Image Compression Using Data-Dependent Triangulations

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 method to speed up the computation of a high-quality data-dependent triangulation approximating an image using simulated annealing by probability distributions guided by local approximation error and its variance. The triangulation encodes the image, yielding compression rates comparable to or even superior to JPEG and JPEG2000 compression.

The specific contributions of our paper are a speed-up of the simulated annealing optimization and a comparison of our approach to other image approximation and compression methods. Furthermore, we propose an adaptive vertex insertion/removal strategy and termination criteria for the simulated annealing to achieve specified approximation error bounds.

1 Introduction

Storing and transmitting images often requires large amounts of memory or high bandwidths. Good compression algorithms are necessary to reduce these bottlenecks. Lossless compression of images does not provide the necessary compression rates. Therefore, often lossy compression algorithms are used that achieve high compression rates, but cannot reproduce the image data exactly.

The approach presented in this paper uses a piecewise linear, C^0 approximation of the image by a triangulation of the image domain. Storing or transmitting this triangulation requires only little space or bandwidth. It has the advantage that affine transformations of the image can be performed very efficiently by applying the transformation to the vertices of the triangulation only. Therefore, the encoded image can easily be displayed on screens of different resolutions. Small screens of mobile phones or other hand-held devices and large screens like power walls can be used. Since the approximation is a C^0-continuous vector based format, scaling the image to a large resolution does not cause sawtooth distortions. Smoothing of the image is not necessary.

Computing a data-dependent triangulation as high-quality approximation, i.e. low approximation error, of an image is a hard problem, which can be solved using simulated annealing (SA). However, this requires a large number of local, possibly rejected modifications of the triangulation, causing SA to converge

G. Bebis et al. (Eds.): ISVC 2007, Part I, LNCS 4841, pp. 351–362, 2007.

slowly. To speed it up we propose modified probability distributions for the atomic modifications guided by the local approximation error and its variance.

In Section 2 we review related work on piecewise linear approximation and image compression. We outline in Section 3 the basic notation and introduce in Section 4 the concept of SA. In Section 5 we show how this concept is used to approximate images. Here, also the concept of adapted probability distributions, so-called guides, methods to control the number of triangles adaptively, and termination conditions for SA are discussed. In Section 6, we present experimental results of our method compared to other approaches.

2 Related Work

In 3D a 2-manifold can be approximated by a triangle mesh. In [1] an initial mesh is modified optimizing vertex positions and their connectivity to minimize a global approximation error. In [2] a fine mesh is simplified by collapsing edges according to a local criterion. Both methods cannot be applied to images, since they only work on the geometry of the mesh and not on attributes such as color.

Piecewise linear approximation of a 2D scalar field induced by a triangulation requires an error norm to measure the approximation quality. Computing a low-error triangulation is usually done iteratively. *Greedy triangulations* (GT) are computed by minimizing the error in every step. Since they are relatively simple, they are fast, but may fail to find a good approximation because of local minima.

In [3,4] a *greedy refinement strategy* is used. Starting with a Delaunay triangulation vertices are inserted at pixels of maximum L^∞-error or Sobolev-error, emphasizing regions of large changes in the image. The vertices are inserted by subdividing the corresponding triangle.

A *greedy decimation strategy* works by removing vertices from a fine initial triangulation, e.g., progressive meshes [5]. Although defined for 2-manifolds, in [5] it is also applied to color images. The error is measured by the L^2-norm, and a vertex is removed by an edge collapse. The method in [6] is similar, but measures approximation quality also by the appearance of the mesh from different viewpoints. Applied to color images it yields the same method as [5].

The method presented in [7] exhausts the space of valid triangulations more thorough, finding better approximations at higher computational costs. It alternates between refinement and decimation phases. However, it can get caught in infinite loops, e.g., alternating between removal and insertion of the same vertex.

