Adaptive Thinning for Terrain Modelling and Image Compression

Laurent Demaret[1], Nira Dyn[2], Michael S. Floater[3], and Armin Iske[1]

[1] Zentrum Mathematik, Technische Universität München, Munich, Germany, {demaret,iske}@ma.tum.de
[2] School of Mathematical Sciences, Tel-Aviv University, Israel, niradyn@post.tau.ac.il
[3] Computer Science Department, Oslo University, Norway, michaelf@ifi.uio.no

Summary. Adaptive thinning algorithms are greedy point removal schemes for bivariate scattered data sets with corresponding function values, where the points are recursively removed according to some data-dependent criterion. Each subset of points, together with its function values, defines a linear spline over its Delaunay triangulation. The basic criterion for the removal of the next point is to minimize the error between the resulting linear spline at the bivariate data points and the original function values. This leads to a hierarchy of linear splines of coarser and coarser resolutions.

This paper surveys the various removal strategies developed in our earlier papers, and the application of adaptive thinning to terrain modelling and to image compression. In our image test examples, we found that our thinning scheme, adapted to diminish the least squares error, combined with a postprocessing least squares optimization and a customized coding scheme, often gives better or comparable results to the wavelet-based scheme SPIHT.

1 Introduction

This paper concerns the construction of multiresolution approximations to bivariate functions from irregular point samples. These approximations are linear splines over decremental Delaunay triangulations, generated by adaptive thinning algorithms.

A thinning algorithm is a scheme which recursively removes points from a set of scattered data, according to some specific criterion. By recursively removing points, one at a time, a thinning algorithm yields an ordering of the points, which in turn yields a data hierarchy of the input point set.

In *adaptive* thinning algorithms, the criterion for the removal of points depends on both the locations of the given points and the sampled function values at the points. This is in contrast to *non-adaptive* thinning, where the criterion for removal depends only on the geometry of the given planar point set [10].

Linear splines over Delaunay triangulations are used in order to compute multiresolution approximations to the sampled function. Each subset of points defines a unique Delaunay triangulation (up to co-circularity) and a corresponding linear spline function. The resulting sequence of linear splines constitutes the multiresolution approximations of the sampled function.

For every point in the current subset, we maintain an *anticipated error* which provides a local estimate for the true error incurred by its removal. The error is the deviation of the spline function at the 2D data points from the given function values, measured in some specific norm. The point with minimal anticipated error is considered to be the least significant in the current situation, and is removed. In order to obtain good multiresolution approximations to the sampled function, the choice of removal criterion requires care. We customize our removal strategies according to the application.

The idea of thinning scattered data is closely related to *mesh simplification* methods. Indeed, thinning combined with linear splines over Delaunay triangulations is only one of several mesh simplification methods. One of the earliest methods for mesh simplification is the incremental algorithm of Fowler and Little [11]. An early survey paper is that of Lee [16]. De Floriani, Puppo, and co-workers have proposed several algorithms for mesh simplification and developed good data structures for hierarchical triangulations, see [4] and the references therein. Heckbert and Garland [13] give an extensive survey of simplification methods both for terrain models (triangulated scattered data in the plane) and free form models (manifold surfaces represented by 3D triangle meshes). Specific mesh simplification algorithms include techniques like edge-collapse, half-edge collapse, and vertex collapse. For a more recent survey paper on these methods, see the tutorial [12].

Adaptive thinning algorithms are useful for both model simplification and data compression, and have been applied to hierarchical terrain modelling and to image compression. This paper starts with a generic introduction to adaptive thinning algorithms, before going on to various application-specific measures of anticipated errors, and ends with numerical examples of the algorithms applied to terrain modelling and image compression.

In the image compression application, adaptive thinning constructs a hierarchy of most significant pixel positions in a digital image. We use a thinning scheme, whose anticipated error is measured in the ℓ_2 norm, combined with a post-processing step which optimizes the luminances at the most significant pixels. The positions of the pixels in the set of most significant pixels, along with their optimized luminance values, are then converted into a bitstream, using a customized coding scheme for scattered data, developed in our previous paper [5]. At the decoder, the transmitted data is used for the image reconstruction, by evaluating the linear spline over the Delaunay triangulation of the transmitted pixels, interpolating the optimized luminance values at the vertices.

The result is a novel image compression scheme, $\mathbf{AT_2^*}$, which, in our numerical examples, often gives better or comparable compression rates to the well-established wavelet-based compression method SPIHT (Set Partitioning Into Hierarchical Trees).

2 Generic Formulation of Thinning

This section provides a generic introduction to the basic features and concepts of adaptive thinning algorithms. Let $X = \{x_1, \dots, x_N\} \subset \mathbb{R}^2$ denote a finite scattered point set in $\mathbb{R}^2$, and let $f_X = (f(x_1), \dots, f(x_N))^T \in \mathbb{R}^N$ denote a corresponding data vector containing point samples taken from an unknown function $f : \mathbb{R}^2 \to \mathbb{R}$ at the points of X.

Thinning is a recursive point removal scheme for bivariate scattered data, whose generic formulation is given by the following algorithm, where n is the number of removals.

Algorithm 1 (Thinning).

```
(1) Let $X_N = X$;
(2) FOR $k = 1, \dots, n$
    (2a) Locate a removable point $x \in X_{N-k+1}$;
    (2b) Let $X_{N-k} = X_{N-k+1} \setminus x$;
```

Note that thinning constructs, for given data (X, f_X), a nested sequence

$$X_{N-n} \subset \cdots \subset X_{N-1} \subset X_N = X \qquad (1)$$

of subsets of X, where the size $|X_k|$ of any subset X_k in (1) is k, $N-n \le k \le N$. Two consecutive subsets in (1) differ only by one point.

In order to select a specific thinning strategy, it remains to give a definition for a removable point in step **(2a)** above. Before we propose several different preferred removal strategies, let us first discuss our motivation for the construction of the data hierarchy in (1). Our intention is to use the data hierarchy (1) in order to create a multiresolution approximation of the sampled function f from the given data f_X.

The multiresolution approximation of f combines the data hierarchy (1) with *linear splines*. Recall that a linear spline is a continuous function, which is piecewise linear over a partitioning of its domain $\Omega \subset \mathbb{R}^2$. In the setting of this paper, we let the domain Ω coincide with the convex hull $[X]$ of the input point set X. This makes sense, if the convex hull $[Y]$ of any subset $Y \subset X$, constructed by thinning, coincides with the convex hull of X. We ensure this by not removing *extremal* points from X. A convenient choice for the partitioning of Ω is the *Delaunay triangulation* $\mathcal{D}(Y)$ of Y. Although we assume that the reader is familiar with Delaunay triangulation methods, let us recall some of their basic properties, which are relevant to the construction of adaptive thinning algorithms. For a comprehensive discussion of triangulation methods, in particular Delaunay triangulations, we refer to the textbook [17] and the paper [19].

