Meshfree Thinning of 3D Point Clouds

Nira Dyn
Tel-Aviv University

Armin Iske
University of Hamburg

Holger Wendland
University of Sussex

July 26, 2007

Abstract

An efficient data reduction scheme for the simplification of a surface given by a large set X of $3D$ point-samples is proposed. The data reduction relies on a recursive point removal algorithm, termed thinning, which outputs a data hierarchy of point-samples for multiresolution surface approximation. The thinning algorithm works with a point removal criterion, which measures the significances of the points in their local neighbourhoods, and which removes a least significant point at each step. For any point x in the current point set $Y \subset X$, its significance reflects the approximation quality of a local surface reconstructed from neighbouring points in Y. The local surface reconstruction is done over an estimated tangent plane at x by using radial basis functions. The approximation quality of the surface reconstruction around x is measured by using its maximal deviation from the given point-samples X in a local neighbourhood of x. The resulting thinning algorithm is meshfree, i.e., its performance is solely based upon the geometry of the input $3D$ surface point-samples, and so it does not require any further topological information, such as point connectivities. Computational details of the thinning algorithm and the required data structures for efficient implementation are explained and its complexity is discussed. Two examples are presented for illustration.

1 Introduction

Surface reconstruction from large sets of $3D$ points is a challenging problem in computer graphics and reverse engineering. In this context, it is usually required, as a preprocessing step, to reduce the complexity of an input scattered point set $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^3$. The discrete set X of unorganized point samples, often referred to as a $3D$ *point cloud*, may, for instance, be acquired by a $3D$ laser scan from the surface of a geometric object. In relevant applications, the size $|X| = N$ of the point cloud X can be of the order of several millions. This makes an efficient surface reconstruction and visualization, even on modern hardware, a rather difficult task.

Complexity reduction from large $3D$ point clouds has therefore become one of the key problems for surface reconstruction and visualization. In this preprocessing task, called *model simplification*, it is required to extract a *small* subset $Y \subset X$, $|Y| \ll |X|$, of points which approximate the given 3D point cloud X sufficiently well, in view of subsequent methods for surface rendering. Moreover, efficient methods for surface visualization often rely on a multiresolution representation of the surface. Such multiresolution models typically require the construction of a suitable coarse-to-fine data hierarchy

$$Y_1 \subset Y_2 \subset \cdots \subset Y_L \subset X \tag{1}$$

from the given data points X, where each point set Y_ℓ in (1), $1 \leq \ell \leq L$, corresponds to one level of the surface multiresolution model.

There exist several different techniques for point-based surface model simplification, including *clustering* [3, 20, 21], *particle simulation* [23], and *iterative simplification* [1, 2, 9, 12], see the survey [19]. Iterative simplification methods are usually preferred, whenever strong emphasis is put on the quality of the reconstructed surface. Moreover, such progressive model simplifications are typically used in combination with multiresolution methods [2, 9]. Relevant techniques for iterative simplification, such as *edge collapse* and *vertex removal*, are usually based on mesh decimation, see [12] for a survey.

Other related methods for iterative simplification include *thinning* [8], a recursive point removal scheme, where the points in X are removed according to some suitable removal criterion. Thinning algorithms are efficient methods for the construction of a point data hierarchy (1), requiring at most $\mathcal{O}(N \log N)$ operations, under mild assumptions, provided that appropriate data structures are used for their implementation.

We remark that thinning can be viewed as the inverse of point insertion. Both concepts, thinning and insertion, essentially require careful management of the data structure needed to represent the resulting data hierarchy (1), where commonly used data structures are usually *mesh-based*. For a recent account on models and data structures for multiresolution mesh representation, we refer to [10]. For a recent survey on mesh-based thinning algorithms and their application to terrain modelling and image compression, see [6].

This paper proposes a *meshfree* thinning algorithm. Unlike previous (mesh-based) thinning algorithms, the meshfree approach of this paper does not require any sophisticated data structures and algorithms for the generation and maintenance of a mesh. Yet, the complexity of our meshfree thinning algorithm is, under mild assumptions, still $\mathcal{O}(N \log N)$.

Our meshfree thinning algorithm works with local surface approximation, for estimating the significance of a point. The local approximation relies on two ingredients: *principal component analysis* for the estimation of a local tangent plane, and a kernel-based smoothing approximation using *compactly supported radial basis functions* [4, 27].

The idea to use local approximations for the reconstruction of a surface from given $3D$ point-samples is not new. In a series of papers by Alexa et. al [1, 2, 9], local surface approximations are constructed by using the method of *moving least squares* (MLS) [16]. As one ingredient of their surface reconstruction method, a rather naive recursive point removal scheme, based on MLS, is suggested, where the choice for a point to be removed is merely based on the *current* set of points. This is in contrast to our more sophisticated point removal scheme, where information from previously removed points is taken into account, leading to a more accurate significance measure. We remark that an accurate significance measure is very crucial for a greedy thinning algorithm, where early removals of significant points cannot be corrected in later stages of the algorithm, as supported by our previous results in [8, 6].

Moreover, the reconstruction method in [1, 2, 9] is limited to *smooth* surfaces, since it is assumed that any surface point has a unique tangent plane, which in turn is essentially used in the subsequent construction of their local surface reconstruction. In contrast, our significance measure can also handle points where the surface is not smooth, by assigning a high significance to a point whose neighbouring points cannot be approximated sufficiently well by a plane.

The outline of this paper is as follows. In Section 2, we first provide a generic introduction to thinning algorithms, and discuss required properties of a meshfree significance measure. Details on the kernel-based local surface reconstruction are explained in Section 3, and relevant results on its approximation properties are reviewed. These results imply desired properties of the significance measure of a point in a point cloud. In Section 4, an effective significance measure is proposed, based on the local approximation method of Section 3. Then, in Section 5 various important computational aspects are discussed, which are required for an efficient implementation of our meshfree thinning algorithm. In particular, it is shown in Section 5 that the computational complexity of our meshfree thinning algorithm is, under mild assumptions, $\mathcal{O}(N \log N)$. Finally, the efficacy and good performance of the proposed meshfree thinning method is illustrated in Section 6, where we use two popular test data sets, called `Happy Buddha` and `Dragon`, taken from the 3D scanning repository of Stanford University [22].