Another class of algorithms works only with *edge flips* keeping the number of vertices fixed. In [8] the error is measured by the L^2-norm. Starting with an arbitrary triangulation, iteratively a random edge is chosen and flipped, if that reduces the error. As every greedy strategy, this may lead to situations where the algorithm gets stuck in local minima. *Simulated annealing* is used in [9] to improve these results. By also allowing for edge flips that increase the error with a slowly decreasing probability, better approximations can be found. In [10] this approach is extended by operations changing the position of vertices. Specialized to images, [11] simplifies these operations and adds a

greedy refinement strategy to improve the initial triangulation. Our method further improves the performance and results of these approaches.

The main focus of methods to create vector images from raster images is editability of the output and not compression. A general framework for this transformation is described in [12]. Edge detection followed by a constrained Delaunay triangulation is used to split the image into triangles of uniform color. Incident triangles of similar color are combined to polygons. It yields C^{-1} approximation of the image. In [13] also the color gradient of every triangles is estimated, and triangles with similar color gradients are combined, but only yielding a C^{-1} approximation at polygon borders.

Most image compression algorithms are associated with a file format. The GIF and PNG file formats [14] store an image using lossless compression. On natural images both formats usually reach compression ratios of only 0.5 or less. The JPEG format [14] uses a lossy compression. For encoding the image is partitioned into square blocks, transformed into the frequency domain using the discrete cosine transform (DCT), and small coefficients are dropped. The JPEG2000 format [15] does not partition the image and uses a wavelet transformation instead of the DCT. For both formats the user can select a threshold for small coefficients, trading off between file size and image quality. Because our method is also lossy, we use these two formats for comparison.

3 Notation

An image $I : \Omega \to \mathbb{R}^3$ of width w and height h maps every pixel of $\Omega = \mathbb{Z}_w \times \mathbb{Z}_h$ to a color. This color is represented by L, a and b of the CIEL*a*b* color model [16], where color differences, as they are perceived by the human eye, can be measured as the L^2 distance of two colors.

A triangulation $t = (V, E, F)$ consists of a set of vertices $V = \{v_1, \ldots, v_n\} \subset \Omega$, edges $E \subset V^2$, and triangles $F \subset V^3$. The convex hull of $M \subset \mathbb{R}^2$ is denoted by $CH(M)$. A triangulation is valid, if it covers $CH(\Omega)$. We denote by T the set of all valid triangulations t of Ω. The set of triangles incident to an edge e is $F(e) = \{f \in F | e \in f\}$, the set of triangles incident to a vertex v is $F(v) = \{f \in F | v \in f\}$, and $P(f) = \Omega \cap CH(f)$ is the set of pixels within triangle f. The approximation $A_t : \Omega \to \mathbb{R}^3$ induced by t is the piecewise linear interpolant of I at the vertices. For the approximation we measure the error per pixel $p = (x, y)$, triangle f, edge e, vertex v, and set of edges $S_e \subset E$ or vertices $S_v \subset V$

$$\Delta(p) = (I(p) - A_t(p))^2, \qquad \Delta(f) = \sum_{p \in P(f)} \Delta(p), \qquad \Delta(e) = \sum_{f \in F(e)} \Delta(f),$$

$$\Delta(v) = \sum_{f \in F(v)} \Delta(f), \qquad \Delta(S_e) = \sum_{e \in S_e} \Delta(e), \qquad \Delta(S_v) = \sum_{v \in S_v} \Delta(v).$$

The total error used to measure the global approximation quality is

$$\Delta_{\mathrm{G}} = \sqrt{\sum_{(x,y) \in \Omega} \Delta(x, y)/(h \cdot w)}.$$

