Interest Value Driven Adaptive Subdivision

Tobias Isenberg Knut Hartmann Henry König
Department of Simulation and Graphics
Otto-von-Guericke University of Magdeburg
{isenberg|hartmann|hkoenig}@isg.cs.uni-magdeburg.de

Abstract

This paper generalizes adaptive subdivision algorithms by introducing an application-dependent Degree of Interest function. Therefore, geometry-related as well as geometry-independent properties can be integrated to control the subdivision algorithm in a uniform framework. We demonstrate how three different applications benefit from this notion.

1 Introduction

Mesh refinement techniques such as progressive meshes (HOPPE [Hop96]) or subdivision schemes (e. g., LOOP [Loo87]) are used to generate higher resolution meshes starting from a coarse model. Higher resolution meshes usually reduce visual artifacts that are inherent to polygonal rendering. However, as the model gets smoother the number of triangles to be rendered increases significantly. The processing power available in current computer graphics hardware, on the other hand, limits the number of refinement steps possible if interactive or real-time frame rates are required. To push these limits further adaptive subdivision algorithms are used. Adaptive methods refine models only in places where this is actually needed, i. e., where the most severe triangulation artifacts occur.

2 Related Work

HOPPE shows how to extend his progressive meshes to adaptive refinement [Hop97]. He applies this approach to generate meshes that are finely tessellated inside the viewport and coarse elsewhere. For different subdivision schemes (e. g., CATMULL-CLARK, DOO-SABIN, LOOP, Butterfly, KOBBELT, and $\sqrt{3}$; for a comparative discussion see, for example, [ZSD$^+$99]) exist adaptions to achieve adaptive refinement. We concentrate on subdivision schemes for triangular meshes—LOOP and $\sqrt{3}$ in particular—because those are most commonly used in computer graphics due to their support in common computer hardware. ZORIN et al. present a system for interactive editing of polygonal meshes [ZSS97]. They use LOOP's scheme and apply subdivision where the local surface is not sufficiently flat. KOBBELT introduces $\sqrt{3}$ subdivision which offers a slower growth of the number of triangles than other methods [Kob00]. He shows that the adaptive subdivision version of his scheme is better suited than other methods since no additional triangle splits are necessary. AMRESH and FARIN discuss an adaptive method based on the LOOP scheme where the decision whether or not to subdivide is based on the angle between face normals [AF02]. MÜLLER and HAVEMANN show how to dynamically apply adaptive LOOP subdivision to triangular meshes [MH00]. They place special emphasis on algorithms to generate crack-free OPENGL optimized triangle output for fast rendering.

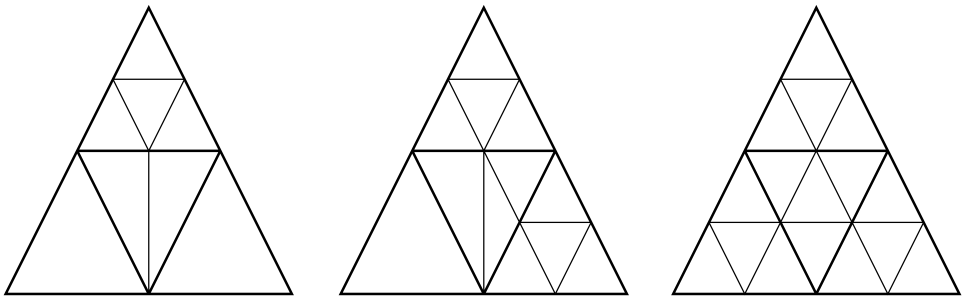
3 Degree of Interest Functions for Adaptive Loop Subdivision

The common goal of most subdivision algorithms is to generate meshes that permit visually pleasing renderings of the models. Therefore, most adaptive subdivision techniques use geometric properties such as flatness—measured by the angle between adjacent faces—to identify polygons where further subdivision is needed. There are, however, various applications that have other requirements as to where fine tessellation is needed. Hence, different criteria have to be used to decide where to apply additional subdivision steps. There are even applications for adaptive subdivision where the criterion for refinement is not entirely geometric for which we will give some examples in Section 4. Thus, we propose a more general method that uses *Degree of Interest functions* (DoIs) for the adaptive refinement decision. A DoI function is specific to the application and describes the importance of surface features which should be the most detailed in the triangle mesh. Hence, the subdivision technique remains the same whereas the features that are to be visualized may vary within or across applications. We demonstrate this approach for LOOP subdivision, an approximative scheme commonly used for triangle meshes [Loo87]. In principle, however, this method is applicable to any subdivision scheme.