2 Thinning Algorithms

In this section, we first provide a generic introduction to thinning algorithms, and then specify several requirements on our meshfree thinning algorithm.

2.1 Generic Formulation

In order to briefly explain the basic features of thinning, let $Y \subset X$ denote a current subset of X, where initially $Y = X$. Using a specific significance measure, we compute for each point $x \in Y$ a (non-negative) significance value $\sigma(x)$. At each removal step, thinning deletes one point with minimal significance from Y. In this sense, thinning is a *greedy* point removal scheme, whose generic formulation is as follows.

Algorithm 1 (Thinning).

```
INPUT: $X$ with $|X| = N$, and $n \in \{1, \ldots, N-1\}$;
  (1) Let $X_N = X$;
  (2) FOR $k = 1, \ldots, n$
      (2a) Locate a least significant point $x \in X_{N-k+1}$;
      (2b) Let $X_{N-k} = X_{N-k+1} \setminus \{x\}$;
OUTPUT: $Y \equiv X_{N-n} \subset X$, of size $|X_{N-n}| = N - n$.
```

Note that the thinning algorithm computes, on input X, a nested sequence

$$X = X_N \supset X_{N-1} \supset \ldots \supset X_j \supset \ldots \tag{2}$$

of subsets of X, where $|X_j| = j$ for $N - n \leq j \leq N$. Moreover, the required data hierarchy in (1) can be constructed from (2) by selecting L suitable breakpoints $k_1, k_2, \ldots, k_L \in \{n, \ldots, N\}$, satisfying $k_1 < k_2 < \cdots < k_L$, and defining the *coarse-to-fine* nested sequence

$$X_{k_1} \subset X_{k_2} \subset \cdots X_{k_L} \subset X.$$

Note that the implementation of thinning requires a significance measure which allows us, for any point x of a current subset $Y \subset X$, obtained during thinning, to determine its significance $\sigma(x)$ in Y. Then, for given significances $\{\sigma(x) : x \in Y\}$, a point $x^* \in Y$ whose significance is minimal among those of all points in Y is said to be *least significant*, in which case x^* may be removed from Y in step **(2b)** of Algorithm 1. For a more comprehensive introduction to thinning algorithms, we refer to the textbook [15, Chapter 4].

The goal of this paper is the construction of an effective significance measure for thinning from 3D point clouds. The proposed significance measure of this paper is meshfree and allows us (under mild assumptions) to implement thinning (Algorithm 1), at asymptotic computational cost $\mathcal{O}(N \log N)$. Details on the efficient implementation of Algorithm 1, required data structures, and other relevant computational aspects are discussed in Section 5.

2.2 Meshfree Thinning from 3D Point Clouds

Before we give details on our meshfree thinning algorithm, let us make a few prior remarks concerning the representation of a surface and the conclusions which follow from such a representation for constructing a suitable significance measure.

We believe that the design of an effective significance measure is intimately connected to the reconstruction process of the surface. Hence, the following points have to be taken into account.

- The reconstruction of a surface should be *local*, meaning that only the nearest neighbours of a point x should be employed to reconstruct the surface in the neighbourhood of the point x. We will refer to this as *local approximation* and give a detailed description in the next section. The idea of locality should be incorporated into the design of the significance measure. This is important since the significance of a point depends on the local behaviour of the surface, and since locality is closely related to low computational complexity.

- The locality of the significance measure should reflect the local behavior of the surface. In a locality of slow changes, the surface should be reconstructable from less information, than in localities of rapid changes. Hence, data sites in locations of slow changes should have a lower significance measure, than data sites in locations where the surface varies rapidly. We describe very precisely in the next section how we locally approximate the surface and how this influences the definition of the significance measure.

- To ensure that all the available information during a thinning process is used in the estimation of the significances of the current points, the significance measure of a current point incorporates also information from previously removed points from that neighbourhood. To be more precise, suppose $Y \subset X$ is the current set of points, and the significance of $x \in Y$ is computed. Then our local reconstruction of the surface depends only on neighbouring points of x from Y, yet the significance $\sigma(x)$ of x is determined by the deviations of the local reconstruction not only at the point x, but also at a set of already removed points in the neighbourhood of x.

In the following two sections we discuss the local approximation method, and the design of an effective significance measure, which reflects these requirements, i.e., small computational costs and high accuracy.

3 Local Surface Approximation

Probably the two most dominant reconstruction processes from scattered data are *moving least squares* ([16, 26]) and *radial basis functions* ([4, 27]). Here, we employ radial basis functions, or more generally, kernel-based approximation methods to locally describe the surface. We first shortly review the relevant material, starting with *function approximation.* Then, we describe how to locally approximate a surface by considering it as a function relative to a local tangent plane.

3.1 Approximation of Functions using Reproducing Kernel Hilbert Spaces

Suppose $Z = \{z_1, \ldots, z_N\} \subset \Omega \subset \mathbb{R}^d$ is a set of pairwise distinct points, which we will also call *data sites* and $f_1, \ldots f_N \in \mathbb{R}$ are given *function values*, assumed to stem from a function $f \in C(\Omega)$, which is otherwise unknown. It is our goal to create a sufficiently good reconstruction $s_{f,Z}$ of f from the data. To make this more precise, we will assume that the target f actually belongs to a Hilbert space $\mathcal{H}$ of functions defined on the domain Ω. We will further assume that point evaluations are continuous on $\mathcal{H}$. In such a situation, it is well known (cf. [27]) that $\mathcal{H}$ is *a reproducing kernel Hilbert space*, i.e., there exists a unique function $\Phi : \Omega \times \Omega \to \mathbb{R}$ satisfying

1. $\Phi(\cdot, x) \in \mathcal{H}$ for all $x \in \Omega$,
2. $f(x) = (f, \Phi(\cdot, x))_{\mathcal{H}}$ for all $x \in \Omega$ and all $f \in \mathcal{H}$.

