Siggraph Asia 2010 Course Notes

Scattered Data Interpolation and Approximation for Computer Graphics

J.P. Lewis
Weta Digital and Victoria University

Frédéric Pighin
Google Inc.

Ken Anjyo
OLM Digital

Abstract. The goal of scattered data interpolation techniques is to construct a (typically smooth) function from a set of unorganized samples. These techniques have a wide range of applications in computer graphics. For instance they can be used to model a surface from a set of sparse samples, to reconstruct a BRDF from a set of measurements, to interpolate motion capture data, or to compute the physical properties of a fluid. This course will survey and compare scattered interpolation algorithms and describe their applications in computer graphics. Although the course is focused on applying these techniques, we will introduce some of the underlying mathematical theory and briefly mention numerical considerations.

Contents

List of Figures

1 Introduction

Many computer graphics problem start with sampled data. For instance, perhaps we are using a range scanner to create a 3D model out of a real object or perhaps we are trying to deform a surface using motion capture data. In these cases and many others, a set of sparse samples of a continuous function (e.g. a surface or a motion) are available and our goal is to evaluate this function at locations or times that were not sampled. Another way to think about this is that we have low resolution representation of some data and we are trying to find a higher resolution representation.

In some applications the sample locations are on a regular grid or other simple pattern. For instance this is usually the case for image data since the image samples are aligned according to a CCD array. Similarly, the data for a B-spline surface are organized on a regular grid in parameter space. The well known spline interpolation methods in computer graphics address these cases where the data has a regular arrangement.

In other cases the data locations are unstructured or scattered. Methods for *scattered data interpolation* (or approximation) are less well known in computer graphics, for example, these methods are not yet covered in most graphics textbooks. These approaches have become widely known and used in graphics research over the last decade however. This course will attempt to survey most of the known approaches to interpolation and approximation of scattered data.

1.1 Applications

To suggest the versatility of scattered data interpolation techniques, we list a few applications:

- **Surface reconstruction** [11, 41]. Reconstructing a surface from a point cloud often requires an implicit representation of the surface from point data. Scattered data interpolation lends itself well to this representation.
- **Image restoration and inpainting** [58, 47, 35]. Scattered data interpolation can be used to fill missing data. A particular case of this is inpainting where missing data from an image needs to be reconstructed from available data.
- **Surface deformation** [48, 40, 32, 33, 7]. Motion capture systems allow the recording of sparse motions from deformable objects such as human faces and bodies. Once the data is recorded, it needs to be mapped to a 3-dimensional representation of the tracked object so that the the object can be deformed accordingly. One way to deform the object is to treat the problem as a scattered data interpolation problem: the captured data represents a spatially sparse and scattered sampling of the surface that needs to be interpolated to all vertices in a (e.g. facial) mesh.

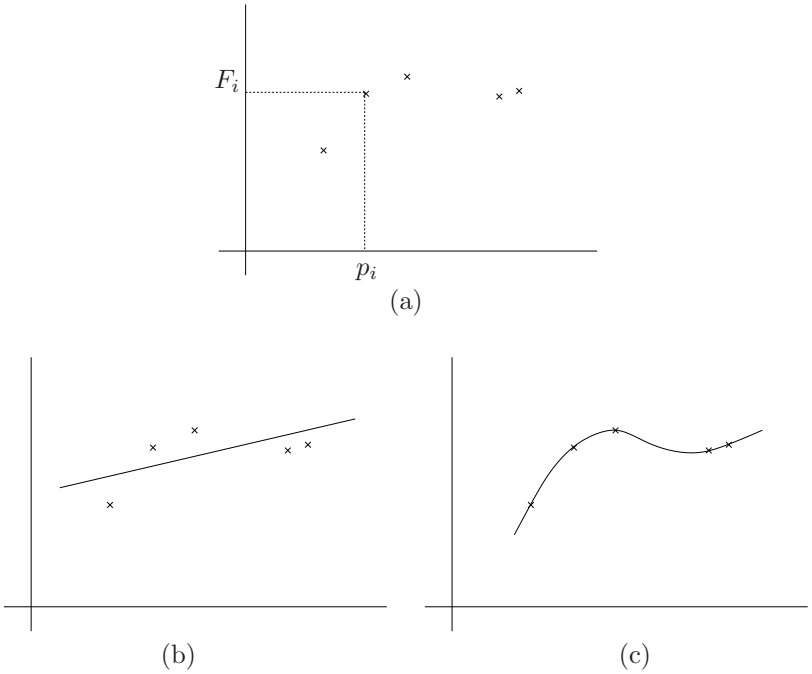


Figure 1: Model fitting vs scattered data interpolation. The input (a) is a set of samples from a 1-dimensional function. A fitted (linear) model (b) or scattered data interpolation (c) can be used to represent the data.

- **Motion interpolation and inverse kinematics** [50].
- **Meshless/Lagrangian methods for fluid dynamics**. In Lagrangian dynamics, the physical properties of a fluid are computed at the location of a set of moving particles. Scattered data interpolation can be used to produce a smooth representation of a field (e.g. velocity).
- **Appearance representation** [64, 59]. Interpolation of measured reflectance data is a scattered interpolation problem.

We will later revisit some of these to illustrate the techniques described in these notes.

2 Problem formulation

From a mathematical point of view, a general scattered data interpolation problem can be formulated in the following way. Let f be an unknown function from $\mathbb{R}^p$ to $\mathbb{R}$. Let $\mathbf{x}_i, \ldots, \mathbf{x}_n$ be a set of points in $\mathbb{R}^p$ and $f_1, \ldots, f_n$ be a set of values in $\mathbb{R}$. The goal is to find a function $\tilde{f}$ such that:

$$\tilde{f}(\mathbf{x}_i) = f_i \text{ for } 1 \leq i \leq n.$$

Note that in general there are many functions that would satisfy these conditions. For instance the piecewise constant function $\tilde{f}(\mathbf{x}) = f_i$ if $\mathbf{x} = \mathbf{x}_i$ and 0 otherwise is always a solution. What will usually be assumed is that $\tilde{f}$ obeys some smoothness properties such as continuity or differentiability. Moreover, often it will be the case that $\tilde{f}$ belongs to a family or class of functions. By constraining the solution space, we are implicitly making some assumptions on the type of function that f is.

To conclude, if we assume that f can be well represented within a function space $\mathcal{F}$, the scattered data interpolation problem can be specified as:

$$\begin{aligned} \text{Find} \quad & \tilde{f} \in \mathcal{F} \\ & \text{such that } \tilde{f}(\mathbf{x}_i) = f_i, \text{ for } 1 \leq i \leq n. \end{aligned}$$

This is the general setup for many types of interpolation problems, and it will reappear in more specific forms in later sections.

2.1 Interpolation (parametric) vs. model fitting (non-parametric)

Scattered data interpolation should be distinguished from *model fitting*. Although both approaches can potentially provide a smooth function that passes near the data, scattered interpolation requires all the original data in order to do its work, whereas model fitting approximates and then discards the original data.

Choosing between model fitting and scattered data interpolation depends on what we know or believe about the data. If the data is free of noise but we do not know any model that correctly summarizes it, then scattered data interpolation is the way to go. If however, we know that the data is tainted by noise, or there is too much data, or we are just happy with a rough representation of the data, then model fitting might be the better choice. Figure 1 illustrates the difference between model fitting and scattered data interpolation with a simple 2D example. In the statistical literature this distinction between keeping all the data and summarizing it is termed "non-parametric" or "parametric". Model fitting is termed parametric because a "model" has a fixed number of parameters (for example, the 1-D Gaussian model has two parameters).

2.2 Interpolation vs. approximation

In the literature, a difference is sometime made between scattered data *interpolation* (SDI) and scattered data *approximation* (SDA). The difference stems from the behavior of the reconstructed function, $\tilde{f}$, at the sample sites. In scattered data interpolation, we require the function to perfectly fit the data:

$$\tilde{f}(\mathbf{x}_i) = f_i, \text{ for } 1 \leq i \leq n.$$

For scattered data approximation, we ask that the function merely pass close to the data.

Stated this way SDI and SDA are two different techniques. For one thing, the SDA formulation let us easily specify other soft constraints, for instance a smoothness term proportional to $\nabla^2 \tilde{f}$. However, given the applications we are interested in, from a practical point of view the difference can be glossed over. This is mainly due to the fact that our "data", $\{\mathbf{x}_i, f(\mathbf{x}_i)\}$, will always contain some amount of noise; which makes it less relevant to fit the data exactly. Also from a practical point of view, it is often desirable to specify some smoothness constraints in the problem formulation. These can be specified explicitly for instance as constraints on the derivatives of $\tilde{f}$ or implicitly by using regularization.

Finally, there is a case though where a clear distinction between approximation and interpolation should be made, it is when the number of samples is such that a reconstruction based on all the samples cannot be computed. In such case, we might be forced to build an approximation using a subset of the data.

2.3 Dimensionality

In the remainder of these notes individual techniques will usually be described for the case of mapping from a multidimensional domain to a single field, $\mathbb{R}^p$ to $\mathbb{R}^1$, rather than the general multidimensional input to vector output ($\mathbb{R}^p$ to $\mathbb{R}^q$) case.

Producing a q-dimensional output can be done by running q separate copies of $\mathbb{R}^p$ to $\mathbb{R}^1$ interpolators. In most techniques some information can be shared across these output dimensions – for example, in radial basis interpolation the matrix of the kernel applied to distances (section 5.6) can be inverted once and reused with each dimension.

Throughout the notes we strived to use the following typesetting conventions for mathematical symbols:

- Scalar values in lower-case: e.g. the particular weight w_k.
- Vector values in bold lower-case: e.g. the weight vector $\mathbf{w}$ or the particular point $\mathbf{x}_k$.

- Matrices in upper-case: e.g. the interpolation matrix $\mathbf{A}$.

3 Issues

While we survey the various scattered data interpolation techniques we will use these criteria as a basis for comparison:

- Smoothness. In general, continuity and differentiability are assumed. But we might require higher order smoothness.
- Extrapolation. It is sometime desirable to be able to model the behavior of f outside of the convex hull of the samples. How this is done can be fairly tricky since by definition the further away we are from the sampled point the less we know about f.
- Local versus global. For a local technique a change in one of the sample value f_i only affects the values of $\tilde{f}$ around $\mathbf{x}_i$. On the contrary with a global technique, the evaluation of $\tilde{f}$ at any point is determined by all the samples.
- Stability. Numerical stability refers to the ability of an algorithm not to amplify errors in the input. Stability is a source of concern for some scattered data interpolation techniques.
- Performance. Scattered data interpolation techniques can require large amount of computation and memory. Some techniques scale better as a function of the size of the dataset.

4 A bit of history

The modern approach (or at least the one covered in these notes) to scattered data interpolation can be traced back to the 1960s with the pioneering work of D. Shepard [55] on what is called today Shepard's interpolation. Shepard's applied his method to surface modelling. In the 1970s, R. Hardy [27], an Iowa State geodesist, invented different methods called the multiquadrics and the inverse multiquadrics. Around the same time, a mathematician at the Université Jean Fourier in France, Jean Duchon [14] took a variational approach similar to that used for splines, and got different interpolants that led to thin plate splines. A comparison, done by Richard Franke [23] at the Naval Postgraduate school in Monterey California, concluded that Hardy's multiquadric method and Duchon's thin plate spline approach were the best techniques.

The more recent history of scattered data interpolation and extrapolation is marked by a great increase in interest for these techniques both theoretically and in their application.

5 Scattered Interpolation Algorithms

5.1 Shepard's interpolation

Shepard's Method [55] is probably the simplest scattered interpolation method, and it is frequently re-invented. The interpolated function is

$$\tilde{f}(\mathbf{x}) = \sum_{k}^{N} \frac{w_k(\mathbf{x})}{\sum_j w_j(\mathbf{x})} f(\mathbf{x}_k),$$

where w_i is the weight function at site i:

$$w_j(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_i\|^{-p},$$

the power p is a positive real number, and $\|\mathbf{x}-\mathbf{x}_i\|$ denotes the distance between the query point $\mathbf{x}$ and data point $\mathbf{x}_i$. Since p is positive, the weight functions decrease as the distance to the sample sites increase. p can be used to control the shape of the approximation. Greater values of p assign greater influence to values closest to the interpolated point. Because of the form of the weight function this technique is sometime referred as inverse distance weighting. Notice that:

- For $0 < p \leq 1$, $\tilde{f}$ has sharp peaks.
- For $p > 1$, $\tilde{f}$ is smooth at the interpolated points, however its derivative is zero at the data points (Fig. 3), resulting in evident "flat spots".

Shepard's method is not an ideal interpolator however, as can be clearly seen from Figs. 2,3.

The *Modified Shepard's Method* [41, 22] aims at reducing the impact of far away samples. This might be a good thing to do for a couple of reasons. First, we might want to determine the local shape of the approximation only using nearby samples. Second, using all the samples to evaluate the approximation does not scale well with the number of samples. The modified Shepard's method computes interpolated values only using samples within a sphere of radius r. It uses the weight function:

$$w_j(\mathbf{x}) = \left[\frac{r - d(\mathbf{x}, \mathbf{x}_i)}{r d(\mathbf{x}, \mathbf{x}_i)}\right]^2.$$

where $d()$ notates the distance between points. Combined with a spatial data structure such as a k-d tree or a Voronoi diagram [43] this technique can be used on large data sets.

5.2 Moving least-squares

Moving least-squares builds an approximation by using a local polynomial function. The approximation is set to locally belong in Π_m^p the set of polynomials

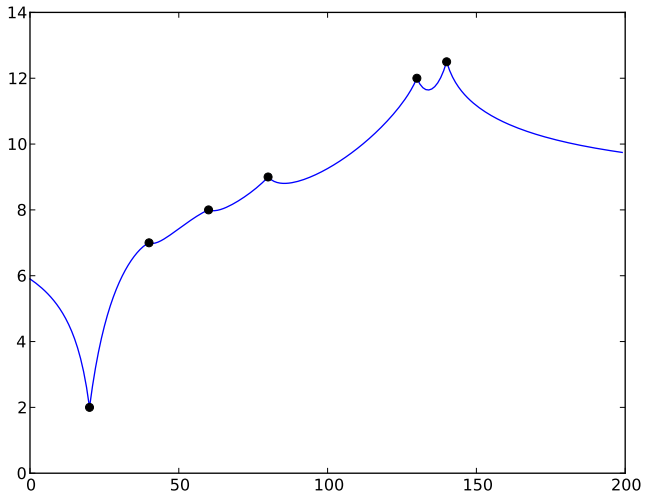


Figure 2: Shepard's interpolation with $p = 1$.

with total degree m in p dimensions. At each point, $\mathbf{x}$, we would like the polynomial approximation to best fit the data in a weighted least-squares fashion, i.e.:

$$\tilde{f}(\mathbf{x}) = \underset{g \in \Pi_m^p}{\arg\min} \sum_i^N w_i(\|\mathbf{x} - \mathbf{x}_i\|)(g(\mathbf{x}_i) - f_i)^2,$$

where w_i is a weighting function used to emphasize the contribution of nearby samples, for instance $w_i(d) = e^{-\frac{d^2}{\sigma^2}}$.

Figure 4 illustrates the technique with a 1-dimensional data set reconstructed with a moving second order polynomial.

Using a basis of Π_m^p, $\boldsymbol{b}(\mathbf{x}) = \{b_1(\mathbf{x}), \ldots, b_l(\mathbf{x})\}$ we can express the polynomial g as a linear combination in that basis: $g(\mathbf{x}) = \boldsymbol{b}^t(\mathbf{x})\mathbf{c}$, where $\mathbf{c}$ is a vector of coefficients. If we then call, $\mathbf{a}$, the expansion of $\tilde{f}$ in the basis $\boldsymbol{b}$, we can then write:

$$\tilde{f}(\mathbf{x}) = \boldsymbol{b}^t(\mathbf{x})\mathbf{a}(\mathbf{x})$$

with

$$\mathbf{a}(\mathbf{x}) = \underset{\mathbf{c} \in \mathbb{R}^l}{\arg\min} \sum_i^N w_i(\|\mathbf{x} - \mathbf{x}_i\|)(\boldsymbol{b}^t(\mathbf{x}_i)\mathbf{c} - f_i)^2.$$

Computing, $\mathbf{a}(\mathbf{x})$, is a linear least-squares problem that depends on $\mathbf{x}$. We can extract the system by differentiating the expression above with respect to $\mathbf{c}$ and

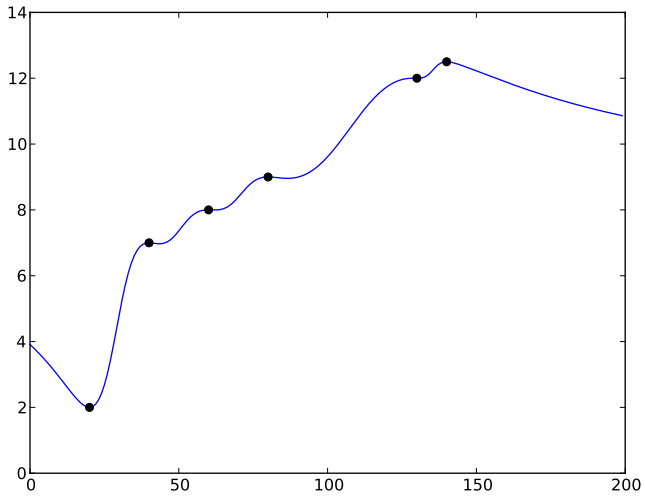


Figure 3: Shepard's interpolation with $p = 2$. Note the the derivative of the function is zero at the data points, resulting in smooth but uneven interpolation. Note that this same set of points will be tested with other interpolation methods; compare Figs. 7, 8, 12, etc.

setting it to 0:

$$\begin{aligned}
& \frac{\partial}{\partial \mathbf{c}} \left(\sum_i^N w_i(\|\mathbf{x} - \mathbf{x}_i\|)(\boldsymbol{b}^t(\mathbf{x}_i)\mathbf{c} - f_i)^2 \right) \Big|_{\mathbf{a}} = 0 \\
\Leftrightarrow \quad & \sum_i^N w_i(\|\mathbf{x} - \mathbf{x}_i\|)\boldsymbol{b}(\mathbf{x}_i)(\boldsymbol{b}^t(\mathbf{x}_i)\mathbf{a} - f_i) = 0 \\
\Leftrightarrow \quad & \left(\sum_i^N w_i(\|\mathbf{x} - \mathbf{x}_i\|)\boldsymbol{b}(\mathbf{x}_i)\boldsymbol{b}^t(\mathbf{x}_i) \right) \mathbf{a} = \sum_i^N f_i w_i(\|\mathbf{x} - \mathbf{x}_i\|)\boldsymbol{b}(\mathbf{x}_i).
\end{aligned}$$

This last equation can be written in matrix form $\boldsymbol{A}\mathbf{a} = \mathbf{d}$ with:

$$\boldsymbol{A} = \sum_i^N w_i(\|\mathbf{x} - \mathbf{x}_i\|)\boldsymbol{b}(\mathbf{x}_i)\boldsymbol{b}^t(\mathbf{x}_i)$$

and

$$\mathbf{d} = \sum_i^N f_i w_i(\|\mathbf{x} - \mathbf{x}_i\|)\boldsymbol{b}(\mathbf{x}_i).$$

Note that the matrix $\boldsymbol{A}$ is square and symmetric. In the usual case, where $\mathbf{w}$ is

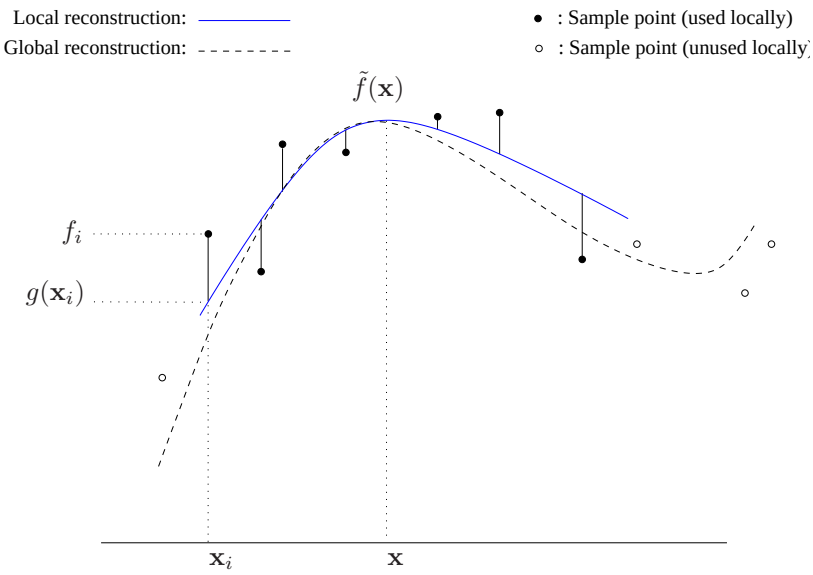


Figure 4: Moving least-squares in 1-D. The function is locally reconstructed using a polynomial of degree 2. The local approximating polynomial is refitted for each evaluation of the reconstructed function.

non-negative $\boldsymbol{A}$ is also symmetric and positive semi-definite.

$$\begin{aligned}
\boldsymbol{x}^t \boldsymbol{A} \boldsymbol{x} &= \boldsymbol{x}^t \left(\sum_i^N w_i(\|\mathbf{x} - \mathbf{x}_i\|) \boldsymbol{b}(\mathbf{x}_i) \boldsymbol{b}^t(\mathbf{x}_i) \right) \boldsymbol{x} \\
&= \sum_i^N w_i(\|\mathbf{x} - \mathbf{x}_i\|) \boldsymbol{x}^t \boldsymbol{b}(\mathbf{x}_i) \boldsymbol{b}^t(\mathbf{x}_i) \boldsymbol{x}) \\
&= \sum_i^N w_i(\|\mathbf{x} - \mathbf{x}_i\|) (\boldsymbol{x}^t \boldsymbol{b}(\mathbf{x}_i))^2 \geq 0.
\end{aligned}$$

If the matrix $\boldsymbol{A}$ has full rank the system can be solved using the Cholesky decomposition.