Before every subdivision step, the application computes the DoI function for each vertex, edge, or face on the triangle mesh. These DoI values are now used instead of geometric properties to decide if additional subdivision steps are required or not by comparing them to a certain threshold value. This decision is straight-forward for DoI values assigned to faces because LOOP subdivision is based on faces. If the DoI is bigger than the threshold t the face is subdivided, otherwise it is not. If DoI values are computed for edges or vertices instead of faces, all faces are subdivided that are adjacent to the edge or vertex of which the DoI value is found to be higher than the given threshold. Essentially this means that the DoI value for a triangle is derived from the maximum of all the DoIs of the vertices or edges adjacent to this triangle. However, averaging or taking the minimum of the DoI values of vertices or edges surrounding a face may also produce interesting results.

If during a subdivision step a face is going to be subdivided and another face adjacent to the previous face is not going to be subdivided, additional triangulation may be necessary since a new vertex is introduced in the edge shared by both faces. We adopt the scheme suggested in [AF02] (see Figure 1). A triangle that is not going to be subdivided is split in half if it has exactly one subdivided neighbor triangle (see Figure 1(a)). If it has two subdivided neighbors it has to be split by two triangulation edges (see Figure 1(b)). In case all three neighbors are subdivided the not subdivided triangle is split as with regular subdivision (see Figure 1(c)).

Vertices that are part of a subdivided triangle are re-positioned according to regular LOOP



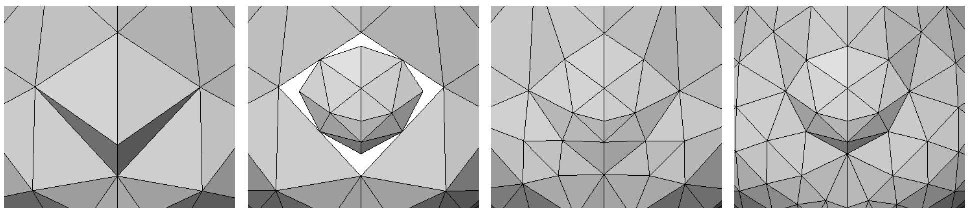
(a) One subdivided neighbor triangle (top).

(b) Two subdivided neighbor triangles (top, right).

(c) Three subdivided neighbor triangles.

Figure 1: Triangulation of a not to be subdivided triangle (middle) which has subdivided neighbor triangles.

subdivision (see [Loo87]). All other vertices remain in their original positions. Not closed patches are also handled as in regular LOOP subdivision. This means, however, that artifacts might occur for patched models where one patch is located exactly next to another patch without actually being connected. This happens when, due to a certain condition of the DoI function, one patch gets subdivided and the other does not (see Figure 2).



(a) Before subdivision.

(b) Adaptive subdivision on open patches.

(c) Adaptive subdivision on closed mesh.

(d) Regular subdivision on open patches.

Figure 2: Artifacts can occur at ends of patches where one patch used to meet another without being attached to it while only one of the patches gets subdivided.

Adaptive subdivision can be used for static as well as dynamic refinement of meshes. In case of dynamic refinement the DoI function has to be re-evaluated for every frame and for every subdivision step. This can be very time consuming. Thus, to speed up the calculation a dynamic tessellation algorithm could be used as suggested in [MH00].

4 Applications

We illustrate the DoI approach for subdivision by applying it to three examples. First, we show how it can be applied to the generation of meshes suited for silhouette extraction. Second, we demonstrate how the preparation of models for haptic rendering benefits from DoI-driven adaptive subdivision where the used meshes have to fulfill certain criteria. The third example shows an application where the DoI is not derived from geometric properties at all but from the user's area of interest in interactive illustrations.

