HIGH-QUALITY ISOSURFACE RENDERING WITH EXACT GRADIENT

Philippe Thévenaz and Michael Unser

Swiss Federal Institute of Technology Lausanne
philippe.thevenaz@epfl.ch, michael.unser@epfl.ch

ABSTRACT

We address the task of rendering by ray tracing the isosurface of a high-quality continuous spline model of volumetric discrete and regular data. By expressing the spline model as a sum of non-negative B-splines, we are able to confine the potential location of the isosurface within a thin binary shell. We then show how to use the space-embedding property of splines to further shrink this shell to essentially a single-voxel width. We also propose a new illumination model that highlights the outline of the rendered isosurface, which provides for a sensitive test of the perceived quality of the rendering. We present experiments to support our claims, along with an efficient algorithm to compute simultaneously an array of B-splines and of its derivatives.

1. INTRODUCTION

We consider the problem of rendering data given by a 3D regular array of measurements $f(\mathbf{k}), \quad \mathbf{k} \in \mathbb{Z}^3$. An early solution called marching cubes [1] has attracted some interest by proposing a heuristic to perform the conversion of the volumetric data $f(\mathbf{k})$ into a list of polygonal faces. Unfortunately, this conversion is ambiguous in some cases [2]; moreover, the number of faces may grow very large which may become a hindrance to the rendering speed. In addition, the planarity of the faces stands in contradiction with the smooth, organic surfaces that one expects from biomedical volumes, a common source of volumetric datasets.

Another method called volumetric rendering considers the simulated propagation of rays in volumes of varying opacities [3]. While it lends itself well to photorealism, including lense effects and antialiasing [4], its computational requirements are very high. It is only by using custom [5] or massively parallel [6] hardware that reasonable rendering times can be achieved. Moreover, the prescription of an adequate volume of opacities proves to be a delicate issue; furthermore, the volume rendering integral must be discretized in practice, which introduces yet another set of difficulties [7].

We prefer to render an isosurface, which is uniquely defined by the frontier of the solid that satisfies $f < f_0$, where f_0 is some arbitrary threshold. Modern methods [8, 9] rely on fitting the discrete data $f(\mathbf{k})$ with a continuous model $f(\mathbf{x})$, and then on rendering—by a technique such as ray tracing—the continuous surface defined by the frontier of $f(\mathbf{x}) < f_0$. Rendering an isosurface is intrinsically faster than rendering a volume of opacities, because one must evaluate an illumination model once only, as opposed to many times for volumetric rendering.

The continuous model $f(\mathbf{x})$ is related to the discrete data $f(\mathbf{k})$ by $f(\mathbf{x}) = \sum f(\mathbf{k})\,\varphi_{\mathrm{int}}(\mathbf{x}-\mathbf{k})$. In recent years, a lot of attention has been devoted to investigate which specific interpolating continuous function φ_{int} would result in the best performance for visualization [10, 11, 12, 13]. In this paper, we deduce from a list of basic requirements that a very reasonable basis function in the context of isosurface rendering is the quadratic B-spline.

Unlike the basis φ_{int} used in most previous visualization works, the quadratic B-spline $\varphi = \beta^2$ is not interpolating. We do still reproduce the data exactly by adopting the following interpolation model:

$$f(\mathbf{x}) = \sum_{\mathbf{k}\in\mathbb{Z}^3} c_{\mathbf{k}}\,\varphi(\mathbf{x}-\mathbf{k}),$$

where the spline coefficients $c_{\mathbf{k}}$ are obtained from the data samples $f(\mathbf{k})$ by a recursive digital filtering method [14, 15]. This filtering step is performed once only, as preprocessing.

A number of approaches exist to reduce the computational cost of finding an isosurface [16, 17, 18]. In this paper, we take advantage of numerous properties of B-splines to derive a scheme where we are able to tell apart those voxels that enclose the isosurface from those that are more remote. In the context of ray tracing, this results in a large speed-up since it is necessary to evaluate the model $f(\mathbf{x})$ only in the near vicinity of the isosurface. This advantage is not available to interpolating bases φ_{int} because they oscillate, and their whole support must be considered instead.

2. FIRST PRINCIPLES

The order of approximation L is a direct index of the quality of a basis function φ. One of several interpretations of L expresses that any polynomial of degree up to $(L-1)$ can be represented exactly as a weighted sum of shifted basis functions $a_0 + \sum a_n\,x^n = \sum c_k\,\varphi(x-k)$. To understand why this is important, it is enough to remember that the reminder of a Taylor series gets smaller the more terms are considered. Thus, we ask that

- The order of approximation of φ must be large.

