Survey of Techniques for Data-dependent Triangulations Approximating Color Images

Burkhard Lehner[1], Georg Umlauf[1], and Bernd Hamann[2]
1. Department of Computer Science, University of Kaiserslautern, Germany,
{lehner|umlauf}@informatik.uni-kl.de
2. Institute for Data Analysis and Visualization (IDAV) and
Department of Computer Science, University of California, Davis, USA,
hamann@cs.ucdavis.edu

Abstract: We present a survey of different techniques to approximate a color image using a piecewise linear interpolation induced by a triangulation of the image domain. We also include a detailed description of a method called *guided simulated annealing* (GSA) we designed that reduces computation time significantly. Finally, we give a short overview of possible extensions.

1 Introduction

Given a scalar field $f : \Omega \to \mathbb{R}$ over a 2-dimensional domain $\Omega \subset \mathbb{R}^2$, a triangulation T of the domain Ω can be used to approximate the scalar field by linear interpolation of the function values at the vertices for every triangle t of T. The approximation quality, i.e., the distance between the approximation and the scalar field, depends on three factors:

- the number of vertices in T
- the vertex positions, i.e., the corners of the triangles
- the connectivity of the vertices, i.e., the vertex combination of the triangles

If T adapts well to features of f, the approximation quality can be high even with a small number of triangles.

Since we are looking for triangulations which define an approximation with low approximation error with respect to f, the criterion to decide which triangulation to use depends on our data, the scalar field f. The result of this search is therefore called a *data-dependent triangulation*. Even if the number of triangles and the position of the vertices is fixed, the number of possible triangulations is large, so that an exhaustive search for the optimal data-dependent triangulation is not possible except for trivial cases.

A subset of all possible triangulations with a fixed number of vertices is the set of all Delaunay triangulations. They are widely used in finite element methods (FEM) because of

their max-min-angle property, i.e., for a fixed set of vertices, there is no other triangulation in which the minimum angle of all triangles is bigger than in the Delaunay triangulation. This avoids long, skinny triangles which may lead to numerical instabilities. But for the purpose of approximating f with a piecewise linear interpolant, long, skinny triangles may be well suited, if f contains high-gradient regions indicative of feature boundaries (see [Rip92]). Therefore, the restriction to Delaunay triangulations may prevent the detection of the optimal triangulation.

Computer images are usually defined as discrete scalar fields $f : \Omega \rightarrow C$, with a domain $\Omega = \{0, \ldots, w-1\} \times \{0, \ldots, h-1\}$ as a regular grid of width w and height h, into a color space C, usually a three dimensional space, e.g., the RGB space. If the approximation is stored instead of the image f, storage space can be saved. The search for a good data-dependent triangulation can be used as a data compression method for images.

This paper provides an overview of algorithms to find good data-dependent triangulations to approximate images. It is organized as follows: Section 2 defines some categories of algorithm types that will be used later to classify the presented methods. Section 3 provides a short introduction into the general method of simulated annealing, a procedure to find a global optimum, or, at least a solution near it, within a very large search space. This method is applied to the specific problem of finding a near-optimal data-dependent triangulation to approximate an image later. Sections 4 to 6 present algorithms for the search for data-dependent triangulations. Section 7 presents some extensions the authors are working on.

2 Algorithm Classification

Finding the best triangulation for approximating an arbitrary scalar field is a non-linear optimization problem. An exhaustive search could be done using a backtracking algorithm that lists all possible triangulations systematically, calculates the approximation quality, and finds the optimum. In all non-trivial cases, the search space is too large to enumerate all triangulations within reasonable time.

Instead, algorithms can be used that iteratively select one triangulation out of a limited set of candidates. After a number of iterations, a good triangulation that has at least an approximation quality close to the optimum is found.

One class of algorithms are *greedy algorithms*. In each iteration, they always select the candidate that best approximates the scalar field. Every decision that is made is final, and is never taken back. Because of the non-linearity of the problem, they often converge to a local optimum, but fail to find the global optimum.