It would seem that the computational cost of moving least-squares is excessive since it requires solving a linear system for each evaluation of the reconstructed function. While this is true, if the weight functions fall quickly to zero then the size of the system can involve only a few data points.

5.2.1 Design issues

The formulation of moving least-squares exposes two sets of parameters that determine what type of approximation is generated: the local approximation

function, g, and the weight functions, w_i. Let us examine their role in the approximation.

The local approximation, g. The local approximation function, g, determines the local shape and degree of smoothness of the approximation. Typically g is a polynomial function, but other classes of function can be selected. This offers a great deal of flexibility especially if the goal is to enforce some local properties. For instance, this is the case for as-rigid-as-possible surface transformation [63], where g is chosen to be a similarity transformation (rotation and uniform scaling). Solving for the transformation is an instance of the classical weighted Procrustes problem [36].

The weight functions, w_i. The role of the weight functions is to determine how much influence each sample has when evaluating the reconstruction at a given point. A weight is a function of the distance between the evaluation point and a given sample. The main idea is that distant samples should have less influence. In this respect, these are traditionally monotonically decreasing positive functions. Beyond this simple observation, a few important issues are determined by the weight function: interpolation, smoothness, and discontinuities.

The way we have formulated the moving least-squares technique it would seem that it provides an approximation to the data but does not interpolate the samples. In fact if we chose a kernel such as:

$$w_i(d) = d^{-\beta}$$

with $\beta > 0$, we would observe that MLS indeed interpolates the data. This is possible because in this case $w_i(\mathbf{x}_i) = \infty$ which yields $\tilde{f}(\mathbf{x}_i) = f_i$.

The shape of the weight functions is in a great part responsible for how smooth the reconstruction is. This issue is clearly a trade-off, since a function that is too narrow can result in spikes in the reconstruction, while a broad function may over-smooth the data. There is an argument for constructing adaptive weight functions (at least for non-uniform samples), such that the speed at which the function falls off depends on the local sample density. This very issue was explored by Pauly *et al.* [44, 45]. In their original attempt [44], they extract the radius, $r.$ of the sphere containing the k-nearest neighbor to build adaptive weight functions of the form:

$$w_i(d) = e^{-9\frac{d^2}{r^2}}$$

They use a similar function in their second attempt [45] that uses a more complex algorithm for determining the shape parameter from the local sampling density. Adamson and Alexa [1] use an anisotropic shape parameter that is

determined by locally fitting an ellipse so that it aligns with the normal and the two principal curvature directions.

If the original function has sharp features, the reconstruction has to be able to handle discontinuities. This is an issue explored by Fleishman *et al.* [20]. They develop a robust framework that allows the detection of sharp features as areas containing outliers. Points of sharp feature are assigned multiple surface patches.

5.2.2 Applications

Surface reconstruction The use of moving least-squares for reconstructing surfaces from unorganized data has been explored pretty thoroughly. Cheng *et al.* [12] give an overview of the diverse techniques.

Fleishman *et al.* [20] uses moving least-squares to reconstruct piecewise smooth surfaces from noisy point clouds. They introduce robustness in their algorithm in multiple ways. One of them is an interesting variant on the moving least squares estimation procedure. They estimate the parameters, β, of their model, f_β, using a robust fitting function:

$$\beta = \arg\min_{\beta} \operatorname{median}_{i} \|f_\beta(x_i) - y_i\|.$$

The sum in the original formulation has been replaced by a more robust median operator.

Image warping Schaefer *et al.* [52] introduces a novel image deformation technique using moving least-squares. Their approach is to solve for the best transformation, $l_v(x)$, at each point, v, in the image by minimizing:

$$\sum_i w_i \|l_v(p_i) - q_i\|$$

where $\{p_i\}$ is a set of control points and $\{q_i\}$ an associated set of displacements. By choosing different transform types for l_v (affine, rigid, ..), they create different effects.

5.3 Partition of unity

The Shephards and MLS methods are specific examples of a general *partition of unity* framework for interpolation. The underlying principal of a partition of unity method is that it is usually easier to approximate the data locally than to find a global approximation that fits it all. Thus, partition of unity assembles

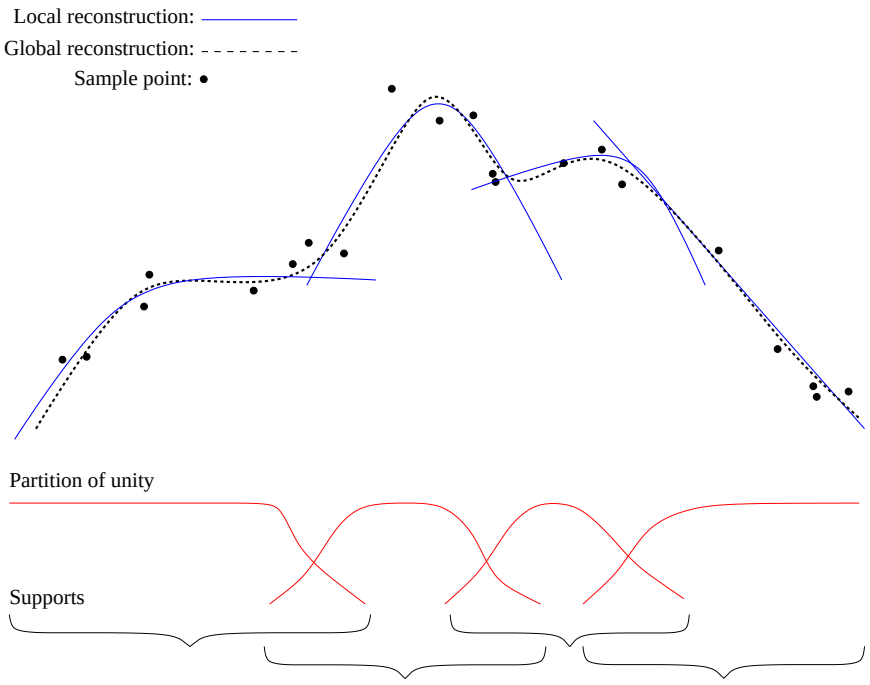


Figure 5: Partition of unity in 1-D. The partition of unity are a set of weighing functions that sum to 1 over a domain. These are used to blend a set of (e.g. quadratic) functions fit to local subsets of the data to create a global reconstruction.

a global approximation from a set of local approximations, i.e., a set of weight functions $\{\phi_k\}$ compactly supported over a domain Ω such that:

$$\sum_k \phi_k = 1 \text{ on } \Omega.$$

We say that the functions $\{\phi_k\}$ form a partition of unity. Now we consider a set of approximations of f, $\{q_k\}$, where q_k is defined over the support of ϕ_k, then we can compute a global approximation as:

$$\tilde{f}(\mathbf{x}) = \sum_k^N \phi_k(\mathbf{x}) q_k(\mathbf{x}).$$

In this description the functions, ϕ_k, only need to be non-negative.

To interpret Shepards method as a partition of unity approach, define

$$\phi_k(\boldsymbol{x}) = \frac{\|\boldsymbol{x} - \boldsymbol{x}_k\|^p}{\sum_j \|\boldsymbol{x} - \boldsymbol{x}_j\|^p}$$

and q_k as the constant function defined by the kth data point.

5.4 Natural neighbor interpolation

Natural Neighbor Interpolation was developed by Robin Sibson [56]. It is similar to Shepard's interpolation in the sense that the approximation is written as a weighted average of the sampled values. It differs in that the weights are volume-based as opposed to the distance-based weights of Shepard's method.

Natural neighbor techniques uses a Voronoi diagram of the sampled sites to formalize the notion of "neighbor": two sites are neighbors if they share a common boundary in the Voronoi diagram. Using duality, this is equivalent to writing that the sites form an edge in the Delaunay triangulation. By introducing the evaluation point $\mathbf{x}$ in the Delaunay triangulation, the natural neighbors of $\mathbf{x}$ are the nodes that are connected to it. The approximation is then written as:

$$\tilde{f}(\mathbf{x}) = \sum_{k \in N} \alpha_k(\mathbf{x}) f(\mathbf{x}_k),$$

where N is the set of indices associated with the natural neighbors of $\mathbf{x}$ and $\alpha_k(\mathbf{x})$ are weight functions.

$$\alpha_k(\mathbf{x}) = \frac{u_k}{\sum_{j \in N} u_j}$$

where u_j is the volume of the intersection of the node associated with the evaluation point and the node associated with the j-th neighbor in the original Voronoi diagram.

5.4.1 Applications

Surface reconstruction In [9] Boisonnat and Cazals represent surfaces implicitly as the zero-crossings of a signed pseudo-distance function. The function is set to zero at the sampled points and is interpolated to the whole 3D space.

5.5 Wiener interpolation and Gaussian Processes

Wiener interpolation differs from polynomial interpolation approaches in that it is based on the expected correlation of the data. Wiener interpolation of discrete data is simple, requiring only the solution of a matrix equation. This section describes two derivations for discrete Wiener interpolation.

Some advantages of Wiener interpolation are:

- The data can be arbitrarily spaced.
- The algorithm applies without modification to multi-dimensional data.
- The interpolation can be made local or global to the extent desired. This is achieved by adjusting the correlation function so that points beyond a desired distance have a negligible correlation.

Figure 6: Non-fractal landscapes invented with Wiener interpolation, from [34]

- The interpolation can be as smooth as desired, for example an analytic correlation function will result in an analytic interpolated curve or surface.
- The interpolation can be shaped and need not be "smooth", for example, the correlation can be negative at certain distances, oscillatory, or (in several dimensions) have directional preferences.
- The algorithm provides an error or confidence level associated with each point on the interpolated surface.
- The algorithm is optimal by a particular criterion (below) which may or may not be relevant.

Some disadvantages of Wiener interpolation:

- It requires knowing or inventing the correlation function. While this may arise naturally from the problem in some cases, in other cases it would require interactive access to the parameters of some predefined correlation models to be "natural".
- It requires inverting a matrix whose size is the number of significantly correlated data points. This can be a practical problem if a large neighborhood is used. A further difficulty arises if the chosen covariance is broad, causing the resulting covariance matrix (see below) to have similar rows and hence be nearly singular. Sophistication with numerical linear algebra will be required in this case.

Terminology

Symbols used below:

f value of a stochastic process at time x

$\hat{f}$ estimate of f

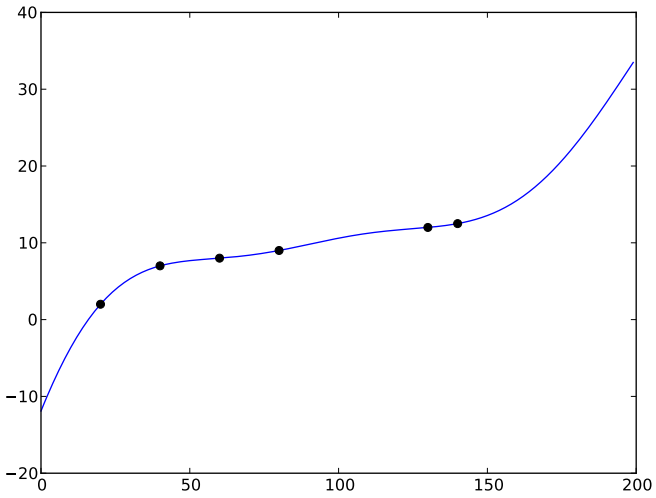


Figure 7: One-dimensional Wiener interpolation, using a Gaussian covariance matrix.

f_j observed values of the process at times or locations x_j

The derivations require two concepts from probability:

- The correlation of two values is the expectation of their product, $\mathbf{E}[xy]$. The autocorrelation or autocovariance function is the correlation of pairs of points from a process:

  $$C(x_1, x_2) = \mathbf{E}\left\{f(x_1)f(x_2)\right\}$$

  For a stationary process this expectation is a function only of the distance between the two points: $C(\tau) = \mathbf{E}[f(x)f(x+\tau)]$. The variance is the value of the autocorrelation function at zero: $\mathrm{var}(x) = C(0)$. (Auto)covariance usually refers to the correlation of a process whose mean is removed and (usually) whose variance is normalized to be one. There are differences in the terminology, so "Correlation function" will mean the autocovariance function of a normalized process here.

- Expectation behaves as a linear operator, so any factor or term which is known can be moved "outside" the expectation. For example, assuming a and b are known,

  $$\mathbf{E}\left\{af + b\right\} = a\mathbf{E}f + b$$

  Also, the order of differentiation and expectation can be interchanged, etc.

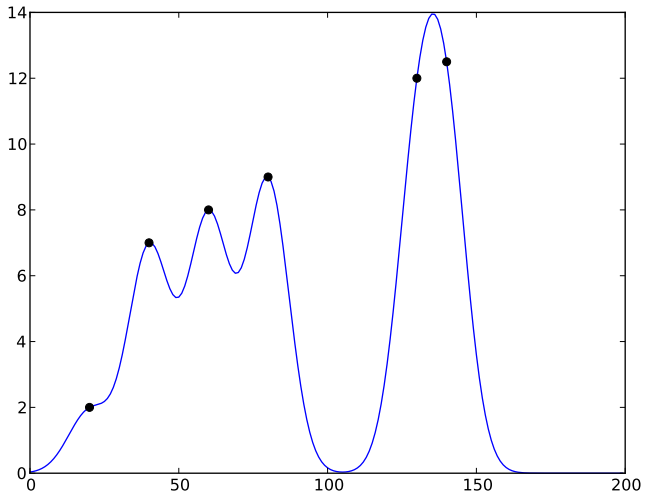


Figure 8: Similar to Fig. 7, but the variance of the Gaussian is too narrow for this data set.

Definition

Wiener interpolation estimates the value of the process at a particular location as a weighted sum of the observed values at some number of other locations:

$$\hat{f} = \sum w_j f_j \tag{1}$$

The weights w_j are chosen to minimize the expected squared difference or error between the estimate and the value of the "real" process at the same location:

$$\mathbf{E}\left\{(f - \hat{f})^2\right\} \tag{2}$$

The reference to the "real" process in (2) seems troublesome because the real process may be unknowable at the particular location, but since it is the *expected* error which is minimized, this reference disappears in the solution.

Wiener interpolation is optimal among linear interpolation schemes in that it minimizes the expected squared error (2). When the data have jointly Gaussian probability distributions (and thus are indistinguishable from a realization of a Gaussian stochastic process), Wiener interpolation is also optimal among nonlinear interpolation schemes.

Derivation #1

The first derivation uses the "orthogonality principle": the squared error of a linear estimator is minimum when the error is 'orthogonal' in expectation to all of the known data, with 'orthogonal' meaning that the expectation of the product of the data and the error is zero:

$$\mathbf{E}\left\{(f-\hat{f})f_k\right\} = 0 \text{ for all } j$$

Substituting $\hat{f}$ from (1),

$$\mathbf{E}\left\{(f-\sum w_j f_j)f_k\right\} = 0 \tag{3}$$

$$\mathbf{E}\left\{ff_k - \sum w_j f_j f_k\right\} = 0$$

The expectation of ff_k is the correlation $C(x-x_k)$, and likewise for $f_j f_k$:

$$C(x-x_k) = \sum w_j C(x_j - x_k)$$

or

$$\mathbf{Cw} = \mathbf{c} \tag{4}$$

This is a matrix equation which can be solved for the coefficients w_j. The coefficients depend on the positions of the data f_j though the correlation function, but not on the actual data values; the values appear in the interpolation (1) though. Also, (4) does not directly involve the dimensionality of the data. The only difference for multi-dimensional data is that the correlation is a function of several arguments: $\mathbf{E}[PQ] = C(x_p - x_q, y_p - y_q, z_p - z_q, \ldots)$.

Derivation #2

The second derivation minimizes (2) by differentiating with respect to each w_k. Since (2) is a quadratic form (having no maxima), the identified extreme will be a minimum (intuitively, a squared difference (2) will not have maxima).

$$\frac{d}{dw_k}\left[\mathbf{E}\left\{(f-\sum w_j f_j)^2\right\}\right] = \mathbf{E}\left[\frac{d}{dw_k}(f-\sum w_j f_j)^2\right] = 0$$

$$2\mathbf{E}\left\{(f-\sum w_j f_j)\frac{d}{dw_k}(f-\sum w_j f_j)\right\} = 0$$

$$\mathbf{E}\left\{(f-\sum w_j f_j)f_k\right\} = 0$$

which is (3).

Cost

From (4) and (1), the coefficients w_j are $\mathbf{w} = \mathbf{C}^{-1}\mathbf{c}$, and the estimate is $\hat{f} = \mathbf{x}^T\mathbf{C}^{-1}\mathbf{c}$. The vector $\mathbf{c}$ changes from point to point, but $\mathbf{x}^T\mathbf{C}^{-1}$ is constant for given data, so the per point estimation cost is a dot product of two vectors whose size is the number of data points.

