Poisson Surface Reconstruction on non-Cartesian Lattices

***Abstract*—In this work, we cast the well-known Poisson surface reconstruction algorithm into a more general setting, we reformulate it in terms of shift invariant spaces (spaces that are spanned by lattice translates of an admissible generating function). The treatment is general, but our specific interest is in reconstructing a surface of a solid model from an oriented point-set within function spaces defined over the body centred cubic lattice. We also propose a general framework for approximating the solution to Poisson?s equation in a hypercube with zero Dirichlet boundary conditions, and a new variational scheme to re-sample the initial oriented point-set onto a regular grid. Once the points have been re-sampled onto a grid, the rest of the pipeline is purely based on digital filtering. We also analyze the error incurred via the digital filtering approximation methodology and propose a Fourier domain error kernel that can be used to design solution filters that fully exploit the approximation capabilities of the target space. Finally, we show that a host of Poisson-like methods fail to take advantage of the full approximation spaces over which they are defined.**

I. Introduction

Reconstructing a solid model from a scattered set of points is a common and well studied problem in computer graphics. Interest in this problem is rooted in the desire to convert real life data-sets – acquired from range and now increasingly from commodity hardware like the Kinect – to 3D polygonal models. The idea also has a applications in model re-meshing and hole filling. [[XXX This needs more meat, it's a pretty weak ending to the intro paragraph.]]

Given an oriented point-set, a common approach is to find an implicit function whose iso-contour corresponds to an approximation of the original surface. In this work we take a "signal processing" approach to this problem – we cast the problem into a general shift-invariant setting. More explicitly, we reconstruct the desired implicit function within a function space spanned by lattice translates of a single generating function. The extension to this class of function spaces allows us to consider anchoring our basis functions at non-Cartesian lattice sites (Section **??**). Our approach is general and applicable to any valid lattice and generator, but our implementation is narrowed to applicable function spaces over the *Body Centred Cubic* (BCC) and *Cartesian* (CC) lattices.

The advantage of moving to the BCC lattice is two fold. First, it's well known that the BCC lattice generates the optimal sampling pattern in $\mathbb{R}^3$. The argument for optimality is simple; the sampling of a function with respect to the BCC lattice is equivalent to a periodization of it's frequency spectrum about the BCC's dual lattice, the *Face Centered Cubic* (FCC) lattice. Optimality follows from the observation that the FCC lattice is the optimal sphere packing lattice in three dimensions, that is, it packs frequency replica as tightly as possible in the Fourier domain, resulting in less *pre-aliasing* from sampling. For isotropically band-limited functions, this tighter packing of frequency content allows for about 30% more information to be captured when compared to samplings generated from a Cartesian lattice. The second advantage to moving to the BCC lattice is that there exist reconstruction filters on the BCC lattice that outperform commonly used filters of equivalent order on the CC lattice in terms of both speed and accuracy [**?**] (in software implementations) – these filters are also more compact than their typical CC counterparts. In the context of our surface reconstruction scheme, this gives rise to regularization matrices that are more sparse on the BCC lattice and tend to provide faster convergence when optimizing the initial point set (Section **??**.)

Our surface reconstruction methodology follows the general Poisson approach, but with a few twists. First, we construct a smoothed approximation to the gradient field of a model's indicator function. The divergence of the gradient field is estimated and subsequently fed into a Poisson solver that outputs the coefficients needed to approximate the smoothed indicator function in a target shift invariant space. This Poisson solver is tailored to cater to the approximation power provided by each space. However, our approach is novel in two notable ways. Firstly, we employ a variational scheme to obtain a lattice-based approximation of the gradient field. This scheme depends only on the generating kernel of the target space and optimizes a functional that incorporates interpolation and smoothness constraints. Secondly, we utilize the theory behind shift-invariant spaces to seek discretizations of the divergence and Laplacian operators that are tailored to exploit the full approximation capabilities of the target space.

