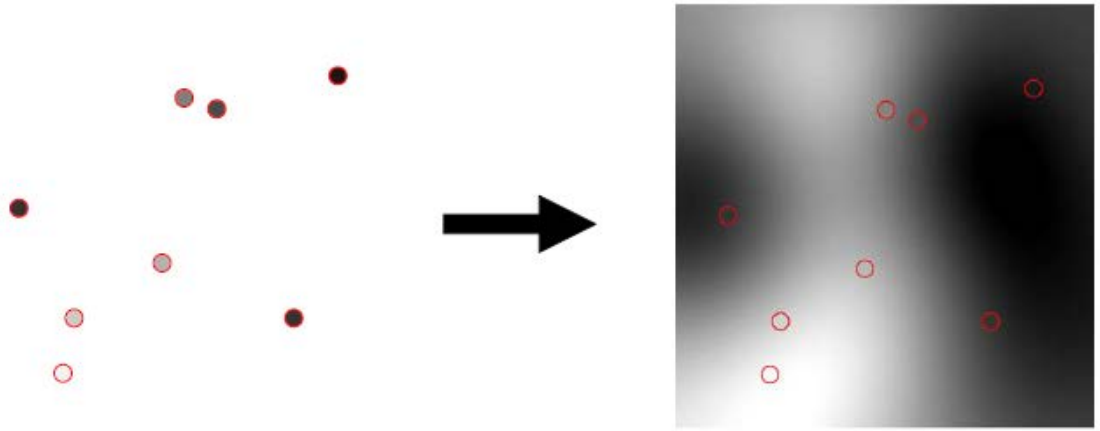
Scattered-Data Interpolation

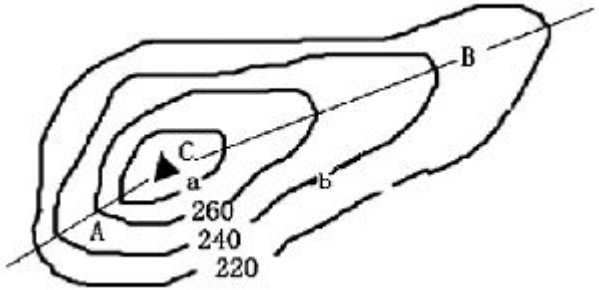
胡新宇 叶振涛 史百丰
李世成 张文涵 唐正纲

Problem

How to compute a complete image from several gray dots?



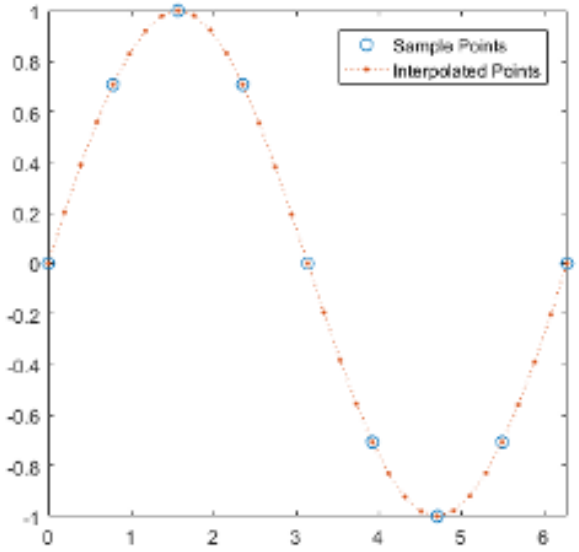
Similar and more intuitively
How to draw the contour for map?

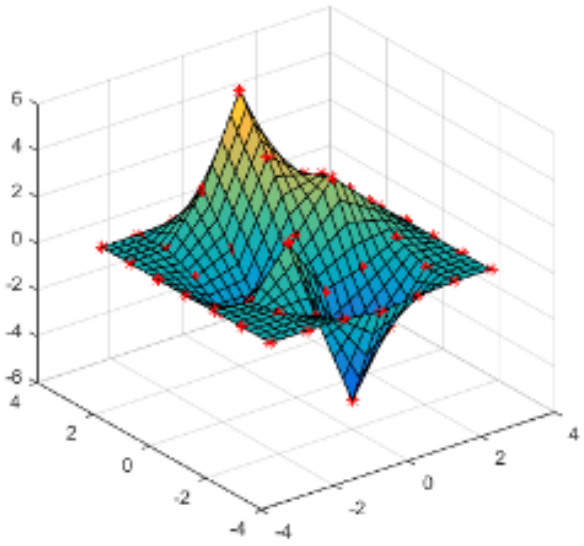


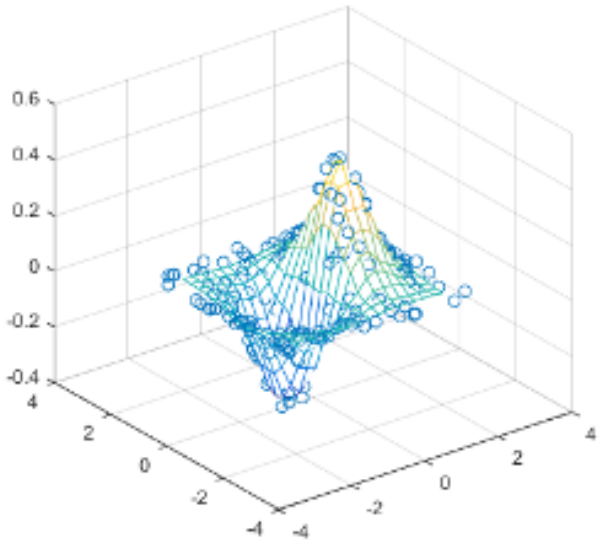
Interpolation

插值是在一组已知数据点的范围内添加新数据点的方法，被广泛用于数据处理

可以使用插值来填充缺失的数据、对现有数据进行平滑处理以及进行预测等





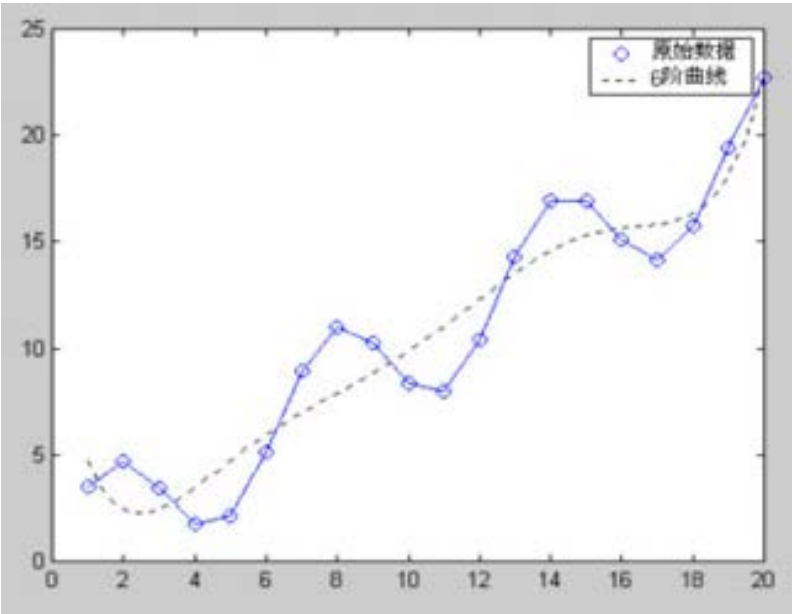


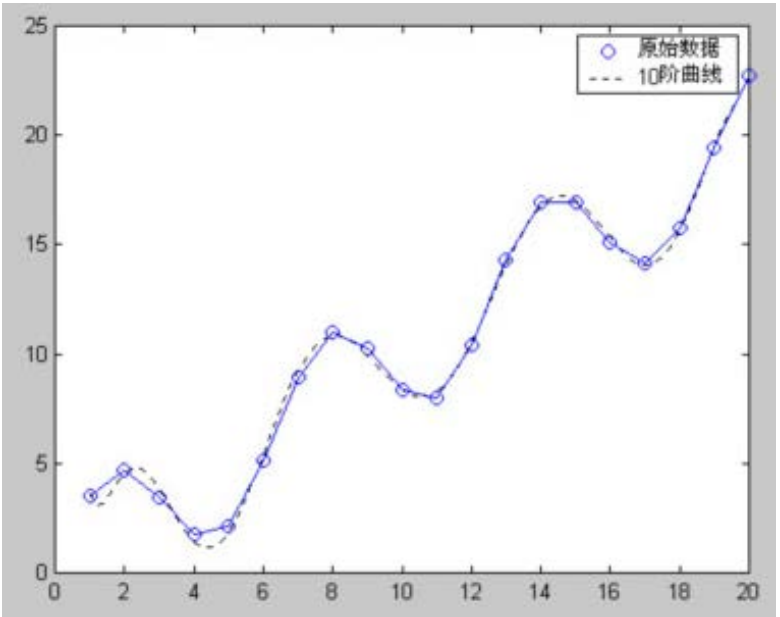
Interpolation

插值是在一组已知数据点的范围内添加新数据点的方法，被广泛用于数据处理

可以使用插值来填充缺失的数据、对现有数据进行平滑处理以及进行预测等

相比于拟合更加精确，要求已知的数据点准确度较高

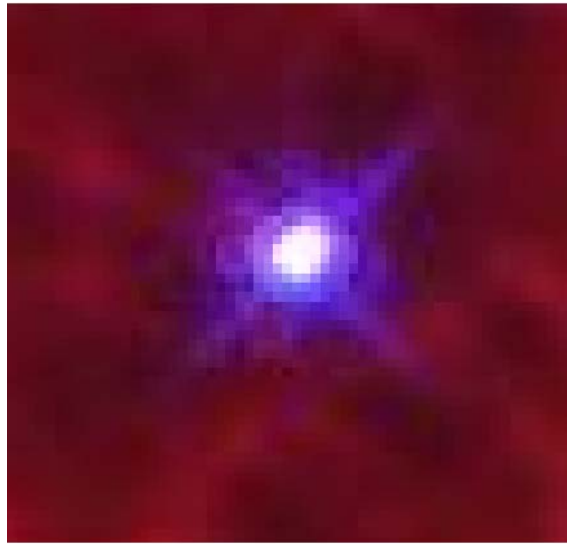




Interpolation

修补图像放大后产生的伪影

Origin



Interpolation

Formalization

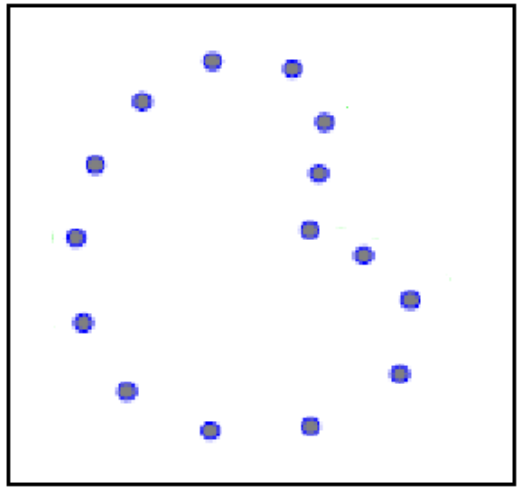
More generally, the interpolation problem can be stated as follows: given a set of N-dimensional data points $\{P_1, P_2, \cdots, P_m\}$, associated with scalar data values $\{v_1, v_2, \cdots, v_m\}$, compute a function $f(P)$: $R^N \rightarrow R$ for any point P in the N-dimensional space such that $f(P_i) = v_i$ for $i = 1,2, \cdots, m$.

In the examples at first, the data points reside on a 2D plane (N = 2), and the data values are intensities/altitudes at each point.

Example (without methods)

Consider a application of surface reconstruction. Mostly, we capture a real-world 3D object as a cloud of data points on the object's surface, and then use algorithm to reconstruct it.

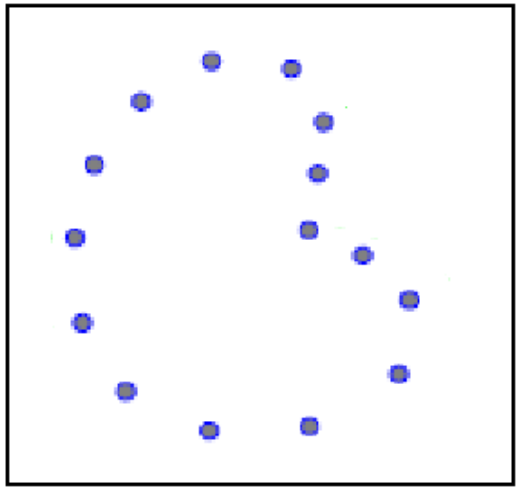
To ease the discussion, we consider a 2D case where the point cloud is highlighted by blue circles and needs to be connected to form a closed curve.



Example (without methods)

Consider a application of surface reconstruction. Mostly, we capture a real-world 3D object as a cloud of data points on the object's surface, and then use algorithm to reconstruct it.

To ease the discussion, we consider a 2D case where the point cloud is highlighted by blue circles and needs to be connected to form a closed curve.