Confidence

The interpolation coefficients w_j were found by minimizing the expected squared error (2). The resulting squared error itself can be used as a confidence measure for the interpolated points. For example, presumably the error variance would be high away from the data points if the data are very uncharacteristic of the chosen correlation function. The error at a particular point is found by expanding (2) and substituting $\mathbf{a} = \mathbf{C}^{-1}\mathbf{c}$:

$$\begin{aligned}
\mathbf{E}\left\{(f-\hat{f})^2\right\} &= \mathbf{E}\left\{(f-\sum w_j f_j)^2\right\} \\
&= \mathbf{E}\left\{f^2 - 2f\sum w_j f_j + \sum w_j f_j \sum w_k f_k\right\} \\
&= \mathrm{var}(x) - 2\sum w_j C(x,x_j) + \sum w_j w_k C(x_j,x_k)
\end{aligned}$$

(switching to matrix notation)

$$\begin{aligned}
&= C(0) - 2\mathbf{w}^T\mathbf{c} + \mathbf{w}^T\mathbf{C}\mathbf{w} \\
&= C(0) - 2(\mathbf{C}^{-1}\mathbf{c})^T\mathbf{c} + (\mathbf{C}^{-1}\mathbf{c})^T\mathbf{C}\mathbf{C}^{-1}\mathbf{c} \\
&= C(0) - \mathbf{c}^T\mathbf{C}^{-1}\mathbf{c} \\
&= C(0) - \sum w_j C(x,x_j)
\end{aligned}$$

Applications: terrain synthesis, Kriging, Gaussian processes [34] used Wiener interpolation in a hierarchical subdivision scheme to synthesize random terrains at run time (Fig. 6). The covariance was specified, allowing the synthesized landscapes to have arbitrary power spectra (beyond the fractal $1/f^p$ family of power spectra). The Wiener interpolation technique is also known as Kriging, and is closely related to Gaussian processes. The latter two ideas have recently been rediscovered in the machine learning (and graphics [38]) communities.

5.6 Radial basis functions

Radial basis functions are the most commonly used scattered data interpolation technique. They are conceptually easy to understand and simple to implement. From a high level point of view, a radial basis functions approximation works by summing a set of replicates of a single basis function. Each replicate is centered

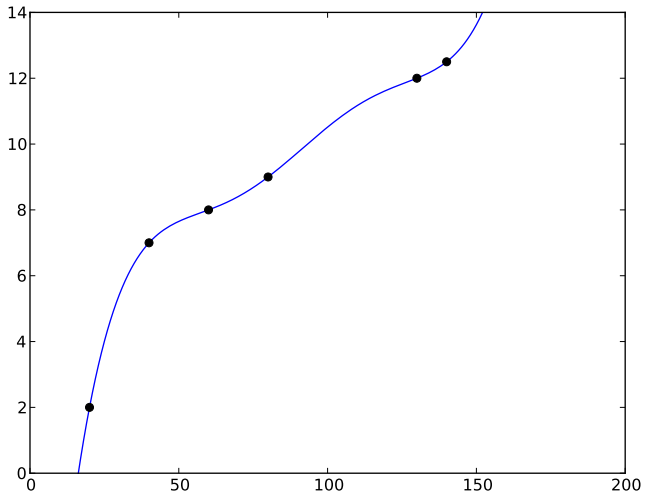


Figure 9: Radial basis interpolation with a Gaussian kernel

at a data point and scaled to respect the interpolation conditions. This can be written as:

$$\tilde{f}(\mathbf{x}) = \sum_{k}^{N} w_k \phi(\|\mathbf{x} - \mathbf{x}_k\|),$$

where ϕ is a function from $[0, \infty[$ to $\mathbb{R}$ and $\{w_k\}$ is a set of q-dimensional weights. It should be fairly clear from this formula why this technique is called "radial": The influence of a single data point is constant on a sphere centered at that point. Without any further information on the structure of the input space, this seems a reasonable assumption. Note also that it is remarkable that the function, ϕ, be univariate: Regardless of the number of dimensions of the input space, we are interested in distances between points.

The choice of the kernel ϕ is an important choice and we delay this topic to a later section. We here list some of the most common kernels in computer graphics applications:

- Gaussian $\quad \phi(r) = \exp(-(r/c)^2)$
- Thin plate spline $\quad \phi(r) = r^2 \log r \qquad$ (in two dimensions)
- Hardy multiquadratic $\phi(r) = \sqrt{r^2 + c^2}, c > 0$

The thin plate spline kernel is of particular practical interest since it is related to the minimization of a bending energy (see section 6).

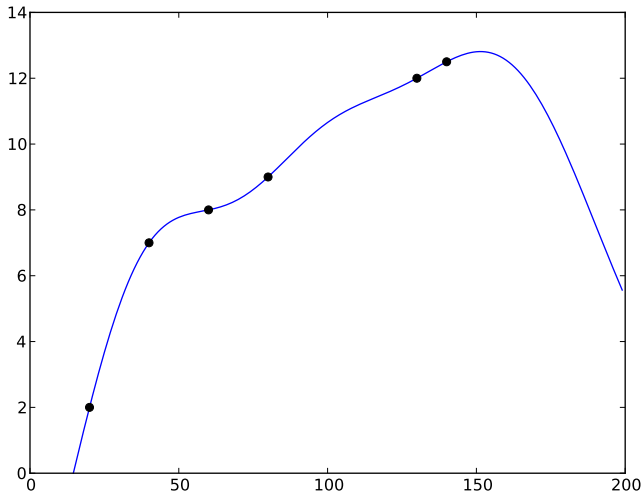


Figure 10: Radial basis interpolation with a Gaussian kernel, varying the kernel width relative to Fig. 9

For a radial basis functions interpolation, the interpolation conditions are written as follow:

$$\tilde{f}(\mathbf{x}_i) = \sum_k^N w_k \phi(\|\mathbf{x}_i - \mathbf{x}_k\|) = f_i, \text{ for } 1 \leq i \leq n.$$

This is a linear system of equations where the unknowns are the vector of weights $\{w_k\}$. To see this, let us call $\phi_{i,k} = \phi(\|\mathbf{x}_i - \mathbf{x}_k\|)$. We can then write the equivalent matrix representation of the interpolation conditions:

$$\begin{bmatrix} \phi_{1,1} & \phi_{1,2} & \phi_{1,3} & \cdots \\ \phi_{2,1} & \phi_{2,2} & \cdots & \\ \phi_{3,1} & \cdots & & \\ \vdots & & & \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \end{bmatrix}$$

This is a square system with as many equations as unknowns. Thus we can form the radial basis function interpolation by solving this system of equations.

5.6.1 Radial basis function with polynomial term

It is sometime convenient to add a polynomial term to radial basis functions approximation. There are several reasons for this:

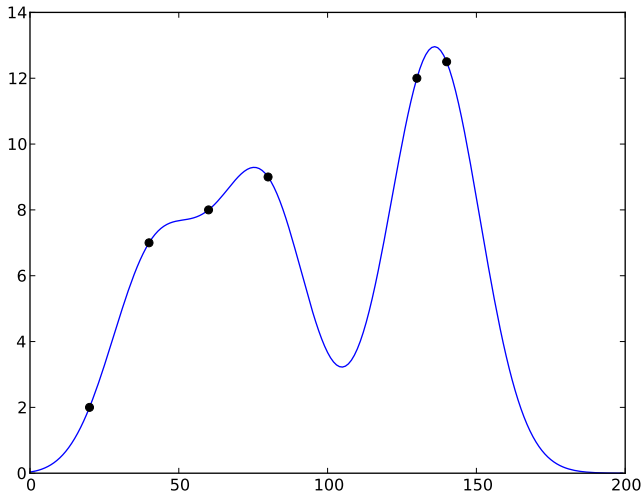


Figure 11: Radial basis interpolation with a Gaussian kernel, varying the kernel width relative to Fig. 9,10. In this figure the kernel width is too narrow to adequately interpolate the data.

- Polynomial reproducibility. In some cases it is useful to have an approximation that can faithfully reproduce polynomials of up to a given degree. For instance, imagine that we are trying to model a 3-dimensional deformation field. In the specific case where the deformation at the sample points is affine, we probably want the RBF approximation to yield an affine function; i.e. a polynomial of degree 1.
- Extrapolation. When using a kernel that vanishes at infinity we are faced with an approximation that does not extrapolate well since its value far from the data points decreases to zero. In these cases, it might be useful to use a polynomial to model the far-field behavior of the function.
- Null space. Most importantly, the RBF kernels corresponding to the differential operators ∇^m have a non-trivial null space that contains exactly these polynomials. For example, in one dimension the operator that produces the second derivative of a function responds zero for either a constant or linear function.

From a mathematical point of view, a radial basis functions approximation with a polynomial term can be written as:

$$\tilde{f}(\mathbf{x}) = \sum_{k}^{N} w_k \phi(\|\mathbf{x} - \mathbf{x}_k\|) + g(\mathbf{x}),$$

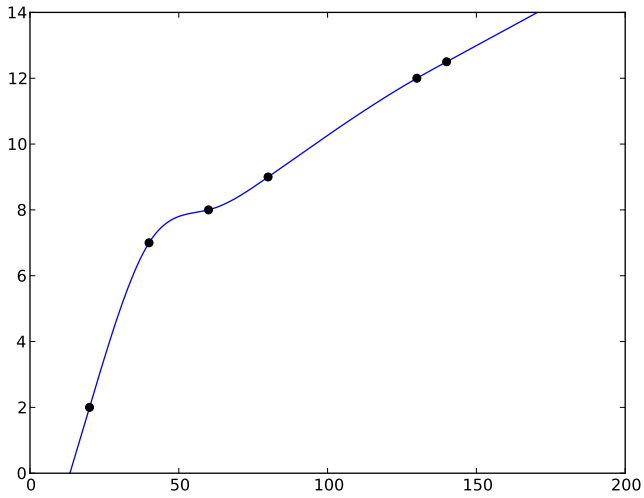


Figure 12: The one-dimensional equivalent of thin-plate spline interpolation is the natural cubic spline, with radial kernel $|r|^3$ in one dimension. This spline minimizes the integrated square of the second derivative (an approximate curvature) and so extrapolates to infinity away from the data.

where g belongs to Π_m^p, the set of real-valued polynomials of p variables and of degree at most $m-1$. Notice that our function $\tilde{f}$ is not now only determined by the weights $\{w_k\}$ but also by the coefficients of the polynomial g.

Let us now examine the interpolation conditions to solve for our unknowns. The conditions now become:

$$\tilde{f}(\mathbf{x}_i) = \sum_k^N w_k \phi(\|\mathbf{x}_i - \mathbf{x}_k\|) + g(\mathbf{x}_i) = f_i, \text{ for } 1 \leq i \leq n.$$

The good news is that this is still a linear system of equations since it depends linearly on both the weights $\{w_k\}$ and the coefficients of g. The bad news is that we no longer have enough equations, in other words the system is under-determined. This means that we need additional constraints to cover for $\binom{m-1+p}{p}$ factors in the polynomial term. These constraints usually come from what are called *zero moment conditions*:

$$\sum_k^N w_k h(\mathbf{x}_k) = 0, \text{ for all } h \in \Pi_m^p.$$

There are different ways of justifying these conditions. For instance one could make an orthogonality argument or use polynomial reproducibility (if all the

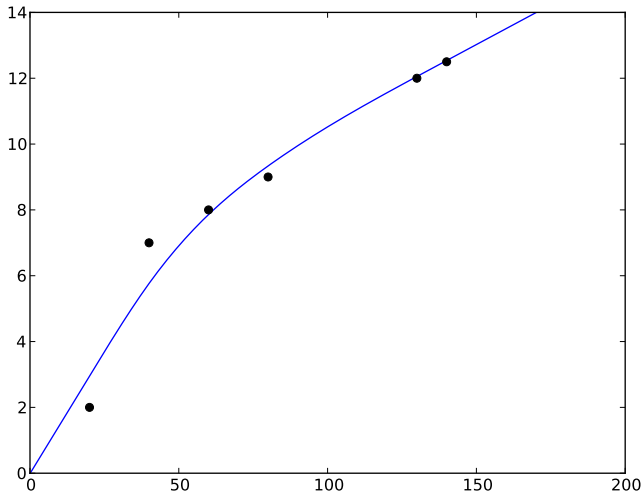


Figure 13: Adding regularization to the plot in Fig. 12 causes the curve to approximate rather than interpolate the data.

data points are on a polynomial of degree at most $m-1$ then the RBF term has to vanish).

To determine the interpolation function, we now have to solve the following system of equations:

$$\sum_k^N w_k \phi(\|\mathbf{x}_i - \mathbf{x}_k\|) + g(\mathbf{x}_i) = f_i, \text{ for } 1 \le i \le n,$$

$$\sum_k^N w_k h(\mathbf{x}_k) = 0, \text{ for all } h \in \Pi_m^p.$$

The second condition is unwieldy since it represents an infinite set of equations (one for each polynomial in Π_m^p). Fortunately, since Π_m^p is a vector space and the condition is linear, it is necessary and sufficient for the condition to hold true for each vector in a basis of Π_m^p. This results in a finite number of equations because Π_m^p has a finite number of dimensions.

5.6.2 Design issues

The kernel, ϕ The choice of ϕ is the important decision in designing an RBF reconstruction. It should be guided by broad considerations such as performance, robustness, continuity, and other factors.

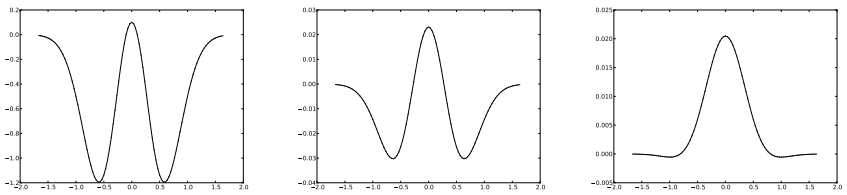


Figure 14: Illustration of ill-conditioning and regularization. From left to right, the regularization parameter is 0, .01, and .1 respectively. Note the vertical scale on the plots.

In term of *performance*, in general training the RBF system is $O(N^3)$ (solving a general $N \times N$ linear system of equations) process and evaluating the reconstruction is $O(N)$. There are algorithms and data structures for speeding-up these computations but one of the most effective performance gain is obtained by using a kernel with a *compact support*. This has the immediate consequence to make the interpolation matrix sparse and to limit the number of samples participating in an evaluation to the samples within the domain. One of the drawback with compact support kernels is that choosing the span of the support can be tricky. Ideally it should be related to the local sample density in order to avoid sampling issues (over-smoothing). If sampling is relatively uniform then compactly support kernels become attractive.

The shape parameter, c For any kernel, ϕ, we can introduce a shape parameter, $c \in \mathbb{R}^+$, by replacing $\phi(d)$ by $\phi(d/c)$. In general this narrows or broadens the kernel, and the effect is particularly significant for compactly supported kernels. The shape parameter can be used to "tune-up" the reconstruction. The scattered data interpolation problem as we stated it in section 2 does not constrain the shape parameter rather it is a free parameter and we must look elsewhere to estimate it. For instance, the shape parameter affects the conditioning of the collocation matrix. As the shape parameter decreases (and the kernel widens), the condition number increases but in some sense the reconstructed function becomes smoother. There have been research efforts [16] to explore algorithms, such as cross-validation, for finding shape parameters that optimize the approximation error.

The use of a broad kernel can cause surprising effects. Fig. 14 shows a section through simple a two-dimensional interpolation problem. The data are zero at the corners of a square, with a single non-zero value in the center. Noting the vertical scale on the plots, the subfigures on the left show wild oscillation that is not motivated by the simple data. In the figures we used regularization to

achieve a more sensible solution. In this approach an original linear system

$$\mathbf{\Phi w} = \mathbf{f}$$

is replaced by the alternate system

$$(\mathbf{\Phi} + \lambda \mathbf{I})\mathbf{w} = \mathbf{f}$$

thereby reducing the condition number of the matrix. This also has the effect of causing the reconstructed curve to approximate rather than interpolate the data (Fig. 13).

Another way of estimating the shape parameter could be to optimize the geometric property of the reconstruction. For instance, one might be interested in minimizing the total curvature of the reconstruction.

The polynomial, g The main role of the polynomial g is to provide polygonal reproducibility. That is to say if $g \in \Pi_m^p$ then the reconstruction can perfectly fit any polygon of degree at most m. This can be a very useful property for many applications. For instance for a 3-D deformation system being able to reproduce affine transformations is useful. Another advantage of including a polynomial might be for extrapolation. Specially in the case where a kernel with compact support is used, the polynomial can be used to model a global trend in the data. In practice, it is rare to use polynomials of degree higher than 1 (linear function) but it is certainly possible. Finally, when using a polynomial term in the reconstruction, care must be taken that there is enough data to fit the polynomial. This is not a problem in the typical case of dealing with polynomials of degree 1.

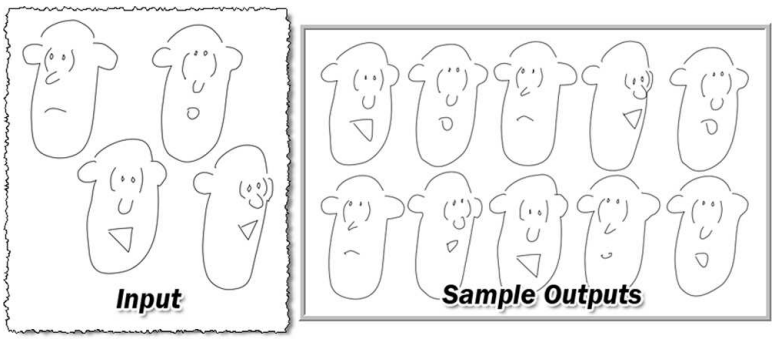
5.6.3 Applications

Mesh deformation. One application of scattered data interpolation is in image-based modeling. The work by Pighin *et al.* [48] describes a system for modeling 3-dimensional faces from facial images. The technique works as follow: after calibrating the cameras, the user selects a sparse set of correspondences across the multiple images. After triangulation, these correspondences yield a set of 3-dimensional constraints. These constraints are used to create a deformation field from a set of radial basis functions. Following their notation, the deformation field f can be written:

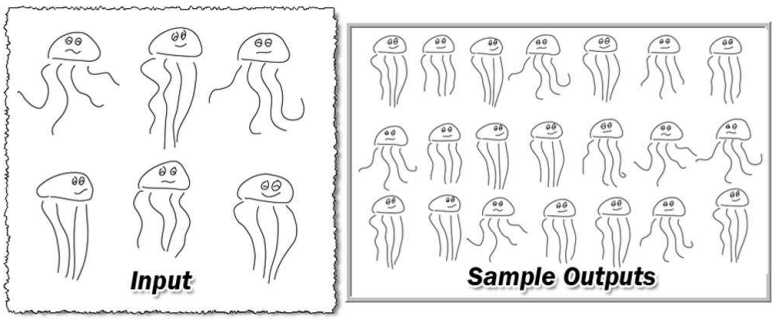
$$f(\mathbf{x}) = \sum_i c_i \phi(\mathbf{x} - \mathbf{x}_i) + M\mathbf{x} + t, ,$$

where ϕ is an exponential kernel. M is a 3×3 matrix and t a vector that jointly represent an affine deformation. The vanishing moment conditions can then be written as:

$$\sum_i c_i = 0 \text{ and } \sum_i c_i \mathbf{x}_i^T = 0$$



(a) cartoon face



(b) jellyfish drawing

Figure 15: Examples of doodles by [3]. The left images were drawn by an artist; the right images were synthesized using the proposed technique.

This can be simplified to just $\sum_i c_i \mathbf{x}_i^T = 0$ by using the notation that $\boldsymbol{x} \equiv \{1, x, y, z\}$.

