3D Shape Representation: Transforming Polygons into Voxels *

Stijn Oomes[1,2], Peter Snoeren[1], and Tjeerd Dijkstra[2]

[1] Nijmegen Institute for Cognition and Information, Nijmegen, The Netherlands
[2] Vision Laboratories, Ohio State University, Columbus, USA

Abstract. We developed a method that transforms a polygonal representation of an object into a voxel representation on a $3D$ regular grid. By applying scale-space theory we derived expressions for an anti-aliased voxel representation of points, lines, and triangles. The algorithm 'paints' the triangles on the voxel grid and 'fills' the resulting surface. The method can be applied in 3D (medical) imaging and shape analysis in $3 + 1$ dimensional scale-space.

1 Introduction

Objects in computer graphics are usually polyhedra; the surface of a shape is represented as a polygon mesh. The polygons are the graphics primitives that are rendered with shading algorithms. Another common representation is one in which the object is a volumetric density defined on a 3D regular grid. The points on the grid have a scalar value that corresponds to the measurement value of for instance a MRI or PET scan used in medical imaging. Methods have been developed to extract a polygonal representation from the volumetric data in order to visualize the boundary of the object, with the *marching cubes* algorithm of Lorensen and Cline (1987) as the most common technique. In this paper we will describe a method for the reverse transformation: from a polyhedral surface to a volumetric density representation, also referred to as *voxelization*. We have developed our technique as a first step in obtaining a dynamic shape hierarchy (Koenderink & van Doorn 1986). The result of our algorithm is a representation where the voxels have values 1 deep inside the object and 0 far outside, with a transitional region in between.

2 Polygons and Voxels

Polyhedral objects are usually represented as a list of vertices with their x, y and z coordinates, completed by a list of indices that point to the vertices that form one polygon. Let a orthogonal grid of $N \times N \times N$ voxels with spacing d be defined in the same space as the vertex coordinates and let $\mathbf{x}_{ijk}$ be an arbitrary point of the grid. In this paper voxels are *not* considered as

* This work was supported by the Netherlands Organization for Scientific Research. We thank Flip Phillips, Del Lindsey, and Jim Todd for criticism and discussion.

gridpoints, *nor* as cuboids but as spherical blobs with a Gaussian sensitivity profile with their centers located at the gridpoints (Figure 1). One can think of the voxels as integrating the contribution from graphics primitives in their neighborhood. The density at gridpoint $\mathbf{x}_{ijk}$ contributed by a point source at $\mathbf{x}$ is given by the Gaussian:

$$\Gamma_\sigma(\mathbf{x}-\mathbf{x}_{ijk}) = \frac{1}{\sqrt{2\pi}^3\ \sigma^3}\exp\left[-\frac{(\mathbf{x}-\mathbf{x}_{ijk})^2}{2\sigma^2}\right] \tag{1}$$

where σ is the smallest spatial scale. The spatial resolution d and metrical resolution R of the voxel representation determine σ. By assuming that the frequency distribution of voxel values is Gaussian in the Fourier domain, one can derive that $\sigma = \frac{d\sqrt{2R}}{\pi}$ [Koenderink (1984)]. For voxel values with an accuracy of 8 bits ($R = 8\ln 2$) the smallest spatial scale is close to d.

3 Voxelization

The scene is set for representing ideal geometrical entities as points, lines, and triangles on a regular grid. The format will be to take a graphics primitive in a parameterized form $\mathbf{v}(s,t,...)$ and give an expression for the voxel values. The general trick is to take the Dirac delta function $\delta[\mathbf{x}-\mathbf{v}(s,t,...)]$ and integrate over the entire parameter space. The integrated delta function is convolved with a Gaussian and normalized by dividing by the maximum value

$$I(\mathbf{x}_{ijk}) = \frac{\Gamma_\sigma \otimes \int\int \delta\ ds\ dt...}{\max[\Gamma_\sigma \otimes \int \delta]} = \frac{\int\int \exp\left[-\frac{(\mathbf{x}_{ijk}-\mathbf{v}(s,t,...))^2}{2\sigma^2}\right]\ ds\ dt...}{\sqrt{2\pi}^3\ \sigma^3\ \max[\Gamma_\sigma \otimes \int \delta]} \tag{2}$$

