Real-time linear silhouette enhancement

Christopher Dyken and Martin Reimers

Abstract. We present a simple method for improving the rendered appearance of coarse triangular meshes. We refine and modify the geometry along silhouette edges in real-time, and thus increase the geometric complexity only where needed. We address how to define the improved silhouette and a method to blend the modified and the original geometry continuously.

§1. Introduction

Coarse triangular meshes are used extensively in real-time rendering applications such as computer games and virtual reality software. Fragment level techniques, like texture mapping and per-pixel lighting, effectively make objects appear to have more geometric detail than they do in reality. Although highly effective in many cases, even the most advanced shading technique cannot hide the piecewise linear silhouette of a coarse polygonal mesh. This artifact can be seen even in current, cutting edge computer games.

The silhouette separates an object from the background and is characterised by a sharp change in contrast and texture, which is very important in the human perception of shape [9]. Therefore, the silhouette region conveys a very significant amount of the visual information.

We propose a straightforward technique to improve the visual appearance of the silhouette of a coarse polygonal mesh. We replace linear silhouette edges with smooth curves and refine the local geometry accordingly during rendering. We define a continuous "silhouetteness" test and use this to blend flat and curved geometry in order to avoid transitional artifacts. We also propose a simple and efficient rendering technique for the silhouette enhanced geometry.

The idea of improving silhouettes of piecewise linear geometry has been addressed before. In [8] a high resolution version of the model is used to create a stencil clipping the outline of the coarse geometry, at the cost of an extra rendering pass. Our method is related to the PN-triangle construction [5] which is based on replacing each triangle of the mesh with a triangular spline patch

xxx and xxx (eds.), pp. 1–4.

ISBN 0-9728482-x-x

defined by its geometry and shading normals. This implicitly improves the silhouettes, but increases the geometric complexity globally. A similar approach is taken in [10], where subdivision is used instead of spline patches. In [1] a local refinement technique for subdivision surfaces is described.

The outline of the rest of the paper is as follows. After some preliminaries, we propose a new continuous *silhouetteness* classification method. In section 3 we define smooth edge curves based on vertex positions and normals and construct a Bézier patch for each triangle. We use silhouetteness to define a view dependent geometry with smoother silhouettes. We conclude with implementational issues and conclusion in sections 4 and 5.

§2. Silhouettes

We assume for simplicity that Ω is a closed triangle mesh with consistently oriented triangles $T_1, \dots, T_N$ and vertices $\boldsymbol{p}_1, \dots, \boldsymbol{p}_n$ in $\mathbb{R}^3$. An edge of Ω is defined as $e_{ij} = [\boldsymbol{p}_i, \boldsymbol{p}_j]$ where $[\cdot]$ denotes the convex hull of a set. The *triangle normal* $\boldsymbol{n}_t$ of a triangle $T_t = [\boldsymbol{p}_i, \boldsymbol{p}_j, \boldsymbol{p}_k]$ is defined as the normalisation of the vector $(\boldsymbol{p}_j - \boldsymbol{p}_i) \times (\boldsymbol{p}_k - \boldsymbol{p}_i)$. Since our interest is rendering Ω we also assume that we are given shading normals, $\boldsymbol{n}_{ti}, \boldsymbol{n}_{tj}, \boldsymbol{n}_{tk}$ associated with the vertices of T_t. The *view point* $\boldsymbol{o} \in \mathbb{R}^3$ is the position of the observer and for a point $\boldsymbol{p}$ on a Ω, the *view direction vector* is $\boldsymbol{p} - \boldsymbol{o}$. If $\boldsymbol{n}$ is the surface normal in $\boldsymbol{p}$ we say that Ω is *front facing* in $\boldsymbol{p}$ if $(\boldsymbol{p} - \boldsymbol{o}) \cdot \boldsymbol{n} \leq 0$, otherwise it is *back-facing*.