In this setting, it seems to be natural to define the reconstruction $s_{f,Z} \equiv s_0$ as the *minimal-norm* interpolant, i.e., as the solution of

$$\min\{\|s\|_{\mathcal{H}} : s \in \mathcal{H},\ s(z_j) = f(z_j), 1 \le j \le N\}. \tag{3}$$

Again, it is well known (cf. [27]) that the solution to this minimization problem can be written as a linear combination of the reproducing kernel Φ of $\mathcal{H}$:

Lemma 3.1 *The solution s_0 of (3) can be represented as*

$$s_0(x) = \sum_{j=1}^{N} \alpha_j \Phi(x, z_j), \tag{4}$$

where the coefficient vector $\alpha = (\alpha_1, \ldots, \alpha_N)^T \in \mathbb{R}^N$ can be computed by solving the linear system $A_{\Phi,Z}\alpha = f|Z$ with the coefficient matrix

$$A_{\Phi,Z} = (\Phi(z_i, z_j)).$$

It is important for the design of a significance measure, to understand the approximation quality of the interpolant $s_0 = s_{f,Z}$. This quality is measured in terms of the so called *fill distance* or *mesh norm*

$$h_{Z,\Omega} := \sup_{x \in \Omega} \min_{z_j \in Z} \|x - z_j\|,$$

which gives the radius of the largest data site free ball in Ω, where $\|\cdot\|$ denotes the Euclidean norm on $\mathbb{R}^d$. Furthermore, since we employ *Wendland* functions [25] for the reconstruction process, which are known to be reproducing kernels in Sobolev spaces, we restrict ourselves here to such Sobolev spaces $H^\tau(\Omega)$ with $\tau > d/2$, ensuring that point evaluation functionals are continuous by Sobolev's embedding theorem (cf. [11]). In such a situation, the following result is known (cf. [17]).

Lemma 3.2 *Suppose $f \in H^\tau(\Omega)$, where $\tau > d/2$ and Ω is a bounded region with a sufficiently smooth boundary. Then, for dense enough data sets, the error between f and its reconstruction $s_{f,Z}$ based upon the reproducing kernel of $H^\tau(\Omega)$ satisfies the estimate*

$$\|f - s_{f,Z}\|_{L_\infty(\Omega)} \leq C h^{\tau - d/2} \|f\|_{H^\tau(\Omega)},$$

where $h = h_{Z,\Omega}$ and $C > 0$ is a constant depending only on Ω but not on Z or f.

This result clearly shows that for a smooth target function f, only few data sites are necessary to well approximate (reconstruct) f up to a given accuracy by scattered data interpolation, when employing a smooth kernel. However, in general the precise degree of smoothness of the target function f is unknown and it often happens that a rough target function is approximated by a smoother kernel. In such a situation, it is well-known that the interpolation operator is in general *not* bounded. Nonetheless, the following result holds (cf. [18]).

Lemma 3.3 *Suppose $\tau \geq \beta > d/2$. Suppose the target function f comes from the rougher Sobolev space $H^\beta(\Omega)$, while the interpolant $s_{f,Z}$ is formed using the kernel of the smoother Sobolev space $H^\tau(\Omega)$. Then,*

$$\|f - s_{f,Z}\|_{L_\infty(\Omega)} \leq C h^{\beta - d/2} \rho^\tau \|f\|_{H^\beta(\Omega)},$$

where

$$\rho = \rho_{Z,\Omega} := \frac{h_{Z,\Omega}}{q_Z}, \qquad q_Z = \min_{j \neq k} \|z_j - z_k\|.$$

Note that the *uniformity measure* ρ always satisfies $\rho \geq 1$ and is close to one if the data set is almost uniformly distributed, while it tends to infinity if the data set becomes more and more non uniform.

Both results have some important consequences for our way of defining a significance measure. We can conclude that:

- the error bound is larger, if the target function is less smooth,
- the error bound is larger for a non uniform data set, than for a uniform one.

This implies for the significance of a data site:

- If the function is smooth around the data site and if there are still enough other data sites in the neighbourhood, the data site should have a low significance.
- If the function is rough around the data site and if there are still enough other data sites in the neighbourhood, the data site should have a medium significance.
- If the function is rough around the data site and if there are only few other data sites in the neighbourhood, the data site should have a high significance.

Note that the error estimates indicate that it is not only necessary for the function to be smooth but also to have a small Sobolev norm. Unfortunately, this norm is in general not at our disposal.

We will give the exact definition of our significance measure in the next section. Before, we will address another computational issue. Interpolation in the above form has two disadvantages. The first one comes from the fact that the corresponding interpolation matrix $A_{\Phi,Z}$ becomes ill conditioned whenever two data sites are too close together. More precisely, the smallest eigenvalue $\lambda_{\min}(A_{\Phi,Z})$ of $A_{\Phi,Z}$ can be bounded from below by

$$\lambda_{\min}(A_{\Phi,Z}) \geq C q_Z^{2\tau-d},$$

which becomes worse for smoother kernels (cf. [27, Chapter 12]). The second drawback always appears when the data is noisy. In such a situation interpolation is not the right tool. Instead of solving (3) it is then more appropriate to solve

$$\min \left\{ \sum_{j=1}^{N} [s(z_j) - f_j]^2 + \lambda \|f\|_{\mathcal{H}}^2 \right\}. \tag{5}$$

The solution s_λ to (5) can be written in the form (4). The only difference is that the coefficient vector $\alpha \in \mathbb{R}^N$ is now chosen by solving the linear system

$$(A_{\Phi,Z} + \lambda I)\alpha = f|Z,$$

where I denotes the identity matrix. Obviously, there is the question of how to choose the *smoothing parameter* $\lambda > 0$. This has been extensively studied in particular in the context of *smoothing splines*, see for example [24]. In our application, we follow a new, deterministic approach from [28]. The error estimate for s_λ, under the conditions of Lemma 3.2, is

$$\|f - s_\lambda\|_{L_\infty(\Omega)} \le C \left\{ h^{\tau - d/2} + \sqrt{\lambda} \right\} \|f\|_{H^\tau(\Omega)}. \tag{6}$$

This together with the fact that

$$\lambda_{\min}(A_{\Phi,Z} + \lambda I) \ge \lambda$$

indicates that a good choice for the smoothing parameter λ is given by $\lambda \approx h^{2\tau - d}$. We term the approximation so constructed by *smoothing approximation*. Note that (6) indicates that our conclusions, which we have derived for interpolation also hold for smoothing approximation at least if the target function belongs to the reproducing kernel Hilbert space. A result analogous to Lemma 3.3 for smoothing approximation is still missing, but numerical tests indicate that the smoothing approximation behaves similarly to the interpolant, also in the case of a rougher target function.

