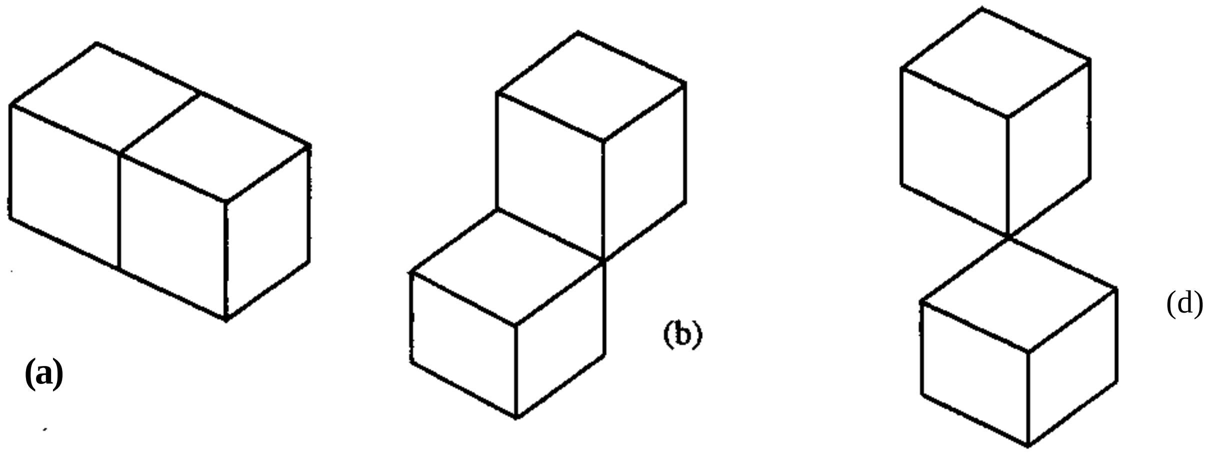


Figure 7: Neighbor types: (a) Face neighbors; (b) Edge neighbors; (c) Vertex neighbors

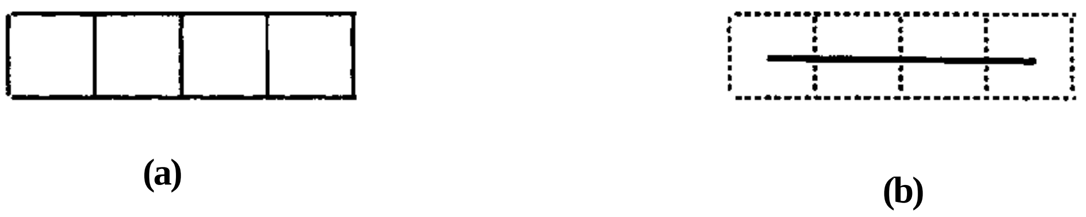


Figure 8: (a) An object, and (b) It's skeleton

used in termination of traditional grassfire algorithms: it should be possible to define a "previous" and "next" pixel, so that a line can be drawn through the pixel at hand. This also holds for the object in Figure 9, although the object is now shown to be tilted to the principal axis.

The key question is, if the given model is as shown in Figure 10 (a), how can one get to the skeleton in part (b) of the figure, and after that to part (c) ? Traditional grassfire algorithms do this by identifying the so-called *flame-front,* and progressively stripping it away until only an inner core remains. This works when the object is an odd number of layers thick, but difficulties arise when the object is an even number of layers thick, as shown in Figure 11.

