Max–Planck–Institut für biologische Kybernetik
Max Planck Institute for Biological Cybernetics

Technical Report No. TR-179

Simultaneous Implicit Surface Reconstruction and Meshing

Joachim Giesen[1], Markus Maier,[2]
Bernhard Schölkopf[3]

November 2008

[1] Friedrich-Schiller-Universität Jena, email: giesen@informatik.uni-jena.de
[2] MPI for Biological Cybernetics, Department Schölkopf, email: markus.maier@tuebingen.mpg.de
[3] MPI for Biological Cybernetics, Department Schölkopf, email: bs@tuebingen.mpg.de

Simultaneous Implicit Surface Reconstruction and Meshing

Joachim Giesen, Markus Maier, Bernhard Schölkopf

Abstract. We investigate an implicit method to compute a piecewise linear representation of a surface from a set of sample points. As implicit surface functions we use the weighted sum of piecewise linear kernel functions. For such a function we can partition $\mathbb{R}^d$ in such a way that these functions are linear on the subsets of the partition. For each subset in the partition we can then compute the zero level set of the function exactly as the intersection of a hyperplane with the subset.

1 Introduction

A classic problem in point based graphics is to turn point cloud data in $\mathbb{R}^3$ into a continuous surface model—the so-called surface reconstruction problem. Besides its practical aspects this problem has spurred the development of many new and interesting mathematical and computational techniques in topology, sampling theory and level set methods.

Two of the most important continuous surface representations used in surface reconstruction are triangle meshes and implicit surfaces. Both representations have their advantages and disadvantages.

In the implicit representation a surface is given by the zero level set of a function mapping the space in which the surface is embedded to the reals. There are a number of editing operations that can be computed efficiently on level set surface models (see, for example, [MBWB02]). If our goal is an implicit surface representation of the surface, the surface reconstruction problem reduces to the estimation of an implicit surface function that fits the point cloud data. There is a vast literature about the definition and computations of implicit surface functions from point cloud data using variational level set methods ([MT93]; [ZOMK00]; [ZOF01]; [ZO02]). [CBC*01] use radial basis functions, which are centered on the data points, to estimate the implicit surface function. Since the problem of finding a good implicit surface function from a set of data points is very similar to regression problems considered in machine learning, this problem recently also received some attention from the machine learning community. [SSB05] propose an approach which is based on modified Support Vector regression. [WSC06] use multi-scale compactly supported basis functions and a novel regularization method, in order to obtain globally smooth implicit surface functions.

Also the representation of a surface as a (triangle) mesh has advantages. For many operations on surfaces like smoothing or rendering a mesh representation is better suited than an implicit surface representation. Also for some applications, notably plotting on a 3D printer, a mesh representation is more convenient than an implicit one. Thus, algorithms that compute an (approximate) mesh representation of a surface given by an implicit surface function are essential. The most famous of these is the marching cubes algorithm by [LC87]. But obviously it would be desirable to have an implicit method that directly computes a mesh representation from the sample points while keeping all the advantages of implicit methods.

In this work we present first steps to achieve this goal by working with implicit surface functions whose zero level set is also piecewise linear and can be computed exactly. Unfortunately at the moment our approach is not practical in terms of running time and memory consumption, but it might lead the way to a more efficient approach to combine the advantages of the implicit and the mesh representation. We will present a reasonable upper bound on the theoretical complexity of our approach, which demonstrates that such hopes are not completely unjustified.

This paper is organized as follows: In Section 2 we give a short overview of the problem of computing an implicit surface function from point cloud data. The section contains two optimization problems that define the solutions to the surface reconstruction problem. The optimization problems are parameterized by a kernel function. The flexibility in choosing the kernel function in the end makes it possible to compute a mesh representation from a point cloud by these implicit methods. Section 3 makes up the main part of the paper. It introduces a family of kernel functions that allows to compute a mesh representation from a point cloud. In this section we show how

to compute the faces of the mesh that corresponds to the zero level set of the implicit surface function and derive some theoretical complexity bounds. Finally, in Section 4 we report on some preliminary experiments before we conclude the paper in Section 5 with a discussion of the advantages and disadvantages of our approach.

2 Implicit Surface Reconstruction

In this section we review implicit surface reconstruction and show how it naturally leads to optimization problems. We will discuss two such optimization approaches that are parameterized by a so-called kernel function. Later the right choice of kernel will lead to our goal of providing an implicit approach that leads to a piecewise linear reconstruction.