4 The Principle of Simulated Annealing

Simulated annealing is a stochastic, iterative method to find the global extremum s^* of a given function $f(s) : D \to \mathbb{R}$ in a large search space D. It is based on the following approach: Starting with an arbitrary setting $s_0 \in D$, slightly modify s_0 to get $s'_0 \in D$. Using a probability distribution $p_{\text{accept}} : \mathbb{R}^2 \times \mathbb{N} \to [0,1]$, 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 yields a sequence of settings s_i which converges to the global extremum s^* under certain assumptions on p_{accept} [17],

$$p_{\text{accept}}(f(a), f(b), i) = \begin{cases} \exp\left((f(a) - f(b))/\tau_i\right) & \text{for } f(b) > f(a) \\ 1 & \text{for } b \leq a \end{cases}, \qquad (1)$$

where $\tau_i = \tau_0 \tau_{\text{base}}^i$ is a temperature initialized with τ_0, that is decreased to zero by a factor $\tau_{\text{base}} \in]0,1[$ in every iteration. Since settings that increase f might be accepted depending on (1), the sequence of s_i can escape local minima. The temperatures τ_0 and τ_{base} define the annealing schedule. If τ_i decreases too fast, the sequence can get stuck in a local minimum. If it decreases too slowly, the sequence converges to a better local minimum using more iterations. It can be shown that for the right annealing schedule the probability to find the global minimum converges to one, usually requiring a large number of iterations [17].

5 Simulated Annealing for Image Approximation

5.1 Basic Simulated Annealing (BSA)

The basic approach to find an approximation A_t for an image I is based on [10,11]. To find t minimize $\Delta_{\text{G}}(t)$ for all $t \in T$ using (1), where τ_0 and τ_{base} are user-defined parameters. There are three modifications to generate t'_i from t_i:

Edge Flips. change mesh connectivity: If the union of two triangles $\{v_a, v_b, v_c\}$, $\{v_b, v_c, v_d\}$ is convex, they are replaced by $\{v_a, v_b, v_d\}$, $\{v_a, v_d, v_c\}$, see Fig. 1 and [8].

Local Vertex Moves. change vertex positions locally: A vertex of t_i is assigned a new position in its vicinity while changing the local connectivity as little as possible without generating degenerate triangles, see Fig. 2 and [10].

Global Vertex Moves. change vertex distribution globally in two steps:

a) A vertex is removed from t_i and the resulting hole is triangulated.
b) A new vertex $v \notin t_i$ is inserted to t_i either by splitting the triangle that contains v, or by splitting the two adjacent triangles into two triangles, if v lies on an edge of t_i, and a locally optimal data-dependent triangulation for v using the method in [8] is applied.

Then, one iteration of BSA consists of five steps:

1. Select one modification at random, with probability p_f for an edge flip, p_l for a local vertex move, and $p_g = 1 - p_f - p_l$ for a global vertex move.

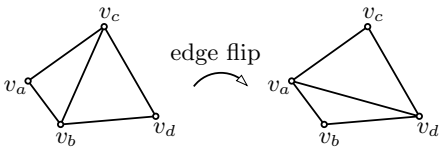


Fig. 1. An edge flip

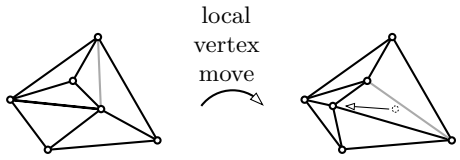


Fig. 2. A local vertex move. The gray edge is flipped to prevent a degenerate triangle.

2. Select the edge for the edge flip or the vertex for the local/global vertex at random, where every edge or vertex has the same uniform probability.
3. If the selected modification is a vertex move, select the new vertex position at random, where every new position has the same uniform probability.
4. Generate t_i' from t_i using the selected modification as determined in 1.-3.
5. The modified triangulation t_i' is accepted or rejected using (1), i.e.,

$$t_{i+1} = \begin{cases} t_i' & \text{with probability } p_{\text{accept}}(\Delta_{\text{G}}(t_i), \Delta_{\text{G}}(t_i'), i), \\ t_i & \text{if } t_i' \text{ is rejected.} \end{cases}$$

The initial triangulation t_0 is computed similar to that in [11]: Starting with the four corner vertices, we repeatedly insert a new vertex v into $f \in F$ with largest error $\Delta(f)$ at its error barycenter, but instead of using a Delaunay criterion as [11], we construct a locally optimal data-dependent triangulation for v using the method in [8].