By changing the order of integration, a simple expression emerges that integrates a 'Gauss delta function' over the parameter space and divides it by its maximum value. The maximum value is reached for the point that is the 'centroid' of the geometrical entity. The voxel values $I(\mathbf{x}_{ijk})$ are dimensionless and the expression is independent of the dimensionality of the vector space. The method is not limited to the examples given below and can be applied to any parameterized curve, surface or volume.

Point. The simplest case is a point $\mathbf{p}$ and gives voxel values

$$I(\mathbf{x}_{ijk}) = \exp[-\frac{(\mathbf{x}_{ijk}-\mathbf{p})^2}{2\sigma^2}]. \tag{3}$$

Note that $\mathbf{p}$ does not have to coincide with a gridpoint because the representation is robust under shifts. Thus a point is represented by several non-zero voxels; this loss in resolution is compensated by a gain in robustness.

Line. A line $\mathbf{l}(s)$ is defined by its endpoints $\mathbf{p}$ and $\mathbf{q}$ as $\mathbf{l}(s) = \mathbf{p} + s(\mathbf{q} - \mathbf{p})$ with $0 \leq s \leq 1$. For convenience, we define some new variables that have the form of an inner product of two differences of vectors: $A = (\mathbf{x} - \mathbf{p})^2$, $B = (\mathbf{x} - \mathbf{p}) \cdot (\mathbf{q} - \mathbf{p})$, and $C = (\mathbf{q} - \mathbf{p})^2$. The voxel values are

$$I(\mathbf{x}_{ijk}) = \frac{\exp(\frac{B^2 - AC}{2\sigma^2 C}) \left[\operatorname{erf}(\frac{B}{\sigma\sqrt{2C}}) - \operatorname{erf}(\frac{B-C}{\sigma\sqrt{2C}}) \right]}{2 \operatorname{erf}(\frac{\sqrt{C}}{\sigma 2\sqrt{2}})} \tag{4}$$

where the denominator is calculated for $\mathbf{x} = (\mathbf{p} + \mathbf{q})/2$. This expression results in an anti-aliased line that is robust for small shifts. In figure 2 we show a line of length 16 on a grid of length 32 with $\sigma = 1.06 = \sqrt{2 * 8 * \ln 2}/\pi$. By thresholding at level $I = 0.5$ we get back the original line exactly.

Triangle. The most important case in the context of this paper is the triangular polygon $\triangle(s,t)$ with vertices $\mathbf{p}$, $\mathbf{q}$ and $\mathbf{r}$, that is defined as $\triangle(s,t) = \mathbf{p} + s(\mathbf{q} - \mathbf{p}) + t(\mathbf{r} - \mathbf{p})$ with $0 \leq s \leq 1$ and $0 \leq t \leq 1 - s$. Again we define some new variables: $A = (\mathbf{x} - \mathbf{p})^2$, $B = (\mathbf{x} - \mathbf{p}) \cdot (\mathbf{q} - \mathbf{p})$, $C = (\mathbf{x} - \mathbf{p}) \cdot (\mathbf{r} - \mathbf{p})$, $D = (\mathbf{q} - \mathbf{p}) \cdot (\mathbf{r} - \mathbf{p})$, $E = (\mathbf{q} - \mathbf{p})^2$, and $F = (\mathbf{r} - \mathbf{p})^2$ and use them to get the expression

$$I(\mathbf{x}_{ijk}) = \frac{\int_0^1 \int_0^{1-s} \exp\left(\frac{-A + 2sB + 2tC - 2stD - s^2E - t^2F}{2\sigma^2}\right) \, dt \, ds}{\max\left[\int_0^1 \int_0^{1-s} \exp\left(\frac{-A + 2sB + 2tC - 2stD - s^2E - t^2F}{2\sigma^2}\right) \, dt \, ds\right]} \tag{5}$$