The silhouette of a triangle mesh is the set of edges where one of the adjacent faces is front-facing while the other is back-facing. Let $\boldsymbol{p}_{ij}$ be the midpoint of an edge e_{ij} shared by two triangles T_s and T_t in Ω. Defining $f_{ij} : \mathbb{R}^3 \to \mathbb{R}$ by

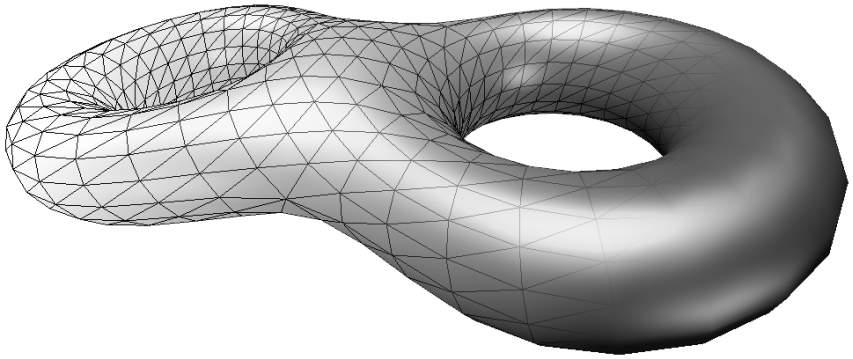
$$f_{ij}(\boldsymbol{x}) := \left(\frac{\boldsymbol{p}_{ij} - \boldsymbol{x}}{\|\boldsymbol{p}_{ij} - \boldsymbol{x}\|} \cdot \boldsymbol{n}_s \right) \left(\frac{\boldsymbol{p}_{ij} - \boldsymbol{x}}{\|\boldsymbol{p}_{ij} - \boldsymbol{x}\|} \cdot \boldsymbol{n}_t \right), \tag{1}$$

we see that e_{ij} is a silhouette edge when observed from $\boldsymbol{o}$ in case $f_{ij}(\boldsymbol{o}) \leq 0$ (and it is not occluded by other objects).

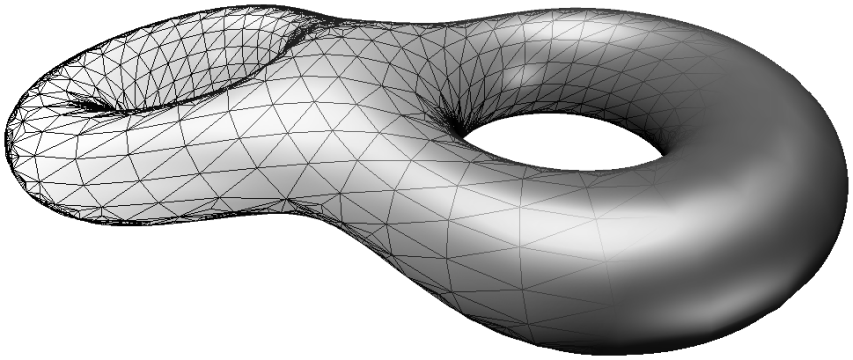
Our objective is to render silhouette edges of Ω as smooth curves. Since these curves does not in general lie in Ω and since "silhouetteness" is a binary function of the view-point, a naive implementation leads to transitional artifacts; the rendered geometry depends discontinuously on the view-point. We propose instead to make the rendered geometry depend continuously on the view-point. To that end we define the *silhouetteness* of e_{ij} seen from $\boldsymbol{x} \in \mathbb{R}^3$ to be

$$\alpha_{ij}(\boldsymbol{x}) := \begin{cases} 1 & f_{ij}(\boldsymbol{x}) \leq 0 \\ 1 - f_{ij}(\boldsymbol{x})/\beta_{ij} & 0 < f_{ij}(\boldsymbol{x}) \leq \beta_{ij} \\ 0 & \beta_{ij} < f_{ij}(\boldsymbol{x}), \end{cases} \tag{2}$$

where $\beta_{ij} \geq 0$ is a constant. This continuous silhouetteness classification extends the standard binary classification by adding a transitional region, see Figure 2. A silhouetteness $\alpha_{ij} \in (0, 1)$ implies that e_{ij} is nearly a silhouette edge.



(a) Coarse mesh



(b) Coarse mesh refined along silhouette

Figure 1. Replacing triangles along the silhouette with triangular spline patches yields a smoother silhouette.

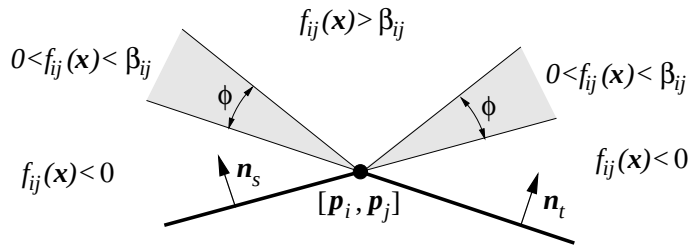


Figure 2. Values of f_{ij} in (1) looking along the edge e_{ij} with the transitional region with angle ϕ marked gray.