An implicit surface S in $\mathbb{R}^d$ is defined as the zero-level set of some function $f : \mathbb{R}^d \to \mathbb{R}$:

$$S = \left\{ x \in \mathbb{R}^d \,:\, f(x) = 0 \right\} \quad .$$

Note that the properties of S (even whether it is manifold at all) depend on the function f. In the surface reconstruction problem f has to be defined by a surface sample $\{x_1, \ldots, x_n\} \subset \hat{S} \subset \mathbb{R}^d$ from an unknown surface $\hat{S}$. Of course there are many ways to define f from a sample—most of them do not lead to "good" reconstructions of $\hat{S}$ from the sample. One way to control the quality of reconstruction is to regularize, i.e., to restrict the choice of f. We will consider expansions of kernel functions centered at the training points, i.e., elements of

$$\mathcal{H} = \left\{ f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x) + b \,:\, \alpha_i \in \mathbb{R}, i = 1, \ldots, n \right\},$$

with a symmetric kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ and an offset $b \in \mathbb{R}$. Not any function from $\mathcal{H}$ provides an equally well reconstruction. We want to find a function in $\mathcal{H}$ that, on the one hand, is close to the given points, and on the other hand has so-called regularity properties, such as smoothness. That is, we want to formulation the implicit reconstruction problem as an optimization problem whose objective function has two terms: a term that measures how close a function is to the desired function values in the points is called fidelity term. A term that measures regularity is called regularization term. Note that by our choice of $\mathcal{H}$ the only parameters to optimize are the weights α_i and the offset b. In the following we only want to optimize the α_i and keep the offset b fixed.

Before we state the objective functions for implicit surface reconstruction we need to address one technicality. Clearly, the desired value of the implicit surface function for points on the surface is 0. However, if we only used these points, for both of the considered regularizers an optimal function would be the constant 0 function, which is clearly not the solution we want. Here information on the surface normals in the sample points can be used. The normals often can be estimated from the sample points themselves [AB99]. One approach to utilize normal information is described in [SSB05]: for every point x on the surface one creates off-surface points on the normal line through x in a distance $\pm d$ from x and sets the function value to $\pm d$. This ensures that one does not obtain the trivial solution $f(x) \equiv 0$ as a solution. In the following we assume that we are given a set of pairs $(x_1, y_1), \ldots, (x_n, y_n)$ where $x_1, \ldots, x_n$ are the given surface and off-surface points and $y_1, \ldots, y_n$ are the desired function values in these points We chose $y_i = 0$ if x_i is on the surface and $y_i = \pm d$ if x_i is an off-surface point. This choice of the function values is motivated by the signed distance function corresponding to the surface (i.e., the estimated function should be a local approximation to the signed distance function).

We will now derive two fairly general optimization approaches to pick an implicit reconstruction function from $\mathcal{H}$. Both optimization problems have two terms: a fidelity and a regularization term. The first problem is linear in the optimization variables (the weights α_i), whereas the second problem is a convex quadratic function of these variables.

2.1 Linear Optimization Problem

Here our choice for the fidelity term is

$$d\left(f; (x_1, y_1), \ldots, (x_n, y_n)\right) = \frac{1}{n} \sum_{i=1}^{n} \left| f(x_i) - y_i \right|.$$

Plugging in $f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i) + b$ we get

$$d\left(f; (x_1, y_1), \ldots, (x_n, y_n)\right) = \frac{1}{n} \sum_{i=1}^{n} \left| b - y_i + \sum_{j=1}^{n} \alpha_j k(x_i, x_j) \right|.$$

For the regularization term we choose $\sum_{i=1}^{n} |\alpha_i|$, i.e., the sum of the absolute values of the coefficients α_i. Note that it is well known that this regularizer favors sparse solutions (cf. [SS02] and [BV04]).

Minimizing a weighted sum of the fidelity and the regularization term above, we get the following unconstrained optimization problem:

$$\min_{\alpha_1,\ldots,\alpha_n} \quad \frac{1}{n}\sum_{i=1}^{n}\left|b - y_i + \sum_{j=1}^{n}\alpha_j k(x_i,x_j)\right| + \lambda\sum_{i=1}^{n}|\alpha_i|,$$

where $\lambda > 0$ is a weight parameter that determines the relative importance of fidelity and regularization term. Using standard methods from linear optimization (cf. [BT97]), this optimization problem can easily be transformed to a constrained linear optimization problem with $3n$ variables and $3n$ constraints: first we introduce new variables $t_1,\ldots,t_n$, replace the sum $\sum_{i=1}^{n}\left|b - y_i + \sum_{j=1}^{n}\alpha_j k(x_i,x_j)\right|$ by $\sum_{i=1}^{n} t_i$ and introduce the constraints

$$b - y_i + \sum_{j=1}^{n}\alpha_j k(x_i,x_j) \le t_i$$
$$-\left(b - y_i + \sum_{j=1}^{n}\alpha_j k(x_i,x_j)\right) \le t_i.$$

