EUROGRAPHICS 2006 / E. Gröller and L. Szirmay-Kalos
(Guest Editors)

Volume 25 (2006), Number 3

Implicit Surface Modelling with a Globally Regularised Basis of Compact Support

C. Walder[1,2], B. Schölkopf[1] and O. Chapelle[1]

[1] Max-Planck-Institut für biologische Kybernetik, Tübingen, Germany
[2] The University of Queensland, Brisbane, Queensland 4072, Australia.

Abstract

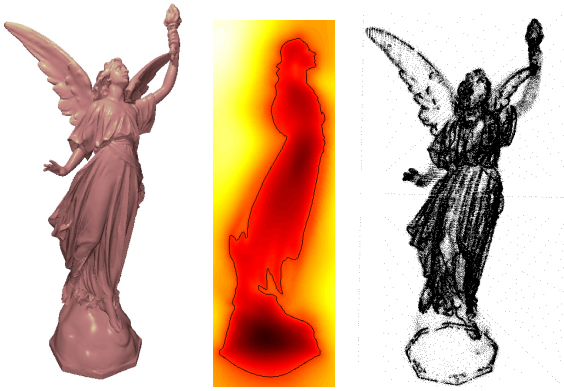
We consider the problem of constructing a globally smooth analytic function that represents a surface implicitly by way of its zero set, given sample points with surface normal vectors.
The contributions of the paper include a novel means of regularising multi-scale compactly supported basis functions that leads to the desirable interpolation properties previously only associated with fully supported bases. We also provide a regularisation framework for simpler and more direct treatment of surface normals, along with a corresponding generalisation of the representer theorem lying at the core of kernel-based machine learning methods.
We demonstrate the techniques on 3D problems of up to 14 million data points, as well as 4D time series data and four-dimensional interpolation between three-dimensional shapes.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Curve, surface, solid, and object representations

1. Introduction

The problem of reconstructing a surface from a set of points frequently arises in computer graphics. Numerous methods of sampling physical surfaces are now available, including laser scanners, optical triangulation systems and mechanical probing methods. Inferring a surface from millions of points sampled with noise is a non-trivial task however, for which a variety of methods have been proposed.

The class of *implicit* or *level set* surface representations is a rather large one, however other methods have also been suggested. These other methods include Bernstein-Bézier representations within a Delaunay triangulated tessellation [BBX95], as well as point based representations that analyse the point cloud directly and locally as needed at rendering time [ABCO*01]. Within the realm of implicit surface methods there also exist distinct classes such as those defined on a numerical grid [Set98, WB98], "blobby" methods which implement a form of structural regularisation [Mur91], tangent plane projection approaches [HDD*92], as well as the more recent partition of unity [OBA*03] and moving least squares [SOS04] methods.



(a) (b) (c)

Figure 1: *(a) Rendered implicit surface model of "Lucy", constructed from 14 million points with normals. (b) A planar slice that cuts the nose – the colour represents the value of the embedding function and the black line its zero level. (c) A black dot at each of the 364,982 compactly supported basis function centres which, along with the corresponding dilations and magnitudes, define the implicit.*

The implicit surface methods closest to the present work are those that construct the implicit using regularised function approximation [Wah90], such as the "Variational Implicits" of Turk and O'Brien [TO99], which produce excellent results, but at a cubic computational fitting cost in the number of points. The effectiveness of this type of approach is undisputed however, and has led researchers to look for ways to overcome the computational problems. Presently, two main options exist.

The first approach uses compactly supported kernel functions (we define and discuss kernel functions in Section 2), leading to fast algorithms that are easy to implement [MYC*01]. Unfortunately however these methods are suitable for benign data sets only. As noted in [CBC*01], compactly supported basis functions "yield surfaces with many undesirable artifacts in addition to the lack of extrapolation across holes". A similar conclusion was reached in [OBA*03] which states that local processing methods are "more sensitive to the quality of input data [than] approximation and interpolation techniques based on globally-supported radial basis functions" – a conclusion corroborated by the results within a different paper from the same group [OBS03]. The second means of overcoming the aforementioned computational problem does not suffer from these problems however, as demonstrated by the FastRBF^{TM} algorithm [CBC*01], which uses the the Fast Multipole Method (FMM) [GR97] to overcome the computational problems of non-compactly supported kernels. The resulting method is non-trivial to implement however and to date exists only in the proprietary FastRBF^{TM} package.

We believe that by applying them in a different manner compactly supported basis functions can lead to high quality results, and the present work is an attempt to bring the reader to the same conclusion. In Section 3 we introduce a new technique for regularising such basis functions which allows high quality, highly scalable algorithms that are relatively easy to implement. Before doing so however, we present in Section 2 the other main contribution of the present work, which is to show how surface normal vectors can be incorporated directly into the regularised regression framework that is typically used for fitting implicit surfaces, thereby avoiding the problematic approach of constructing "off-surface" points for the regression problem. To demonstrate the effectiveness of the method we apply it to various two to four-dimensional problems in Section 4 before summarising in the final Section 5.

2. Implicit Surface Fitting by Regularised Regression

Here we discuss the use of regularised regression [Wah90] for the problem of implicit surface fitting. In Section 2.1 we motivate and introduce a clean and direct means of making use of normal vectors. The following Section 2.2 extends on the ideas in Section 2.1 by formally generalising the important *representer theorem*. The final Section 2.3 discusses the choice of regulariser (and associated kernel function), as well as the associated computational problems that we overcome in Section 3.

2.1. Regression Based Approaches and the Use of Normal Vectors

Of the implicit methods that use regularised interpolation [Wah90], the usual approach is to solve the following problem [CBC*01, MYC*01]:

$$\arg\min_f \|f\|_{\mathcal{H}}^2 + C\sum_{i=1}^{m}\left(f(\mathbf{x}_i) - y_i\right)^2. \tag{1}$$

Here the y_i are some estimate of the signed distance function at the $\mathbf{x}_i$, and f is the embedding function which takes on the value zero on the implicit surface. The norm $\|f\|_{\mathcal{H}}$ is a *regulariser* which takes on larger values for less "smooth" functions. We take $\mathcal{H}$ to be a reproducing kernel Hilbert space (RKHS) with representer of evaluation (kernel function) $k(\cdot,\cdot)$, so that we have the *reproducing* property [Wah90]:

$$f(\mathbf{x}) = \langle f, k(\mathbf{x},\cdot)\rangle_{\mathcal{H}}. \tag{2}$$

The solution to this problem has the form

$$f(\mathbf{x}) = \sum_{i}^{m}\alpha_i k(\mathbf{x}_i,\mathbf{x}). \tag{3}$$

Note as a technical aside that the thin-plate kernel case – which we will adopt – does not have an associated RKHS as it is only *conditionally* positive definite. Here we consider the RKHS case for clarity only, as it is simpler and yet sufficient to demonstrate the ideas involved.