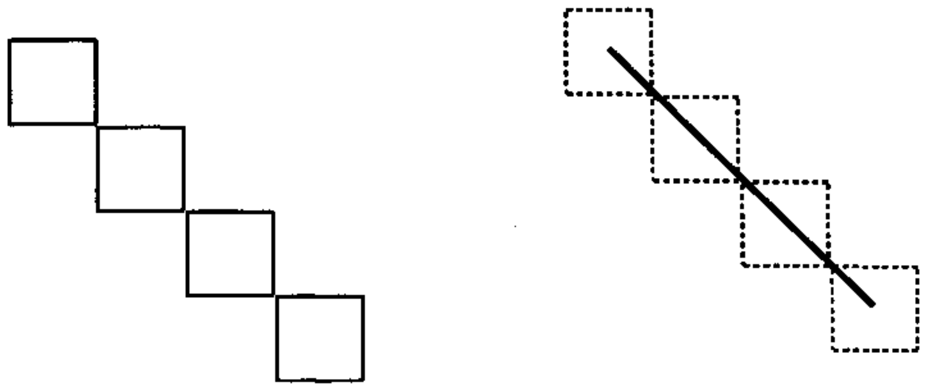
The algorithm being presented here overcomes this difficulty. In a sense, this algorithm removes only *half* a voxel at a time, as will be seen. It works by examining the connectivity of each pixel with it's neighbors. One takes the *dual* interpretation of a pixel as just a point in 2-space ("dual" as in graph theory), located at the center of the pixel. Edges (dual edges) are then made by joining the neighbors to the current pixel. In making these edges, the following guidelines are followed:

1. An edge is always made between a pixel and any edge-neighbors.

2. An edge is made between a pixel and any vertex-neighbor if and only if there is no other pixel which is a face neighbor to both.

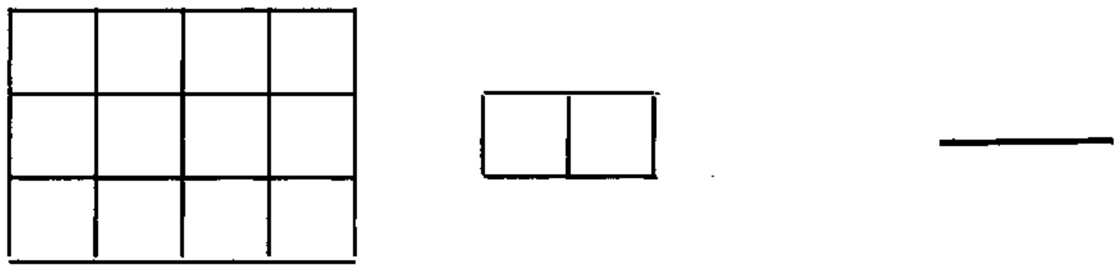
Figure 12 illustrates the process. Note that in part (a) of the figure, edges of the dual graph are made between edge-neighbors, while in part (b), an edge is shown between two vertex neighbors. Part (c) further clarifies rule 2 above.

There is yet a third rule for making edges, which will be explained further down.



(a) (b)

Figure 9: (a) An object, and (b) It's skeleton



(a) (b) (c)