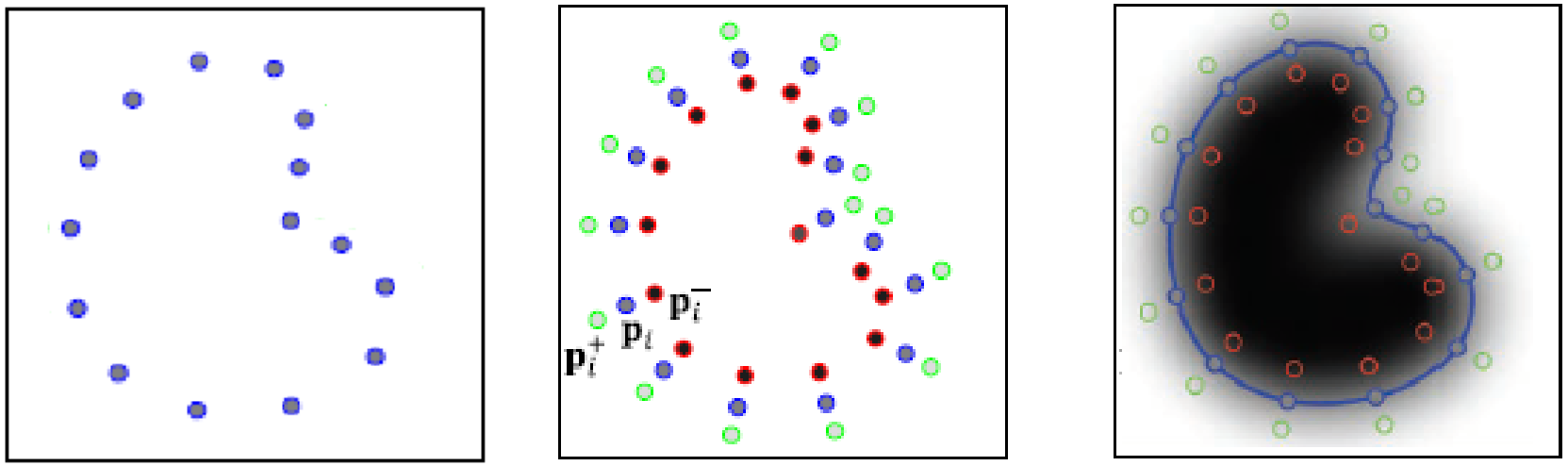


An immediate idea: consider each point in the cloud as a data point associated with the intensity 0. We only need to find an interpolation function $f(P)$: $R^2 \rightarrow R$ with $f(P_i) = 0$ for all P_i.

But some other things need to do.

Example (without methods)

A trivial and invalid solution $f(R^2) = 0$, whatever methods used to compute the interpolation function .
Instead, we add some artificial data points with positive or negative intensities at locations outside or inside the curve to be reconstructed.
For all P_i, an outward unit normal vector $\overrightarrow{n_i}$ can be computed according to the original cloud and place new points $P_i + d\overrightarrow{n_i}$ outside the curve with green color and intensity d, and $P_i - d\overrightarrow{n_i}$ inside the curve with red color and intensity –d.



Example (without methods)

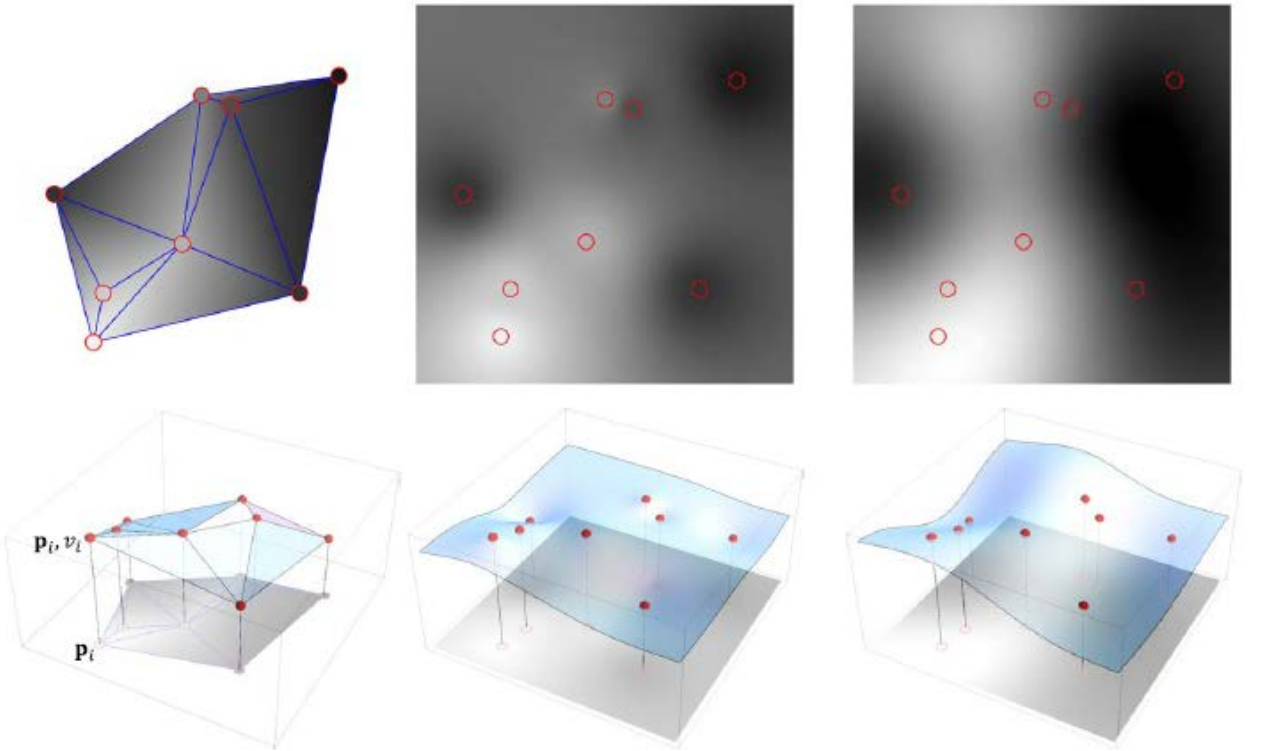
The idea can easily extended to 3D, and we can use some software to reconstruct the surface after obtaining the interpolating function.

Interpolating function

Reconstructed surface

Interpolation Function

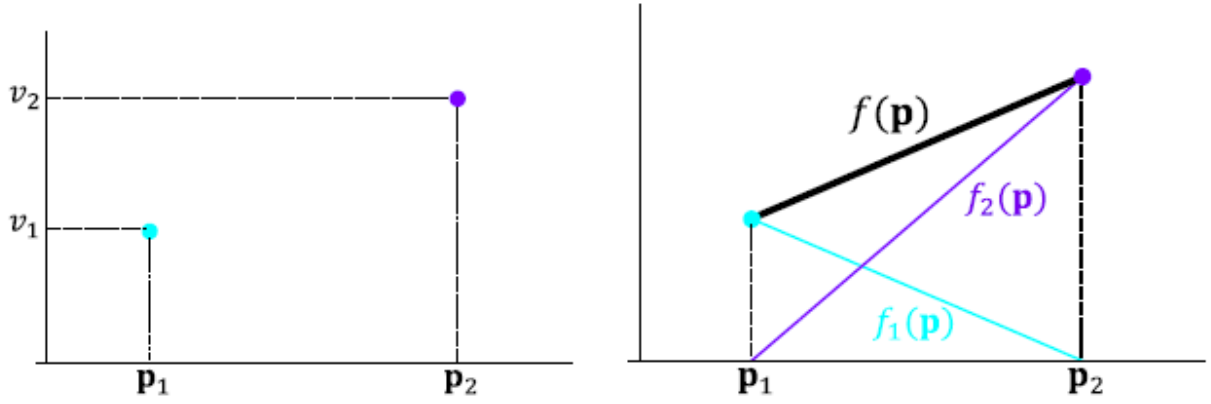
To estimate an interpolation function, we most pay attention to smoothness and low oscillation which are crucial to both intuition and application value.



Piecewise Linear Shepherd's Radial Basis Functions

Piecewise linear interpolation

In the east case of two 1D points, the function is obvious:

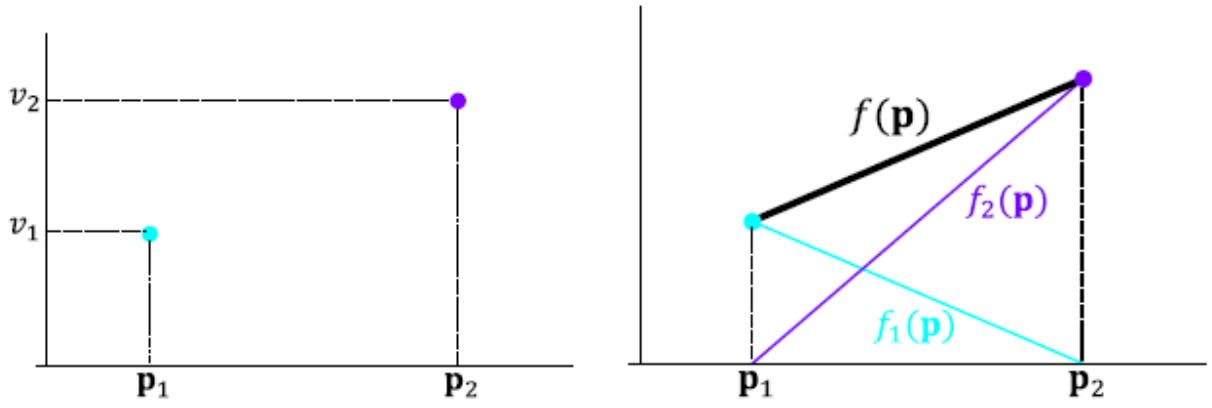


Two points: $(P_1, v_1), (P_2, v_2)$ the interpolation function is a line segment connecting two points, such as the black line.

$$f(\mathbf{p}) = \frac{\mathbf{p}_2 - \mathbf{p}}{\mathbf{p}_2 - \mathbf{p}_1} v_1 + \frac{\mathbf{p} - \mathbf{p}_1}{\mathbf{p}_2 - \mathbf{p}_1} v_2$$

Piecewise linear interpolation

In the east case of 2 1D points, the function is obvious:



We provide another interpretation of $f(P)$ here, which may has little power now but be quite useful in the methods introduced later. (bump function)

$$f(\mathbf{p}) = \frac{\mathbf{p}_2 - \mathbf{p}}{\mathbf{p}_2 - \mathbf{p}_1} v_1 + \frac{\mathbf{p} - \mathbf{p}_1}{\mathbf{p}_2 - \mathbf{p}_1} v_2 = f_1(\mathbf{p}) + f_2(\mathbf{p})$$

Piecewise linear interpolation

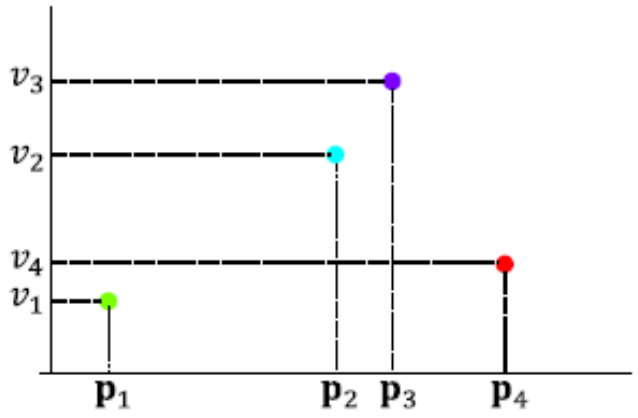
When points increase, we can linearly interpolate between every two successive points, and connect the resulting line segments into a polyline.

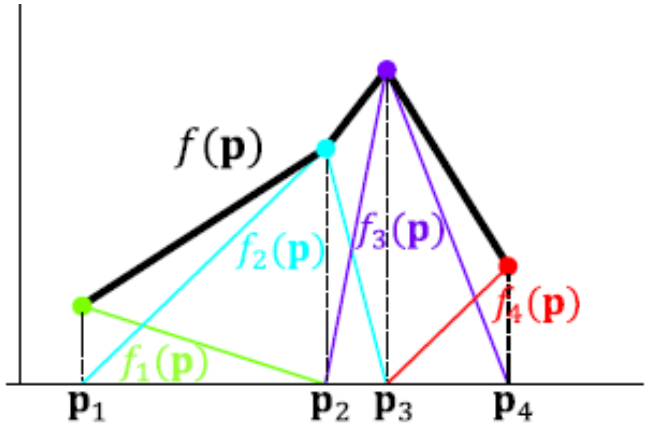
Following the alternative interpretation of linear interpolation, the polyline $f(P)$ can be constructed as the sum of a set of polylines $f_i(P)$:

$$f_i(\mathbf{p}) = \begin{cases} \frac{\mathbf{p}-\mathbf{p}_{i-1}}{\mathbf{p}_i-\mathbf{p}_{i-1}} v_i \,, & \text{if } i > 1 \text{ and } \mathbf{p} \in [\mathbf{p}_{i-1}, \mathbf{p}_i) \,, \\ \frac{\mathbf{p}_{i+1}-\mathbf{p}}{\mathbf{p}_{i+1}-\mathbf{p}_i} v_i \,, & \text{if } i < n \text{ and } \mathbf{p} \in [\mathbf{p}_i, \mathbf{p}_{i+1}] \,, \\ 0 \,, & \text{otherwise} \,, \end{cases}$$