Firstly, we remark that a Delaunay triangulation $\mathcal{D}(Y)$ of a finite planar point set Y is one, such that for any triangle in $\mathcal{D}(Y)$ its corresponding circumcircle does not contain any point from Y in its interior. This property is termed the *Delaunay property.* Moreover, there is a unique triangulation of Y with the Delaunay property, provided that no four points in Y are co-circular [17]. We assume this condition on the given set X in order to avoid lengthy but immaterial discussions concerning the non-uniqueness of $\mathcal{D}(Y)$, for $Y \subset X$.

Secondly, note that the removal of one point y from Y requires an update of $\mathcal{D}(Y)$ in order to obtain the Delaunay triangulation $\mathcal{D}(Y \setminus y)$. Due to the Delaunay property, this update of $\mathcal{D}(Y)$ is *local.* Indeed, the required retriangulation, incurred by the removal of the vertex y in $\mathcal{D}(Y)$, can be performed by the retriangulation of its cell $C(y)$. Recall that the cell $C(y)$ of y is the union of all triangles in $\mathcal{D}(Y)$ which contain y as a vertex. Figure 1 shows, for a vertex y in a Delaunay triangulation, the retriangulation of its cell $C(y)$.

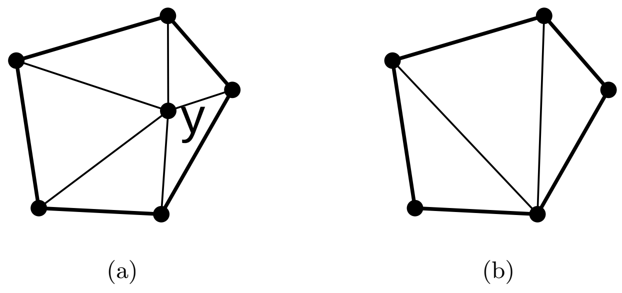


Fig. 1. Removal of the vertex y, and retriangulation of its cell. The five triangles of the cell in (a) are replaced by the three triangles in (b).

For any $Y \subset X$ let

$$\mathcal{S}_Y = \left\{ s : s \in C([Y]) \text{ and } s\big|_T \text{ linear for all } T \in \mathcal{D}(Y) \right\},$$

be the spline space containing all continuous functions over $[Y]$ whose restriction to any triangle in $\mathcal{D}(Y)$ is linear. Any element in $\mathcal{S}_Y$ is referred to as a *linear spline* over $\mathcal{D}(Y)$. For given function values f_Y, there is a unique linear spline $L(Y; f) \in \mathcal{S}_Y$ which interpolates f at the points of Y,

$$L(Y; f)(y) = f(y), \qquad \text{for all } y \in Y.$$

3 Adaptive Anticipated Error Measures

The aim of adaptive thinning is to construct a data hierarchy of the form (1) from the given data (X, f_X), such that for any subset $Y = X_k \subset X$ in (1) the interpolatory linear spline $L(Y; f) \in \mathcal{S}_Y$, is *close* at X to the given function values $f_X \in \mathbb{R}^N$.

In order to establish this, we require that, for some norm $\|\cdot\|$ on $\mathbb{R}^N$, the *approximation error*

$$\eta(Y; X) = \|L(Y; f)\big|_X - f_X\| \tag{2}$$

is small. Note that $\eta(Y; X)$ in (2) depends also on the input values f_X, but for notational simplicity we omit this.

For the discrete ℓ_∞-norm, the approximation error $\eta(Y; X)$ becomes

$$\eta_\infty(Y; X) = \max_{x \in X} |L(Y; f)(x) - f(x)|, \tag{3}$$

whereas for the discrete ℓ_2-norm, we obtain

$$\eta_2(Y; X) = \sqrt{\sum_{x \in X} |L(Y; f)(x) - f(x)|^2}. \tag{4}$$

Ideally, for any k, $N - n < k < N$, we would like to locate a subset $Y \subset X$ which minimizes the error in (2) among all subsets of X of equal size $|Y| = k$. We remark, however, that the problem of finding an algorithm which outputs for any possible input (X, f_X, k), $N - n < k < |X|$, such an optimal subset Y^* of size $|Y^*| = k$ satisfying

$$\eta(Y^*; X) = \min_{\substack{Y \subset X \\ |Y| = k}} \eta(Y; X) \tag{5}$$

is closely related to the NP-hard *k-center problem.* For a comprehensive discussion on the k-center problem we refer to the textbook [14, Section 9.4.1] and the survey [20]. To overcome this difficulty, thinning algorithms are based on a *greedy* removal strategy. The application of greedy algorithms to the k-center problem is developed in [15].

Greedy algorithms are in general known as efficient and effective methods of dynamic programming for solving optimization problems *approximately.* Greedy algorithms typically go through a sequence of steps, where for each step a choice is made that looks best at the moment. For a general introduction to greedy algorithms we recommend the textbook [2, Chapter 16].

For the thinning Algorithm 1, the most natural removal criterion in step **(2a)** for approximately solving (5) by a greedy algorithm is

Definition 1. (Removal Criterion AT)
For $Y \subset X$, a point $y^* \in Y$ is said to be `removable` *from Y, iff it satisfies*

$$\eta(Y \setminus y^*; X) = \min_{y \in Y} \eta(Y \setminus y; X).$$

We refer to the adaptive thinning algorithm, resulting from this removal criterion in Algorithm 1, as **AT**.

Let us make a few remarks on the idea of this particular definition of a removable point. When using the above removal criterion **AT**, we assign to each current point $y \in Y$ an *anticipated error*

$$e(y) = \eta(Y \setminus y; X),$$

which is incurred by the removal of y. Moreover, we interpret the value $e(y)$ as the significance of the point y in the current subset Y. In this sense, a point y^* whose removal gives the least anticipated error $e(y^*)$ is considered as *least significant* in the current situation, and so it is removed from Y.

For the implementation of the thinning algorithm, we use a priority queue of the points, according to their significances. This priority queue has a least significant point at its head, and is updated after each removal of its head. For more details concerning the efficient maintenance of the priority queue of the scattered points, we refer to our paper [8].

In the remainder of this section, we propose different removal criteria, which are adapted to terrain modelling and to image compression. Let us briefly explain the differences between these two applications. In terrain modelling, it is of primary importance to keep the *maximal* deviation $\eta_\infty(Y;X)$ between the linear spline interpolant $L(f;Y)$ and the given point samples f_X as small as possible. This is in contrast to applications in image compression, where the quality measure relies on the *mean square error*

$$\bar{\eta}_2^2(Y;X) = \eta_2^2(Y;X)/N. \tag{6}$$

We develop two classes of customized adaptive thinning criteria. Those for terrain modelling work with the error measure η_∞ in (3), whereas the anticipated error measures for image compression rely on the discrete ℓ_2-error η_2^2 in (4). Accordingly, we denote by $\mathbf{AT}_\infty$ the adaptive thinning algorithm **AT** which works with the ℓ_∞-norm, whereas $\mathbf{AT_2}$ is the algorithm **AT** for the choice of the ℓ_2-norm. The removal criterion for $\mathbf{AT}_\infty$ cannot be computed locally, but alternative local removal criteria are suggested in the next subsection.