The support W of φ has immediate implications on the computational cost of estimating the continuous model $f(\mathbf{x})$. Thus, we ask that

- The support of φ must be short.

The ray-tracing technique requires the knowledge of a normal to the isosurface. If we model locally the continuous model $f(\mathbf{x})$ as the truncated Taylor series $f(\mathbf{x}+\Delta\mathbf{x}) = f(\mathbf{x}) + \langle\Delta\mathbf{x}, \boldsymbol{\nabla} f(\mathbf{x})\rangle$, then the isosurface condition $f(\mathbf{x}+\Delta\mathbf{x}) = f(\mathbf{x}) = f_0$ implies that $\langle\Delta\mathbf{x}, \boldsymbol{\nabla} f(\mathbf{x})\rangle = 0$, which expresses that the normal is indeed parallel to the gradient of the data since $\boldsymbol{\nabla} f(\mathbf{x})$ is perpendicular to every vector $\Delta\mathbf{x}$ that belongs to the isosurface. Thus, we ask that

- φ must be continuously differentiable.

Only one family of functions maximizes L while minimizing W; its members are called MOMS [19]. Within this family, the shortest-support function that is continuously differentiable is unique; it is the quadratic spline.

3. RAY TRACING

Ray tracing is a rendering method that has evolved over the years into many variants. In its basic form, rays go through a virtual pinhole camera consisting of an eye $\mathbf{E}$ and of a projection plane. On one side of the eye, the rays intersect the image under construction (the projection plane); one pixel is painted per ray. On the other side of the eye, a given ray will propagate until it reaches an obstacle, say, the isosurface or the bounding box of the volumetric data to render. The illumination model will then determine which color has to be assigned to the pixel belonging to the ray.

Let $\mathbf{P}$ be the (3D) coordinate of a pixel on the projection plane. Each coordinate $\mathbf{x}$ of the ray satisfies $\mathbf{x} = \mathbf{E} + t\,(\mathbf{P} - \mathbf{E})$, where t is some rectilinear parameter. To determine the intersection of the ray with the isosurface, one must essentially solve for the value of t_0 such that

$$f_0 = f(\mathbf{x}_0) = f(\mathbf{E} + t_0\,(\mathbf{P} - \mathbf{E})).$$

Without loss of generality, from now on we consider that $f_0 = 0$. Finding the whole isosurface is thus equivalent to finding the set of $\mathbf{x}_0$ such that $f(\mathbf{x}_0) = 0$.

Besides enjoying a finite support, a very interesting property of B-splines is their non-negativity. To see why, consider some fixed $\mathbf{x}_0$: the finite support of the tensor-product B-spline $\beta(\mathbf{x})$ will determine the set C of those coefficients $c_{\mathbf{k}}$ that contribute to the evaluation of the continuous model for $\mathbf{x}_0$; meanwhile, the same set C has also to be used for every $\mathbf{x}$ in some unit cube[1] X around $\mathbf{x}_0$. Now, whenever all members of C happen to have the same sign, it can be ascertained that the unit cube contains no part of the isosurface because $0 \leq \beta^n(\mathbf{x})$.

This suggests the following preprocessing steps for image rendering: first, subtract the threshold f_0 from the data; then, determine the coefficients $c_{\mathbf{k}}$ out of these reduced data; produce an array of binary values $b_{\mathbf{k}}$ where each element is set to true when $c_{\mathbf{k}} < 0$, to false otherwise; finally, update this array by computing its morphological gradient with a structural element that is a centered cuboid of odd size $(1 + 2\,\lfloor\frac{n+1}{2}\rfloor)$. Since none of the preprocessing steps depends on the viewing orientation, the volumetric data can be stored directly as $\{b, c\}$.

The rendering itself can now proceed. While exploring a ray for finding the isosurface, it is enough to hop[2] from voxel to voxel, based on the content of the binary array b. This allows for easy operations such as cutouts (realized by binary operations on b), and such as the representation of b by an octree, which could potentially lead to a dramatic acceleration of the rendering by transforming hops into strides. It is only when an element $b_{\mathbf{k}}$ indicates a potential zero-crossing that it is necessary to actually perform the costly operation of evaluating f for some non-integer $\mathbf{x}$.