4.1 Silhouette Generation

AZUMA et al. showed how to generate view dependent models using the progressive meshes algorithm [ACD+01]. These meshes are finely tessellated in areas tangential to the viewing direction and sparsely tessellated where perpendicular to it. This means that silhouettes extracted from the model will appear as smooth lines with fewer triangulation artifacts. To achieve similar meshes with our subdivision method the DoI functions are based on the angle between face normal and viewing direction. This is achieved by computing the dot product of both vectors

$$DoI_f = \vec{n} \cdot \vec{v} \tag{1}$$

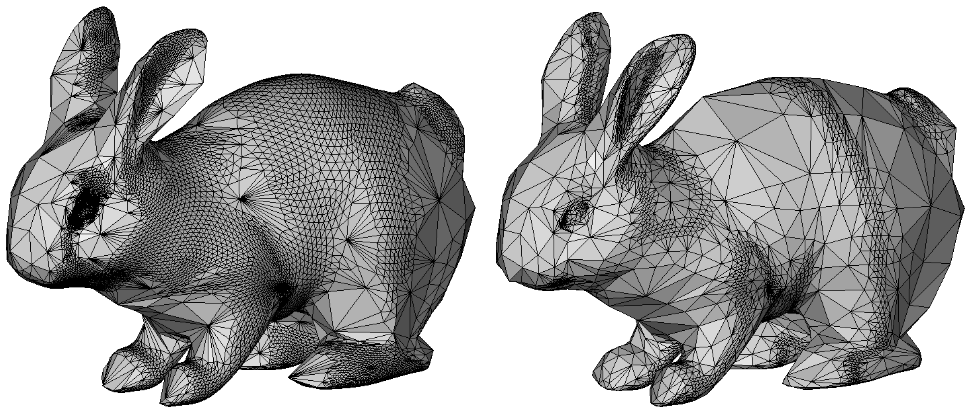
(see Figure 3(a)). A method that produces even less faces for the same number of adaptive subdivision steps considers edges where front facing and back facing triangles meet. Those silhouette edges are assigned a DoI value of 1 while all other edges are assigned a DoI value of 0

$$DoI_e = \begin{cases} 1 & : \quad [(\vec{n_1} \cdot \vec{v} \geq 0) \wedge (\vec{n_2} \cdot \vec{v} < 0)] \vee [(\vec{n_1} \cdot \vec{v} \leq 0) \wedge (\vec{n_2} \cdot \vec{v} > 0)] \\ 0 & : \quad otherwise \end{cases} \tag{2}$$

with n_i being the normals of the two faces. This results in a binary DoI function. Hence, the threshold t the DoI function is to be compared to can be set to anywhere in the interval $(0;1)$. The result of using this DoI function is shown in Figure 3(b).
Meshes generated with adaptive subdivision using these DoI functions have to be updated in every frame since they are view dependent. For generating static meshes which better support silhouette generation one can use the observation that the probability for an edge to be a silhouette edge depends on the angle between its both adjacent faces. In addition, concave edges cannot become silhouette edges at all. Thus, the DoI in this case can be derived directly from the angle between the two faces

$$DoI_e = \arccos(\vec{n_{f_1}} \cdot \vec{n_{f_2}}) \tag{3}$$

(see Figure 4). If the edge is concave it could additionally be set to zero because these edges are never silhouette edges. The result of this approach is similar to the one presented in [AF02].
All three of the DoI functions discussed above allow for the generation of meshes with far less triangles than meshes generated with regular subdivision. The original model used



(a) Faces with an angle of less than 20 degrees to the viewing direction are subdivided.

(b) Edges where front facing and back facing triangles meet are subdivided.

Figure 3: Dynamic adaptive subdivision (3 steps) for better silhouette generation. In both cases the model is viewed directly from the front.