Choosing the $(\mathbf{x}_i, y_i)$ pairs for Equation 3 is in itself a non-trivial problem. Given unit length surface normal vectors $\mathbf{n}_i$ at surface points $\mathbf{x}_i$, FastRBF^{TM} [CBC*01] constructs a regression problem with target-value pairs:

$$(\mathbf{x}_i, 0), (\mathbf{x}_i + d\mathbf{n}_i, y_{i+}), (\mathbf{x}_i - d\mathbf{n}_i, -y_{i-}),$$

for some heuristically chosen scalar d. The targets y_{i+} and y_{i-} are taken to be the distance to the nearest data point. These manufactured "off-surface" points may contradict one another however, and further heuristics are employed in FastRBF^{TM} to detect and simply discard such problematic cases. We now propose a more direct way of using the normal vectors, novel in the context of implicit fitting with kernel function interpolation, which avoids these problems. The approach is suggested by the fact that the normal direction of the implicit surface is given by the gradient of the embedding function – thus normal vectors can be incorporated by regression with gradient targets. The function that we seek is the minimiser of:

$$\|f\|_{\mathcal{H}}^2 + C_1\sum_{i=1}^{m}\left(f(\mathbf{x}_i)\right)^2 + C_2\sum_{i=1}^{m}\|(\nabla f)(\mathbf{x}_i) - \mathbf{n}_i\|_{\mathbb{R}^d}^2, \tag{4}$$

which uses the given surface point/normal pairs $(\mathbf{x}_i, \mathbf{n}_i)$ directly. By imposing stationarity and using the reproducing property (Equation 2) we can solve for the optimal f. A detailed derivation of this procedure is given in [WSC06]. Here we provide only the result, which is that we have to solve for m coefficients α_i as well as a further md coefficients β_{lj} to obtain the optimal solution

$$f(\mathbf{x}) = \sum_i^m \alpha_i k(\mathbf{x}_i, \mathbf{x}) + \sum_i^m \sum_l^d \beta_{li} k_l(\mathbf{x}_i, \mathbf{x}), \tag{5}$$

where we have defined $k_l(\mathbf{x}_i, \mathbf{x}) \doteq [(\nabla k)(\mathbf{x}_i, \mathbf{x})]_l$, the partial derivative of k (in its first argument) in the l-th component. Note that for the remainder of the paper square brackets with subscripts indicate single elements a matrix or vector so that $[\mathbf{a}]_i$ is the i-th element of the vector $\mathbf{a}$. The coefficients α and $\beta_\mathbf{l}$ of the solution are found by solving

$$\mathbf{0} = (K + I/C_1)\alpha + \sum_l K_l \beta_l \tag{6}$$

along with, for $m = 1 \ldots d$

$$N_m = K_m \alpha + (K_{mm} + I/C_2)\beta_\mathbf{k} + \sum_{l \neq m} K_{lm} \beta_\mathbf{l}, \tag{7}$$

where we've defined the following matrices and vectors:

$$\begin{array}{cc} [N_l]_i = [\mathbf{n}_i]_l; & [\alpha]_i = \alpha_i \\ [\beta_l]_i = \beta_{li}; & [K]_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j) \\ [K_l]_{i,j} = k_l(\mathbf{x}_i, \mathbf{x}_j)\,; & [K_{lm}]_{i,j} = k_{lm}(\mathbf{x}_i, \mathbf{x}_j). \end{array}$$

Note that the k_{lm} are the second derivatives of $k(\cdot,\cdot)$ (defined similarly to the first).

In summary, minimum norm approximation in an RKHS with gradient target values is optimally solved by a function in the span of the kernels and derivatives thereof as per Equation 5 (*cf.* Equation 3), and the coefficients of the solution are given by Equations (6) and (7). It turns out, however, that we can make a more general statement, which we do briefly in the next sub-Section.

2.2. The Representer Theorem with Linear Operators

The representer theorem, much celebrated in the Machine Learning community, says that the function minimising an RKHS norm along with some penalties associated with the function value at various points (as in Equation 1 for example) is a sum of kernel functions at those points (as in Equation 3). As we saw in the previous section however, if gradients also appear in the risk function to be minimised, then gradients of the kernel function appear in the optimal solution. We now make a more general statement – the case in the previous section corresponds to the following if we choose the linear operators L_i (which we define shortly) as either identities or partial derivatives. The theorem is a generalisation of [SHS01] (using the same proof idea) with equivalence if we choose all L_i to be identity operators. The case of general linear operators was in fact dealt with already in [Wah90] (which merely states the earlier result in [KW71]) – but only for the case of a specific loss function c. The following theorem therefore combines the two frameworks:

Theorem 1 Denote by $\mathcal{X}$ a non-empty set, by k a reproducing kernel with reproducing kernel Hilbert space $\mathcal{H}$, by $(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_m, y_m) \subset \mathcal{X} \times \mathbb{R}$ a training set, by Ω a strictly monotonic increasing real-valued function on $[0, \infty)$, by $c : (\mathcal{X} \times \mathbb{R}^2)^m \to \mathbb{R} \cup \{\infty\}$ an arbitrary cost function, and by $L_1, \ldots L_m$ a set of linear operators $\mathcal{H} \to \mathcal{H}$. Each minimiser $f \in \mathcal{H}$ of the regularised risk functional

$$c((\mathbf{x}_1, y_1, (L_1 f)(\mathbf{x}_1), \ldots (\mathbf{x}_m, y_m, (L_m f)(\mathbf{x}_m)) + \Omega(||f||^2_\mathcal{H}) \tag{8}$$

admits the form

$$f(\cdot) = \sum_{i=1}^m \alpha_i (L_i^* k)(\cdot, \mathbf{x}_i),$$

where L_i^* denotes the adjoint of L_i.