3.2 Local Surface Approximation by Projection

Here we indicate how function approximation can be used for surface approximation. Let $\mathcal{S} \subset \mathbb{R}^3$ denote the surface, and let x be a point on the surface. Suppose that we have a set $\mathcal{N} \subset \mathcal{S}$ of neighbouring points to x. Our local surface approximation is defined by the following two steps

1. Estimate a tangent plane $\mathcal{T}$ to $\mathcal{S}$ through x using the information given by $\mathcal{N}$.
2. Project the points from $\mathcal{N}$ to $\mathcal{T}$ to define the data sites Z, and use the signed distance from each point y in $\mathcal{N}$ to its projection point on $\mathcal{T}$ as the data values. The points on the smoothing approximation to this data, in the vicinity of the point x, constitute the local approximation to the surface in the vicinity of x.

Obviously, both steps might be problematic, especially for surface points, where the surface is locally subject to strong variations, e.g. where the surface is oscillatory or in the vicinity of feature points (points where the surface is not smooth). In such critical situations, the information in $\mathcal{N}$ may either not suffice to determine a unique tangent plane, or two or more points from $\mathcal{N}$ are projected to the same point on the tangent plane. In either case, we mark the point x as indispensable by assigning a high significance measure to it. This allows us to handle smooth surfaces with features.

The significance measure of a point on the surface, where the surface is smooth, is presented in the next section. We end this section by describing the procedure we use for estimating the tangent plane $\mathcal{T}$.

The approach taken in this paper is based on a *principal component analysis* (PCA). PCA has been applied in this context by Hoppe in [13, 14]. To estimate from $\mathcal{N}$ a local tangent plane to $\mathcal{S}$ at x and its normal vector, we first compute the *center of gravity*

$$\widehat{x(\mathcal{N})} = \widehat{x} = \frac{1}{|\mathcal{N}|} \sum_{y \in \mathcal{N}} y$$

and then the *covariance matrix*

$$\mathrm{cov}(\mathcal{N}) = \sum_{y \in \mathcal{N}} (y - \widehat{x})(y - \widehat{x})^T \in \mathbb{R}^{3 \times 3}.$$

Note that $\mathrm{cov}(\mathcal{N})$ is symmetric and positive semi-definite, and so its three eigenvalues λ_1, λ_2, and λ_3 are non-negative, and its eigenvectors, corresponding to different eigenvalues, are orthogonal. Now the eigenvalues of $\mathrm{cov}(\mathcal{N})$ can be used to measure whether the points in $\mathcal{N}$ are close to a plane or not. To be more precise, if two eigenvalues of $\mathrm{cov}(\mathcal{N})$, say λ_1 and λ_2, are close to each other and the third one, λ_3, is much smaller, then the eigenvectors corresponding to λ_1 and λ_2 span the tangent plane whereas the eigenvector corresponding to λ_3 determines the normal to it.

This procedure not only gives us a way of estimating a tangent plane, based on $\mathcal{N}$, but also gives us a quantitative way of marking a point to be indispensible, whenever the situation of two large eigenvalues and one small eigenvalue does not prevail for $\mathrm{cov}(\mathcal{N})$.

4 Significance Measure

In this section, we propose an effective significance measure denoted by σ_∞, at points which we estimate to be points where the surface is smooth. For any such $x \in Y$, $Y \subset X$, the significance $\sigma_\infty(x)$ measures the *quality* of the surface approximation in the local neighbourhood of x, based on a small number of closest points to x in Y.

To explain the construction of σ_∞ in details, we first introduce some notions and notation. Recall that $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^3$ denotes a (large) 3D point cloud sampled from an unknown surface $\mathcal{S}$. For any subset $Y \subset X$, obtained during the thinning algorithm, and for any $x \in Y$ we denote by $\mathcal{N}_x^Y \subset Y \setminus \{x\}$ a (small) set of neighbours of x in Y. The *neighbourhood* sets $\{\mathcal{N}_x^Y : x \in Y\}$ are constructed during the performance of the thinning algorithm. Initially, when $Y = X$, $\mathcal{N}_x^X$ consists of the m closest points to x in $X \setminus \{x\}$, where $m = \mathcal{O}(1)$ is a fixed parameter determined at the beginning of the algorithm. Note that x is not contained in $\mathcal{N}_x^X$. Moreover, by our way of updating the neighbourhood sets, x will at no stage of the thinning algorithm be contained in any of its neighbourhood sets $\mathcal{N}_x^Y$, for $Y \subset X$.

To further explain this, during the thinning algorithm the current neighbourhood sets $\{\mathcal{N}_x^Y : x \in Y\}$ are updated, after the removal of a least significant point $x^* \in Y$ (in step **(2b)** of Algorithm 1). Only neighbourhood sets which contain x^* are updated. This is done by replacing x^* in such a neighbourhood set $\mathcal{N}_x^Y$, by one closest point to x from

$$\bigcup_{y \in \mathcal{N}_x^Y} \mathcal{N}_y^Y \setminus (\{x\} \cup \mathcal{N}_x^Y).$$

We assume that the above set is not empty, which is an assumption that excludes isolated clusters of points in X and during the thinning algorithm. As observed in all our numerical experiments, this *non-clustering* assumption holds for reasonable point clouds X.

The points in $\mathcal{N}_x^Y$ are used to compute a *local* approximation to the surface $\mathcal{S}$ in the neighbourhood of the surface point $x \in \mathcal{S}$, as explained in the previous section. First we determine an estimated tangent plane $\mathcal{T}_x^Y$ at the surface point x and its normal $\mathbf{n}_x^Y$. If this stage is successful, then the points from the local neighbourhood $\mathcal{N}_x^Y$ are projected onto the tangent plane $\mathcal{T}_x^Y$. Denoting by $\pi_x^Y : \mathbb{R}^3 \to \mathcal{T}_x^Y$ the orthogonal projection onto $\mathcal{T}_x^Y$, we define for any $y \in \mathbb{R}^3$,

$$\delta(y; \mathbf{n}_x^Y) = (y - \pi_x^Y(y)) \cdot \mathbf{n}_x^Y,$$

as the *signed distance* between y and $\mathcal{T}_x^Y$.