3.1 Anticipated Errors for Terrain Modelling

In this subsection, we propose three locally computable, alternative removal criteria, **AT1**, **AT2** and **AT3**, which reduce the computational costs of the resulting thinning algorithm, in comparison with $\mathbf{AT}_\infty$. The removal criteria **AT1** and **AT2** require the retriangulation of $Y \setminus y$ for the computation of the anticipated error of any y in Y. Due to the Delaunay property, only the retriangulation of the cell $C(y)$ is required. The removal criterion **AT3** does not require the retriangulation of the cell $C(y)$ for the computation of the anticipated error of y in Y.

The first alternative, **AT1**, measures the anticipated error of a point y only in its cell $C(y)$,

$$e_1(y) = \eta_\infty(Y \setminus y; X \cap C(y)).$$

Definition 2. (Removal Criterion AT1)
For $Y \subset X$, a point $y^* \in Y$ is said to be `removable` *from Y, iff it satisfies*

$$e_1(y^*) = \min_{y \in Y} e_1(y).$$

We remark that the adaptive thinning algorithm $\mathbf{AT}_\infty$ is not equivalent to **AT1**. This is confirmed by the following counter-example.

Example 1. **($\mathbf{AT}_\infty \neq$ AT1)**
Consider the eight data points, $X = \{x_1, \ldots, x_8\}$ and the values f_X given as follows.

i	1	2	3	4	5	6	7	8
x_i	$(1,0)$	$(2,0)$	$(3,0)$	$(4,0)$	$(5,0)$	$(6,0)$	$(7,0)$	$(1,1)$
$f(x_i)$	5	-1	0	-3	0	-1.1	2.5	0

In this case, the extremal points of X are x_1, x_7 and x_8, so that only the five points x_i, $i = 2, \ldots, 6$, can be removed. It is easy to see that both $\mathbf{AT}_\infty$ and **AT1** remove the point $x_3 = (3,0)$ first, and then they remove $x_5 = (5,0)$. In the third step, however, the algorithm **AT1** removes $x_6 = (6,0)$, whereas the algorithm $\mathbf{AT}_\infty$ removes $x_4 = (4,0)$.

Now let us turn to the adaptive thinning algorithm **AT2**, being a simplification of the previous **AT1**. In order to further reduce the required computational costs, the removal criterion **AT2** depends only on the sample values f_Y of the points in the current subset $Y \subset X$. This is in contrast to both $\mathbf{AT}_\infty$ and **AT1**, which depend on points in X which were removed in previous steps.
The anticipated error of **AT2** is, for any $y \in Y$, given by

$$e_2(y) = \eta_\infty(Y \setminus y; Y).$$

Note that this expression can be rewritten as

$$e_2(y) = |L(Y \setminus y; f)(y) - f(y)|.$$

Definition 3. (Removal Criterion AT2)
For $Y \subset X$, a point $y^* \in Y$ is said to be `removable` *from Y, iff it satisfies*

$$e_2(y^*) = \min_{y \in Y} e_2(y).$$

We have also explored an adaptive thinning algorithm, **AT3**, which is faster than **AT2**. The algorithm **AT3** does not only ignore the points already removed but also computes an *anticipated error* for each point without needing to temporarily retriangulate its cell.

The basic idea behind **AT3** is to define this anticipated error $e_3(y)$ as the maximum of the *directional anticipated errors* at y in a certain sample of directions. For each neighbouring vertex z of y in $\mathcal{D}(Y)$ we consider the unique point p lying at the intersection of the boundary of $C(y)$ and the straight line passing through z and y (other than z itself). Such a point exists, since $C(y)$ is a *star-shaped polygon.*

The point p is either a vertex of the cell's boundary $\partial C(y)$ or a point on one of its sides. In either case, p lies on at least one edge of $\partial C(y)$. Let us denote such an edge by $[z_2, z_3]$; see Figure 2. Then the triangle $T_z = [z, z_2, z_3]$, with vertices in $Y \setminus y$, contains y. We call T_z a *directional triangle* of y. We then let

$$e_3^z(y) = |L(T_z; f)(y) - f(y)|$$

be the (unique) directional anticipated error of y in the direction $z - y$, where $L(T_z; f)$ is the linear function which interpolates f at the vertices of T_z. Now, we let

$$e_3(y) = \max_{z \in V_y} e_3^z(y)$$

for the anticipated error of the adaptive thinning algorithm **AT3**, where V_y is the set of all neighbouring vertices of y in $\mathcal{D}(Y)$.

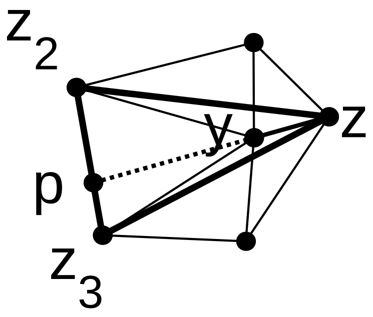


Fig. 2. Directional triangle of y.

Definition 4. (Removal Criterion AT3)
For $Y \subset X$, a point $y^* \in Y$ is said to be `removable` *from Y, iff it satisfies*

$$e_3(y^*) = \min_{y \in Y} e_3(y).$$

A detailed analysis of the complexity of the adaptive thinning algorithms $\mathbf{AT}_\infty$, **AT1**, **AT2**, and **AT3** can be found in [8]. It is shown in [8] that the asymptotic computational costs of any of these three algorithms is

$\mathcal{O}(N \log(N))$, but with different constants. For further illustration, we refer to the numerical examples in Section 4, where these algorithms are applied to one selected test case from terrain modelling.

3.2 Anticipated Errors for Image Compression

In this subsection, we propose one customized removal criterion for image compression. In this application, the quality of image compression schemes is usually measured in *Peak Signal to Noise Ratio* (PSNR) (7), see the discussion in Section 5. It is sufficient for the moment to say that PSNR is an equivalent measure to the reciprocal of the *mean square error* $\overline{\eta}_2^2(Y;X)$ in (6).

The aim of adaptive thinning, when applied to image compression, is to keep the mean square error as small as possible. This is accomplished by using adaptive thinning algorithms, which generate subsets $Y \subset X$ in (1), whose square error $\eta_2^2(Y;X)$ is, among subsets of equal size, small. Therefore, we work with the discrete ℓ_2-norm η_2 in (4).