Additionally, we consider the possibility of approximating the smoothed gradient field representation within function spaces that are spanned by shifted versions of a single generating kernel. In the context of gradient estimation, introducing a shift in the direction of the derivative has shown to improve overall gradient approximation fidelity in terms of gradient orientation and magnitude, as well as displaying the ability to capture higher frequency details that are smoothed out in non-shifted schemes [**?**]. Since the divergence operator is intimately connected to the gradient operator, these results are of interest to us. Using an appropriate discretization of the derivative/divergence operator is an important step in recovering the indicator function of the initial model. [[XXX Need to emphasize the error-kern here]]

To summarize, our contributions are as follows:

- We cast the Poisson surface reconstruction problem into the framework of shift-invariant spaces.
- Specifically, we investigate the reconstruction of surfaces

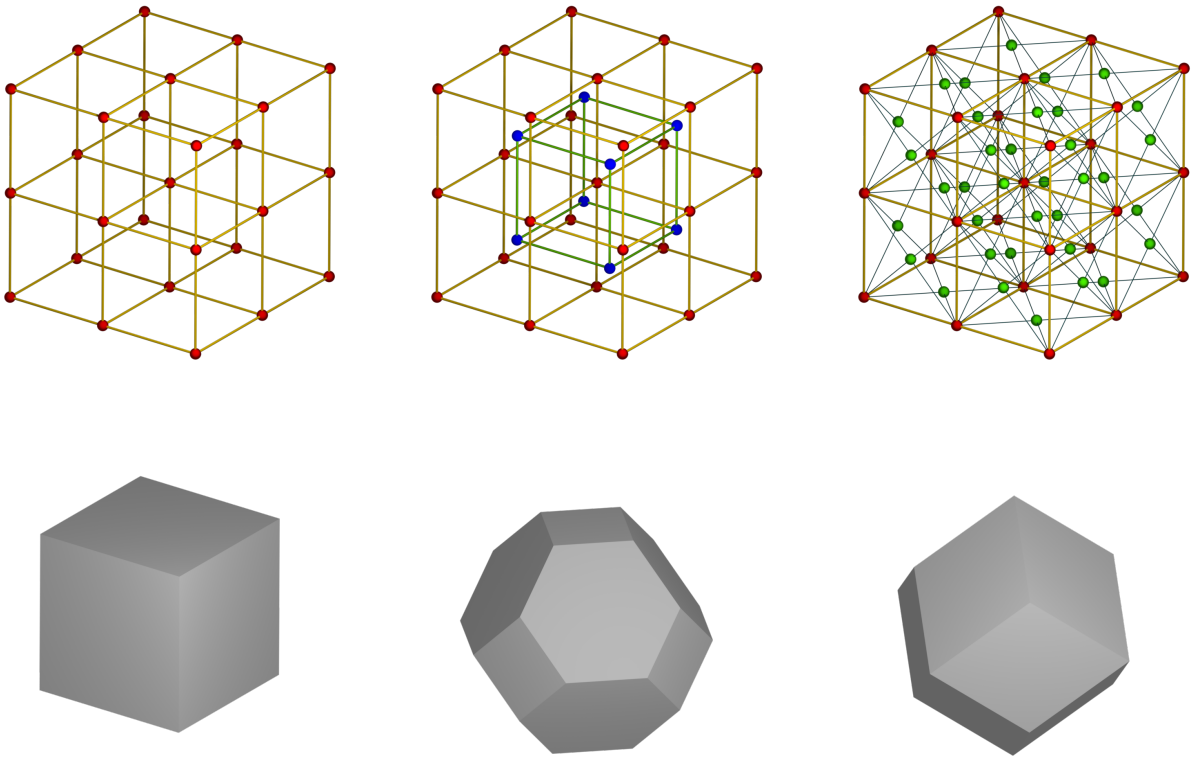

Fig. 1. Sampling lattices (top) with with their corresponding Voronoi regions (bottom). Left to right are the CC, BCC and FCC lattices respectively.

in both a sub-optimal (Cartesian) and an optimal (BCC) box-spline function space.

