Neural Splines: Fitting 3D Surfaces with Infinitely-Wide Neural Networks

Francis Williams[1] Matthew Trager[1, 2*] Joan Bruna[1] Denis Zorin[1]

[1]New York University, [2]Amazon

francis.williams@nyu.edu, mtrager@cims.nyu.edu, bruna@cims.nyu.edu, dzorin@cs.nyu.edu

Abstract

We present Neural Splines, a technique for 3D surface reconstruction that is based on random feature kernels arising from infinitely-wide shallow ReLU networks. Our method achieves state-of-the-art results, outperforming recent neural network-based techniques and widely used Poisson Surface Reconstruction (which, as we demonstrate, can also be viewed as a type of kernel method). Because our approach is based on a simple kernel formulation, it is easy to analyze and can be accelerated by general techniques designed for kernel-based learning. We provide explicit analytical expressions for our kernel and argue that our formulation can be seen as a generalization of cubic spline interpolation to higher dimensions. In particular, the RKHS norm associated with Neural Splines biases toward smooth interpolants.

1. Introduction

Estimating a 3D surface from a scattered point cloud is a classical and important problem in computer vision and computer graphics. In this task, the input is a set of 3D points sampled from an unknown surface and, possibly, normals of the surface at those points. The goal is to estimate a representation of the complete surface from which the input samples were obtained, for example, a polygonal mesh or an implicit function. This problem is challenging in practice: It is inherently ill-posed, since an infinite number of surfaces can interpolate the data. Furthermore, the input 3D points are often incomplete and noisy, as they are acquired from range sensors such as LIDAR, structured light, and laser scans. Ideally, the recovered surface should not interpolate noise but preserve key features and surface details.

Many early surface reconstruction techniques consider a kernel formulation of the surface reconstruction problem, using translation-invariant kernels such as biharmonic RBFs [9], Gaussian kernels [22], or compactly supported RBFs [34]. Currently, the most widely used method for surface reconstruction is *Screened Poisson Surface Reconstruction* [31], which solves a variant of the Poisson equation to find an implicit function whose zero-level set produces an approximating surface. We show in this paper that this method can also be viewed as a kernel method for a particular choice of kernel.

More recently, many papers have used neural networks to represent an implicit function or a local chart in a manifold atlas as a means of reconstructing a surface [44, 19, 26, 3, 24, 40, 42]. These methods can be integrated into a data-driven learning pipeline or directly applied in the so called "overfitting" regime, where a massively overparameterized (*i.e.*, more parameters than input points) neural network is fitted to a single input point cloud as a functional

*Work done prior to joining Amazon

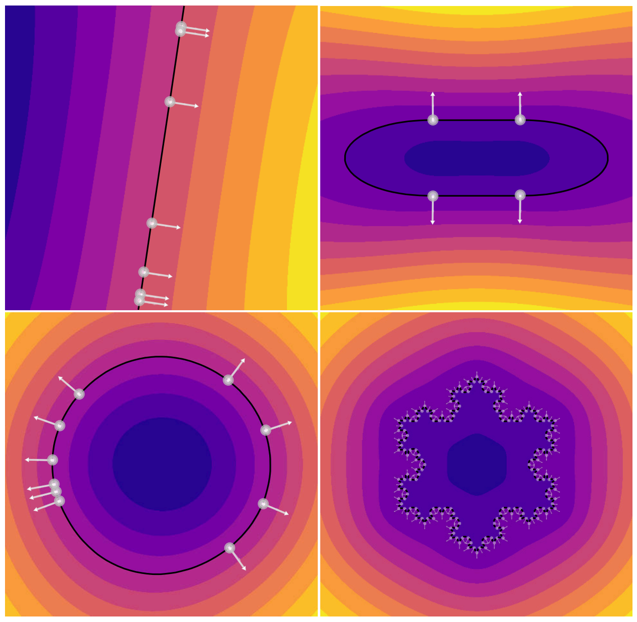


Figure 1. Neural Splines use points (the white dots) and normals (the white arrows) as input and estimate a scalar function whose zero level set (the black lines) corresponds to the reconstructed surface and whose gradient agrees with the normals.

representation for a surface. Empirical evidence has shown that these methods enjoy some form of "implicit regularization" that biases the recovered surface towards smoothness. Moreover, employing early stopping in gradient descent can prevent these approaches from interpolating noise.

Under certain parameter initializations, infinitely-wide neural networks *de facto* behave as kernel machines [30, 14], defining Reproducing Kernel Hilbert Spaces (RKHS), whose kernel is obtained by linearizing the neural network mapping around its initialization. While the kernel regime simplifies non-linear neural network learning into a convex program and provides a simple explanation for the successful optimization of overparameterized models, it cannot explain the good generalization properties observed for high-dimensional problems due to the inability of the RKHS to approximate non-smooth functions [4]. However, the situation for low-dimensional problems such as surface reconstruction is entirely different, and RKHS can provide powerful characterizations of regularity. In this context, [45] shows that in the univariate case the RKHS norm associated with a wide ReLU network is a weighted curvature, and leads to cubic spline interpolants. In higher dimensions, similar (albeit more complex) characterizations of the RKHS norm exist [35] (see also Proposition 2). In order to assess the benefits of neural networks on such low-dimensional problems, it is thus important to first understand their linearized counterparts, given by their associated kernel machines.

In this work, we demonstrate that in fact kernels arising from shallow ReLU networks are extremely competitive for 3D surface reconstruction, achieving state-of-the-art results: outperforming classical methods as well as non-linear methods based on far more complex neural network optimization.

Kernels provide many advantages over neural networks in our context: *(i)* They are well understood theoretically. *(ii)* Kernel regression boils down to solving a linear system, and avoids gradient descent that suffers from slow convergence. *(iii)* Kernel-based interpolants are represented using a number of parameters that is linear in the size of the input, whereas overparameterized neural networks require many more parameters than points. *(iv)* The inductive bias of kernel methods can be characterized explicitly via the RKHS norm (Section 3.3). *(v)* Kernel methods lend to scalable and efficient implementations (Section 3.5). We provide explicit expressions for two kinds of infinite-width ReLU kernels, their derivatives, and their corresponding RKHS norms. We further argue that these kernels can be viewed as a multi-dimensional generalization of cubic spline interpolation in 1D. Moreover, we show that Poisson Surface Reconstruction can itself be viewed as a kernel method and give an expression for its RKHS norm, suggesting that kernels are a broad framework which enable the rigorous understanding both traditional and modern surface reconstruction techniques.