Furthermore algorithms can be divided into deterministic and non-deterministic (stochastic) algorithms. In *deterministic* algorithms, every decision is based on a unique predicate, and for the same input the algorithm produces the same result. *Stochastic* algorithms employ a (pseudo) random number generator for decisions. Even for the same input, the result may be different, but the probability of a result that has an extremely high approximation error is very small. Examples for stochastic algorithms are Monte Carlo methods, genetic

algorithms (GA), and simulated annealing methods (see Section 3).

3 The Principle of Simulated Annealing

Simulated annealing is a stochastic, iterative method to solve a global optimization problem, i.e., finding the global extremum s^* of a given function $f : D \to \mathbb{R}$ in a large search space D.

Starting with an arbitrary setting $s_0 \in D$, slightly modify s_0 to get $s'_0 \in D$. Using a probability function $p_{\text{accept}} : \mathbb{R} \times \mathbb{R} \times \mathbb{N} \to [0, 1]$, s'_0 is accepted with probability $p_{\text{accept}}(f(s_0), f(s'_0), 0)$, i.e., $s_1 = s'_0$, otherwise it is rejected, i.e., $s_1 = s_0$. Iterating the process of changing s_i to $s'_i \in D$ and accepting it with probability $p_{\text{accept}}(f(s_i), f(s'_i), i)$ yields a sequence of settings s_i which converges to the global extremum s^* under certain assumptions on p_{accept} [KCDGV83].

In order to minimize f the probability function p_{accept} should satisfy the following properties, for $a, b, c \in f(D)$:

- A setting that reduces f has a larger probability to be accepted than a setting that increases f:
$$c > a > b \Rightarrow p_{\text{accept}}(a, b, i) > p_{\text{accept}}(a, c, i).$$
- For the sequence to converge, the probability for accepting settings that increase f must converge to zero, whereas the probability for accepting settings that reduce f must not converge to 0:
$$\lim_{i \to \infty} p_{\text{accept}}(a, b, i) = \begin{cases} 0 & \text{for } b > a \\ \text{const} \in]0, 1] & \text{for } b \leq a \end{cases}.$$

For example, for a greedy method the probability function

$$p_{\text{accept}}(a, b, i) = \begin{cases} 0 & \text{for } b \geq a \\ 1 & \text{for } b < a \end{cases}$$

is used that accepts new settings only if they improve the result. This method can get stuck in a local minimum. A better choice for p_{accept} is

$$p_{\text{accept}}(a, b, i) = \begin{cases} \exp\left((a - b)/\tau_i\right) & \text{for } b \geq a \\ 1 & \text{for } b < a \end{cases}, \tag{1}$$

where $\tau_i = \tau_0 \tau_{\text{b}}^i$ is a temperature that starts at an initial value τ_0 and decreases exponentially to zero by a factor $\tau_{\text{b}} \in [0, 1]$ in every iteration. Since also settings that increase f might be accepted, the sequence can escape a local minimum.

The temperatures τ_0 and τ_{b} define the annealing schedule. Their choice is vital for the result of the optimization process. If τ_i decreases too rapidly, the sequence can get stuck in

a local minimum; if it decreases too slowly, the sequence converges to a better local minimum (and possibly the global minimum), but it requires one to perform more iterations to reach it. It can be shown that by choosing the right annealing schedule the probability for finding the global minimum converges to one, but this usually implies a large number of iterations [KCDGV83].

4 Refinement Algorithms

Refinement algorithms start with a very coarse triangulation with a small number of triangles. Iteratively they insert more vertices and triangles, refining the triangulation. The set of candidate triangulations for the next iteration is the set of all triangulations that can be created from the current triangulation by inserting one vertex at all possible positions. Since the number of possible positions can be high (or even infinite, if the domain is infinite), heuristics are employed to limit the number of insertion positions.