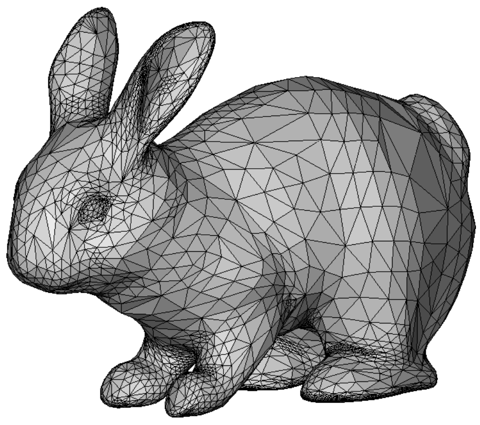
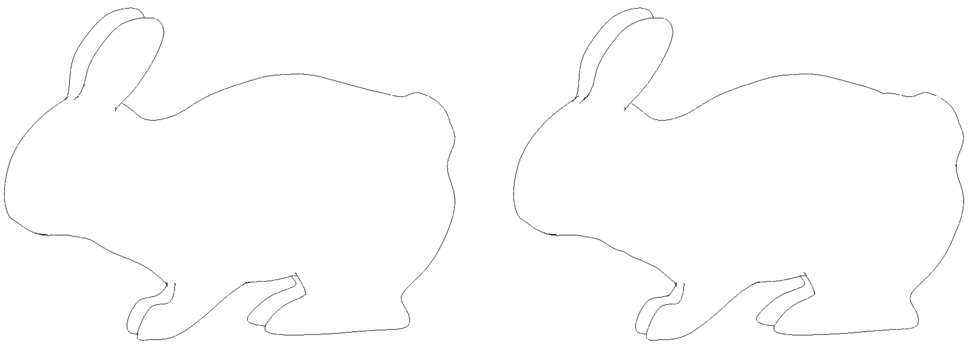


Figure 4: Static adaptive subdivision for better silhouette generation: adjacent faces with an angle higher than 30 degrees are subdivided.

in Figures 3 and 4 has 902 triangles. Three steps of regular subdivision generate a mesh with 57 728 triangles. The mesh in Figure 3(a) has 26 872 triangles, the mesh in Figure 3(b) has only 11 188 triangles, and the static mesh of Figure 4 has 4 532 triangles. Especially the binary DoI function using back facing and front facing triangles generates meshes that yield silhouettes which have almost the same quality as the ones generated from fully subdivided meshes (see Figure 5).


(a) Silhouette computed from regularly subdivided model.

(b) Silhouette from adaptive subdivision where back facing and front facing triangles meet.

Figure 5: Comparison of silhouettes computed from regularly and adaptively subdivided models.

4.2 Pre-Computation for Haptic Rendering

A second example incorporates adaptive subdivision into a haptic rendering algorithm. These algorithms have two major tasks:

- the determination of penetrations into virtual objects and
- the calculation of corresponding haptic feedback to user interaction.

The major disadvantage of polygonal meshes in haptic rendering is that users perceive discontinuities during the exploration of objects. This problem is usually overcome by *force shading*. This technique interpolates the direction of the reaction force to reduce the discontinuities along edges. Unfortunately, this causes lateral forces that can disturb users during the exploration and can influence their perception.
To overcome this problem, there are basically two approaches. The first works directly on the original triangular mesh and uses the LOOP subdivision algorithm for a dynamic, successive refinement. The area of interest is defined by the haptic interaction point (HIP) of the haptic display. This yields a binary DoI function that classifies the triangle closest to the HIP and its 1-neighborhood as interesting and all other triangles as not interesting. To determine the surface contact point (SCP, see Figure 6), the triangles with a high DoI are successively subdivided. Afterwards, the intersection point with the resulting triangles is calculated to determine new DoIs. This subdivision is done until a final subdivision

level is reached. Now, the current SCP can be determined. The limited time to determine the SCP prohibits the application of adaptive subdivision within the intersection calculation. Hence, meshes within different subdivision levels have to be pre-calculated and stored (see [RBR01]).

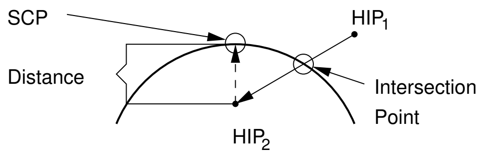


Figure 6: The situation after penetrating an object. HIP_1 denotes the previous and HIP_2 the current interaction point.

As opposed to the approach just discussed our solution is based on a piecewise interpolation of the displayed surface using Bézier patches. To get a better interpolation of the surface, we use CLOUGH-TOCHER split-triangle scheme [CT65]. Each triangle of the original triangle mesh is split into three sub-triangles. Afterwards, for each sub-triangle a cubic patch is constructed that is defined by a control mesh with ten control points (see Figure 7).