for $\mathbf{p} \in [\mathbf{p}_1, \mathbf{p}_n]$

Piecewise linear interpolation





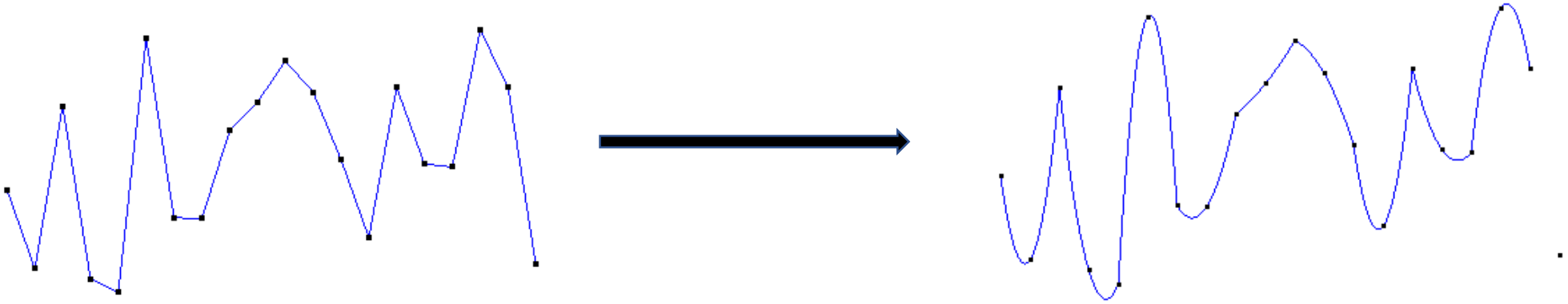
$$f_i(\mathbf{p}) = \begin{cases} \frac{\mathbf{p}-\mathbf{p}_{i-1}}{\mathbf{p}_i-\mathbf{p}_{i-1}} v_i\,, & \text{if } i > 1 \text{ and } \mathbf{p} \in [\mathbf{p}_{i-1}, \mathbf{p}_i)\,, \\ \frac{\mathbf{p}_{i+1}-\mathbf{p}}{\mathbf{p}_{i+1}-\mathbf{p}_i} v_i\,, & \text{if } i < n \text{ and } \mathbf{p} \in [\mathbf{p}_i, \mathbf{p}_{i+1}]\,, \\ 0\,, & \text{otherwise}\,, \end{cases}$$

for $\mathbf{p} \in [\mathbf{p}_1, \mathbf{p}_n]$ $\qquad f(\mathbf{p}) = \sum_{i=1}^{n} f_i(\mathbf{p})$

Quadratic interpolation

Similar to the Piecewise linear interpolation, we can generalize the form of $f(p)$, using three consecutive points to construct the function:

$$f(p) = \frac{(p-p_2)(p-p_3)}{(p_1-p_2)(p_1-p_3)}v_1 + \frac{(p-p_1)(p-p_3)}{(p_2-p_1)(p_2-p_3)}v_2 + \frac{(p-p_1)(p-p_2)}{(p_3-p_1)(p_3-p_2)}v_3$$



Not smooth at breaking points.

Cubic interpolation

To construct the smooth connect, we introduce a certain third degree polynomial

$$f(x) = ax^3 + bx^2 + cx + d \qquad f'(x) = 3ax^2 + 2bx + c$$

Four consecutive points p_0, p_1, p_2, p_3, we try to have $f(0) = p_1, f(1) = p_2$, $x \in [0,1]$ and $f'(x)$ is continuous with $x \in [0,1]$.

$$f(0) = d \qquad a = 2f(0) - 2f(1) + f'(0) + f'(1)$$

$$f(1) = a + b + c + d \qquad b = -3f(0) + 3f(1) - 2f'(0) - f'(1)$$

$$f'(0) = c \qquad c = f'(0)$$

$$f'(1) = 3a + 2b + c \qquad d = f(0)$$

Cubic interpolation

$$f(0) = p_1 \qquad a = -\tfrac{1}{2}p_0 + \tfrac{3}{2}p_1 - \tfrac{3}{2}p_2 + \tfrac{1}{2}p_3$$

$$f(1) = p_2 \qquad b = p_0 - \tfrac{5}{2}p_1 + 2p_2 - \tfrac{1}{2}p_3$$

$$f'(0) = \frac{p_2 - p_0}{2} \qquad c = -\tfrac{1}{2}p_0 + \tfrac{1}{2}p_2$$

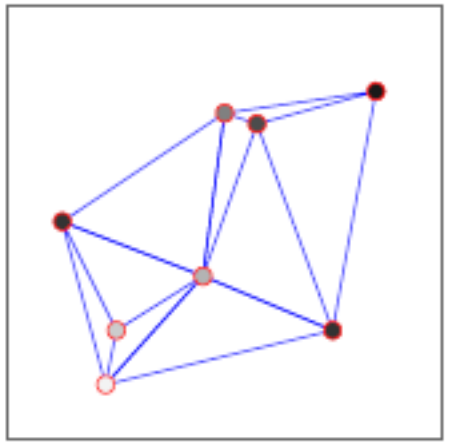
$$f'(1) = \frac{p_3 - p_1}{2} \qquad d = p_1$$

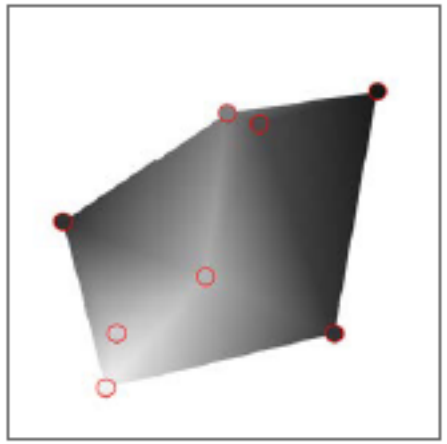
So our cubic interpolation formula becomes:

$$f(p_0, p_1, p_2, p_3, x) = (-\tfrac{1}{2}p_0 + \tfrac{3}{2}p_1 - \tfrac{3}{2}p_2 + \tfrac{1}{2}p_3)x^3 + (p_0 - \tfrac{5}{2}p_1 + 2p_2 - \tfrac{1}{2}p_3)x^2 + (-\tfrac{1}{2}p_0 + \tfrac{1}{2}p_2)x + p_1$$

Tessellation

To apply piecewise linear interpolation to higher dimensions, we will need to tessellate the data points into elementary domains, within which linear interpolation is performed.
When come to the case of 2D , we first use tesellation to construct triangles with the data points and form a convex hull.
For all triangles, we can perform the method of 1D to compute the values of edges, and then compute the inner points by the values of vertexes and edges.

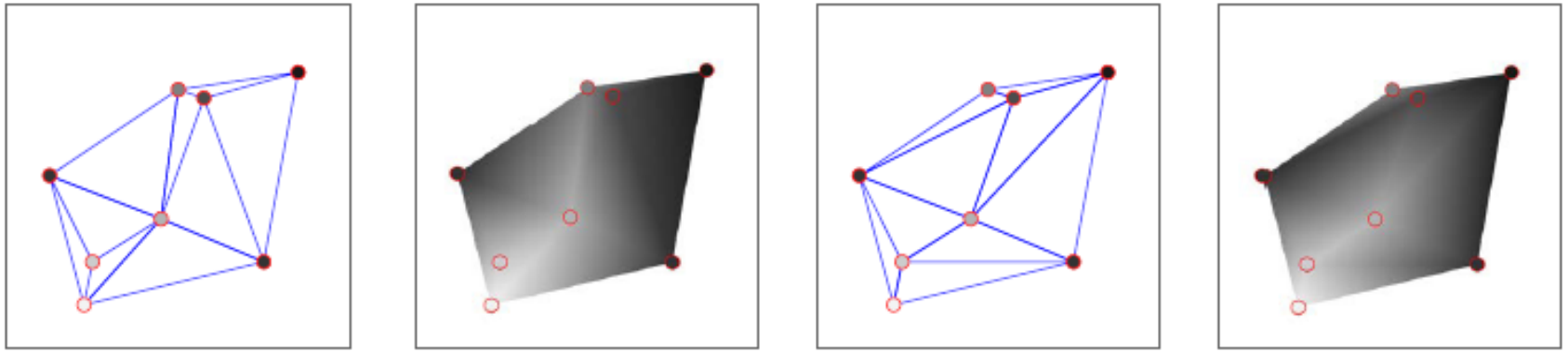




Tessellation

As we can see , the piecewise linear interpolation is simple and direct but not smooth at derivatives of functions e.g. at the data points.

In addition, in two dimensions and beyond, the interpolation also depends on how the data points are tessellated.



Scattered-Data Interpolation - Part 2

Baifeng Shi Shicheng Li

Peking University

{*1700012701, 1700012933*}*@pku.edu.cn*

May 7, 2019

Overview

1. Shepard's Interpolation
2. Radial Basis Function

Previously on linear interpolation

- We have n points $(x_i)_{i=1}^n \in \mathbb{R}$.
- We can define n 'bump functions' f_i for each x_i.
- For linear interpolation, $f_i^-(x) = \frac{x-x_i}{x_i-x_{i-1}} v_i$, $f_i^+(x) = \frac{x_{i+1}-x}{x_{i+1}-x_i} v_i$.
- But the problem is, it's too 'spiky'.

Previously on linear interpolation

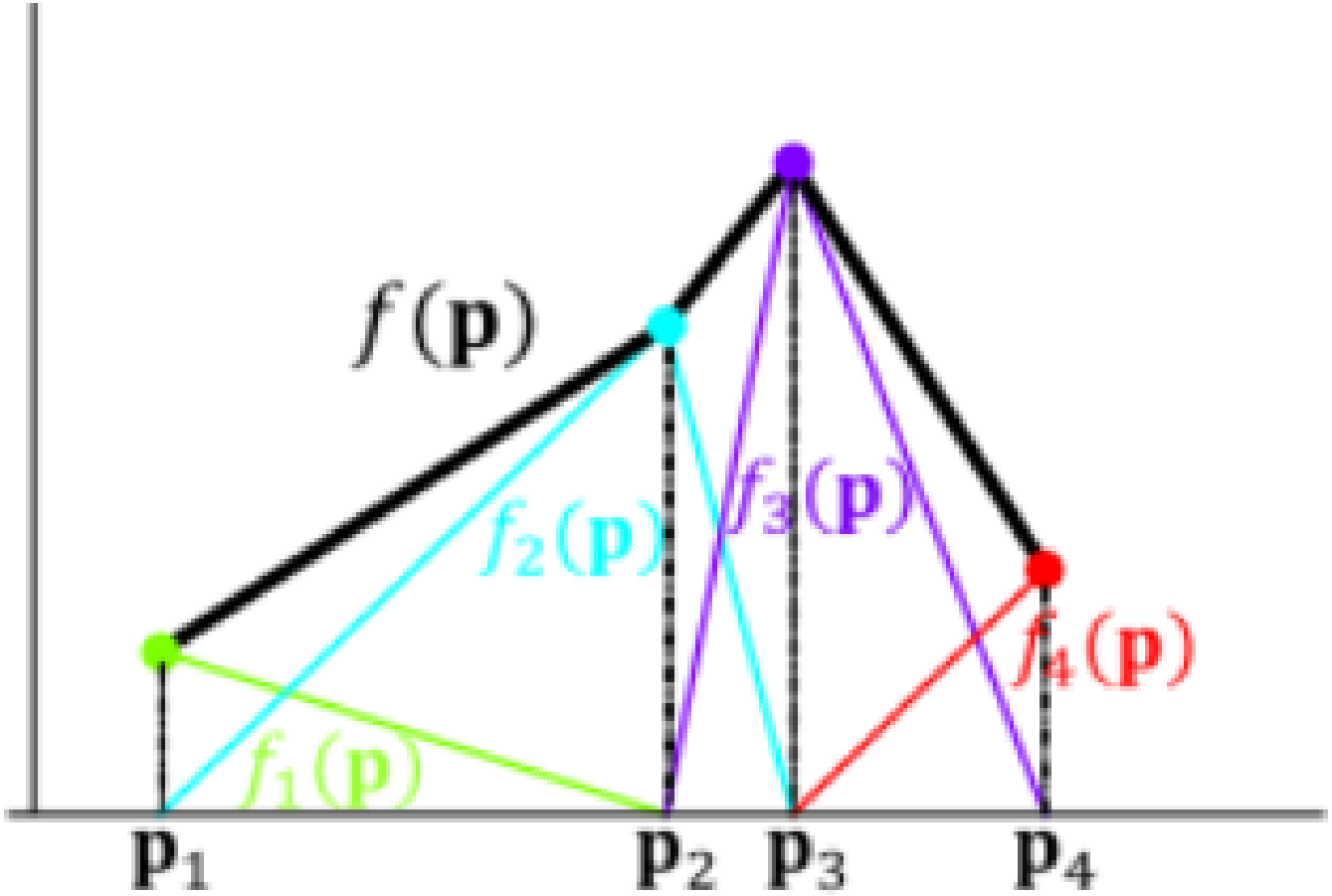


Figure: spiky bump function

Previously on linear interpolation

- How do we find a smoother bump function?

Shepard's Interpolation

- Shepard's bump function

$$f_i(x) = \frac{\|x - x_i\|^{-\alpha}}{\sum_{j=1}^{n} \|x - x_j\|^{-\alpha}} v_i,$$

  where $\alpha > 0$ and $\|\cdot\|$ is Euclidean norm.
- Notice that $f_i(x) = 1$ when $x = x_i$, and 0 otherwise.
- Thus, we can use $\sum_{i=1}^{n} f_i(x)$ as an interpolation directly **without** any parametrization.