1.1. Additional Related Work

Methods for 3D surface reconstruction can mostly be divided by their choice of surface representation: These are *(i)* the zero level set of a volumetric scalar function [31, 3, 24, 33, 36, 10, 40, 42], *(ii)* the fixed point of a projection operator onto locally fitted patches [1, 25, 29], *(iii)* a mesh connecting the input points [28], *(iv)* a collection of local parametric maps [44, 26, 17, 19], or *(v)* the union of parametric shapes [21, 20, 16, 13, 43]. For a recent up-to-date survey of surface reconstruction techniques, we refer the reader to [6].

The "random feature" kernels used this paper arise from training the top-layer weights in two-layer networks of infinite width and were described in [15, 32]. More recently, a lot of work has focused on a different kernel, known as the "neural tangent kernel" [30, 14], that linearly approximates the training of *both* layers of a neural network. We chose to use random feature kernels since, when the input dimension is small, typical initializations in neural networks lead to mainly training the top layer weights. More broadly, the function spaces associated with shallow neural networks were studied in [4, 7, 35, 39, 45].

2. Neural Spline Formulation

We formulate the problem of surface reconstruction as the task of finding a scalar function $f : \mathbb{R}^3 \to \mathbb{R}$ whose zero level set $S = \{p\colon f(p) = 0\} \subset \mathbb{R}^3$ is the estimated surface (see Figure 1). In its most general form in arbitrary dimensions, we write our problem as follows.

We assume we are given a set of s input points $\mathcal{X} = \{x_j\}_{j=1}^{s} \subset \mathbb{R}^d$, function values $\mathcal{Y} = \{y_j\}_{j=1}^{s} \subset \mathbb{R}$, and normals $\mathcal{N} = \{n_j\}_{j=1}^{s} \subset \mathbb{R}^d$. Our goal is to optimize an objective function of the form:

$$\min_{\theta \in \mathbb{R}^{d_\theta}} L(\theta), \text{ with}$$

$$L(\theta) = \frac{1}{2}\sum_{j=1}^{s} |f(x_j;\theta) - y_j|^2 + \|\nabla_x f(x_j;\theta) - n_j\|^2. \quad (1)$$

Here $f(x;\theta)$ is a family of functions $\mathbb{R}^d \to \mathbb{R}$ parameterized by $\theta \in \mathbb{R}^{d_\theta}$. For surface reconstruction, we have $d = 3$ and $y_j = 0$, since the reconstructed surface is given by the zero level set.

We begin by introducing the Neural Spline family of functions through the lens of finite-width shallow neural networks in Section 2.1, and turn to the infinite-width limit formulation in Section 2.2. While our approach is simple and our derivations straightforward, we differ slightly from a standard kernel regression setting since we employ a multi-dimensional kernel that includes the gradient of the fitted function. This formulation allows us to fit points and normals simultaneously.

2.1. Finite-Width Kernel

We first assume that the model $f(x;\theta)$ is a two-layer ReLU neural network with m neurons, but we keep the bottom layer weights *fixed* from initialization:

$$f(x;\theta) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} c_i [a_i^T x + b_i]_+, \ \theta = (c_1, \ldots, c_m) \in \mathbb{R}^m \tag{2}$$

Here we write $[z]_+ = \max(z, 0)$ for the ReLU function. The bottom-layer weights $(a_{i1}, \ldots, a_{id}, b_i) \in \mathbb{R}^{d+1}$ are fixed randomly according to some specified distribution.

Remark 1. *In typical initialization schemes for neural networks (*e.g.*, Kaiming He initalization [27]), the weights of each layer are initialized with a variance that is inversely proportional to the number of inputs of that layer. Our choice of fixing (a_i, b_i) is motivated by the fact that using a standard two-layer network, for a given target model and as $m \to \infty$, only the top layer weights tend to vary throughout training [45].*

Our model (2) is linear in the parameters θ, so our objective function $L(\theta)$ is *convex*. However, since we assume $m \gg s$, we expect to have an infinite set of global minimizers (an affine subspace in $\mathbb{R}^{d_\theta}$). Our goal is to find the minimizer with smallest parameter norm:

$$\theta^* = \operatorname{argmin}\{\|\theta\|^2 \colon \theta \in \mathbb{R}^{d_\theta} \ L(\theta) = 0\}.$$

This minimizer is given explicitly by $\theta^* = W^\dagger \delta$ where $W^\dagger$ denotes the Moore-Penrose pseudo-inverse of W and

$$W = \frac{1}{\sqrt{m}} \begin{bmatrix} [a_j^T x_i + b_j]_+ \\ \mathbf{1}[a_j^T x_i + b_j] a_j \end{bmatrix}_{\substack{i=1,\ldots,s \\ j=1,\ldots,m}} \qquad \delta = \begin{bmatrix} y_i \\ n_i \end{bmatrix}_{i=1,\ldots,s} \tag{3}$$

so that $W \in \mathbb{R}^{(s+ds)\times m}$ and $\delta \in \mathbb{R}^{s+ds}$. If W has full rank, then $W^\dagger = W^T(WW^T)^{-1}$ and we can equivalently look for $z^* \in \mathbb{R}^{(s+ds)}$ such that $K_{\mathcal{X}} z^* = W\theta^* = \delta$ where $K_{\mathcal{X}} = WW^T \in \mathbb{R}^{(s+ds)\times(s+ds)}$. Note that $K_{\mathcal{X}}$ can be viewed as a Gram matrix associated with the multi-dimensional kernel $K(x, x') \in \mathbb{R}^{(d+1)\times(d+1)}$ described by

$$\frac{1}{m}\sum_{i=1}^{m} \begin{bmatrix} \varphi_{w_i}(x)\varphi_{w_i}(x') & \varphi_{w_i}(x)\nabla_{x'}\varphi_{w_i}(x')^T \\ \nabla_x\varphi_{w_i}(x)\varphi_{w_i}(x') & \nabla_x\varphi_{w_i}(x)\nabla_{x'}\varphi_{w_i}(x')^T \end{bmatrix}, \tag{4}$$

where for compactness we used $w_i = (a_i, b_i)$, $\varphi_{w_i}(x) = [a_i x + b_i]_+$ and $\nabla_x \varphi_{w_i}(x) = \mathbf{1}[a_i x + b_i]_+ a_i$.