4. ROOT FINDING

The set X of coordinates $\mathbf{x}$ that share a set C of coefficients $c_{\mathbf{k}}$ is centered on a voxel for even degrees, and on the corner of a voxel for odd degrees. To simplify the discussion, we will consider only even degrees from now on.

[1]Note that $\mathbf{x}_0$ is not necessarily in the geometric center of this cube. For odd degrees, the center is $\mathbf{k} = \lfloor\mathbf{x}_0\rfloor + \frac{1}{2}\,\mathbf{1}$, while for even degrees it is $\mathbf{k} = [\mathbf{x}_0]$.

[2]We observe experimentally that the average jump length is about 0.653; we leave to the reader the task to find the exact value by an argument akin to Buffon's needle.

To find a solution to the equation $0 = f(\mathbf{E} + t\,(\mathbf{P} - \mathbf{E}))$ within a voxel indicated by $b_{\mathbf{k}}$ as potentially containing a part of the isosurface, we first identify the two rectilinear parameters t_1 and t_2 that correspond to the locations where the ray enters, respectively exits the voxel $\mathbf{k}$. We then compute f_1 and f_2, which should bracket the isosurface, a condition revealed by $f_1\,f_2 \leq 0$. If such is the case, we proceed with a standard root-finding method such as Brent's [20] to get t_0 such that $f(\mathbf{E}+t_0\,(\mathbf{P}-\mathbf{E})) = 0$. As each root candidate $\mathbf{x}$ is examined in turn, the continuous model $\sum c_{\mathbf{k}}\,\beta^2(\mathbf{x} - \mathbf{k})$ is computed efficiently using the recursion given in the appendix.

5. VOXEL PRUNING

The series of preprocessing steps proposed above already results in a significant reduction of the computational burden with respect to ray stepping. Nevertheless, yet another property of splines allows one to reduce it further: multiresolution space embedding. For odd degrees, it is possible to represent any spline $f(x) = \sum c_1(k)\,\beta^n(x - k)$ with a basis function consisting of a B-spline of same degree but with a support scaled-down by a factor 2 (other factors are possible [14]). The same is true for even degrees, up to an additional shift. In both cases, we write that

$$f(x) = \sum_{k\in\mathbb{Z}} c_1(k)\,\beta^n(x-k) = \sum_{k\in\mathbb{Z}} c_2(k)\,\beta^n(2\,x-k-\frac{n \bmod 2}{2})$$

$$c_2(k) = \sum_{l\in\mathbb{Z}} c_1(l)\,u_2^n(2\,l - k)$$

$$u_2^n(k) = \begin{cases} 2^{-n}\begin{pmatrix} n+1 \\ k + \lfloor\frac{n+1}{2}\rfloor \end{pmatrix} & k \in [-\lfloor\frac{n+1}{2}\rfloor, \lfloor\frac{n}{2}\rfloor + 1] \\ 0 & k \notin [-\lfloor\frac{n+1}{2}\rfloor, \lfloor\frac{n}{2}\rfloor + 1]. \end{cases}$$

The argument applies recursively to yield ever-finer coefficients $c_{2,4,8,\ldots}$. We take advantage of this alternate representation of $f(x)$ by remarking that the coefficients $c_{2^d}(k)$, $d \in \mathbb{N}$ are all multiplied by a non-negative basis function. Thus, it is easy to detect those cases where the signs of $c_1 \in C$ differ, but where in reality no part of the isosurface belongs to X: when this is indeed the case, there will be a sufficient depth 2^d such that all coefficients $c_{2^d}(k)$ have the same sign over the set X—note that $c_\infty(k)$ ultimately converges to some $f(x)$.

Thus, we propose an additional preprocessing step that consists in visiting each relevant voxel in b, and that removes all those cases where it can be determined by the procedure above that no isosurface is contained within the voxel. For a quadratic model $n = 2$, this procedure can as much as halve the original number of candidates in b. Ideally, the resulting binary volume b should be a 6-connected, 1 voxel-width shell. In practice, at any depth d it is necessary to produce and examine the sign of $\mathcal{O}(2^{3\,d}\,n^3)$ coefficients, which grows very quickly out of manageable size, even considering that this preprocessing can be performed off-line. Nevertheless, we will see in Section 7 that a shallow recursion depth is enough to bring significant benefits.