Shepard's Interpolation

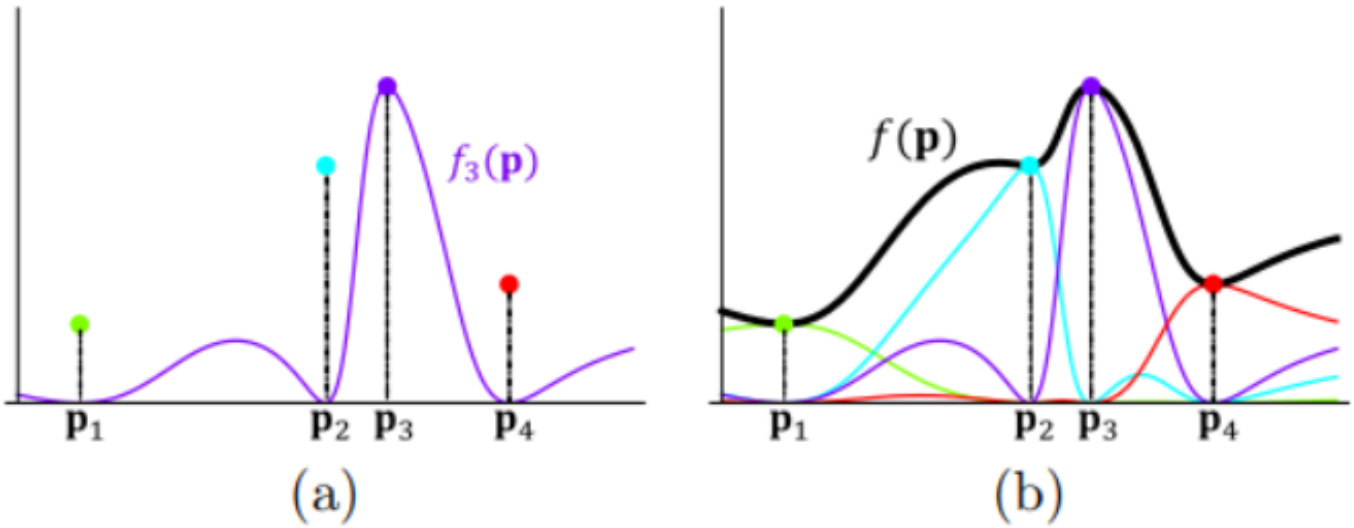


Figure: Shepard's Interpolation

Shepard's Interpolation

- However, Shepard's Interpolation has imposed an unnecessary feature on the interpolated function.
- f has a first-order derivative of 0 at every data point.

Shepard's Interpolation

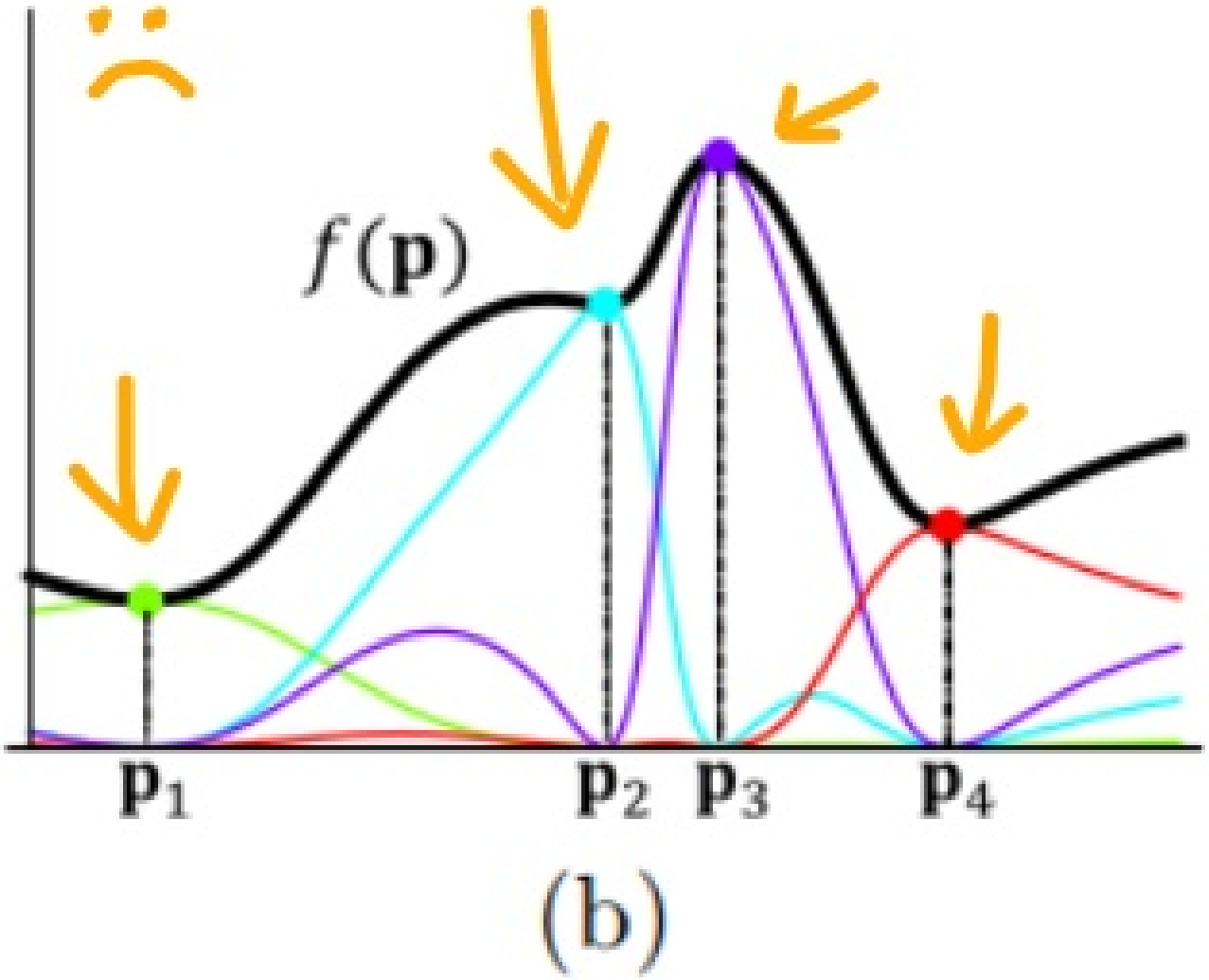


Figure: flat top

Radial Basis Function

Definition (Radial kernel)

Given a radial function $\phi : [0, \infty) \to \mathbb{R}$, a normed vector space $(V, \|\cdot\|)$ and a point $c \in V$, a radial kernel with center c is defined by

$$\phi_c(x) := \phi(\|x - c\|)$$

Radial Basis Function

Definition (Radial basis)

Given a radial function $\phi : [0, \infty) \to \mathbb{R}$ and a normed vector space $(V, \|\cdot\|)$, the set $\{\phi_c | c \in V\}$ is said to be a radial basis if for any sequence $(c_i)_{i=1}^n$, $\phi_{c_1}, \cdots, \phi_{c_n}$ are linearly independent and A is non-singular, where

$$A = \begin{bmatrix} \phi(\|x_1 - x_1\|) & \cdots & \phi(\|x_1 - x_n\|) \\ \vdots & \ddots & \vdots \\ \phi(\|x_n - x_1\|) & \cdots & \phi(\|x_n - x_n\|) \end{bmatrix}$$

Note: $\{\phi_c | c \in V\}$ is a basis of 'the space of functions'.

Radial Basis Function

- Some commonly used radial function:
- Gaussian: $\phi(r) = e^{-\epsilon r^2}$
- Multiquadratic: $\phi(r) = \sqrt{1 + \epsilon r^2}$
- Inverse quadratic: $\phi(r) = \frac{1}{1+\epsilon r^2}$
- Thin plate spline: $\phi(r) = r^2 ln(r)$
- Polyharmonic spline: $\phi(r) = \begin{cases} r^k & k = 1, 3, 5, \cdots \\ r^k ln(r) & k = 2, 4, 6, \cdots \end{cases}$

Radial Basis Function

- So how do we interpolate with radial basis functions?
- Note that $\phi_{x_i}(x_j) \neq 0$ when $i \neq j$.

Radial Basis Function

- Solving a linear equation is enough.
- $A\mathrm{c} = \mathrm{v}$
- $$A = \begin{bmatrix} \phi(\|x_1 - x_1\|) & \cdots & \phi(\|x_1 - x_n\|) \\ \vdots & \ddots & \vdots \\ \phi(\|x_n - x_1\|) & \cdots & \phi(\|x_n - x_n\|) \end{bmatrix}$$
- $c = [c_1, \cdots, c_n]^T$
- $v = [v_1, \cdots, v_n]^T$
- Note that A is non-singular according to the definition.

Radial Basis Function

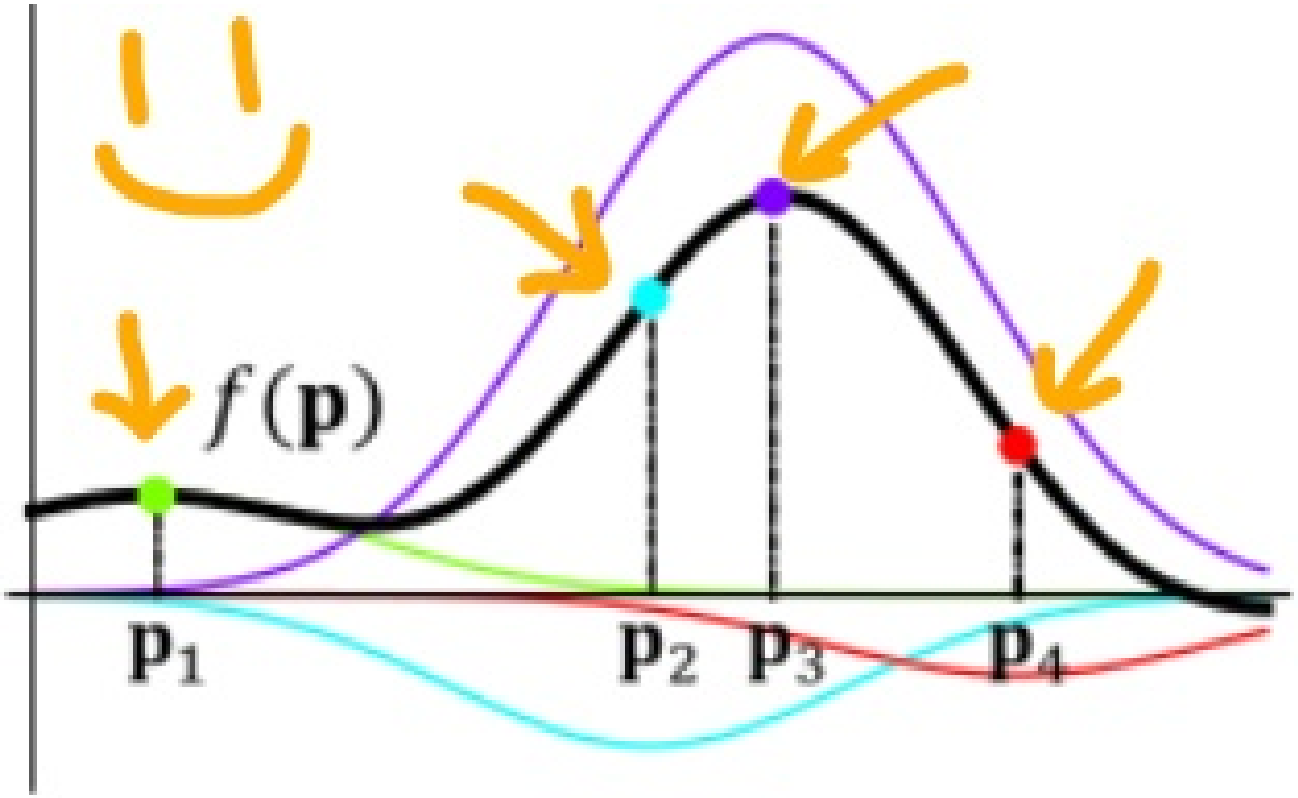


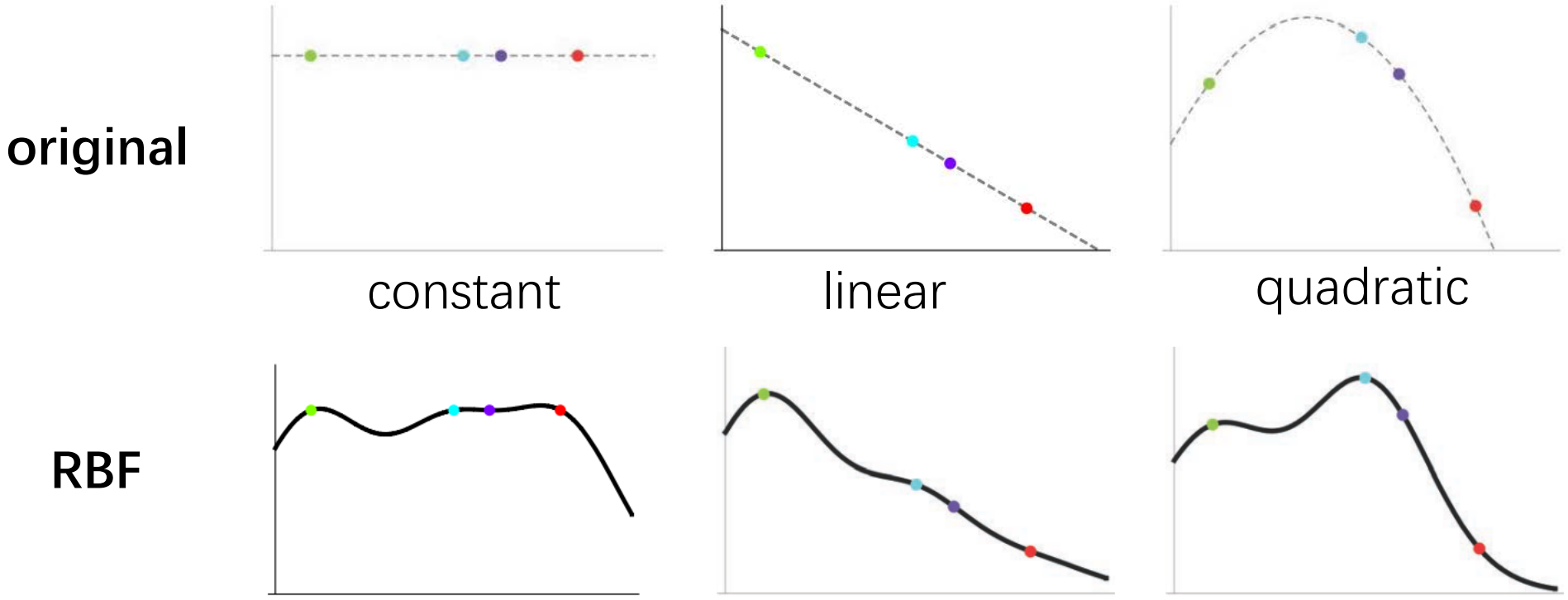
Figure: RBF Interpolation

Applications in 2 papers

张文涵 & 唐正纲

Reproducing Polynomials

- It would be desirable for an interpolation function to be no more complex than the original function