5.2 Guided Simulated Annealing (GSA)

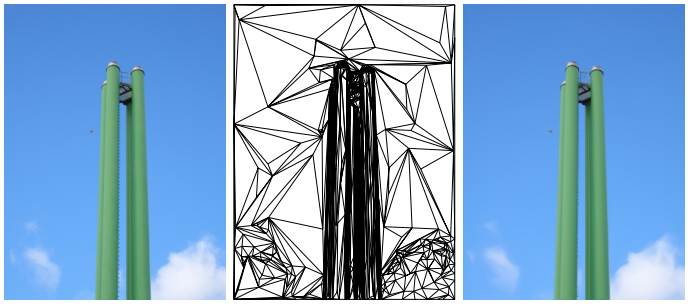
The BSA finds a high-quality data-dependent approximation to an image with a large number of iterations. Therefore, we adapt the probability distributions, so-called guides, to speed up BSA: In steps 2. and 3. we select edges, vertices and new positions in regions of large approximation error with a higher probability. We use edge/vertex guides for the selection of edges and vertices and position guides for the selection of new vertex positions.

Edge Guide. Denote by $E_{\text{flippable}}$ the set of edges incident to two triangles forming a convex quadrilateral. To prefer edges in regions of large approximation error, $e \in E_{\text{flippable}}$ is selected with probability

$$p_{\text{flip}}(e) = \Delta(e)/\Delta(E_{\text{flippable}}). \tag{2}$$

Local Vertex Guide. Denote by V_{movable} the set of all vertices except the four corners of an image. To prefer vertices in regions of large approximation error, $v \in V_{\text{movable}}$ is selected with probability

$$p_{\text{local}}(v) = \Delta(v)/\Delta(V_{\text{movable}}). \tag{3}$$



(a) Original image. (b) Triangulation. (c) Approximation.

Fig. 3. Original image (550 kB) (a), the triangulation (711 vertices) (b), the approximate image (∼4 kB) (c)

Global Vertex Guide. For the global move we have to remove an "unimportant" vertex changing the approximation error only a little. For this we use the variance $\mathrm{Var}(v)$ of the gradient of A_t for the triangles incident to a vertex v. Within each triangle f of A_{t_i} the color gradient $g(f) = \nabla\left(A_{t_i}\right)|_f$ is constant. So, the mean gradient and its variance for v are defined as

$$\bar{g}(v) = \frac{1}{|F(v)|} \sum_{f \in F(v)} g(f) \quad \text{and} \quad \mathrm{Var}(v) = \frac{1}{|F(v)|} \sum_{f \in F(v)} (g(f) - \bar{g}(v))^2.$$

If $\mathrm{Var}(v)$ is small, the color gradients of the triangles incident to v are similar. Removing v and triangulating the hole leads to new triangles with a similar color gradient, changing approximation quality only a little. So, we use $w(v) = (\mathrm{Var}(v)+\varepsilon)^{-1}, \varepsilon > 0$, to guide the removal and select $v \in V_{\text{movable}}$ for a global move with probability

$$p_{\text{global}}(v) = w(v) \Big/ \sum_{v \in V_{\text{movable}}} w(v). \tag{4}$$

For the new vertex position we prefer a position with a large approximation error $\Delta(x, y)$. The position guides for the local and the global move only differ in the set F_{select} of possible triangles:

$$F_{\text{select}} = \begin{cases} F & \text{for a global move,} \\ F(v) \cup \{f \in F | f \cap F(v) \in E\} & \text{for a local move.} \end{cases}$$

The new position is selected using the following guide:

