scattered data interpolation for computer graphics

J.P. Lewis
Weta Digital

Ken Anjyo
OLM Digital

Fred Pighin (*)
Google Inc

SIGGRAPH Asia 2010

Course: Scattered Data Interpolation for Computer Graphics

schedule

9:00-9:15	Introduction and survey of applications
9:15-9:30	Non-RBF algorithms
9:30-9:40	break
9:40-10:15	RBF and variants; connection to Laplacian splines
10:15-10:50	Case studies: Skinning, NPR Shading, Stereoscopic 3D
10:50-11:00	break
11:00-11:15	Greens functions, RBF and Gaussian process connections
11:15-12:00	Functional analysis, RBF and RKHS connections
12:00-12:15	Open problems, questions, conclusion

course website

- check back for corrections, errata:

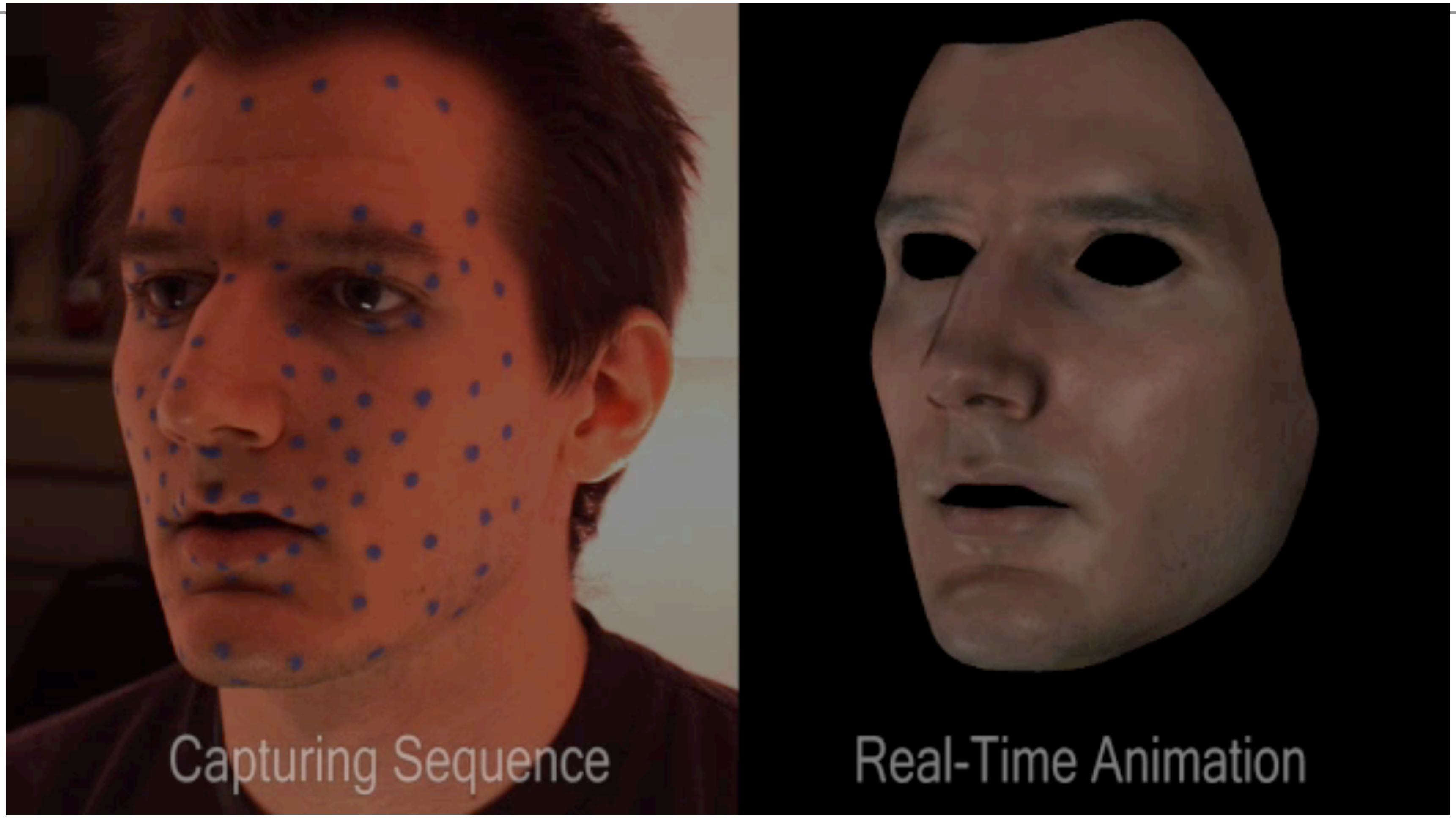
Any mathematical document of this size will contain typos.
Please obtain a corrected version of these notes at:
http://scribblethink.org/Courses/ScatteredInterpolation

http://scribblethink.org/Courses/ScatteredInterpolation

- contact us if you find a bug: zilla@computer.org, anjyo@olm.co.jp

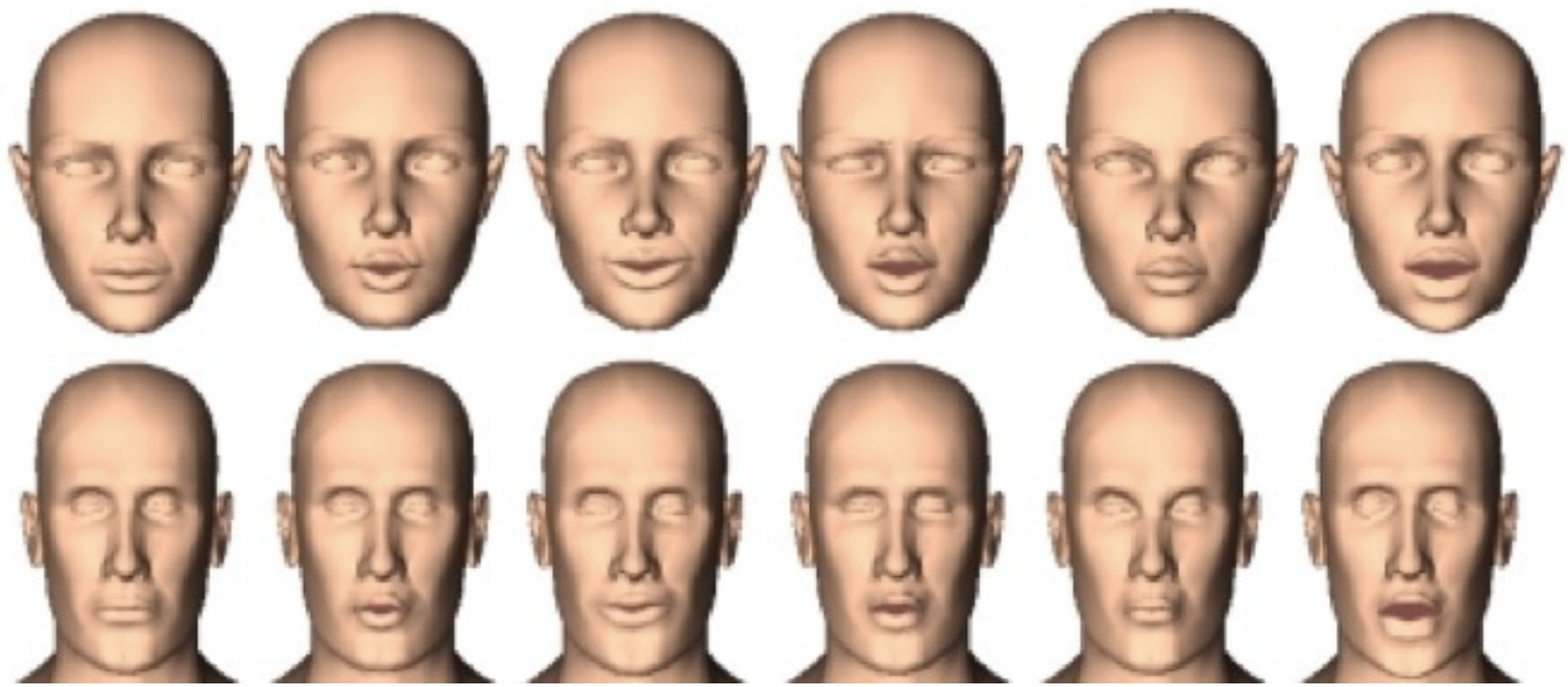
quick survey of applications

wrinkles



Bickel, Lang, Botsch, Otaduy, Gross
Pose-Space Animation and Transfer of Facial Details
ACM SCA 2008

facial retargeting

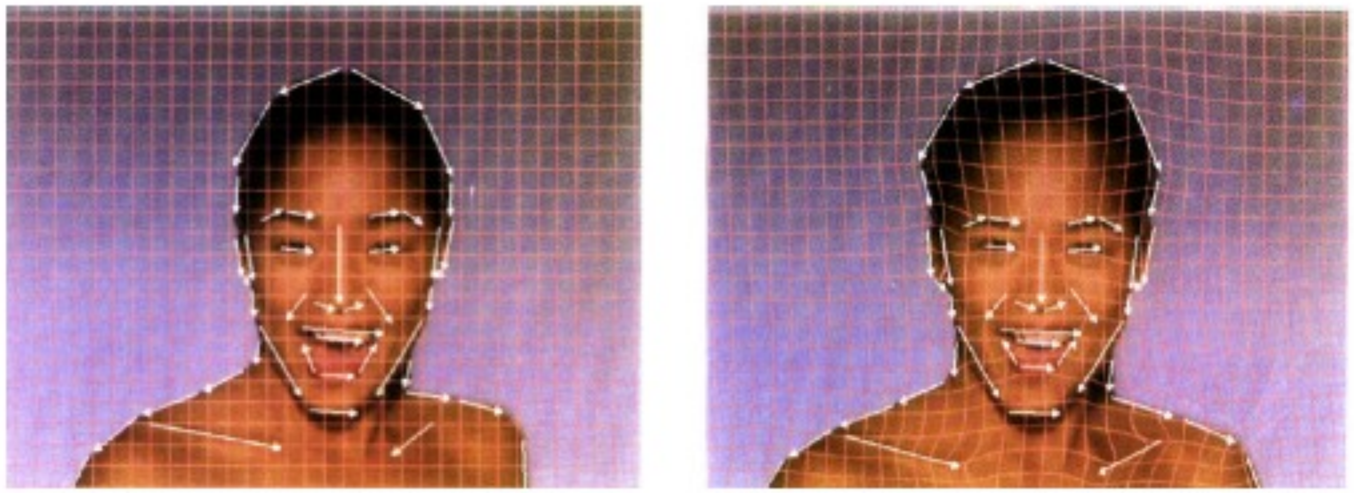


Noh and Neumann, Expression Cloning, SIGGRAPH 2001

editing NPR light and shade

H. Todo, K. Anjyo, W. Baxter, and T. Igarashi.
Locally controllable stylized shading.
SIGGRAPH 07

image registration, morphing



from Beier & Neely, Feature-Based Image Metamorphosis, SIGGRAPH 1992

inpainting

Figure 4. (a) Old photograph. (b) Selected retouching area. (c) Result produced with our algorithm. The algorithm can repaint disconnected areas.

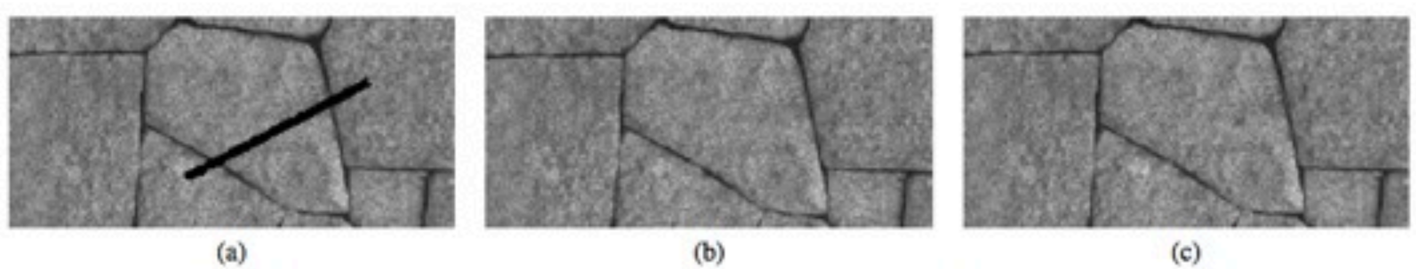


Figure 5. (a) "Stone", from [2]. (b) Result produced with Hirani and T. Totsuka's algorithm [2]. (c) Restored image obtained with our algorithm.