2.2. Infinite-Width Kernel

As the number of neurons m tends to infinity, the kernel (4) converges to $K_\infty(x, x')$ defined by

$$\mathbb{E}_{w\sim\mathcal{D}} \begin{bmatrix} \varphi_w(x)\varphi_w(x') & \varphi_w(x)\nabla_{x'}\varphi_w(x')^T \\ \nabla_x\varphi_w(x)\varphi_w(x') & \nabla_x\varphi_w(x)\nabla_{x'}\varphi_w(x')^T \end{bmatrix}, \tag{5}$$

where $\mathcal{D}$ is the chosen distribution of bottom layer weights $w = (a, b)$. We use this kernel to characterize the interpolant in the infinite-width limit. For this, we simply replace K in (4) with K_∞ and solve the linear system the described in the previous section. More concretely (and rearranging terms), our goal is to recover $\alpha_1^*, \ldots, \alpha_s^* \in \mathbb{R}^{d+1}$ such that

$$\sum_{i=1}^{s} K_\infty(x_j, x_i)\alpha_i^* = \begin{bmatrix} y_j \\ n_j \end{bmatrix} \in \mathbb{R}^{d+1}, \quad j = 1, \ldots, s. \tag{6}$$

Our interpolant f^* is such that $\begin{bmatrix} f^*(x) \\ \nabla f^*(x) \end{bmatrix} = K_\infty(x, x_i)\alpha_i^*$ for all $x \in \mathbb{R}^d$. It can also be viewed as the solution to

$$\begin{aligned} &\underset{f\in\mathcal{H}}{\text{minimize}} \ \|f\|_{\mathcal{H}}, \\ &\text{subject to } f(x_i) = 0 \text{ and } \nabla f(x_i) = n_i \end{aligned} \tag{7}$$

where $\mathcal{H}$ is the RKHS corresponding to the one-dimensional kernel $\mathbb{E}_w \varphi_w(x)\varphi_w(x') = \mathbb{E}_{(a,b)}[a^T x + b]_+[a^T x' + b]_+$. In Section 3.3 we give an expression for the norm $\|\cdot\|_{\mathcal{H}}$ that defines the inductive bias of solutions. In Appendix B, we provide analytic expressions for the kernel (5) for two natural distributions over the weights (a, b):

$$\text{Uniform:} \qquad a \sim \mathcal{U}[\mathbb{S}^{d-1}], \ b \sim \mathcal{U}[-k, k], \tag{8}$$

$$\text{Gaussian:} \qquad (a, b) \sim \mathcal{N}(0, Id_d). \tag{9}$$

The uniform distribution (8) corresponds to the default initialization of linear layers in PyTorch and, as we argue in Section 3.1, it leads to a direct generalization of cubic spline interpolation. However, the Gaussian initialization actually leads to simpler analytical expressions for K_∞ and produces almost the same results. We refer to Appendix B for a discussion and comparison of the two distributions.

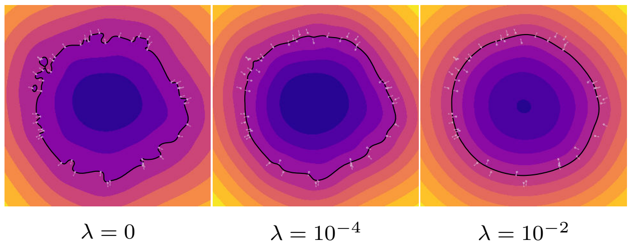
3. Discussion

3.1. Connection to Cubic Splines in 1D

When $d = 1$, the uniform initialization (8) is such that $a \sim \mathcal{U}(\mathbb{S}^0) = \mathcal{U}(\{-1, 1\})$ and, assuming $-k \le x \le x' \le k$, the top-left element in $K_\infty(x, x')$ is given by

$$\begin{aligned} &K_{\text{spline}}(x, x') = \\ &= \frac{1}{2}\int_{-k}^{k} [x + b]_+[x' + b]_+ db + \frac{1}{2}\int_{-k}^{k} [-x + b]_+[-x' + b]_+ db \\ &= \frac{1}{12}\,(3\,x' - x + 2\,k)(x + k)^2 - \frac{1}{12}\,(3\,x - x' - 2\,k)\,(x' - k)^2. \end{aligned} \tag{10}$$

The expression assuming $-k \le x' < x \le k$ is obtained by swapping x and x'. For fixed x, the map $K_{\text{spline}}(x, \cdot)$ is piecewise cubic and twice continuously differentiable (C^2). This implies that kernel regression with (10) yields *cubic spline interpolation*. Applying the Neural Spline objective (1) with derivative constraints at the samples in $d = 1$ also yields a piecewice cubic interpolant, although this curve is in general only C^1. For other distributions of a and b, the kernel is no longer cubic, but the norm in the RKHS is a weighed norm of curvature (see [45] for details). In this sense, our approach with the initialization (8) can be viewed as a multi-dimensional version of spline interpolation.



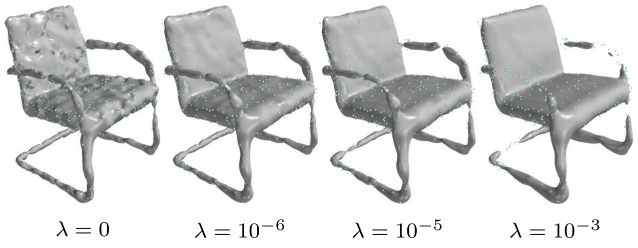


Figure 2. *The effect of adding a regularization in 2D (Top) and 3D (Bottom).* Fitting the points and normals with no regularization yields a surface which perfectly interpolates the noisy input data (left images). Increasing λ leads to smoother solutions which do not interpolate noise in the input. If λ is too large, the output loses geometric details (*e.g.*, the arms and base of the chair in the bottom right image).