Let us now discuss the adaptive thinning algorithm $\mathbf{AT_2}$, whose anticipated error is given by

$$e(y) = \eta_2^2(Y \setminus y; X), \qquad \text{for } y \in Y.$$

By the additivity of η_2^2 and by the observations $X = (X \setminus C(y)) \cup (X \cap C(y))$, and $\eta_2^2(Y \setminus y; X \setminus C(y)) = \eta_2^2(Y; X \setminus C(y))$, for any $y \in Y$, we get

$$\begin{aligned}\eta_2^2(Y \setminus y; X) &= \eta_2^2(Y \setminus y; X \setminus C(y)) + \eta_2^2(Y \setminus y; X \cap C(y)) \\ &= \eta_2^2(Y; X \setminus C(y)) + \eta_2^2(Y \setminus y; X \cap C(y)) \\ &= \eta_2^2(Y; X) + \eta_2^2(Y \setminus y; X \cap C(y)) - \eta_2^2(Y; X \cap C(y)).\end{aligned}$$

Hence, for any $Y \subset X$, the minimization of $\eta_2^2(Y \setminus y; X)$ is equivalent to minimizing the difference

$$e_\delta(y) = \eta_2^2(Y \setminus y; X \cap C(y)) - \eta_2^2(Y; X \cap C(y)), \qquad \text{for } y \in Y,$$

where $C(y)$ is the cell of y in $\mathcal{D}(Y)$.

Definition 5. (Removal Criterion $\mathbf{AT_2}$)
For $Y \subset X$, a point $y^* \in Y$ is said to be `removable` *from Y, iff it satisfies*

$$e_\delta(y^*) = \min_{y \in Y} e_\delta(y).$$

We remark that we can establish the complexity $\mathcal{O}(N \log(N))$ for the adaptive thinning algorithm $\mathbf{AT_2}$, by following along the lines of the analysis in [8]. This, however, is beyond the scope of this survey.

4 Adaptive Thinning in Terrain Modelling

We have implemented the thinning algorithms **AT1**, **AT2**, and **AT3**, together with one non-adaptive thinning algorithm, called **NAT** [8]. The algorithm **NAT**, proposed in [10], ignores the given samples f_X, and favours evenly distributed subsets of points $Y \subset X$. We refrained from implementing the algorithm $\mathbf{AT}_\infty$, since it requires significantly more computations [8], and due to our experience in the univariate setting [7], we do not expect $\mathbf{AT}_\infty$ to be significantly better than **AT1**.

In this section, we compare the performance of the four algorithms **AT1**, **AT2**, **AT3**, and **NAT** in terms of both approximation quality and computational cost on one specific example from terrain modelling. The corresponding data set, *Hurrungane*, contains 23092 data points. Each data point is of the form $(x, f(x))$, where $f(x)$ denotes the terrain's height value sampled at the location $x \in \mathbb{R}^2$. This data set is displayed in Figure 3 (a) (2D view) and in Figure 3 (b) (3D view).

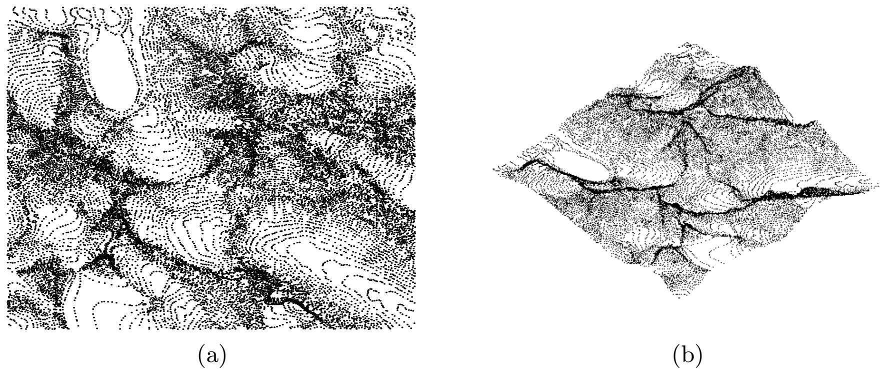

(a) (b)

Fig. 3. Hurrungane: (a) 2D view and (b) 3D view.

For all four thinning algorithms, we have recorded both the required seconds of CPU time (without considering the computational costs required for building the initial data structures, such as the Delaunay triangulation) and the sequence of approximation errors $\eta_\infty(Y; X)$ after the removal of $n = 1000, 2000, \ldots, 22000$ points from X.

Not surprisingly, we found that **NAT** is the fastest method but also the worst one in terms of its approximation error. For example, for $n = 22000$ the algorithm **AT1** takes 247.53 seconds of CPU time, whereas **NAT** takes only 11.37 seconds. On the other hand, we obtain in this particular example $\eta_\infty(Y; X) = 278.61$ for **NAT**, but only $\eta_\infty(Y; X) = 30.09$ when using **AT1**. The two corresponding triangulations $\mathcal{D}(Y)$ output by **NAT** and **AT1** are displayed in Figure 4 (a) and (b) (2D view), and in Figure 5 (a) and (b) (3D view).

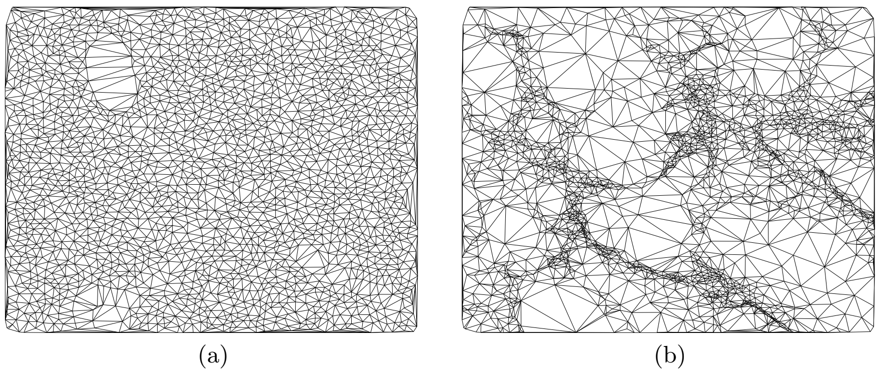


Fig. 4. Thinned Hurrungane with 1092 points, 2D view. (a) **NAT** and (b) **AT1**.