Learning doodles by example. Baxter and Anjyo [3] proposed the concept of a latent doodle space, a low-dimensional space derived from a set of input doodles, or simple line drawings. The latent space provides a foundation for generating new drawings that are similar, but not identical to, the input examples, as shown in Figure 15. This approach gives a heuristic algorithm for finding stroke correspondences between the drawings, and then proposes a few latent variable methods to automatically extract a low-dimensional latent doodle space from the inputs. Let us suppose that several similar line drawings are resampled by (1), so that each of the line drawings is represented as a feature vector by combining all the x and y coordinates of each point on each stroke into one vector. One of the latent variable methods in (2) then employs PCA and thin plate spline RBF as follows. We first perform PCA on these feature vectors, and form the latent doodle space from the first two principal components. With the thin plate spline RBF, we synthesize new drawings from the

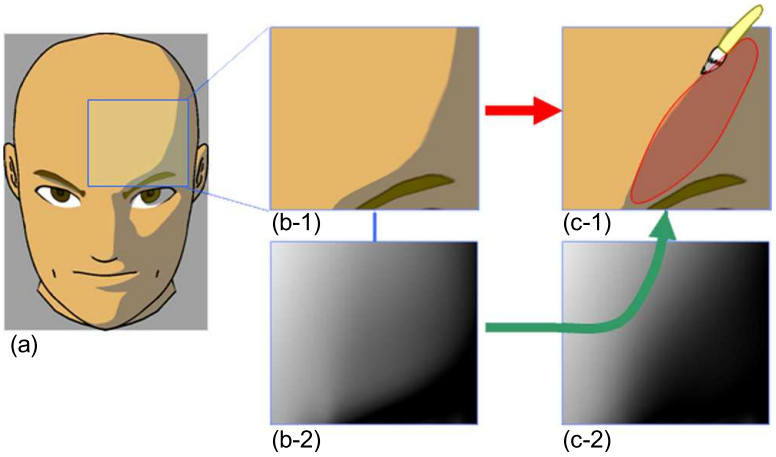


Figure 16: The interpolation problem in [57]. (a) is the original cartoon-shaded face; (b-1) is the close-up view of (a) and (b-2) is the corresponding intensity distribution (continuous gradation); (c-1) is the painted dark area and (c-2) is the corresponding intensity distribution that we wish to obtain.

latent doodle space. The drawings are then generated at interactive rates with the prototype system in [3].

Locally controllable stylized shading. Todo et al. [57] proposes a set of simple stylized shading algorithms that allow the user to freely add localized light and shade to a model in a manner that is consistent and seamlessly integrated with conventional lighting techniques. The algorithms provide an intuitive, direct manipulation method based on a paint-brush metaphor to control and edit the light and shade locally as desired. For simplicity we just consider the cases where the thresholded Lambertian model is used for shading surfaces. This means that, for a given threshold c, we define the dark area $\boldsymbol{A}$ on surface $\boldsymbol{S}$ as being $\{\boldsymbol{p} \in \boldsymbol{S} | d(\boldsymbol{p}) \equiv \langle L(\boldsymbol{p}), N(\boldsymbol{p}) \rangle < c\}$, where $L(\boldsymbol{p})$ is a unit light vector, and $N(\boldsymbol{p})$ is the unit surface normal at $\boldsymbol{p} \in \boldsymbol{S}$. Consider the cartoon-shaded area as shown in Figure 16, for example. Then let us enlarge the dark area in Figure 16 (a) using the paint-brush metaphor, as illustrated in (c-1) from the original area (b-1). Suppose that the dark area $\boldsymbol{A}'$ enlarged by the paint operation is given in the form: $\boldsymbol{A}' = \{\boldsymbol{p} \in \boldsymbol{S} | f(\boldsymbol{p}) < c\}$. Since we define the shaded area by the thersholded Lambertian, we'd like to find such a continuous function f as described above. Instead of finding f, let us try to get a continuous function $o(\boldsymbol{x}) := d(\boldsymbol{x}) - f(\boldsymbol{x})$, which we may call an *offset* function. The offset function may take 0 outside a neighborhood of the painted area. More precisely, let U be an open neighborhood of the painted area C. The desired function $o(\boldsymbol{x})$ should

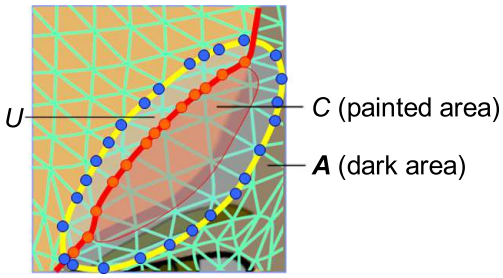


Figure 17: The constraint points for the interpolation problem in Figure 16. The triangle meshes cover the shaded face in Figure 16 (c-1). $\boldsymbol{A}$ is the initial dark area, whereas C and U denote the painted area and its neighborhood, respectively. U is the area surrounded by the closed yellow curve. The blue or red dots mean the boundary points $\{\boldsymbol{x}_i\}$ of these areas.

Figure 18: Toon-shaded 3D face in animation. *left*: with a conventional shader; *right* : result by [57].

then be a continuous function satisfying:

$$o(\boldsymbol{x}) = \begin{cases} 0 & , \text{ if } \boldsymbol{x} \in \partial U \cup (\partial A - C) \\ c - d(\boldsymbol{x}) & , \text{ if } \boldsymbol{x} \in U - \partial(A \cup C). \end{cases} \tag{5}$$

To get $o(\boldsymbol{x})$, we *discretize* the above condition (5), as shown in Figure 17. Suppose that $\boldsymbol{S}$ consists of small triangle meshes. Using a simple linear interpolation method, we get the points $\{\boldsymbol{x}_i\}$ on the triangle mesh edges (or as the nodes) which represent the constraint condition (5) to find an RBF f in the form:

$$f(x) = \sum_{i=1}^{l} w_i \phi(\boldsymbol{x} - \boldsymbol{x}_i) + p(\boldsymbol{x}), \tag{6}$$

where $\phi(\boldsymbol{x}) = \|\boldsymbol{x}\|$. This means that we find f in (6) under the condition (5) for $\{\boldsymbol{x}_i\}_{i=1}^{l}$ as a function defined on $\boldsymbol{R}^3$, rather than on $\boldsymbol{S}$. We consequently set $o(\boldsymbol{x}) := f(\boldsymbol{x})$. Figure 18 demonstrates the painted result. Just drawing a single image, of course, would not need the continuous function f. The reason why we need such an RBF technique as described above is that we wish to make the light and shade animated. This can be performed using a simple linear interpolation of the offset functions at adjacent keyframes.

Example-based skinning. Kurihara and Miyata [33] introduced the weighted pose-space deformation algorithm. This general approach interpolates the skin of character as a function of the degrees of freedom of the pose of its skeleton (Fig. 19). Particular sculpted or captured example shapes are located at particular poses and then interpolated with a scattered intepolation technique.

In the case of [33], the shapes were captured using a medical scan of one of the authors' hands, resulting in very plausible and detailed shapes (Fig. 20). The video accompanying the paper shows that the interpolation of these shapes as the fingers move is also very realistic.

Kurihara and Miyata used a cardinal interpolation scheme, normalized radial basis. In this scheme, for n sample shapes there are n separate radial basis interpolators, with the kth interpolator using data that is 1 at the k training pose and 0 at all the other poses. The RBF matrix is based on the distance between the poses (in pose space) and so has to be formed and inverted only once.

Another important development in this paper is the way it uses skinning weights to effectively determine a separate pose space for each vertex. That is, the distance between two poses is defined as

$$d_j(\boldsymbol{x}_a, \boldsymbol{x}_b) = \sqrt{\sum_{k}^{n} w_{j,k} (\boldsymbol{x}_{a,k} - \boldsymbol{x}_{b,k})^2}$$

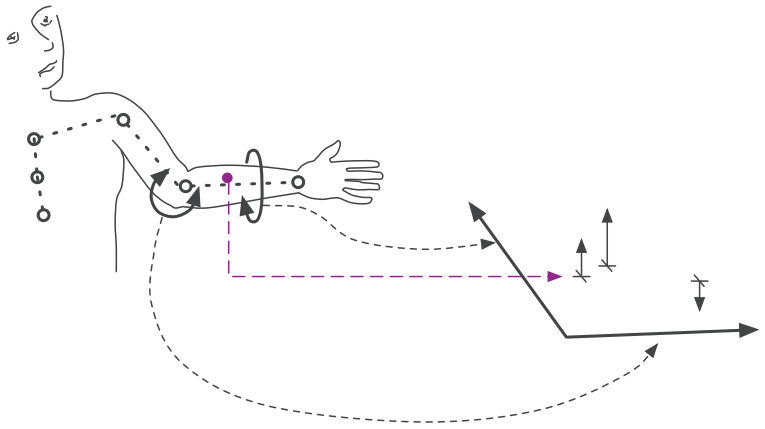


Figure 19: Scattered interpolation of a creature's skin as a function of skeletal pose.

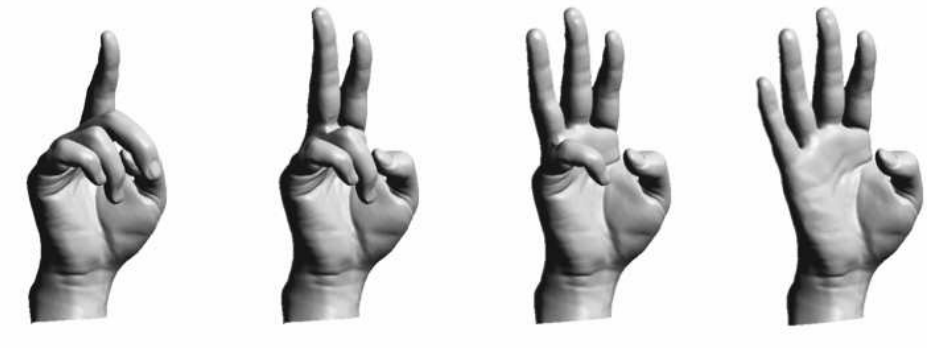


Figure 20: Hand poses synthesized using weighted pose space deformation (from [33]).

where $w_{j,k}$ are the skinning weights for the kth degree of freedom for the jth vertex, and $\boldsymbol{x}_{a,k}$ denotes the kth component of location $\boldsymbol{x}_a$ in the pose space.

5.6.4 Computational considerations

We recall from this section that the RBF reconstruction with a polynomial term is determined by two sets of equations:

$$\sum_k^N w_k \phi(\|\mathbf{x}_i - \mathbf{x}_k\|) + g(\mathbf{x}_i) = f_i, \text{ for } 1 \leq i \leq n,$$

$$\sum_k^N w_k h(\mathbf{x}_k) = 0, \text{ for all } h \in \Pi_m^p.$$

To better understand these equations, we choose a basis of Π_m^p, $\{q_1, \ldots, q_l\}$, with $l = dim(\Pi_m^p)$ and let, $\mathbf{a} = \{a_1, \ldots, a_l\}$, be the expansion of g on this basis (i.e. $g(\mathbf{x}) = \sum_j^l a_j q_j(\mathbf{x})$). Thus the previous equations become:

$$\sum_k^N w_k \phi(\|\mathbf{x}_i - \mathbf{x}_k\|) + \sum_j^l a_j q_j(\mathbf{x}_i) = f_i, \text{ for } 1 \leq i \leq n,$$

$$\sum_k^N w_k q_j(\mathbf{x}_k) = 0, \text{ for } 1 \leq j \leq l.$$

We can rewrite these in matrix form as:

$$\begin{bmatrix} \Phi & Q \\ Q^t & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix}$$

where $\Phi_{i,j} = \phi(\|\mathbf{x}_i - \mathbf{x}_j\|)$, $Q_{i,j} = q_j(\mathbf{x}_i)$, $\mathbf{w} = \{w_1, \ldots, w_N\}$, and $\mathbf{f} = \{f_1, \ldots, f_N\}$.

From a computational point of view the RBF reconstruction requires solving a linear system of equations with the matrix:

$$A = \begin{bmatrix} \Phi & Q \\ Q^t & 0 \end{bmatrix}$$

A is called the interpolation or collocation matrix. From here on, we will study this matrix to answer two questions. First, we would like to know if the system of equations admits a solution and if it is unique. Second, we would like to use to use an efficient procedure to estimate the solution and a procedure that would ideally scale well with large number of samples.

Well-posedness

We would like to know when this system of equations has a solution and if it is unique. The first think to notice about A is that it is square (the number of equations equals the number of unknowns) and symmetric ($A^t = A$). Linear algebra tells us that a square system of equations always has a solution. If

the matrix is non-invertible or rank-deficient, the system has multiple solutions that form a vector space. But in practice we would like the system to have a single solution (in such case we would qualify the problem as being "well-posed") because if there is more than a single solution it means that some of the data is not being used (we could drop some rows and columns of the matrix and still get a solution).

Determining the uniqueness of the solution (or the well-posedness of the problem) depends on three quantities:

- The position of the samples $\{\mathbf{x}_i\}$. Clearly if there is redundancy in the data, the system will be rank-deficient.
- The kernel ϕ.
- The degree, m, of the polynomial.

Unfortunately, necessary and sufficient conditions to characterize this general non-singular case are still open. If we restrict the kernel, ϕ, to a particular space of function then more can be said. One such restriction is the space of kernels that produce symmetric definite matrices. Roughly speaking, it can be shown that these kernels always produce invertible interpolation matrices assuming that the sample sites do not lie on a polynomial of degree less than m [10].

Alternately, one way to circumvent uniqueness issues is to cast the problem where we seek to minimize the squared norm of $[\mathbf{w}^t \ \mathbf{a}^t]^t$ subject to a set of linear constraints. This yields an objective function that is a quadratic form. This problem is guaranteed to have a unique solution (convex objective function).

Estimating the solution

From a numerical point of view, solving the system depends on two quantities:

- The *fill distance* [60]. The fill distance expresses how well the data, D, fills a region of interest Ω:

  $$h_{D,\Omega} = \sup_{\mathbf{x}\in\Omega} \min_{\mathbf{x}_j\in D} \|\mathbf{x} - \mathbf{x}_j\|_2.$$

  It is the radius of the largest "data-site free ball" in Ω. The fill distance can be used to bound the *approximation error*.

- The *separation distance* [60]. The separation distance, q_D, measures how close the samples are together. If two data sites are very close then the interpolation matrix will be near-singular.

  $$q_D = \frac{1}{2} \min_{j\neq k} \|\mathbf{x}_j - \mathbf{x}_k\|_2.$$

  It can be shown that the minimum eigenvalue of A, hence its condition number, is related to the separation distance:

  $$\lambda_{min}(A) \geq C q_D^{2\tau-d}$$

The matrix A is usually not sparse and for large number of samples tends to be ill-conditioned (if the density of the samples augment the separation distance will likely decrease).

Stable procedures (e.g. QR or SVD decomposition) for solving such system are of time complexity $O((N+l)^3/6)$ with order $O((N+l)^2)$ in storage. As the size of the training set grows this quickly becomes prohibitive. There are a several venues for addressing this scalability issue.

For instance Beastson *et al.* [5] propose a numerical approach combining an iterative solver and a preconditioner. The preconditioner is a matrix representing a change of coordinate system. The main idea is that in a different coordinate system the system of equations requires fewer iterations to obtain a good estimate of the solution. As a solver, they recommend using the generalized minimal residual method (GMRES).

The most common approach is to perform a far field expansion of the kernel ϕ. This method called fast multipole expansion [62] allows speeding up the computations of $\sum_k^N w_k \phi(\|\mathbf{x}-\mathbf{x}_k\|)$ by splitting the kernel into sums of separable functions.

In the domain decomposition method [4] the data set is subdivided into several smaller data sets and the interpolations equations are solved iteratively. This technique has the advantage of allowing parallel computations.

A different approach is to process the data incrementally: Start from a small sub-set and leave the rest for refining or validating the approximation. Hon *et al.* [29] offers such a greedy approach.

Most of these techniques would introduce errors in the computation of the solution but since the reconstruction is an approximation of the true function, these errors might be acceptable.

5.7 Scattered interpolation on meshes: Laplace, Poisson, Thinplate

The previous subsection on radial basis methods mentioned choices of kernels for using RBFs to produce thin-plate interpolation. There is an equivalent formulation that does not use RBFs. This formulation is a boundary value problem where the differential equation represents a penalty on roughness and the boundary conditions are the known values of the interpolation problem. In practice, this results in a single linear system solve for the unknowns via discretization of the Laplacian operator on a mesh. In this *mesh-based* approach the problem domain is often on a regular grid, so initially this may not seem like scattered interpolation. However, the unknowns can be scattered at arbitrary sites in this grid, so it is effectively a form of scattered interpolation in which the locations are simply quantized to a fine grid. In addition, forms of the

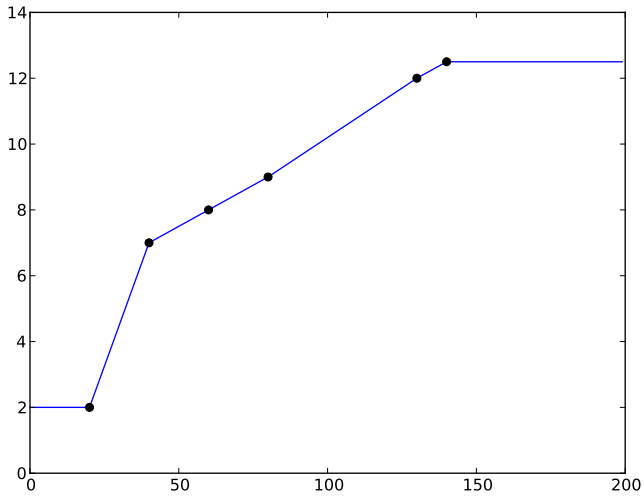


Figure 21: Laplace/Poisson interpolation in one dimension is piecewise linear interpolation. Note that this figure was created by solving the appropriate linear system (rather than by plotting the expected result).

Laplacian (Laplace Beltrami) operator have been devised for triangular meshes [26], allowing scattered interpolation on irregular geometry.

Scattered interpolation of sites on regular meshes is commonly used in the work on image editing, in particular e.g. [47], and operations with irregular mesh (manifold) Laplacians are also finding frequent use in mesh deformation (e.g. [7]).

The connection between the mesh-based Laplacian interpolation and RBFs is through the definition of the thin-plate. Specifically, the Laplace or Poisson equation seeks to minimize the integral of the squared first derivative over the domain, with the solution (via the calculus of variations) that the second derivative is zero:

$$\min_f \int \|\nabla f\|^2 d\Omega \quad \Rightarrow \quad \nabla^2 f = 0$$

Similarly, the thin plate seeks to minimize the integral of the squared second derivative over the domain, with the solution that the fourth derivative is zero:

$$\min_f \int (\frac{d^2}{dx^2} f(x,y))^2 + (\frac{d^2}{dy^2} f(x,y))^2 \, dxdy \quad \Rightarrow \quad \nabla^2 f = 0$$

The Laplace equation can be solved via finite differences, resulting in a linear

system. The finite-difference approximation for the second derivative is:

$$\frac{d^2 f}{dx^2} \approx \frac{1}{h^2}\left(1 \cdot f(x+1) - 2 \cdot f(x) + 1 \cdot f(x-1)\right)$$

The weight stencil $(1, -2, 1)$ is important. If $f(x)$ is digitized into a vector $f = [f[1], f[2], \cdots, f[m]]$ then the second derivative can be approximated with a matrix expression

$$Lf \propto \begin{bmatrix} -1 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & 1 & -2 & 1\cdots \\ & & \cdots & & \end{bmatrix} \begin{bmatrix} f[1] \\ f[2] \\ f[3] \\ \vdots \end{bmatrix}$$

In two dimensions the corresponding finite difference is the familiar Laplacian stencil

$$\begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix} \tag{7}$$

These weights are applied to the pixel sample $f(x, y)$ and its four neighbors $f(x, y-1), f(x-1, y), f(x+1, y), f(x, y+1)$. A two-dimensional array of pixels is in fact regarded as being "vectorized" into a single column vector $f(x, y) \equiv f_k$ with $k = y \times \text{xres} + x$.

Regardless of dimension the result is a linear system of the form $Lf = 0$, with some additional constraints that specify the value of $f(x_k)$ at specific positions. When the constraints are applied this becomes an $Ax = b$ system rather than a $Ax = 0$ nullspace problem.