Figure 10: (a) An object, and (b) It's skeleton

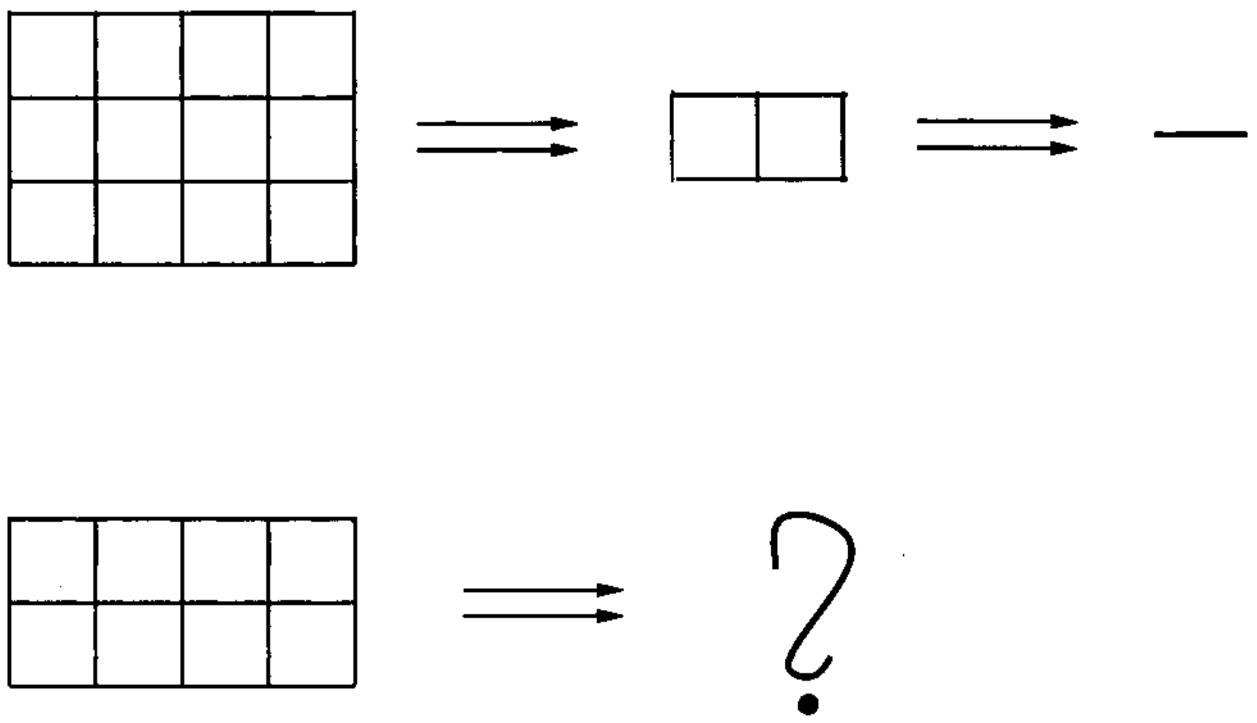


Figure 11:

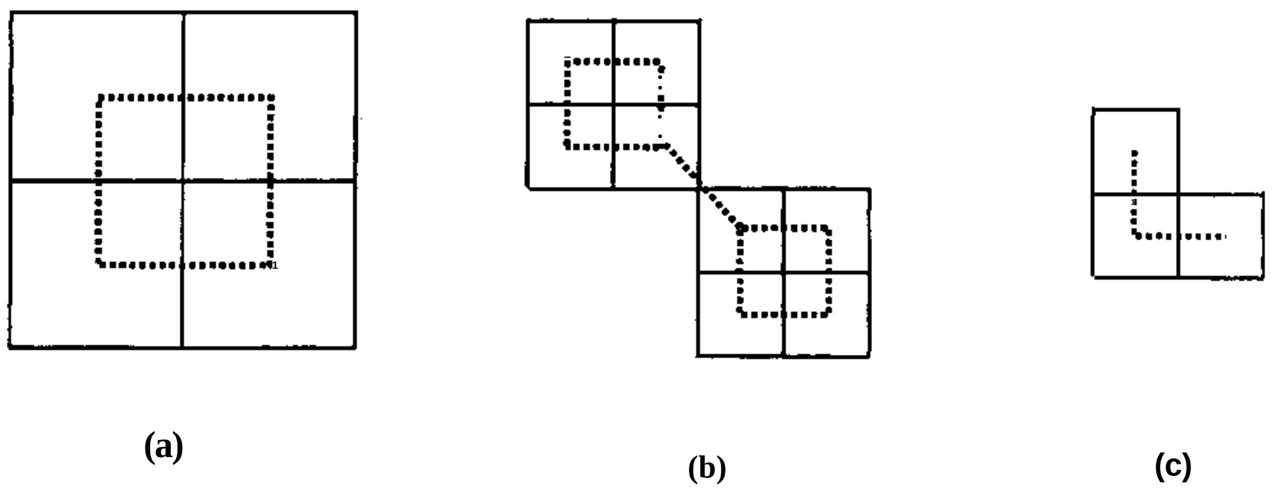


Figure 12:

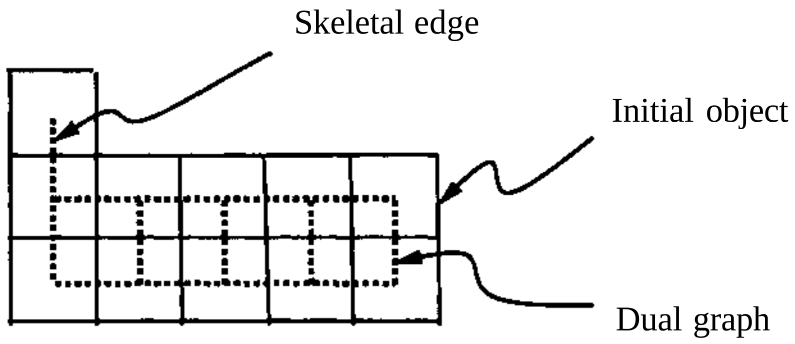


Figure 13:

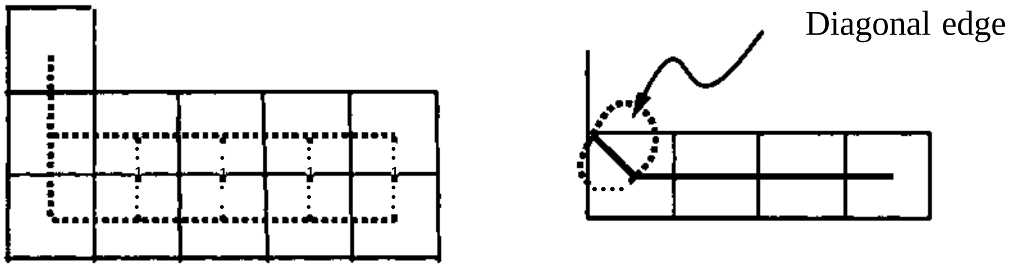


Figure 14:

Once these edges are made, faces are identified from cycles of edges. Note that this is possible since the entire graph is necessarily embeddable on the plane. We align a new grid of pixels such that the vertices of the dual graph are coincident with the vertices of the pixels. Then, occupied (or partially occupied) pixels are treated as full, and this serves as the starting point for the next stage of the same process. Note that the object will have shrunk by one pixel.

When constructing the dual graph as described above, some edges might not participate in any cycles, and thus will not lie on the boundary of any pixel in the next stage. Such edges are *skeletal* — i.e., they are a part of the final skeleton. Figure 13 illustrates the formation of such edges. Once they are formed, skeletal edges are always carried along from stage to stage.

At intermediate stages between the initial object and the skeleton, there is likely to be a mix of skeletal edges and pixels. Special care needs to be taken to ensure that connectivity is maintained in such situations. This gives the third rule for constructing edges: when a skeletal vertex (i.e., a vertex bounding a skeletal edge) is in the boundary of a pixel, a dual edge is made between the the center of the pixel and the skeletal vertex. This is illustrated in Figure 14.

The process of thinning as described here is shown in Figure 15 in full detail, over all the stages.

5 Skeletonization in 3D

The 3D algorithm is a straightforward extension of this algorithm. There are now three levels of hierarchy among the neighbors: face-, edge-, and vertex-neighbors. The rules for edge-making are