Reproducing Polynomials

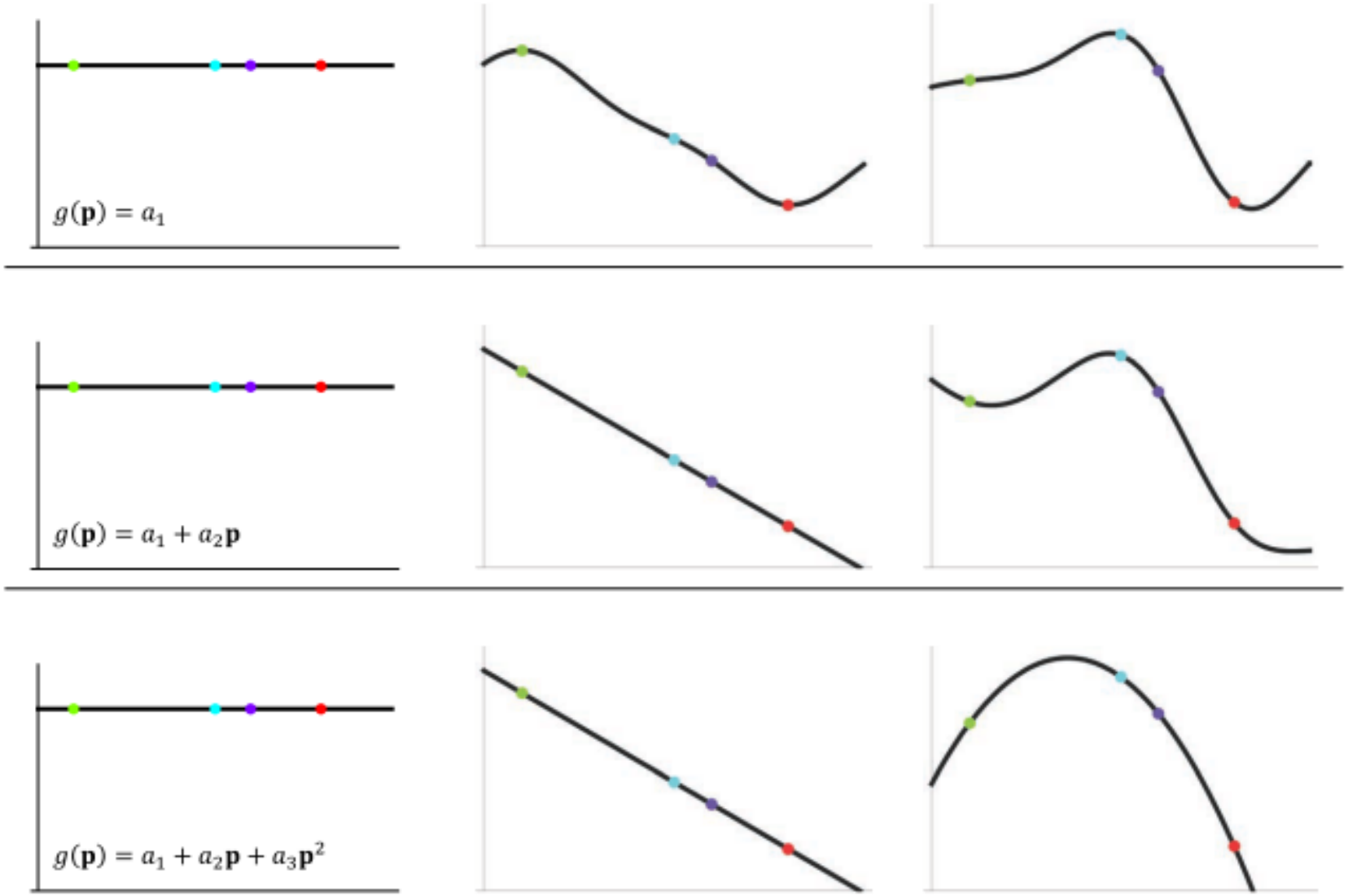
- introduce a polynomial term into the interpolation function
- $f(\boldsymbol{p}) = g(\boldsymbol{p}) + \sum_{i=1}^{n} f_i(\boldsymbol{p})$
- where $g(\boldsymbol{p})$ is a polynomial of maximum degree k
- $g(\boldsymbol{p}) = \sum_{i=1}^{l} a_i m_i(\boldsymbol{p})$
- where $m_i(\boldsymbol{p})$ is a monomial of the coordinate components of $\boldsymbol{p}$ of degree k or less, and a_i are coefficients
- For example, in 2D, p_x, p_y are the coordinates of $\boldsymbol{p}$
- Constant polynomial: $g(\boldsymbol{p}) = a_0$
- Quadratic polynomial: $g(\boldsymbol{p}) = a_0 + a_1 p_x + a_2 p_{\mathrm{y}} + a_3 p_x^2 + a_4 p_y^2 + a_5 p_x p_y$

Reproducing Polynomials

- Now, we have more unknowns(n+l) than constrains(n)
- add one more orthogonal constraint for each monomial term in $g(\boldsymbol{p})$
- $\sum_{i=1}^{n} c_i m_j(p_\mathrm{i}) = 0, \quad \text{for } j = 1, \dots, l$
- Linear system: $\begin{bmatrix} A & M \\ M^T & 0 \end{bmatrix} \begin{bmatrix} c \\ a \end{bmatrix} = \begin{bmatrix} v \\ 0 \end{bmatrix}$

- $M = \begin{bmatrix} m_1(p_1) & \dots & m_l(p_1) \\ \dots & \dots & \dots \\ m_1(p_n) & \dots & m_l(p_n) \end{bmatrix}, \quad a = [a_1, \dots, a_l]^T$

Reproducing Polynomials



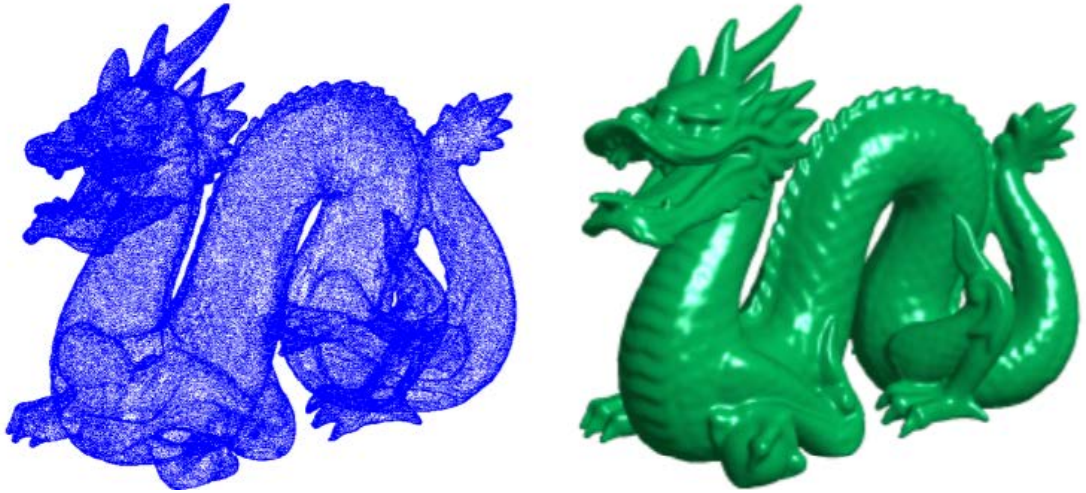
Practical issues in surface reconstruction

- **Reconstruction and Representation of 3D Objects with Radial Basis Functions**
- **A multi-scale approach to 3D scattered data interpolation with compactly supported basis functions**

Reconstruction and Representation of 3D Objects with Radial Basis Functions

Introduction

- Application：
  - reconstructing smooth, manifold surfaces
  - Interpolating incomplete meshes (hole-filling)
- Challenge：
  - Storage： O(n^2)
  - Arithmetic operation: O(n^3)
  - 3000GB, 2001



Fitting a Radial Basis Function (RBF) to a 438,000 point-cloud.

Introduction

- 如何形式化问题?
  - Section 2： formulate the surface fitting problem as a scattered data interpolation problem.
  - Section 3： RBFs
- 如何加速计算?
  - Section 4： new fast methods which overcome problems in the past.
  - Section 5： RBF center reduction.
  - Section 6： RBF approximation as a means of smoothing noisy surface data and demonstrates this by reconstructing smooth surfaces from noisy LIDAR (laser rangefinder) data.
- 如何应用?
  - Section 7： iso-surface extraction.
  - Section 8\9： results

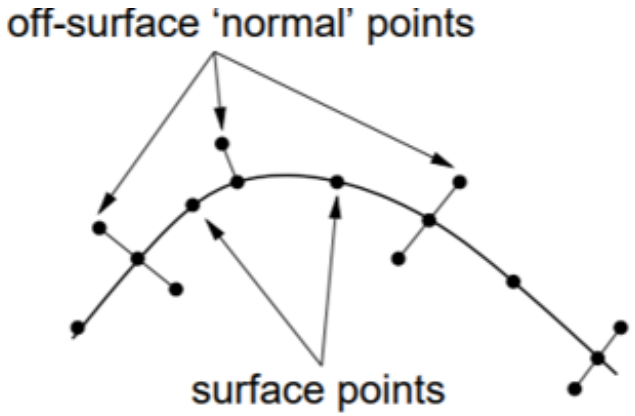
Fitting an implicit function to a surface

- find a function f which implicitly defines a surface M and satisfies the equation

$$f(x_i, y_i, z_i) = 0, \qquad i = 1, \ldots, n,$$

- off-surface points are appended to the input data

$$\begin{aligned} f(x_i, y_i, z_i) &= 0, & i &= 1, \ldots, n & &(\textit{on}\text{-surface points}), \\ f(x_i, y_i, z_i) &= d_i \neq 0, & i &= n+1, \ldots, N & &(\textit{off}\text{-surface points}). \end{aligned}$$



Radial Basis Function interpolation

Problem 3.1. *Given a set of distinct nodes* $X = \{x_i\}_{i=1}^N \subset \mathbb{R}^3$ *and a set of function values* $\{f_i\}_{i=1}^N \subset \mathbb{R}$, *find an* interpolant $s : \mathbb{R}^3 \to \mathbb{R}$ *such that*

$$s(x_i) = f_i, \qquad i = 1, \ldots, N. \tag{2}$$

$$S = \{s \in \mathrm{BL}^{(2)}(\mathbb{R}^3) : s(x_i) = f_i, \quad i = 1, \ldots, N\}.$$

$\mathrm{BL}^{(2)}(\mathbb{R}^3)$: Beppo-Levi space with square integrable second derivatives

Define semi-norm: a measure of smoothness

$$\|s\|^2 = \int_{\mathbb{R}^3} \left(\frac{\partial^2 s(x)}{\partial x^2}\right)^2 + \left(\frac{\partial^2 s(x)}{\partial y^2}\right)^2 + \left(\frac{\partial^2 s(x)}{\partial z^2}\right)^2$$
$$+ 2\left(\frac{\partial^2 s(x)}{\partial x \partial y}\right)^2 + 2\left(\frac{\partial^2 s(x)}{\partial x \partial z}\right)^2 + 2\left(\frac{\partial^2 s(x)}{\partial y \partial z}\right)^2 dx.$$