$$= \frac{\int_0^1 \exp(\frac{C^2 - AF + 2s(BF - CD) + s^2(D^2 - EF)}{\sigma^2 F}) \left[\operatorname{erf}(\frac{-C + sD}{\sigma\sqrt{2F}}) - \operatorname{erf}(\frac{F - C + s(D - F)}{\sigma\sqrt{2F}}) \right] ds}{\int_0^1 \exp(\frac{(D^2 - EF)(3s-1)^2}{18\sigma^2 F}) \left[\operatorname{erf}(\frac{2F - D + 3s(D - F)}{3\sigma\sqrt{2F}}) - \operatorname{erf}(\frac{-D - F + 3sD}{3\sigma\sqrt{2F}}) \right] ds}$$

The denominator is calculated for the centroid $\mathbf{x} = (\mathbf{p} + \mathbf{q} + \mathbf{r})/3$ of the triangle. Because of the crossed term $-2stD$ the double integral can not be separated and we are left with an expression with no analytical solution.

4 Painting and Filling

In our implementation, the triangular polygons are 'painted' separately on the grid by applying equation (5). Representing a polyhedral surface can be considered as the set theoretic *union* of voxelized polygons that overlap in order to get their edges aligned and their vertices to touch. If different polygons contribute to the value of one voxel, we choose the largest value (Koenderink & van Doorn 1986).

Now that the surface of the object is represented on a $3D$ grid, we have to 'fill' the shape to get a representation with value 1 inside the shape, value 0 outside and with smoothly varying values near the boundary. We use a boundary filling algorithm (Foley *et al.* 1990) that starts at a seed gridpoint within the surface.

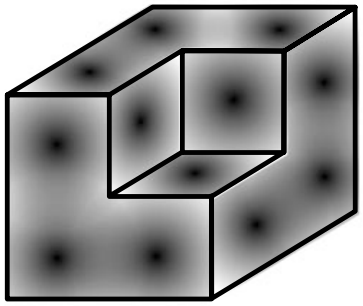

Fig. 1 Voxel grid with Gaussian density profiles

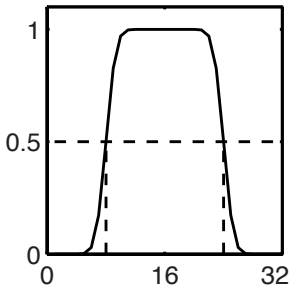

Fig. 2 Example of the voxel values for a line (parallel cross-section).

5 Discussion

The method gives a computationally expensive anti-aliased voxelization of ideal geometrical entities. In our implementation we have made no attempt to optimize. This can easily be done by precomputing the analytic expressions for a variety of graphics primitives and interpolating. Wang (1994) has proposed an anti-aliased voxelization algorithm based on a cone filter that implements the precomputing idea and he obtains volume sampling times of a few seconds on a Silicon Graphics Indigo^2, the same workstation that we use. Our algorithm is related to his, only his filter can lead to spurious structure and is not dependent on the metrical resolution.

The voxelization method proposed here can be applied in 3D (medical) imaging if one wants to go back and forth between the different representations, using a marching cubes-like algorithm to go in the one direction. Finally, this method can be used to acquire the input format for an analysis of the shape of an object in $3 + 1$ dimensional scale-space.

References

Foley, J.D., van Dam, A., Feiner, S.K. & Hughes, J.H.: *Computer Graphics* (Second Edition) Addison Wesley Publishing Company (1990)

Koenderink, J.J.: "The structure of images". *Biol. Cybern.*, **50**, 363-370 (1984)

Koenderink, J.J. & van Doorn, A.J.: "Dynamic shape". *Biol. Cybern.*, **53**, 383-396 (1986)

Lorensen, W.E. & Cline, H.E.: "Marching cubes: A high resolution 3D surface construction algorithm". *Computer Graphics*, **21**, 163-168 (1987)

Wang, S.W. & Kaufman, A.E.: "Volume-sampled 3D modeling". *IEEE Computer Graphics and Applications*, **14**, 26-32 (1994)