Position Guide

a) To prefer triangles with a large approximation error, select a triangle $f_{\text{select}} \in F_{\text{select}}$ with probability

$$p_{\text{triselect}}(f) = \Delta(f)/\Delta(F_{\text{select}}). \tag{5}$$

b) To prefer positions with a large approximation error, select the new position $\omega_{\text{select}} \in f_{\text{select}}$ with probability

$$p_{\text{posselect}}(\omega) = \Delta(\omega)/\Delta(f_{\text{select}}). \tag{6}$$

5.3 Acceleration Trees

Using guides leads to less iterations to achieve highly similar approximation error, see Table 1. But computing the guides requires additional effort increasing the total computation time. Every guide involves computing weights for all vertices/edges/triangles from a set of candidates, see (2) – (5), and for the selection the sum of weights normalizes the probabilities. The complexity for computing these sums can be reduced by storing the weights in a binary tree, the so-called acceleration tree, and updating only those that are changed by a modification with $\mathcal{O}(\log n)$. All leaves have the same depth $\lceil \log(n) \rceil$ and contain the weights of the vertices/edges/triangles. The inner nodes contain the sum of their siblings, i.e., the sum of all leaves of the corresponding subtree. If the number of leaves is not a power of two, some inner nodes have no right sibling. The root contains the sum of all weights. Fig. 4 shows an example of an acceleration tree.

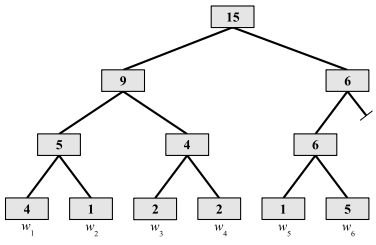


Fig. 4. An example of the acceleration tree for six vertices/edges/triangles

To select a vertex/edge/triangle according to its probability, we use a uniformly distributed random number $r \in [0, 1[$ and multiply it with the sum of weights in the root of the tree and trace it down as follows: Denote by w_{left} and w_{right} the weight of the left and right sibling of the current node. If $r < w_{\text{left}}$, we proceed with the left sibling; if $r \geq w_{\text{left}}$, we set $r = r - w_{\text{left}}$ and proceed with the right sibling. We repeat this until we reach a leaf, and select the corresponding vertex/edge/triangle. This also requires only $\mathcal{O}(\log n)$ operations.

A separate acceleration tree instance is used for each of the following guides:

Edge Guide. The leaves store $\Delta(e)$ for every $e \in E_{\text{flippable}}$.
Local Vertex Guide. The leaves store $\Delta(v)$ for every $v \in V_{\text{movable}}$.
Global Vertex Guide. The leaves store $w(v)$ for every $v \in V_{\text{movable}}$.
Position Guide. The leaves store $\Delta(f)$ for every $f \in F$ for global vertex moves.

We do not use acceleration trees for local vertex moves, because F_{select} contains only a small number of triangles and does not grow linearly with the number of triangles or vertices of the triangulation. An acceleration tree cannot be used to speed up the selection of a pixel within a triangle for the second step of the position guide. Fig. 6 shows the speed-up that is gained by acceleration trees.

5.4 Adaptive Number of Vertices

In general, the more triangles used, the better the approximation. But it depends very much on the complexity, size of details and noise of the image how many triangles are needed to achieve a user-specified error. Therefore, also the number of vertices is changed during the simulated annealing procedure.

The approximant A_t is piecewise linear and we observed that the approximation error is $O(n^{-1})$, where n is the number of vertices. Thus, there exists a constant α with $\Delta_{\mathrm{G}} \approx \frac{\alpha}{n}$ for which a given error Δ_{goal} can be achieved by an estimated number of vertices $n_{\mathrm{goal}} \approx n\,\Delta_{\mathrm{G}}/\Delta_{\mathrm{goal}}$. To reach n_{goal} vertices, in every iteration additional vertices can be inserted or removed with probability

$$p_{\mathrm{change}} = n/m \cdot |1 - \Delta_{\mathrm{G}}/\Delta_{\mathrm{goal}}|\,,$$

where m is the number of iterations to reach n_{goal} for constant p_{change}. A vertex is inserted or removed by a variant of the Global Vertex Move:

- If $\Delta_{\mathrm{G}} \geq \Delta_{\mathrm{goal}}$ a new vertex is inserted by **Global Vertex Move** b) with position guides followed by the local optimal data-dependent triangulation.
- If $\Delta_{\mathrm{G}} < \Delta_{\mathrm{goal}}$ a vertex is removed by **Global Vertex Move** a) with the global vertex guide.