Proof Decompose $f(\cdot)$ into

$$f(\cdot) = \sum_{i=1}^m \alpha_i (L_i^* k)(\mathbf{x}_i, \cdot) + f_\perp(\cdot)$$

with $\alpha_i \in \mathbb{R}$ and $\langle f_\perp(\cdot), (L_i^* k)(\mathbf{x}_i, \cdot)\rangle_\mathcal{H} = 0$, for each $i = 1 \ldots m$. Due to the reproducing property we can write, for $j = 1 \ldots m$,

$$\begin{aligned} (L_j f)(\mathbf{x}_j) &= \langle (L_j f)(\cdot), k(\mathbf{x}_j, \cdot)\rangle_\mathcal{H} \\ &= \sum_{i=1}^m \alpha_i \langle (L_j L_i^* k)(\mathbf{x}_i, \cdot), k(\mathbf{x}_j, \cdot)\rangle_\mathcal{H} \\ &\quad + \langle (L_j f_\perp)(\cdot), k(\mathbf{x}_j, \cdot)\rangle_\mathcal{H} \\ &= \sum_{i=1}^m \alpha_i \langle (L_j L_i^* k)(\mathbf{x}_i, \cdot), k(\mathbf{x}_j, \cdot)\rangle_\mathcal{H}. \end{aligned}$$

Thus, the first term in Equation 8 is independent of $f_\perp$. Moreover, it is clear that due to orthogonality

$$\Omega\left(\left\|\sum_{i=1}^m \alpha_i (L_i^* k)(\mathbf{x}_i, \cdot) + f_\perp\right\|^2_\mathcal{H}\right) \geq \Omega\left(\left\|\sum_{i=1}^m \alpha_i (L_i^* k)(\mathbf{x}_i, \cdot)\right\|^2_\mathcal{H}\right),$$

so that for any fixed $\alpha_i \in \mathbb{R}$, Equation 8 is minimised when $f_\perp = 0$. □

2.3. Thin Plate Regulariser and Associated Kernel

We have not yet discussed the choice of kernel function. As is well known (see for example [Wah90]), the kernel function is a Green's function related to the regularisation operator that defines the function norm in the RKHS – practically speaking, the choice of regulariser (the function norm in Equation 4) leads to a particular kernel function $k(\cdot,\cdot)$ to be used in Equation 5. Here we mean regulariser in the usual sense of an operator that penalises some measure of the complexity of a function, leading to well behaved or smooth solutions. For geometrical problems, an excellent regulariser

is the *thin-plate* energy, which for arbitrary order m and dimension d is given by [Wah90]:

$$\|f\|^2_{\mathcal{H}} = \langle \psi f, \psi f \rangle_{\mathcal{L}_2} \tag{9}$$

$$= \sum_{i_1=1}^{d} \cdots \sum_{i_m=1}^{d} \int_{x_1=-\infty}^{\infty} \cdots \int_{x_d=-\infty}^{\infty} \cdot \left(\frac{\partial}{\partial x_{i_1}} \cdots \frac{\partial}{\partial x_{i_m}} f \right) \left(\frac{\partial}{\partial x_{i_1}} \cdots \frac{\partial}{\partial x_{i_m}} f \right) \mathrm{d}x_1 \ldots \mathrm{d}x_d, \tag{10}$$

where ψ is a regularisation operator that takes all partial derivatives of order m.

Experience has shown that in two and three dimensions order $m = 2$ works well, whereas in four dimensions $m = 3$ is appropriate. The above regulariser corresponds to a "radial" kernel function of the form

$$k(\mathbf{x}, \mathbf{y}) = t(||\mathbf{x} - \mathbf{y}||),$$

where [Duc77]

$$t(r) = \begin{cases} r^{2m-d} \ln(r) & \text{if } 2m > d \text{ and } d \text{ is even,} \\ r^{2m-d} & \text{otherwise.} \end{cases}$$

There are a number of good reasons to use this regulariser rather than those leading to compactly supported kernels, as we touched on in the introduction. The main problem with compactly supported kernels is that the corresponding regularisers are somewhat poor for geometrical problems – they always draw the function towards some nominal constant as one moves away from the data, thereby implementing the non-intuitive behaviour of regularising the constant function and making interpolation impossible.

The problem with compactly supported bases can also be seen from the fact that the support of the basis function must be comparable to the size of the smallest details one wishes to capture from the data, making interpolation impossible on scales larger than the finest details of the shape. To overcome this one may try fitting the function with basis functions of large support (to interpolate unsampled regions) followed by fitting to the residual errors with basis functions of diminishing support (to capture smaller details), leading to a final function that is the sum of the various intermediate functions, say $f = \sum_{i=1}^{n} f_i$. Since regularisation is done on each scale separately however, the overall regulariser is, roughly speaking, something like the expression $\sum_{i=1}^{n} ||f_i||^2_{\mathcal{H}_i}$. Clearly this is not a sensible regulariser – one simple example demonstrating this being the case $f_1 + f_2 = 0$, since even this arguably simplest of functions corresponds to a "regularisation" term $||f_1||^2_{\mathcal{H}_1} + ||f_2||^2_{\mathcal{H}_2}$ that could be arbitrarily large.

Our experiments with such an approach led to spurious components of zero set, as well as unconvincing and irregular interpolation of unsampled regions – findings that are corroborated by the discussions in [CBC*01, OBA*03, OBS03], for example. The scheme we propose in Section 3 solves these problems, previously associated with compactly supported basis functions.

3. A Fast Scheme using Compactly Supported Basis Functions

Here we present a fast approximate scheme for solving the problem developed in the previous Section, in which we restrict the class of functions to the span of a compactly supported, multi-scale basis, as described in Section 3.1. An algorithm for choosing the basis is given in Section 3.2 and the means of mimicking the thin-plate regulariser within this basis is shown in Section 3.3. Finally in Section 3.4 we compare the method to the FMM approach taken in [CBC*01].