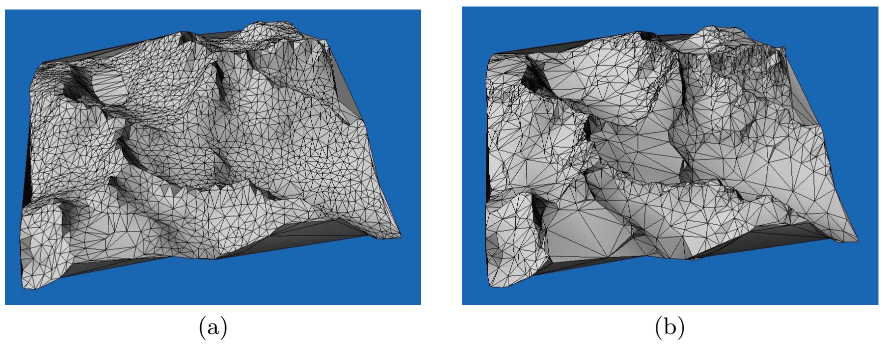


Fig. 5. Thinned Hurrungane with 1092 points, 3D view. (a) **NAT** and (b) **AT1**.

In Figure 6 (a) and in Figure 7 (a) the approximation error $\eta_\infty(Y;X)$ as a function of the number of removed points is plotted for the four different thinning algorithms. In Figure 6 (b) and in Figure 7 (b) the corresponding seconds of CPU time are displayed.

The graphs show that, with respect to approximation error, the three adaptive thinning algorithms **AT1**, **AT2**, and **AT3** are much better than **NAT**. Among the three adaptive thinning algorithms, **AT1** is the best, followed by **AT3**, and **AT2** is the worst. Note that by definition **AT3** can only be inferior to **AT2** after one removal. In the numerical example, **AT3** has continued to

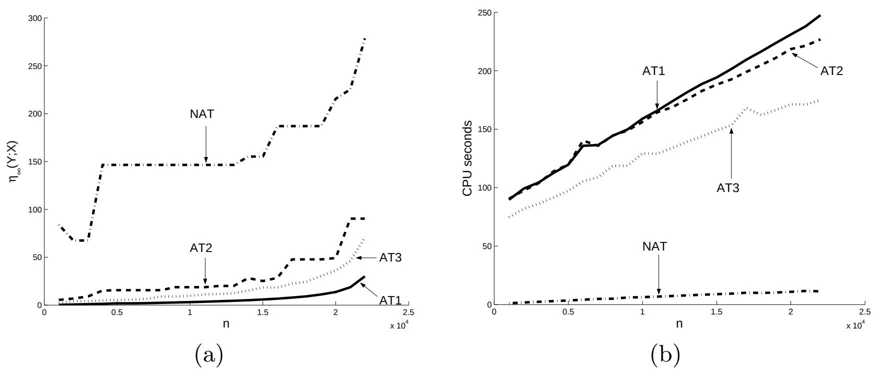


Fig. 6. Hurrungane: comparison between **NAT** (dash-dot line), **AT1** (solid), **AT2** (dashed), and **AT3** (dotted), (a) approximation error and (b) seconds of CPU time.

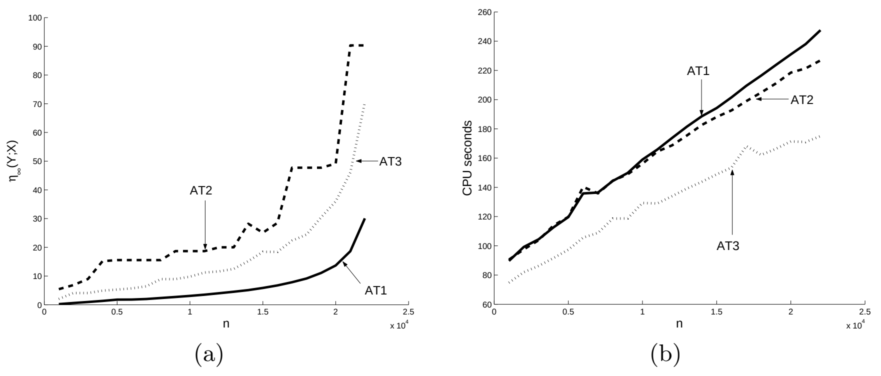


Fig. 7. Hurrungane: comparison between **AT1** (solid line), **AT2** (dashed), and **AT3** (dotted), (a) approximation error and (b) seconds of CPU time.

be inferior for about 50 removal steps, after which its approximation error is smaller than that of **AT2**.

As to the computational costs for the adaptive thinning algorithms, **AT3** is the fastest, and **AT1** the slowest, see Figure 6 (b) and Figure 7 (b). Our conclusion is that **AT1** is our recommended thinning algorithm. But if computational time is a critical issue, **AT3** is a good alternative.

5 Adaptive Thinning in Image Compression

This section reviews our recent research on the application of adaptive thinning to image compression. The information reduction and efficient coding of digital images are essential for fast transmission across an information channel, such as the internet. For a comprehensive introduction to image compression and coding, we recommend the textbook [22].

Many of the well-established techniques in image compression, including JPEG2000 [22], are based on wavelets and related techniques, see [3] for a recent survey on wavelet-based image coding. When working with wavelets, digital images are represented by using *rectangular grids* of wavelet coefficients. This is in contrast to adaptive thinning, which works with *scattered data*.

Adaptive thinning constructs a hierarchy of sets of most significant pixels, where for each set the image is approximated by the linear spline over the Delaunay triangulation of the pixels in the set. The idea to approximate an image by first identifying significant pixels is not new (see e.g. [9]). In this section we go further and so obtain a competitive compression scheme, based on adaptive thinning.

Any compression scheme is mainly concerned with the following sequence of tasks.

(1) data reduction;
(2) encoding of the reduced data at the sender;
(3) transmission of the encoded data from the sender to the receiver;
(4) decoding of the transmitted data at the receiver;
(5) data reconstruction.

Adaptive thinning is mainly used in the above step **(1)**, the data reduction. In the following discussion, we first explain how adaptive thinning works in image data reduction, before we show how the reconstruction step **(5)** is accomplished. For a discussion of scattered data coding, required in steps **(2)** and **(4)**, we refer to our paper [5].

Before we explain any details on our compression scheme, we wish to make a few preliminary remarks concerning our image model. Note that many images contain discontinuities. It might therefore seem like a good idea to approximate such images by *discontinuous* functions over triangulations, instead of keeping to the *continuous* piecewise linear functions of Algorithm 1.

One possibility would be to use piecewise *constant* functions over triangulations. This, however, reduces the approximation quality of the model, because piecewise constants have only $\mathcal{O}(h)$ approximation order, rather than $\mathcal{O}(h^2)$ for piecewise linear functions. Another possibility would be to use piecewise linear functions that are not necessarily globally continuous. This, however, requires assigning several height values to each vertex, and so in this approach there is significantly more data attached to each triangulation, which leads to higher coding costs.

Unless one works with a very sophisticated coding scheme, which makes fundamental use of the statistical correlation of the data around the image's discontinuities, either of these two discontinuous models leads, due to our experience, to inferior compression ratios. Therefore, we prefer to work with a continuous image model, so as the one using continuous piecewise linear functions in Algorithm 1.