Only in case $\pi_x^Y(y) \neq \pi_x^Y(z)$ for $y \neq z,\ y, z \in \mathcal{N}_x^Y$, a local approximation $s(\cdot; \mathcal{N}_x^Y) : \mathcal{T}_x^Y \to \mathbb{R}$ to the surface $\mathcal{S}$ at x is constructed. It is the smoothing approximation fitting the data

$$\left\{ (\pi_x^Y(y), \delta(y; \mathbf{n}_x^Y)) : y \in \mathcal{N}_x^Y \right\}.$$

The significance in Y, of any $x \in Y$, for which such a local approximation is constructed, is given by

$$\sigma_\infty(x) = \max_{y \in \mathcal{M}_x^Y} |s(\pi_x^Y(y); \mathcal{N}_x^Y) - \delta(y; \mathcal{N}_x^Y)|, \tag{7}$$

where $\mathcal{M}_x^Y$ is a set of points from the neighbourhood of x in $(X \setminus Y) \cup \{x\}$, called the *test set* of x relative to Y.

The construction of the test sets $\{\mathcal{M}_x^Y : x \in Y\}$, being maintained during the performance of the thinning algorithm, is done as follows. With the initialization of the thinning algorithm, when $Y = X$, we let $\mathcal{M}_x^X = \{x\}$ for every $x \in X$. During the thinning algorithm, the current test sets $\{\mathcal{M}_x^Y : x \in Y\}$ are updated after the removal of a least significant point $x^* \in Y$ (in step **(2b)** of Algorithm 1). The update of the test sets is done by distributing the points from $\mathcal{M}_{x^*}^Y$ among the test sets $\{\mathcal{M}_x^Y : x \in \mathcal{N}_{x^*}^Y\}$. More precisely, for each $y \in \mathcal{M}_{x^*}^Y$, a closest point x is located among the points in $\mathcal{N}_{x^*}^Y$, i.e., $\|y-x\| = \min_{z\in\mathcal{N}_{x^*}^Y} \|y-z\|$, and y is added to the test set $\mathcal{M}_x^Y$. The point $y \in \mathcal{M}_{x^*}^Y$ is added to a unique test set. The number of test sets that are updated is at most $|\mathcal{N}_{x^*}^Y|$.

It is important to note that the test sets so constructed, form a partitioning of X, i.e., for any set Y generated during the thinning algorithm, the test sets $\{\mathcal{M}_x^Y : x \in Y\}$ are pairwise disjoint and their union coincides with X.

The significance $\sigma_\infty(x)$ of $x \in Y$ in (7) measures the quality of the smoothing approximation $s(\cdot;\mathcal{N}_x^Y)$ in the local neighbourhood of x. Indeed, a small value $\sigma_\infty(x)$ indicates that the surface $\mathcal{S}$ is smooth in the vicinity of x, and/or that the density of the points $\{\pi_x^Y(y) \ : \ y \in \mathcal{N}_x^Y\}$ is high (see Subsection 3.1), while a large value $\sigma_\infty(x)$ indicates the opposite. Hence, $\sigma_\infty(x)$ is a reasonable significance measure.

It is easy to conclude from the thinning algorithm, from the definition of the test sets, and from the significance measure σ_∞, approximation results about the quality of the approximation of X and $\mathcal{S}$ by the current set Y, generated by the thinning algorithm, and by the union of the corresponding local approximating surfaces.

To formulate these results, we recall that for two non-empty point sets, A and B, the *directed Hausdorff distance* $H(A,B)$ from A to B is defined by

$$H(A,B) = \sup_{a\in A} \inf_{b\in B} \|a-b\|.$$

Note that for $A \subset B$, $H(A,B) = 0$. The *Hausdorff distance* between the sets A and B is defined to be $\mathrm{haus}(A,B) = \max\{H(A,B), H(B,A)\}$.

Corollary 4.1 *At any stage of the thinning algorithm,*

$$\mathrm{haus}(X,Y) = H(X,Y) \le R_Y, \tag{8}$$

with $Y \subset X$ denoting the current set of points, and

$$R_Y = \max_{y\in Y} \max_{x\in\mathcal{M}_y^Y} \|x-y\|.$$

An approximation to $\mathcal{S}$ during thinning with current set Y is the set of points

$$\mathcal{G}_Y = \bigcup_{y\in Y} \left\{\pi_y^Y(x) + \mathbf{n}_y^Y s(\pi_y^Y(x);\mathcal{N}_y^Y) : x \in \mathbb{R}^3, \ \|x-y\| \le R_y^Y\right\},$$

with $R_y^Y = \max_{x\in\mathcal{M}_y^Y} \|x-y\|$, defined for any $y \in Y$.

Corollary 4.2 *At any stage of the thinning algorithm, with $Y \subset X$ denoting the current set of points, we have*

$$H(X,\mathcal{G}_Y) \le \max_{y\in Y} \sigma_\infty(y).$$

A direct consequence of the last corollary is

$$H(\mathcal{S}, \mathcal{G}_Y) \leq \max_{y \in Y} \sigma_\infty(y) + H(\mathcal{S}, X),$$

where $H(\mathcal{S}, X)$ measures the quality of the representation of the surface $\mathcal{S}$ by the given point cloud X. The last two inequalities are of interest only if for each $y \in Y$ a local approximation is constructed, and $\sigma_\infty(y)$ is obtained by (7).

5 Implementation and Computational Aspects

The discussion in this section is concerning data structures which are used in order to obtain an *efficient* implementation of our meshfree thinning algorithm. Recall from the discussion in Section 2, that our aim is to implement meshfree thinning, Algorithm 1, at complexity $\mathcal{O}(N \log N)$.

Let us elaborate on the tasks, required for updating the data after the removal of a least significant point, before explaining further details on the utilized data structures. Note that by the removal of a point $x^* \in Y$ from Y, the points from its *dual neighbourhood*,

$$\mathcal{D}_{x^*}^Y = \{x \in Y : x^* \in \mathcal{N}_x^Y\}, \tag{9}$$

are affected, since x^* is removed from their neighbourhoods. Therefore the significances $\{\sigma_\infty(y) : y \in \mathcal{D}_{x^*}^Y\}$ need to be updated, after the update of the neighbourhoods $\{\mathcal{N}_y^Y : y \in \mathcal{D}_{x^*}^Y\}$. Also the significances of the points in $\mathcal{N}_{x^*}^Y$ have to be updated, since their test sets $\{\mathcal{M}_x^Y : x \in \mathcal{N}_{x^*}^Y\}$ are updated with the removal of x^*. Thus, steps **(2a)**–**(2b)** in Algorithm 1 of our meshfree thinning algorithm are accomplished by the performance of the following tasks:

(T1) Locate a least significant point $x^* \in Y$ with $\sigma_\infty(x^*) = \min_{x \in Y} \sigma_\infty(x)$;

(T2) Remove x^* from the data structure.