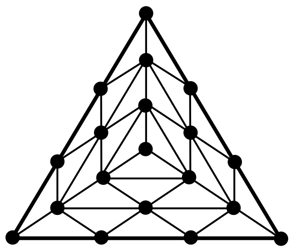


Figure 7: The Clough-Tocher triangle-split scheme.

To reduce the amount of triangles that are necessary to interpolate the surface and for the fitting of the patches we can employ adaptive subdivision. The DoI used in this case has to incorporate two aspects. First, to reduce discontinuities and to minimize lateral forces, the connectivity between two patches on the subdivided surface has to be at least G^1. Second, the intersection calculation for Bézier patches and the determination of the haptic reaction force require that the patch for a triangle is entirely on one side of the triangle.
A necessary and sufficient condition for C^r continuity between two adjacent patches $\mathbf{b}$ and $\hat{\mathbf{b}}$ that share the line $u = 0$ is (see Figure 8(a), also see [Far02]):

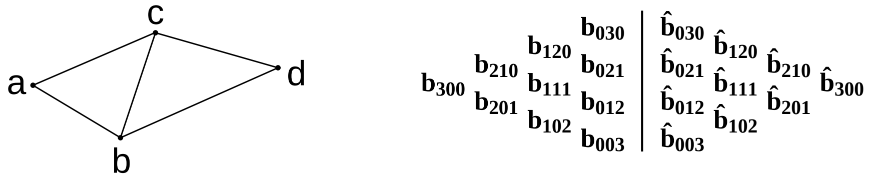
$$\hat{\mathbf{b}}_{(p,j,k)} = \mathbf{b}^p_{\mathbf{j}_0}(\mathbf{d})\,; \qquad p = 0,...,r. \tag{4}$$

In the case of $r = 1$ Equation 4 becomes (see Figure 8(b)):

$$\hat{\mathbf{b}}_{(1,j,k)} = v_1\mathbf{b}_{1,j,k} + v_2\mathbf{b}_{0,j+1,k} + v_3\mathbf{b}_{0,j,k+1}. \tag{5}$$

Equation 5 means that each vertex of the control net $\hat{\mathbf{b}}_{(1,j,k)}$ with $j+k=2$ of the triangle $\hat{\mathbf{b}}$ can be described as a barycentric combination of the vertices of a boundary sub-triangle of the control net of $\mathbf{b}^3$. Moreover, all of these combinations have to be identical. For a cubic patch one will get three triple of barycentric combinations that are identified by another index. Afterwards, these barycentric combinations are used to determine an error value for each line:

$$error = DoI_e = \sum_{i=1}^{3} abs(abs(v_{i_1}) - abs(v_{i_2})) + abs(abs(v_{i_1}) - abs(v_{i_3})). \tag{6}$$



(a) The Bézier triangle $\mathbf{b}^{\mathbf{n}}$ is given by b, c, a and $\hat{\mathbf{b}}^{\mathbf{n}}$ is given by b, c, d.

(b) Arrangement of the control points for two adjacent cubic patches.

Figure 8: Two adjacented sub-triangles and the arrangement of the corresponding control net.

By using this DoI function those triangles get subdivided that used to have too severe errors which would have resulted in perceivable artifacts. Figure 9 shows as an example the mesh of a bone. At the one end of the bone some adjacent triangles do not produce smooth Bézier patch transitions. Those triangles get subdivided and the resulting mesh has no perceivable artifacts anymore.

4.3 Interactive Illustration

Priority values not based on geometric properties are used in a number of interactive illustration systems (e. g., ZoomIllustrator [PRS97], TextIllustrator [SS00], and AGILE [HS02]). The DoI functions or dominance values used in these systems are extracted by analyzing user interactions or visible text segments in order to guarantee a coherent multi-modal presentation.
An analysis of scientific illustrations reveals that human artists display details within the local environment of the most dominant objects. Moreover, these details fade out with increasing distance to the dominant object. Hence, to enhance such interactive illustration applications, the DoI values provided by the systems can directly be used for adaptive subdivision. This makes sure that the most dominant structures are enhanced through the finer tessellation whereas irrelevant geometric parts or objects remain coarse (see example in Figure 10). Therefore, the DoI functions for geometric primitives like points, edges,