Thus $t_i \ge \left|b - y_i + \sum_{j=1}^{n}\alpha_j k(x_i,x_j)\right| \ge 0$, and since $\sum_{i=1}^{n} t_i$ is minimized, we have equality in the minimum. Then we introduce $\beta_i,\gamma_i \ge 0$, set $\alpha_i = \beta_i - \gamma_i$ $(i = 1,\ldots,n)$ and replace $\sum_{i=1}^{n}|\alpha_i|$ by $\sum_{i=1}^{n}(\beta_i+\gamma_i)$. Since we minimize $\sum_{i=1}^{n}(\beta_i+\gamma_i)$ we must have either $\beta_i = 0$ or $\gamma_i = 0$ in the minimum (otherwise, if $\beta_i \ge \gamma_i > 0$, we could decrease $\sum_{i=1}^{n}(\beta_i+\gamma_i)$ further by replacing β_i by $\beta_i - \gamma_i$ and setting $\gamma_i = 0$). Thus, in the minimum, $|\alpha_i| = \beta_i + \gamma_i$. These two steps yield the following minimization problem:

$$\min_{\substack{t_1,\ldots,t_n,\\ \beta_1,\ldots,\beta_n,\\ \gamma_1,\ldots,\gamma_n}} \quad \frac{1}{n}\sum_{i=1}^{n} t_i + \lambda\sum_{i=1}^{n}(\beta_i+\gamma_i)$$

subject to (for all $i = 1,\ldots,n$),

$$\begin{array}{l}\sum_{j=1}^{n}(\beta_j-\gamma_j)k(x_i,x_j) + b - y_i \le t_i\\ \sum_{j=1}^{n}(\gamma_j-\beta_j)k(x_i,x_j) - b + y_i \le t_i\\ \beta_i,\gamma_i \ge 0\end{array}$$

Given a solution $t_1^*,\ldots,t_n^*,\beta_1^*,\ldots,\beta_n^*,\gamma_1^*,\ldots,\gamma_n^*$ to this optimization problem, the optimal approximation function is given by

$$f^*(x) = \sum_{i=1}^{n}\alpha_i^* k(x,x_i) + b$$

with $\alpha_i^* = \beta_i^* - \gamma_i^*$ for $i = 1,\ldots,n$.

2.2 Quadratic Optimization Problem

In this approach we choose the fidelity term to be

$$d(f;(x_1,y_1),\ldots,(x_n,y_n)) = \frac{1}{n}\sum_{i=1}^{n}\left(f(x_i) - y_i\right)^2.$$

Using $\sum_i \alpha_i^2$ as a regularizer, we obtain the following (unconstrained) optimization problem:

$$\min_{\alpha_1,\ldots,\alpha_n} \quad \frac{1}{n}\sum_{i=1}^{n}\left(y_i - b - \sum_{j=1}^{n}\alpha_j k(x_i,x_j)\right)^2 + \lambda\sum_{i=1}^{n}\alpha_i^2.$$

We can write the fidelity term

$$\sum_{i=1}^{n}\left(y_i - b - \sum_{j=1}^{n}\alpha_j k(x_i,x_j)\right)^2$$

as

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{n}\alpha_j k(x_i,x_j)+b\right)^2 \quad - \quad 2\sum_{i=1}^{n} y_i\left(\sum_{j=1}^{n}\alpha_j k(x_i,x_j)+b\right) \\ + \quad \sum_{i=1}^{n} y_i^2.$$

With $1_n = (1,\ldots,1)^T$, $y = (y_1,\ldots,y_n)$, $\alpha = (\alpha_1,\ldots,\alpha_n)$ and $K = ((k_{ij}))$, $k_{ij} = k(x_i,x_j)$ we can write the first term in matrix form as

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{n}\alpha_j k(x_i,x_j)\right)^2 + 2b\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_j k(x_i,x_j) + nb^2,$$

which is the same as

$$\alpha^T K^T K\alpha + 2b1_n^T K\alpha + nb^2.$$

We can write the second term as

$$2\sum_{i=1}^{n} y_i\left(\sum_{j=1}^{n}\alpha_j k(x_i,x_j)+b\right) = 2y^T K\alpha + 2y^T b$$