- We present a variational scheme that not only considers the usual centered generating kernels, but allows for the possibility of recovering each component of the gradient field in a space spanned by a shifted kernel. On the Cartesian lattice, these components are orthogonal whereas on the BCC lattice, the components are non-orthogonal and correspond to the directions associated with the underlying box-spline matrix.
- We present an extension of the error kernel of Blu and Unser to general linear operators, and show how to design filters that respect the approximation order of the approximation space.
- While our focus is on comparing the surface reconstruction algorithm on both the CC and BCC lattices, we also provide qualitative and quantitative comparisons between the presented technique (on both the Cartesian and BCC lattices) and similar methods.

[[XXX OVERALL: What is lacking from this intro is the true purpose of this work (from the surface reconstruction aspect.) We want to show that the BCC beats the CC lattice. NOT that our scheme is better than Kazhdan's. We do show this, but we need to emphasize it above!]]

II. Preliminaries

A. Poisson Surface Reconstruction

We denote the indicator function of a model M as

$$\chi_M(\mathbf{x}) := \begin{cases} 1, & \text{if } \mathbf{x} \text{ is in the interior of the model } M \\ \frac{1}{2}, & \text{if } \mathbf{x} \text{ is on the model's surface} \\ 0, & \text{if } \mathbf{x} \text{ is exterior to } M \end{cases}.$$

Integral to formulating surface reconstruction as a Poisson problem, is the observation that if one obtains some function $\vec{V}(\mathbf{x}) \approx \nabla \chi_M$ that approximates the smoothed gradient of the indicator function, finding an approximation to the indicator function reduces to finding a $\tilde{\chi}$ such that

$$\Delta \tilde{\chi} = \nabla \cdot \vec{V}. \tag{1}$$

We define the set of input surface points to be $P := \{(\mathbf{p}_1, \mathbf{n}_1), (\mathbf{p}_2, \mathbf{n}_2) \dots (\mathbf{p}_M, \mathbf{n}_M) : (\mathbf{p}_i, \mathbf{n}_i) \in \mathbb{R}^3 \times \mathbb{R}^3\}$ where $\mathbf{p}_i$ and $\mathbf{n}_i$ are the position and normal vector of the i^{th} sample respectively. Orientation can be succinctly represented with only two values, however, it is convenient to keep it denoted as the normal at a point. This is because the length of the normal can be used to encode the sampling density of the point-set at a point. For the following discussion, we also need to introduce the notation $\mathcal{P}_{\mathbf{s}}$, which denotes a patch of the surface model about the sample location $\mathbf{s}$.

Inspired by the approach of Kazhdan et al. [**?**] the approximation $\vec{V}(\mathbf{x})$ is defined to be

$$\begin{aligned}\vec{V}(\mathbf{x}) &:= \sum_{(\mathbf{s},\mathbf{n})\in P} |\mathcal{P}_s|\, F(\mathbf{x}-\mathbf{s})\cdot\mathbf{n} \\ &\approx \sum_{(\mathbf{s},\mathbf{n})\in P}\int_{\mathcal{P}_s} F(\mathbf{x}-\mathbf{p})\nabla\chi(\mathbf{p})\mathrm{d}\mathbf{p}.\end{aligned} \tag{2}$$

Thus, $\vec{V}(\mathbf{x})$ approximates the gradient of the indicator smoothed by some filter F, broken up over surface patches of the model. We assume the point sampling to be uniform, thus the area of each patch $|\mathcal{P}_s|$ should be a constant, and our resulting approximation will be scaled by an unknown factor. We return to this in Section **??**. The choice of F places computational constraints on the resulting algorithm.

F is often explicitly chosen to be a compactly supported function. This simplifies and reduces the computational cost of the method when used in conjunction with an oct-tree based representation of $\vec{V}(\mathbf{x})$ [**?**]. However, there is a disadvantage in that F may over smooth data in areas of high curvature [**?**], much in the same way that a representation of a sampled signal may fail to take advantage of the full approximation power of the function space in which it is represented if it is not properly prefiltered. Theoretically, one could reconstruct the surface at higher resolutions, allowing the basis functions to become more "fine", but increasing the total amount of memory used. However, this is not realistic in praxis. Another approach is to impose a type of "interpolation" constraint on the original data [**?**].