(T3) Distribute the points in the test set $\mathcal{M}_{x^*}^Y$ among the points in the test sets $\{\mathcal{M}_x^Y : x \in \mathcal{N}_{x^*}^Y\}$, as explained in Section 4.

(T4) Update the neighbourhood sets $\{\mathcal{N}_x^Y : x \in \mathcal{D}_{x^*}^Y\}$, as explained in Section 4, and update the dual neighbourhood sets $\{\mathcal{D}_x^Y : x \in \mathcal{N}_{x^*}^Y\}$ accordingly.

(T5) For each $y \in \mathcal{D}_{x^*}^Y \cup \mathcal{N}_{x^*}^Y$, update its significance $\sigma_\infty(y)$.

(T6) Update the data structure.

In order to analyze the required computational costs, first note that we store the indices of the points in the neighbourhoods $\{\mathcal{N}_x^Y : x \in Y\}$, the dual neighbourhoods $\{\mathcal{D}_x^Y : x \in Y\}$ and the test sets $\{\mathcal{M}_x^Y : x \in Y\}$, separately, and thus need to update their index sets only. Moreover, note that the number of points in $\mathcal{D}_x^Y$ depends on the density of the points in Y in the vicinity of x. Here we assume that $|\mathcal{D}_x^Y| = \mathcal{O}(1)$ for any $x \in Y$, as observed in all our numerical experiments. Under this reasonable assumption and the non-clustering assumption of the previous section, and since $|\mathcal{N}_{x^*}^Y| = m$, the updates in task **(T4)**, **(T5)** require $\mathcal{O}(1)$ operations.

As for the complexity of task **(T3)**, note that the sizes of the test sets $\{\mathcal{M}_x^Y : x \in Y\}$ are increasing during the performance of the thinning algorithm. Under the simplistic assumption that the points in $X \setminus Y$ are uniformly distributed over the test sets $\{\mathcal{M}_x^Y : x \in Y\}$, the distribution of the points in the test set $\mathcal{M}_{x^*}^Y$ among the test sets $\{\mathcal{M}_x^Y : x \in \mathcal{N}_{x^*}^Y\}$ requires $c^*(N-n)/n$ operations, where $N = |X|$, $n = |Y|$, and $c^* = \mathcal{O}(1)$ is the cost of finding a nearest point among m points.

Task **(T6)** can usually be accomplished in $\mathcal{O}(\log N)$ operations, provided that additional pointers to the nodes of the data structure are stored. Another possibility, which does not require such an interference with the internal data structure is based upon a *balanced binary search tree.* In fact, this is the data structure which we employed in our implementation. To be more precise, we use an AVL-tree, even if any balanced binary search tree would do it. To built the tree we define an ordering on the points by using their significances σ_∞. Hence, a point x is *smaller* (i.e., less significant) than another point y, iff $\sigma_\infty(x) < \sigma_\infty(y)$. In case of equality, we use their point indices to decide on their ordering in the tree.

We remark that an AVL-tree, like a heap, can be built in $\mathcal{O}(N \log N)$ operations [5]. Moreover, any removal or insertion (or update of) a node can be done in $\mathcal{O}(\log N)$ operations, where this includes operations required for rebalancing the tree. Hence, when using an AVL-tree, under the assumption that the number of points in $\mathcal{D}_x^Y$ is $\mathcal{O}(1)$, task **(T5)** requires only $\mathcal{O}(\log N)$ operations.

Likewise, in task **(T2)** the deletion of the least significant point x^* from the AVL-tree can be accomplished in only $\mathcal{O}(\log N)$ operations. We remark that an AVL-tree is organized such that the *left most* element in the tree is least significant. Thus, in task **(T1)** x^* can be located by $\mathcal{O}(\log N)$ operations, since the (balanced) AVL-tree has depth $\mathcal{O}(\log N)$.

It remains to discuss the computational complexity of the preprocessing stage of finding m nearest neighbours to a point $x \in X$. To solve this problem efficiently, we use another tree-based data structure, namely a *kd*-tree [5] to store the points in X. A *kd*-tree is a binary tree which can also be built in $\mathcal{O}(N \log N)$ operations. Moreover, the k nearest neighbours problem can be answered in $\mathcal{O}(k \log N)$ operations. Therefore, under the assumption $m = \mathcal{O}(1)$, the neighbourhood set $\mathcal{N}_x^X$ can be obtained in only $\mathcal{O}(\log N)$

operations for any $x \in X$. Moreover, note that the construction of the dual neighbourhoods $\mathcal{D}_x^X$, for all $x \in X$, from the sets $\mathcal{N}_x^X$, $x \in X$, requires only $\mathcal{O}(N)$ operations. For more details concerning *kd*-trees, we refer to the textbook [5].

In our implementation, we aimed to have in the neighbourhood set $\mathcal{N}_x^Y$ the closest m points to x from $Y \setminus \{x\}$, for each $x \in Y$, and for each set Y generated during the thinning algorithm. In order to keep the computational complexity as required, the updates in task **(T4)** in our implementation, are based on the initial *kd*-tree. Although not all the neighbourhood sets consist of the m closest points, this strategy gives good results if $N - n$ is still large enough, and guarantees that the updates of $\{\mathcal{N}_x^Y : x \in \mathcal{D}_{x^*}^Y\}$ in task **(T4)** can be performed by $\mathcal{O}(1)$ operations. Note that by this strategy, any point from $Y \setminus \{x^*\}$ can replace x^* in $\{\mathcal{N}_x^Y : x \in \mathcal{D}_{x^*}^Y\}$, yet the number of replacing points, which is the number of dual neighbourhoods that have to be updated, is at most $|\mathcal{D}_{x^*}^Y|$, which according to our assumption is $\mathcal{O}(1)$, and so the updates of $\{\mathcal{D}_x^Y : x \in \mathcal{N}_{x^*}^Y\}$ in task **(T4)** in our implementation, can be performed by $\mathcal{O}(1)$ operations.