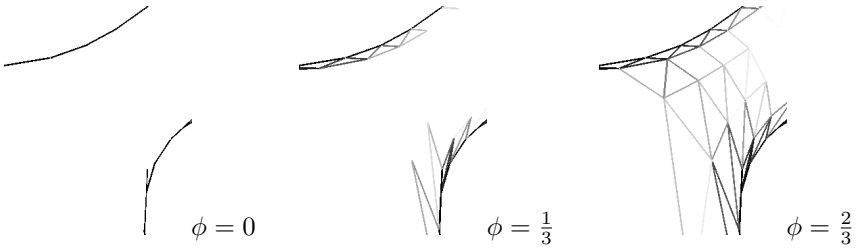


Figure 3. Edges with $\alpha_{ij} > 0$ for different angles ϕ in (3).

We will use silhouetteness to control the view dependent interpolation between silhouette geometry and non-silhouette geometry.

The constant β_{ij} could depend on the local geometry. As an example we could let the transitional region define a "wedge" with angle ϕ with the adjacent triangles as in Figure 2. This amounts to setting

$$\beta = \sin\phi\cos\phi\sin\theta + \sin^2\phi\cos\theta, \tag{3}$$

where θ is the angle between $\boldsymbol{n}_s$ and $\boldsymbol{n}_t$. We have illustrated the effect of varying ϕ in Figure 3. More elaborate methods could be considered, however we found that the heuristic choice of $\beta_{ij} = 0.25$ worked well in practice.

§3. View dependent geometry

In this section we propose a scheme to define smooth silhouettes based on vertex positions, shading normals and view point. We define for each edge e_{ij} in Ω a smooth *edge curve* on the Bézier form

$$\boldsymbol{C}_{ij}(t) = \boldsymbol{p}_i B_0^3(t) + \boldsymbol{c}_{ij} B_1^3(t) + \boldsymbol{c}_{ji} B_2^3(t) + \boldsymbol{p}_j B_3^3(t), \tag{4}$$

where $B_i^3(t) = \binom{3}{i} t^i (1-t)^{3-i}$, see e.g. [7]. This cubic curve interpolates its end points $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$, and it remains to determine the two inner control points $\boldsymbol{c}_{ij}$

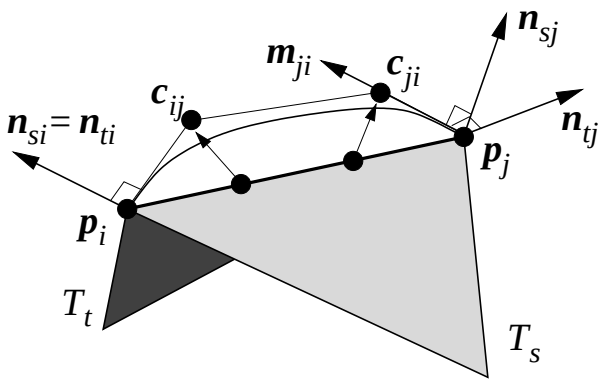


Figure 4. The edge curve control points in case $\boldsymbol{p}_i$ is a smooth edge end and $\boldsymbol{p}_j$ is a feature edge end.

and $\boldsymbol{c}_{ji}$. Let T_s and T_t be the two triangles adjacent to e_{ij}. In the event that the two shading normals $\boldsymbol{n}_{si}, \boldsymbol{n}_{ti}$ are equal we say that the edge end $\boldsymbol{p}_i$ is *smooth*, otherwise it is a *feature* edge end, see Figure 4.

Since shading normals by assumption equals surface normals, we require the edge curve tangents to be orthogonal to the shading normals at the end points. This is equivalent to the conditions

$$(\boldsymbol{c}_{ij} - \boldsymbol{p}_i) \cdot \boldsymbol{n}_{qi} = 0, \qquad q = s, t. \tag{5}$$

This is an under-determined Hermite type of problem which has been addressed before. Sabin [11] determines the interior coefficients uniquely by requiring in addition that the surface normal is parallel to the curve normal at the endpoints, yielding an approximation to a geodesic curve. Note that this method applies only in case $\boldsymbol{n}_{si} = \boldsymbol{n}_{ti}$. In [10] a minimal energy approach is used. Farin [7] describes a method for which the end point tangent direction at $\boldsymbol{p}_i$ is found by projecting $\boldsymbol{p}_j$ into the tangent plane at $\boldsymbol{p}_i$ and the tangent length is determined somewhat heuristically.