Radial Basis Function interpolation

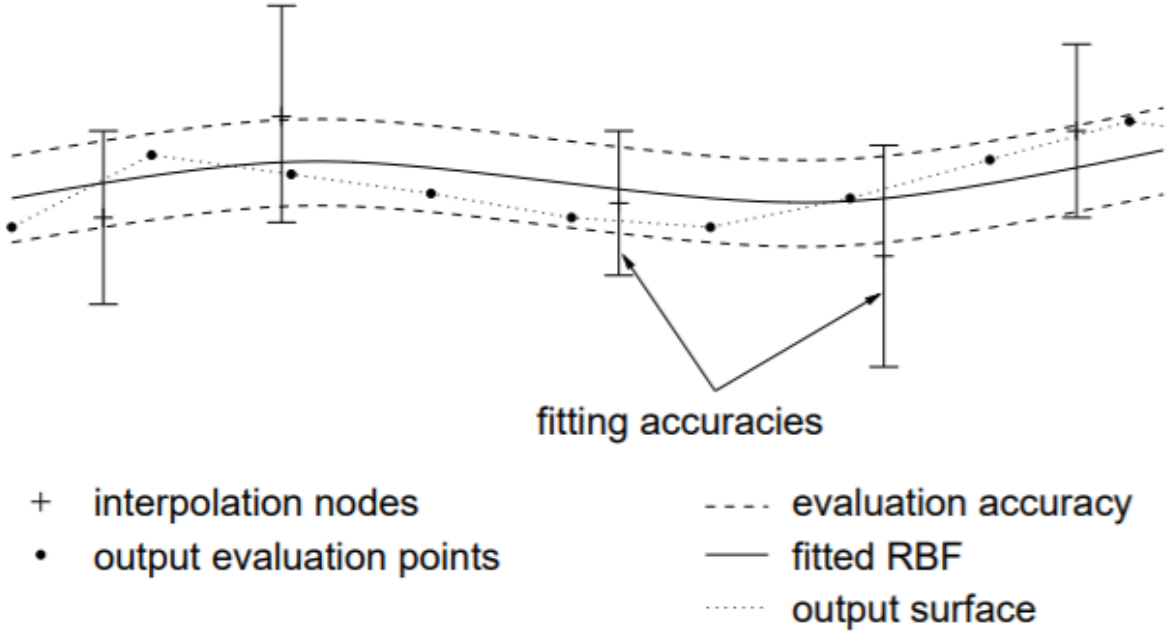
$$s^\star = \arg\min_{s \in S} \|s\|, \quad \Rightarrow \quad s^\star(x) = p(x) + \sum_{i=1}^{N} \lambda_i |x - x_i|,$$

where p is a linear polynomial, the coefficients λ_i are real numbers and $|\cdot|$ is the Euclidean norm on $\mathbb{R}^3$.

RBF: $$s(x) = p(x) + \sum_{i=1}^{N} \lambda_i \phi(|x - x_i|),$$

ϕ: basis function(compact support/non-compact support)
x_i :the centers of the RBF

Fast methods



	Direct methods	Fast methods
Fitting		
storage required	$N(N+1)/2$	$O(N)$
flops to solve system	$N^3/6+O(N^2)$	$O(N\log N)$
Evaluation		
flops per evaluation	$O(N)$	$O(1)$ after $O(N\log N)$ setup

RBF center reduction

A simple greedy algorithm:

1. Choose a subset from the interpolation nodes x_i and fit an RBF only to these.
2. Evaluate the residual, $\varepsilon_i = f_i - s(x_i)$at all nodes.
3. If $|\varepsilon_i|<$ fitting accuracy then stop.
4. Else append new centers where ε_i is large.
5. Re-fit RBF and goto 2

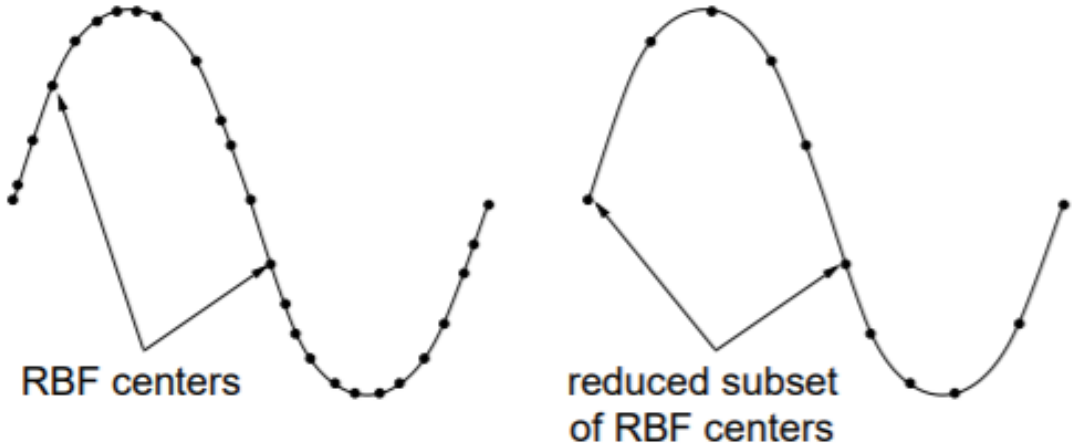
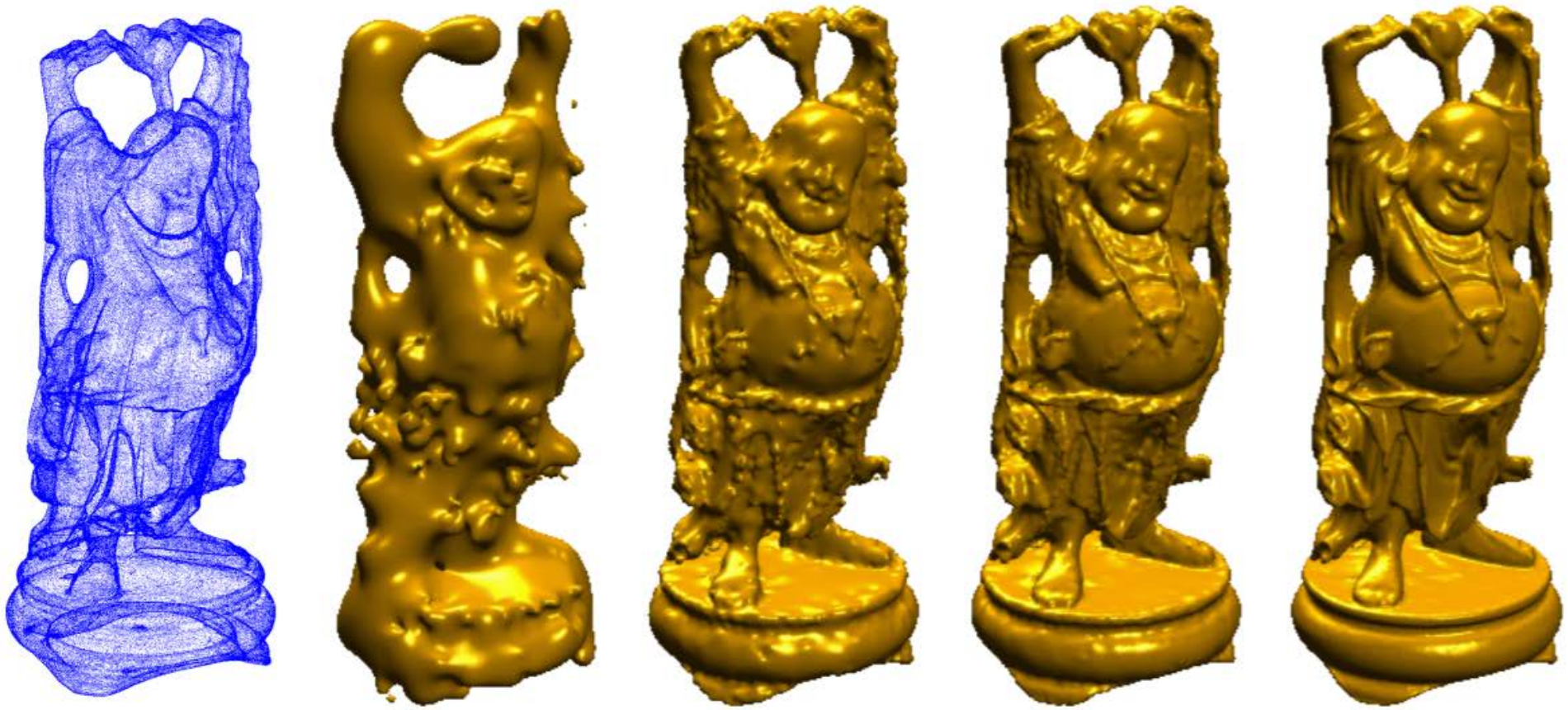


Figure 7: Illustration of center reduction.

RBF center reduction



RBF approximation of noisy data

$$\min_{s \in \mathrm{BL}^{(2)}(\mathbb{R}^3)} \rho \|s\|^2 + \frac{1}{N} \sum_{i=1}^{N} \left(s(x_i) - f_i\right)^2 \quad \Rightarrow \quad s^\star(x) = p(x) + \sum_{i=1}^{N} \lambda_i |x - x_i|,$$

Linear system:

$$\begin{pmatrix} A - 8N\pi\rho I & P \\ P^\mathsf{T} & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ c \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}$$

RBF approximation of noisy data

(a) (b) (c)

Figure 8: RBF approximation of noisy LIDAR data. (a) 350,000 point-cloud, (b) the smooth RBF surface approximates the original point-cloud data, (c) cut-away view illustrating the RBF distance field and the preservation of the gap between the arm and the torso.

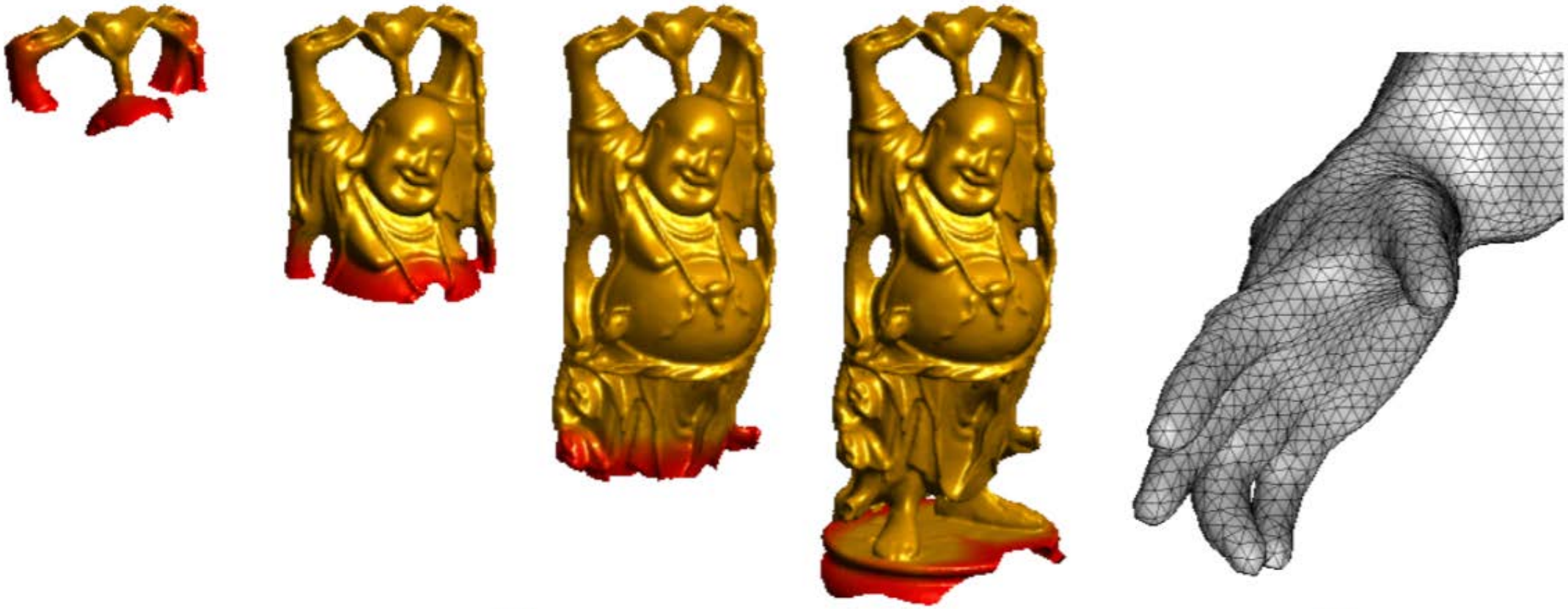
RBF approximation of noisy data

(a) (b) (c)

Figure 9: (a) Exact fit, (b) medium amount of smoothing applied (the RBF approximates at data points), (c) increased smoothing.

Surface evaluation

In any case, only a small subset of centers is required to seed the surface



(a) (b)

Figure 10: Iso-surfacing an RBF. (a) Surface-following from a single seed, (b) example of an optimised mesh.

Result

Figure	Number of surface points	Number of inter--polation nodes	Number of RBF centers	Peak RAM (MB)	Fitting time	Surfacing time	Relative accuracy
Face	14,806	29,074	3,564	29	68s	27s	7×10^{-4}
Hand	13,348	26,696	4,299	29	97s	32s	1×10^{-3}
Dragon	437,645	872,487	72,461	306	2:51:09	0:04:40	8×10^{-4}
Buddha	543,652	1,086,194	80,518	291	4:03:26	0:04:07	5×10^{-4}
Cherub statue	331,135	662,269	83,293	187	3:09:06	0:06:41	4×10^{-4}
Skeleton hand	327,323	654,645	85,468	188	3:08:44	0:04:04	3×10^{-4}
LIDAR statue	345,910	518,864	518,864	390	3:08:21	0:25:39	6×10^{-3}

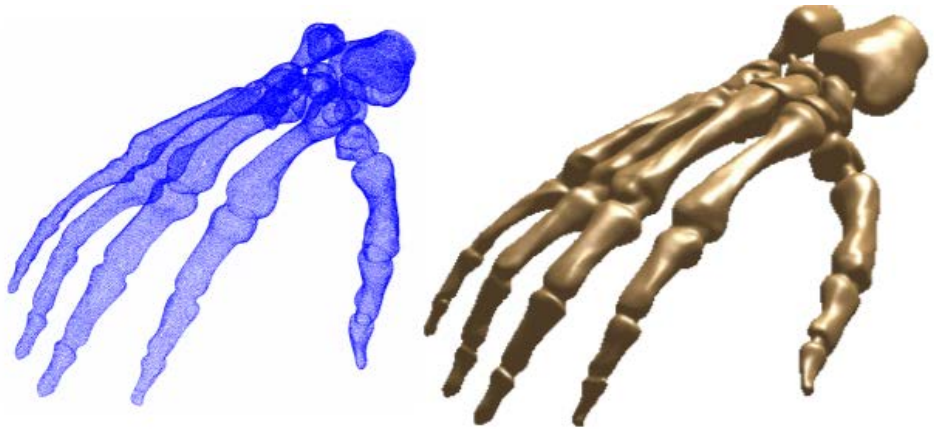
Table 2: Comparison of RBF fitting and evaluation times on a 550MHz PIII with 512MB RAM.