3.1. Restricting the Set of Available Functions

Computationally, using the thin-plate spline leads to the problem that the linear system we need to solve (Equations 6 and 7), which is of size $m(d+1)$, is dense in the sense of having almost all non-zero entries. Since solving such a system naïvely has a cubic time complexity in m, we propose forcing $f(\cdot)$ to take the form:

$$f(\cdot) = \sum_{k=1}^{p} \pi_k f_k(\cdot), \tag{11}$$

where the individual basis functions are

$$f_k(\cdot) = \phi(||\mathbf{v}_k - \cdot||/s_k),$$

for some compactly supported function $\phi : \mathbb{R}^+ \to \mathbb{R}$ which takes the value zero for arguments greater than one. The $\mathbf{v}_k$ and s_k are the basis function centres and dilations (or scales), respectively. For ϕ we choose the B_3-spline function:

$$\phi(r) = \sum_{n=0}^{4} \frac{(-1)^n}{d!} \binom{n}{d+1} (r + (\frac{d+1}{2} - n))^d_+, \tag{12}$$

although this choice is rather inconsequential since, as we shall ensure, the regulariser is unrelated to the function basis – any smooth compactly supported basis function could be used. In order to achieve the same interpolating properties as the thin-plate spline, we wish to minimise our regularised risk function given by Equation 4 within the span of Equation 11. The key to doing this is to note that as given before in Equation 9, the regulariser (function norm) can be written as $\|f\|^2_{\mathcal{H}} = \langle \psi f, \psi f \rangle_{\mathcal{L}_2}$. Given this fact, a straightforward calculation leads to the following system for the optimal π_k (in the sense of minimising Equation 4):

$$\left(K_{\mathrm{reg}} + C_1 K_{xv}^{\mathsf{T}} K_{xv} + C_2 \sum_{l=1}^{d} K_{xvl}^{\mathsf{T}} K_{xvl} \right) \pi = C_2 \sum_{l=1}^{d} K_{xvl} N_l, \tag{13}$$

where we have defined the following matrices:

$$[K_{\mathrm{reg}}]_{k,k'} = \langle \psi f_k, \psi f_{k'} \rangle_{\mathcal{L}_2}; \quad [K_{xv}]_{i,k} = f_k(\mathbf{x}_i); \quad [K_{xvl}]_{i,k} = [(\nabla f_k)(\mathbf{x}_i)]_l; \quad [\pi]_k = \pi_k; \quad [N_l]_i = [\mathbf{n}_i]_l.$$

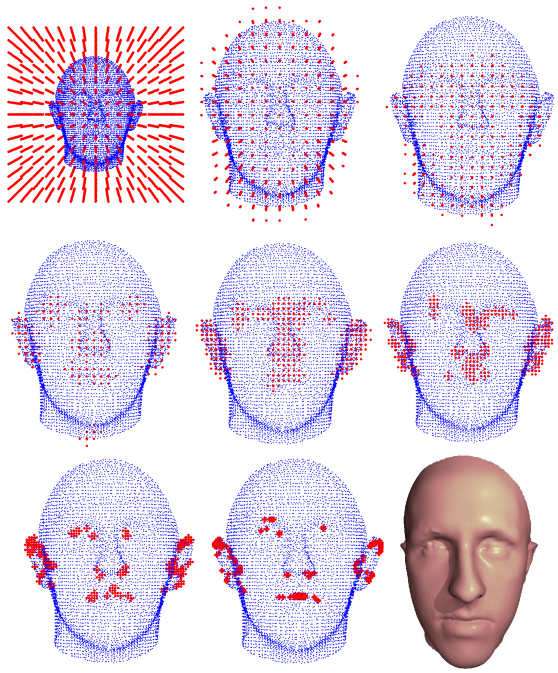


Figure 2: *The input data set taken from a light based depth scanner (blue) with the basis centres produced by Algorithm 1 (red) (basis function support decreasing from top-left to bottom-right), and a rendering of the resultant fitted surface (which appears elongated due to the difference in image plane projection method). The parameters of Algorithm 1 were* $\varepsilon = 0.02$, $t = 1.3$, $r = 1/3$ *and* $s = 0.2$.

The point of the approximation is that the coefficients that we need to determine are now given by a *sparse* p-dimensional positive semi-definite linear system, which can be constructed efficiently by simple code that takes advantage of software libraries for fast nearest neighbour type searches (see *e.g.* [MPL00]). Moreover, the system can be solved efficiently using *conjugate gradient* type methods [GV96]. In the following Section 3.2 we describe a way of constructing a basis with $p \ll m$ that results in a high degree of sparsity in the linear system, but still contains good solutions. The critical question of how to compute K_{reg} is then answered Section 3.3.

3.2. Construction of the Function Basis

The success of the above scheme hinges on the ability to construct a good function basis, *i.e.* one that:

1. Spans a set of functions containing good solutions to the original system given by Equation 4 (in particular having a support which covers a given region of interest – here assumed to be the unit hyper-box).
2. Leads to fast solution of Equation 13.
3. Can be evaluated and constructed quickly.

To these ends, we make use of some standard ideas for 3D point cloud simplification [PGK02]. The main idea that we use is that the closer a set of points within a given region is to being linear, the less information is required to sufficiently describe the shape within that region. The way in which we apply this idea for the present task is described precisely by the pseudo-code of Algorithm 1. The basic idea can be seen more easily in Figure 2 however, and the remainder of this sub Section could be safely skipped at first reading.

Note that although Algorithm 1 is only intuitively justified, we will be minimising precisely the well justified risk of Equation 4 within the span of the resultant basis functions. The results of the overall algorithm therefore do not depend too strongly on the output of Algorithm 1, in the sense that adding more basis functions can only make the overall solution better. The algorithm makes use of two auxiliary functions, *curvature* and *grid*. The first of these functions, *curvature*$(\mathcal{X})$, used also in [PGK02], returns the ratio of the smallest to the sum of all of the eigenvalues of the covariance matrix of the set $\mathcal{X} = \{\mathbf{x}_i \in \mathbb{R}^d\}_{1 \le i \le p}$. If we define the covariance matrix as $C = DD^\mathsf{T}$ where

$$D = \left[\mathbf{x}_1 - \overline{\mathcal{X}}, \mathbf{x}_2 - \overline{\mathcal{X}}, \ldots, \mathbf{x}_p - \overline{\mathcal{X}}\right],$$

and $\overline{\mathcal{X}}$ is the empirical mean of $\mathcal{X}$, then letting the diagonal matrix $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots \lambda_d)$ and the ortho-normal matrix V be the solutions to the eigen-system $CV = \Lambda V$, the function returns

$$\text{curvature}(\mathcal{X}) = \frac{\min_{i \in 1 \ldots d} |\lambda_i|}{\sum_{j=1}^{d} |\lambda_j|},$$