Note that insertion/removal of vertices can happen in every iteration before step 1., independently of the rest of the simulated annealing. Its acceptance probability is one, since a vertex removal always increases Δ_{G}.

5.5 Termination Conditions and Image Compression

There are several termination conditions for simulated annealing focussing on run-time, memory consumption or approximation quality:

Simulation Time. The iteration stops after a predefined time. It can be used in combination with the next two conditions.

Triangulation Complexity. Changing the number of vertices adaptively, the iteration stops after a predefined number of vertices or triangles is reached.

Specified Error. The iteration stops after reaching the predefined error Δ_{goal}. If Δ_{G} differs from Δ_{goal} only by $|1 - \frac{\Delta_{\mathrm{G}}}{\Delta_{\mathrm{goal}}}| < 0.03$ for example, only a fixed number of additional iterations $m_{\mathrm{post}} = \max(1000, 0.1m)$ are done where m is the number of iterations done so far. If during these iterations Δ_{G} differs from Δ_{goal} by more than 3%, the simulation continues.

For every approximation A_t we stored coordinates, i.e. $\lceil \log_2(|\Omega|) \rceil$ bits per vertex, and colors, i.e. 24 bits per vertex, for every vertex and the connectivity of the mesh with [18], i.e. on average two bits per triangle. Since only an approximation is stored, the compression is lossy. Experiments revealed that further compression of the color and coordinate components is hardly possible, so these are stored uncompressed. Using the second termination condition enables the user to set a limit for the required memory. For the third termination condition GSA uses as many vertices and triangles as needed to achieve Δ_{goal}.

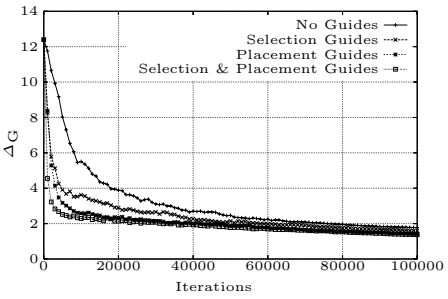


Fig. 5. Decrease of Δ_G using different combinations of guides

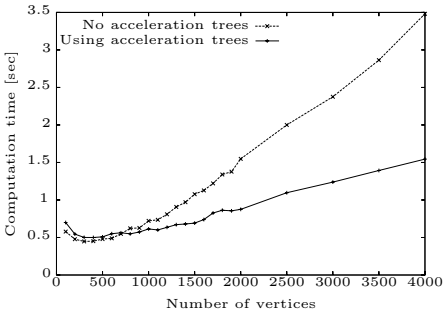


Fig. 6. Time for 500 iterations of simulated annealing for different numbers of vertices with/without acceleration trees

6 Results

All experiments were performed on a computer with an Intel Pentium 4PC, 2.66 GHz, 512 kB cache, 1 GB RAM.

Fig. 5 compares the approximation error as a function of number of iterations of the simulated annealing method for different combinations of guides. It shows that using guides does not corrupt the convergence of the simulated annealing method. The graphs show that the fastest decrease in approximation error is achieve by using all guides, and that placement guides have a higher impact than selection guides.

Fig. 6 compares the computation time of 500 iterations of GSA for Fig. 3(a) with and without acceleration trees for an increasing fixed number of vertices. It shows that for more than 700 vertices the overhead for the acceleration trees is compensated by the faster computation of the weights. For images with high complexity and many vertices the acceleration trees give a significant speed-up.