The greedy refinement algorithm presented in [GH95] inserts vertices into a Delaunay triangulation. This algorithm can be used to create high-quality approximations of height fields, but it can easily be generalized to work on arbitrary scalar fields. Starting with a simple triangulation of the domain, consisting of just two triangles for a rectangular domain, in every step a new vertex is inserted at the position of the largest distance between the approximation and the scalar field. So, the set of candidates to select the next triangulation from consists of only one triangulation. This procedure is repeated until a specified error condition is met. Although specified for height fields, Garland and Heckbert also applied their algorithm to gray-scale images, showing that the approximation looks much better using their triangulation instead of a triangulation that distributes the same number of vertices uniformly over the domain.

Furthermore they modified their algorithm, dropping the Delaunay constraint for the triangulation, and using a locally optimal data-dependent triangulation instead, further improving the results.

The approach discussed in [SHB$^+$01] is similar. Schätzl et al. are using an error norm derived from the Sobolev norm to calculate the distance between the approximation and the original data, which emphasizes the regions of high curvature. In each iteration, they subdivide the triangle with the largest error. For this triangle, they detect "significant points" (data sites with high distance to the triangulation) near the midpoints of the edges and insert these into the triangulation, subdividing the selected triangle and its neighbors.

Since both approaches are greedy algorithms, they tend to get stuck in local optima. The approach described in [Ped01] attempts to improve this behavior. It uses Delaunay triangulations, and calculates a first triangulation by adding vertices at the site with largest distance from the current triangulation, just in the same way as in [GH95]. After falling below an error threshold, another greedy method is used to remove vertices, until the threshold is exceeded again. Some iterations of these refinement and decimation procedures are performed. This way, an approximation with fewer triangles is found that also meets the error threshold. Often the iterations quickly get stuck, adding exactly those ver-

tices in the refinement step that were removed in the preceding decimation step, limiting the improvement of this approach over [GH95]. Its restriction to Delaunay triangulations is another drawback.

5 Decimation Algorithms

Decimation algorithms start with a very fine triangulation with a lot of triangles, and iteratively remove elements from it. The set of candidates to select the next triangulation is the set of all triangulations that can be created from the current triangulation by removing one element.

One example of a greedy decimation algorithm are the progressive meshes, described in [Hop96]. Although defined for 2-manifolds, in [Hop96] it is also applied to images. Starting with a full triangulation that has a vertex at every pixel position of the image, one vertex after the other is removed using the edge collapse operation (see Fig. 1). From the set of triangulations that result from collapsing one of the edges of the current triangulation, the one with the best approximation quality is selected. A priority queue can be used to optimize this selection process: For every edge e_i of the triangulation the change in approximation quality $\Delta(e_i)$ is computed and stored. The edge e with the smallest change ($\forall i : \Delta(e) \leq \Delta(e_i)$) is collapsed. $\Delta(e_i)$ has to be recomputed only for those e_i that are incident to a triangle that was changed by the edge collapse operation.

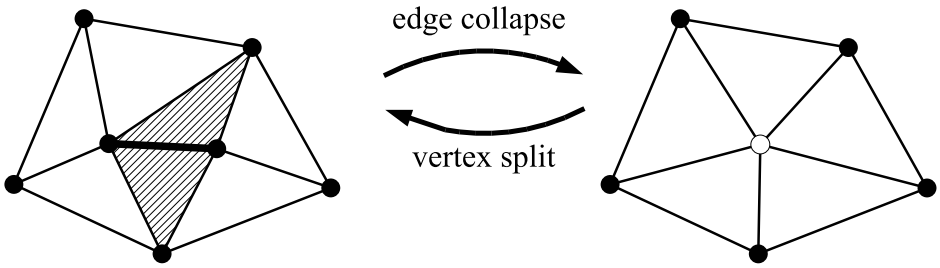

Figure 1: An edge collapse operation. The thick edge is collapsed to the unfilled vertex, the marked triangles are removed.

One example of the application of this approach to an image is shown in Fig. 4(f) (and reproduced in color on p. 12), the numerical results are presented in Table 1.

The inverse operation of the edge collapse is the vertex split. If the information for reversing the edge collapse is stored in a list, a *progressive mesh* can be defined as a series of triangulations, each having one vertex more than the previous one. This data structure can be used to provide a fine-grain set of triangulations with different approximation quality, and also other methods like selective refinement are possible.