In summary, building the initial AVL-tree, the *kd*-tree, and the sets $\mathcal{N}_x^X$, $\mathcal{D}_x^X$, $\mathcal{M}_x^X$, for all $x \in X$, requires only $\mathcal{O}(N \log N)$ operations. Task **(T1)** requires $\mathcal{O}(\log N)$ operations, and so do tasks **(T2)** and **(T6)**. Tasks **(T4)** and **(T5)** for both strategies of updating the neighbourhoods, require, under our above assumptions, $\mathcal{O}(1)$ operations, whereas task **(T3)** requires $c^*(N-n)/n$ operations.

Therefore, the total cost needed for the performance of tasks **(T1)**–**(T6)** after the removal of *one* least significant point from Y, with $n = |Y|$, is $\mathcal{O}(\log N) + c^*(N-n)/n$. Since the number of point removals is bounded by N, the total computational complexity of our meshfree thinning algorithm is (under the above assumptions) only at most $\mathcal{O}(N \log N)$ operations, as stated in the introduction.

6 Numerical Simulations

We have implemented the proposed thinning algorithm *efficiently* by using the data structures AVL-tree and *kd*-tree, as explained in Section 5. The update of $\mathcal{N}_x^Y$, for $x \in \mathcal{D}_{x^*}^Y$, and the distribution of the points in the test set $\mathcal{M}_{x^*}^Y$, as required after the removal of x^*, is based on the initial *kd*-tree.

In order to illustrate the efficacy of our meshfree thinning algorithm, we applied our implementation to two standard test data sets from the 3D scanning repository of Stanford University [22]. These two data sets, called `Happy Buddha` and `Dragon`, are displayed in Figure 1 **(a)** and **(b)**.

In either test case, we decided to use one of the compactly supported functions Φ from [25] for the construction of the local surface approximations. We remark that this choice for Φ allows us to effectively control the

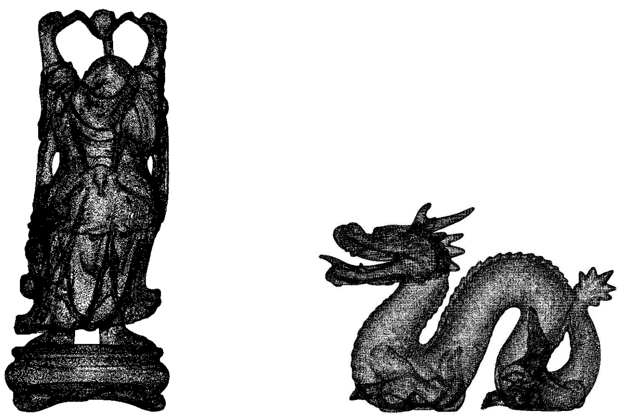

Figure 1: Two test data sets from the Stanford 3D scanning repository. **(a)** `Happy Buddha` of size $|X| = 543,521$; **(b)** `Dragon` of size $|X| = 435,544$.

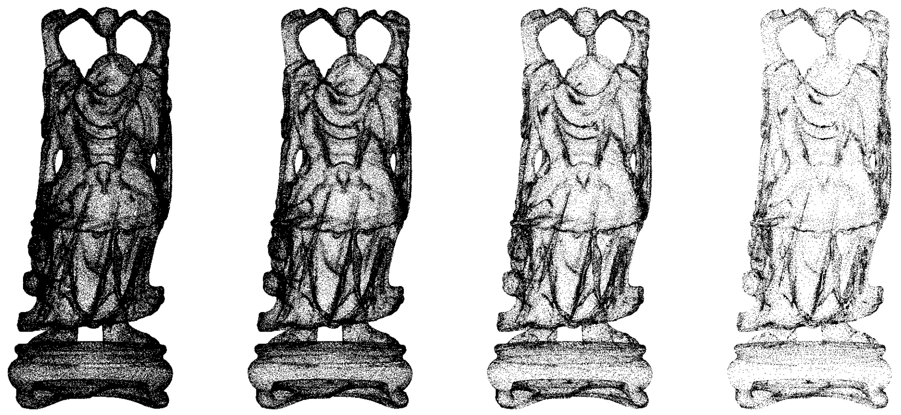

Figure 2: `Happy Buddha`. Data hierarchy output by meshfree thinning.

smoothness of the surface approximation s in (4). A popular alternative for the choice of the kernel are Duchon's celebrated *thin plate splines* [7], $\Phi(x, y) = \|x - y\|^2 \log(\|x - y\|)$, which would provide comparable numerical results, at slightly larger computational costs though, since in this case linear polynomials need to be added to the form of the minimizer in (4).

Figure 2 shows for the first test case, `Happy Buddha`, a nested sequence $Y_1 \subset Y_2 \subset Y_3 \subset Y_4 \subset X$ of four 3D point clouds, which were output by our meshfree thinning algorithm. The sizes of the subsets Y_ℓ are 10 %, 25 %, 50 %, and 75 % of the size $|X| = 543,521$ of the original data X. For the other test case, `Dragon`, Figure 3 shows a data hierarchy of four 3D point clouds, which were output by our meshfree thinning algorithm, comprising 10 %, 25 %, 50 %, and 75 % of points from the initial point cloud consisting of $|X| = 435,544$ data points.

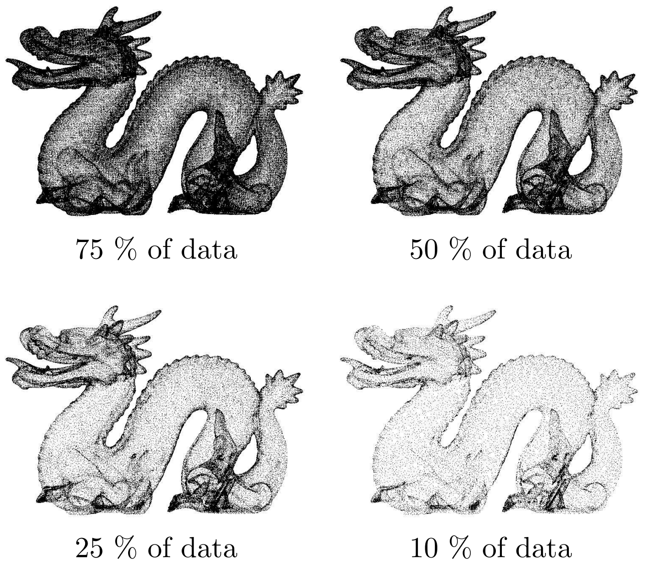

Figure 3: `Dragon`. Data hierarchy output by meshfree thinning.