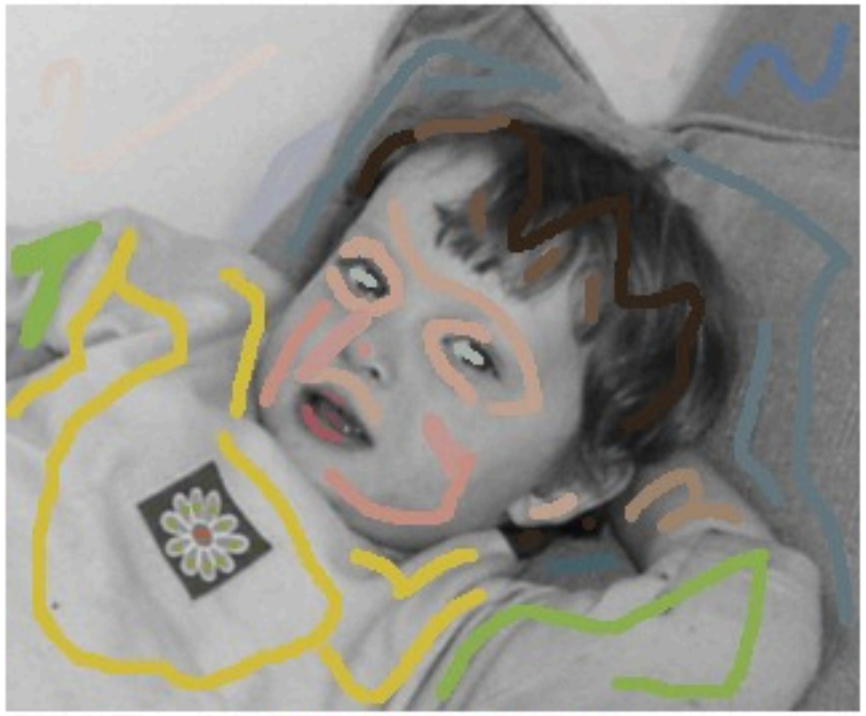
Savchenko, Kojekine, Unno
A Practical Image Retouching Method

implicit surfaces

(a) (b)

Carr, Beatson et al.
Reconstruction and Representation of 3D Objects with Radial Basis Functions
SIGGRAPH 2001

colorization

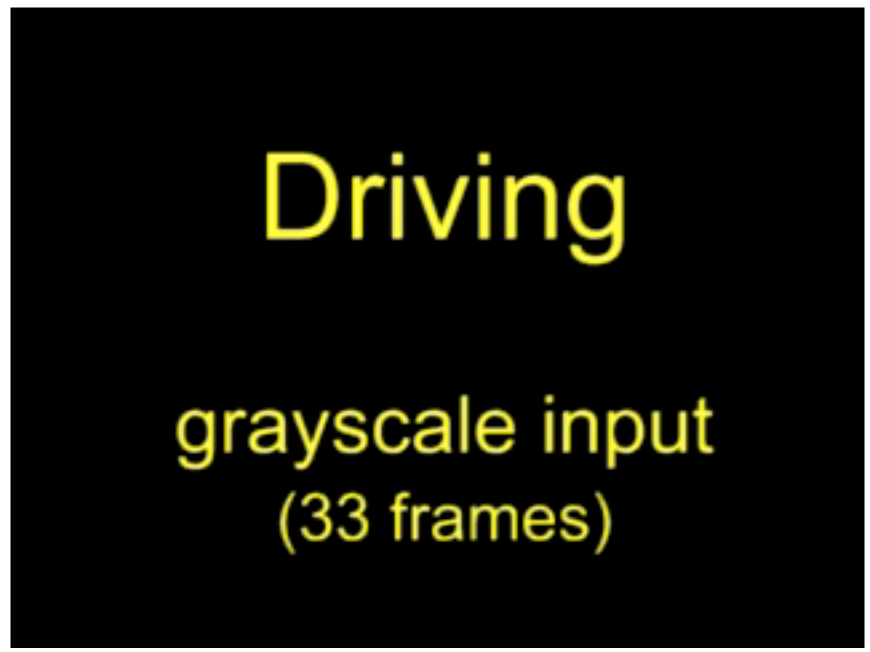


Marked B/W image

Result

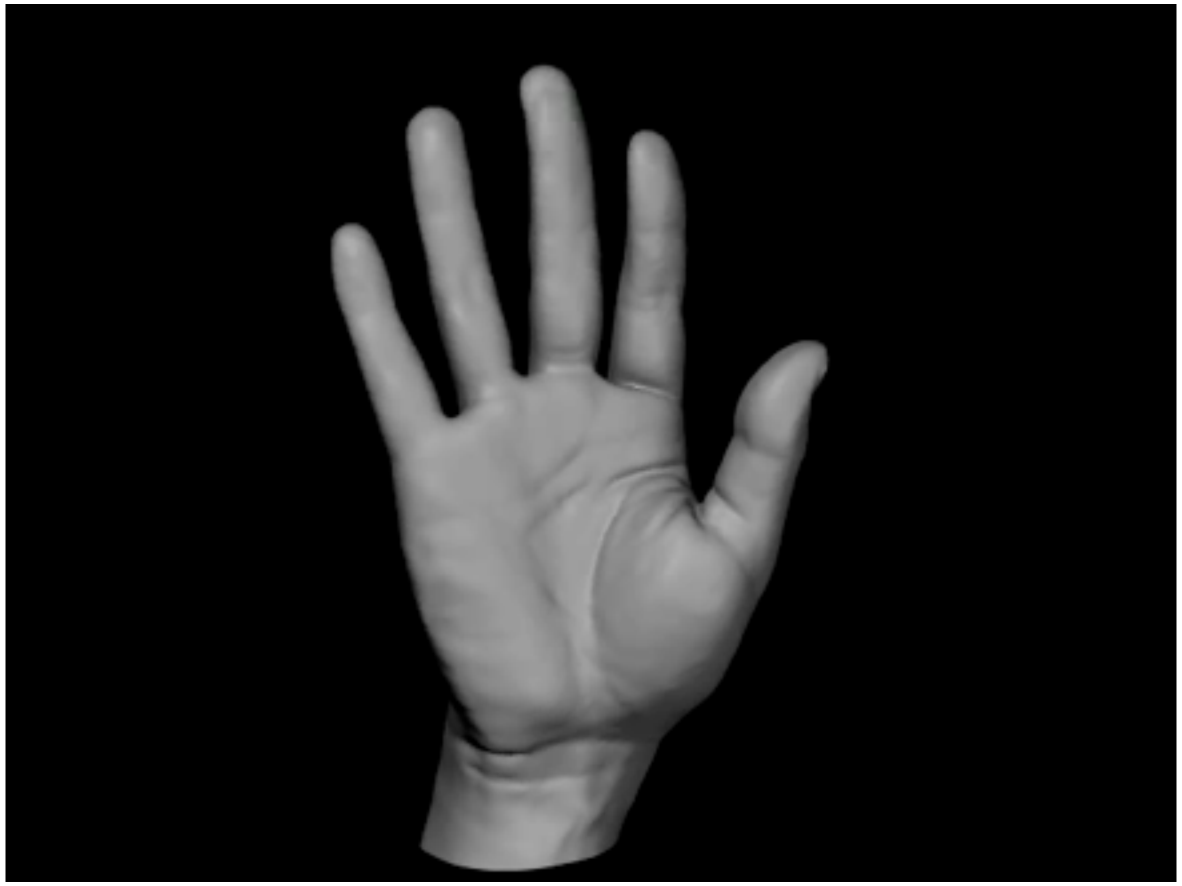
Levin, Lischinski, Weiss,
Colorization using Optimization
SIGGRAPH 2004

colorization



Levin, Lischinski, Weiss,
Colorization using Optimization
SIGGRAPH 2004

body animation

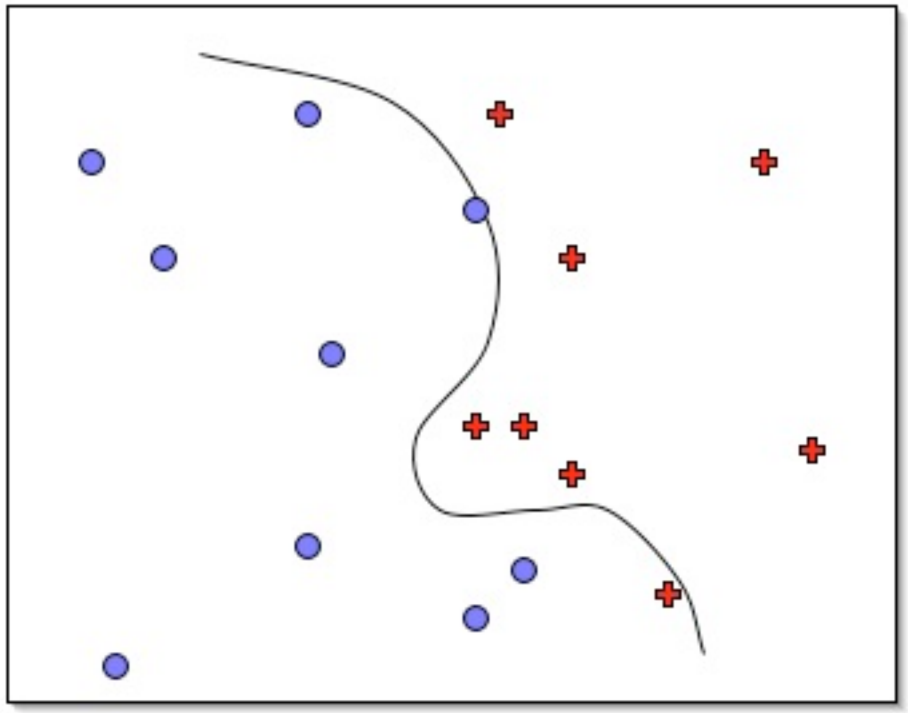


Kurihara & Miyata,
Modeling Deformable Human Hands from Medical Images,
ACM SCA 2004

fluids

- "meshfree"

machine learning

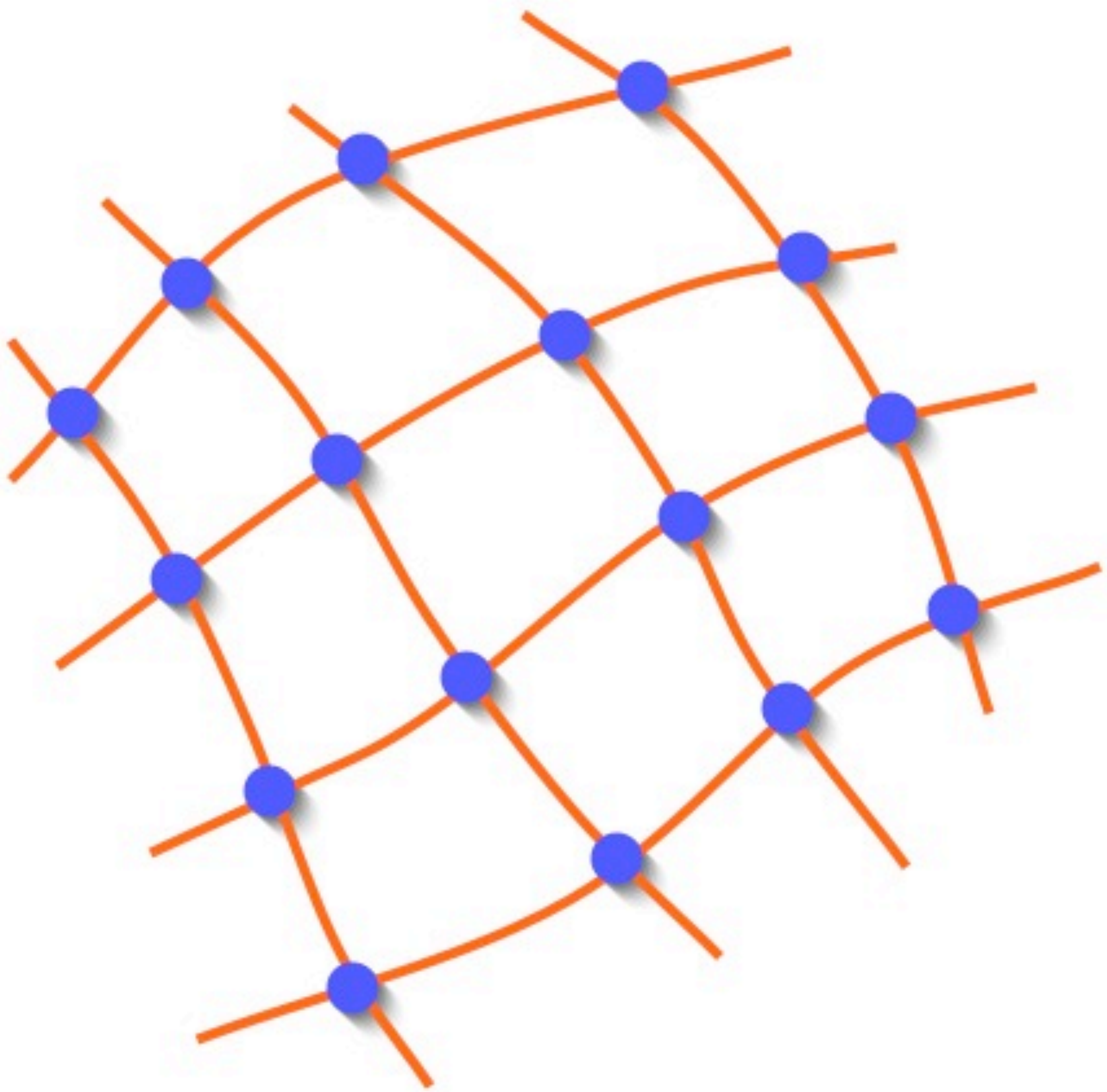


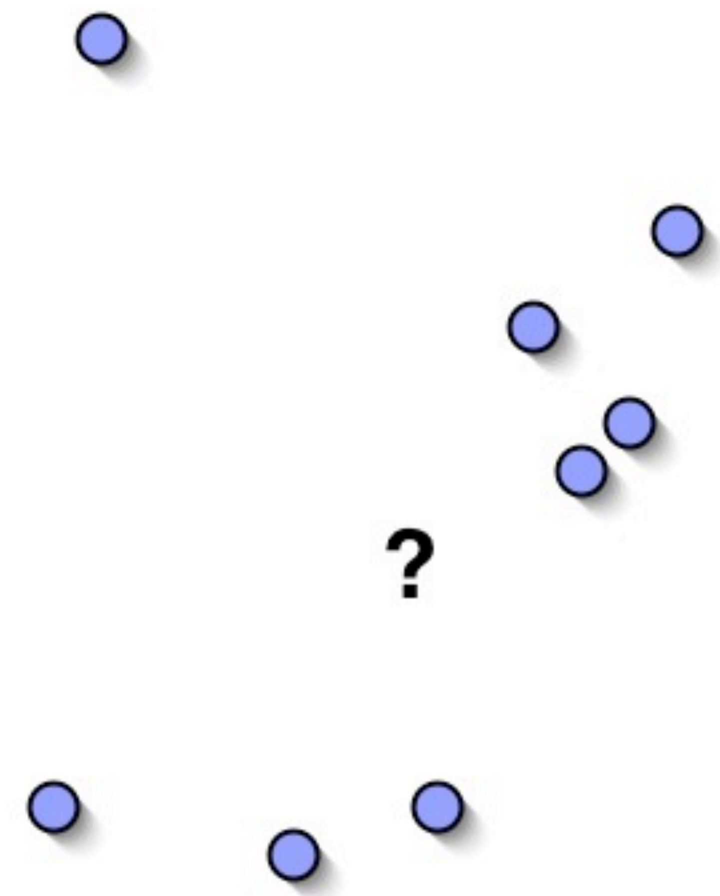
From training data, learn a function $R^N \rightarrow -1, 1$
.... by interpolating the training data

"S3D" (Stereoscopic movies)