Another example of a decimation strategy is the *adaptive thinning* (AT), proposed in [DDFI04] (and later improved in [DDI06]) that works on Delaunay triangulations. Start-

ing with a dense Delaunay triangulation with a vertex at every pixel position, iteratively the vertex that increases the approximation error least (the so-called least significant vertex) is removed. After every removal the Delaunay criterion is restored. Because of the restriction to Delaunay triangulations, the result for many images is not as good as with using arbitrary triangulations, as [Rip92] shows. Furthermore the greedy structure of the algorithm tends to step into local minima, missing a better solution with the same number of vertices.

6 Modification Algorithms

The class of *modification algorithms* starts with an arbitrary initial triangulation, and improves the approximation quality by performing a number of modification operations. The algorithms differ in the set of modification operations they utilize and the type of decision to determine what operation to perform next.

A modification algorithm that extends adaptive thinning (AT) (see Section 5) was proposed in [DI06]. It uses a modification operation called *exchange operation* (E), which is defined by a pair (v, p), where v is a vertex of the Delaunay triangulation, and p is a pixel that is *not* a vertex. The exchange operation removes the vertex v from the Delaunay triangulation and adds a new vertex at position p. The input to this algorithm is a Delaunay triangulation produced by the AT algorithm. To this triangulation, a series of exchange operations is applied. In every iteration, the exchange operation (v, p) that decreases the approximation error most is executed. The algorithm stops when there is no exchange operation decreasing the error any more. Then, every vertex of the resulting triangulation is locally optimal in the sense that removing it and adding another vertex will always increase the approximation error. Nevertheless, the globally optimal Delaunay triangulation may be different, because removing a set of vertices and inserting the same number of others may still decrease the approximation error. Furthermore, this algorithm is still restricted to Delaunay triangulations.

In [Law77] the theoretical basis for algorithms that use an edge swap as the only modification operation was presented. An edge swap operation replaces two triangles (v_a, v_b, v_c) and (v_b, v_d, v_c) that share the edge (v_b, v_c) and that form a convex quadrilateral with the triangles (v_a, v_b, v_d) and (v_a, v_d, v_c) that share the other diagonal (v_a, v_d) of the quadrilateral (see Fig. 2). This operation keeps the number and position of the vertices fixed, and only modifies their connectivity. It is discussed in [Law77], what conditions must be met to guarantee the validity of the triangulation after performing the edge swap operation. Furthermore, the following general algorithm is discussed in detail: From the set of edges E an edge $e \in E$ is selected that fulfills a specified predicate. The edge e is swapped, and this procedure is repeated until there is no $e \in E$ that fulfills the predicate. It is shown in [Law77] that this algorithm always terminates when the predicate is the circumcircle criterion, and that the final triangulation is the Delaunay triangulation of the sites.

In [DLR90] a set of fixed samples (vertices) of a scalar function $f : \mathbb{R}^2 \to \mathbb{R}$ is given, and the triangulation T connecting these vertices is sought, so that the piecewise linear

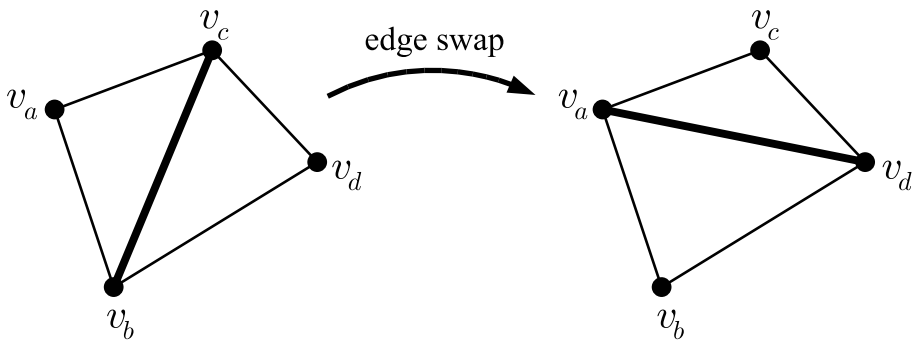

Figure 2: An edge swap. The thick edge is swapped.