Ideally, the filter F should correspond to some globally supported RBF. However, as the support of F grows, so does the cost of evaluating it at lattice sites. Moreover, the added benefit of a globally supported RBF is partially lost as the subsequent steps are entirely lattice-based and do not take the RBF into consideration. Our approach circumvents the choice of F entirely by posing the gradient approximation problem as a variational problem that, in addition to imposing the interpolation constraint, also imposes constraints on the "compactness" and "smoothness" of the gradient approximation. The only kernel we need to consider is the one that generates our target shift-invariant space, i.e. the space used to recover the smoothed indicator function. This type of variational reconstruction can be seen as an approximation to radial basis approaches [**?**], and is well-suited to our Poisson formulation in shift-invariant spaces.

Before presenting our approach (Section **??**), we review the necessary background behind shift-invariant spaces and its connection to irregular sampling.

B. Sampling Lattices

In the univariate case, there is only one way to uniformly sample a signal; take equally spaced samples along the independent axis. However, when sampling multivariate signals, the situation is not as straightforward. A common approach is to inherit the ideas from the univariate case, and sample the function at regular intervals in each axis (known as a Cartesian sampling), while another might be to place samples at the centres of spheres arranged in the densest possible sphere packing (otherwise known as an FCC sampling). If $\mathbf{L}$ is a unimodular matrix, the set $\{\mathbf{L}\cdot\mathbf{n} : \mathbf{n}\in\mathbb{Z}^3\}$ (linear combinations of the vectors of $\mathbb{L}$ with integer coefficients), represents all possible sampling lattices. $\mathbf{L}$ is often called the *generating matrix* for a lattice. To control the sampling rate of a function, a uniform scaling parameter h is introduced. Typically, for the problem of surface reconstruction, our input data are not uniformly distributed in space (or even about the surface of the model). However, the concept of a sampling lattice is fundamental in forming the definition of our target approximation spaces (Section **??**).

The Cartesian lattice is given by the generating matrix $\mathbf{L}_C := \mathbf{I}_3$, where $\mathbf{I}_3$ is the 3×3 identity matrix. There are many factors that contribute to the ubiquity of the Cartesian lattice in practice. Possibly one of its most attractive features is its separability. Owing to this, the implementation of generating functions that span function spaces on the Cartesian lattice may be realized by a tensor product extension of existing univariate techniques. Filtering techniques are also easy to implement, as the Fast Fourier Transform is also separable, and may be applied independently to each axis of the data. However, in terms of sampling efficiency, the BCC lattice is the optimal lattice in $\mathbb{R}^3$, while the Cartesian is sub-optimal.

The Body-Centered Cubic (BCC) lattice is generated by the integer matrix

$$\mathbf{L}_B := \begin{bmatrix}-1 & 1 & 1\\ 1 & -1 & 1\\ 1 & 1 & -1\end{bmatrix}.$$

However, the BCC lattice is non-separable, and requires special care in data retrieval and indexing. Figure **??** shows these lattices and their Voronoi regions, including the dual of the BCC lattice, the FCC lattice (the Cartesian lattice is dual to itself.)

C. Box Splines

Box splines are compact piece-wise polynomial functions $M_{\mathbf{X}} : \mathbb{R}^s \to \mathbb{R}$. The matrix $\mathbf{X}$ denotes a collection of direction vectors, $\mathbf{X} := [\xi_1\ \xi_2 \ldots \xi_n]$. When $n = s$, the function $M_{\mathbf{X}}(\mathbf{x})$ is defined to be $\frac{1}{det|\mathbf{X}|}$ inside the polytope formed by the Minkowski sum of the direction vectors of $\mathbf{X}$, and zero otherwise (that is, $\frac{1}{det|\mathbf{X}|}$ when $\mathbf{x}\in\{\xi_1 t_1+\ldots+\xi_s t_s : t_i\in[0,1)\}$). When $n > s$, the box spline is recursively defined as

$$M_{\mathbf{X}}(\mathbf{x}) := \int_0^1 M_{\mathbf{X}/\xi_n}(\mathbf{x}-t\xi_n)\mathrm{d}t, \tag{3}$$

where $M_{\mathbf{X}/\xi_n}$ is the matrix obtained by removing the column vector ξ_n from $\mathbf{X}$. Box splines are a natural fit for our purposes, and are chosen to span our function spaces. One motivating factor that encourages the use of box splines on the BCC lattice over the Cartesian lattice is that the support of the box splines on the BCC lattice contains less lattice sites as opposed to their Cartesian counterparts. At the same time, they maintain the same order of accuracy and smoothness. This implies that one should expect roughly the same quality reconstruction, with less memory accesses per reconstructed value.