- (discussed at 10:30)

definition



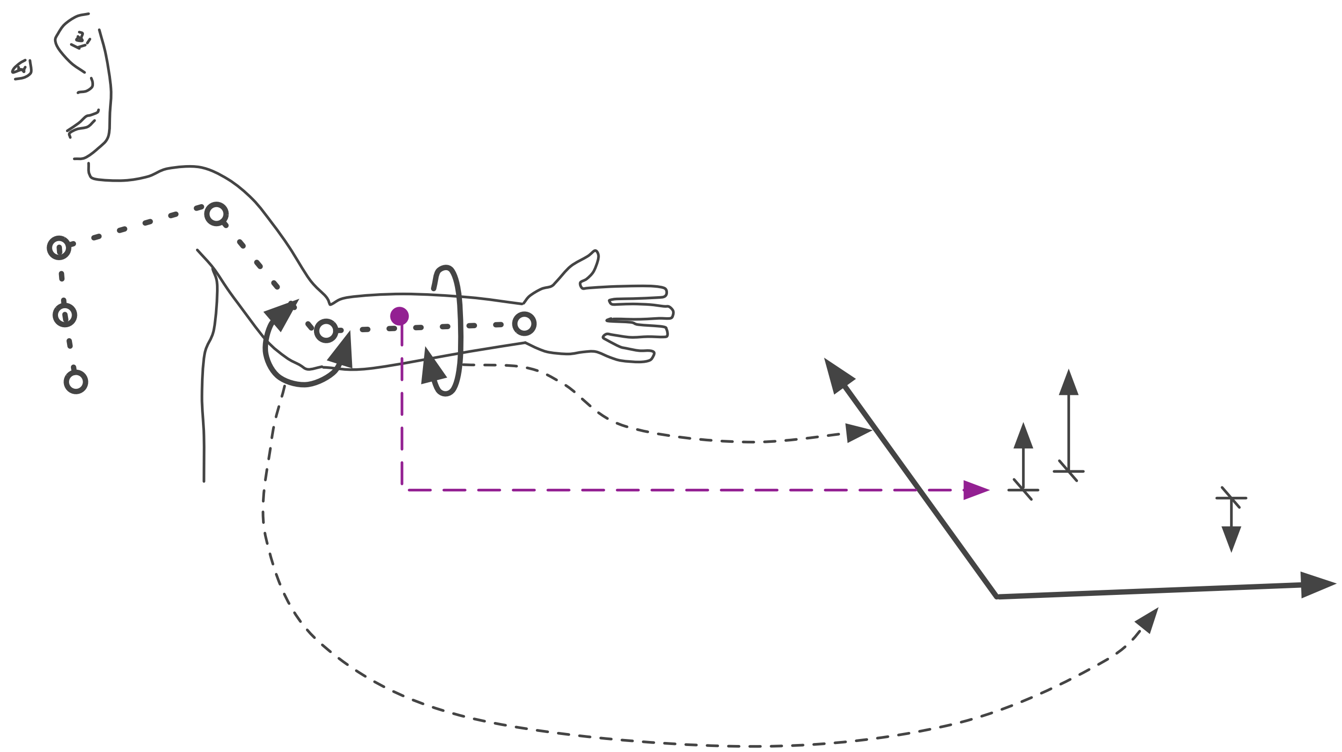


graphics history

- all graphics textbooks discuss splines;
  none cover scattered interpolation (yet)

- < 1999: 3 papers?
  since: lots!

example



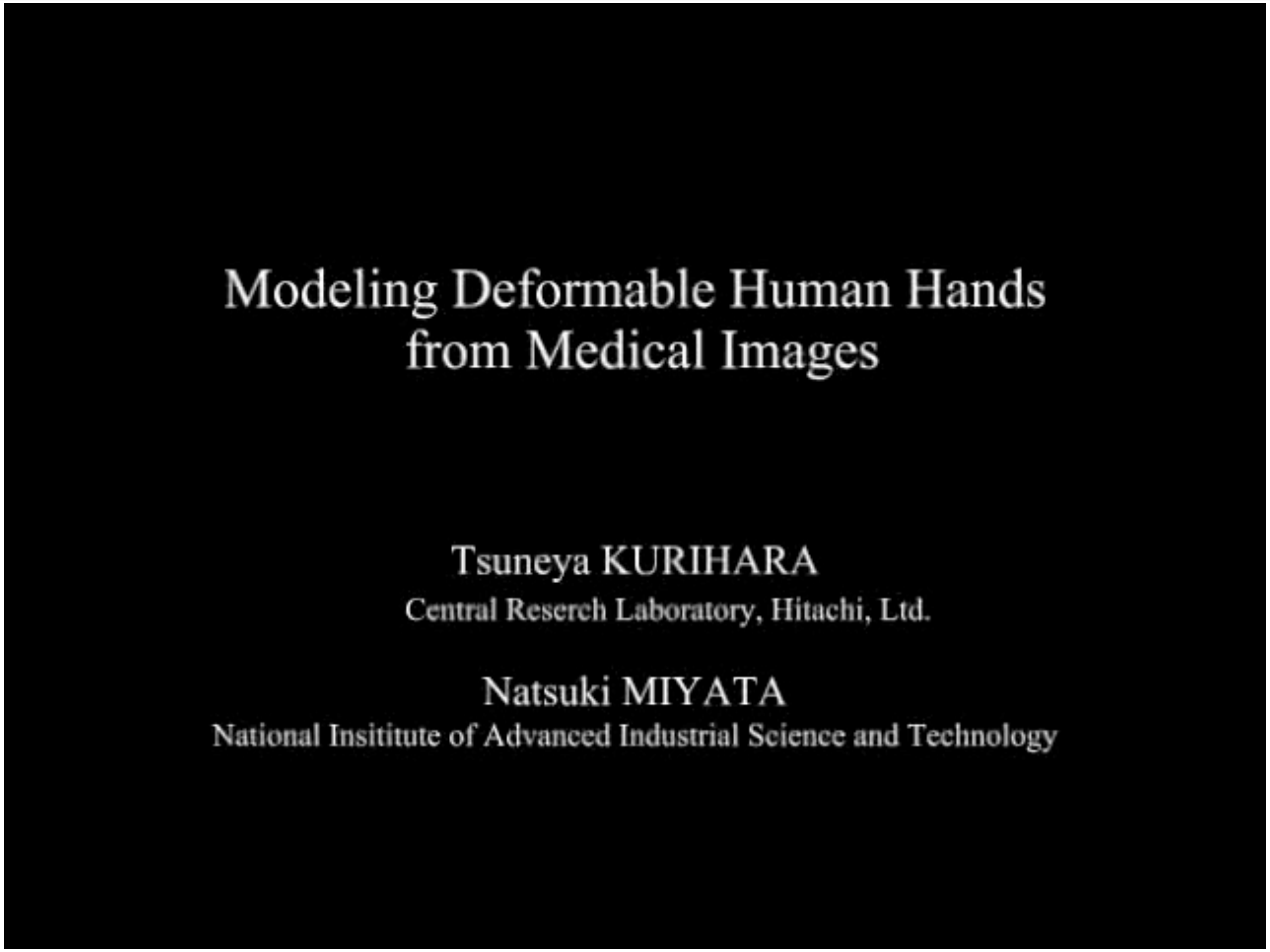
schedule

9:00-9:15	Introduction and survey of applications
9:15-9:30	Non-RBF algorithms
9:30-9:40	break
9:40-10:15	RBF and variants; connection to Laplacian splines
10:15-10:50	**Case studies: Skinning, NPR Shading, Stereoscopic 3D**
10:50-11:00	break
11:00-11:15	Greens functions, RBF and Gaussian process connections
11:15-12:00	Functional analysis, RBF and RKHS connections
12:00-12:15	Open problems, conclusion

Application: weighted PSD skinning

Kurihara & Miyata,
Modeling Deformable Human Hands from Medical Images,
ACM SCA 2004

Application: weighted PSD skinning

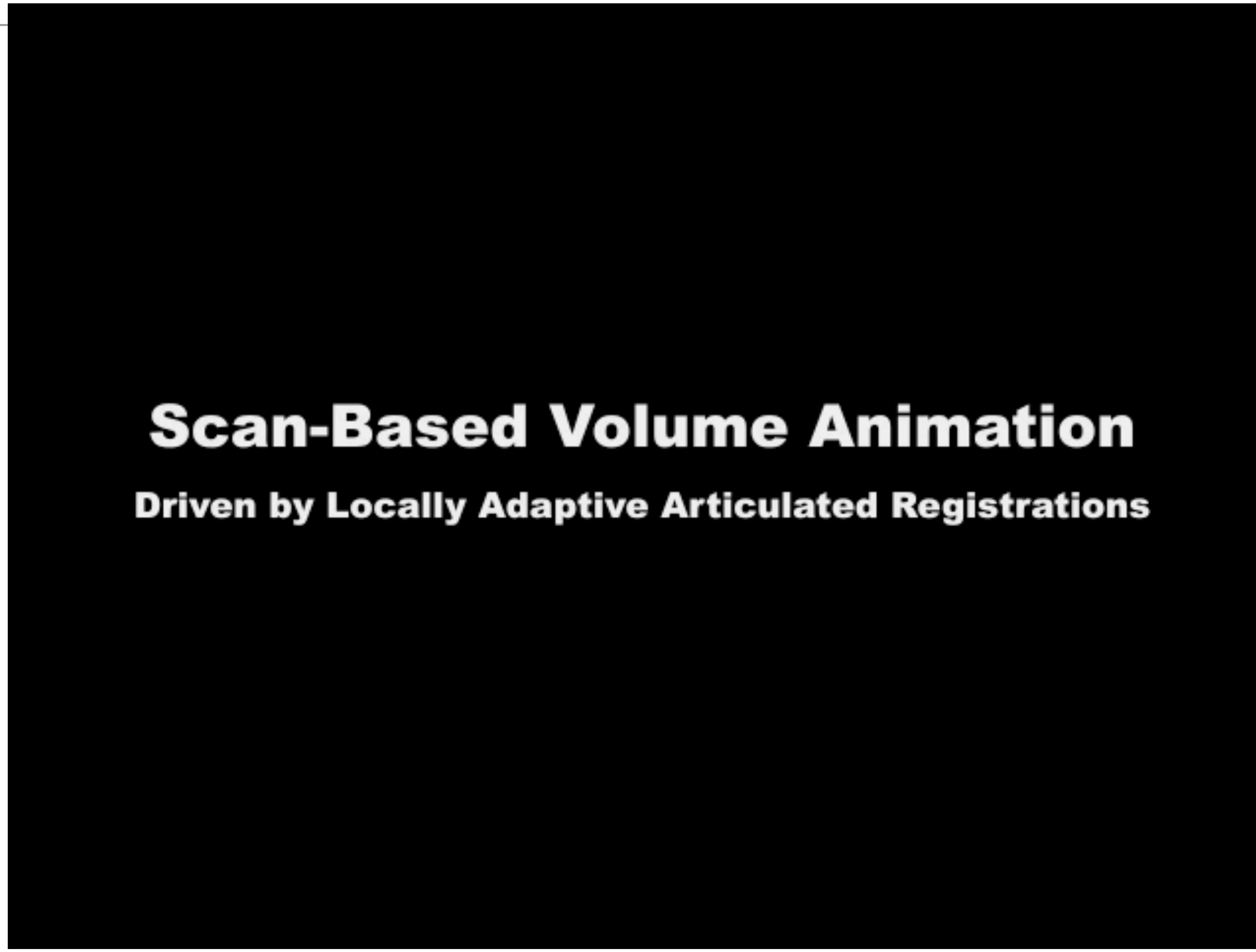


Kurihara & Miyata,
Modeling Deformable Human Hands from Medical Images,
ACM SCA 2004

"volume skinning"

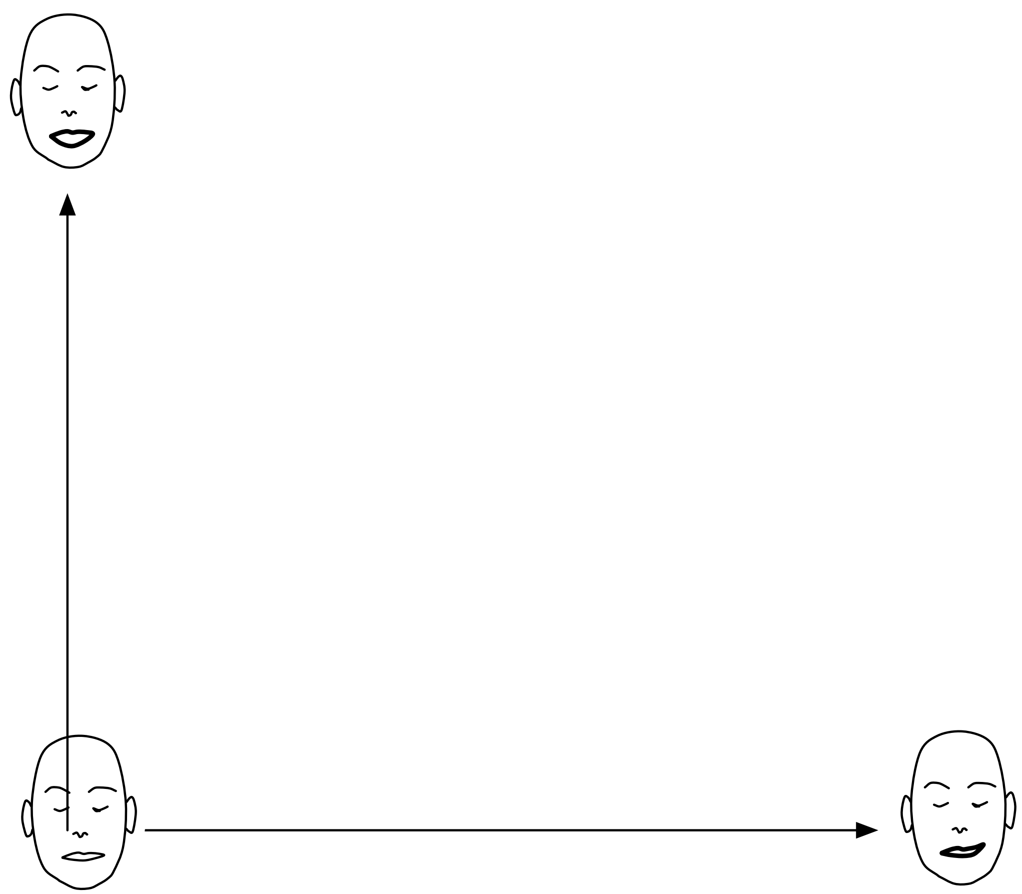
Taehyun Rhee et al.
Scan-Based Volume Animation Driven by Locally Adaptive Articulated Registrations,
IEEE Trans. Visualization and Computer Graphics March 2011

"volume skinning"