6. ILLUMINATION MODEL

Once an intersection $\mathbf{x}_0$ between a ray and the isosurface has been located, one must paint the pixel whence the ray originates. We use here a simplified version of Phong's illumination model and consider ambient, diffusion, and specular effects, without taking into account the effects related to light attenuation with distance [21].

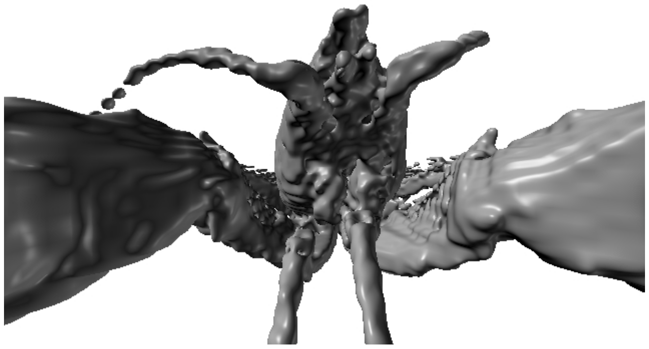

Fig. 1. Phong rendering of a lobster with two positional light sources.

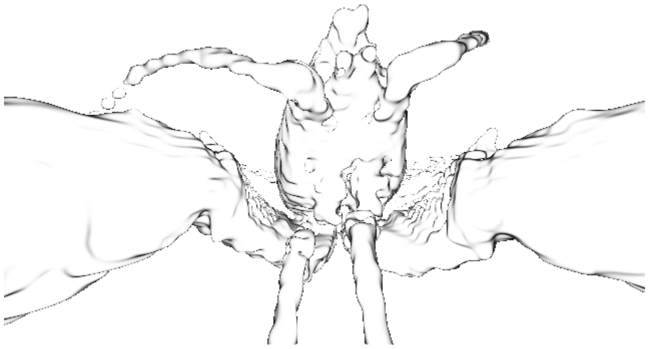

Fig. 2. Rendering of a lobster with our proposed outline illumination model.

We must first obtain the gradient of f at $\mathbf{x}_0$, which we determine analytically by differentiating the continuous model. This yields $\partial f(\mathbf{x}_0)/\partial x = \sum c_{\mathbf{k}}\, \partial\beta^2(\mathbf{x}_0 - \mathbf{k})/\partial x$, which can be computed efficiently as indicated in the appendix. We proceed similarly for y and z; finally, we get the normal $\mathbf{N} = \boldsymbol{\nabla} f/\|\boldsymbol{\nabla} f\|$ which is a unit-norm vector perpendicular to the isosurface.

Let $\mathbf{L}_k$ be a unit-norm vector joining $\mathbf{x}_0$ to the k-th light source of intensity I_k, and let $\mathbf{V}$ be a unit-norm vector joining $\mathbf{x}_0$ to the eye $\mathbf{E}$. Our simplified Phong illumination model is then given by

$$I = k_a\, I_a + \sum_k I_k\,(k_d\,\langle \mathbf{N}, \mathbf{L}_k\rangle + k_s\,\langle \mathbf{V}, \mathbf{R}_k\rangle^{n_s}),$$

where $(k_a\, I_a)$ is the contribution of ambient light, k_d the coefficient of diffusion and k_s the specular coefficient, while n_s is the specular exponent. In addition, $\mathbf{R}_k = 2\,\mathbf{N}\,\langle \mathbf{N}, \mathbf{L}_k\rangle - \mathbf{L}_k$ is the reflection of $\mathbf{L}_k$ around $\mathbf{N}$.

Alternatively, we propose a very simple illumination model that, despite its concise expression, demands high rendering quality because it tends to highlight only the difficult cases. Our proposed illumination model is given by

$$I = \left(1 - \langle \mathbf{N}, \mathbf{V}\rangle^2\right)^{n_o},$$

where n_o is the outline exponent. Its purpose is to emphasize the contours of the rendering, since those are certainly very important from a perceptual point of view. Although there are even less physical justifications for our model than there are in the case of Phong (we don't even need any light source!), renderings such as that of Figure 2 read extremely well.

7. EXPERIMENTS

Figure 1 shows our Phong rendering of a close-up of the $120 \times 120 \times 34$ lobster volume[3], while Figure 2 shows the same volume rendered with our outline illumination model. The size of both images is 640×360, without antialiasing. The isosurface threshold $f_0 = 85.41$ has been determined by a K-means algorithm. The illumination model makes the only distinction between those two images.