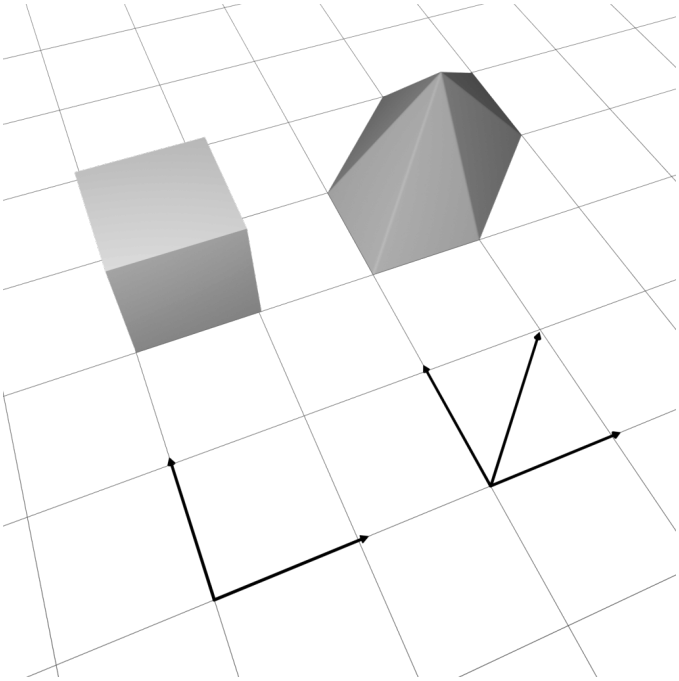

Fig. 2. An example of box splines in 2D. On the left is the box spline comprised of the direction vectors ξ_1, ξ_2, (the case n=s) which is the indicator function of the Minkowski sum of the vectors. The box spline on the right, includes a third vector ξ_3, which can be thought of as "smearing" the first spline along the direction ξ_3.

The evaluation of a box spline, in general, can be achieved by evaluating a simple recursive expression [**?**]. However, for many purposes, this form of evaluation is too slow. There has been a line of research [**?**] aimed towards obtaining fast piecewise polynomial representations of the box splines on the BCC lattice, and showing their advantage in speed and accuracy over splines on the Cartesian lattice when the data to be interpolated are sufficiently smooth.

1) Cartesian Lattice: Choosing $\mathbf{X} = \mathbf{X_{C2}} := [\mathbf{I_3}\ \mathbf{I_3}]$ (the 3×6 matrix created by the union of two identity matrices) generates a triliniear piecewise polynomial. It is easy to see, by definition, when $\mathbf{X_{C2}}$ is used in conjunction with Equation **??** on page **??**, the resulting function $M_{\mathbf{X_{C2}}}(\mathbf{x})$ corresponds to the non-centered tensor product extension of the linear B-spline in three dimensions. Likewise, the non-centered cubic tensor product box spline is given by $M_{\mathbf{X_{C4}}}(\mathbf{x})$, where $\mathbf{X_{C4}} := [\mathbf{I_3}\ \mathbf{I_3}\ \mathbf{I_3}\ \mathbf{I_3}]$. A box spline is easily centered by taking $M_{\mathbf{X}}(\mathbf{x} + \mathbf{c_X})$, with $\mathbf{c_X} := \sum_{\xi \in \mathbf{X}} \frac{1}{2}\xi$. We will assume, unless otherwise stated, that $M_{\mathbf{X}}$ defines the centered box spline generated by $\mathbf{X}$.