3.2. Regularization and Robustness to Noise

To deal with noisy data, we can optionally add a simple regularizer term to our formulation that corresponds to penalizing the RKHS norm of the interpolant ("kernel ridge regression"). Concretely, we replace (6) with

$$\sum_{i=1}^{s} K_\infty(x_j, x_i)\alpha_i^* + \delta_{ij}\lambda I d_{d+1} = \begin{bmatrix} y_j \\ n_j \end{bmatrix} \in \mathbb{R}^{d+1}, \quad (11)$$

for $j = 1, \dots, s$. The regularizer term affects the spectrum of the Gram matrix $K_\infty(x_j, x_i)$ by smoothing its smallest eigenvalues, with a similar effect to early stopping in gradient descent (see *e.g.* [2]). Figure 2 shows examples of applying this regularizer on 2D and 3D problems.

3.3. Inductive Bias of Interpolants

Our interpolant f^* belongs to the Hilbert space given by

$$\begin{aligned} &\mathcal{H} = \{f(x) = \textstyle\int c(w)\varphi_w(x)d\tau(w) \mid \|f\|_{\mathcal{H}} < +\infty\}, \text{ with} \\ &\|f\|^2_{\mathcal{H}} = \inf\{\textstyle\int c(w)^2 d\tau(w) \mid f(x) = \int c(w)\varphi_w(x)d\tau(w)\} \end{aligned} \quad (12)$$

where $\tau(w)$ is a measure over the weights $w = (a, b)$ (*e.g.*, (8) or (9)), and $c(a, b)$ can be viewed as a continuous analog of the outer-layer weights c_i of the finite network given by (2). The inductive bias of Neural Splines is thus determined by the RKHS norm in (12) since our method outputs the interpolating function which minimizes that norm. As noted in [35], if $f(x) = \int c(w)\varphi_w(x)d\tau(w)$ and $d\tau(w) = d\tau_a(a)d\tau_b(b)$ then, by differentiating twice, we note that the Laplacian of f is given by

$$\Delta f(x) = \int_{\{ax+b=0\}} c(a, b)d\tau_a(a). \quad (13)$$

By comparing (12) and (13), we see that the RKHS norm and the Laplacian of f are closely related. More precisely, the Laplacian $\Delta f(x)$ is the *Dual Radon Transform* of $c(a, b)$. Under certain assumptions, (13) can be inverted, yielding an explicit expression for c in terms of Δf. Intuitively, this shows that bounding the RKHS norm imposes a constraint on the Laplacian of f, and thus encourages f to be smooth. We report the following statement from [35] and we refer to Appendix D for more details.

Proposition 2. *[35] Let $f(x) = \int c(a, b)[ax + b]_+ d\tau(a, b)$, and the constant $\gamma_d = \frac{1}{2(2\pi)^{d-1}}$. If we assume that $c(a, b) = c(-a, -b)$ holds, then*

$$c(a, b) = \gamma_d \frac{\mathcal{R}\{(-\Delta)^{\frac{d+1}{2}} f(x)\}(a, b)}{\tau(a, b)},$$

where $\mathcal{R}\{f\}(a, b)$ is the Radon Transform of f.

3.4. Poisson Surface Reconstruction as a Kernel

We cast Screened Poisson Surface Reconstruction [31] in kernel form to facilitate comparisons. In its simplest form, Poisson reconstruction, extracts the level set of a smoothed indicator function determined as the solution of

$$-\Delta f = \nabla \cdot V,$$

where V is a vector field obtained from normals n_i at samples x_i, and we use f to denote the (smoothed) indicator function as it plays the same role as f in (1). The equation above is closely related to (1): specifically, it is the equation for the minimizer of $\int_{\mathbb{R}^3} \|\nabla_x f(x) - V\|^2 dx$ and the second term in (1) can be viewed as a approximation of this term by sampling at x_i. The screened form of Poisson reconstruction effectively adds the first term with $y_i = 0$, as the indicator function at points of interest is supposed to be zero. For the Poisson equation, the solution can be explicitly written as an integral

$$f(x) = \int_{\mathbb{R}^3} \frac{\nabla_z \cdot V(z)dz}{|x - z|}.$$

The vector field V is obtained by interpolating the normals using a fixed-grid spline basis and barycentric coordinates of the sample points with respect to the grid cell containing it. This is equivalent to using a non-translation invariant non-symmetric locally-supported kernel $K_B(z, x)$:

$$V(z) = \sum_i K_B(z, x_i)n_i\ .$$

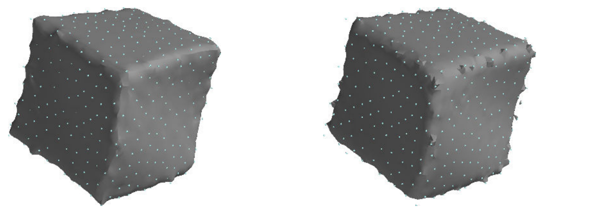


Figure 3. *Left:* Poisson reconstruction using the approximate Kernel (15). *Right:* Screened Poisson Reconstruction of the same input.

Lemma 3. *Let $\{c_j \in \mathbb{R}^3\}_{j=1}^g$ be a set of points arranged on a regular grid, $B_1(x-z)$ be the trilinear basis function, and $B_n(x-z)$ be a degree-n spline basis function (See Appendix C for equations for B_1 and B_n). The kernel corresponding to Poisson Surface Reconstruction is*

$$K_{\text{PR}}(x, x') = \int_{\mathbb{R}^3} \frac{K_B(z, x')dz}{|x-z|}, \tag{14}$$

where $K_B(z, x) = \sum_j B_1(x - c_j)B_n(z - c_j)$.

To study the qualitative properties of this kernel, we replace $K_B(z,x)$ with a radial kernel $B_n^1(|z-x|)$ (see Appendix C) which has qualitatively similar behavior (see Figure 3). Since both B_n^1 and the Laplace kernel $\frac{1}{|x-z|}$ are radial functions, their convolution is also radial, yielding a translation-invariant radial approximation $K_{\text{PR}}^{\text{approx}}$ of K_{PR}:

$$K_{\text{PR}}^{\text{approx}}(x, x') = \int_{\mathbb{R}^3} \frac{B_n^1(|z-x'|)dz}{|x-z|}. \tag{15}$$

Lemma 4. *The RKHS norm of the corresponding to the approximate Poisson kernel $K_{\text{PR}}^{\text{approx}}$ is*

$$\|f\|_{\mathcal{H}} = \int \frac{|\mathcal{F}[f]|^2}{\mathcal{F}[K_{PR}^{\text{approx}}]} d\omega \tag{16}$$

where $\mathcal{F}[\cdot]$ is the Fourier transform.