[3]Lobster dataset courtesy of, and © by, Advanced Visual Systems, via Mark Kessler, University of Michigan Medical School.

	$\mathrm{card}(b)$	
Initial	$83'228$	100%
Pruning after one level	$67'237$	80%
Pruning after two levels	$56'234$	67%
Pruning after three levels	$50'463$	60%
Pruning after four levels	$47'621$	57%

Table 1. Effect of pruning on $\mathrm{card}(b)$

We now examine the practical benefits of the approach presented in this paper and analyze the performance of our algorithm. Table 1 shows that the whole lobster volume contains $83'228$ voxels which are close enough to the isosurface that a change in the sign of the spline coefficients c can be detected within the volume of influence C of each coefficient $c_{\mathbf{k}}$. After four of the pruning recursions described in Section 5, we were able to reject 43% of these candidates, a nearly optimal score.

The vantage point we have selected is inside the volume, in front of the head of the lobster, midway between its pincers. Since this corresponds to an extreme close-up, the footprint of a voxel may cover many pixels of the projection plane. For this configuration, we have painted $109'291$ pixels, which is a number much higher than the number of voxels containing the isosurface.

8. CONCLUSION

We have proposed two complementary ways to accelerate the rendering of an isosurface by ray tracing in the context of spline interpolation. At first, by expressing the spline model as a sum of non-negative basis functions, we are able to obtain a binary volume that indicates every potential location of the isosurface. As second step, we have used the space-embedding property of splines to prune candidates in this binary volume. The gain can be as large as the spline degree, a factor two in our case.

We have implemented the renderer in which we propose, as alternative to Phong's, a new illumination model that highlights the outline of the isosurface. This illumination model is highly sensitive to the accuracy of the volume gradient, which we compute exactly at essentially no cost thanks to a recursive scheme presented in the appendix.

9. APPENDIX

We propose here an efficient recursive scheme to compute simultaneously an array of B-spline values and of their derivatives for $(n+1)$ arguments spaced one unit apart, and that can be initialized by $\beta^0(x) = \frac{1}{2}\,(\operatorname{sign}(x+\frac{1}{2}) - \operatorname{sign}(x-\frac{1}{2}))$

$$\left\{ \begin{array}{l} \beta^n(x) = \frac{1}{n}\,((x+\frac{n+1}{2})\,\frac{\partial\beta^n(x)}{\partial x} + (n+1)\,\beta^{n-1}(x-\frac{1}{2})) \\ \frac{\partial\beta^n(x)}{\partial x} = \beta^{n-1}(x+\frac{1}{2}) - \beta^{n-1}(x-\frac{1}{2}) \end{array} \right.$$

For better efficiency, we complement the scheme above by making use of the property known as the partition of unity. This results in extra savings. For example, letting $n = 2$ results in only 13 operations to get $\{\beta^2(x-1), \beta^2(x), \beta^2(x+1), \frac{\partial\beta^2(x-1)}{\partial x}, \frac{\partial\beta^2(x)}{\partial x}, \frac{\partial\beta^2(x+1)}{\partial x}\}$ under the hypothesis that $-\frac{1}{2} < x < \frac{1}{2}$, as follows:

$$\begin{array}{rcl} \beta^1_{-1} & = & \frac{1}{2} + x \\ \beta^1_1 & = & 1 - \beta^1_{-1} \\ \dot\beta^2(x-1) & = & \beta^1_{-1} \\ \dot\beta^2(x) & = & \beta^1_1 - \beta^1_{-1} \\ \dot\beta^2(x+1) & = & -\beta^1_1 \\ \beta^2(x-1) & = & \frac{1}{2}\,\beta^1_{-1}\,\dot\beta^2(x-1) \\ \beta^2(x) & = & \frac{1}{2}\,((x+\frac{3}{2})\,\dot\beta^2(x) + 3\,\beta^1_{-1}) \\ \beta^2(x+1) & = & 1 - \beta^2(x) - \beta^2(x-1). \end{array}$$

10. REFERENCES

[1] W.E. Lorensen and H.E. Cline, "Marching cubes: A high-resolution 3D surface construction algorithm," *Computer Graphics*, vol. 21, no. 4, pp. 163–169, July 1987.

[2] Jane Wilhelms and Allen Van Gelder, "Topological considerations in isosurface generation—Extended abstract," *Computer Graphics*, vol. 24, no. 5, pp. 79–86, November 1990.