2) BCC Lattice: On the BCC lattice, a choice of

$$\mathbf{X_B} := \begin{bmatrix} -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix}$$

generates a piecewise linear polynomial function [**?**]. The support of this box spline is a rhombic dodecahedron, which incidentally contains the set of first nearest neighbors to the Voronoi region of the lattice at the origin, and is a natural fit for the lattice. The quintic box spline is given by the matrix $\mathbf{X_{B2}} := [\mathbf{X_B}\ \mathbf{X_B}]$, or may also be seen as the convolution of the linear box spline with itself. However, there exist both an efficient piecewise polynomial representation [**?**], and an efficient implementation [**?**] of these piecewise quintic polynomial functions.

3) Shifted Basis Functions: We also consider shifted box splines $M_{\mathbf{X}}^{\xi_i}(\mathbf{x}) := M_{\mathbf{X}}(\mathbf{x} - \frac{1}{2}\xi_i)$, where ξ_i is any direction vector in $\mathbf{X}$. In the univariate case with linear B-splines, it has been shown that introducing a shift can offer an improvement when representing scalar data [**?**]. More generally and relevant to our purposes, they have also been shown to reduce error when approximating the gradient of scalar fields [**?**].

D. Shift Invariant Spaces

As our target reconstruction spaces, we consider the shift-invariant spaces defined by

$$\mathbb{V}(\varphi, h\mathbf{L}) := \left\{ g(\mathbf{x}) := \sum_{\mathbf{m} \in \mathbb{Z}^3} c[\mathbf{m}]\varphi(\frac{\mathbf{x}}{h} - \mathbf{L}\mathbf{m}) : c \in \ell_2 \right\},$$

where φ is the generating function for the space, h is a scaling parameter that controls the sampling rate or granularity, and $\mathbf{L}$ is the generating matrix for a given sampling lattice. Representing a function f in a shift invariant space boils down to finding an appropriate coefficient $c[\mathbf{m}]$ vector that brings g as close as possible to the true underlying function f.

If the generating function that spans the space satisfies the *Strang-Fix* conditions of order k, and f belongs to the Sobolev space of at least order k, then it can be shown that the minimum approximation error (in the least-squares sense) decays asymptotically as $O(h^k)$, and the space is said to have approximation order k. We often only have access to f through a discrete amount of samples, $\mathbf{f} := [f_1\ f_2\ \ldots\ f_M]^T$ located at sample points $S := \{\mathbf{x}_1, \mathbf{x}_2 \ldots \mathbf{x}_M\}$ respectively. When this is the case, the minimum-error approximation is unattainable. However, interpolative and quasi-interpolative methods [**?**], [**?**], which can be seen as oblique projections onto $\mathbb{V}(\varphi, h\mathbf{L})$, maintain the same order of accuracy.

For sufficiently smooth functions f that are uniformly sampled and properly prefiltered, the trilinear and linear reconstruction filters (Cartesian and BCC respectively) assure second order convergence, while the tricubic and Quintic assure fourth order. Another nicety of these spaces is that data processing techniques — derivative approximation for example — can often be represented by a convolution of the coefficient vector with either an *Infinite Impulse Response* (IIR) or *Finite Impulse Response* (FIR) filter.

Typically, as memory is a limited quantity, we are only interested in a finite number of points on a lattice. We denote the set of such points as $\{\mathbf{s}_1, \mathbf{s}_2, \ldots\ \mathbf{s}_N\} \subset \mathbf{L}\mathbb{Z}^3$. Corresponding to each lattice site is a coefficient c_k; we collect all such coefficients in the column vector $\mathbf{c} := [c_1\ c_2\ \ldots\ c_N]^T$. We also use the short hand notation $\varphi_k(\mathbf{x}) := \varphi(\frac{\mathbf{x}}{h} - \mathbf{s}_k)$; from this perspective, our function space is now

$$\left\{ \sum_{k=0}^{N} c_k \varphi_k(\mathbf{x}) \right\} \subset \mathbb{V}(\varphi, h\mathbf{L}). \tag{4}$$

Further, we denote $\mathbf{P}$ to be the $M \times N$ matrix whose elements are given by sampling the basis functions anchored at each lattice site, or $\mathbf{P}_{i,j} := \varphi_j(\mathbf{x}_i)$. Clearly, if our data are sampled at the lattice sites ($\mathbf{x}_i = \mathbf{s}_i$), then the problem of finding an interpolative scheme reduces to finding $\mathbf{c}$ such that $\mathbf{Pc} = \mathbf{f}$, which only involves inverting the square matrix $\mathbf{P}$. However, our sample locations are almost certain to not coincide with lattice sites, and we must consider alternative schemes for finding $\mathbf{c}$.