To describe our method, we define the linear interpolant to e_{ij} in cubic form,

$$\boldsymbol{L}_{ij}(t) = \boldsymbol{p}_i B_0^3(t) + \boldsymbol{l}_{ij} B_1^3(t) + \boldsymbol{l}_{ji} B_2^3(t) + \boldsymbol{p}_j B_3^3(t), \tag{6}$$

where $\boldsymbol{l}_{k\ell} = (2\boldsymbol{p}_k + \boldsymbol{p}_\ell)/3$. The control points of $\boldsymbol{L}_{ij}$ are used to determine the control points of $\boldsymbol{C}_{ij}$ uniquely as follows. For a smooth edge end we define $\boldsymbol{c}_{ij}$ to be the projection of $\boldsymbol{l}_{ij}$ onto the edge end tangent plane defined by $\boldsymbol{p}_i$ and $\boldsymbol{n}_{si}$, i.e.

$$\boldsymbol{c}_{ij} = \frac{2\boldsymbol{p}_i + \boldsymbol{p}_j}{3} - \frac{(\boldsymbol{p}_j - \boldsymbol{p}_i) \cdot \boldsymbol{n}_{si}}{3} \boldsymbol{n}_{si}. \tag{7}$$

For a feature edge end, i.e. $\boldsymbol{n}_{si} \neq \boldsymbol{n}_{ti}$, the intersection of the two tangent planes

at $\boldsymbol{p}_i$ defined by $\boldsymbol{n}_{si}$ and $\boldsymbol{n}_{ti}$ is the line $\boldsymbol{p}_i + x\boldsymbol{m}_{ij}$ with

$$\boldsymbol{m}_{ij} = \frac{\boldsymbol{n}_{ti} \times \boldsymbol{n}_{si}}{\|\boldsymbol{n}_{ti} \times \boldsymbol{n}_{si}\|}.$$

Projecting $\boldsymbol{l}_{ij}$ onto this line yields

$$\boldsymbol{c}_{ij} = \boldsymbol{p}_i + \frac{(\boldsymbol{p}_j - \boldsymbol{p}_i) \cdot \boldsymbol{m}_{ij}}{3}\boldsymbol{m}_{ij}. \tag{8}$$

It is easy to verify that in either case the conditions (5) are satisfied. The method has linear precision in the sense that it reproduces $\boldsymbol{L}_{ij}$ in case the shading normals are orthogonal to e_{ij}.

It is worthwhile to mention that the naive approach of handling the feature edges by using the smooth edge end method (7) with the average of the two shading normals leads in some cases to sharp edge curves with undesired inflections.

The edge curve coefficients defined in (7) and (8) were used in the PN-triangle construction [5] to define for each triangle of Ω a Bézier patch on the form

$$s(u,v,w) = \sum_{i+j+k=3} \boldsymbol{b}_{ijk} B^3_{ijk}(u,v,w), \qquad B^n_{ijk}(u,v,w) = \frac{n!}{i!j!k!} u^i v^j w^k$$

Our approach is similar, defining the boundary coefficients of the patch as a convex combination of the edge curves (4) and the linear interpolants (6), weighted by silhouetteness;