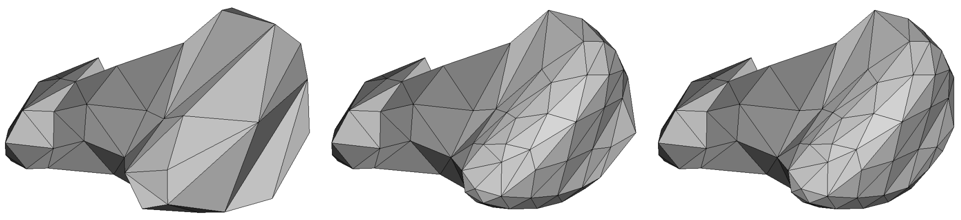

(a) No subdivision steps. (b) One subdivision step. (c) Two subdivision steps.

Figure 9: Triangles that produce perceivable discontinuities in haptic rendering are being subdivided.

and faces are based on the weighted distance $d(p,o)$ of the primitives p to the dominant object(s) o of the geometric model M. For faces f this yields

$$DoI_f = max_{o\in M}\left[d(f,o), \mathrm{DoI}(o)\right]. \tag{7}$$

The distance for primitives constituting a geometric object is zero. In any other case the distance function $d(p,o)$ of p in respect to a complex geometric object o is determined by the Euclidean distance between the primitive p and the center of the bounding object approximation of o. Alternatively, one can also use the minimal Euclidean distance between the primitive p and a point on the internal skeleton of o (for information on how to compute an internal skeleton of polygonal meshes see DEUSSEN et al. [DHR+99]). This distance function is well suited for curved line features like muscles, ligaments, or long bones where the approximation by bounding objects like boxes, spheres, or ellipsoids is not very accurate. However, a distance function which considers the Euclidean distance of p to the center of the bounding object is more appropriate in other application domains with more compact objects.

5 Summary

Adaptive subdivision algorithms adjust the accuracy of the tessellation to yield visually pleasing renderings with polygonal meshes of minimal size. We introduced a generic Degree of Interest function to decide where refinements of the mesh are required most. We demonstrated how this notion benefits three example applications and showed how to derive the necessary DoI functions. Those functions, for example, emphasize silhouette lines, minimize undesired lateral forces in haptic rendering, and emulate illustration styles found in scientific illustrations. This proves that generalized adaptive subdivision algorithms can be successfully applied to achieve a large variety of different goals.

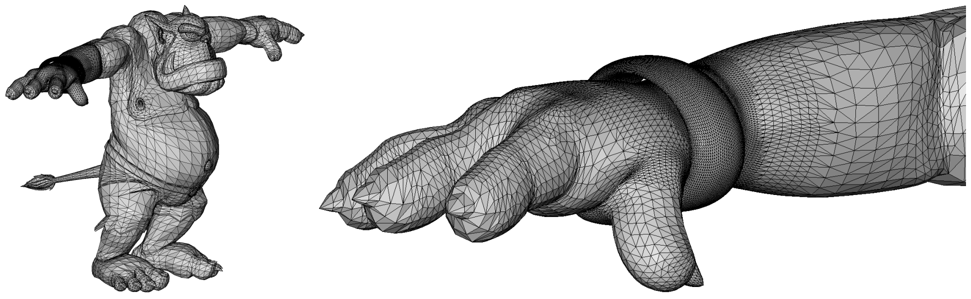


Figure 10: Olaf's bracelet is the object of focus. Thus, its triangles and all triangles of objects close to it are subdivided. The closer the triangles are to the focus point the more subdivision steps are used.

References

[ACD$^+$01] Daniel I. Azuma, Brian Curless, Tom Duchamp, David H. Salesin, Werner Stuetzle, and Daniel N. Wood. View-Dependent Refinement of Multiresolution Meshes with Subdivision Connectivity. Technical report, University of Washington, October 2001. UW-CSE-2001-10-02.

[AF02] Ashish Amresh and Gerald Farin. Adaptive Subdivision Schemes for Triangular Meshes. In Gerald Farin, H. Hagen, and B. Hamann, editors, *Hierarchical and Geometric Methods in Scientific Visualization*. Springer-Verlag, 2002. To appear.