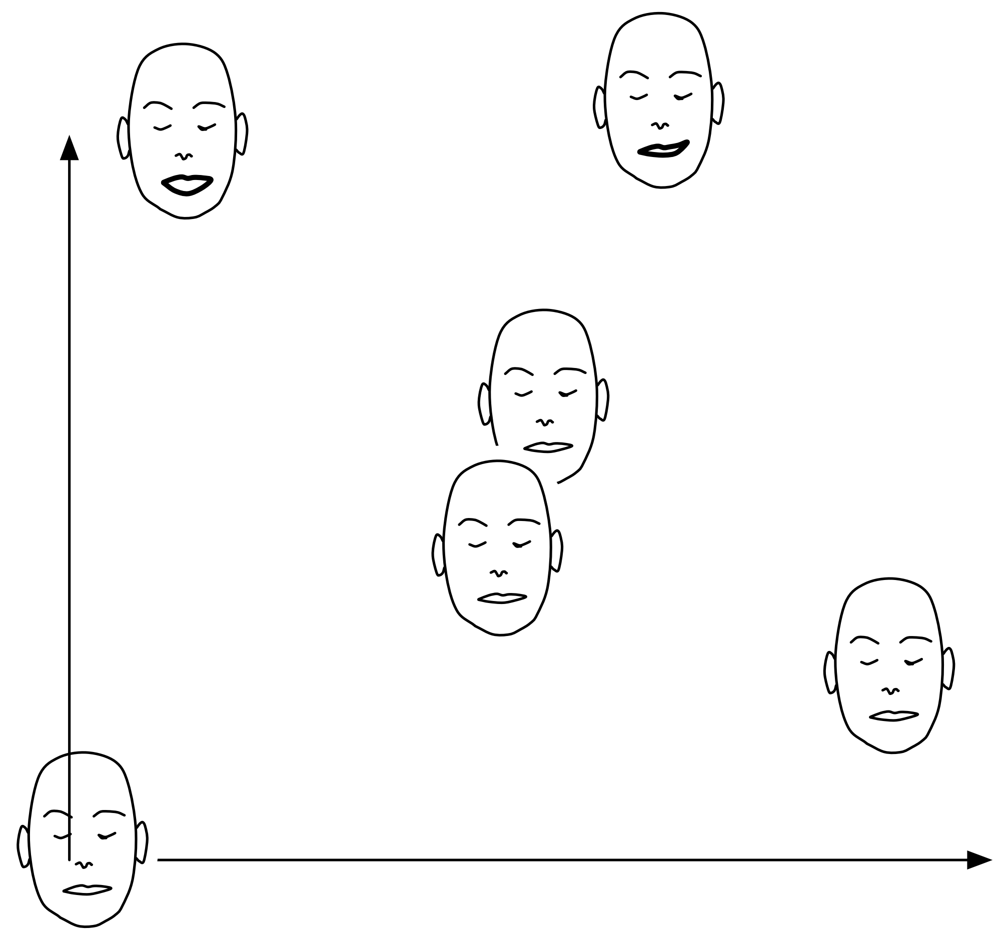


Taehyun Rhee et al.
Scan-Based Volume Animation Driven by Locally Adaptive Articulated Registrations,
IEEE Trans. Visualization and Computer Graphics March 2011

Open Problems



blendshape



desired

- Duchon: Green's function for Laplace ∇^{2m} in various dimensions *n*

$$R(\mathbf{x}) \propto \begin{cases} |\mathbf{x}|^{2m-n} \log |\mathbf{x}| & \text{if } 2m-n \text{ is an even integer,} \\ |\mathbf{x}|^{2m-n} & \text{otherwise,} \end{cases}$$

- for m=2, n=1, this is R(x) = |x|^3
- for m=2, n=3, this is R(x) = |x|^1

- Duchon: Green's function for Laplace ∇^{2m} in various dimensions *n*

$$R(\mathbf{x}) \propto \begin{cases} |\mathbf{x}|^{2m-n} \log |\mathbf{x}| & \text{if } 2m-n \text{ is an even integer,} \\ |\mathbf{x}|^{2m-n} & \text{otherwise,} \end{cases}$$

- for m=2, n=100, this is $R(\mathbf{x}) \propto |\mathbf{x}|^{-96}$
- singular at origin, numerically useless!

Questions or Discussion?

thank you!

- check back for corrections, errata:

  http://scribblethink.org/Courses/ScatteredInterpolation

- contact: zilla@computer.org, anjyo@olm.co.jp

- Acknowledgment: Geoffrey Irving

Scattered Data Interpolation in Computer Graphics

Euclidean invariant

Euclidean invariant
Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

Scattered interpolation is generally Euclidean invariant, versus regular interpolation schemes

Rotate(Interpolate(data)) = Interpolate(Rotate(data))

Shepard Interpolation

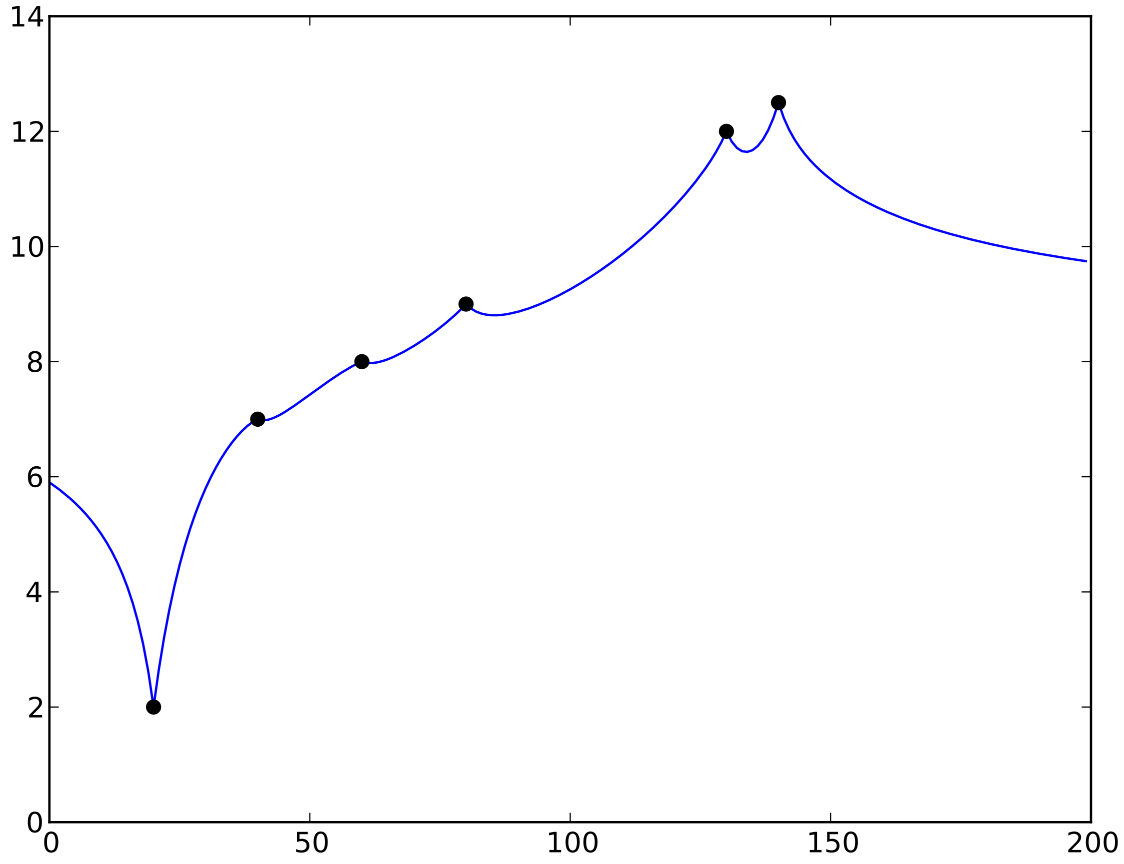
$$\hat{d}(\mathbf{p}) = \frac{\sum w_k(\mathbf{p}) d_k}{\sum w_k(\mathbf{p})}$$

weights set to an inverse power of the distance:
$w_k(\mathbf{p}) = \|\mathbf{p} - \mathbf{p}_k\|^{-p}$.

Note: singular at the data points $\mathbf{p} = \mathbf{p}_k$.

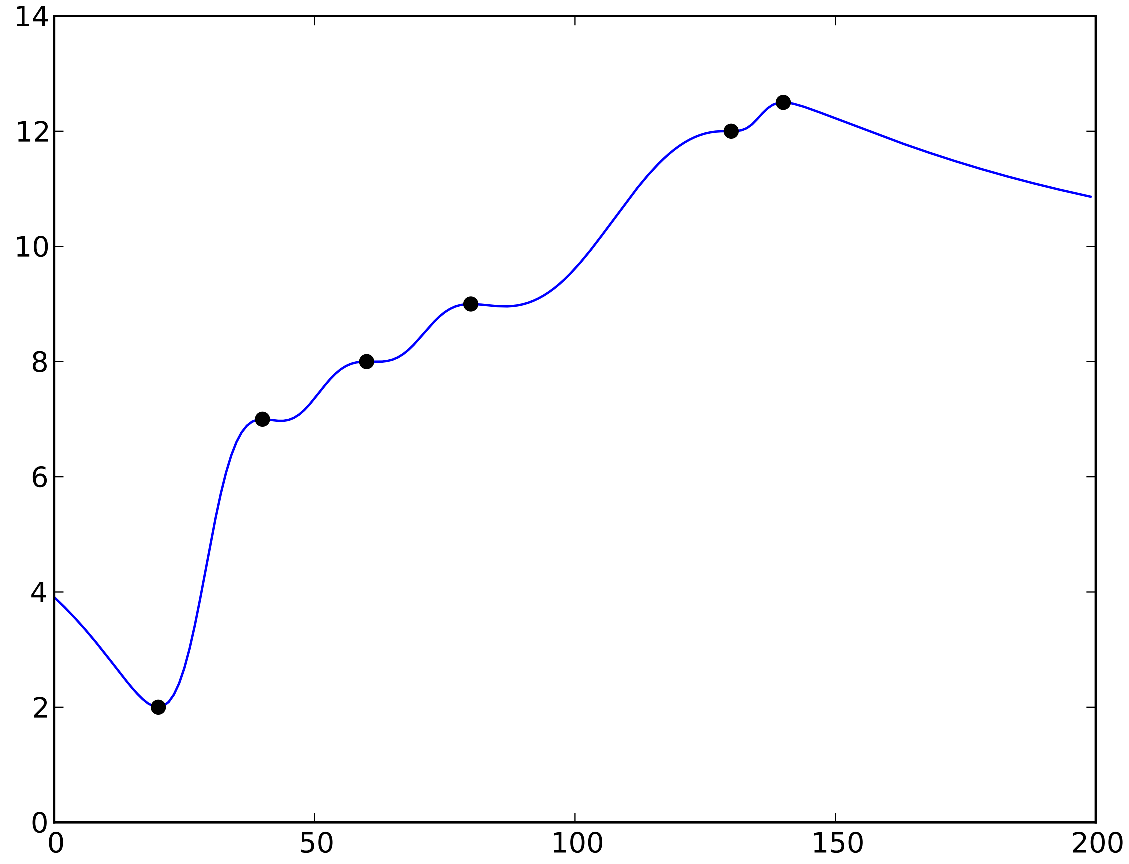
Comparison: Shepard's $p = 1$

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process



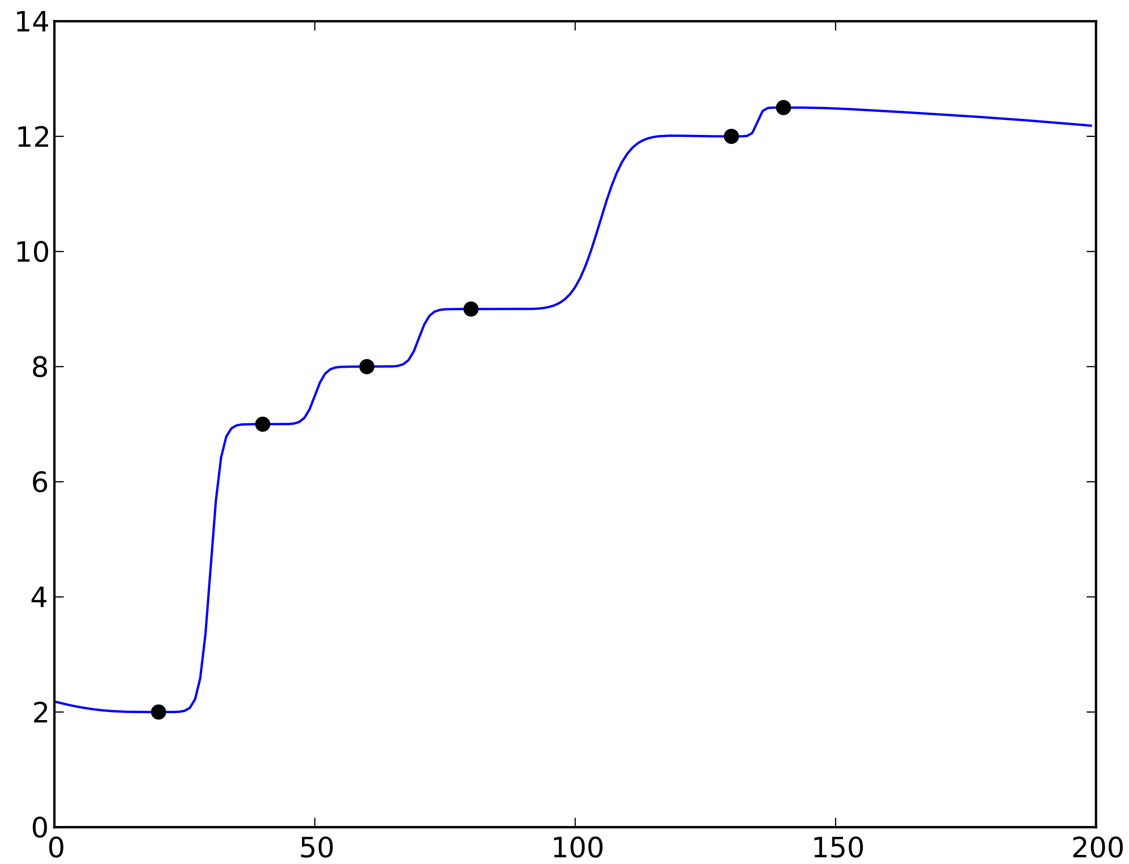
Comparison: Shepard's $p = 2$

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process



Comparison: Shepard's $p = 5$

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process



Kernel smoothing

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

Nadaraya-Watson

$$\hat{d}(\mathbf{p}) = \frac{\sum R(\mathbf{p}, \mathbf{p}_k)\, d_k}{\sum R(\mathbf{p}, \mathbf{p}_k)}$$

Same as Shepard's if $R(\mathbf{p}, \mathbf{p}_k) \equiv \|\mathbf{p} - \mathbf{p}_k\|^{-p}$

Foley and Nielsen

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

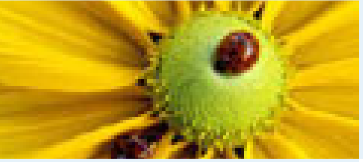
- Use Shepards to interpolate onto a regular grid
- Interpolate the grid with a regular spline
- Interpolate the residual with a second Shepards
- iterate...