approximation induced by T fits f best. Different cost functions $c_i : \mathcal{T} \rightarrow \mathbb{R}^+$ (with $\mathcal{T}$ is the set of all possible triangulations of the set of vertices) are defined that judge the quality of the representation. To find the best triangulation with respect to a cost function c, an arbitrary initial triangulation T_0 is improved iteratively: An edge of the triangulation T_i is chosen randomly, and if swapping that edge (if possible) reduces $c(T)$, it is swapped in T_{i+1}. This edge is then called "locally optimal." This procedure is repeated until every edge in the triangulation is locally optimal. An edge may be swapped several times, because swapping an edge of an incident triangle can lead to the edge being not locally optimal any more. The algorithm is guaranteed to stop, because every swapped edge reduces the cost function c which has by definition a lower bound of 0, and the set of triangulations $\mathcal{T}$ is finite. If f is known, $c(T)$ can be chosen to be the L_2 distance between f and the approximation induced by T. This cost function achieves results that are superior to cost functions that do not take f into account and only use the geometry of the triangles. Since this method is a greedy method, only making moves that reduce the cost function, it easily falls into a local minimum of the cost function, where all the edges are locally optimal, but which is not the global minimum of the cost function. Furthermore, the result depends on the order of edges that are tested for swapping.

In [Sch93], a simulated annealing approach (see Section 3) is used to improve the results of [DLR90] and to find a lower local (and hopefully the global) minimum of the cost function.

The methods in [Law77], [DLR90], and [Sch93] all have the number and position of the vertices fixed. Even better approximations can be found, if also the position of the vertices can be modified during the optimization procedure. In [KH01] a single simulated annealing approach is used to optimize both the vertex positions and their connectivity at the same time. In addition to the edge swap operation, a vertex move operation is used to change the position of one vertex of the triangulation (see Fig. 3). For each iteration of the simulated annealing loop the type of operation (edge swap or local vertex move) is chosen randomly. The algorithm can be applied to scattered data problems, and also to color images.

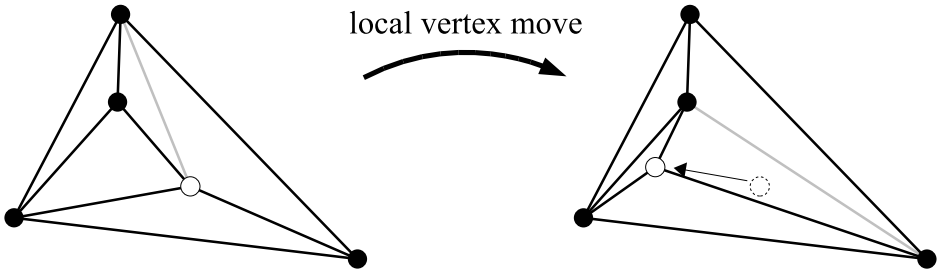

Figure 3: A local vertex move. The unfilled vertex is moved, the gray edge is swapped to prevent a degenerate triangle.

The approach of [PK03] specializes on images. It simplifies the modification operations of [KH01]. Furthermore, it uses the following greedy refinement strategy to start with a good initial triangulation: Starting with an elementary triangulation of the image domain, consisting of two triangles, the next site is added within the triangle with the largest approximation error. The position of the inserted site is the error barycenter of the pixels of that triangle, i.e., the average of all pixel coordinates weighted by their approximation error. This process is iterated until the specified number of sites is reached.

Further improvements were implemented by the authors and described in [LUH07]. In that approach, additionally to the position and connectivity of the vertices, also the number of triangles in the triangulation is modified. For this, a total approximation error is specified by the user, and the algorithm adds or removes vertices during the simulated annealing process to meet this error as good as possible. So this approach is the first to include all three factors that have an influence to the approximation quality into the optimization process. The approximation quality is measured using the CIEL*a*b* color space (see [WS82]) that takes the reception of the human eye into account.