[CT65] R. W. Clough and J. L. Tocher. Finite element stiffness matrices for analysis of plates in bending. In *Proceedings of Conference on Matrix Methods in Structural Analysis (Wright-Patterson Air Force Base, Ohio)*, 1965.

[DHR$^+$99] Oliver Deussen, Jörg Hamel, Andreas Raab, Stefan Schlechtweg, and Thomas Strothotte. An Illustration Technique Using Hardware-Based Intersections and Skeletons. In I. S. MacKenzie and J. Stewart, editors, *Proceedings of Graphics Interface'99 (Kingston, Ontario, June, 1999)*, pages 175–182. Canadian Human-Computer Communications Society, 1999.

[Far02] Gerald Farin. *CAGD*. Academic Press, San Diego, California, 5th edition, 2002.

[Hop96] Hugues Hoppe. Progressive Meshes. In *Proceedings of SIGGRAPH 96 (New Orleans, LA, August 4–9, 1996), Computer Graphics* Proceedings, Annual Conference Series, pages 99–108, Reading, MA, 1996. ACM SIGGRAPH, Addison Wesley Publishing Company.

[Hop97] Hugues Hoppe. View-Dependent Refinement of Progressive Meshes. In *Proceedings of SIGGRAPH 97 (Los Angeles, CA, August 3–8, 1997), Computer Graphics* Proceedings, Annual Conference Series, Reading, MA, 1997. ACM SIGGRAPH, Addison Wesley Publishing Company.

[HS02] Knut Hartmann and Thomas Strothotte. A Spreading Activation Approach to Text Illustration. In Andreas Butz, Antonio Krüger, Patrick Olivier, Stefan Schlechtweg, and Michele Zhou, editors, *Proceedings of the 2nd International Symposium on Smart Graphics (June 11-13, 2002, Hawthorne, NY, USA)*, pages 39–46, New York, 2002. ACM Press.

[Kob00] Leif Kobbelt. $\sqrt{3}$ Subdivision. In Stephen N. Spencer, editor, *Proceedings of SIGGRAPH 2000 (New Orleans, LA, July 23–28, 2000), Computer Graphics* Proceedings, Annual Conference Series, pages 103–112, Reading, MA, 2000. ACM SIGGRAPH, Addison Wesley Publishing Company.

[Loo87] Charles T. Loop. Smooth Subdivision Surfaces Based on Triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.

[MH00] Kerstin Müller and Sven Havemann. Subdivision Surface Tesselation on the Fly using a versatile Mesh Data Structure. *Computer Graphics Forum (Proceedings of Eurographics)*, 19(3):151–159, August 2000.

[PRS97] Bernhard Preim, Andreas Raab, and Thomas Strothotte. Coherent Zooming of Illustrations with 3D-Graphics and Text. In W. A. Davis, M. Mantei, and R. V. Klassen, editors, *Proceedings of Graphics Interface '97*, pages 105–113. Canadian Information Processing Society, 1997.

[RBR01] Chris Raymaekers, Koen Beets, and Frank Van Reeth. Fast Haptic Rendering of Complex Objects Using Subdivision Surfaces. In Karl Reinig, editor, *Proceedings of the Sixth PHANTOM Users Group Workshop (Aspen, Colorado, October 27–30, 2001)*, 2001.

[SS00] Stefan Schlechtweg and Thomas Strothotte. Generating Scientific Illustrations in Digital Books. In *Smart Graphics. Papers from the 2000 AAAI Spring Symposium (Stanford, 20-22 March 2000)*, pages 8–15, Menlo Park, 2000. AAAI Press.

[ZSD$^+$99] Denis Zorin, Peter Schröder, Tony DeRose, Jos Stam, Leif P. Kobbelt, and Joe Warren. Subdivision for Modeling and Animation. Siggraph 99 course notes, ACM SIGGRAPH, 1999.

[ZSS97] Denis Zorin, Peter Schöder, and Wim Sweldens. Interactive Multiresolution Mesh Editing. In *Proceedings of SIGGRAPH 97 (Los Angeles, CA, August 3–8, 1997), Computer Graphics* Proceedings, Annual Conference Series, pages 259–268, Reading, MA, 1997. ACM SIGGRAPH, Addison Wesley Publishing Company.