Ignoring all the constant terms, we have the following minimization problem

$$\min_{\alpha} \quad \frac{1}{n}\left(\alpha^T K^T K\alpha + 2b1_n^T K\alpha + 2y^T K\alpha\right) + \lambda\alpha^T\alpha,$$

and thus

$$\min_{\alpha} \quad \alpha^T\left(\frac{1}{n}K^T K + \lambda\mathbb{I}_n\right)\alpha + \frac{2}{n}\left(b1_n^T K + y^T K\right)\alpha.$$

Since K is symmetric, $K^T K = K^2$ is symmetric and positive semi-definite, and thus $\frac{1}{n}K^T K + \lambda\mathbb{I}_n$ is positive definite. That means, we can find a solution to this problem analytically in analogy to the derivation of least squares methods. This yields the following solution:

$$\alpha = \left(\frac{1}{n}K^T K + \lambda\mathbb{I}_n\right)^{-1} K\left(b1_n + y\right).$$

3 A Kernel Function for Piecewise Linear Reconstruction

As we mentioned earlier the properties of the implicit surface S depend on the function f that we use in its definition. For the function class $\mathcal{H}$ this boils down to that many properties of S depend on the choice of kernel function k. Here we study a kernel function that can be used with both optimization approaches that we discussed before and allows for piecewise linear reconstruction. The kernel function is given as

$$k(x,y) = \max\left\{0, h - \frac{h}{r}\parallel x-y\parallel_\infty\right\}$$

where $h \in \mathbb{R}$ and $r \in \mathbb{R}$ are parameters. Clearly, this kernel function is continuous, because norms are continuous and the maximum of two continuous functions is continuous. However, it is not positive definite. The support of the kernel functions is an (axis-aligned) cube with side length $2r$. In two dimensions the graph of $k(.,y)$ is a pyramid centered at y with height h. An example can be seen in Figure 1. In the following we will often call h the height parameter of the kernel and r the width parameter of the kernel.

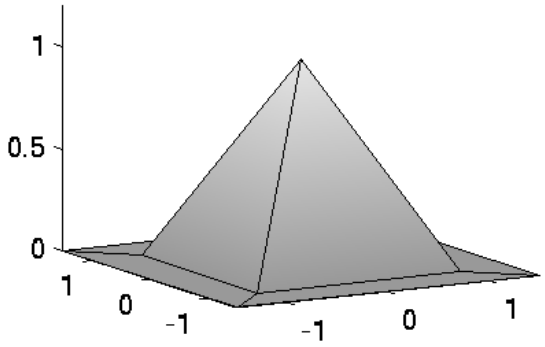

Figure 1: The kernel function for piecewise linear reconstruction in two dimensions, centered at $(0,0)$.

3.1 Decomposition of a single Kernel Function into Linear Functions

In this section we will show how to decompose $\mathbb{R}^d$ into regions, such that a single kernel function $k(.,y)$ is linear in each of these regions. For a kernel function in two dimensions these regions are shown in Figure 2. Let $y = \left(y^{(1)},\ldots,y^{(d)}\right) \in \mathbb{R}^d$ be a point and $r > 0$. For $i = 1,\ldots,d$ denote $x = \left(x^{(1)},\ldots,x^{(d)}\right)$ and

$$Q_{y,i} = \left\{ x \in \mathbb{R}^d \,:\, r > x^{(i)} - y^{(i)} > \max_{j \neq i} \left| x^{(j)} - y^{(j)} \right| \right\}$$
$$Q_{y,-i} = \left\{ x \in \mathbb{R}^d \,:\, r > y^{(i)} - x^{(i)} > \max_{j \neq i} \left| x^{(j)} - y^{(j)} \right| \right\}$$

and set

$$Q_{y,0} = \left\{ x \in \mathbb{R}^d \,:\, \max_{1 \leq j \leq d} \left| x^{(j)} - y^{(j)} \right| > r \right\}.$$

That means, for $i = 1,\ldots,d$ a point $x = \left(x^{(1)},\ldots,x^{(d)}\right)$ is in $Q_{y,i}$, if $||x-y||_\infty < r$, $\left|x^{(i)} - y^{(i)}\right| = ||x-y||_\infty$ and $x^{(i)} \geq y^{(i)}$, and it is in $Q_{y,-i}$, if the last condition is replaced by $y^{(i)} \geq x^{(i)}$. On the other hand, $x \in Q_{y,0}$ if $||x-y||_\infty > r$.