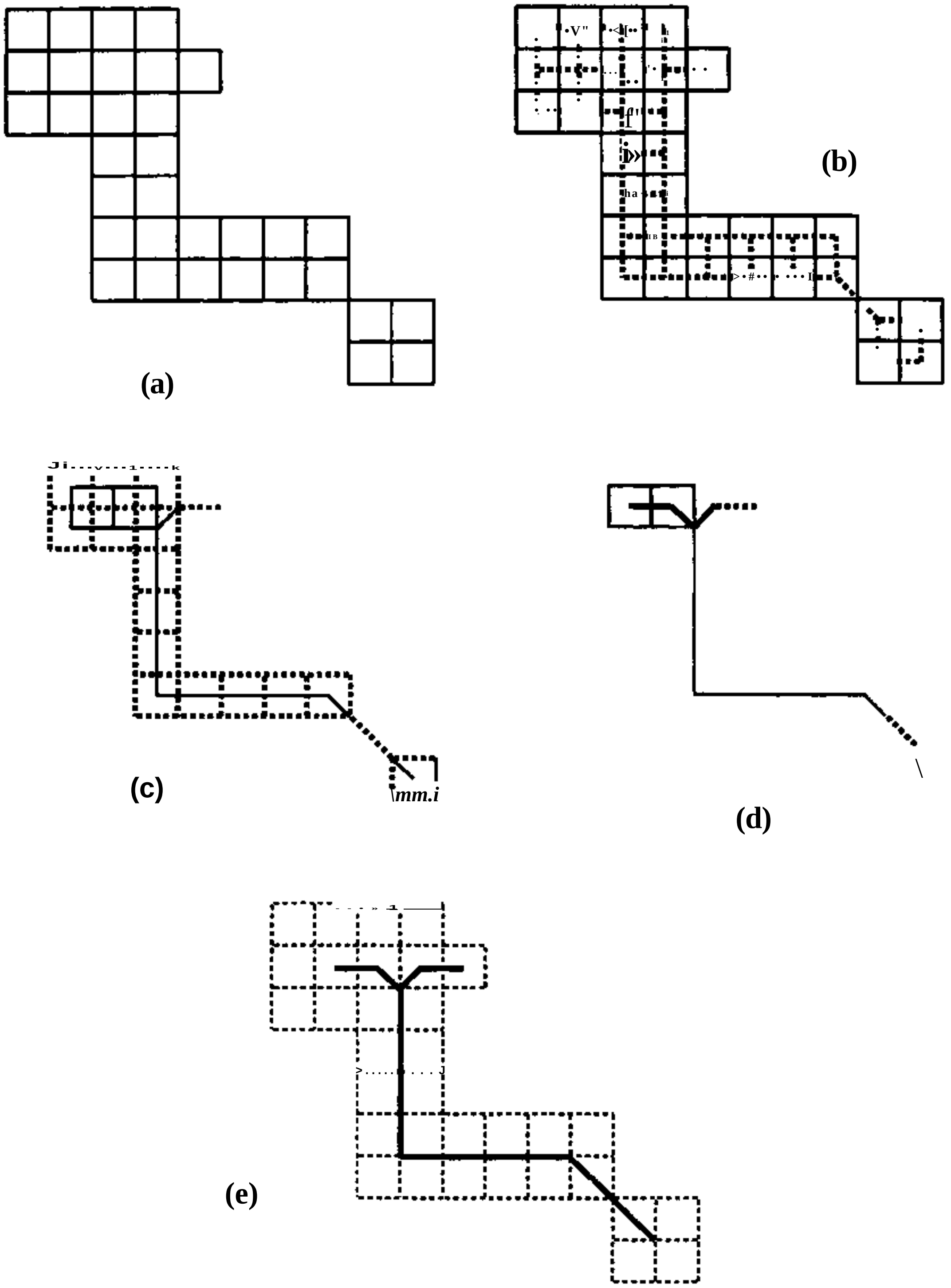


Figure 15:

similar: make edges between face-neighbors first, then if necessary between edge-neighbors, and then if necessary between vertex-neighbors. Faces are once again identified by finding cycles in the graph. Note that these faces could be squares or triangles. Once faces are identified, regions enclosed by these are found. These are then mapped onto voxels such that vertices in the dual graph coincide with vertices of the voxels.

The faces and/or edges formed in constructing the dual graph may or may not lie on the boundary of an enclosed volume. If they do not, they are considered skeletal. As in 2D, such skeletal entities are carried along from stage to stage, and care is taken to ensure connectivity at such points. As before, edges are made between the duals of voxels, and any skeletal vertices on the boundary of the voxel.