which is a real number in $[0, 1/d]$ that is smaller for point sets that lie closer to a linear manifold. The second function, *grid*$(\mathcal{S}, w)$ returns points from a grid of spacing w immediately surrounding the set $\mathcal{S} \subset \mathbb{R}^d$, that is

$$\text{grid}(\mathcal{S}, w) = \mathcal{D}(\mathcal{S}, w) \bigcap \left\{ w\mathbf{z} : \mathbf{z} \in \mathcal{Z}^d \right\},$$

where $\mathcal{D}(\mathcal{S}, w)$ is the union of the set of hyper-spheres centered at all elements of $\mathcal{S}$ (in other words, a *dilation* of $\mathcal{S}$):

$$\mathcal{D}(\mathcal{S}, w) = \left\{ \mathbf{x} \in \mathbb{R}^d : \left(\min_{\mathbf{x}' \in \mathcal{S}} \left\| \mathbf{x} - \mathbf{x}' \right\| \right) \le w \right\}.$$

3.3. Computing the Regularisation Matrix

We now come to the crucial point of calculating K_{reg}, which can be thought of as the regulariser in Equation 11. The present Section is highly related to [WCS05], however there numerical methods were resorted to for the calculation of K_{reg} – presently we shall derive closed form solutions. Also worth comparing to the present Section is [FG06], where a prior over the expansion coefficients (here the π) is used to mimic a given regulariser within an arbitrary basis, achieving a similar although less precise result in comparison to the present approach. As we have already noted we can write

Algorithm 1
Input: Data $\mathcal{X} = \{\mathbf{x}_i \in \mathbb{R}^d\}_{1 \leq i \leq m}$ (scaled to lie within the unit hyper-box); Curvature tolerance $\varepsilon \in [0,1]$; Initial basis support s; support decrement ratio $t > 1$; "Basis grid width" to "basis support" ratio $r \in [0,1]$.
Output: Basis function centre/support pairs $\mathcal{B} = \{(\mathbf{v}_k, s_k) \in \mathbb{R}^d \times \mathbb{R}^+\}$.

$\tilde{\mathcal{X}} \leftarrow \mathcal{X}$
$\mathcal{B} \leftarrow \text{grid}\left(\left\{\mathbf{x} : \|\mathbf{x}\|_\infty \leq 1\right\}, rs\right)$
while $\tilde{\mathcal{X}} \neq \emptyset$ **do**
  $s \leftarrow s/t$
  for all $\mathbf{x} \in \tilde{\mathcal{X}}$ **do**
    if $\text{curvature}\left(\{\mathbf{x}' \in \mathcal{X} : \|\mathbf{x}' - \mathbf{x}\| < s\}\right) < (\varepsilon/d)$ **then**
      $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\mathbf{v}, s) : \mathbf{v} \in \text{grid}\left(\{\mathbf{x}\}, rs\right)\}$
      $\tilde{\mathcal{X}} \leftarrow \tilde{\mathcal{X}} \setminus \mathbf{x}$
    end if
  end for
end while

$\|f\|_{\mathcal{H}}^2 = \langle \psi f, \psi f \rangle_{\mathcal{L}_2}$ [Wah90], so that for the function given by Equation 11 we have:

$$\begin{aligned}\left\| \sum_{j=1}^{p} \pi_j f_j(\cdot) \right\|_{\mathcal{H}}^2 &= \left\langle \psi \sum_{j=1}^{p} \pi_j f_j(\cdot), \psi \sum_{k=1}^{p} \pi_k f_k(\cdot) \right\rangle_{\mathcal{L}_2} \\ &= \sum_{j,k=1}^{p} \pi_j \pi_k \left\langle \psi f_j(\cdot), \psi f_k(\cdot) \right\rangle_{\mathcal{L}_2} \\ &\doteq \pi^{\mathsf{T}} K_{\text{reg}} \pi.\end{aligned}$$

To build the sparse matrix K_{reg}, a fast range search library (*e.g.* [MPL00]) can be used to identify the non-zero entries – that is, all those $[K_{\text{reg}}]_{i,j}$ for which i and j satisfy:

$$\|\mathbf{v}_i - \mathbf{v}_j\| \leq s_i + s_j.$$

In order to evaluate $\left\langle \psi f_j(\cdot), \psi f_k(\cdot) \right\rangle_{\mathcal{L}_2}$, it is necessary to solve the integral of Equation 10, the full derivation of which we relegate to [WSC06] – here we just provide the main results. It turns out that since the f_i are all dilations and translations of the same function $\phi(\|\cdot\|)$, then it is sufficient solve for the following function of s_i, s_j and $\mathbf{d} \doteq \mathbf{v}_i - \mathbf{v}_j$:

$$\left\langle \psi\phi((\cdot)s_i - \mathbf{d}), \psi\phi((\cdot)s_j) \right\rangle_{\mathcal{L}_2},$$

which it turns out is given by

$$\mathcal{F}_{\boldsymbol{\omega}}^{-1}\left[\frac{(2\pi j \|\boldsymbol{\omega}\|)^{2m}}{|s_1 s_2|^d} \Phi\left(\frac{\boldsymbol{\omega}}{s_1}\right) \Phi\left(\frac{\boldsymbol{\omega}}{s_2}\right) \right](\mathbf{d}), \tag{14}$$

where $j^2 = -1$, $\Phi = \mathcal{F}_{\mathbf{x}}[\phi(\mathbf{x})]$, and by $\mathcal{F}$ and $\mathcal{F}^{-1}$ we mean the Fourier and inverse Fourier transform operators in the subscripted variable.

Computing Fourier transforms in d dimensions is in general a difficult task, but for radial functions $g(\mathbf{x}) = g_r(||\mathbf{x}||)$ it can be made easier by the fact that the Fourier transform in d dimensions (as well as its inverse) can be computed by the single integral:

$$\mathcal{F}_{\mathbf{x}}\left[g_r(\|\mathbf{x}\|)\right](\|\boldsymbol{\omega}\|) = \frac{(2\pi)^{\frac{d}{2}}}{\|\boldsymbol{\omega}\|^{\frac{d-2}{2}}} \int_0^\infty r^{\frac{d}{2}} g_r(r) J_{\frac{d-2}{2}}(\|\boldsymbol{\omega}\| r) \mathrm{d}r,$$

where $J_\nu(r)$is the ν-th order Bessel function of the first kind.