5.1 Adaptive Thinning and Image Reduction and Reconstruction

A digital image is a rectangular grid of *pixels*. Each pixel bears a color value or greyscale *luminance*. For the sake of simplicity, we restrict the following discussion to greyscale images. The image can be viewed as a matrix $F = (f(i,j))_{i,j}$, whose entries $f(i,j)$ are the luminance values at the pixels. The pixel positions $(i,j) \in X$ are pairs of non-negative integers i and j, whose range is often of the form $[0..2^p - 1] \times [0..2^q - 1]$, for some positive integers p, q, where we let $[0..n] = [0,n] \cap \mathbb{Z}$ for any non-negative integer $n \in \mathbb{Z}$. In this case, the size of the pixel set X is $2^p \times 2^q$. Likewise, the entries $f(i,j)$ in F are non-negative integers whose range is typically $[0..2^r - 1]$, for some positive integer r. In the examples of the test images below, we work with 256 greyscale luminances in [0..255], so that in this case $r = 8$.

A well-known quality measure for the evaluation of image compression schemes is the *Peak Signal to Noise Ratio* (PSNR),

$$\mathrm{PSNR} = 10 * \log_{10}\left(\frac{2^r \times 2^r}{\bar{\eta}_2^2(Y;X)}\right), \tag{7}$$

which is an equivalent measure to the reciprocal of the mean square error $\bar{\eta}_2^2(Y;X)$ in (6). The PSNR is expressed in `dB` (decibels). Good image compressions typically have PSNR values of 30 `dB` or more [22] for the reconstructed image. The popularity of PSNR as a measure of *image distortion* derives partly from the ease with which it may be calculated, and partly from the tractability of linear optimization problems involving squared error metrics. More appropriate measures of *visual distortion* are discussed in [22].

Adaptive thinning, when applied to digital images, recursively deletes pixels using the thinning Algorithm 1, in combination with the adaptive removal criterion $\mathbf{AT_2}$ of Section 3. In other words, the pixel positions form the initial point set X on which the adaptive thinning algorithm is applied. At any step of the algorithm, a *removable pixel* (point) is removed from the image. The output of adaptive thinning is a set $Y \subset X$ of pixels combined with their corresponding luminances F_Y.

However, due to the regular distribution of pixel positions, the Delaunay triangulations of X, and of its subsets $Y \subset X$, might be non-unique. To avoid this ambiguity, we apply a small perturbation to the pixels X and apply the thinning algorithm to the perturbed pixels.

As a postprocess to the thinning, we further minimize the mean square error by *least squares approximation* [1]. More precisely, we compute from the output set Y and the values F the unique *best ℓ_2-approximation* $L^*(Y;F) \in \mathcal{S}_Y$ satisfying

$$\sum_{(i,j)\in X} |L^*(Y;F)(i,j) - f(i,j)|^2 = \min_{s\in\mathcal{S}_Y} \sum_{(i,j)\in X} |s(i,j) - f(i,j)|^2. \tag{8}$$

Such a unique solution exists since $Y \subset X$. The compressed information to be transferred consists of the output set Y and the corresponding optimized luminances $\{f^*(i,j) = L^*(Y;F)(i,j) : (i,j) \in Y\}$.

Following along the lines of our papers [5, 6], we apply a uniform quantization to these *optimized* luminances. This yields the quantized symbols $\{Q(f^*(i,j)) : (i,j) \in Y\}$, corresponding to the quantized luminance values $\{\tilde{f}(i,j) : (i,j) \in Y\}$, where $\tilde{f}(i,j) \approx f^*(i,j)$ for $(i,j) \in Y$. The elements of the set $\{(i,j,Q(f^*(i,j))) : (i,j) \in Y\}$ are coded by using the customized scattered data coding scheme of [5].

At the receiver, the reconstruction of the image F (step **(5)**) is then accomplished as follows. The *unique* Delaunay triangulation $\mathcal{D}(Y)$ of the pixel positions Y is computed at the decoder, using the same perturbation rules applied previously at the encoder. This defines, in combination with the decoded luminance values $\tilde{F}_Y = \{\tilde{f}(i,j) : (i,j) \in Y\}$, the unique linear spline $\tilde{L}(Y;\tilde{F}_Y) \in \mathcal{S}_Y$ satisfying $\tilde{L}(Y;\tilde{F}_Y)(i,j) = \tilde{f}(i,j)$ for every $(i,j) \in Y$. Finally, the reconstruction of the image is given by the image matrix $\tilde{F} = (\tilde{L}(Y;\tilde{F}_Y)(i,j))_{(i,j)\in X}$.

We denote the novel image compression scheme, presented in this subsection, by $\mathbf{AT}_\mathbf{2}^*$.

5.2 Comparison between $\mathbf{AT}_\mathbf{2}^*$ and SPIHT

In this subsection we compare the performance of our compression scheme $\mathbf{AT}_\mathbf{2}^*$ with that of the wavelet-based compression scheme *Set Partitioning Into Hierarchical Trees* (SPIHT) [18] on two test images. We work with greyscale values of the luminances $f(i,j)$ in [0..255], i.e., $r = 8$. In the test examples below, we use the range [0..31] for the quantized symbols $Q(f^*(i,j))$, with $(i,j) \in Y$. In each test case, the compression rate, measured in *bits per pixel* (`bpp`), is fixed. The quality of the resulting reconstructions is then evaluated by the comparison of the differences in PSNR, and in visual quality.

We remark that the good compression rate of SPIHT is, at low bit rates, comparable with that of the powerful method *EBCOT* [21], which is the basis algorithm of the standard *JPEG2000* [22].

A Geometric Test Image

We first consider one artificial test image, `Reflex`, of small size 128×128 ($p = q = 7$). This geometric test image is displayed in Figure 8 (a). The purpose of this test case is to demonstrate the good performance of our compression scheme $\mathbf{AT}_\mathbf{2}^*$ on texture-free images with sharp edges.

In this test case, we fix the compression rate to 0.251 `bpp`. The resulting reconstructions corresponding to $\mathbf{AT}_\mathbf{2}^*$ and to SPIHT are displayed in Figure 8 (b),(d). Our compression scheme $\mathbf{AT}_\mathbf{2}^*$ yields the PSNR value 41.73 `dB`, whereas SPIHT provides the inferior PSNR value 30.42 `dB`. Hence, with respect to this quality measure, our compression method $\mathbf{AT}_\mathbf{2}^*$ is much better.