	Original mesh			New mesh			RBF representation	
Figure	# vertices	# facets	Storage	# vertices	# facets	Storage	RBF centers	Storage
Dragon	437,645	847,414	15.4MB	126,998	254,016	4.5MB	72,461	1.4MB
Buddha	543,652	1,086,798	19.6MB	96,766	193,604	3.5MB	80,518	1.6MB
Skeleton hand	327,323	654,666	11.8MB	81,829	163,698	2.9MB	85,468	1.7MB

Advantage of storage and time cost

Result

Fill small hole



Represent complicated object with sufficient sampling

A Multi-scale Approach to 3D Scattered Data Interpolation with Compactly Supported Basis Functions

Introduction

- Studying geometric modeling with implicit surfaces remains active.
- Interpolation and approximation of scattered data with RBFs is popular.

- A hard choice: local or global RBFS, which are better?
  - Local: simpler and faster computation procedure, more details preserved, but sensible in density of points.
  - Global: sophisticated but useful in repairing incomplete data.

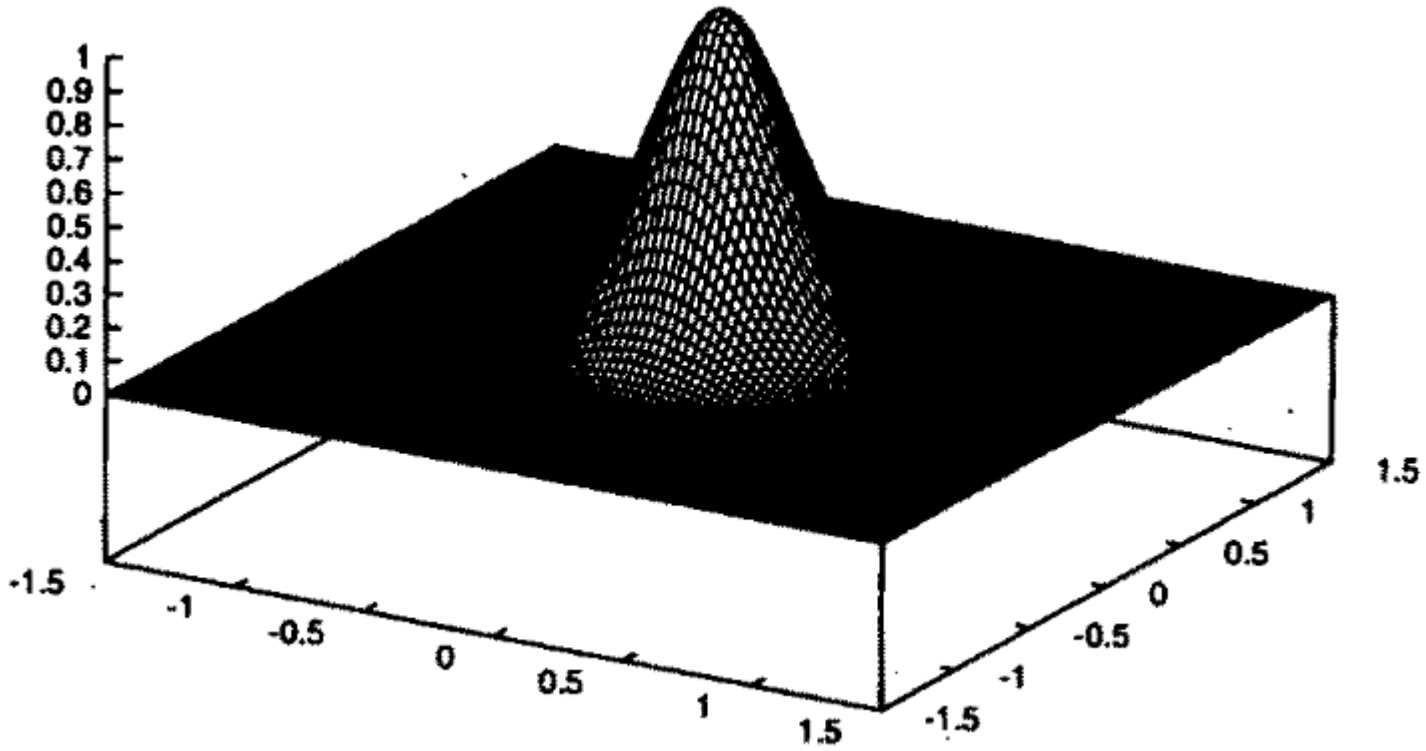
- Combine them in a hierarchical way!
  - We need a hierarchy from sparse to fine (and from global to local).

Single-level interpolation

- Given a point set **P = {p_i}** sampled from a surface.
- Every point has it's normal which indicates the surface's orientation.
- We want to generate a 3D function and its zero level-set **{x | f(x) = 0}** interpolates **P**. (then rebuild the surface)

- We use local RBFs as **f**:
- $f(x) = \sum_{p_i \in P} \psi_i(x) = \sum_{p_i \in P} [g_i(x) + \lambda_i] \phi_\sigma(||x - \mathrm{p_i}||)$
- $\phi_\sigma = \phi\left(\frac{r}{\sigma}\right) \quad \phi(r) = (1-r)_+^4(4\mathrm{r}+1)$
- $g_i,\ \lambda_i$ are unknown functions or coefficients to be determined.

ϕ

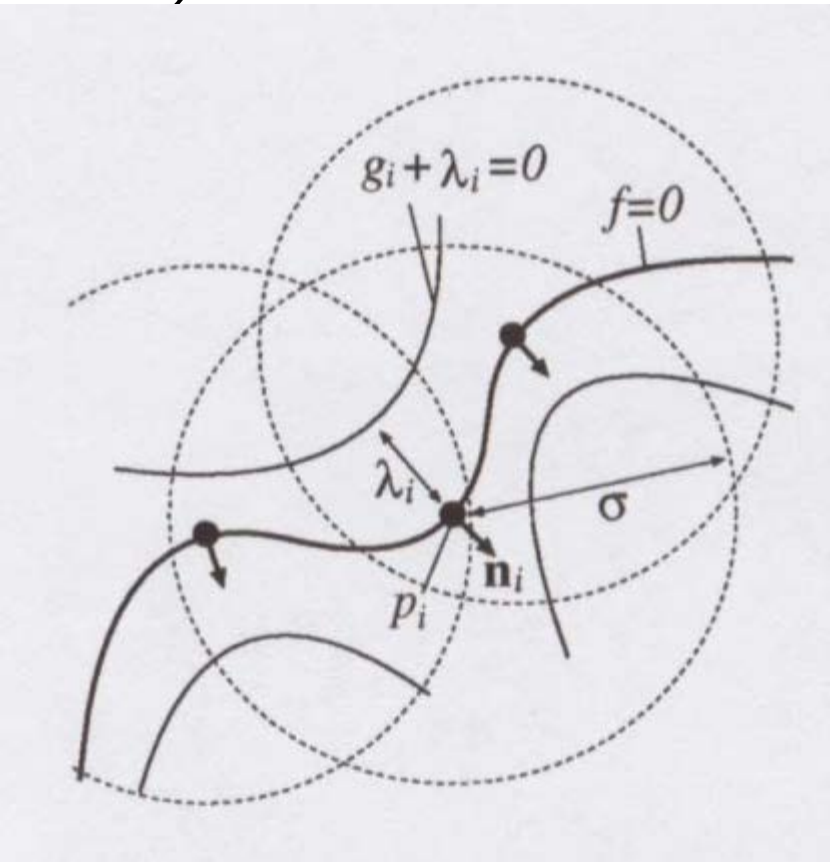


Single-level interpolation

- g_i is designed to interpolate p_i 's neighborhood (radius = σ)

- $g_i(x) = \min_h \sum_{(u_j,v_j,w_j)=p_j \in P} \phi_\sigma(\|p_j - p_i\|)\left(w_j - h(u_j, v_j)\right)^2$
- $h(u,v) \equiv Au^2 + Bv^2 + Cuv$
- (u,v,w) is a coordinate system,where w has the same direction as $\mathbf{p_i}$'s normal.

- $\lambda_{1\sim n}$ should be chosen to make $f(p_j) = 0$ for any j
- (it can be regarded as the distance g_i transposed)



Single-level interpolation

- Solving out values of λ_i is indeed solving linear equations.

- When the coefficient matrix is positive definite and scarce, **preconditioned biconjugate gradient method** is very efficient! (details are beyond the paper)

- https://link.springer.com/article/10.1007/BF02123482
- https://blog.csdn.net/OORRANNGGE/article/details/82744157

Single-level interpolation

- How to choose a good σ?

- Intuitively it should be related to points' density (average distances between point in a small neighborhood).

- One way to determine σ:
- Divide all points recursively to 8 parts like octree, until the subspace have $\leq$ 8 points, then calculate it's diameter (maximum distance between points) $\sigma = c *$ average of diameters.

Multi-level interpolation via offsetting

- But average is not a good idea!
  - using small σ + Irregular sampling may cause a bad interpolation.
  - Tiny and sophisticated details need small σ.

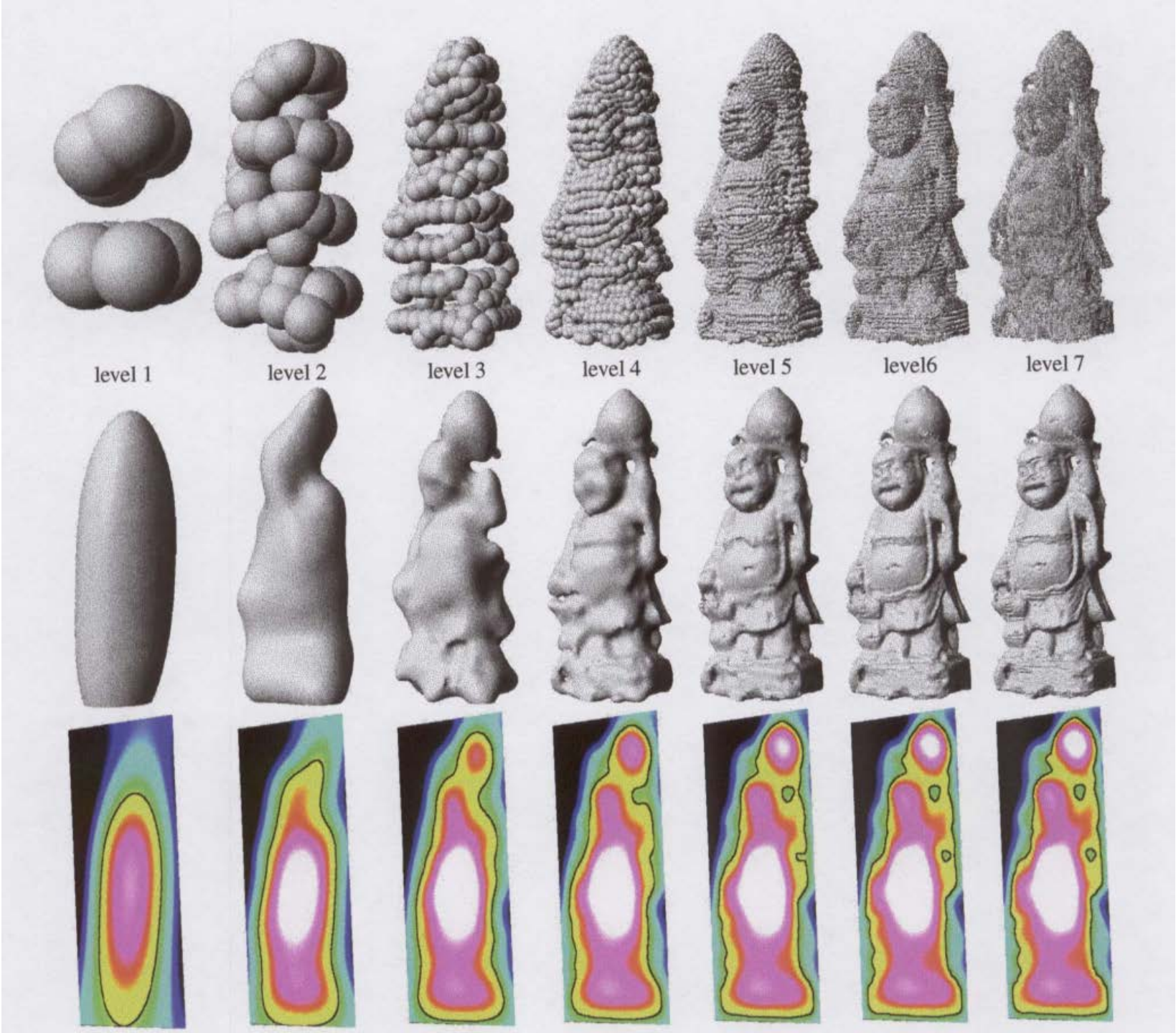
- Hierarchy can combine all sizes of σ!