T.A.Foley and G.M.Nielson Multivariate interpolation to scattered data using delta iteration. In E.W.Cheny, ed., Approximation Theory II, p.419-424, Academic Press NY 1980.

Moving Least Squares

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

- Fit a polynomial (or other basis) independently at each point
- Use weighted least squares, de-weight data that are far away
- For interpolation, weights must go to infinity at the data points



Moving Least Squares

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

Synthesis: $\hat{d}(x) = \sum_0^m a_i^{(x)} x_k^i$

Solve:

$$\min_{\mathbf{a}} \quad \sum_k^n w_k^{(x)} \left(\sum_0^m a_i^{(x)} \, x_k^i - d_k\right)^2$$

m - degree of polynomial

$$w_k^{(x)} = \frac{1}{\|x - x_k\|^p}$$

Moving Least Squares

$$\min_{\mathbf{a}} \quad \sum_{k}^{n} w_k^{(x)} (d_k - \sum_{0}^{m} a_i^{(x)} \, x_k^i)^2$$

call $x_k^i \equiv \mathbf{b}_k \in R^{m+1}$, the polynomial basis evaluated at the kth point

$$= \min_{\mathbf{a}} \quad \sum_{k}^{n} w_k^{(x)} (d_k - \mathbf{b}^T \mathbf{a})^2$$

Matrix version:

$$\min_{\mathbf{a}} \quad \|\mathbf{W}(\mathbf{B}\mathbf{a} - \mathbf{d})\|^2$$

$\mathbf{W}$ is diagonal matrix with sqrt of $w_k^{(x)}$.

MLS = Shepard's when $m = 0$

Euclidean invariant
Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

$$\min_a \quad \sum_k^n w_k^{(x)} (a \cdot 1 - d_k)^2$$

$$\frac{d}{da}\left[\sum_k^n w_k^{(x)} (a^2 - 2ad_k + d_k^2)\right] = 0$$

$$\frac{d}{da}\left[\sum_k^n w_k a^2 - 2w_k a d_k + w_k d_k^2\right]$$

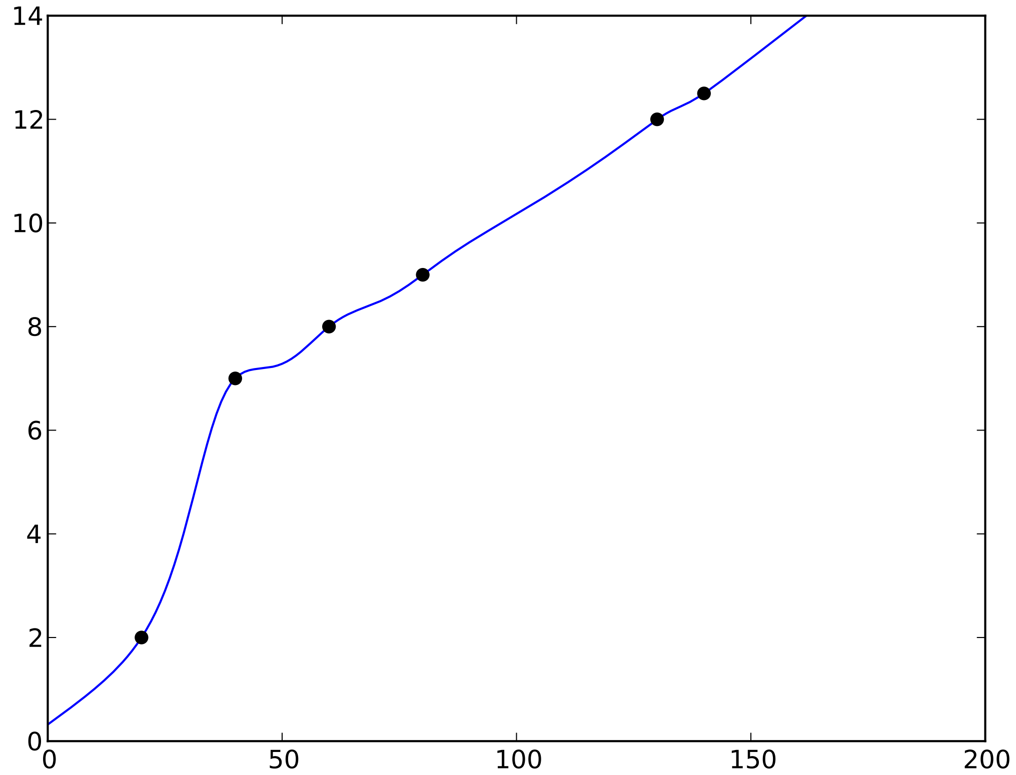
$$= \sum_k^n 2w_k a - 2w_k d_k = 0$$

$$a = \frac{\sum_k^n w_k d_k}{\sum_k^n w_k}$$

$$\hat{d}(x) = a \cdot 1$$

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

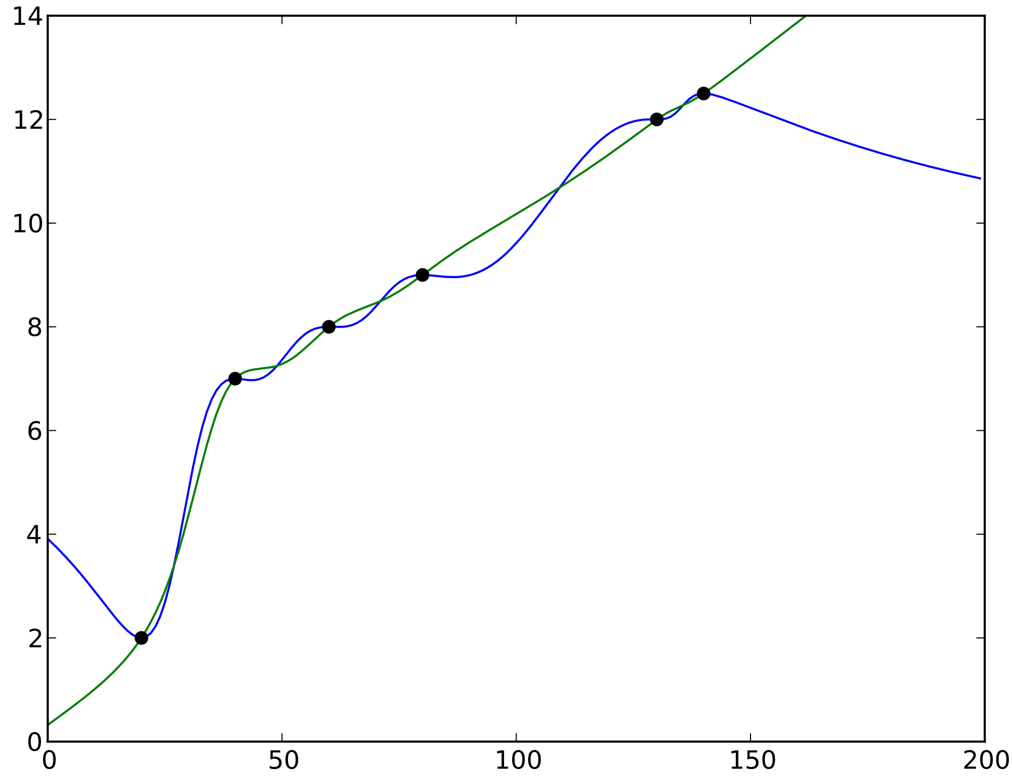
Moving Least Squares



$m = 1$, i.e. local linear regression

Moving Least Squares

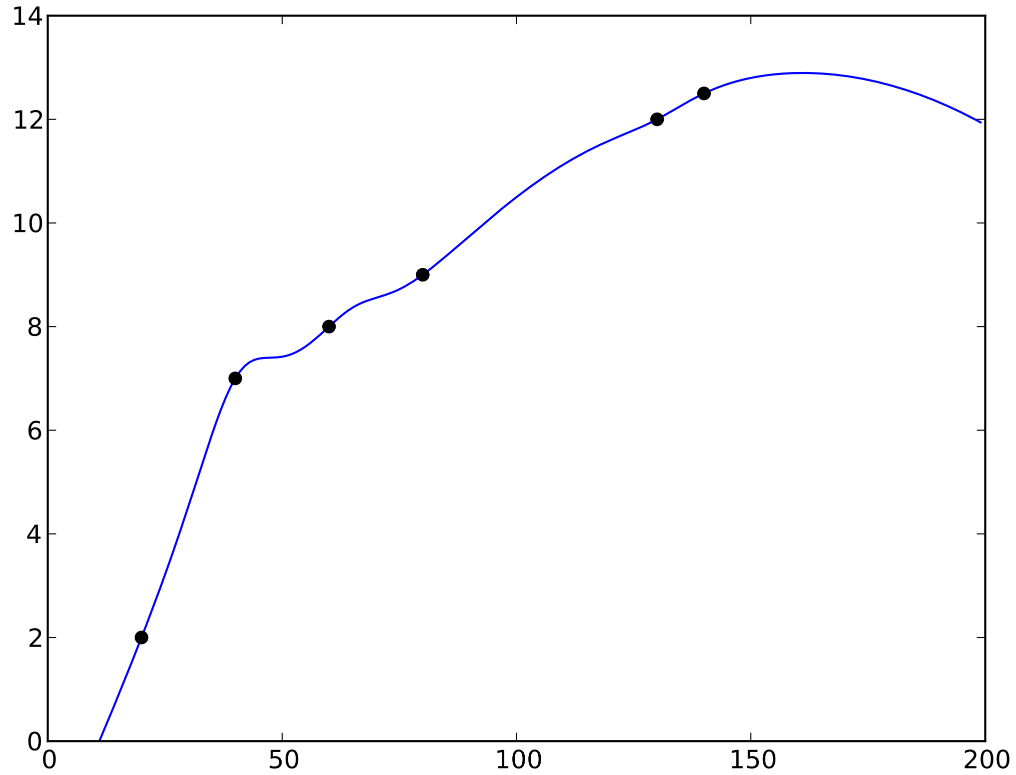
Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process



$m = 0, 1,$ comparison

Moving Least Squares

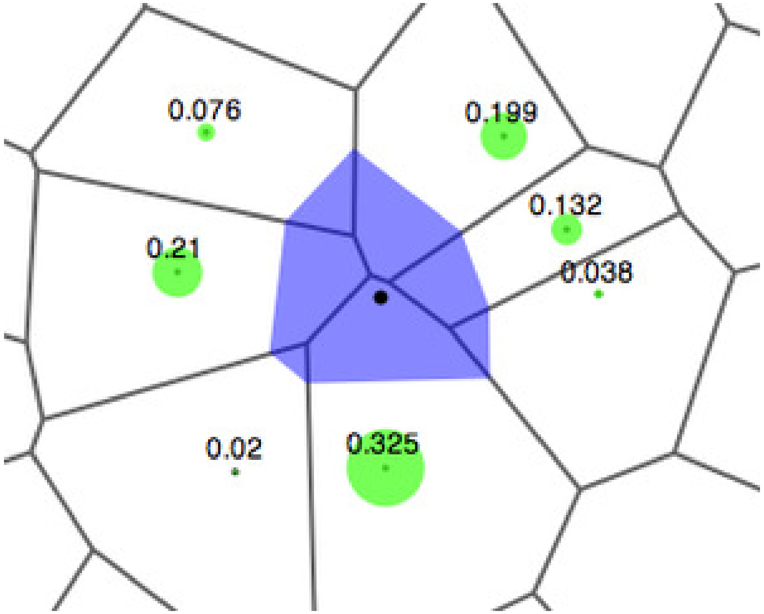
Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process



$m = 2$, i.e. local quadratic regression

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

Natural Neighbor Interpolation



wikipedia

Natural Neighbor Interpolation

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

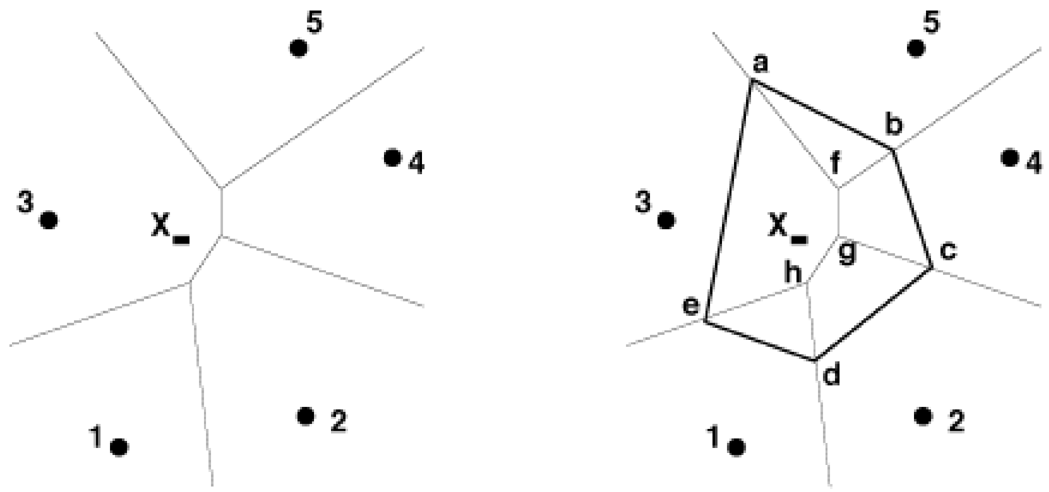


Image: N. Sukmar, Natural Neighbor Interpolation and the Natural Element Method (NEM)

Notation

$$
\begin{aligned}
R(x,y) &\quad \text{symmetric pos. def.} \\
R(x,y) = \phi(\|x-y\|) & \\
\mathbf{R} &\quad \text{matrix version} \\
R_{xy} &\quad \text{element of matrix}
\end{aligned}
$$

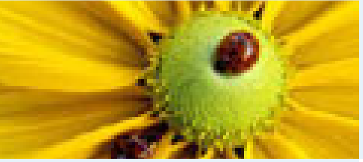
R is kernel or covariance

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

Gaussian Process regression

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

from
Generalized Stochastic Subdivision,
ACM TOG July 1987



Gaussian Process regression

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process ...