The kernel function $k(.,y)$ centered at y can be expressed as a linear function on these regions:

$$k(x,y) = \begin{cases} 0, & x \in Q_{y,0} \\ h - \frac{h}{r}\left(x^{(i)} - y^{(i)}\right), & x \in Q_{y,i} \text{ for } i \in \{1,\ldots,d\} \\ h - \frac{h}{r}\left(y^{(i)} - x^{(i)}\right), & x \in Q_{y,-i} \text{ for } i \in \{1,\ldots,d\} \end{cases}$$

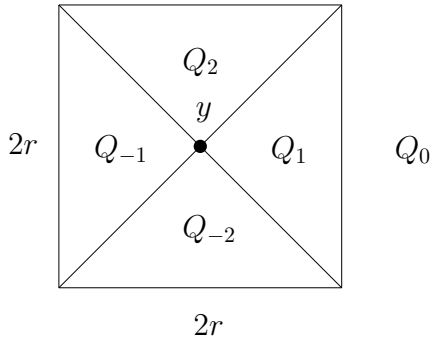

Figure 2: Regions in $\mathbb{R}^2$ in which the kernel function $k(.,y)$ is linear.

Clearly the regions $Q_{y,i}$ $(i = -d, \ldots, d)$ are disjoint, and

$$\mathbb{R}^d = \overline{\bigcup_{i=-d}^{d} Q_{y,i}}.$$

Thus, it is enough to give the kernel function in the regions $Q_{y,i}$ $(i = -d, \ldots, d)$, because we can extend it to $\mathbb{R}^d$ by continuity of $k(.,y)$.

Remark 1 *We have seen, that we can decompose $\mathbb{R}^d$ into $2d+1$ regions, such that a kernel function $k(.,y)$ is linear on these regions. However, if we require the regions to be convex, we have to decompose the region $Q_{y,0}$ further. The easiest possibility would be to introduce regions $Q'_{y,-1}, Q'_{y,1}, \ldots, Q'_{y,-d}, Q'_{y,d}$, where*

$$Q'_{y,i} = \left\{ x \in \mathbb{R}^d : x^{(i)} - y^{(i)} > r, x^{(i)} - y^{(i)} > \max_{j \neq i} \left| x^{(j)} - y^{(j)} \right| \right\}$$
$$Q'_{y,-i} = \left\{ x \in \mathbb{R}^d : y^{(i)} - x^{(i)} > r, y^{(i)} - x^{(i)} > \max_{j \neq i} \left| x^{(j)} - y^{(j)} \right| \right\}.$$

In our work we did not require the regions to be convex, and thus will not give further details here. It is straightforward to transfer the methods we used to this case.

3.2 Decomposition of the Implicit Surface Function

Now we do not consider a single kernel function anymore, but a set of kernel functions centered at sample points $\{y_1, \ldots, y_n\} \subseteq \mathbb{R}^d$. We show that $\mathbb{R}^d$ can be decomposed into regions such that the implicit surface functions that we consider are linear in each region. Remember that the implicit surface functions we consider are of the form

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x, y_i) + b.$$

In the last section we have seen that for every point y_i we can find regions $Q_{y_i,-d}, \ldots, Q_{y_i,d}$, such that $k(.,y_i)$ is linear in these regions and the closure of the union of these regions is $\mathbb{R}^d$. Thus

$$\mathbb{R}^d = \overline{\bigcup_{(j_1,\ldots,j_n) \in \{-d,\ldots,d\}^n} \bigcap_{i=1}^{n} Q_{y_i,j_i}}$$

and for all $(j_1, \ldots, j_n) \in \{-d, \ldots, d\}^n$, the implicit surface function f is linear on the region

$$Q_{(j_1,\ldots,j_n)} := \bigcap_{i=1}^{n} Q_{y_i,j_i}.$$

Note that the form of these regions does not depend on the weight parameters $\alpha_1, \ldots, \alpha_n$.