Multi-level interpolation via offsetting

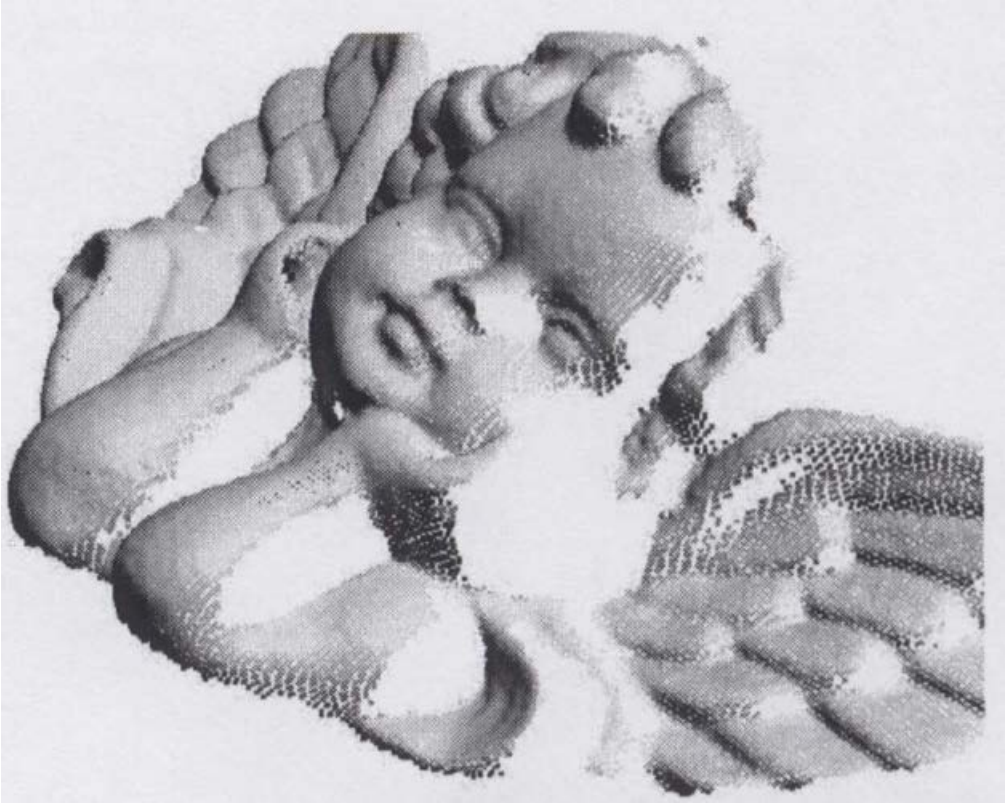
- Hierarchy of point sets:
- $\{P_1, P_2, \cdots, P_m\} = P$ (divided like octree, P_k contains all centroids of k-level subspace (so $|P_k| <= 8^k$))
- σ_k is used for k-level's interpolation, $\sigma_k = \frac{\sigma_{k-1}}{2}$, $\sigma_1 = cL$ (L = diameter of 1-level subspace, c = 0.75 chosen intuitively)

- 1-level interpolation is just like before (using σ_1 as neighborhood's radius to find $f_1(x)$ over P_1)
- K-level interpolation is generated on the foundation of k-1 level:
- $f_k(x) = f_{k-1}(x) + \sum_{p_i^k \in P^k} [g_i^k(x) + \lambda_i^k] \phi_{\sigma_k} (\|x - p_i^k\|)$
- When $\sigma_k \leq \sigma_0$ (σ_0 is what we calculated in single level interpolation), it's time to terminate.

Results



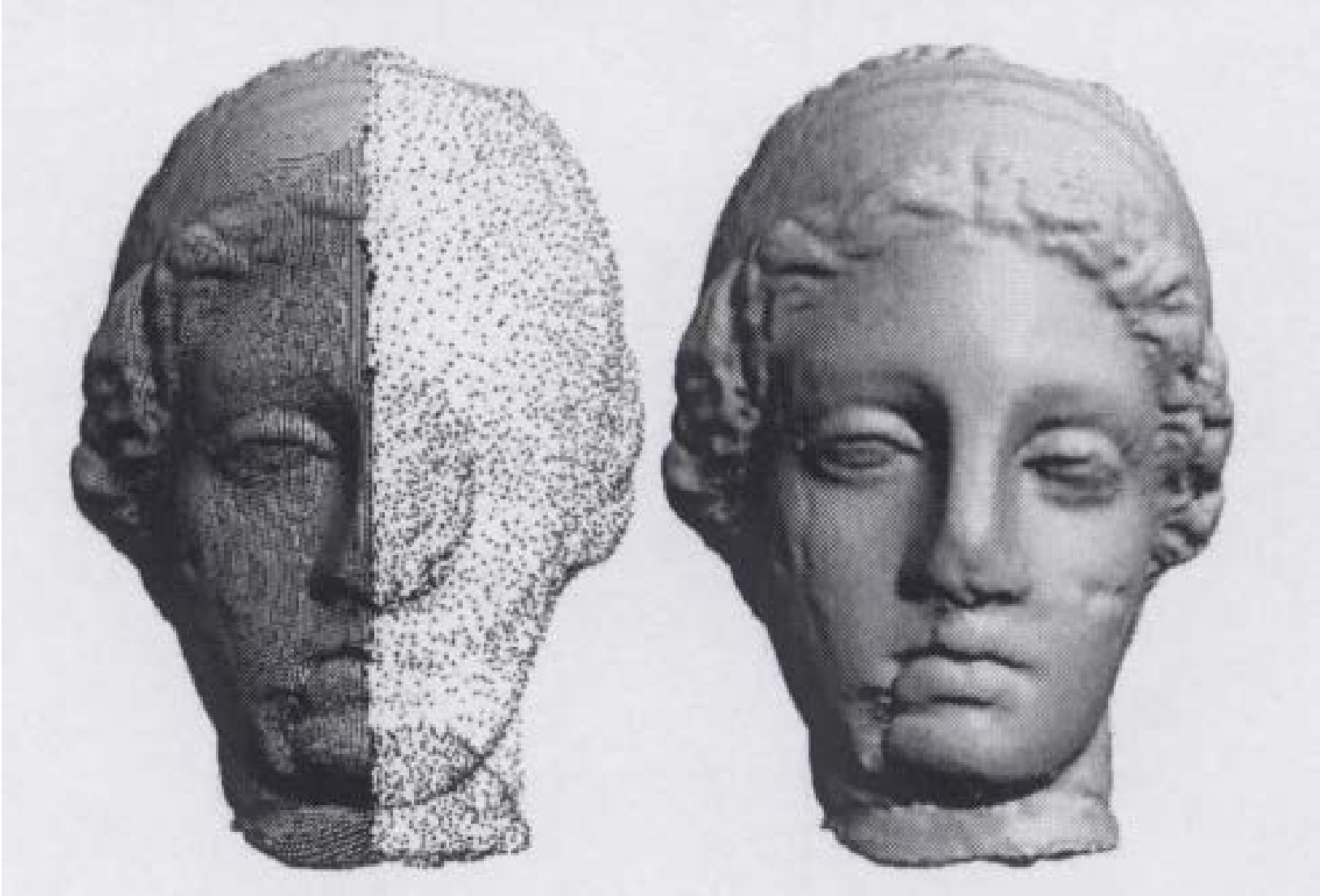
Results



Results

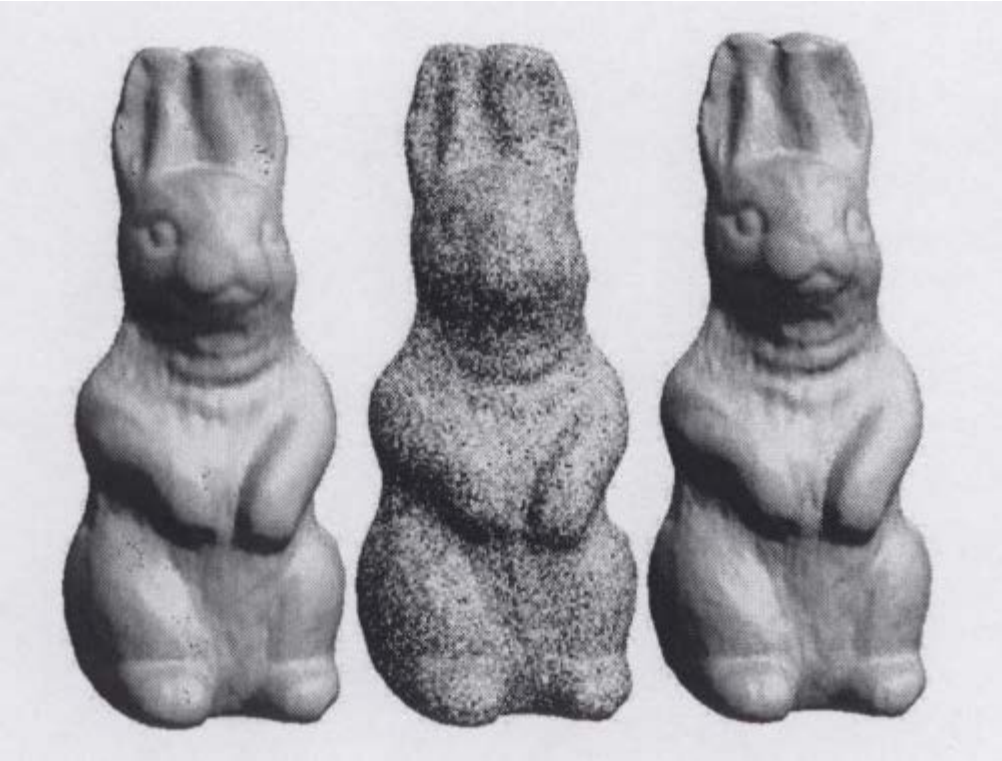
- Compared with Fast RBF.
- No visual difference, but 4 times faster!

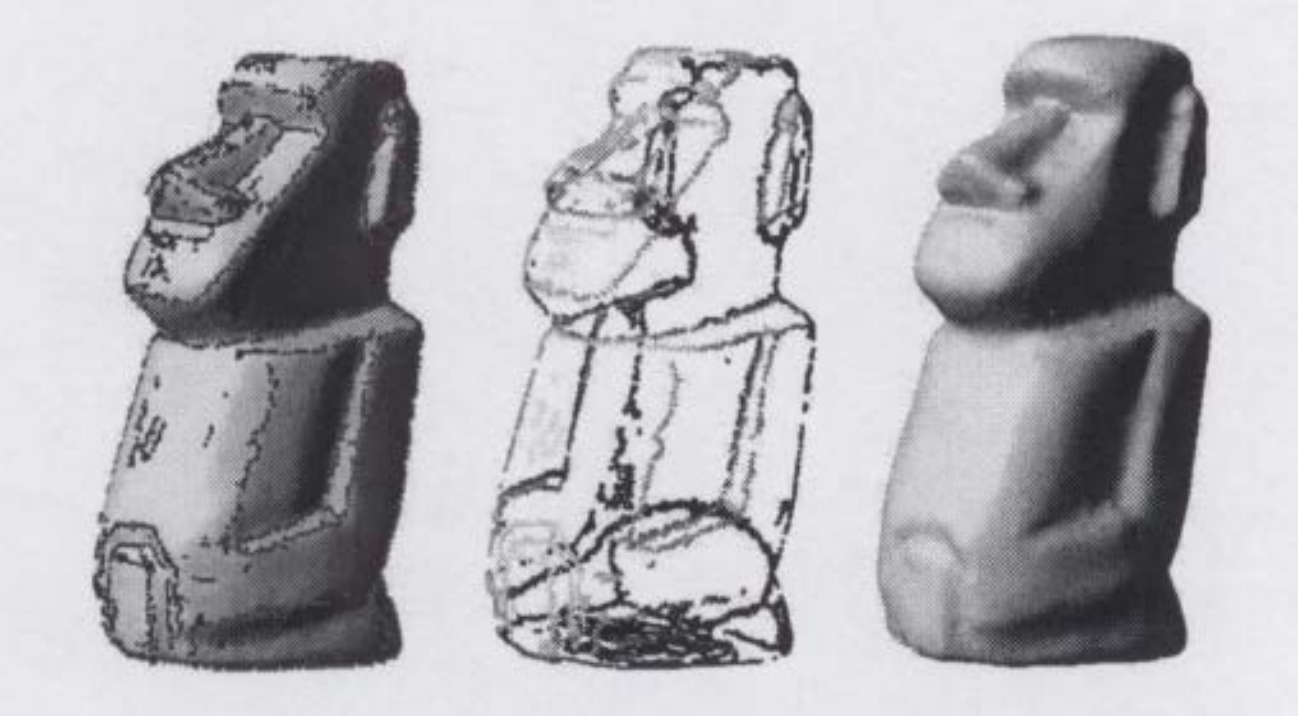
Results



- 90% of right part's points are removed.
- But interpolation has no visual difference between left and right!

Robustness





- noised or zeroed normals don’t disturb the result so much.
- (2% zero)

- Only edge points.