linear estimator $\hat{d}_t = \sum w_k d_{t+k}$

orthogonality $E[(d_t - \hat{d}_t)d_m] = 0$

$$E[d_t d_m] = E[\sum w_k d_{t+k}\, d_m]$$

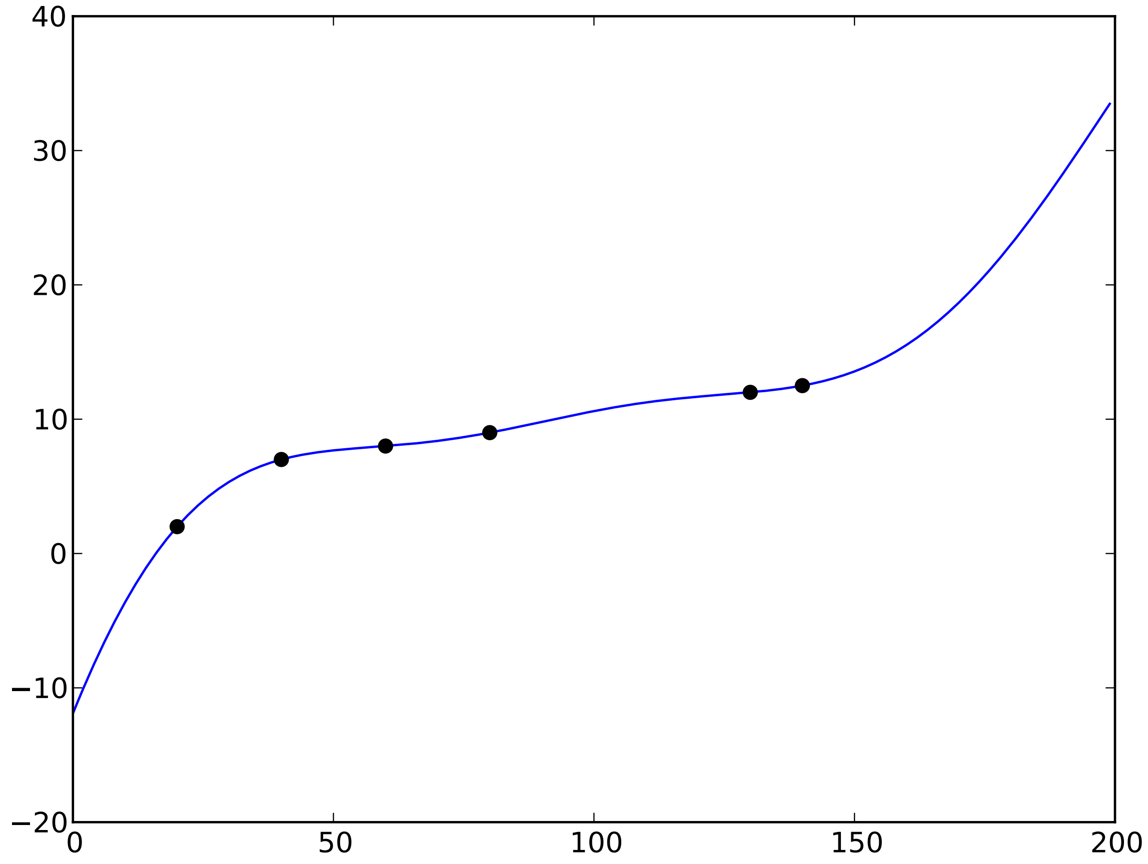
autocovariance $E[d_t d_m] = R(t-m)$

linear system $R(t-m) = \sum w_k R(t+k-m)$

Note no requirement on the actual spacing of the data. Related to the "Kriging" method in geology.

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

Comparison: GaussianProcess



Laplace/Poisson Interpolation

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

i.e. "Laplacian Splines" Objective: Minimize a roughness measure, the integrated derivative (or gradient) squared:

$$\min_f \int \left(\frac{df(x)}{dx} \right)^2 dx$$

or

$$\min_f \int \int \|\nabla f\|^2 ds$$

(subject to some constraints, to avoid a trivial solution)

function, operator

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

- function: $f(x) \to y$
- operator: $Mf \to g$, e.g. Matrix-vector multiplication

“Null space of the operator”

Euclidean invariant Shepard Interpolation
Comparison: Shepard’s $p = 1$
Comparison: Shepard’s $p = 2$
Comparison: Shepard’s $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard’s when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

$$\min_f \int \left(\frac{df(x)}{dx} \right)^2 dx$$

Gives zero for $f(x) =$ any constant.

Laplace/Poisson: solution approaches

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

- direct matrix inverse
- Jacobi (because matrix is quite sparse)
- Jacobi variants (SOR)
- Multigrid

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

Laplace/Poisson: Discrete

Local viewpoint:

roughness $$R = \int |\nabla u|^2 du \approx \sum (u_{k+1} - u_k)^2$$

for a particular k: $$\frac{dR}{du_k} = \frac{d}{du_k}[(u_k - u_{k-1})^2 + (u_{k+1} - u_k)^2]$$

$$= 2(u_k - u_{k-1}) - 2(u_{k+1} - u_k) = 0$$

$$u_{k+1} - 2u_k + u_{k-1} = 0 \to \nabla^2 u = 0$$

Note 1,-2,1 pattern.

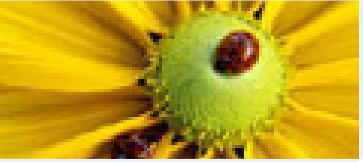
Laplace/Poisson Interpolation

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

Discrete/matrix viewpoint: Encode derivative operator in a matrix $\mathbf{D}$

$$\mathbf{Df} = \begin{bmatrix} -1 & 1 & \\ & -1 & 1 \\ & & \ldots \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \end{bmatrix}$$

$$\min_f \int \left(\frac{df}{dx}\right)^2 \approx \min_{\mathbf{f}} \|\mathbf{Df}\|^2 = \min_{\mathbf{f}} \quad \mathbf{f}^T\mathbf{D}^T\mathbf{Df}$$



Laplace/Poisson Interpolation

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

$$\min_{\mathbf{f}} \mathbf{f}^T \mathbf{D}^T \mathbf{D} \mathbf{f}$$

$$\frac{d}{d\mathbf{f}} \left(\mathbf{f}^T \mathbf{D}^T \mathbf{D} \mathbf{f} \right) = 2\mathbf{D}^T \mathbf{D} \mathbf{f} = 0$$

i.e.

$$\frac{d^2 f}{dx^2} = 0 \quad \text{or} \quad \nabla^2 = 0$$

$f = 0$ is a solution; last eigenvalue is zero, corresponds to a constant solution.

Discrete Laplacian

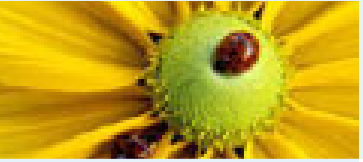
Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

Notice

$$\mathbf{D}^T\mathbf{D} = \begin{bmatrix} 1 & -2 & 1 & \\ & 1 & -2 & 1 \\ & & \dots & \end{bmatrix}$$

Two-dimensional stencil

$$\mathbf{D}^T\mathbf{D} = \begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix}$$



Jacobi iteration

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

Local viewpoint

Jacobi iteration sets each f_k to the solution of its row of the matrix equation, independent of all other rows:

$$\sum A_{rc} f_c = b_r$$

$$\rightarrow \qquad A_{rk} f_k = b_k - \sum_{j \neq k} A_{rj} f_j$$

$$f_k \leftarrow \frac{b_k}{A_{kk}} - \sum_{j \neq k} A_{kj}/A_{kk} f_j$$

Jacobi iteration

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

apply to Laplace eqn
Jacobi iteration sets each f_k to the solution of its row of the matrix equation, independent of all other rows:

$$\ldots f_{t-1} - 2f_t + f_{t+1} = 0$$
$$2f_t = f_{t-1} + f_{t+1}$$
$$f_k \leftarrow 0.5 * (f[k-1] + f[k+1])$$

In 2D,

```
 f[y][x] = 0.25 * ( f[y+1][x] + f[y-1][x] +
f[y][x-1] + f[y][x+1] )
```

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

But now let's interpolate

1D case, say f_3 is known. Three eqns involve f_3. Subtract (a multiple of) f_3 from both sides of these equations:

$$
\begin{aligned}
f_1 - 2f_2 + f_3 = 0 \quad &\rightarrow \quad f_1 - 2f_2 + 0 = -f_3 \\
f_2 - 2f_3 + f_4 = 0 \quad &\rightarrow \quad f_2 + 0 + f_4 = 2f_3 \\
f_3 - 2f_4 + f_5 = 0 \quad &\rightarrow \quad 0 - 2f_4 + f_5 = -f_3
\end{aligned}
$$

$$
L = \begin{bmatrix} 1 & -2 & 0 & \\ & 1 & 0 & 1 \\ & & 0 & -2 \\ & & \cdots & \end{bmatrix} \text{one column is zeroed}
$$

Multigrid inpainting

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

Program demonstration.

Remove dog's spots. Combine Wiener filtering to separate fur from luminance, with Laplace interpolation to adjust the luminance.

Applications: Spot Removal

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

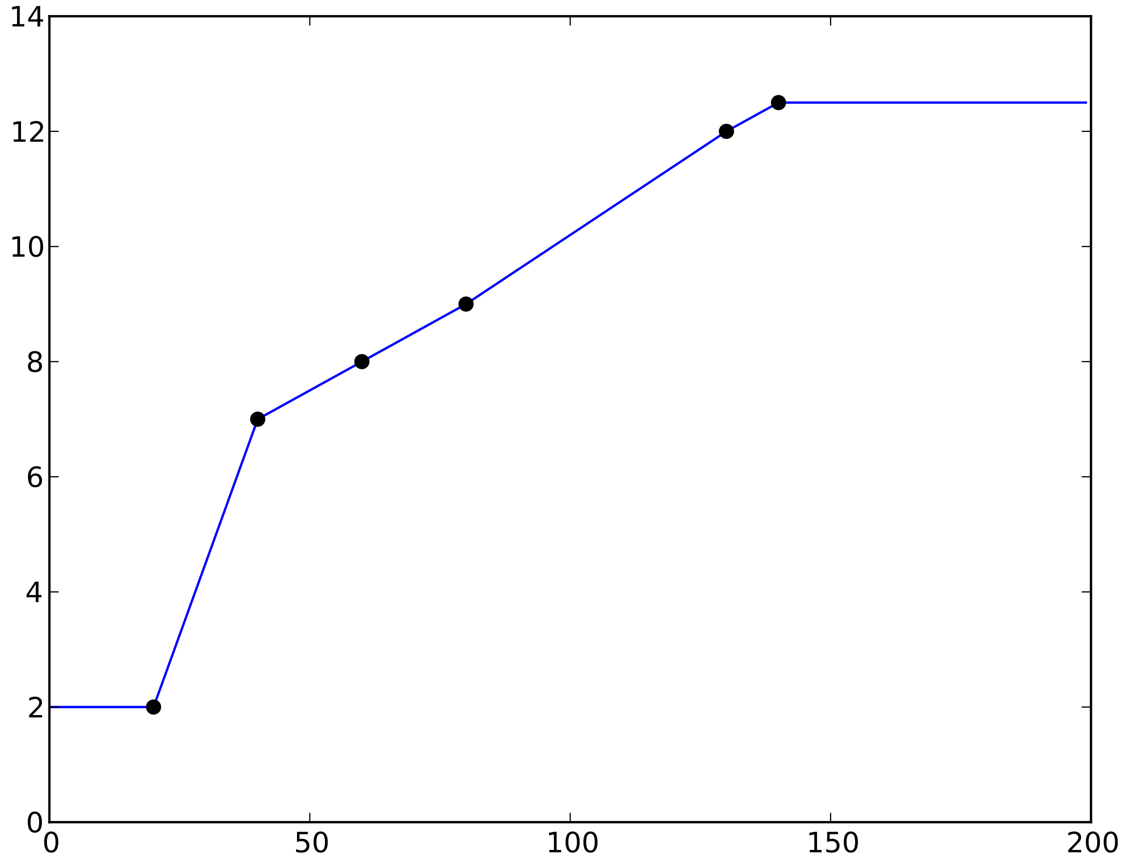
From: Lifting Detail from Darkness, *SIGGRAPH 2001*