We discuss the kernel formulation of Poisson Reconstruction in more detail in Appendix C.

3.5. Fast and Scalable Implementation

We provide a fast and scalable implementation of Neural Spline kernels based on FALKON [38], a recently-proposed solver for kernel-ridge-regression which runs in parallel on the GPU. While naïve kernel ridge-regression with N points requires solving and storing an $N \times N$ dense linear system, FALKON uses conjugate gradient descent requiring only $\mathcal{O}(N)$ storage, and $\sqrt{N}$ convergence (though in practice we find that even for very large inputs, we converge in fewer than 10 iterations). To speed up convergence, FALKON can optionally store an $M \times M$ preconditioner matrix in CPU memory (where $M \ll N$, see paragrph below). To maximize performance and reduce memory overhead, we rely on KeOps [12] to evaluate kernel matrix-vector products symbolically on the GPU, which means our implementation uses only a *small constant* amount GPU memory and can be readily used on commodity hardware. Section 4.4 compares the performance of our implementation against other state of the art surface reconstruction techniques. We note that in principle, low-dimensional kernel methods can be accelerated using fast multipole-based approaches [23] (in particular, in the context of 3D surface reconstruction this was used in [9]); this yields optimal $O(N)$ time complexity, for dense matrix-vector multiplication.

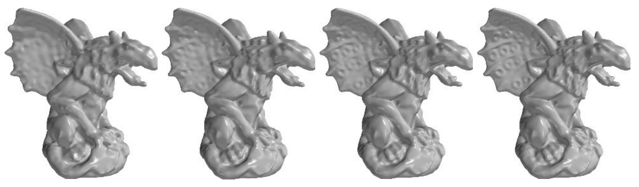


Figure 4. Fitting a range scan with 100k points using varying numbers of Nyström samples. Larger numbers of Nyström samples will lead to reconstructions which preserve finer details (*e.g.* the bumps on the wing of the gargoyle). In this case, 15% of the input samples recovers approximately the same level of detail as 75%.

Nyström Subsampling Full kernel ridge regression predicts a function which is supported on every input point x_i as in (6), requiring N coefficients to store the resulting function. We rely on Nyström sampling [18] to instead produce a kernel function which is supported on a small M-sized subset of the input points (while still minimizing a loss on all the points). This is equivalent to approximating the kernel matrix with a low rank linear system. To choose Nyström samples, we leverage the geometric nature of our problem and select these by downsampling the input point cloud to have a blue-noise distribution using Bridson's algorithm [8]. We demonstrate the effect of varying the number of Nyström samples qualitatively in Figure 4.

4. Experiments and Results

We now demonstrate the effectiveness of Neural Splines on the task of surface reconstruction. For all the experiments in this section, we used the analytical form of the kernel (5) with Gaussian initialization (8). Appendix A.7 compares the uniform (8) and Gaussian (9) kernels, showing almost no measurable difference in the reconstructions produced by either. We compare the empirical and analytical kernels in detail in Appendix A.6. An implementation of Neural Splines is available at `https://github.com/fwilliams/neural-splines`.

	Method	mean	median	std
IoU	Screened Poisson [31]	0.6340	0.6728	0.1577
	IGR [24]	0.8213	0.8566	0.1461
	SIREN [40]	0.7997	0.8248	0.1203
	Fourier Feat. Nets [42]	0.8143	0.8321	0.1047
	Biharmonic RBF [9]	0.8247	0.8642	0.1350
	SVR [22]	0.7625	0.7819	0.1300
	Ours	**0.9167**	**0.9438**	0.0985
Chamfer	Screened Poisson [31]	2.22e-4	1.70e-4	1.76e-4
	IGR [24]	6.66e-4	1.07e-4	4.69e-3
	SIREN [40]	1.01e-4	8.62e-5	5.40e-5
	Fourier Feat. Nets [42]	9.19e-5	8.68e-5	3.47e-5
	Biharmonic RBF [9]	1.11e-4	8.97e-5	7.06e-5
	SVR [22]	1.14e-4	1.04e-4	5.99e-5
	Ours	**5.32e-5**	**4.07e-5**	3.53e-5

Table 1. Quantitative results on ShapeNet: Our method quantitatively outperforms state of the art neural network based methods and classical methods by a large marchin in both IoU and Chamfer distances.

4.1. Sparse Reconstruction on Shapenet

We performed a quantitative evaluation on a subset of the Shapenet dataset [11] to demonstrate that the inductive bias of Neural Splines is particularly effective for reconstructing surfaces from sparse points. We chose 1024 random points and normals sampled from the surface of 20 shapes per category across 13 categories (totalling 260 shapes). Using this dataset, we compared our method against Implicit Geometric Regularization (IGR) [24], SIREN [40], Fourier Feature Networks [42], Biharmonic RBF (Biharmonic) [9], SVM surface modelling (SVR) [22], and Screened Poisson Surface Reconstruction (Poisson) [31]. The first three techniques are modern neural network based methods, while the latter three techniques are classical methods based on kernels or solving a PDE. As criteria for the benchmark, we consider the Intersection over Union (IoU) and Chamfer Distance between the reconstructed shapes and the ground truth shapes. The former metric captures the accuracy of the predicted occupancy function, while the latter metric captures the accuracy of the predicted surface. Under both metrics, our method outperforms all other methods by a large margin. Table 1 reports quantitative results for the experiment and Figure 5 shows visual results on a few models. We report per-category results in Appendix A.4 and show many more figures in Appendix A.3.

Parameter Selection Several of the above methods have parameters which can be tuned to increase performance. To ensure a fair comparison, we ran parameters sweeps for each Shapenet model for these methods, reporting the maximimum of each metric under consideration (see Appendix A.1 for a detailed description). For our method we *did not* tune parameters. We used no regularization and 1024 as Nyström samples for all models.