Unfortunately the integrals required to attain Equation 14 in closed form cannot be solved for general dimensionality d, regularisation operator ψ and basis function form ϕ, however we did manage to solve them for arguably the most useful case: $d = 3$ with the $m = 2$ thin plate energy and the B_3-spline basis function of Equation 12. The resulting expressions are rather unwieldy however, so we give only an implementation in the C language in the Appendix of [WSC06], where we also show that for the cases that cannot be solved analytically the required integral can at worst always be transformed to a two dimensional integral for which one can use numerical methods. Also note that if one would like to include a few additional thin-plate kernel functions in the basis (if the infinite support is useful, for example), then computing the necessary regulariser inner products could not be simpler – due to the reproducing property we have $\left\langle \psi k(x,\cdot), \psi f_j(\cdot) \right\rangle_{\mathcal{L}_2} = f_j(\mathbf{x})$.

3.4. Relationship to the Fast Multipole Method

We believe that the scheme presented here in Section 3 can be useful for a wide range of problems, particularly in computer graphics. Specifically it can be applied to any problem involving large amounts of fairly low-dimensional data (1-4 dimensions posing no problem) in which it is necessary to estimate a regularised function that either satisfies some set of constraints or minimises some cost function.

A very different and important approximation scheme based on the FMM [BG97] is taken in [CBC*01] however, and so we now briefly compare the two approaches. The FMM based method proceeds as follows. In order to approximately solve solve the system

$$\mathbf{y} = (K + I/C_1)\alpha \tag{15}$$

for the coefficients vector α of Equation 3, a conjugate gradients type method [GV96] is employed. Conjugate gradient methods only require that the right hand side of 15 be evaluated for a given α. This evaluation can done quickly using the FMM, which allows one to approximately compute sequences of summations of the form

$$v(\mathbf{x}_i) = \sum_i^m \alpha_i k(\mathbf{x}_i, \mathbf{x}); \quad i = 1 \ldots m \tag{16}$$

in time $O(m\log(m))$ – an improvement over the naïve $O(m^2)$. Thus, an approximate solution of the system can be attained in time $O(pm\log(m))$, where p is the number of iterations required by the conjugate gradients solver.

The FMM is a very different tool to the sparse approximation we describe here. Compactly supported basis functions cannot be used to compute general summations of the form 16, but they do have advantages for the specific problem we consider in the present work. One important advantage is that the present method is simpler to implement. A second, and very important advantage however, is that various properties of the function (the derivatives for example, as used heavily in the present work) can be computed with minimal additional implementation effort – the "hard work" is done by a range search algorithm, for which many well developed software libraries exist. To understand this, note that evaluating Equation 11 can be done efficiently by simply using a range search library to identify the contributing basis functions, over which the the summation can then be directly computed. Modifying this to give the gradient of the function, for example, is no more difficult than implementing the gradient computation for a single basis function. In contrast, the FMM method requires a far greater additional effort for each different function to be computed – for details on the FMM see [BG97].

To summarise, the main obstacle to be overcome in applying compactly supported basis functions for regularised function approximation is the computation of K_{reg}. However given the analysis in Section 3.3 this is trivial.

4. Experiments

Here we demonstrate the method on various problems. Firstly however we explain how we ray-trace the implicit surfaces.

4.1. Ray Tracing

To render 3D implicit surfaces we use a ray tracer, the usual choice for high quality 3D graphics. To ray trace a given object one need only be able to find the first interesection (or absence thereof) between the object and a light ray with given starting point $\mathbf{x}_0$ and direction $\mathbf{n}$. This is particularly simple for the present case – as the embedding function is smooth and differentiable everywhere the following iterative second order approximation line search for the zero set converges rather quickly, and in our experiments took, on average, approximately 2-3 iterations per intersection search.

Note that we found using a first order approximation to be slower. This is probably due to the fact that a considerable part of the time needed to compute the Hessian of the function is taken by the range search which finds the contributing basis functions. Since these need to be found in a first order method anyway, it is not too much more expensive to use a more accurate second order approximation which converges in fewer iterations.

At each iteration we make the second order approximation $f(\mathbf{x}) = (\mathbf{x}-\mathbf{x}_0)^\mathsf{T} A(\mathbf{x}-\mathbf{x}_0) + \mathbf{w}^\mathsf{T}(\mathbf{x}-\mathbf{x}_0) + b$ (here A, $\mathbf{w}$ and b are available analytically) and solve with respect to λ the equation

$$f(\mathbf{x}_0+\lambda\mathbf{n}) = \lambda^2\mathbf{n}^\mathsf{T}A\mathbf{n} + \lambda\mathbf{w}^\mathsf{T}\mathbf{n} + b = 0,$$

which is a quadratic function of λ with roots:

$$\lambda^* = \frac{2t_c}{-t_b \pm \sqrt{t_b^2 - 4t_a t_c}},$$

where we have defined

$$t_a = \mathbf{n}^\mathsf{T}A\mathbf{n}; \quad t_b = \mathbf{w}^\mathsf{T}\mathbf{n}; \quad t_c = b.$$

We take the lambda of smallest magnitude, since we seek the *first* intersection. Note that if the system cannot be solved for real λ then the real part of λ^* minimises of $(f(\mathbf{x}_0+\lambda\mathbf{n}))^2$. The implementation of our ray tracer is rather simple and not at all optimised, so we do not provide timing results in the following Section 4.

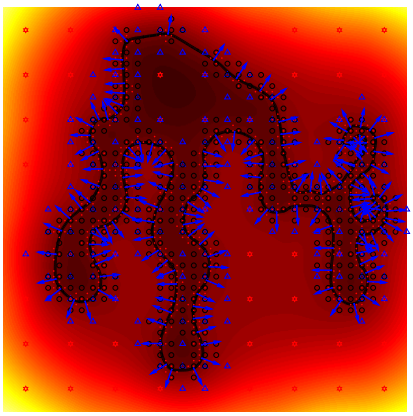


Figure 3: *An example solution in two dimensions. Data points are depicted as dots with normals vectors as arrows, the zero set as a black contour and the embedding function value as the background colour. All of the basis function centers of various scales (produced by Algorithm 1 with the parameter setting $\varepsilon = 0$ and a lower limit on the basis support) are plotted with various markers corresponding to the basis function support (the stars have the largest and the red dots the smallest support).*

4.2. 2D Examples

In order to show how the embedding function behaves (rather than just its zero zet) we include a two dimensional example in Figure 3. Note that in spite of the function basis including various scales of basis function at irregular positions, the regularisation scheme outlined in Section 3.3 leads to a solution that is well behaved over the entire support of the function. Further evidence of the efficacy of the regulariser is given in the accompanying video which shows the performance on a rather complex fractal-like data set.