In matrix terms the corresponding thin-plate problem is simply

$$L^2 f = 0$$

where (again) some entries of f are known (i.e. constrained) and are pulled to the right hand side.

In both cases the linear system is typically huge, with the number of equations being the number of unknown pixels in the image inpainting case (this can easily be 10,000 or more). On the other hand, the matrix is sparse, so a sparse solver can (and must) be used. The Laplace/Poisson equations are also suitable for solution via multigrid techniques, which have time linear in the number of variables. A Python implementation of Laplace interpolation can be found in appendix A.

5.7.1 Mesh-based vs. meshless methods

There is a fundamental difference between mesh-based (e.g. Laplacian) and meshless (e.g. RBFs) methods. It has to do with the space in which the distance/influence in sampling space is measured. With a meshless method the

distance is measured in the input space of the reconstruction. With a mesh-based method the distance or influence is measured following the geometry of the mesh.

Let us illustrate this through a practical example. Imagine we have recorded some motion data around the mouth of a performer for the purpose of lip-synching a digital clone. Now we face the problem of deforming the lips of the facial mesh according to the mocap data. If we apply an RBF-based solution, the motion recorded on the upper-lip will influence the lower-lip geometry far too much especially if the lips are close. This is because we measure influence in 3-space regardless of the mesh or face geometry. If we apply a Laplace-style method, the motion bleeding will be much less noticeable because the influence of the sample gets propagated along the mesh not in 3-space.

Based on this example it would seem that mesh-based methods are always preferable. This is often true but it does assume that a "mesh" is available. For most data sets we do not have a mesh even though we might suspect that the samples have some structure or (in mathematical terminology) belong to a manifold. Finding a good parameterization of the data so that its properties are preserved via interpolation is one of the most important and difficult issues in scattered data interpolation. Section 8 gives a brief overview of scattered data interpolation on manifolds.

5.8 Comparison and Performance

Several general statements about the performance of the various methods can be made.

- Radial Basis and Wiener interpolation both generally require solving a linear system to determine the weights w_k. This typically requires $O(N^3)$ time (though the matrix is symmetric so Choleski can be used), but this can be done as a setup or "training" precomputation.
- Shepard's method is simple and has no setup or training cost. On the other hand it provides rather poor interpolation.
- All methods except for moving least squares have a runtime cost that is linear in the number of data points.
- Moving least squares have no training cost, but require solving a linear system at runtime. On the other hand this system may involve a small subset of the total number of data points.
- For the roughness penalty methods that have equivalent RBF and relaxation formulations, the RBF approach scales with the number of known data points, whereas the direct (Laplace/Poisson) approach scales with the number of unknown points (i.e. locations to be synthesized). Laplace/Poisson interpolation is poor (e.g. it is piecewise linear in the one dimensional case).

The iterated Laplacian provides better interpolation but is more difficult to solve with fast (multigrid) methods.

6 Where do RBFs come from?

In the previous section we saw that Radial Basis Functions are a simple and versatile method for interpolating scattered data, generally involving only a standard linear system solve to find the interpolation weights, followed by interpolation of the form

$$f(\mathbf{x}) = \sum_k w_k \phi(||\mathbf{x} - \mathbf{x}_k||)$$

where $(\mathbf{x})$ is the resulting interpolation at point $\mathbf{x}$, and ϕ is a radially symmetric "kernel" function.

Several choices for the kernel were mentioned, such as $\exp(-r^2/\sigma^2)$ and some more exotic choices such as $r^2 \log r$.

Where do these kernels come from and how should you choose one? This section provides the theory to answer this question.

6.1 What kernels will interpolate?

Conditions for the choice of kernel are given in [10], p. 100, but the discussion there involves functional analysis and is more technical than needed in this section. Roughly speaking, strictly monotonic kernels work well.

This becomes evident by considering the solution for the weights,

$$\begin{bmatrix} \phi_{1,1} & \phi_{1,2} & \phi_{1,3} & \cdots \\ \phi_{2,1} & \phi_{2,2} & \cdots & \\ \phi_{3,1} & \cdots & & \\ \vdots & & & \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \end{bmatrix}$$

where $\phi_{a,b}$ denotes $\phi(\|\mathbf{x}_a - \mathbf{x}_b\|)$. We can imagine choices of ϕ that will result in the matrix above having identical rows, and thus prevent finding suitable weights. To the extent that ϕ is monotonic, however, the rows of the R matrix will be distinct.

6.2 Kernels, Differential Operators, and "roughness penalties"

Although Laplace and thin-plate interpolation is usually done by either sparse linear solves or relaxation/multigrid, it can also be done by radial basis interpolation, and this is faster if there are relatively few points to interpolate.

In these cases the kernel ϕ is the "Green's function of the corresponding squared differential operator". This section will explain what this means, and give an informal derivation for simplified case. Specifically we're choosing a a discrete one-dimensional setting with uniform sampling, so the problem can be expressed in linear algebra terms rather than with calculus.

The goal is find a function f that interpolates some scattered data points d_k while simultaneously minimizing the roughness of the curve. The roughness is measured in concept as

$$\int |Df(x)|^2 dx \tag{8}$$

where D is a "differential operator".

Whereas a "function" takes a single number and returns another number, an "operator" is something that takes a whole function and returns another whole function. The derivative is a prototypical operator, since the derivative of a function is another function. A differential operator is simply an operator that involves some combination of derivatives, such as such as $\frac{d^2}{dx^2}$ for example.

We're working discretely, so D is a matrix that contains a finite-difference version the operator. For example, for the second derivative, the finite difference is

$$\frac{d^2}{dx^2} \approx \frac{1}{h^2}\left(f_{t+1} - 2f_t + f_{t-1}\right)$$

This finite difference has the weight pattern $1, -2, 1$, and for our purposes we can ignore the data spacing h by considering it to be 1, or alternately by folding it into the solved weights.

The finite difference version of the whole operator can be expressed as a matrix as

$$D = \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & 1 & -2 & 1\cdots \\ & & \cdots & & \end{bmatrix}$$

The discrete equivalent of the integral (8) is then $\|Df\|^2 = f^T D^T D f$.

Then our goal (of interpolating some scattered data while minimizing the roughness of the resulting function) can be expressed as

$$\min_f \|Df\|^2 + \lambda^T S(f - d).$$

$$= \min_f f^T D^T D f + \lambda^T S(f - d). \tag{9}$$

S is a "selection matrix", a wider-than-tall permutation matrix that selects elements of f that correspond to the known values in d. So for example, if f is a vector of length 100 representing a curve to be computed that passes through

5 known values, S will be 5×100 in size, with all zeros in each row except a 1 in the element corresponding to the location of the known value. The known value itself is in d_k, a vector of length 5. λ is a Lagrange multiplier that enforces the constraint.

Now take the derivative of Eq. (9) with respect to f,

$$\frac{d}{df} = 2D^T Df + S^T \lambda$$

and we can ignore the 2 by folding it into λ.

If we know λ, the solution is then

$$f = (D^T D)^{-1} S^T \lambda \tag{10}$$

- Although continuously speaking the differential operator is an "instantaneous thing", in discrete terms it is a convolution of the finite difference mask with the signal. Its inverse also has interpretation as a convolution.
- if D is the discrete version of $\frac{d}{dx}$ then $D^T D$ is the discrete Laplacian, or the "square" of the original differential operator. Likewise if D is the discrete Laplacian then $D^T D$ will be the discrete thin plate, etc.
- λ has the same size as d, so $S^T \lambda$ is a vector of discrete deltas of various strengths. In the example $S^T \lambda$ has 5 non-zero values out of 100.

6.3 Green's functions

Earlier we said that the kernel is the "Green's function of the corresponding squared differential operator".

A *Green's function* is a term from differential equations. A linear differential equation can be abstractly expressed as

$$Df = b$$

where D is a differential operator as before, f is the function we want to find (the solution), and b is some "forcing function". In the Greens function approach to solving a differential equation, the solution is assumed to take the form of a convolution of the Green's function with the right hand side of the equation (the forcing function). (We are skipping a detail involving boundary conditions, but the statement is essentially correct). For linear differential equations the solution can always (in theory) be expressed in this form.

The Green's function G is the "convolutional inverse" of the squared differential operator. It is the function such that the operator applied to it gives the delta function,

$$DG = \delta \tag{11}$$

While the theory of Green's functions is normally expressed with calculus, we'll continue to rely on linear algebra instead. In fact, the appropriate theory has already been worked out in the previous subsection and merely needs to be interpreted.

Specifically, in Eq. (10), $(D^T D)^{-1}$ is the Green's function of the squared differential operator. In discrete terms Eq. 11 is

$$D^T D(D^T D)^{-1} = I$$

with the identity matrix rather than the δ function on the right hand side. And $S^T\lambda$ is a sparse set of weighted impulses that is "convolved" (via matrix multiplication) with the Green's function. Thus we see that Eq. 10 is the Green's function solution to the original "roughness penalty" goal.

7 Modeling physical systems with scattered data interpolation

Scattered data interpolation techniques have also found many applications in solving physics problems. There are generally three ways SDI techniques can come into play for solving physical problems. First, it could well be we are dealing with sampled data that represents physical quantities such as velocity. These quantities often obey physical properties that have to be maintained during interpolation. Second, scattered data interpolation can be used within a physical simulation as a way of expressing a physical function or field and its derivatives from a set of values at specific locations. Finally, scattered data interpolation can also be used when simulating coupled systems with fluid-structure interaction.

7.1 Interpolating physical properties

Many quantities in physics are expressed as vector fields (velocity, magnetic field). A vector field can be represented as a mapping from $\mathbb{R}^3$ to $\mathbb{R}^3$. The domain does not have to be $\mathbb{R}^3$ but could well be a manifold embedded in $\mathbb{R}^3$ (e.g. a surface) but we leave this case to the next section.

A case that is of particular interest in computer graphics is the situation where the data stems from an incompressible fluid, i.e. a fluid having neither sources nor sinks. In such case, we say that the fluid has a velocity field $\mathbf{v}$ that is divergence-free if:

$$\nabla \cdot \mathbf{v} = \frac{\partial v}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial v}{\partial z} = 0$$

This property is not trivial to maintain during a simulation.

If the velocity was sampled from a divergence-free phenomenon then the interpolated values should be divergence-free as well. RBF-based interpolants can be constructed to be divergence-free. One examples of these are called matrix-valued radial basis functions [39]. The procedure for constructing such an interpolant can be explained in the following way: Let us assume a scalar-valued function, ϕ, we can form the matrix valued function:

$$\Phi_{div} := (\Delta I + \nabla\nabla^t \phi),$$

where Δ is the gradient operator and $\nabla\nabla^t$ (or ∇^2) is the laplacian operator. One can prove that Φ_{div} is a matrix-valued function with divergence-free columns. Although, Φ_{div} , is not an RBF, if ϕ is an RBF, we can build divergence-free interpolants:

$$\tilde{\mathbf{f}}(\mathbf{x}) = \sum_{k}^{N} \Phi_{div}(\mathbf{x} - \mathbf{x}_k) w_k,$$

where $\{w_k\}$ is a set of weight vectors that can be computed from the following constraints:

$$\tilde{\mathbf{f}}(\mathbf{x}_i) = \sum_{k}^{N} \Phi_{div}(\mathbf{x}_i - \mathbf{x}_k) w_k = \mathbf{f}_i, \text{ for } 1 \leq i \leq n.$$

Recent work by H. Wendland [61] has shown that divergence-free kernels can be used to solve stationary Navier-Stokes problems.

Another classical property of physical fields is to be curl-free. For instance this is the case for the gravity field. There are techniques for interpolating curl-free phenomena. One way of doing this is to perform a Helmholtz-Hodge decomposition on the kernel to obtain a divergence-free and a curl-free component [25]. E.J. Fuselier [24] also proposes curl-free matrix-valued RBFs.

7.2 Scattered data interpolation in mesh-free methods

The simulation of physical phenomena often require solving partial differential equations. It is impossible to solve these equations except for the simplest problems, as a result discretization techniques are used to perform the computations using a discrete number of locations. Methods such as the finite elements method use a mesh to represent the computational volume. The physical unknowns are estimated at each node in the mesh. These mesh-based or Eulerian methods are well-suited when the volume of the simulation is well defined (e.g. the interior of a piston). For open system such as splashing fluid, mesh-free, particle or Lagrangian methods are better suited. In a mesh-free approach the computation volume is not defined by a static mesh but rather the medium is represented by a set of moving particles. In this framework, running the

simulation involves computing the trajectory of the particles and their physical properties (velocity, temperature, etc.) through time. Scattered data interpolation naturally comes into play in mesh-free methods from the need to interpolate simulated values form the particles to the entire computation volume.

Mesh-free methods can be more computationally demanding than their Eulerian counterparts, also handling boundary conditions is more difficult.

In what follows we briefly look at the use of radial basis functions and moving least-squares in mesh-free methods.

7.2.1 RBF-based mesh-free methods

Radial basis functions can also be used to solve partial differential equations (PDE) [21, 15, 30]. RBF based methods are collocation methods that proceeds by choosing a finite-dimensional space (usually, polynomials up to a certain degree) and a number of points or sites in the domain (called collocation points). The goal is then to select that solution which satisfies the given equation at the collocation points. Solving PDEs through radial basis functions is a fairly new approach. A key advantage of using RBF is that, unlike Eulerian-type techniques, it does not require a grids. As such these techniques fall within the category of mesh-free or mesh-less techniques. They have been applied to linear elliptic PDEs, time-dependent problems, and to non-linear problems.

Let us consider the case of an elliptic partial differential equations. A standard formulation could be:

$$\begin{aligned} Lu &= f, \text{ in } \Omega \\ u &= g, \text{ in } \partial\Omega \end{aligned}$$

where L is a differential operator, Ω is the domain and interest, and $\partial\Omega$ is the boundary of the region. The first set of equations represent a law of evolution and the second are boundary conditions. The vector-field quantity u can be approximated using radial basis functions, u_ϕ, for instance:

$$u_\phi(\mathbf{x}, t) = \sum_k^N w_k(t)\phi(\|\mathbf{x} - \mathbf{x}_k\|) + g(\mathbf{x}), \text{ with } \mathbf{x} \in \Omega.$$

Note that we assumed u_ϕ to be a function of time, t but does not need to. We can bring the approximation, u_ϕ, in the original equations:

$$\sum_{k}^{N} w_k(t) L\phi(\|\mathbf{x}_i - \mathbf{x}_k\|) + Lg(\mathbf{x}_i) = f_i, \text{ for } 1 \leq i \leq n,$$

$$\sum_{k}^{N} w_k(t) \phi(\|\mathbf{x}_i - \mathbf{x}_k\|) + g(\mathbf{x}_i) = g_i, \text{ for } 1 \leq i \leq q,$$

$$\sum_{k}^{N} w_k(t) h(\mathbf{x}_k) = 0, \text{ for all } h \in P_m^p.$$

This results in a linear system of equations that is traditionally solved using a preconditioned iterative solver. In this framework, the sites $\{\mathbf{x}_i\}$, are called collocations sites. The idea is to choose a finite-dimensional space of candidate solutions (usually, polynomials up to a certain degree) and a number of points in the domain (called collocation points), and to select that solution which satisfies the given equation at the collocation points.

RBF collocation methods have been shown numerically (and theoretically) to be very accurate even for a small number of collocation points.

7.3 Fluid-structure interaction

Simulating a system with fluid-structure interaction requires using two different simulators. For instance, if we are interested in studying the behavior of an aircraft wing, we need to simulate the flow around the wing perhaps using a grid-based Navier-Stokes solver. Part of the grid will be modeling the exterior surface of the wing. We also need to simulate the structural deformations of the wing using structural mechanics with a different mesh. The simulator would alternate between a computational fluid dynamic and a computational structural dynamics solvers. When switching solver loads must be transferred from a mesh to the other and the CFD mesh needs to be deformed repeatedly.

Along these lines, Jakobson describes a practical method for wing optimization [31].

8 Scattered data interpolation on a manifold

So far we have assumed that our sample space is a subset of R^p without considering any topological information. Indeed this is the best we can do if there is no topological information available [1]. For many applications though we have

[1] A different approach could be to use a manifold learning technique to recover some topological information as well as approximating the data but this is beyond the topics of these notes.

some information about the topology of the domain. For instance, if we are dealing with BRDF samples, we know these should be distributed on a sphere. Or if we are dealing with orientation, we know these should belong to the space $SO(3)$. It is always possible to perform scattered data interpolation without taking topological information into account but the penalty for doing so can be severe, for instance interpolating two rotations might produce a non-rigid transformation.

Computer graphics applications give rise to far more complex manifold than the 3- or 4 -dimensional sphere. For instance, consider the space of all possible configurations of a facial mesh or the set of all possible images of an object.

Because of the practical importance of spherical data, we treat the well studied case of the sphere before moving on to the more general case.

8.1 Interpolating on a manifold

We do not pretend to provide a mathematically accurate presentation of interpolation theory or techniques on a manifold, rather we strive to provide some intuitions.

Manifolds are mathematical objects that have properties similar to surfaces. In general, a manifold is a space for which at every point, we can find a neighborhood that can be mapped to R^p in a continuous and bijective way (both properties are important). These maps are called charts and a collection of charts is called an atlas. This definition infers some string structure on a manifold because we can use the charts to reason in Euclidian space and use the Euclidian norm. Also if we endow our manifolds with some differential structure such as a tangent space, we can define curves over the manifolds and from then derive the notion of shortest path, or geodesic, between two samples. This notion of geodesic is important because if we would like to interpolate smoothly between two samples at constant speed, it would correspond to following a geodesic with run-length parameterization.

The observation then is that by passing through the tangent space and using geodesic distance. it becomes possible to reduce much of the work to Euclidian space. For instance, once can consider so called zonal kernels of the form:

$$\kappa(x, y) = \psi(d(x, y)),$$

where ψ is a real function of a single variable and d is the geodesic distance. Also an interesting result for providing error estimate is the fact that for a differentiable manifold M if (U, φ) is a chart and $\Phi : M \times M \to R$ is a positive definite kernel on M, then:

$$\Psi(u, v) := \Phi(\varphi^{-1}(u), \varphi^{-1}(v)),$$

is a positive definite kernel on $\varphi(U)$.

8.2 Interpolating on the sphere

Interpolating data on the sphere is a rich topic. For a more comprehensive survey of the topic we refer to Fasshauer and Schumaker [17]. A lot of work has been done on polynomial interpolation and more recently on radial basis functions. We restrict our discussion to spherical harmonics and RBFs.

8.2.1 Spherical harmonics

There are multiple ways to define spherical harmonics. For instance, if we define harmonic functions, the functions, f, that satisfy $\Delta f = 0$; a spherical harmonic of degree l is the restriction of a homogeneous harmonic polynomial of degree l to the sphere. So we can build spherical harmonics of order d in terms of cartesian coordinates by considering functions of the form:

$$g(x, y, z) = \sum_{i+j+k=d} a_{i,j,k} x^i y^j z^k, \text{ such that } \Delta g = 0.$$

Then from these functions we can build a basis by extracting linearly independent vectors a. The popularity of spherical harmonics can be explained by their ability to approximate well smooth functions over the sphere.

8.2.2 RBF on the sphere

There are different ways to consider RBF on a sphere. The simplest way is to consider the points to be in the ambient space $\mathbb{R}^3$ and then restrict the evaluation of approximation to the sphere. This would not however respect geodesic distance but this is easily fixable by using the geodesic distance instead of the Euclidian distance in the RBF expression:

$$\tilde{f}(\mathbf{x}) = \sum_{k}^{N} w_k \phi(d(\mathbf{x} - \mathbf{x}_k)).$$

Whether the resulting linear system has full rank or not is a difficult question to answer. But wide classes of functions that yield positive definite systems have been identified.