		Ground Truth		Scans	
	Method	d_C	d_H	$d_{\vec{C}}$	$d_{\vec{H}}$
Anchor	DGP [44]	0.33	8.82	0.08	2.79
	IGR [24]	**0.22**	4.71	0.12	1.32
	SIREN [40]	0.27	6.18	0.13	1.88
	FFN [42]	0.31	4.49	0.10	0.10
	Ours	**0.22**	**4.65**	0.11	1.11
Daratech	DGP [44]	**0.2**	3.14	0.04	1.89
	IGR [24]	0.25	4.01	0.08	1.59
	SIREN [40]	0.29	4.46	0.12	1.65
	FFN [42]	0.34	5.97	0.10	0.10
	Ours	0.21	4.35	0.08	1.14
DC	DGP [44]	0.18	4.31	0.04	2.53
	IGR [24]	0.17	2.22	0.09	2.61
	SIREN [40]	0.18	2.27	0.09	1.92
	FFN [42]	0.20	2.87	0.10	0.12
	Ours	**0.14**	**1.35**	0.06	2.75
Gargoyle	DGP [44]	0.21	5.98	0.06	3.41
	IGR [24]	**0.16**	3.52	0.06	0.81
	SIREN [40]	0.29	3.90	0.13	1.93
	FFN [42]	0.22	5.04	0.09	0.09
	Ours	**0.16**	**3.20**	0.08	2.75
Lord Quas	DGP [44]	0.14	3.67	0.04	2.03
	IGR [24]	**0.12**	1.17	0.07	0.98
	SIREN [40]	0.13	0.89	0.06	0.96
	FFN [42]	0.35	3.90	0.06	0.06
	Ours	**0.12**	**0.69**	0.05	0.62

Table 2. Quantitative results on the Surface Reconstruction Benchmark: The *Ground Truth* contains the Chamfer (d_C) and Hausdorff (d_H) distances between the reconstruction and ground truth. The *Scans* column contains the one sided Chamfer ($d_{\vec{C}}$) and Hausdorff ($d_{\vec{H}}$) distance between the reconstruction and noisy inputs which measures how much the reconstruction overfits noise in the input.

4.2. Surface Reconstruction Benchmark

We evaluated our method on the Surface Reconstruction Benchmark [5] which consists of simulated noisy range scans (points and normals) taken from 5 shapes with challenging properties such as complex topologies, sharp features, and small surface details. We evaluate our method against Deep Geometric Prior (DGP) [44], Implicit Geometric Regularization (IGR) [24], SIREN [40], and Fourier Feature Networks (FFN) [42]. We remark that DGP establishes itself as superior to a dozen other classical methods on this benchmark and IGR furhter outperforms DGP. As in [24] and [44], Table 2 reports the Hausdorff (d_H) and Chamfer (d_C) distances between the reconstruction and ground-truth. We also report the one sided Hausdorff ($d_{\vec{H}}$) and Chamfer ($d_{\vec{C}}$) distances between the scan and the reconstruction, which measures how much the reconstructions overfits noise in the input. Our reconstructions are quantitatively closer to ground truth on all but one model. Figure 6 shows visual examples of a few models from the benchmark. All models are shown in Appendix A.3. As with the Shapenet benchmark, we did parameter sweeps for those methods which have tunable parameters, choosing the best model for each metric under consideration. See Appendix A.2 for details.

4.3. Large Scale Reconstruction of Full Scenes

Figure 7 shows a full scene consisting of 9 million points reconstructed using our method. This is the same scene used

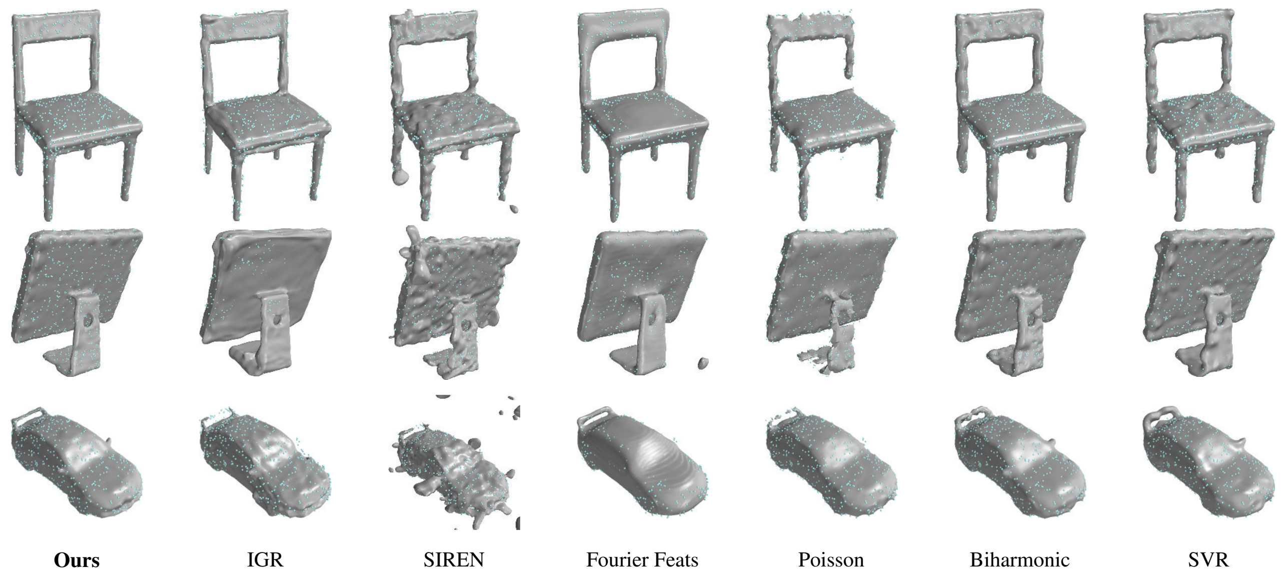


Figure 5. *Comparisons between reconstruction techniques on ShapeNet models.* For techniques requiring parameter sweeps, we show the result with the highest IoU. See Appendix A.3 for more figures from each Shapenet class.