Recovered fur: detail

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

Comparison: Laplace

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process



Thin plate spline

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

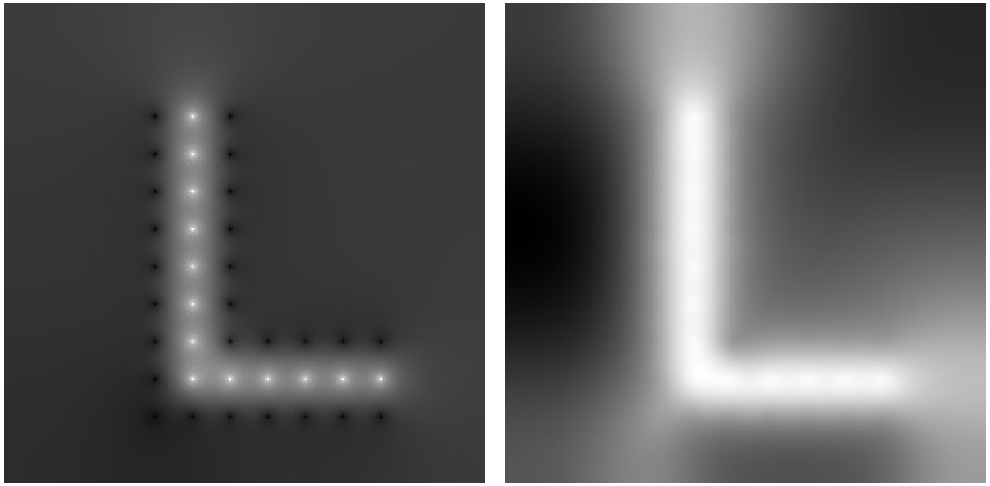
Minimize the integrated second derivative squared (approximate curvature)

$$\min_f \int \left(\frac{d^2 f}{dx^2} \right)^2 dx$$

Null space: $f = ax + c$

Membrane vs. Thin Plate

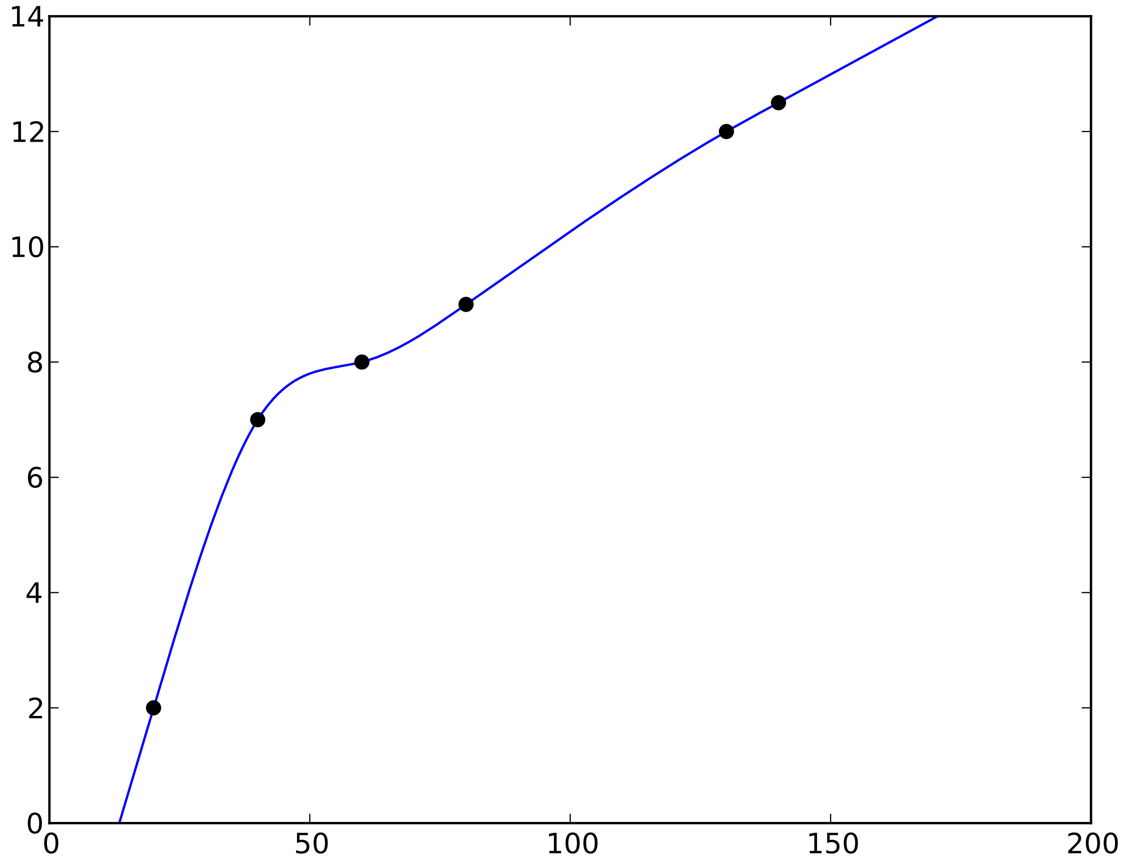
Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process



Left - membrane interpolation, right - thin plate.

Comparison: Cubic

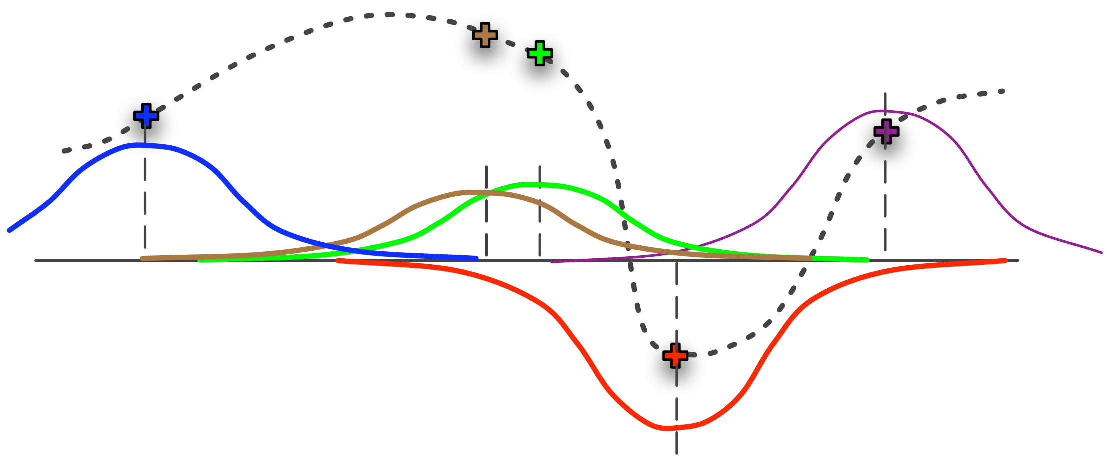
Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process



Radial Basis Functions

Euclidean invariant
Shepard
Interpolation
Comparison:
Shepard's $p = 1$
Comparison:
Shepard's $p = 2$
Comparison:
Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least
Squares
Moving Least
Squares
Moving Least
Squares
MLS = Shepard's
when $m = 0$
Moving Least
Squares
Moving Least
Squares
Moving Least
Squares
Natural Neighbor
Interpolation
Natural Neighbor
Interpolation
Notation
Gaussian Process
regression
Gaussian Process

$$\hat{d}(\mathbf{p}) = \sum_k^N w_k R(\|\mathbf{p} - \mathbf{p}_k\|)$$



Data at arbitrary (irregularly spaced) locations can be interpolated with a weighted sum of radial functions situated at each data point.

Radial Basis Functions: History

Euclidean invariant
Shepard
Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

- Broomhead & Lowe, 1988
- Werntges, ICNN 1993
- in Graphics: 1999-2001

Radial Basis Functions: Theory

Euclidean invariant
Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

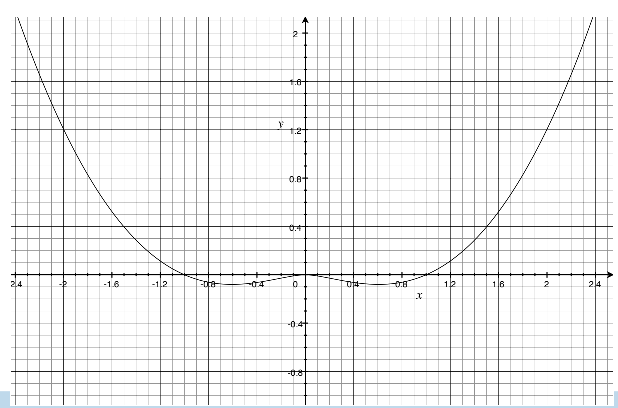
- Micchelli - for a large class of functions, the RBF matrix is non-singular

Radial Basis Functions (RBFs)

Euclidean invariant
Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

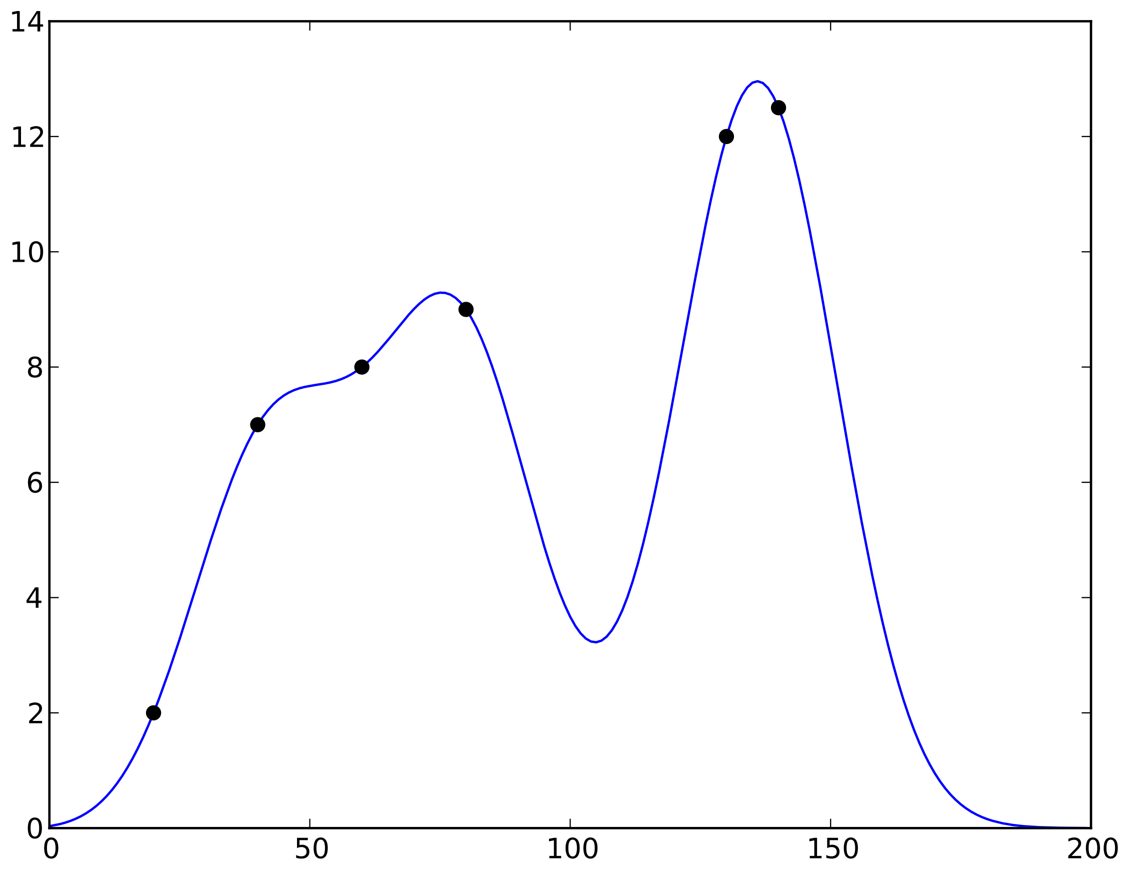
- *any* monotonic function can be used?!
- common choices:
  - Gaussian $R(r) = \exp(-r^2/\sigma^2)$
  - Thin plate spline $R(r) = r^2 \log r$
  - Hardy multiquadratic $R(r) = \sqrt{(r^2 + c^2)}, c > 0$