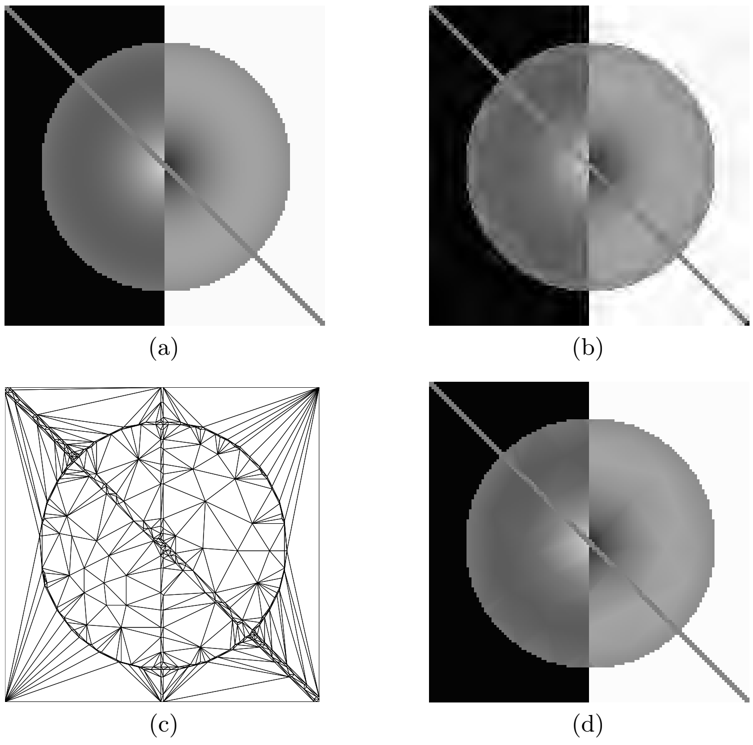


Fig. 8. `Reflex`. (a) Original image of size 128×128. Compression at 0.251 `bpp` and reconstruction by (b) SPIHT with PSNR 30.42 `db`, (d) $\mathbf{AT_2^*}$ with PSNR 41.73 `db`. (c) The Delaunay triangulation of the 384 most significant pixels output by $\mathbf{AT_2}$.

Moreover, the reconstruction by $\mathbf{AT_2^*}$ provides also a superior *visual quality* to that of the reconstructed image by SPIHT, see Figures 8 (b),(d). Indeed, our compression scheme $\mathbf{AT_2^*}$ manages to localize the sharp edges of the test image `Reflex`. Moreover, it avoids undesired oscillations around the edges, unlike SPIHT. This is due to the well-adapted distribution of the 384 most significant pixels, output by the adaptive thinning algorithm $\mathbf{AT_2}$, whose Delaunay triangulation is displayed in Figure 8 (c).

We have recorded the results of this example, along with those of the following test case, in Table 1.

A Popular Test Case of a Real Image

We considered also applying our compression scheme $\mathbf{AT_2^*}$ to one popular test case of a real image, called `Fruits`, which is also used as a standard test case in the textbook [22]. The original image `Fruits`, of size 512×512, is displayed in Figure 9.

It is remarkable that our compression scheme $\mathbf{AT_2^*}$ is, at low bitrates, quite competitive with SPIHT. This is confirmed by the following comparison between SPIHT and $\mathbf{AT_2^*}$ at the bitrate 0.185 `bpp`. The different PSNR values are shown in the second row of Table 1. Note that the PSNR obtained by $\mathbf{AT_2^*}$ is only slightly smaller than that obtained by SPIHT.

Now let us turn to the visual quality of the reconstructions. The reconstruction by SPIHT is shown in Figure 11 (a), whereas Figure 11 (b) shows the reconstruction by $\mathbf{AT_2^*}$.

The set Y of most significant pixel positions obtained by $\mathbf{AT_2}$, along with its Delaunay triangulation $\mathcal{D}(Y)$, are displayed in Figure 10. Note that by the distribution of the most significant pixels, the main features of the image, such as sharp edges and silhouettes, are captured very well. Moreover, our compression scheme $\mathbf{AT_2^*}$ manages to denoise the test image `Fruits` quite successfully, in contrast to SPIHT.

On balance, in terms of the *visual* quality of the two reconstructions of `Fruits`, we feel that our compression scheme $\mathbf{AT_2^*}$ is at least as good as SPIHT.

		Peak Signal to Noise Ratio (PSNR)	
Test Case	`bpp`	SPIHT	$\mathbf{AT_2^*}$
Reflex	0.251	30.42	41.73
Fruits	0.185	32.33	31.85

Table 1. Comparison between the compression schemes SPIHT and $\mathbf{AT_2^*}$.

Acknowledgment

The assistance of Konstantinos Panagiotou, Bertolt Meier, Eugene Rudoy, Georgi Todorov and Rumen Traykov with the implementation of the compression schemes and the preparation of the numerical examples is gratefully appreciated. The authors were partly supported by the European Union within the project MINGLE (Multiresolution in Geometric Modelling), HPRN-CT-1999-00117.

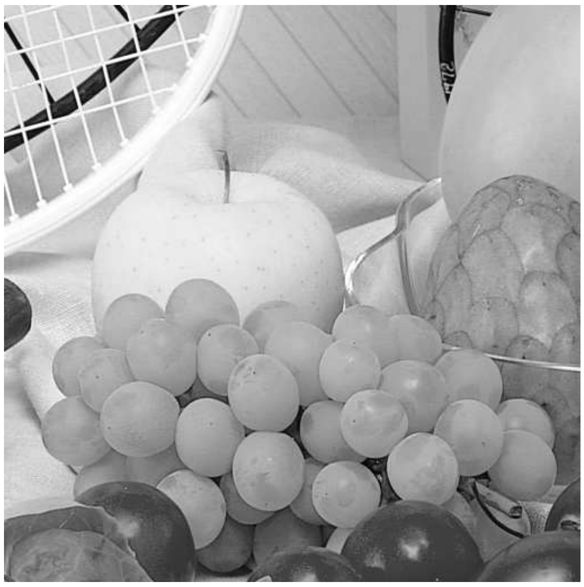


Fig. 9. `Fruits`. Original image of size 512×512.