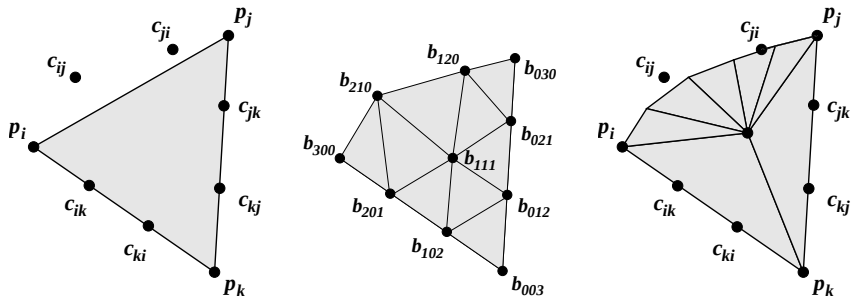
$$\begin{aligned}
&\boldsymbol{b}_{300} = \boldsymbol{p}_i, \quad \boldsymbol{b}_{201} = \alpha_{ik}\boldsymbol{c}_{ik} + (1–\alpha_{ik})\boldsymbol{l}_{ik}, \quad \boldsymbol{b}_{102} = \alpha_{ki}\boldsymbol{c}_{ki} + (1–\alpha_{ki})\boldsymbol{l}_{ki},\\
&\boldsymbol{b}_{030} = \boldsymbol{p}_j, \quad \boldsymbol{b}_{120} = \alpha_{ji}\boldsymbol{c}_{ji} + (1–\alpha_{ji})\boldsymbol{l}_{ji}, \quad \boldsymbol{b}_{210} = \alpha_{ij}\boldsymbol{c}_{ij} + (1–\alpha_{ij})\boldsymbol{l}_{ij},\\
&\boldsymbol{b}_{003} = \boldsymbol{p}_k, \quad \boldsymbol{b}_{012} = \alpha_{kj}\boldsymbol{c}_{kj} + (1–\alpha_{kj})\boldsymbol{l}_{kj}, \quad \boldsymbol{b}_{021} = \alpha_{jk}\boldsymbol{c}_{jk} + (1–\alpha_{jk})\boldsymbol{l}_{jk}.
\end{aligned}$$

As in [5], we define the central control point to be

$$\boldsymbol{b}_{111} = \tfrac{3}{12}\big(\boldsymbol{b}_{201} + \boldsymbol{b}_{102} + \boldsymbol{b}_{021} + \boldsymbol{b}_{012} + \boldsymbol{b}_{210} + \boldsymbol{b}_{120}\big) - \tfrac{1}{6}\big(\boldsymbol{b}_{300} + \boldsymbol{b}_{030} + \boldsymbol{b}_{003}\big),$$

since this choice reproduce quadratic polynomials. See Figure 5 for an illustration of the resulting control mesh. Note that if $\alpha_{ij} = 1$ for all edges, the resulting patches equals the ones in the PN-triangle construction.

Our construction yields for a given viewpoint a spline surface that is a blend between the geometry of the PN-triangle construction and Ω itself. The blend is such that the surface is smooth near a silhouette and flat elsewhere. The patches depends continuously on the viewpoint and moreover, neighbouring patches have a common edge curve, and thus meet continuously. Figure 1 shows the result on a mesh with only smooth edges, while the mesh in Figure 6 has numerous sharp features. Note that in the latter example we could have used the same approach to render the obvious feature lines smoothly as well.



(a) The edge curve control points of a triangle with one silhouette edge.

(b) The control points of the corresponding patch.

(c) A possible rendering strategy for a curved patch.

Figure 5. We replace every triangle with a triangular Bézier patch defined by the silhouetteness and edge curves of adjacent edges.

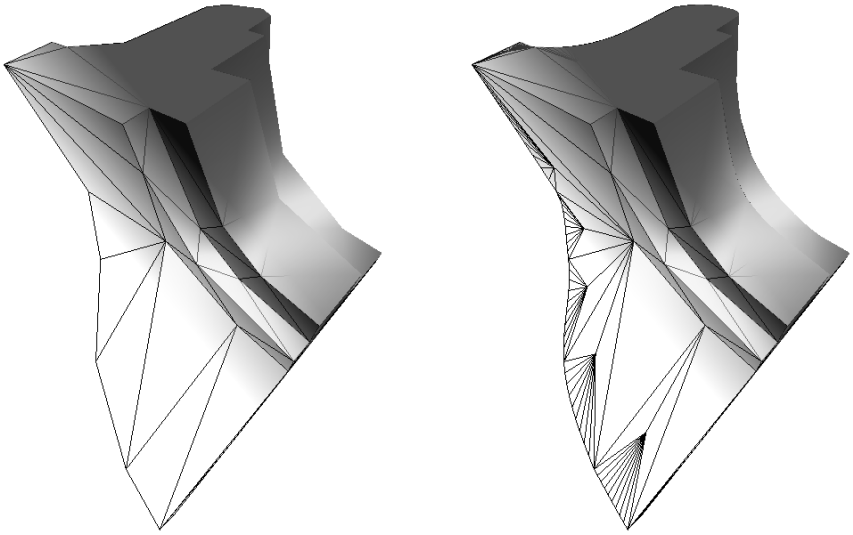


Figure 6. Silhouette improvement of a model with numerous feature edges.