Another interesting issue is polynomial reproducibility. We can augment an RBF reconstruction with a polynomial term. In spherical context, the polynomial term takes the form of spherical harmonics:

$$\tilde{f}(\mathbf{x}) = \sum_{k}^{N} w_k \phi(d(\mathbf{x} - \mathbf{x}_k)) + \sum_{k} \sum_{l} d_{k,l} Y_{k,l}(\mathbf{x}).$$

8.2.3 Interpolating rotations

Interpolating rotations in R^3 is of great practical importance in computer graphics (e.g. keyframing). The space of rotations is called the special orthogonal group (written $SO(3)$). Its two most common parameterization are in terms of Euler angles or the 3-dimensional projective space (quaternions). The quaternion representation identifies an element A in $SO(3)$ with a vector, ωx, in R^3 such that $Ax = x$ and $\|x\| = 1$. x is the rotation axis and ω the rotation angle (chosen in $[0, \pi]$).

Whereas interpolating between two rotations can be done using the well-known Spherical Linear Interpolation (SLERP) formula, interpolation between an arbitrary number of rotations is less studied. Let us first treat the case of interpolating between two rotations. This is best understood using quaternion. Let us consider p and q two elements of $SOP(3)$ in quaternion form. We have:

$$slerp(p, q, t) = p(p^{-1}q)^t$$

where, p and q are quaternions and t is a parameter ranging from 0 to 1. Or, alternatively:

$$slerp(p, q, t) = \frac{sin((1-t)\theta)}{sin(\theta)}p + \frac{sin(t\theta)}{sin(\theta)}q,$$

where θ is the angle between p and q (i.e. $\cos(\theta) = p \cdot q$).

There are several ad-hoc procedures for interpolating between multiple quaternions [46, 42]. Generally these methods tend to lift the problem onto the tangent space around some average, do the interpolation in Euclidian space, and project back onto the 4-dimensional sphere. This works well for orientations are bundled close together but not so well for rotations that are far apart.

There is some theoretical work on defining positive definite kernels on $SO(3)$ [19] but it has yet to find practical applications.

9 Guide to the deeper theory

This chapter does not aim at developing a rigorous mathematics behind the techniques described in this course, but at giving intuitive meanings to the mathematical concepts that support those techniques. We do hope that this chapter will give a good introduction to learn the mathematics behind the scene more deeply.

Those mathematical concepts come mainly from *functional analysis*, the mathematical field of modern calculus. The modern calculus introduces a variety of function spaces, such as Hilbert space or Sobolev space, while generalizing concepts in classical calculus. As described below, this gives us a theoretical basis for solving the optimization problems or regularization problems of our practical situations.

9.1 Elements of functional analysis

A function classically means a mapping that gives a real value, denoted by $f(x)$, for a given real number x. One of the motivations for generalizing the concept of function was to give a mathematical validation of Dirac delta function, δ, which has following properties:

$$f(0) = \int f(x)\delta(x)dx \tag{12}$$

or

$$f(t) = \int f(x)\delta(t-x)dx$$

for any ordinary function $f(x)$.

Alternatively the Dirac delta function could be expressed as

$$\delta(t) = \begin{cases} +\infty & (t=0) \\ 0 & (otherwise). \end{cases}$$

But this suggests that δ function is not an ordinary function. The Heaviside function $H(t)$ is also well-known (mainly in signal processing) with its derivative being δ function:

$$H(t) = \begin{cases} 1 & (t \geq 0) \\ 0 & (otherwise). \end{cases}$$

However, $H(t)$ is not differentiable nor continuous at $t = 0$ in a classical sense. How can we explain these things in a mathematically correct way? What we need is therefore to give alternative definitions of functions, derivative, and many more technical terms in classical calculus.

9.2 Brief Introduction to the theory of generalized functions

To achieve a new concept of "derivative", we first take a look at the formula, known as *integration by parts*. For simplicity, we consider a one-dimensional case. Then we have:

$$\int_a^b f'(x)g(x)dx = -\int_a^b f(x)g'(x)dx + [f(t)g(t)]_{t=a}^{t=b}. \tag{13}$$

We derive this formula, supposing that both f and g are smooth (differentiable).

Next let us suppose that g vanishes at the boundary (i.e., $g(a) = g(b) = 0$ in (13)). We then have:

$$\int_a^b f'(x)g(x)dx = -\int_a^b f(x)g'(x)dx. \tag{14}$$

Further, what if the above f is not differentiable (for example, $f = \delta$)? The left hand in (14) is therefore meaningless, but the right hand is still valid. Equation (14) might therefore tell us that, instead of taking the derivative of f, we can think of doing so for g. This could be understood if we consider f as a *functional*, rather than a function, as described later.

Function space and functional Keeping the above things in mind, we next introduce a concept of function space. A function space $\Im$ is a collection of functions defined on a certain domain (typically, region Ω in $\boldsymbol{R}^n$). Here are the examples of function spaces:

Function Space Examples:

1. $\mathcal{P}_m := \{P(x)|P(x)$ is a polynomial of at most m-th order[2] $\}$.
2. $C^m(\Omega)$ is the totality of m-th order smoothly differentiable functions on Ω, where $m = 0$ (the totality of continuous functions), $1, 2, \cdots$, or ∞.
3. $C_0^\infty(\Omega)$ is the totality of infinitely many times differentiable functions on Ω with compact support (i.e., each function of this function space vanishes outside a large ball in $\boldsymbol{R}^n$).
4. $L^p(\Omega) := \{f : \Omega \to R \cup \{\pm\infty\} | \int |f(x)|^p dx < \infty\}$[3], where p is a positive number.

A function space $\Im$ is usually treated as a linear topological space. This means that $\Im$ is a vector space, where convergence of a function sequence is defined (see the section on Hilbert space for details). Next recall a function $f : \Omega \to R$. f is then defined on Ω, and gives a real number $f(x)$ when x is specified. Now we consider mapping F from a function space $\Im$ to R. We call the mapping $F : \Im \to R$ a *functional*. A functional F is defined on a function space, and gives a real number $F(\varphi)$ when an element φ of the function space is specified.

Functional Examples:

1. (*Thin plate spline*) Let Ω be a domain in $\boldsymbol{R}^2$. Let $\boldsymbol{B}_2^2(\Omega)$ be the function space defined as:

$$\boldsymbol{B}_2^2(\Omega) := \{\varphi(x) : \Omega \to R \cup \{\pm\infty\} | \frac{\partial^2\varphi}{\partial x_1^2}, \frac{\partial^2\varphi}{\partial x_1 \partial x_2}, \frac{\partial^2\varphi}{\partial x_2^2} \in L^2(\Omega)\}. \tag{15}$$

[2] This space was denoted Π_{m+1} in subsection 5.6.1

[3] Rigorously, dx should be denoted by $d\mu(x)$ with Lebesgue measure μ. But we don't have to consider such mathematical details in these course notes.

To get a thin plate spline curve, we then consider the functional F on $\boldsymbol{B}_2^2$ as:

$$F(\varphi) := \iint_\Omega \left(\left| \frac{\partial^2 \varphi}{\partial x_1^2} \right|^2 + 2 \left| \frac{\partial^2 \varphi}{\partial x_1 \partial x_2} \right|^2 + \left| \frac{\partial^2 \varphi}{\partial x_2^2} \right|^2 \right) dx_1 dx_2. \tag{16}$$

2. An ordinary function f can also be identified with a functional. The functional, denoted by T_f, is defined as

$$T_f(\varphi) := \int_\Omega f(x)\varphi(x)dx, \tag{17}$$

for any φ in a certain function space $\Im$[4].

3. (*Dirac delta function*) Consider the function space $C_0^\infty(\Omega)$, where $0 \in \Omega$. For any element $\varphi \in C_0^\infty(\Omega)$, Dirac delta function is defined as:

$$T_\delta(\varphi) := \varphi(0). \tag{18}$$

We then note that the functionals in (17) and (18) are linear. This means, for instance, that we have:

$$T_f(\alpha\varphi + \beta\psi) = \alpha T_f(\varphi) + \beta T_f(\psi),$$

for any φ, ψ and any $\alpha, \beta \in \boldsymbol{R}$.

We will show that, a (continuous) linear functional is the generalized function which gives the theoretical basis on discussions in this course notes. Before doing this, we need to investigate more about the relation between f and T_f.

Functions as functionals Now go back to T_f, for an ordinary function f. Then we want to identify the function f with the functional T_f. To make it, we should investigate whether the following property holds:

$$T_f = T_g \Leftrightarrow f = g \tag{19}$$

For example, it is easy to see that this property holds, if f is a continuous function on Ω, and if the linear functional T_f is defined on $C_0^\infty(\Omega)$. Moreover we can get this identification (19) for a wider class of functions. Let's skip, however, the mathematical proof of (19) and mathematical details in the background. Rather, what we should recognize now is that *an ordinary function f can be identified with a functional T_f on a certain function space* through (19).

[4] We of course assume $T_f(\varphi)$ in (17) takes a finite value for any φ in $\Im$.

Generalized function and its derivative We thus define a *generalized function* in the following way.

Definition: Let $\Im$ be a function space[5]. Let T be a functional: $\Im \to \boldsymbol{R}$. T is called a *generalized function (or distribution)*, if it satisfies the following conditions:

1. T is linear:

$$T(\alpha\varphi + \beta\psi) = \alpha T(\varphi) + \beta T(\psi), \text{for } \varphi, \psi \in \Im, \alpha, \beta \in \boldsymbol{R}. \tag{20}$$

2. T is continuous:

$$\lim_{n\to\infty} \varphi_n = \varphi \text{ in } \Im \Rightarrow \lim_{n\to\infty} T(\varphi_n) = T(\varphi) \text{ in } \boldsymbol{R}. \tag{21}$$

Next we define *derivative* of a generalized function. The hint of making it again lies in the formula of integration by parts in (14). For simplicity we consider one-dimensional case, taking $\Im = C_0^\infty(\Omega)$. Suppose that f is smooth (differentiable). We then note that equation (14) suggests

$$T_{\frac{df}{dx}}(\varphi) = \int \frac{df}{dx}\varphi dx = -\int f\frac{d\varphi}{dx}dx = -T_f\left(\frac{d\varphi}{dx}\right). \tag{22}$$

It therefore seems natural to define the derivative of the generalized function T, as follows:

Definition: Let T be a generalized function: $\Im \to \boldsymbol{R}$.The derivative of T, denoted by T', is defined by

$$T'(\varphi) = -T(\frac{d\varphi}{dx}). \tag{23}$$

We note that T' itself is also a generalized function. The above definition by (23) is reasonable, because, if we consider the case where T is induced by a smooth function (i.e., $T = T_f$), it follows from (22) that $T'(\varphi) = T_{\frac{df}{dx}}(\varphi)$. In the following T' is sometimes denoted by $\frac{dT}{dx}$.

As an exercise, let us calculate the derivative of Heaviside function $H(x)$ ($= 1$ if $x \geq 0$, and $= 0$, otherwise) in the sense of distribution. Instead of H itself, we therefore consider T_H. Then we can differentiate it as a linear functional.

$$\begin{aligned}\frac{dT_H}{dx}(\varphi) &= -T_H\left(\frac{d\varphi}{dx}\right) = -\int_{-\infty}^{+\infty} h(x)\frac{d\varphi}{dx}dx = -\int_0^\infty \frac{d\varphi(x)}{dx}dx \\ &= -[\varphi(t)]_{t=0}^{t=\infty} = \varphi(0) \equiv T_\delta(\varphi)\end{aligned}$$

[5] $\Im$ could be $C_0^\infty(\Omega), C^\infty(\Omega)$, and other function spaces, which would bring us various generalized functions [54]. However, in our course notes, we mostly set $\Im$ as $C_0^\infty(\Omega)$.

Similarly we can inductively define the n-th order derivative of T.

Definition: The n-order derivative of a generalized derivative of T, denoted by $\frac{d^n T}{dx^n}$ (or by $T^{(n)}$), is defined by

$$\frac{d^n T}{dx^n}(\varphi) := (-1)^n T(\frac{d^n \varphi}{dx^n}). \tag{24}$$

Again $\frac{d^n T}{dx^n}$ is a generalized function. We could therefore say that any generalized function (on $\Im$) is infinitely many times differentiable. Let $\Im'$ be the totality of the generalized functions on $\Im$. We call $\Im'$ the dual space of $\Im$, which again constitutes a vector space.

The dual space $\Im'$ includes δ function, Heaviside function, and regular functions (such as continuous or smooth functions). In practice we can consider most generalized function as being in the form T_f with an ordinary function f. More precisely, if a function f is *locally integrable*[6], T_f is then a generalized function. The regular functions and Heaviside function are locally integrable, while δ function is not. On the other hand, though the definition of Dirac δ in (18) looks a bit artifical, if we *symbolically* use the integral representation like $\int \varphi(x)\delta(x)dx$ instead of $T_\delta(\varphi)$, we still have equation (12) valid in the sense of distribution, which simply means $\int \varphi(x)\delta(x)dx \equiv T_\delta(\varphi) = \varphi(0)$.

Once we get such a mathematical concept of generalized function as described above[7], we can reformulate the problems in classical calculus. For instance the problems of solving ordinary/partial differential equations (ODE/PDE) is described in the following way. Let $P(\xi)$ be an m-th order polynomial with constant coefficients. For a one-dimensional case, $P(\xi) = \sum_{n=0}^{m} a_n \xi^n$. We then define the differential operator $P(\frac{d}{dx})$ as being $P(\frac{d}{dx})u = \sum_{n=0}^{m} a_n \frac{d^n u}{dx^n}$. Similarly, in a multi-dimensional case, we consider $P(\xi) \equiv P(\xi_1, \xi_2, \cdots, \xi_n) =$

$$P(\xi) \equiv \sum_{|\alpha|=\alpha_1+\alpha_2+\cdots+\alpha_n \geq 0}^{m} C_{\alpha_1,\alpha_2,\cdots,\alpha_n} \xi_1^{\alpha_1} \cdot \xi_2^{\alpha_2} \cdots \xi_n^{\alpha_n},$$

where $C_{\alpha_1,\alpha_2,\cdots,\alpha_n}$ are constant coefficients. Then we set

$$P(D) \equiv P\left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \cdots, \frac{\partial}{\partial x_n}\right)$$

[6] This means that $\int_K |f(x)|dx < +\infty$ holds for any compact set (i.e., bounded and closed) K in Ω.

[7] The theory of hyperfunction [51] gives an alternative theoretical basis on calculus. However, it requires algebraic concepts, such as sheaf and cohomology, so that it is not so elementary, compared to the distribution theory [54], which we discuss in these course notes.

as a partial differential operator. If $P(D)$ is a monomial, for instance, like $D^\alpha = \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \cdots \partial x_n^{\alpha_n}}$, we put

$$D^\alpha u := \frac{\partial^{|\alpha|} u}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \cdots \partial x_n^{\alpha_n}}.$$

Classically, for a given function f on Ω, we want to find a (smooth) solution u that satisfies $P(D)u = f$ on Ω[8]. Now this is understood as the following problem:

Solving differential equation in $\mathcal{S}'$:
For a given $F \in \mathcal{S}'$, find a $T \in \mathcal{S}'$ such that $P(D)T = F$.

Of course we may take the given F as an ordinary function f, in the sense that we put $F = T_f$. According to the above formulation, we want to find a solution in the much wider space $\mathcal{S}'$. Even if we can find the solution in $\mathcal{S}'$, it is not easy to see whether the solution can be represented as an ordinary function. The solution would therefore be called a *weak solution*. In addition, when we *differentiate* a generalized function T, we would then refer to $\frac{dT}{dx}$ as the *weak derivative* of T. In this theoretical approach, what we should do first is to assure the existence of the solution, whereas we need a practical solution. So there still exists a gap between the theory and our practical demand. However, it should be noted that *Finite Element Methods* (FEM) and several related approaches have the theoretical basis from the distribution theory.

9.3 Hilbert Space

We first explain pre-Hilbert space. To make it, we need the following definition.

Definition: Let $\boldsymbol{F}$ be a vector space over $\boldsymbol{R}$. The two-term operation, denoted by $\langle\ ,\ \rangle$ (or $\langle\ ,\ \rangle_{\boldsymbol{F}}$), is called the *inner product*, if it satisfies the following conditions:

- $\langle , \rangle$ is symmetric and bilinear:
  $\langle f, g\rangle = \langle g, f\rangle$,
  $\langle \alpha_1 f_1 + \alpha_2 f_2, g\rangle = \alpha_1\langle f_1, g\rangle + \alpha_2\langle f_2, g\rangle$,
  $\langle f, \beta_1 g_1 + \beta_2 g_2, \rangle = \beta_1\langle f, g_1\rangle + \beta_2\langle f, g_2\rangle$
  for any $f, f_1, f_2, g, g_1, g_2 \in \boldsymbol{F}$, and any $\alpha, \alpha_1, \alpha_2, \beta, \beta_1, \beta_2 \in \boldsymbol{R}$.
- $\langle f, f\rangle \geq 0$, for any$f \in \boldsymbol{F}$, and $\langle f, f\rangle = 0$, if and only if $f = 0$.

The vector space with the inner product is called a *pre-Hilbert space*. It is then noted that any pre-Hilbert space $\boldsymbol{F}$ is a normed space with the norm $\|\ \|$ (or

[8] We skip the discussion on the initial condition, for simplicity.

denoted by $\| \ \|_{\boldsymbol{F}}$, if necessary), which is induced by the inner product: $\|f\| := \sqrt{\langle f, f \rangle}$. The following formula always holds, known as Schwarz' inequality:

$$|\langle f, g \rangle| \leq \|f\| \|g\|, \text{ for any } f, g \in \boldsymbol{F}. \tag{25}$$

Now, similar to the case of $\Im$ and $\Im'$, we consider $\boldsymbol{F}$ and its dual space $\boldsymbol{F}'$, which is the totality of continuous linear functionals on $\boldsymbol{F}$. Let T be a linear functional on $\boldsymbol{F}$. This means that T satisfies the condition (20) for $\Im = \boldsymbol{F}$). The continuity of T in (21) is then expressed using the norm:

$$\textit{There exists a constant } C \textit{ such that } |T(f)| \leq C\|f\| \textit{ holds for any } f \in \boldsymbol{F}. \tag{26}$$

Further, if the pre-Hilbert space $\boldsymbol{F}$ is complete, i.e., any Cauchy sequence $(f_n) \subset \boldsymbol{F}$ has its limit in $\boldsymbol{F}$, $\boldsymbol{F}$ is then called a *Hilbert space*.

Hilbert Space Examples:

1. $\boldsymbol{R}^n$. For $\boldsymbol{x} = (x_1, x_2, ..., x_n)^T$ and $\boldsymbol{y} = (y_1, y_2, ..., y_n)^T \in \boldsymbol{R}^n$, we have $\langle \boldsymbol{x}, \boldsymbol{y} \rangle := \sum_{i=1}^{n} x_i y_i$.
2. $l^2 := \{\boldsymbol{c} = (c_j)_{j=1}^{\infty} | c_j \in \boldsymbol{R}, \sum_{j=1}^{\infty} |c_j|^2 < \infty\}$. The inner product is given by: $\langle \boldsymbol{c}, \boldsymbol{d} \rangle := \sum_{i=1}^{\infty} c_i d_i$, where $\boldsymbol{c} = (c_j)_{j=1}^{\infty}$ and $\boldsymbol{d} = (d_j)_{j=1}^{\infty} \in l^2$.
3. L^2 *space*: $L^2(\Omega) = \{f : \Omega \to \boldsymbol{R} \cup \{\pm\infty\} | \int_{\Omega} |f(x)|^2 dx < \infty\}$. For f and $g \in L^2(\Omega)$, we have $\langle f, g \rangle := \int_{\Omega} f(x) g(x) dx$.
4. *Sobolev space*: $W_2^m(\Omega) := \{f \in L^2(\Omega) | D^{\alpha} f \in L^2(\Omega), |\alpha| \leq m\}$, where $D^{\alpha} f$ means a weak derivative. The inner product is given by $\langle f, g \rangle := \sum_{|\alpha| \leq m} \int_{\Omega} D^{\alpha} f(x) D^{\alpha} g(x) dx$.