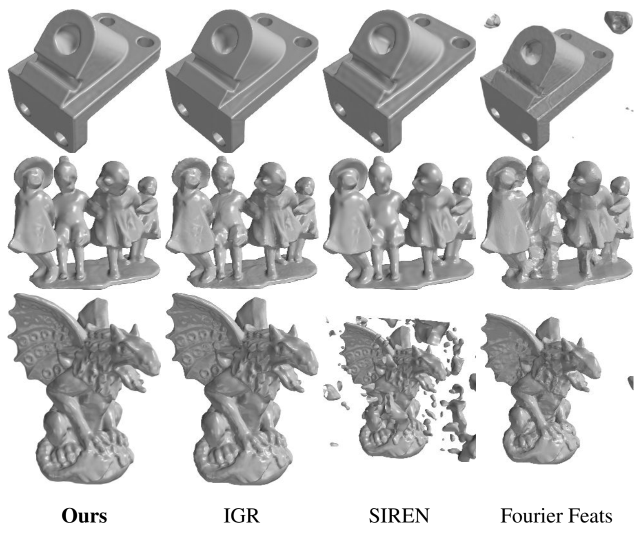


Figure 6. *Comparisons between reconstruction techniques on the Surface Reconstruction Benchmark.* For techniques requiring parameter sweeps, we show the result with the lowest Chamfer Distance. See Appendix A.3 for more figures from each model.

in [40], and contains many thin features that are difficult to reconstruct (*e.g.*, curtains, plant, and lampshade). To generate the input point cloud for this experiment, we densely sampled a mesh extracted from a 2048^3 occupancy grid of the scene. To perform the reconstruction, we subdivided the space into 8x8x8 cells containing between 10k and 500k samples and reconstructed each cell interdependently using up to 15k Nyström samples. The whole process takes 1.5 hours on a machine with an NVIDIA 1080Ti GPU.

	FFN	IGR	SIREN	Poisson	Ours
Runtime (sec)	337.71	1413.94	161.41	1.64	11.91
Max VRAM (MiB)	3711	5093	1607	N.A.	5285

Table 3. Average runtime and GPU memory utilization of various methods on the Surface Reconstion Benchmark [5] models. All Neural Network based methods were run for 5000 iterations.

4.4. Timing and Performance

Table 3 compares the average running time and GPU memory usage of our method and others when reconstructing point clouds from the Surface Reconstruction Benchmark described in Section 4.2. To ensure a fair comparison we ran all neural network based methods for 5000 iterations. We do not report exact CPU memory usage for the methods since it is hard to measure but we remark that, by observation, system memory usage never exceeded 3GiB for any of the methods. Appendix A.5 reports the running times and memory usages for individual models in the benchmark.

4.5. Our Method versus Neural Networks

Implicit Geometric Regularization [24] demonstrates that ReLU networks have a natural inductive bias which makes them good at reconstructing shapes. However, methods based on ReLU networks suffer from slow convergence (see Figure 8). SIREN [40] drastically improves convergence speed by replacing ReLU with sinusoidal activations and using a clever initialization. While SIREN does indeed improve convergence, it performs poorly when samples are sparse, suggesting that their inductive bias is perhaps ill suited for sparse reconstruction tasks (see Figure 5). By projecting input points onto random Fourier features, Fourier Feature Networks (FFNs) [42] present a principled approach

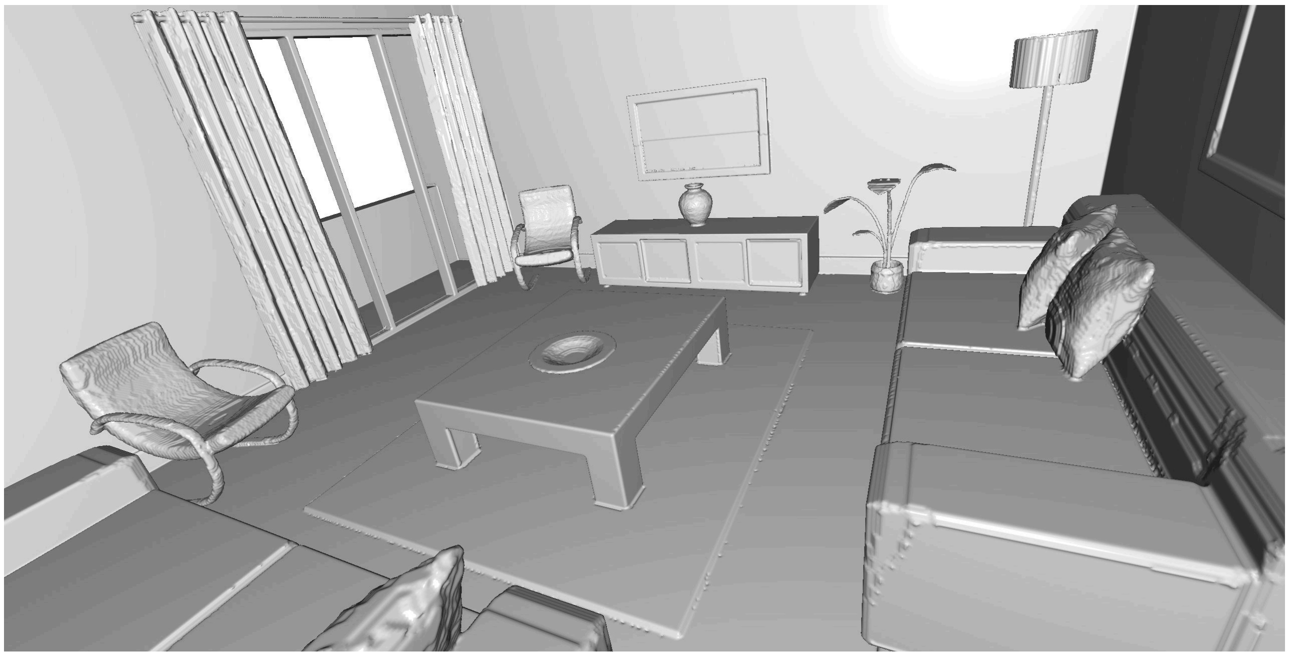


Figure 7. Using our method to reconstruct a full scene consisting of 9 million points.

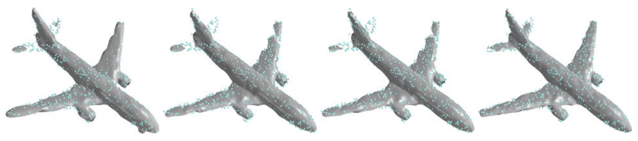


Figure 8. Implicit Geometric Regularization [24] and other methods based on ReLU networks suffer from slow convergence for sharp and thin features (*e.g.*, the tail of the airplane only begins to appear after 100k iterations).