[3] James T. Kajiya and Brian P. Von Herzen, "Ray tracing volume densities," *Computer Graphics*, vol. 18, no. 3, pp. 165–174, July 1984.

[4] Kevin L. Novins, François X. Sillion, and Donald P. Greenberg, "An efficient method for volume rendering using perspective projection," *Computer Graphics*, vol. 24, no. 5, pp. 95–102, November 1990.

[5] U. Kanus, M. Meißner, W. Straßer, A. Kufman, R. Amerson, R.J. Carter, B. Culbertson, P. Kuekes, and G. Snider, "Implementation of Cube-4 on the Teramac custom computing machine," *Computer and Graphics*, vol. 21, no. 2, pp. 199–208, 1997.

[6] Craig M. Wittenbrink and Arun K. Somani, "Time and space optimal data parallel volume rendering using permutation warping," *Journal of Parallel and Distributed Computing*, vol. 46, no. 2, pp. 148–164, November 1997.

[7] Kevin L. Novins and James Arvo, "Controlled precision volume integration," in *Proceedings of the 1992 ACM Workshop on Volume Visualization (VVS'92)*, Boston MA, USA, October 19-20 1992, pp. 83–89.

[8] Marc Levoy, "Volume rendering," *IEEE Computer Graphics and Applications*, vol. 8, no. 3, pp. 29–37, May 1988.

[9] Željka Mihajlović, Alan Goluban, and Damir Kovačić, "The B-spline interpolation in visualization of the three-dimensional objects," in *Proceedings of the Twentieth International Conference on Information Technology and Interfaces (ITI'98)*, Pula, Croatia, June 16-19 1998, pp. 425–431.

[10] Ingrid Carlbom, "Optimal filter design for volume reconstruction and visualization," in *Proceedings of the 1993 IEEE International Symposium on Visualization (VIS'93)*, San Jose CA, USA, October 1993, pp. 54–61.

[11] Stephen R. Marschner and Richard J. Lobb, "An evaluation of reconstruction filters for volume rendering," in *Proceedings of the 1994 IEEE International Symposium on Visualization (VIS'94)*, R. Daniel Bergeron and Arie E. Kaufman, Eds., Washington DC, USA, October 1994, pp. 100–107.

[12] Mark J. Bentum, Barthold B.A. Lichtenbelt, and Tom Malzbender, "Frequency analysis of gradient estimators in volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 2, no. 3, pp. 242–254, September 1996.

[13] Torsten Möller, Raghu Machiraju, Klaus Mueller, and Roni Yagel, "Evaluation and design of filters using a Taylor series expansion," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, no. 2, pp. 184–199, April-June 1987.

[14] Michael Unser, Akram Aldroubi, and Murray Eden, "B-spline signal processing: Part I—Theory," *IEEE Transactions on Signal Processing*, vol. 41, no. 2, pp. 821–833, February 1993.

[15] Michael Unser, Akram Aldroubi, and Murray Eden, "B-spline signal processing: Part II—Efficient design and applications," *IEEE Transactions on Signal Processing*, vol. 41, no. 2, pp. 834–848, February 1993.

[16] Jane Wilhelms and Allen Van Gelder, "Octrees for faster isosurface generation—Extended abstract," *IEEE Transactions on Medical Imaging*, vol. 19, no. 7, pp. 739–758, July 2000.

[17] Yarden Livnat, Han-Wei Shen, and Christopher R. Johnson, "A near optimal isosurface extraction algorithm using the span space," *IEEE Transactions on Visualization and Computer Graphics*, vol. 2, no. 1, pp. 73–84, March 1996.

[18] Yarden Livnat, Han-Wei Shen, and Christopher R. Johnson, "Correction—A near optimal isosurface extraction algorithm using the span space," *IEEE Transactions on Visualization and Computer Graphics*, vol. 2, no. 2, pp. 184, June 1996.

[19] Thierry Blu, Philippe Thévenaz, and Michael Unser, "Minimum support interpolators with optimum approximation properties," in *Proceedings of the 1998 IEEE International Conference on Image Processing (ICIP'98)*, Chicago IL, USA, October 4-7 1998, vol. III, pp. 242–245.

[20] Richard P. Brent, *Algorithms for Minimization Without Derivatives*, Prentice-Hall, 1973.

[21] Bui-Tuong Phong, "Illumination for computer generated pictures," *Communications of the ACM*, vol. 18, no. 6, pp. 311–317, June 1975.