Furthermore, we implemented so-called *guides*, heuristics to improve the convergence of the simulated annealing. They assign higher probabilities to edges or vertices in regions of high approximation error for being selected for an edge swap or a vertex move operation, respectively. This concentrates the computation to regions where a modification operation is more likely to improve the approximation. Experiments show that the employment of these guides reduce the number of iterations required to achieve a specified approximation error by a factor of eight.

We compared the compression ratio of this method with the well-known image compression standards JPEG and JPEG2000. We showed that the image compression method using a data-dependent triangulation can compete with these standards for a large variety of color images. Fig. 4 (also reproduced in color on p. 12) shows the results of image compression using different data-dependent triangulations (Figs. 4(b), 4(c) and 4(f)), with JPEG (Fig. 4(d)), and with JPEG2000 (Fig. 4(e)). All compressed images have a file size of $\sim 5\,750$ bytes.

Table 1 shows some numerical results for the images in Fig. 4. The approximation error is measured as the RMSE error of the approximation in the CIEL*a*b* color space (called

Lab in the table). Smaller values correspond to a better approximation quality. Only for comparison of our method to the method of [PK03] we measured the error in the RGB color space that was employed in the original paper. The method of [LUH07] has the best approximation quality of the five methods. Especially JPEG compression leads to poor results for this high compression ratio of 1:140 (0.18 bits per pixel).

(a) Lena image. (b) Method of [LUH07]. (c) Method of [PK03].

(d) JPEG. (e) JPEG2000. (f) Progressive Meshes [Hop96].

Figure 4: Original image (∼770 kB, courtesy USC SIPI) (a), and approximations (∼5 750 bytes) using Simulated Annealing [LUH07] (b), Simulated Annealing [PK03] (c), JPEG (d), JPEG2000 (e), and Progressive Meshes [Hop96] (f), reproduced in color on p. 12

	Sim. Anneal. [PK03]	Sim. Anneal. [LUH07]		Prog. Meshes [Hop96]	JPEG	JPEG 2000
Colormodel	RGB	RGB	Lab	Lab	Lab	Lab
RMSE	17.26	16.45	6.00	6.16	10.38	6.81

Table 1: Numerical results of approximation error (Root Mean Square Error, RMSE) for different algorithms for data-dependent triangulations, and for JPEG and JPEG2000 compression.

When comparing the approximations of the data-dependent triangulations to JPEG and JPEG2000, the triangulations reveal another advantage when it comes to decompressing and displaying the image: Decompression of JPEG and JPEG2000 images is usually done on the CPU, and the color values of every pixel have to be transferred to the graphics card,

which often is a bottleneck. But even low-cost graphics cards provide acceleration methods to render a triangle onto the screen while at the same time linear interpolating the color at its vertices. (For example, this can be used for Gouraud shading of triangle meshes.) Therefore, only the vertex positions and their colors have to be transferred to the graphics card, which is only a fraction of the data of the complete image. Decompressing and displaying a compressed image is therefore very efficient even on todays low-cost graphics cards. Furthermore, affine transformations can easily be applied to a data-dependent triangulation by just transforming the coordinates of every vertex. Scaling, rotating, and shearing can easily be applied to an image, whereas every pixel needs to be transformed when dealing with a JPEG or JPEG2000 image.

7 Work in Progress

The authors are currently working on extending the concept introduced in [LUH07] to video clips and video streams. This extended concept is based on applying the image approximation technique to the frames of a video clip, and gaining a speed-up by using the data-dependent triangulation of one frame as initial triangulation for the next one.

Furthermore, they extend the concept of [LUH07] to tetrahedrizations for approximating a scalar field defined over a 3-dimensional domain. This could be applied to a video clip or stream, where two dimensions are the x- and y-coordinates of the image, and the third dimension is time, i.e., the frame index.