3.3 Representing the regions by Nef polyhedra

From a computational point of view it is interesting to observe that the regions from the last section can be represented by Nef polyhedra, which are point sets generated from a finite number of open half spaces by set complement and set intersection operations (cf. [HK05]). For a kernel function $k(.,y)$ based at a point y we have

$$Q_{y,0} = \bigcup_{j=1}^{d} \left\{ x : x^{(j)} - y^{(j)} > r \right\} \cup \bigcup_{j=1}^{d} \left\{ x : y^{(j)} - x^{(j)} > r \right\}$$

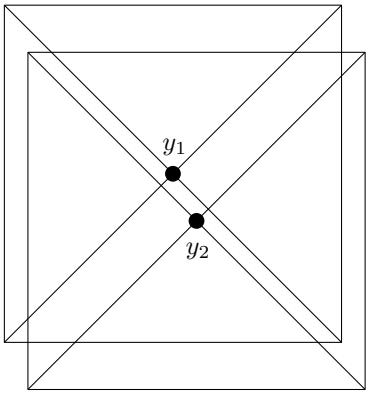

Figure 3: Intersection regions of two points in two dimensions.

and for $i = 1, \ldots, d$

$$
\begin{aligned}
Q_{y,-i} = & \\
& \bigcap_{j \neq i} \left\{ x, : y^{(i)} - x^{(i)} > x^{(j)} - y^{(j)} \right\} \cap \\
& \bigcap_{j \neq i} \left\{ x : y^{(i)} - x^{(i)} > y^{(j)} - x^{(j)} \right\} \cap \left\{ x : y^{(i)} - x^{(i)} < r \right\} \\
Q_{y,i} = & \\
& \bigcap_{j \neq i} \left\{ x : x^{(i)} - y^{(i)} > x^{(j)} - y^{(j)} \right\} \cap \\
& \bigcap_{j \neq i} \left\{ x : x^{(i)} - y^{(i)} > y^{(j)} - x^{(j)} \right\} \cap \left\{ x : x^{(i)} - y^{(i)} < r \right\}.
\end{aligned}
$$

Thus, all the regions $Q_{y,-d}, \ldots, Q_{y,d}$, where the kernel function $k(.,y)$ is linear, can be represented as Nef polyhedra. Using the computational geometry algorithms library CGAL, one can readily compute unions and intersections of Nef polyhedra in two and three dimensions (cf. [HK05]).

3.4 Computing all Intersection Regions

So far we discussed how to define and represent (by Nef polyhedra) the subdivision of $\mathbb{R}^d$ into regions on which the implicit surface function is linear—the regions $Q_{(j_1,\ldots,j_n)}$ for $(j_1, \ldots, j_n) \in \{-d, \ldots, d\}^n$ as defined above. Note that many of the intersections will be empty, see for example in Figure 3 the intersection regions for a special configuration of two points in two dimensions. In this section we give an algorithm to actually compute the non-empty intersection regions.

The following algorithm computes all (nonempty) intersection regions:

- Given:
  - points $x_1, \ldots, x_n$
  - kernel widths $r_1, \ldots, r_n$
- Keep a list L of intersection regions.
- Initialize: $L = Q_{x_1,-d}, \ldots, Q_{x_1,d}$
- For all $j = 2, \ldots, n$
  - Compute $Q_{x_j,-d}, \ldots, Q_{x_j,d}$
  - For all $Q \in L$
    - Remove Q from L

* Compute intersection of Q with $Q_{x_j,-d},\ldots,Q_{x_j,d}$.
* For all $l \in \{-d,\ldots,d\}$ for which $Q \cap Q_{x_j,l} \neq \emptyset$, add $Q \cap Q_{x_j,l}$ to L.

- Output: L

Note that the regions do not depend on the kernel heights and the weights $\alpha_1,\ldots,\alpha_n$ and that usually not all of the regions will contain a part of the implicit surface. How the implicit surface can be computed exactly using the list of nonempty intersection regions is the topic of the next section.

3.5 Computing the Zero Level Set within a Region

Once we have computed the non-empty intersection regions (what we did in the last section), we basically already have computed the whole piecewise linear reconstruction since the implicit reconstruction function is linear on each of these regions.