§4. Implementational issues

The algorithm proposed in the previous sections could be implemented as follows. Precompute all the edge curves and store the inner coefficients. For a given view point we calculate the silhouetteness of the edges according to (2). A straight forward implementation is linear in the number of triangles. More sophisticated methods exists, however the break even point appears to be around 10.000 triangles, see [4] and the references therein.

We next tag a triangle as flat if its three edges have zero silhouetteness and curved otherwise. A flat triangle is rendered in the standard way, while a curved triangle could be rendered as depicted in Figure 5 (c). The patch is split into three sub-patches with a common apex at the parametric midpoint of the patch. The sub-patches are refined independently along the base, and the three sub-patches rendered using a triangle fan.

The OpenGL specification [2] encourages the use of perspective-correct interpolation of associated data like texture coordinates, colours and shading normals when rasterising polygons. Therefore associated data should be interpolated linearly when refining the triangular patches.

Our algorithm should be a strong candidate for GPU implementation. However, determining silhouetteness requires connectivity information and refinement generates new geometry. These operations are not supported by current versions of OpenGL or DirectX GPU programs. However, the stencil shadow volume algorithm [6], whose popularity is growing rapidly in real-time rendering applications, have the same functionality requirements. Therefore, it is likely that future revisions of graphics API's have the functionality needed for our algorithm to be implemented on the GPU.

§5. Final remarks

We have proposed a practical algorithm for improving the visual quality of coarse triangle meshes. To overcome transitional artifacts, we introduced a continuous silhouette classification method that could be usefull in other similar applications. Our method of enhancing the silhouettes gave significantly better visual quality in our experiments. Sharp features can be handled through the use of shading normals. We believe our method is applicable in many real-time graphics applications such as flight simulators, computer games and virtual reality settings.

The method could be improved in several ways. Feature lines and boundaries are easily accomodated by our method; we simply set $\alpha_{ij} = 1$ for such edges in order to render them smoothly. If a high resolution version of the mesh is known as in [8], the edge curves could be defined using this fine geometry. One alternative is to let the edge curve $\boldsymbol{C}_{ij}$ approximate a geodesic curve connecting $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$.

§6. References

1. Alliez, P., N. Laurent, H. Sanson, and F. Schmitt, Efficient view-dependent refinement of 3D meshes using $\sqrt{3}$-subdivision, The Visual Computer 19(**4**) (2003), 205–221.
2. Akeley, K., and M. Segal, *The OpenGL® Graphics System: A Specification, version 2.0*, J. Leech and P. Brown (eds.), Silicon Graphics, 2004.
3. Akenine-Möller, T., and E. Haines, *Real-time Rendering*, 2nd ed., A. K. Peters, Ltd., 2002.
4. Cohen, E., B. Gooch, A. Hartner, and M. Hartner, Object space silhouette algorithms, SIGGRAPH 2003 Course Notes, 2003.
5. Boyd, C., J. Peters, J. L. Mitchell, and A. Vlachos, Curved PN triangles, *Proc. 2001 Symposium on Interactive 3D graphics*, ACM Press, 2001, 159–166.
6. Everitt, C., and M. J. Kilgard, Practical and robust stenciled shadow volumes for hardware-accelerated rendering, NVIDIA Corp., 2002.
7. Farin, G., *Curves and Surfaces for CAGD*, 5th ed., Morgan-Kaufmann, 2001.
8. Gortler S., X. Gu, H. Hoppe, P. Sander, J. Snyder, Silhouette clipping, Computer graphics and Interactive Techniques proc., ACM Press/Addison-Wesley, 2000, 327–334.
9. Koenderinc, J. J. What does the occluding contour tell us about solid shape. Perception 13(**3**) (1984), 321–330.
10. Overveld, K. van, and B. Wyvill, An algorithm for polygon subdivision based on vertex normals, Comp. Graph. Int. '97 proc., 1997, 3–12.
11. Sabin, M., Numerical geometry of surfaces, Acta Numerica **3**, A. Iserles (ed.) Cambridge University Press 1994, 411–466.

Christopher Dyken and Martin Reimers
CMA/IFI, University of Oslo
Postbox 1053, Blindern
NO-0316 Oslo, NORWAY
dyken@cma.uio.no,
http://www.math.uio.no/~dyken/
martinre@ifi.uio.no,
http://heim.ifi.uio.no/~martinre/