E. Irregular Sampling

In the work by Kazdhan et al. [**?**] the range data are splatted via a linear filter onto a regular Cartesian grid. This tends to over-smooth the resulting reconstruction, as it allows the surface to deviate from the original sample locations. Ideally, we would like to find a function that not only interpolates the input data, but provides the representation with the smallest possible "shell" about the initial point set (i.e., the fewest non zero coefficients) while still interpolating the data and eliciting a unique solution.

Thus, we recast the problem of finding a representation of g within our function space as an optimization problem. There are similar works that impose variational constraints on random point samples for images [**?**], as well as multi-resolution techniques for volumetric data [**?**] and an extension to box spline spaces [**?**]. In the same vein, we seek a function $g \in \mathbb{V}(\varphi, h\mathbf{L})$ that attempts to interpolate the input data, and minimize a functional E. Specifically, we seek a g that minimizes

$$\sum_{i=1}^{M} (g(\mathbf{x}_i) - f_i)^2 + E(g). \tag{5}$$

It is important to introduce a bit of notation before proceeding, $\mathbf{D}_{\mathbf{v}} f := \mathbf{v} \cdot \nabla f$ defines the directional derivative of f in the direction $\mathbf{v}$. The Beppo-Levi inner product of order n is defined as $\langle f, g \rangle_{BL^n} := \sum_{|\mathbf{p}|=n} (n!/\mathbf{p}!) \langle \mathbf{D}^{\mathbf{p}} f, \mathbf{D}^{\mathbf{p}} g \rangle$, where $\mathbf{p}$ is an integer vector such that $|\mathbf{p}| := p_1 + p_2 + p_3$. We also use the short hand $\mathbf{D}^{\mathbf{p}} f := \partial^{|\mathbf{p}|} f / (\partial^{p_1} x_1 \partial^{p_2} x_2 \partial^{p_3} x_3)$. The second order Beppo-Levi norm of f is defined as $||f||^2_{BL^2} := \langle f, f \rangle_{BL^2}$.

The second order Beppo-Levi norm can be thought to measure the "smoothness" of a function. However, constraining the smoothness of our approximation alone is not sufficient to elicit the desired solution in our case. We propose that the smoothed normal field should also minimize the inner product of the function with itself. This has the physical interpretation of forcing the solution to be close to zero everywhere except near the input points. Our proposed cost functional is then $E(g) := \lambda_1 ||g|| + \lambda_2 ||g||_{BL^2}$. Here, λ_1 controls the amount to which the function is penalized for large coefficient values. When combined with the interpolation constraint, this imposes the behavior of forcing the function to be zero away from the surface. The parameter λ_2 controls the smoothness of the resulting normal field.

Finding g that minimizes equation (**??**) involves rewriting the minimization problem in terms of $\mathbf{c}$ [**?**]. Following some straightforward algebraic manipulations (see [**?**] for details), the coefficient vector can be obtained by taking $\mathbf{c} = (\mathbf{P}^T\mathbf{P} + \lambda_1\mathbf{G} + \lambda_2\mathbf{S})^{-1}\mathbf{P}^T\mathbf{f}$, where $G_{i,j} := \langle \varphi_i, \varphi_j \rangle$ and $S_{i,j} := \langle \varphi_i, \varphi_j \rangle_{BL^2}$. Since φ is known in closed form, the entries of both $\mathbf{G}$ and $\mathbf{S}$ can be evaluated analytically. The above vector equation can be efficiently solved using the Conjugate Gradient method.

III. Conclusion

The conclusion goes here.

Appendix A
Proof of the First Zonklar Equation

Appendix one text goes here.

Appendix B

Appendix two text goes here.

Acknowledgment

The authors would like to thank...

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.