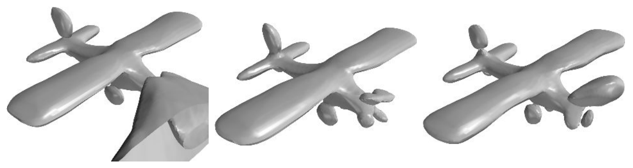


Figure 9. Fourier Feature Networks with Gaussian features [42] are sensitive to the choice of variance (σ) of the Gaussian when data is sparse. The images above show the result of training a network with the same initialization and Fourier features with different initial scales for the Gaussian distribution. Small changes in σ lead to large changes in the final reconstruction.

rooted in Kernel methods to control the inductive bias of the solution as well as convergence speed. While FFNs are capable of producing high quality solutions, they are sensitive to the choice of feature distribution when data is sparse (see Figure 9) and thus require tuning to work well. In contrast, our technique converges in seconds (Section 4.4), has a well suited inductive bias for shape representation (Section 3.3), and requires minimal parameter tuning.

5. Conclusion and Future Work

We have shown that Neural Spline kernels arising from infinitely wide shallow ReLU networks are very effective tools for 3D surface reconstruction, outperforming state-of-the-art methods while being computationally efficient and conceptually simple. In a sense, our work bridges the gap between traditional reconstruction methods and modern neural network based methods by leveraging the deep connection between neural networks and kernels.

We remark that our kernel formulation is fully differentiable. In the future, we hope to integrate Neural Splines into deep learning pipelines and apply them to other 3D tasks such as shape completion and sparse reconstruction. On the theory side, we would like to investigate and compare the approximation properties of different kernels (in particular those arising from infinite width sinusoidal networks) in the context of 3D reconstruction.

Acknowledgements This work is partially supported by the Alfred P. Sloan Foundation, NSF RI-1816753, NSF CAREER CIF 1845360, NSF CHS-1901091, and Samsung Electronics.

References

[1] Anders Adamson and Marc Alexa. Approximating and intersecting surfaces from points. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 230–239. Eurographics Association, 2003.

[2] Alnur Ali, J. Zico Kolter, and Ryan J. Tibshirani. A Continuous-Time View of Early Stopping for Least Squares. *arXiv:1810.10082 [cs, stat]*, Feb. 2019.

[3] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data, 2019.

[4] Francis Bach. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research*, 18(19):1–53, 2017.

[5] Matthew Berger, Joshua A Levine, Luis Gustavo Nonato, Gabriel Taubin, and Claudio T Silva. A benchmark for surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(2):1–17, 2013.

[6] Matthew Berger, Andrea Tagliasacchi, Lee M Seversky, Pierre Alliez, Gael Guennebaud, Joshua A Levine, Andrei Sharf, and Claudio T Silva. A survey of surface reconstruction from point clouds. In *Computer Graphics Forum*, volume 36, pages 301–329. Wiley Online Library, 2017.

[7] Alberto Bietti and Julien Mairal. On the Inductive Bias of Neural Tangent Kernels. *arXiv:1905.12173 [cs, stat]*, Oct. 2019.

[8] Robert Bridson. Fast poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH 2007 Sketches*, SIGGRAPH '07, page 22–es, New York, NY, USA, 2007. Association for Computing Machinery.

[9] Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76, 2001.

[10] Rohan Chabra, Jan Eric Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction, 2020.

[11] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository, 2015.

[12] Benjamin Charlier, Jean Feydy, Joan Alexis Glaunès, François-David Collin, and Ghislain Durif. Kernel operations on the GPU, with autodiff, without memory overflows. *arXiv preprint arXiv:2004.11127*, 2020.

[13] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning, 2020.

[14] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. In *Advances in Neural Information Processing Systems*, pages 2933–2943, 2019.

[15] Youngmin Cho and Lawrence K Saul. Kernel methods for deep learning. In *Advances in neural information processing systems*, pages 342–350, 2009.

[16] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition, 2020.

[17] Theo Deprelle, Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Learning elementary structures for 3d shape generation and matching, 2019.

[18] Petros Drineas and Michael W Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *journal of machine learning research*, 6(Dec):2153–2175, 2005.

[19] Matheus Gadelha, Rui Wang, and Subhransu Maji. Deep manifold prior, 2020.

[20] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape, 2020.

[21] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T. Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions, 2019.

[22] Joachim Giesen, Simon Spalinger, and Bernhard Schölkopf. Kernel methods for implicit surface modeling. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1193–1200. MIT Press, 2005.

[23] Leslie Greengard and Vladimir Rokhlin. A fast algorithm for particle simulations. *Journal of computational physics*, 73(2):325–348, 1987.

[24] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes, 2020.

[25] Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. In *ACM SIGGRAPH 2007 papers*, pages 23–es. ACM New York, NY, USA, 2007.

[26] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2mesh: A self-prior for deformable meshes, 2020.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015.

[28] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 71–78, 1992.

[29] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao Zhang. Edge-aware point set resampling. *ACM transactions on graphics (TOG)*, 32(1):1–12, 2013.

[30] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks, 2018.

[31] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013.

[32] Nicolas Le Roux and Yoshua Bengio. Continuous neural networks. In *Artificial Intelligence and Statistics*, pages 404–411, 2007.

[33] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. *2019*

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Jun 2019.

[34] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. Sparse surface reconstruction with adaptive partition of unity and radial basis functions. *Graphical Models*, 68(1):15–24, 2006.

[35] Greg Ongie, Rebecca Willett, Daniel Soudry, and Nathan Srebro. A function space view of bounded norm infinite width relu nets: The multivariate case, 2019.

[36] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019.

[37] Alexander D Poularikas. *Transforms and applications handbook*. CRC press, 2018.

[38] Alessandro Rudi, Luigi Carratino, and Lorenzo Rosasco. Falkon: An optimal large scale kernel method, 2018.

[39] Pedro Savarese, Itay Evron, Daniel Soudry, and Nathan Srebro. How do infinite width bounded norm networks look in function space?, 2019.

[40] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions, 2020.

[41] Donald C Solmon. Asymptotic formulas for the dual radon transform and applications. *Mathematische Zeitschrift*, 195(3):321–343, 1987.

[42] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains, 2020.

[43] Francis Williams, Jerome Parent-Levesque, Derek Nowrouzezahrai, Daniele Panozzo, Kwang Moo Yi, and Andrea Tagliasacchi. Voronoinet: General functional approximators with local support. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[44] Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo. Deep geometric prior for surface reconstruction. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019.

[45] Francis Williams, Matthew Trager, Claudio Silva, Daniele Panozzo, Denis Zorin, and Joan Bruna. Gradient dynamics of shallow univariate relu networks, 2019.