We will use the following basic theorem in explaining RKHS in the next section:

Theorem 1 (Riesz): Let $\boldsymbol{H}$ be a Hilbert space, and T be a real-valued continuous linear functional on $\boldsymbol{H}$. Then there exists one and only one function $\boldsymbol{t} \in \boldsymbol{H}$ such that

$$T(f) = \langle \boldsymbol{t}, f \rangle \tag{27}$$

for all $f \in \boldsymbol{H}$.

This theorem says that Hilbert space $\boldsymbol{H}$ is isomorphic to its dual space $\boldsymbol{H}'$ as linear topological spaces[9]

[9] The norm $\| \ \|$ in $\boldsymbol{H}'$ is then given by $\|T\| := sup_{\|f\| \leq 1} |T(f)|$, for $T \in \boldsymbol{H}'$.

Fourier Analysis in Hilbert Space We first recall the orthogonal relation between trigonometric functions:

$$\int_{-\pi}^{\pi} \cos nx \cos mx \ dx = \begin{cases} \pi & (m = n) \\ 0 & (m \neq n), \end{cases}$$

$$\int_{-\pi}^{\pi} \sin nx \sin mx \ dx = \begin{cases} \pi & (m = n) \\ 0 & (m \neq n), \end{cases}$$

$$\int_{-\pi}^{\pi} \cos nx \sin mx \ dx = 0.$$

Let f and g be the elements of a Hilbert space $\boldsymbol{H}$. We say that f is orthogonal to g, if $\langle f, g \rangle = 0$.

Consider now the function space $L^2(-\pi, \pi)$, where the inner product is given by: $\langle f, g \rangle := \frac{1}{\pi} \int_{-\pi}^{\pi} f(x)g(x)dx$. The subset $S := \{1, \cos nx, \sin nx \ | n = 1, 2, \cdots\}$ of $L^2(-\pi, \pi)$ is then known as a *complete orthonormal system* of $L^2(-\pi, \pi)$. This means that the system $S \equiv \{\varphi_1, \varphi_2, \cdots\}$ satisfies the following conditions:

$$\langle \varphi_i, \varphi_j \rangle = \delta_{ij} \ \text{ for } i, j = 1, 2, \cdots . \tag{28}$$

$$f = \sum_{j=1}^{\infty} \langle f, \varphi_j \rangle \varphi_j, \ \text{ for any } f \in L^2(-\pi, \pi). \tag{29}$$

The conditions (28) and (29) of course give the definition of complete orthonormal system (CONS) for a general Hilbert space. The coefficients $\langle f, \varphi_j \rangle$ in (29) are then referred to as the Fourier coefficients of f. The right hand of equation (29) is called the Fourier series of f, with regard to the CONS S.

CONS Examples

1. *Legendre polynomials* $\{P_n(x)\}$

$$P_n(x) := \frac{1}{2^n n!} \frac{d^n}{dx^n}(x^2 - 1)^n \ (n = 0, 1, \cdots).$$

Then we have

$$\int_{-1}^{1} P_m(x) P_n(x) dx = \frac{2}{2n + 1} \delta_{mn}.$$

Therefore $\{\sqrt{\frac{2n+1}{2}} P_n(x); n = 0, 1, 2, \cdots\}$ constitutes a CONS of $L^2(-1, 1)$.

2. *Laguerre polynomials* $\{L_n(x)\}$

$$L_n(x) := e^x \frac{d^n}{dx^n}(x^n e^{-x}) \ (n = 0, 1, \cdots).$$

This time we have

$$\int_{0}^{\infty} L_m(x) L_n(x) dx = (n!)^2 \delta_{mn}.$$

Therefore $\{\frac{1}{n!}L_n(x)e^{-x/2}; n = 0, 1, 2, \cdots\}$ constitutes a CONS of $L^2(0, \infty)$.

For our practical purposes, we may assume that Hilbert space always has a complete orthonormal system[10].

9.4 Reproducing Kernel Hilbert Space

A radial basis function (RBF) is closely related to a reproducing kernel Hilbert space (RKHS). Actually an RBF appear in solving a regularization problem on a certain RKHS $\boldsymbol{H}(\Omega)$, where Ω is a domain in $\boldsymbol{R}^n$ or $\boldsymbol{R}^n$ itself. As mentioned in earlier sections, there are several RBFs that are used in scattered data interpolation, such as *Gaussian*, and *thin plate spline*. The choice of an RBF that is most suited in solving an interpolation/regularization problem depends not only on the smoothness of the function to be desired but also on the dimension n. In this section we will briefly sketch this interesting relation among RBF, smoothness of the function, and the space dimension. Now let us start with the definition of RKHS.

9.4.1 Reproducing Kernel

Let E be an abstract set, and $\boldsymbol{H}$ be a Hilbert space consisting of the (real-valued[11]) functions defined on E, with the inner product $\langle , \rangle$.

Definition The function $K : E \times E \to \boldsymbol{R}$ is called a *reproducing kernel* of $\boldsymbol{H}$, if it satisfies the following conditions[12]:

1. For any fixed $y \in E$, $K(x, y)$ belongs to $\boldsymbol{H}$ as a function of x on E.
2. For any $f \in \boldsymbol{H}$, we have $f(y) = \langle f(x), K(x, y)\rangle_x$.

Definition If Hilbert space $\boldsymbol{H}$ has the reproducing kernel, which satisfies the above conditions (1) and (2), then $\boldsymbol{H}$ is referred to as a *reproducing kernel Hilbert space* (RKHS).

The following proposition will be used in characterizing the reproducing kernel in the next section.

Proposition 1. For the reproducing kernel K, we have:

$$K(y, z) = \langle K(x, y), K(x, z)\rangle_x \tag{30}$$

[10] Rigorously, it is a necessary and sufficient condition for Hilbert space H to have a complete orthonormal system that H is *separable* as a metric space. This means that there exists a dense, and countable subset of H. However we don't have to care about it for our practical situations.

[11] Formally, we can treat complex-valued functions, but the "complex" case may be skipped in this course.

[12] In condition (2), the inner product $\langle , \rangle_x$ means that we get the inner product value of the two functions with variable x.

Proof. From condition (1) of the reproducing kernel, we have $K(y,z) \in \boldsymbol{H}$ for a fixed z. Then, by putting $f(y) = K(y,z)$ in condition (2), we have

$$K(y,z) = \langle K(x,z), K(x,y)\rangle_x = \langle K(x,y), K(x,z)\rangle_x.$$

The next theorem is well known as a classical result in the reproducing kernel theory (N. Aronszajn [2], S. Bergman [6]).

Theorem 2: Hilbert space $\boldsymbol{H}(E)$ has a reproducing kernel, if and only if the following condition is satisfied:
For any $y \in E$, there exists a positive constant $C = C_y$, such that

$$|f(y)| \leq C_y \|f\|, \text{ for any } f \in \boldsymbol{H}. \tag{31}$$

Proof. [*only if* part] The assertion follows from Schwarz' inequality (25). [*if* part] For a fixed $y \in E$, let us consider the linear functional $\delta_y : \boldsymbol{H}(E) \to \boldsymbol{R}$, which is defined as $\delta_y(f) := f(y)$, for $f \in \boldsymbol{H}(E)$. According to (26), condition (31) means that δ_y is continuous. The assertion thus follows from Riesz' Theorem (**Theorem 1** with (27)).

Relating to the above theorem, we note that the reproducing kernel K is uniquely determined for an RKHS $\boldsymbol{H}(E)$ (also see Theorem 3 in the next section).

RKHS Examples

1. $\boldsymbol{R}$: Let E be $\{1\}$. Here we identify $\boldsymbol{H}(E)$ with $\boldsymbol{R}$. An arbitrary element of $\boldsymbol{H}(E)$ is a map $f : E \equiv \{1\} \to \boldsymbol{R}$. Specifying $f \in \boldsymbol{H}(E)$ therefore means specifying a real number x with $f(1) = x$. Let the inner product $\langle , \rangle$ for $\boldsymbol{H}(E)$ be the ordinary multiplication in $\boldsymbol{R}$: $\langle x, y\rangle = x \cdot y$. We define $K : E \times E \to \boldsymbol{R}$ as $K(1,1) := 1$. It is then easy to see K satisfies condition (1) in the definition of the reproducing kernel. As for condition (2), we have: $\langle f(1), K(1,1)\rangle = f(1)1 = f(1)$.
2. l^2: Let $\boldsymbol{a} = (a_j)_{j=1}^{\infty} \in l^2$. Then $\boldsymbol{a}$ defines a map $\alpha : \boldsymbol{N} \to \boldsymbol{R}$ with $\alpha(i) := a_i (i \in \boldsymbol{N})$. We thus identify $\boldsymbol{a} \leftrightarrow \alpha$. By setting $E = \boldsymbol{N}$, we have $\boldsymbol{H}(\boldsymbol{N}) \equiv \{\alpha : \boldsymbol{N} \to \boldsymbol{R} | \sum_{i=1}^{\infty} |\alpha(i)|^2 < \infty\} \cong l^2$ with its kernel function K as being $K(i,j) = \delta_{ij}$.
3. Let A be an n-th order, symmetric, and positive semi-definite matrix. Then $A(\boldsymbol{R}^n)$ is RKHS and its reproducing kernel is A (see the discussions in the next section):

$$A(\boldsymbol{R}^n) \equiv \{Ax \in \boldsymbol{R}^n \mid x \in \boldsymbol{R}^n\}.$$

9.5 Fundamental Properties

Proposition 2. Let $K : E \times E \to \boldsymbol{R}$ be the kernel function of RKHS $\boldsymbol{H}(E)$. Then K is a symmetric, positive semi-definite function.

Proof. (30) in Proposition 1 says that K is symmetric, since the inner product itself is symmetric. For $(x_1, x_2, \cdots, x_n)^T \in E^n$ and $(a_1, a_2, \cdots, a_n)^T \in \boldsymbol{R}^n$, we have, using (30),

$$\begin{aligned}\sum_{i,J=1}^{n} a_i a_j K(x_i, x_j) &= \sum_{i,j=1}^{n} a_i a_j \langle K(x, x_i), K(x, x_j)\rangle_x \\ &= \left\langle \sum_{i=1}^{n} a_j K(x, x_i), \sum_{j=1}^{n} a_j K(x, x_j) \right\rangle_x \\ &= \left\| \sum_{k=1}^{n} a_k K(x, x_k) \right\|_x^2 \geq 0.\end{aligned}$$

The following theorem says that RKHS is constructed by spedifying a symmetric, positive semi-definite function. It should also be noted that the proof is constructive so that it might be useful even in our practical situations.

Theorem 3. Suppose that K is a symmetric, positive semi-definite function on $E \times E$. Then there exists a Hilbert space $\boldsymbol{H}$ that has K as its reproducing kernel.

Sketch of the proof. We put $\boldsymbol{F} := \{\sum_{i=1}^{l} \alpha_i K(x, x_i) |\ l \in \boldsymbol{N}, \alpha_i \in \boldsymbol{R}, x_i \in E\}$. By defining addition and multiplication by constant as usual, we can make $\boldsymbol{F}$ a vector space. Also we can introduce the inner product for $f = \sum_{i=1}^{m} \alpha_i K(x, x_i)$ and $g = \sum_{j=1}^{n} \beta_j K(x, x_j) \in \boldsymbol{F}$ as follows: $\langle f, g\rangle := \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_i \beta_j K(x_i, x_j) \in \boldsymbol{R}$. Since K is positive semi-definite, it follows that $\langle f, f\rangle \geq 0$. It is easy to see that F is a pre-Hilbert space. Next we set $\boldsymbol{H}$ as the completion[13] of $\boldsymbol{F}$. Then, with $g(x) \equiv g_y(x) = K(x, y)$, we have $\langle (f(x), K(x, y)\rangle_x = \langle f, h_y\rangle = \sum_{i=1}^{m} \alpha_i K(x_i, y) = \sum_{i-1}^{m} \alpha_i K(y, x_i) = f(y)$, for any $f \in \boldsymbol{F}$. This also holds for any f of $\boldsymbol{H}$, because, for any Cauchy sequence $\{f_m\}$ of $\boldsymbol{F}$, we have

$$\begin{aligned}|f_m(y) - f_n(y)| &\leq |\langle (f_m - f_n)(x), K(x, y)\rangle_x| \\ &\leq \|f_m - f_n\|_x \cdot \|K(x, y)\|_x = \|f_m - f_n\| \cdot K(y, y).\end{aligned}$$

This means that $\{f_m(y)\} \subset \boldsymbol{R}$ converges at any y. $\boldsymbol{H}$ therefore includes $f = \lim_{n\to\infty} f_n$, because f also satisfies condition (2) in the definition of reproducing kernel.

One more remark is about an RKHS that has the complete orthonormal system (CONS). Let us describe the reproducing kernel K with CONS $\equiv \{\varphi_j\}_{j=1}^{\infty}$. From condition (1) of the reproducing kernel, we first get

$$K(x, y) = \sum_{i=1}^{\infty} \alpha_i(y) \varphi_i(x).$$

[13] The completion simply means that all the limits of Cauchy sequences of $\boldsymbol{F}$ are included into its completion. For example, for $\boldsymbol{Q}$, the totality of rational numbers, its completion is $\boldsymbol{R}$, the totality of real numbers.

Then, taking $f(y) = \varphi_i(y)$ in condition (2), we have

$$\varphi_i(y) = \langle \varphi_i, K(\cdot, y) \rangle = \langle \varphi_i, \sum_{k=1}^{\infty} \alpha_k(y) \varphi_k \rangle$$
$$= \sum_{k=1}^{\infty} \alpha_k(y) \langle \varphi_i, \varphi_k \rangle = \sum_{k=1}^{\infty} \alpha_k(y) \delta_{ik} = \alpha_i(y).$$

We therefore have

$$K(x, y) = \sum_{i=1}^{\infty} \varphi_i(x) \varphi_i(y).$$

9.6 RKHS in L^2 space

We briefly describe an infinite-dimensional RKHS in $L^2(E)$. E is assumed to be a domain in $\boldsymbol{R}^n$, and we say L^2 instead of $L^2(E)$ from now on. Supposing that K is a symmetric, positive semi-definite function on $E \times E$, we first define the real-valued function $\kappa(f)$ on $E : \kappa(f)(y) := \langle K(x, y), f(x) \rangle_x$, for any $y \in E$. Let us further suppose that

$$\iint |K(x, x')|^2 dx dx' < \infty. \tag{32}$$

Then κ can be considered as a linear operator: $L^2 \to L^2$, because, using Schwarz' inequality (25), we have

$$\int |\kappa(f)(y)|^2 dy = \int \langle K(x, x'), f(x) \rangle_x^2 dx'$$
$$= \int \left(\int K(x, x') f(x) dx \right)^2 dx'$$
$$\leq \iint |K(x, x')|^2 dx \int |f(y)|^2 dy dx'$$
$$= \iint |K(x, x')|^2 dx dx' \int |f(y)|^2 dy$$
$$= \|f\|_{L^2}^2 \cdot \iint |K(x, x')|^2 dx dx' < \infty.$$

This yields that $\kappa(f) \in L^2$ and that κ is a continuous linear operator, known as *Hilbert-Schmidt integral operator*. According to Mercer's theorem, we then have the eigen decomposition: $K(x, x') = \sum_{\nu \geq 1} \lambda_\nu \phi_\nu(x) \phi_\nu(x')$ where $\nu \in N$, and λ_ν, ϕ_ν are eigen value and eigen functions of κ, respectively. Assumption (32) yields $\sum_{\nu \geq 1} \lambda_\nu^2 < \infty$, so that we have $\lim_{k \to \infty} \lambda_k = 0$. We now assume that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq \cdots > 0$. We then know that $\{\phi_n\}_{n=1}^{\infty}$ is a CONS of L^2, which consequently gives the following result:

Theorem 4. Let $\boldsymbol{H}_\kappa$ be the totality of the functions $f \in L^2$, satisfying $\sum_{k\geq 1} \frac{f_k g_k}{\lambda_k} < \infty$. We then have:

1. $\boldsymbol{H}_\kappa$ is a Hilbert space with the inner product: $\langle f, g\rangle_\kappa = \sum_{k\geq 1} \frac{f_k g_k}{\lambda_k} < \infty$
2. For $g \in \boldsymbol{H}_\kappa$, we have

$$\langle g, \phi_\nu\rangle_\kappa = \frac{\langle g, \phi_\nu\rangle_{L^2}}{\lambda_\nu},\ \langle \phi_\nu, \phi_\nu\rangle_\kappa = \frac{1}{\lambda_\nu}\ (\text{i.e.,}\ \ \|\phi_\nu\|_\kappa = \frac{1}{\sqrt{\lambda_\nu}})\ (\nu = 1, 2, \cdots).$$

3. K is the reproducing kernel of $\boldsymbol{H}_\kappa$: $f(x) = \langle f, K(\cdot, x)\rangle_\kappa$ for any $f \in \boldsymbol{H}_\kappa$.

We skipped the mathematical details and the rigorous proof of the above theorem. Instead, we should keep in mind the relation between $\boldsymbol{H}_\kappa$ and L^2 through the CONS derived from the Hilbert-Schmidt operator κ. We also note that a similar result is obtained in a finite-dimesional case, where K simply means an n-th order symmetric, positive semi-definite matrix and $L^2 \cong \boldsymbol{R}^n$.

9.7 RBF and RKHS

In this section, though a few theorems are stated, we don't give any rigorous explanations and proofs for them. We would just like to sketch the overall flow from the theory to our practice.

9.7.1 Regularization problem in RKHS

For simplicity, let $E = \Omega = \boldsymbol{R}^n$. Suppose that $(\boldsymbol{x}_i, f_i)$ are given as sample points, where $f_i \in \boldsymbol{R}$, $\boldsymbol{x}_i \in \boldsymbol{R}^n (i = 1, 2, \cdots, N)$. Let us consider the following regularization problem: Find a function f defined on $\boldsymbol{R}^n$ such that

$$\min_f \left\{ \sum_{i=1}^{N} (f_i - f(\boldsymbol{x}_i))^2 + \lambda J_m^n(f) \right\}, \tag{33}$$

where

$$J_m^n(f) := \sum_{\alpha_1+\alpha_2+\cdots+\alpha_n=m} \frac{m!}{\alpha_1!\alpha_2!\cdots\alpha_n!} \|D^\alpha f\|_{L^2}^2, \tag{34}$$

$$D^\alpha f := \frac{\partial^m f}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \cdots \partial x_n^{\alpha_n}}.$$

The regularization term $\lambda J_m^n(f)$ in (33) prescribes smoothness of a solution. Now where can we find a solution of this problem (33)? According to the definition of $J_m^n(f)$, we should find a solution to (33) in the following space:

$$\boldsymbol{B}_m^n := \{f : \boldsymbol{R}^n \to R \cup \{\pm\infty\} | D^\alpha f \in L^2(\boldsymbol{R}^n), \text{for any } \alpha(|\alpha| = m)\}. \tag{35}$$

So recall the thin plate spline case. We then start with $\boldsymbol{B}_2^2$ in (15) to minimize the energy functional (16).