6 Summary

Skeletal abstractions being very useful to the design community, a means of generating approximate Medial Axis Transforms of arbitrary 3D solids is presented. The procedure is similar in spirit to grassfire algorithms found (mainly) in computer vision literature.

The main advantages of the proposed technique over traditional grassfire algorithms are:

1. A skeleton of reduced dimensions is guaranteed to be obtained.

2. Since the proposed algorithm can resolve down to half a voxel/pixel, symmetry with respect to the original (discrete) object is guaranteed.

One criticism of the MAT has been that it is too sensitive to perturbations in the surface topology of the original model. There are two reasons for this, described below. Perturbations in the surface of the original model can result in either concativies or convexities.

- In the case of convexities, the MAT would throw out a "spike" into the convexity.

- In the case of concavities, the medial axes/surfaces could curve to accomodate the change.

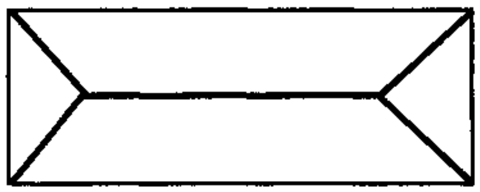
Both these situations are illustrated in Figure 16.

It is expected that the proposed approach will robustly handle both the above situations. Convexities are not an issue, since the approximate MAT generated here never contains any spikes. Concavities will not cause a problem so long as the perturbation does not show up in the discrete model. An appropriate choice of the level of discretization would allow users to fine-tune the sensitivity of the MAT to the original model.

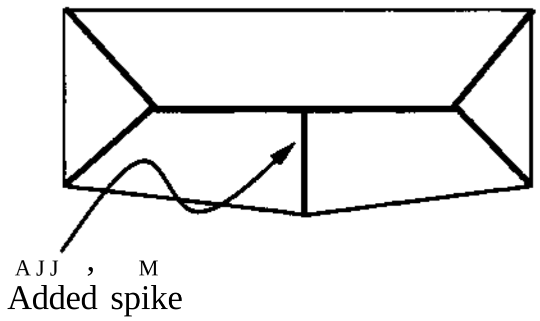
References

[Blum 67] Blum, H., "A transformation for extracting new descriptors of shape", in *Proceedings,* ***Symposium Models for Perception of Speech and Visual Form,*** **Whaten-**Dunn, W., ed., MIT Press, Cambridge, MA, 1967.

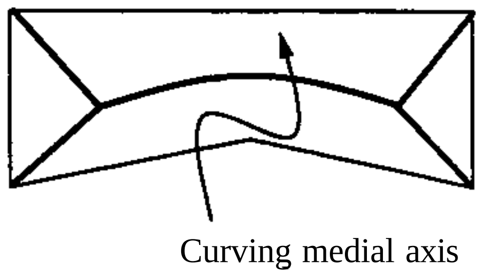
[Chern 89] Chern, J., Levent Gursoz, Fritz Prinz, and Mark Hall. "Geometric Abstractions using Medial Axis Transformation", *???,* 1989.



(a) Original model (polygon) and medial axes



(b) Added convexity, and corresponding medial axes



(c) Added concavity, and curvature of medial axis.

Figure 16: Sensitivity of MAT to variations in surface geometry of original model

[Jang 90] Jang, Ben-Kwei, and Roland T. Chin. "Analysis of Thinning Algorithms Using Mathematical Morphology", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 12,6, June 1990.

[Tsao 81] Tsao, Y.F., and K.S. Fu. "A Parallel Thinning Algorithm for 3D Pictures", *IEEE Computer Graphics and Image Processing,* 17, pp 315-331,1981.

[Xia 89] Xia, Yun. "Skeletonization Via the Realization of the Fire Front's Propagation and Extinction in Digital Binary Shapes", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 11,10, October 1989.