Figure 4: *Various values of the regularisation parameters lead to various amounts of "smoothing" – here we set $C_1 = C_2$ in Equation 4 to an increasing value from left to right of the figure.*

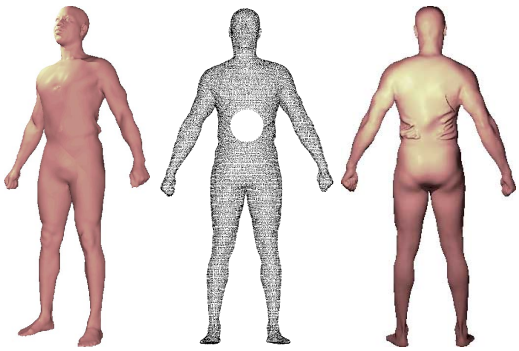


Figure 5: *Missing data is handled by the interpolating properties of the regulariser – the ray traced model was constructed from the incomplete data set depicted in the middle. Note that the model is wearing a loose t-shirt.*

Figure 6: *Ray traced three dimensional implicits, "Happy Buddha" (543K points with normals) and the "Thai Statue" (5 million points with normals).*

4.3. 3D Examples

Here we fit models to several 3D data sets of up to 14 million data points. For each model, the timing of the various stages of the fitting process is given in Table 1. The following general observations can be made regarding this table:

- High compression ratios can be achieved, in the sense that a relatively small number of basis functions can represent the shape.
- The fitting time scales rather well, from 38 seconds for the Stanford Bunny with 35 thousand points to 4 hours 23 minutes for the Lucy statue with 14 million points. Moreover, after accounting for the different hardware the times seem to be similar to those of the FMM approach [CBC*01].

Some example ray traced images of the resulting models are given in Figures 1 and 6, and the well-behaved nature of the implicit over the entire 3D volume of interest is shown for the Lucy data-set in the accompanying video.

In practice the system is extremely robust – we achieved excellent results without any parameter adjustment – smaller values of C_1 and C_2 in Equation 4 simply lead to the smoothing effect shown in Figure 4. The system also handles missing and noisy data gracefully, as shown in Figures 5 and 7.

4.4. 4D Examples

It is also possible to construct higher dimensional implicit surfaces, particularly interesting being a 4D representation (3D + "time") of a continuously evolving 3D shape – one possible use for this is as means of constructing 3D animation sequences from a time series of 3D point cloud data. For example, optical scanning devices can produce 3D scans at a rate of the order of a hundred frames per second. In this case both spatial and temporal information can help to resolve noise or missing data problems within individual scans. We demonstrate this in the accompanying video, which shows

Name	# Points	# Bases	Basis	K_{reg}	K_{xv} & $K_{zv\nabla}$	Multiply	Solve	Total
Stanford Bunny	34834	9283	0.44	2.4	3.65	11.74	20.44	38.67
Face	75970	7593	0.73	1.93	7.04	20.29	15.97	45.96
Armadillo	172974	45704	6.62	8.52	37.04	123.36	72.34	247.88
Dragon	437645	65288	14.38	16.3	70.87	322.79	1381.38	1805.72
Happy Buddha	543197	105993	117.4	27.39	99.35	423.71	2909.33	3577.18
Asian Dragon	3609455	232197	441.61	60.86	608.29	1884.97	1009.47	4005.2
Thai Statue	4999996	530966	3741.95	197.53	1575.62	3121.15	2569.48	11205.7
Lucy	14027872	364982	1425.77	170.45	3484.08	9367.71	1340.48	15788.5

Table 1: *Timing results with a 2.4GHz AMD Opteron 850 processor, for various 3D data sets. Column one is the number of points, each of which has an associated normal vector, and column two is the number of basis vectors produced by Algorithm 1. The remaining columns are all in units of seconds: column three is the time taken by Algorithm 1, columns four and five are the times required to construct the indicated matrices, column six is the time required to multiply the matrices as per Equation 13, column seven is the time required to solve that same equation for π and the final column is the total fitting time.*

that 4D surfaces yield superior 3D animation results in comparison to constructing a separate 3D models for each instant. It is also interesting to note that the interpolating behaviour that we saw in three dimensions also occurs in four dimensions – in the accompanying video we effectively interpolate between two three dimensional shapes.

5. Summary

We have presented a number of ideas that are both theoretically and practically useful for the computer graphics community, and demonstrated them within the framework of implicit surface fitting. The work revolves mainly around the effective use of compactly supported function bases for low dimensional function approximation problems.

Many authors have discussed and demonstrated fast but limited quality results that occur with compactly supported function bases. The present work is unique in precisely minimising a well justified regulariser within the span of such a basis, thereby achieving fast *and* high quality results.

We also show how normal vectors can be incorporated directly into the common regression based implicit surface fitting framework, giving a corresponding generalisation of the representer theorem.

We demonstrated the efficacy of the algorithm on 3D problems consisting of up to 14 million data points. In the accompanying video we also showed the advantage of constructing a 4D surface (3D + time) for 3D animation, rather than a sequence of 3D surfaces.

Acknowledgements

We thank Christian Wallraven for providing the face data used in both Figure 2 and the accompanying video.