In the following we want to solve the regularization problem like (33) in RKHS. The main reason for this is the following nice property of RKHS:

Representer Theorem[14] Let $\boldsymbol{H}$ be an RKHS, with its reproducing kernel K and norm $\| \ \|_{\boldsymbol{H}}$. Consider the regularization problem of the following form: Find $f \in \boldsymbol{H}$ such that

$$\min_{f \in \boldsymbol{H}} \left\{ \sum_{i=1}^{N} (f_i - f(\boldsymbol{x}_i))^2 + \lambda \|f\|_{\boldsymbol{H}}^2 \right\}. \tag{36}$$

The solution f can then be found in the form:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i). \tag{37}$$

It would therefore be nice, if we could have $\boldsymbol{B}_m^n$ as the RKHS in the above theorem. However, J_m^n cannot be the squared norm for $\boldsymbol{B}_m^n$, as described next.

The functional J_m^n has the following properties:

1. $J_m^n(f) = 0 \ \Leftrightarrow \ f \in \mathcal{P}_{m-1}$ (the totality of at most (m-1)-th order polynomials).
2. $J_m^n(f) = (-1)^m \langle f, \Delta^m f \rangle_{L^2}$.

Since the null space of J_m^n is equal to $\mathcal{P}_{m-1}$, we first represent $\boldsymbol{B}_m^n$ as being the direct sum of the two function spaces: $\boldsymbol{B}_m^n = \boldsymbol{H}_m^n \oplus \mathcal{P}_{m-1}$. Then let us solve the regularization problem on $\boldsymbol{H}_m^n$:

Theorem 5 [37]. If $m > \frac{n}{2}$, *then* $\boldsymbol{H}_m^n$ is an RKHS with:

$$\langle f, g \rangle_{\boldsymbol{H}_m^n} := \sum_{\alpha_1 + \alpha_2 + \cdots + \alpha_n = m} \frac{m!}{\alpha_1! \alpha_2! \cdots \alpha_n!} \langle D^\alpha f, D^\alpha g \rangle_{L^2} = \langle (-1)^m \Delta^m f, g \rangle_{L^2}. \tag{38}$$

This also means $\|f\|_{\boldsymbol{H}_m^n}^2 = J_m^n(f)$.

With the above theorem, the regularization problem (33) is restated as:

$$\begin{aligned} &\min_{f \in \boldsymbol{B}_m^n} \left\{ \sum_{i=1}^{N} (f_i - f(\boldsymbol{x}_i))^2 + \lambda J_m^n(f) \right\} \\ \Leftrightarrow \ &\min_{g \in \boldsymbol{H}_m^n, \ p \in \mathcal{P}_{m-1}} \left\{ \sum_{i=1}^{N} \{f_i - (g(\boldsymbol{x}_i) + p(\boldsymbol{x}_i))\}^2 + \lambda \|g\|_{\boldsymbol{H}_m^n}^2 \right\}. \end{aligned} \tag{39}$$

[14] This is one of the variations of the representer theorem. Please refer to [53].

We thus know the solution to (39) is represented in the form of

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i) + p_{m-1}(\boldsymbol{x}), \tag{40}$$

where $p_{m-1}(\boldsymbol{x}) \in \mathcal{P}_{m-1}$.

9.7.2 RBF as Green's function

Considering the inner product of $\boldsymbol{H}_m^n$ (38) in Theorem 5, we assume that $K(\boldsymbol{x}, \boldsymbol{y}) = G(\boldsymbol{x} - \boldsymbol{y})$. Let G then be a Green's function in the sense that

$$\Delta^m G(\boldsymbol{x}) = \delta(\boldsymbol{x}), \tag{41}$$

where δ means a generalized function as Dirac δ. We therefore have the solution f in (40) as

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i G(\boldsymbol{x} - \boldsymbol{x}_i) + p(\boldsymbol{x}), \tag{42}$$

where p is a polynomial $\in \mathcal{P}_{m-1}$.

This brings us to our familiar class of radial basis functions:

$$G(\boldsymbol{x}) = \begin{cases} \beta_{mn} |\boldsymbol{x}|^{2m-n} \log |\boldsymbol{x}| & \text{if } 2m-n \text{ is an even integer,} \\ \gamma_{mn} |\boldsymbol{x}|^{2m-n} & \text{otherwise,} \end{cases} \tag{43}$$

where β_{mn} and γ_{mn} are constants.

For example, for the thin plate spline, we have $m = n = 2$ so that $G(\boldsymbol{x}) = |\boldsymbol{x}|^2 \log |\boldsymbol{x}|$ and the polynomial p in (42) is of the first order (linear). Another familiar case is where $m = 2$ and $n = 3$. The we have $G(\boldsymbol{x}) = |\boldsymbol{x}|$ and a linear polynomial for (42).

Regularization with RKHS norm Let us consider another regularization problem, where, instead of $J_m^n(f)$ in (33) and (34), we take $\sum_{m \geq 0} a_m J_m^n(f)$ with a_m being constants and $a_0 \neq 0$. According to the scenario of establishing the above theorems, we have the following result in which the regularization problem is directly solved in an RKHS.

1. We can find a Green's function for the operator $\sum_{m \geq 0} (-1)^m \Delta^m$. The solution is then given by $f(\boldsymbol{x}) = \sum_{k=1}^{N} c_k G(\boldsymbol{x} - \boldsymbol{x}_k)$. Note that this time we don't need a polynomial term like (42).
2. *Gaussian RBF.* In a particular case, where $a_m = \frac{\sigma^{2m}}{m! 2^m} (\sigma > 0)$, we have the Green's function as $G(\boldsymbol{x}) = c\ exp(-\frac{\|\boldsymbol{x}\|^2}{2\sigma^2})$.

10 Open issues

In this section, we look into some more recent research related to scattered data interpolation. Not surprisingly, a lot of this research focusses on radial basis functions and their variants.

10.1 Optimal sampling

So far we have assumed that the input samples are given to us and that for the sampling process was completely our of our control. For some applications, this is not the case. For instance, maybe we are using building an RBF reconstruction in order to approximate an expensive rendering algorithm. In this case we are using RBFs as a procedural approximation of an algorithm. Here we control sampling and can decide how many samples we want to use and what they are (by changing the orientation of the incoming and outgoing rays and also perhaps the wavelength). This problem is also important in machine learning for instance for training neural networks.

There has been some investigation conducted in this area. For instance De Marchi *et al.* [13] study optimal sampling with respect to stability and conclude that good interpolation points are always uniformly distributed according to some criteria (e.g. asymptotically). Rendall and Allen [49] study this problem in the context of volume mesh deformation driven by surface motion. They develop a technique for selecting a subset of the surface points.

10.2 Non-data centered RBFs

In these notes we have considered that the RBF kernels are centered at the sample sites. This might not be optimal nor desirable for some application. For instance, if my data set is very dense, using a kernel per sample might lead to unacceptable performance. Instead we could use much fewer kernel in the reconstruction than there are samples. One way of doing this is to use a so-called *center selection* algorithm to pick as many sampling sites as needed to center the kernels. A different approach is to have "free floating" kernels where the center of the kernels are also parameters to be determined. This could be solved as an optimization problem by choosing the center locations that minimizes the reconstruction error at the sample points.

The problem of center selection is a component of a wider problem called *structure selection* whose goal is to find an optimal number of radial basis functions and their centers. This has been a subject of investigation in the neural network community. For instance, Hatanaka *et al.* [28] uses a genetic algorithm to solve this problem.

11 Further Readings

There are several recent books [10, 60, 18] covering scattered data interpolation topics though unfortunately all of them require a level of mathematics well beyond that required for this course. [18] is perhaps the most accessible. Radial basis and other interpolation methods are also covered in machine learning texts [8] since they can be used for regression and classification.

Acknowledgments

We thank Geoffrey Irving and Armin Iske for discussions of several topics.

12 Appendix

A Python program for Laplace interpolation

```
def solve(d,constrained):

    n = d.size
    nc = constrained.size
    A00 = sparse.lil_matrix((n,n))
    A = sparse.lil_matrix((n+nc,n+nc))

    A00.setdiag(-2.*n_.ones(n))
    koff = 1
    A00.setdiag(n_.ones(n),koff)
    A00.setdiag(n_.ones(n),-koff)
    A00[0,0] = -1.
    A00[n-1,n-1] = -1.

    A[0:n,0:n] = A00

    S = sparse.lil_matrix((nc,n))
    for ir in range(nc):
        S[ir,constrained[ir]] = 1.
    St = S.T

    A[0:n,n:(n+nc)] = St
    A[n:(n+nc),0:n] = S

    A = A.tocsr()
```

```
    b = n_.zeros((n+nc,1))
    for i in range(nc):
        b[n+i] = d[constrained[i]]

    f = linsolve.spsolve(A,b)
    f = f[0:n]

    return f


def test():
    x = n_.array([2.,4., 6.,  8.,  13., 14.])
    y = n_.array([2.,7., 8.,  9.,  12., 12.5])
    n = x.size
    LEN = 200
    x = x * LEN / 20.

    X = n_.arange(float(LEN))
    Y = n_.zeros((LEN,))
    for i in range(x.size):
        ii = int(x[i])
        Y[ii] = y[i]

    f1 = solve(Y,x)
```

References

[1] Anders Adamson and Marc Alexa. Anisotropic point set surfaces. In *Afrigraph 2006: Proceedings of the 4th International Conference on Computer Graphics, Virtual Reality, Visualization and Interaction in Africa*, pages 7–13. ACM Press, 2006.

[2] N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68:337–404, 1950.

[3] W. Baxter and K. Anjyo. Latent doodle space. *Computer Graphics Forum*, 25:477–485, 2006.

[4] R. K. Beatson, W. A. Light, and S. Billings. Fast solution of the radial basis function interpolation equations: Domain decomposition methods. *SIAM J. Sci. Comput.*, 22(5):1717–1740, 2000.

[5] R.K. Beatson, J.B. Cherrie, and C.T. Mouat. Fast fitting of radial basis functions: Methods based on preconditioned gmres iteration. *Advances in Computational Mathematics*, 11:253– 270, 1999.

[6] S. Bergman. The kernel function and the conformal mapping. *Mathematical Surveys*, 5, 1950.

[7] Bernd Bickel, Mario Botsch, Roland Angst, Wojciech Matusik, Miguel Otaduy, Hanspeter Pfister, and Markus Gross. Multi-scale capture of facial geometry and motion. *ACM Trans. Graph.*, 26(3):33, 2007.

[8] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[9] Jean-Daniel Boissonnat and Frédéric Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. In *SCG '00: Proceedings of the sixteenth annual symposium on Computational geometry*, pages 223–232, New York, NY, USA, 2000. ACM.

[10] Martin D. Buhmann. *Radial Basis Functions : Theory and Implementations*. Cambridge University Press, 2003.

[11] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76, New York, NY, USA, 2001. ACM.

[12] Z.-Q. Cheng, Y.-Z. Wang, K. Xu B. Li, G. Dang, and S.Y.-Jin. A survey of methods for moving least squares surfaces. In *IEEE/EG Symposium on Volume and Point-Based Graphics*, 2008.

[13] Stefano De Marchi, Robert Schaback, and Holger Wendland. Near-optimal data-independent point locations for radial basis function interpolation. *Advances in Computational Mathematics*, 23:317–330, 2005. 10.1007/s10444-004-1829-1.

[14] J. Duchon. Math. comp. *Interpolation des Fonctions de Deux Variables Suivant le Principe de la Flexion des Plaques Minces*, 10:5–12, 1976.

[15] Gregory Fasshauer. Solving differential equations with radial basis functions: multilevel methods and smoothing. *Advances in Computational Mathematics*, 11:139–159, 1999. 10.1023/A:1018919824891.

[16] Gregory Fasshauer and Jack Zhang. On choosing "optimal" shape parameters for rbf approximation. *Numerical Algorithms*, 45:345–368, 2007. 10.1007/s11075-007-9072-8.

[17] Gregory E. Fasshauer and Larry L. Schumaker. Scattered data fitting on the sphere. In *Proceedings of the international conference on Mathematical methods for curves and surfaces II Lillehammer, 1997*, pages 117–166, Nashville, TN, USA, 1998. Vanderbilt University.

[18] Gregory F. Fasshauer. *Meshfree Approximation Methods with MATLAB*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2007.

[19] Frank Filbir and Dominik Schmid. Stability results for approximation by positive definite functions on so (3). *J. Approx. Theory*, 153(2):170–183, 2008.

[20] Shachar Fleishman, Daniel Cohen-Or, and Cláudio T. Silva. Robust moving least-squares fitting with sharp features. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 544–552, New York, NY, USA, 2005. ACM.

[21] C. Franke and R. Schaback. Solving partial differential equations by collocation using radial basis functions. *Applied Mathematics and Computation*, 93(1):73 – 82, 1998.

[22] R. Franke and G. Nielson. Smooth interpolation of large sets of scattered data. *International Journal of Numerical Methods in Engineering*, 15:1691–1704, 1980.

[23] Richard Franke. Math. comp. *Scattered data interpolation: tests of some methods*, 38(157):181–200, 1982.

[24] Edward J. Fuselier. *Refined error estimates for matrix-valued radial basis functions*. PhD thesis, Texas A & M University, College Station, Texas, 2006.

[25] Edward J. Fuselier and Grady B. Wright. Stability and error estimates for vector field interpolation and decomposition on the sphere with rbfs. *SIAM J. Numer. Anal.*, 47(5):3213–3239, 2009.

[26] Eitan Grinspun, Mathieu Desbrun, and et al. Discrete differential geometry: An applied introduction. SIGGRAPH Course, 2006.

[27] R. L. Hardy. Journal of geophysical research. *Multiquadric equations of topography and other irregular surfaces*, 76(8):19051915, 1971.

[28] T. Hatanaka, N. Kondo, and K. Uosaki. Multi-objective structure selection for radial basis function networks based on genetic algorithm.

[29] Y. C. Hon, R. Schaback, and X. Zhou. Adaptive greedy algorithm for solving large rbf collocation problems. *Numer. Algorithms*, 32:13–25, 2003.

[30] Hsin-Yun Hu, Zi-Cai Li, and Alexander H. D. Cheng. Radial basis collocation methods for elliptic boundary value problems. *Comput. Math. Appl.*, 50(1-2):289–320, 2005.

[31] S. Jakobsson and O. Amoignon. Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization. *Computers & Fluids*, 36(6):1119 – 1136, 2007.

[32] Pushkar Joshi, Wen C. Tien, Mathieu Desbrun, and Frédéric Pighin. Learning controls for blend shape based realistic facial animation. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 187–192, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[33] Tsuneya Kurihara and Natsuki Miyata. Modeling deformable human hands from medical images. In *Proceedings of the 2004 ACM SIGGRAPH Symposium on Computer Animation (SCA-04)*, pages 357–366, 2004.

[34] J. P. Lewis. Generalized stochastic subdivision. *ACM Transactions on Graphics*, 6(3):167–190, July 1987.

[35] J.P. Lewis. Lifting detail from darkness. In *SIGGRAPH '01: Proceedings of the SIGGRAPH 2001 conference Sketches & applications*. ACM Press, 2001.

[36] A. Lorusso, D. W. Eggert, and R. B. Fisher. A comparison of four algorithms for estimating 3-d rigid transformations. In *BMVC '95: Proceedings of the 1995 British conference on Machine vision (Vol. 1)*, pages 237–246, Surrey, UK, UK, 1995. BMVA Press.

[37] J. Meinguet. Multivariate interpolation at arbitrary points made simple. *Z. Angew. Math.*, 30:292–304, 1979.

[38] Tomohiko Mukai and Shigeru Kuriyama. Geostatistical motion interpolation. *ACM Trans. Graph.*, 24(3):1062–1070, 2005.

[39] Francis J. Narcowich and Joseph D. Ward. Generalized hermite interpolation via matrix-valued conditionally positive definite functions. *Math. Comput.*, 63(208):661–687, 1994.

[40] Jun-yong Noh and Ulrich Neumann. Expression cloning. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 277–288, New York, NY, USA, 2001. ACM.

[41] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. *ACM Trans. Graph.*, 22(3):463–470, 2003.

[42] Sang Il Park, Hyun Joon Shin, and Sung Yong Shin. On-line locomotion generation based on motion blending. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 105–111, New York, NY, USA, 2002. ACM.

[43] Sung W. Park, Lars Linsen, Oliver Kreylos, John D. Owens, and Bernd Hamann. Discrete sibson interpolation. *IEEE Transactions on Visualization and Computer Graphics*, 12(2):243–253, 2006.

[44] Mark Pauly, Markus Gross, and Leif P. Kobbelt. Efficient simplification of point-sampled surfaces. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 163–170, Washington, DC, USA, 2002. IEEE Computer Society.

[45] Mark Pauly, Richard Keiser, Leif P. Kobbelt, and Markus Gross. Shape modeling with point-sampled geometry. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 641–650, New York, NY, USA, 2003. ACM.

[46] Xavier Pennec. Computing the mean of geometric features - application to the mean rotation. Technical Report Tech Report 3371, Institut National de Recherche en Informatique et en Automatique, March 1998.

[47] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Trans. Graph.*, 22(3):313–318, 2003.

[48] Frédéric Pighin, Jamie Hecker, Dani Lischinski, Richard Szeliski, and David H. Salesin. Synthesizing realistic facial expressions from photographs. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 75–84, New York, NY, USA, 1998. ACM.

[49] T. C. S. Rendall and C. B. Allen. Reduced surface point selection options for efficient mesh deformation using radial basis functions. *J. Comput. Phys.*, 229(8):2810–2820, 2010.

[50] C.F. Rose, I. Peter-pike, J. Sloan, and M.F. Cohen. Artist-directed inverse-kinematics using radial basis function interpolation. In *EUROGRAPHICS*, 2001.

[51] M. Sato. Theory of hyperfunctions. I, *Fac. Sci. Univ. Tokyo* Sect. 1, 8, 139-193, 1959; II, ibid, 8, 387-437, 1960.

[52] Scott Schaefer, Travis McPhail, and Joe Warren. Image deformation using moving least squares. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 533–540, New York, NY, USA, 2006. ACM.

[53] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

[54] L. Schwartz. *Théory des Distributions, vol.I et II*. Hermann, 1950, 1951.

[55] Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *ACM '68: Proceedings of the 1968 23rd ACM national conference*, pages 517–524, New York, NY, USA, 1968. ACM.

[56] Robert Sibson. A brief description of natural neighbor interpolation. In V. Barnett, editor, *Interpreting Multivariate Data*, chapter 2, pages 21–36. John Wiley, Chichester, 1981.

[57] H. Todo, K. Anjyo, W. Baxter, and T. Igarashi. Locally controllable stylized shading. *ACM Trans. Graph.*, 26(3):17, 2007.

[58] Karel Uhlir and Vaclav Skala. Radial basis function use for the restoration of damaged images. In Max Viergever, K. Wojciechowski, B. Smolka, H. Palus, R.S. Kozera, W. Skarbek, and L. Noakes, editors, *Computer Vision and Graphics*, volume 32 of *Computational Imaging and Vision*, pages 839–844. Springer Netherlands, 2006.

[59] R. Peter Weistroffer, Kristen R. Walcott, Greg Humphreys, and Jason Lawrence. Efficient basis decomposition for scattered reflectance data.

In *EGSR07: Proceedings of the Eurographics Symposium on Rendering*, Grenoble, France, June 2007.

[60] Holger Wendland. *Scattered Data Approximation*. Cambridge, 2005.

[61] Holger Wendland. Divergence-free kernel methods for approximating the stokes problem. *SIAM J. Numer. Anal.*, 47(4):3158–3179, 2009.

[62] Lexing Ying. Short note: A kernel independent fast multipole algorithm for radial basis functions. *J. Comput. Phys.*, 213(2):451–457, 2006.

[63] Yuanchen Zhu and Steven J. Gortler. 3d deformation using moving least squares. harvard computer science technical report: Tr-10-07. Technical report, Harvard University, Cambridge, MA, 2007.

[64] Todd Zickler, Ravi Ramamoorthi, Sebastian Enrique, and Peter N. Belhumeur. Reflectance sharing: Predicting appearance from a sparse set of images of a known shape. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(8):1287–1302, 2006.