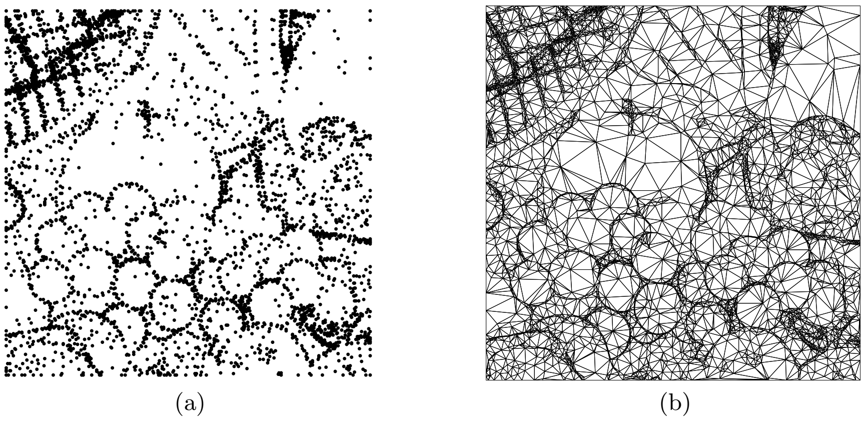
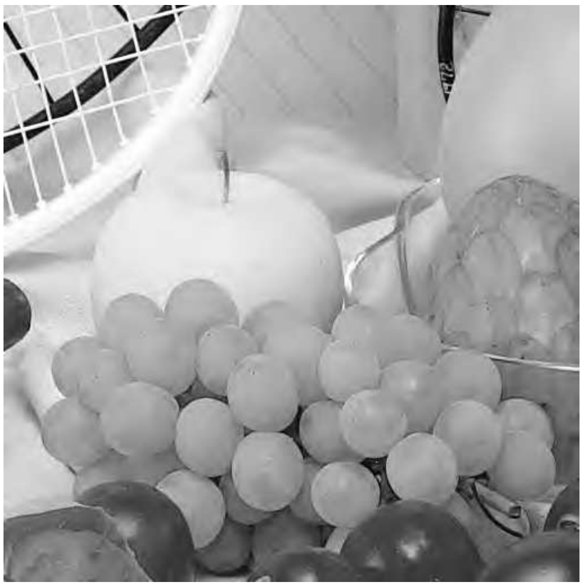
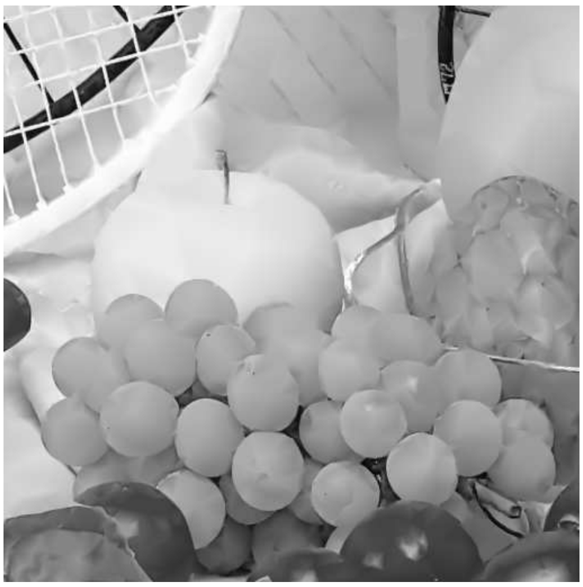


Fig. 10. `Fruits`. (a) 4044 most significant pixels output by $\mathbf{AT_2}$ and (b) their Delaunay triangulation.



(a)



(b)

Fig. 11. `Fruits`. Compression at 0.185 `bpp` and reconstruction by (a) SPIHT with PSNR 32.33 `db` and (b) $\mathbf{AT}_2^*$ with PSNR 31.85 `db`.

References

1. Å. Björck: *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.
2. T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein (2001) *Introduction to Algorithms*, 2nd edition. MIT Press, Cambridge, Massachusetts.
3. G.M. Davis, A. Nosratinia: Wavelet-based image coding: an overview, *Appl. Comp. Control, Signal & Circuits*, B.N. Datta (ed), Birkhauser, 1999, 205–269.
4. L. De Floriani, P. Magillo, E. Puppo (1997) Building and traversing a surface at variable resolution. Proceedings of IEEE Visualization **97** 103–110.
5. L. Demaret and A. Iske (2003) Scattered data coding in digital image compression. *Curve and Surface Fitting: Saint-Malo 2002*, A. Cohen, J.-L. Merrien, and L.L. Schumaker (eds.), Nashboro Press, Brentwood, 107–117.
6. L. Demaret and A. Iske (2003) Advances in digital image compression by adaptive thinning. To appear in the *MCFA Annals*, Volume III.
7. N. Dyn, M.S. Floater, and A. Iske (2001) Univariate adaptive thinning. *Mathematical Methods for Curves and Surfaces: Oslo 2000*, T. Lyche and L.L. Schumaker (eds.), Vanderbilt University Press, Nashville, 123–134.
8. N. Dyn, M.S. Floater, and A. Iske (2002) Adaptive thinning for bivariate scattered data. J. Comput. Appl. Math. **145**(2), 505–517.
9. Y. Eldar, M. Lindenbaum, M. Porat, and Y.Y. Zeevi (1997) The farthest point strategy for progressive image sampling. IEEE Transactions on Image Processing **6**(9), September 1997, 1305–1315.
10. M.S. Floater and A. Iske (1998) Thinning algorithms for scattered data interpolation. BIT **38**, 705–720.
11. R.J. Fowler and J.J. Little (1979) Automatic extraction of irregular network digital terrain models. Computer Graphics **13**, 199–207.
12. C. Gotsman, S. Gumhold, and L. Kobbelt (2002) Simplification and Compression of 3D Meshes. *Tutorials on Multiresolution in Geometric Modelling*, A. Iske, E. Quak, and M.S. Floater (eds.), Springer-Verlag, Heidelberg, 319–361.
13. P.S. Heckbert and M. Garland (1997) Survey of surface simplification algorithms. Technical Report, Computer Science Dept., Carnegie Mellon University.
14. D.S. Hochbaum (1997) *Approximation algorithms for NP-hard problems*. PWS Publishing Company, Boston.
15. A. Iske (2003) Progressive scattered data filtering. J. Comput. Appl. Math. **158**(2), 297–316.
16. J. Lee (1991) Comparison of existing methods for building triangular irregular network models of terrain from grid digital elevation models. Int. J. of Geographical Information Systems **5**(3), 267–285.
17. F.P. Preparata and M.I. Shamos (1988) *Computational Geometry*. Springer, New York.
18. A. Said and W.A. Pearlman (1996) A new, fast, and efficient image codec based on set partitioning in hierarchical trees, IEEE Trans. Circuits and Systems for Video Technology **6**, 243–250.
19. L.L. Schumaker (1987) Triangulation methods. Topics in Multivariate Approximation, C. K. Chui, L. L. Schumaker, and F. Utreras (eds.), Academic Press, New York, 219–232.
20. D.B. Shmoys (1995) Computing near-optimal solutions to combinatorial optimization problems. DIMACS, Ser. Discrete Math. Theor. Comput. Sci. **20**, 355–397.

21. Taubman, D. (2000) High performance scalable image compression with EBCOT, IEEE Trans. on Image Processing, July 2000, 1158–1170.
22. D. Taubman, M.W. Marcellin (2002) *JPEG2000: Image Compression Fundamentals, Standards and Practice*, Kluwer, Boston, 2002.