Acknowledgments

This work was supported by the DFG IRTG 1131, "Visualization of Large and Unstructured Data Sets," University of Kaiserslautern, and NSF contract ACI 9624034 (CAREER Award) and a large ITR grant. We thank the members of the Visualization and Computer Graphics Research Group at IDAV, and the Geometric Algorithms Group at the University of Kaiserslautern.

References

[DDFI04] Laurent Demaret, Nira Dyn, Michael S. Floater, and Armin Iske. Adaptive thinning for terrain modelling and image compression. In Neil A. Dogson, Michael S. Floater, and Malcolm A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, pages 321–340, Heidelberg, 2004. Springer-Verlag.

[DDI06] Laurent Demaret, Nira Dyn, and Armin Iske. Image compression by linear splines over adaptive triangulations. *Signal Processing*, 86(7):1604–1616, 2006.

[DI06] Laurent Demaret and Armin Iske. Adaptive image approximation by linear splines

over locally optimal delaunay triangulations. *Signal Processing Letters, IEEE*, 13(5):281–284, May 2006.

[DLR90] Nira Dyn, David Levin, and Shmuel Rippa. Data Dependent Triangulations for Piecewise Linear Interpolations. *IMA J. of Numerical Analysis*, 10(1):137–154, Jan 1990.

[GH95] Michael Garland and Paul Heckbert. Fast Polygonal Approximation of Terrains and Height Fields. Technical report, CS Department, Carnegie Mellon University, September 1995.

[Hop96] Hugues Hoppe. Progressive meshes. In *SIGGRAPH '96*, pages 99–108, 1996.

[KCDGV83] Scott Kirkpatrick, Jr. Charles D. Gelatt, and Mario P. Vecchi. Optimization by Simulated Annealing. *Science Magazine*, pages 671–680, may 1983.

[KH01] Oliver Kreylos and Bernd Hamann. On Simulated Annealing and the Construction of Linear Spline Approximations for Scattered Data. *IEEE TVCG*, 7(1):17–31, 2001.

[Law77] Charles L. Lawson. Software for C^1 surface interpolation. In *Mathematical Software III*, pages 161–194. Academic Press, New York, 1977.

[LUH07] Burkhard Lehner, Georg Umlauf, and Bernd Hamann. Image Compression Using Data-dependent Triangulations. In George Bebis et al., editor, *International Symposium on Visualization and Computer Graphics (ISVC) 2007, Part I, LNCS 4841, to appear*, Lecture Notes on Computer Science, pages 351–362. Springer, 2007.

[Ped01] Hélio Pedrini. An improved Refinement and Decimation Method for Adaptive Terrain Surface Approximation. In *Proceedings of WSCG*, pages 103–109. Czech Republic, 2001.

[PK03] Vid Petrovic and Falko Kuester. Optimized Construction of Linear Approximations to Image Data. In *Proc. 11th Pacific Conf. on Comp. Graphics and Appl.*, pages 487–491, 2003.

[Rip92] Shmuel Rippa. Long and thin triangles can be good for linear interpolation. *SIAM J. Numer. Anal.*, 29(1):257–270, 1992.

[Sch93] Larry L. Schumaker. Computing Optimal Triangulations Using Simulated Annealing. *Computer Aided Geometric Design*, 10(3-4):329–345, 1993.

[SHB$^+$01] René Schätzl, Hans Hagen, James Barnes, Bernd Hamann, and Kenneth Joy. Data-Dependent Triangulation in the Plane with Adaptive Knot Placement. In Guido Brunnett, Hanspeter Bieri, and Gerald Farin, editors, *Geometric Modelling, Comp. Suppl. 14*, pages 199–218. Springer, 2001.

[WS82] Günter Wyszecki and Walter Stanley Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. Wiley-Interscience, 1982.

(a) Lena image.

(b) Method of Lehner et al.

(c) Method of Petrovic et al.

(d) JPEG compressed image.

(e) JPEG2000 compressed image.

(f) Progressive Meshes of Hoppe.

Figure 5: Benchmark image “Lena” (courtesy USC SIPI), and compressed versions.