Table 1. Δ_G for Fig. 7 for different methods

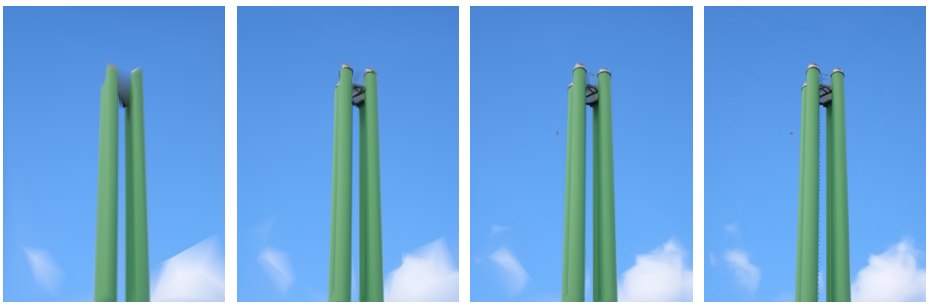
	SA [11]	GSA		GT [5]	JPG	JPG 2000
Color model	RGB	RGB	Lab	Lab	Lab	Lab
Δ_G	17.26	16.45	6.00	6.16	10.38	6.81

For comparison Fig. 7 shows approximations of the Lena image (512 × 512 pixels, 786 440 bytes) using five different methods as listed in Table 6. The file size of the compressed images is ∼ 5 750 bytes. Figs. 7(b) and 7(c) were computed with 300 000 iterations, the triangulations of Figs. 7(b), 7(c), and 7(f) contain 1 000 vertices. For comparison with the results provided in [11], the GSA was also computed using the RGB color model (see Table 6). GSA produced the smallest Δ_G of the tested methods. Its approximation (Fig. 7(b)) misses some high-frequency details, but its overall impression is smooth, whereas

(a) Lena image. (b) GSA. (c) SA [11].

(d) JPEG. (e) JPEG2000. (f) GT [5].

Fig. 7. Original image (∼770 kB) (a), and approximations (∼5 750 bytes) using GSA (b), SA [11] (c), JPEG (d), JPEG2000 (e), and GT [5] (f), (courtesy USC SIPI)



(a) $\Delta_G = 3.0$. (b) $\Delta_G = 2.0$. (c) $\Delta_G = 1.5$. (d) $\Delta_G = 1.0$.

Fig. 8. Approximations of the image shown in Fig. 3(a) with different errors Δ_G

JPEG and JPEG2000 provide some of the detail, but lead to a rough and discontinuous overall impression.

Figs. 3 and 8 show examples of the results of our compression algorithm summarized in Table 1. The original image shown in Fig. 3(a) has 384×512 pixels

Table 1. Comparison of approximations of Fig. 3(a)

Δ_G	Iterations	Time [sec]	Vertices	Size [bytes]	Bits per pixel	Compr. rate	Fig.
3.0	18 715	18.51	73	450	0.0018	1:1 130	8(a)
2.0	24 109	19.46	176	1 039	0.042	1: 568	8(b)
1.5	27 408	22.64	385	2 242	0.091	1: 263	8(c)
1.25	36 173	38.67	711	4 115	0.17	1: 143	3(c)
1.0	47 370	97.92	2 256	11 689	0.47	1: 50	8(d)

and is stored with 589 840 bytes. Fig. 3(b) shows the underlying triangulation of Fig. 3(c) for $\Delta_G = 1.25$. The number of iterations and the number of vertices increase as Δ_G decreases. Thus, best compression rates and lowest computation times are achieved for low image quality and vice versa, see Figs. 8(a) and 8(d).