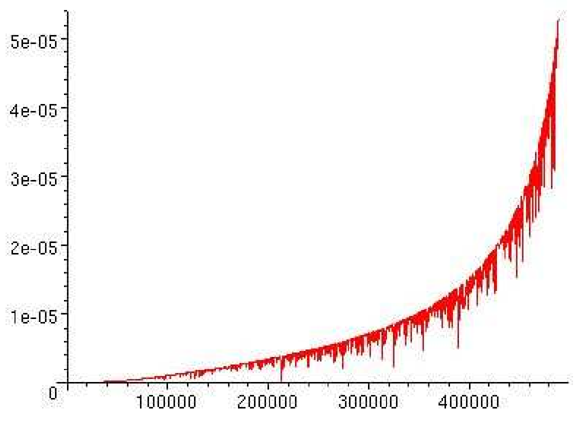

Figure 4: `Happy Buddha`. Least significances during meshfree thinning.

Note that in either test case, each of the four subsets Y_ℓ, $1 \leq \ell \leq 4$, in the constructed data hierarchy, manage to capture the geometric features of the geometric object remarkably well. Moreover, also their topology is maintained correctly.

Finally, Figure 4 shows for the test case `Happy Buddha` the graph of the significances $\sigma_\infty(x^*)$ as a function of the removal step k in Algorithm 1. Note that the least significances are not necessarily monotonically increasing. Yet the global trend of the significances is monotonically increasing.

References

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C.T. Silva. Point set surfaces. In *IEEE Visualization*, pages 21–28, 2001.

[2] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C.T. Silva. Computing and rendering point set surfaces. *IEEE Trans. Vis. Comput. Graph.*, 9(1):3–15, 2003.

[3] D. Brodsky and B. Watson. Model simplification through refinement. In *Proceedings of Graphics Interface*, pages 221–228, Montreal, May 2000. AK Peters.

[4] M. D. Buhmann. *Radial Basis Functions.* Cambridge University Press, Cambridge, UK, 2003.

[5] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms.* MIT Press, Cambridge, Massachusetts, 2nd edition, 2001.

[6] L. Demaret, N. Dyn, M.S. Floater, and A. Iske. Adaptive thinning for terrain modelling and image compression. In N.A. Dodgson, M.S. Floater, and M.A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, pages 321–340, Berlin, 2005. Springer.

[7] J. Duchon. Splines minimizing rotation-invariant semi-norms in Sobolev spaces. In W. Schempp and K. Zeller, editors, *Constructive Theory of Functions of Several Variables*, pages 85–100. Springer, 1977.

[8] N. Dyn, M.S. Floater, and A. Iske. Adaptive thinning for bivariate scattered data. *Journal of Computational and Applied Mathematics*, 145(2):505–517, 2002.

[9] S. Fleishman, D. Cohen-Or, M. Alexa, and C.T. Silva. Progressive point set surfaces. *ACM Trans. Graph.*, 22(4):997–1011, 2003.

[10] L. De Floriani and P. Magillo. Multiresolution mesh representation: models and data structures. In A. Iske, E. Quak, and M.S. Floater, editors, *Tutorials on Multiresolution in Geometric Modelling*, pages 363–417, Berlin, 2002. Springer.

[11] F.G. Friedlander and M.S. Joshi. *Introduction to the Theory of Distributions.* Cambridge University Press, Cambridge, UK, 2nd edition, 1999.

[12] M. Garland and P. Heckbert. Surface simplification using quadratic error metrics. In *SIGGRAPH'97*, Washington, DC, 1997. ACM.

[13] H. Hoppe. *Surface Reconstruction from Unorganized Points.* PhD thesis, University of Washington, 1994.

[14] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *SIGGRAPH'92*, volume 26, pages 71–78, Washington, DC, 1992. ACM.

[15] A. Iske. *Multiresolution Methods in Scattered Data Modelling.* Springer, Berlin, 2004.

[16] D. Levin. The approximation power of moving least-squares. *Math. Comput.*, 67:1517–1531, 1998.

[17] F. J. Narcowich, J. D. Ward, and H. Wendland. Sobolev bounds on functions with scattered zeros, with applications to radial basis function surface fitting. *Math. Comput.*, 74:643–763, 2005.

[18] F. J. Narcowich, J. D. Ward, and H. Wendland. Sobolev error estimates and a Bernstein inequality for scattered data interpolation via radial basis functions. *Constructive Approximation*, 24:175–186, 2006.

[19] M. Pauly, M. Gross, and L. P. Kobbelt. Efficient simplification of point-sampled surfaces. In *Proceedings of the conference on Visualization'02*, pages 163–170, Washington, DC, 2002. IEEE.

[20] J. Rossignac and P. Borrel. Multi-resolution 3d approximations for rendering complex scenes. In B. Falcidieno and T.L. Kunii, editors, *Modeling in Computer Graphics: Methods and Application*, pages 455–465, Berlin, 1993. Springer.

[21] E. Shaffer and M. Garland. Efficient adaptive simplification of massive meshes. In *Proceedings of the conference on Visualization'01*, pages 127–134, Washington, DC, 2001. IEEE.

[22] Stanford. The stanford 3d scanning repository, 2005. Stanford Computer Graphics Laboratory, University of Stanford, `http://graphics.stanford.edu/data/3Dscanrep/`.

[23] G. Turk. Re-tiling polygonal surfaces. In *SIGGRAPH'92*, Washington, DC, 1992. ACM.

[24] G. Wahba. Spline models for observational data. In *CBMS-NSF, Regional Conference Series in Applied Mathematics*, Philadelphia, 1990. SIAM.

[25] H. Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Adv. Comp. Math.*, 4:389–396, 1995.

[26] H. Wendland. Local polynomial reproduction and moving least squares approximation. *IMA J. Numer. Anal.*, 21:285–300, 2001.

[27] H. Wendland. *Scattered Data Approximation*. Cambridge University Press, Cambridge, UK, 2005.

[28] H. Wendland and C. Rieger. Approximate interpolation with applications to selecting smoothing parameters. *Numerische Mathematik*, 101:643–662, 2005.