Within a region $Q_{(j_1,\ldots,j_n)}$, we have

$$\begin{aligned}
f|_{Q_{(j_1,\ldots,j_n)}}(x) &= b + \sum_{\{i:j_i<0\}} \alpha_i \left(h - \frac{h}{r}\left(y_i^{(-j_i)} - x^{(-j_i)} \right) \right) + \\
&\quad \sum_{\{i:j_i>0\}} \alpha_i \left(h - \frac{h}{r}\left(x^{(j_i)} - y_i^{(j_i)} \right) \right) \\
&= b + \sum_{\{i:j_i<0\}} \alpha_i \left(h - \frac{h}{r} e_{-j_i}^T (y_i - x) \right) + \\
&\quad \sum_{\{i:j_i>0\}} \alpha_i \left(h - \frac{h}{r} e_{j_i}^T (x - y_i) \right) \\
&= b + \sum_{\{i:j_i\neq 0\}} \alpha_i h + \alpha_i \frac{h}{r} \operatorname{sgn}(j_i)\, e_{|j_i|}^T (y_i - x) \\
&= b + \sum_{\{i:j_i\neq 0\}} \alpha_i \left(h + \frac{h}{r} \operatorname{sgn}(j_i)\, e_{|j_i|}^T y_i \right) - \\
&\quad \left(\sum_{\{i:j_i\neq 0\}} \alpha_i \frac{h}{r} \operatorname{sgn}(j_i)\, e_{|j_i|} \right)^T x,
\end{aligned}$$

where e_l denotes the l-th unit vector and sgn the sign function. Obviously $f|_{Q_{(j_1,\ldots,j_n)}}$ is a linear function and thus $\{f|_{Q_{(j_1,\ldots,j_n)}} = 0\}$ defines a hyperplane. Computing the intersection of the hyperplane $\{f|_{Q_{(j_1,\ldots,j_n)}} = 0\}$ with the region $Q_{(j_1,\ldots,j_n)}$, we can find the intersection of the zero level set of f with the region $Q_{(j_1,\ldots,j_n)}$, and thus the part of the surface that is within the region $Q_{(j_1,\ldots,j_n)}$. Since a hyperplane can easily be represented by the intersection of two closed half spaces and the region can be represented by a Nef polyhedron, the zero level set can be represented as a Nef polyhedron as well. An obvious algorithm to compute the (piecewise linear) zero level set of the implicit surface function is to go through the list of all nonempty regions and compute the patch of the surface that is within the region.

3.6 An Upper Bound on the Number of Intersection Regions

The complexity of our implicit reconstruction scheme is essentially determined by the number of (non-empty) intersection regions (on which the implicit surface function is linear) and the complexity of the optimization scheme to obtain the weights α_i. Here we derive an upper bound for the number of intersection regions, which essentially provides an upper bound for the complexity of the whole reconstruction scheme.

For a point y let $\mathcal{A}_y$ be the set containing the following d^2+d hyperplanes:

$$\begin{aligned}
\mathcal{A}_y &= \Big\{ x^{(i)} = y^{(i)} + sr, x^{(i)} + sx^{(j)} = y^{(i)} + sy^{(j)} : \\
&\qquad i = 1,\ldots,n; j = i+1,\ldots,n; s = \pm 1 \Big\}.
\end{aligned}$$

Clearly the number of intersection regions for a set of points $x_1,\ldots,x_n$ can be bounded from above by the number of regions of the hyperplane arrangement

$$\mathcal{A} = \mathcal{A}_{x_1} \cup \ldots \cup \mathcal{A}_{x_n}.$$

There are $(d+1)\,dn$ hyperplanes in the arrangement and thus the number of regions can be bounded by the number of regions in a hyperplane arrangement of the same number of hyperplanes in general position, which is

$$1+(d+1)\,dn+\binom{(d+1)\,dn}{2}+\ldots+\binom{(d+1)\,dn}{d}$$

and therefore in $O(n^d)$ (cf. [Sta06]). Note that in three dimensions that bounds the complexity of our reconstruction scheme essentially by $O(n^3)$, which is reasonable from a theoretical point of view (polynomial time algorithm), but unfortunately not sufficient in practice. Note though that this bound is too pessimistic because, due to the compact support of the kernel functions, a point will only influence the regions of neighboring points.

4 Preliminary Experiments

We performed some preliminary experiments that unfortunately show that our method is not practical for large datasets at the moment. While we were able to solve the optimization problems for computing the kernel expansions for moderately sized datasets of several thousand points, the exact mesh computation became impractical already for a few hundred points.

For the solution of the linear optimization problem we used the *CPLEX* solver, whereas the closed form solution of the quadratic optimization problem can be computed with Matlab. Better (visual) results were obtained using the quadratic optimization problem. In both cases, results were dependent on the choice of the parameters b, λ, kernel height and kernel width, as well as on the distance d in which the off-surface points were constructed.

In order to compute the meshes according to the algorithm above we used the computational geometry algorithms library CGAL (cf. [CGA06]) together with the GMP multi-precision library.