References

[ABCO*01] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Point set surfaces. In *IEEE Visualization 2001* (October 2001), IEEE Computer Society, pp. 21–28. 1

[BBX95] BAJAJ C. L., BERNARDINI F., XU G.: Automatic reconstruction of surfaces and scalar fields from 3D scans. *Computer Graphics 29*, Annual Conference Series (1995), 109–118. 1

[BG97] BEATSON R., GREENGARD L.: A short course on fast multipole methods. In *Wavelets, Multilevel Methods and Elliptic PDEs* (1997), pp. 1–37. 6, 7

[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 *ACM SIGGRAPH 2001* (2001), ACM Press, pp. 67–76. 2, 4, 6, 8

[Duc77] DUCHON J.: Splines minimizing rotation-invariant semi-norms in sobolev spaces. *Constructive Theory of Functions of Several Variables* (1977), 85–100. 4

[FG06] FRANZ M. O., GEHLER P. V.: How to choose the covariance for gaussian process regression independently of the basis. In *Proc. Gaussian Processes in Practice Workshop* (2006). 5

[GR97] GREENGARD L., ROKHLIN V.: A fast algorithm for particle simulations. *J. Comp. Phys.* (1997), 280–292. 2

[GV96] GOLUB G. H., VAN LOAN C. F.: *Matrix Computations*, 2nd ed. The Johns Hopkins University Press, Baltimore MD, 1996. 5, 6

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques* (New York, 1992), ACM Press, pp. 71–78. 1

[KW71] KIMELDORF G., WAHBA G.: Some results on

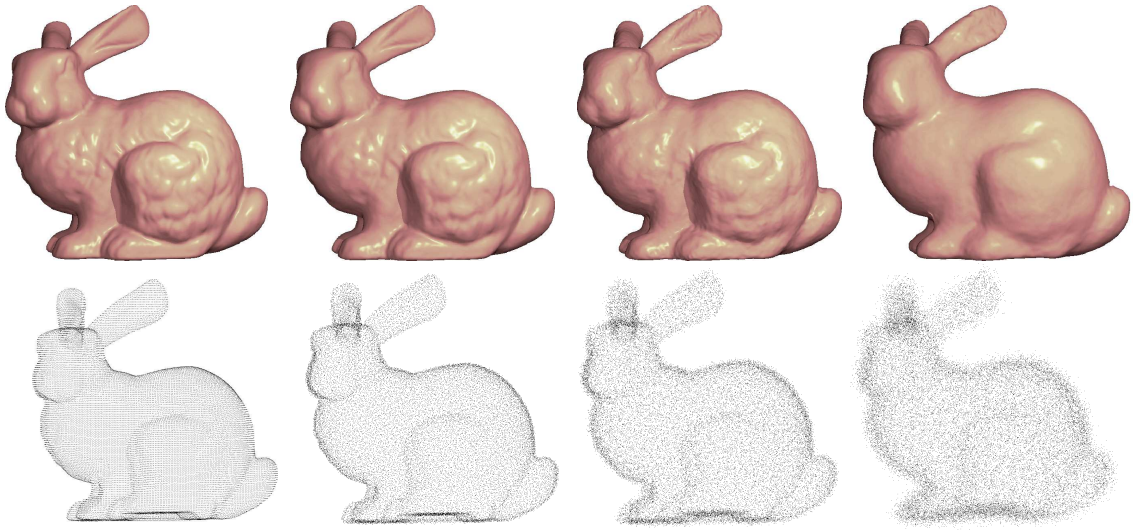


Figure 7: *Reconstruction of the Stanford bunny after adding Gaussian noise with standard deviation, from left to right, 0, 0.6, 1.5 and 3.6 percent of the radius of the smallest enclosing sphere – the normal vectors were similarly corrupted assuming they had length equal to this radius. The parameters C_1 and C_2 were chosen automatically using five-fold cross validation.*

Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications 33* (1971), 82–95. 3

[MPL00] MERKWIRTH C., PARLITZ U., LAUTERBORN W.: Fast nearest neighbor searching for nonlinear signal processing. *Phys. Rev. E 62*, 2 (2000), 2089–2097. 5, 6

[Mur91] MURAKI S.: Volumetric shape description of range data using "blobby model". In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques* (New York, 1991), ACM Press, pp. 227–235. 1

[MYC*01] MORSE B. S., YOO T. S., CHEN D. T., RHEINGANS P., SUBRAMANIAN K. R.: Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *SMI '01: Proc. Intl. Conf. on Shape Modeling & Applications* (Washington, 2001), IEEE Computer Society. 2

[OBA*03] OHTAKE Y., BELYAEV A., ALEXA M., TURK G., SEIDEL H.-P.: Multi-level partition of unity implicits. *ACM Transactions on Graphics 22*, 3 (July 2003), 463–470. 1, 2, 4

[OBS03] OHTAKE Y., BELYAEV A., SEIDEL H.-P.: A multi-scale approach to 3d scattered data interpolation with compactly supported basis functions. In *Proc. Intl. Conf. Shape Modeling* (Washington, 2003), IEEE Computer Society. 2, 4

[PGK02] PAULY M., GROSS M., KOBBELT L. P.: Efficient simplification of point-sampled surfaces. In *VIS '02: Proceedings of the conference on Visualization '02* (Washington, 2002), IEEE Computer Society, pp. 163–170. 5

[Set98] SETHIAN J.: Level set methods and fast marching methods: Evolving interfaces in computational geometry, 1998. 1

[SHS01] SCHÖLKOPF B., HERBRICH R., SMOLA A. J.: A generalized representer theorem. In *COLT '01/EuroCOLT '01: Proceedings of the 14th Annual Conference on Computational Learning Theory* (London, UK, 2001), Springer-Verlag, pp. 416–426. 3

[SOS04] SHEN C., O'BRIEN J. F., SHEWCHUK J. R.: Interpolating and approximating implicit surfaces from polygon soup. In *ACM SIGGRAPH 2004* (Aug. 2004), ACM Press. 1

[TO99] TURK G., O'BRIEN J. F.: Shape transformation using variational implicit functions. In *Proceedings of ACM SIGGRAPH 1999* (August 1999), pp. 335–342. 2

[Wah90] WAHBA G.: *Spline Models for Observational Data*. Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990. 2, 3, 4, 6

[WB98] WHITAKER R. T., BREEN D. E.: Level-set models for the deformation of solid objects. In *Implicit Surfaces 98 Proceedings* (1998), Jules Bloomenthal D. S., (Ed.), Eurographics/ACm Workshop, pp. 19–35. 1

[WCS05] WALDER C., CHAPELLE O., SCHÖLKOPF B.: Implicit surface modelling as an eigenvalue problem. *Proceedings of the 22nd International Conference on Machine Learning* (2005). 5

[WSC06] WALDER C., SCHÖLKOPF B., CHAPELLE O.: *Implicit Surface Modelling with a Globally Regularised Basis of Compact Support*. Tech. rep., Max Planck Institute for Biological Cybernetics, Department of Empirical Inference, Tübingen, Germany, April 2006. 3, 6