7 Conclusions and Future Work

The method for the construction of a piecewise linear representation presented in this paper can be used for the construction of high-quality approximations of images. These approximations have high compression rates comparable or even superior to JPEG compression results. Considering the approach discussed in [10,11], our work extends their approach in the following ways:

1. By using guides with acceleration trees we achieve higher compression rates and approximation quality.
2. By adaptively changing the number of vertices the approximation quality can be controlled.
3. Different termination conditions allow for different optimization objectives.
4. Memory requirements for our image approximation are compared to those of JPEG and JPG2000 image compression.

There are some additional research issues we want to work on in the future. The guides can be further improved, for example, by using sharp features detected in the image as attractors for vertices. We plan to investigate to what extend the sliver triangles can cause aliasing effects when scaling the triangulation. Using spline interpolation instead of linear interpolation could improve the approximation quality. We also plan to extend the method to the compression of videos, enhancing compression rates using the coherence of successive frames.

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, University of Davis. We thank the members of the Visualization and Computer Graphics Research Group at

IDAV. In particular, we express our special thanks to Oliver Kreylos, for his support at the beginning of this project, and Klaus Denker for his help on the implementation of the software.

References

1. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Mesh optimization. In: SIGGRAPH 1993, pp. 19–26 (1993)
2. Gao, J., Zhou, M., Wang, H.: Mesh simplification with average planes for 3-d image. Systems, Man, and Cybernetics 2, 1412–1417 (2000)
3. Garland, M., Heckbert, P.: Fast polygonal approximation of terrains and height fields. Technical report, CS Department, Carnegie Mellon University (1995)
4. Schaetzl, R., Hagen, H., Barnes, J., Hamann, B., Joy, K.: Data-dependent triangulation in the plane with adaptive knot placement. In: Brunnett, G., Bieri, H., Farin, G. (eds.) Geometric Modelling, Comp. Suppl., vol. 14, pp. 199–218. Springer, Heidelberg (2001)
5. Hoppe, H.: Progressive meshes. In: SIGGRAPH 1996, pp. 99–108 (1996)
6. Lindstrom, P., Turk, G.: Image-driven simplification. ACM Trans. Graph. 19, 204–241 (2000)
7. Pedrini, H.: An improved refinement and decimation method for adaptive terrain surface approximation. In: Proceedings of WSCG, Czech Republic, pp. 103–109 (2001)
8. Dyn, N., Levin, D., Rippa, S.: Data dependent triangulations for piecewise linear interpolations. IMA J. of Numerical Analysis 10, 137–154 (1990)
9. Schumaker, L.L.: Computing optimal triangulations using simulated annealing. Computer Aided Geometric Design 10, 329–345 (1993)
10. Kreylos, O., Hamann, B.: On simulated annealing and the construction of linear spline approximations for scattered data. IEEE TVCG 7, 17–31 (2001)
11. Petrovic, V., Kuester, F.: Optimized construction of linear approximations to image data. In: Proc. 11th Pacific Conf. on Comp. Graphics and Appl., pp. 487–491 (2003)
12. Prasad, L., Skourikhine, A.: Vectorized image segmentation via trixel agglomeration. Pattern Recogn. 39, 501–514 (2006)
13. Lecot, G., Levy, B.: Ardeco: Automatic region detection and conversion. In: Eurographics Symposium on Rendering conf. proc. (2006)
14. Miano, J.: Compressed Image File Formats. JPEG, PNG, GIF, XBM, BMP. In: SIGGRAPH series, Addison-Wesley Longman, Amsterdam (1999)
15. Taubman, D.S., Marcellin, M.W.: JPEG2000: Image Compression Fundamentals. In: Standards and Practice, Springer, Heidelberg (2002)
16. Wyszecki, G., Stiles, W.: Color Science: Concepts and Methods, Quantitative Data and Formulae. Wiley-Interscience (1982)
17. Kirkpatrick, S., Vecchi, M.P., Jr. Gelatt, C.D.: Optimization by simulated annealing. Science Magazine, 671–680 (1983)
18. Rossignac, J.: Edgebreaker: Connectivity compression for triangle meshes. IEEE TVCG 5, 47–61 (1999)