5 Discussion

We used piecewise linear implicit surface functions in order to approximate a surface that is given by a set of points and the surface normals in these points (which in most cases can be estimated well just from the sample points). The (piecewise linear) zero level set of this implicit surface function can then be computed exactly, thus giving an (exact) mesh representation. That means that in contrast to the common method of applying the marching cubes algorithm to an arbitrary implicit surface function, the only approximate step is the estimation of the implicit surface function (if desired, and if the data are noise-free, this step can be made very precise by decreasing the regularization strength). We found that it is in principle possible to apply this method to compute a mesh representation from the sample points and surface normals. However, there are a number of disadvantages to this method that we do not want to sweep under the rug:

The kernel functions (and thus the implicit surface function) are not differentiable. Therefore it is not clear how smoothness of the implicit function can be measured and used in regularization. However, it might be possible to use the knowledge of the surface normals.

Since we cannot control the size and the shape of the intersection regions, we cannot give any guarantees on the quality of the resulting mesh representation, i.e., the mesh may be quite irregular.

But the greatest drawback of our method are the time and memory requirements for computing the intersection regions. The high time and memory requirements are due to the excessive number of intersection regions that have to be considered and the fact that it becomes increasingly difficult to describe the intersection regions when more kernel functions have to be intersected. One possibility to accelerate these computations at the cost of exactness would be to use fixed precision arithmetic instead of arbitrary precision arithmetic. Since only a fraction of the intersection regions actually contain a part of the surface, another possibility would be to implement a surface following procedure.

Nevertheless, these are first steps in a new direction: combining the power (and different advantages) of implicit and explicit (mesh) reconstruction approaches. Our work demonstrates that in principle it is possible to combine these two paradigms. Furthermore, it does not look hopeless that out of the ideas that we have presented here a practical reconstruction scheme can arise.

References

[AB99] AMENTA N., BERN M. W.: Surface reconstruction by voronoi filtering. *Discrete & Computational Geometry 22*, 4 (1999), 481–504.

[BT97] BERTSIMAS D., TSITSIKLIS J. N.: *Introduction to Linear Optimization*. Athena Scientific, 1997.

[BV04] BOYD S., VANDENBERGHE L.: *Convex Optimization*. Cambridge University Press, 2004.

[CBC*01] CARR J. C., BEATSON R. K., CHERRIE J. B., MITCHELL T. J., FRIGHT W. R., MCCALLUM B. C., EVANS T. R.: Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM Press, pp. 67–76.

[CGA06] CGAL EDITORIAL BOARD: *CGAL-3.2 User and Reference Manual*, 2006.

[HK05] HACHENBERGER P., KETTNER L.: Boolean operations on 3d selective nef complexes: optimized implementation and experiments. In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling* (New York, NY, USA, 2005), ACM Press, pp. 163–174.

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), ACM Press, pp. 163–169.

[MBWB02] MUSETH K., BREEN D. E., WHITAKER R. T., BARR A. H.: Level set surface editing operators. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM Press, pp. 330–338.

[MT93] MCINERNEY T., TERZOPOULOS D.: A finite element model for 3d shape reconstruction and nonrigid motion tracking. In *IEEE 4th Int. Conf. on Comp. Vision* (1993), pp. 518–523.

[SS02] SCHÖLKOPF B., SMOLA A. J.: *Learning with Kernels*. The MIT Press, Cambridge, MA, 2002.

[SSB05] STEINKE F., SCHÖLKOPF B., BLANZ V.: Support vector machines for 3d shape processing. *Computer Graphics Forum 24*, 3 (2005), 285–294.

[Sta06] STANLEY R. P.: An introduction to hyperplane arrangements. Lecture Notes, 2006.

[WSC06] WALDER C., SCHÖLKOPF B., CHAPELLE O.: Implicit surface modelling with a globally regularised basis of compact support. *Computer Graphics Forum 25*, 3 (2006), 635–644.

[ZO02] ZHAO H., OSHER S.: Visualization, analysis and shape reconstruction of unorganized data sets. In *Geometric Level Set Methods in Imaging, Vision and Graphics*, Osher S., Paragios N., (Eds.). Springer-Verlag, 2002.

[ZOF01] ZHAO H.-K., OSHER S., FEDKIW R.: Fast surface reconstruction using the level set method. In *VLSM '01: Proceedings of the IEEE Workshop on Variational and Level Set Methods (VLSM'01)* (Washington, DC, USA, 2001), IEEE Computer Society, p. 194.

[ZOMK00] ZHAO H.-K., OSHER S., MERRIMAN B., KANG M.: Implicit and nonparametric shape reconstruction from unorganized data using a variational level set method. *Comput. Vis. Image Underst. 80*, 3 (2000), 295–314.