Notice: the last two *increase* as a function of radius

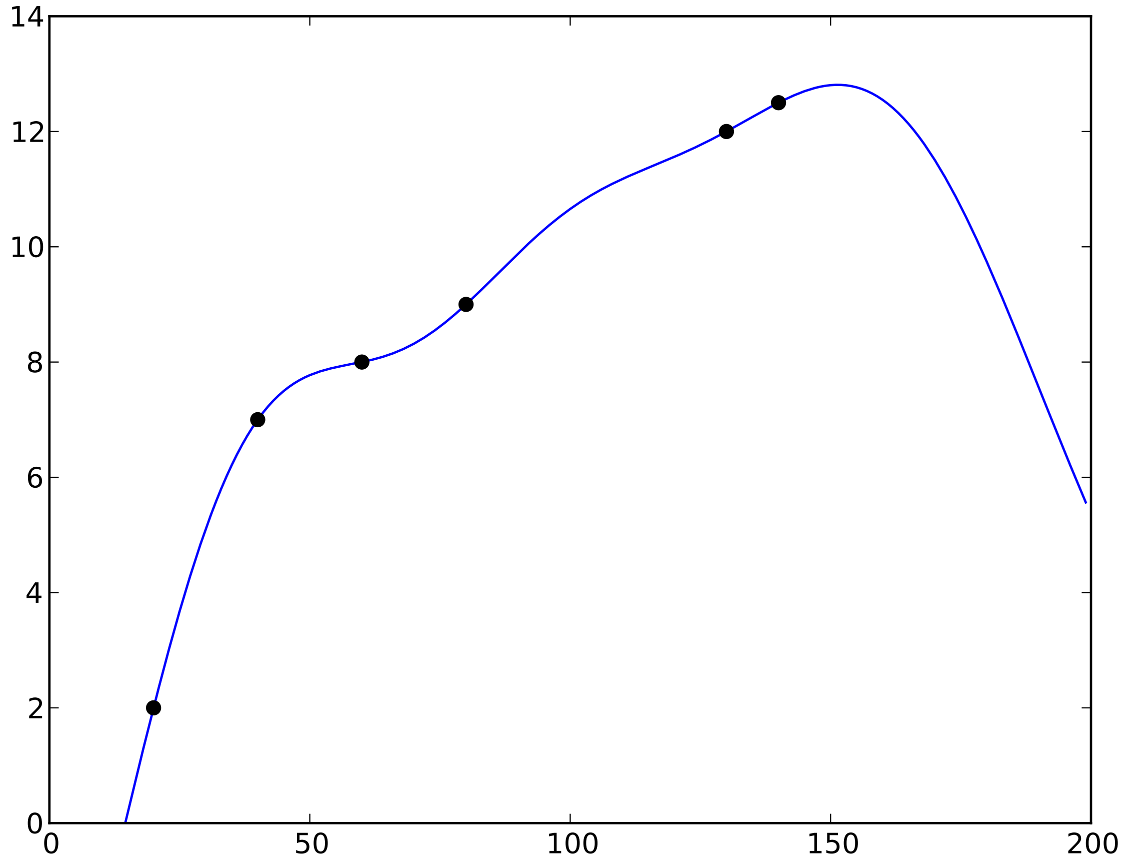


Comparison: RBF-Gauss

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

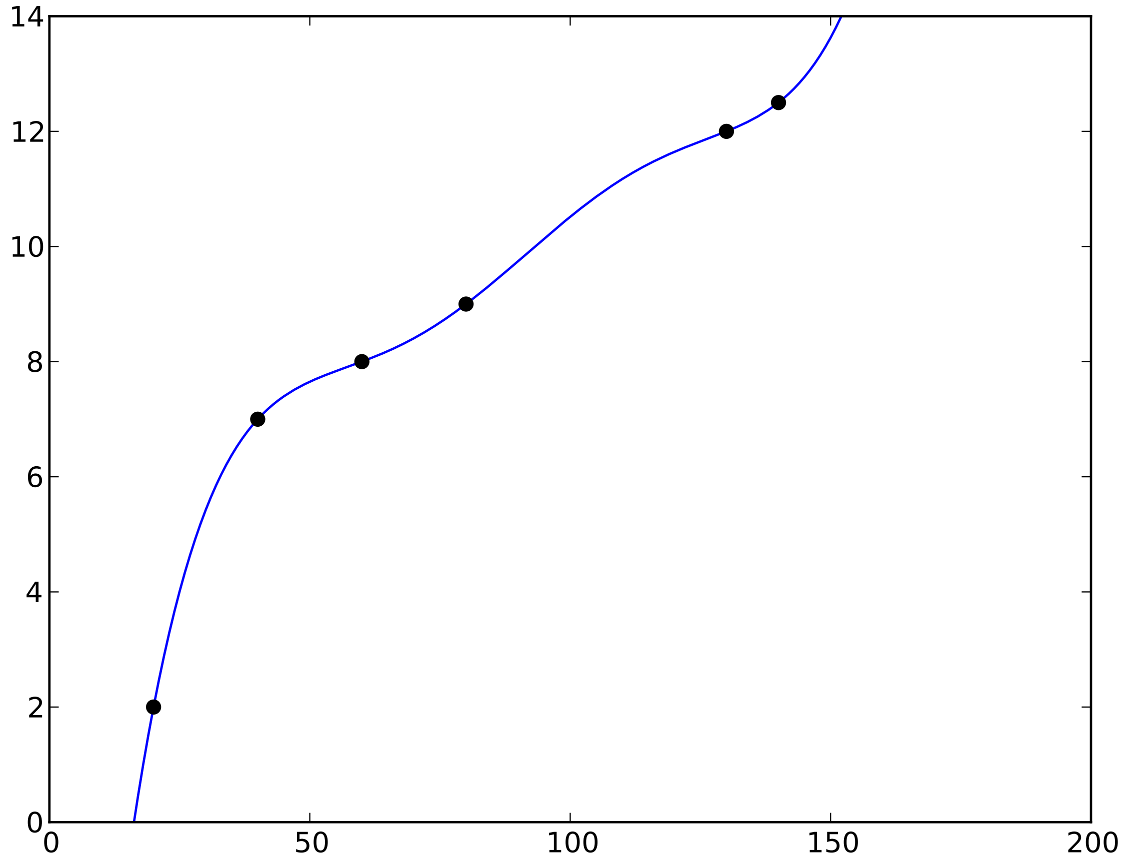


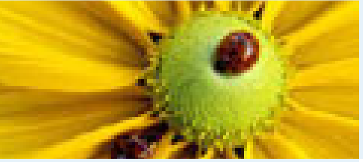
Comparison: RBF-Gauss



Comparison: RBF-Gauss

Euclidean invariant
Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process





Radial Basis Functions

Euclidean invariant
Shepard
Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

$$\hat{d}(\mathbf{p}) = \sum_k^N w_k R(\|\mathbf{p} - \mathbf{p}_k\|)$$

$$e = ||(\mathbf{d} - \mathbf{R}\mathbf{w})||^2$$

$$e = (\mathbf{d} - \mathbf{R}\mathbf{w})^T(\mathbf{d} - \mathbf{R}\mathbf{w})$$

$$\frac{de}{d\mathbf{w}} = 0 = -\mathbf{R}^T(\mathbf{d} - \mathbf{R}\mathbf{w})$$

$$\mathbf{w} = \mathbf{R}^{-1}\mathbf{d}$$

Radial Basis Functions

Euclidean invariant
Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

$$\begin{bmatrix} R_{1,1} & R_{1,2} & R_{1,3} & \cdots \\ R_{2,1} & R_{2,2} & \cdots & \\ R_{3,1} & \cdots & & \\ \vdots & & & \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \end{bmatrix}$$

RBF: multidimensional interpolation

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

$$\mathbf{w}_x = \mathbf{R}^{-1}\mathbf{d}_\mathbf{x}$$

$$\mathbf{w}_y = \mathbf{R}^{-1}\mathbf{d}_\mathbf{y}$$

$$\mathbf{w}_z = \mathbf{R}^{-1}\mathbf{d}_\mathbf{z}$$

Matrix $\mathbf{R}$ is in common to all dimensions

Normalized Radial Basis Function

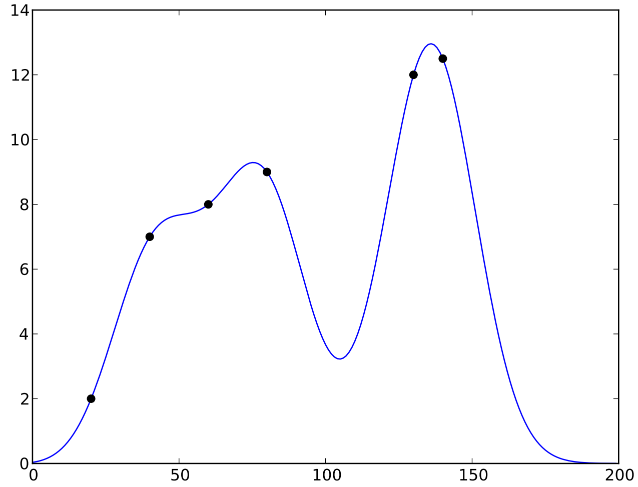
Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

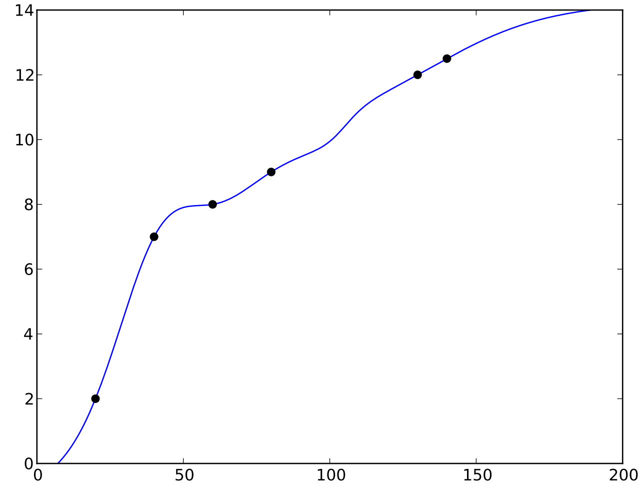
$$\mathbf{R}_i() \quad \Leftarrow \quad \frac{\mathbf{R}(\|x - x_i\|)}{\sum_j \mathbf{R}(\|x - x_j\|)}$$

- removes the "dips" that result from too-narrow σ
- i.e. somewhat less sensitive to choice of σ
- (for decaying kernel), far from the data, closest point dominates

Normalized Radial Basis Function

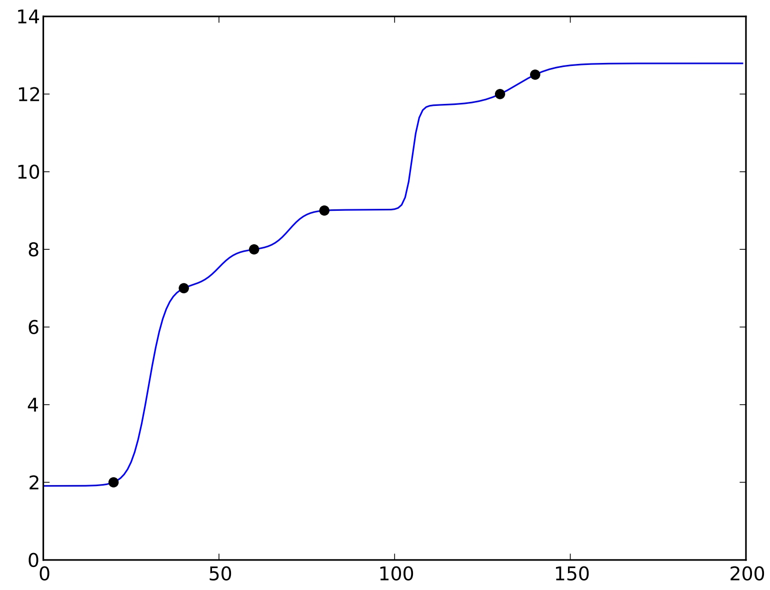
Euclidean invariant
Shepard
Interpolation
Comparison:
Shepard's $p = 1$
Comparison:
Shepard's $p = 2$
Comparison:
Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least
Squares
Moving Least
Squares
Moving Least
Squares
MLS = Shepard's
when $m = 0$
Moving Least
Squares
Moving Least
Squares
Moving Least
Squares
Natural Neighbor
Interpolation
Natural Neighbor
Interpolation
Notation
Gaussian Process
regression
Gaussian Process





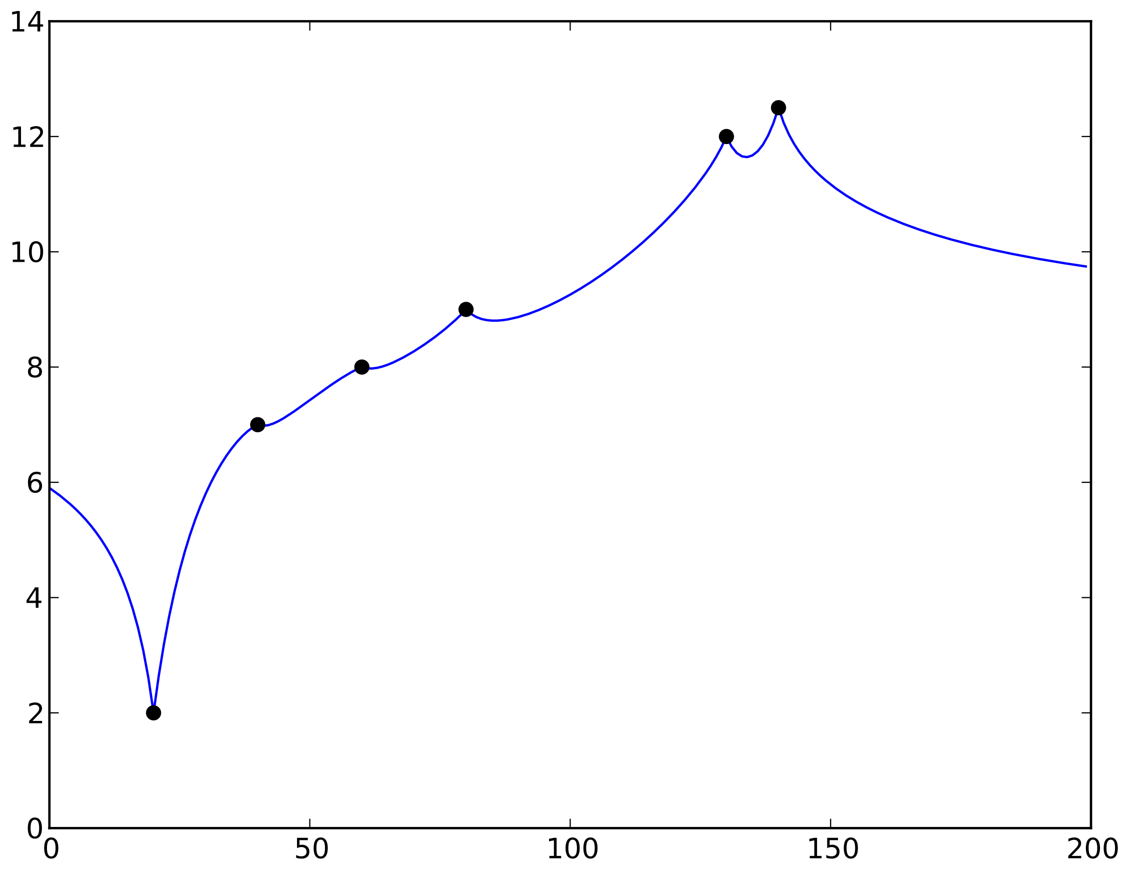
insensitive to sigma, up to a point...

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process



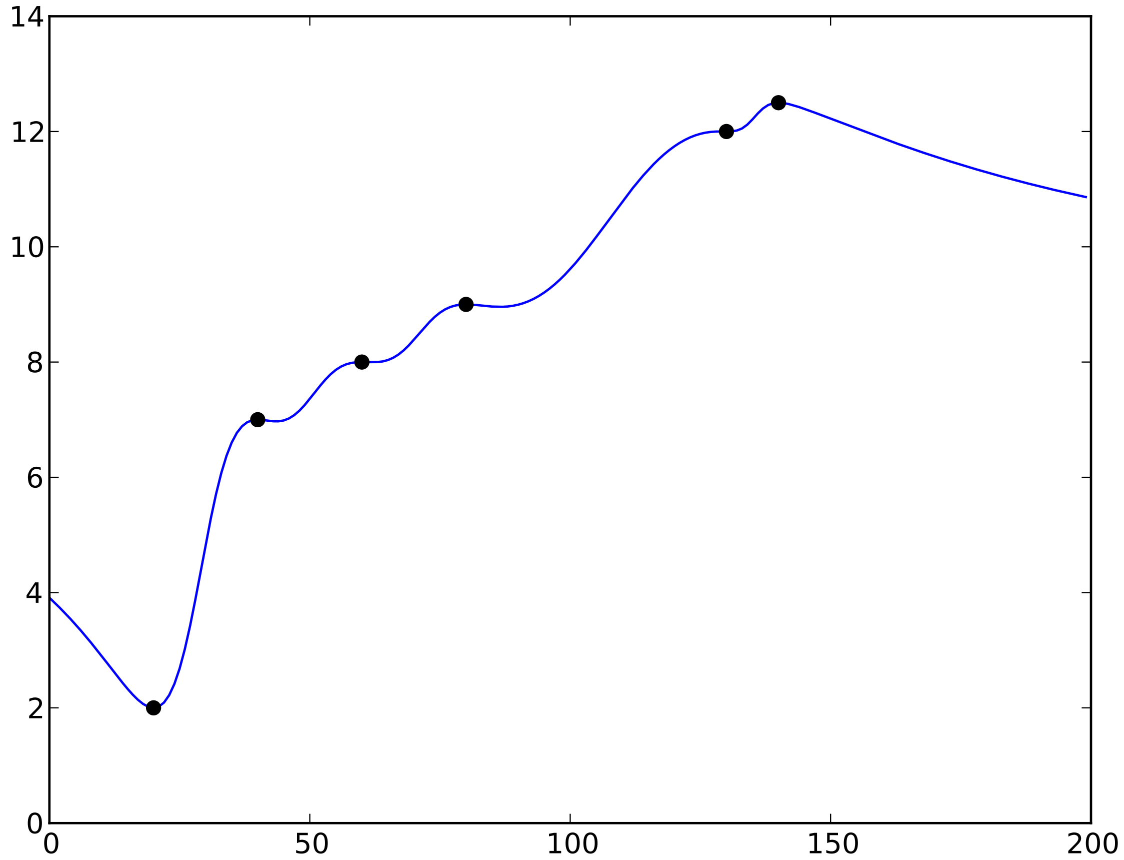
Comparison: Shepard's $p = 1$

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process



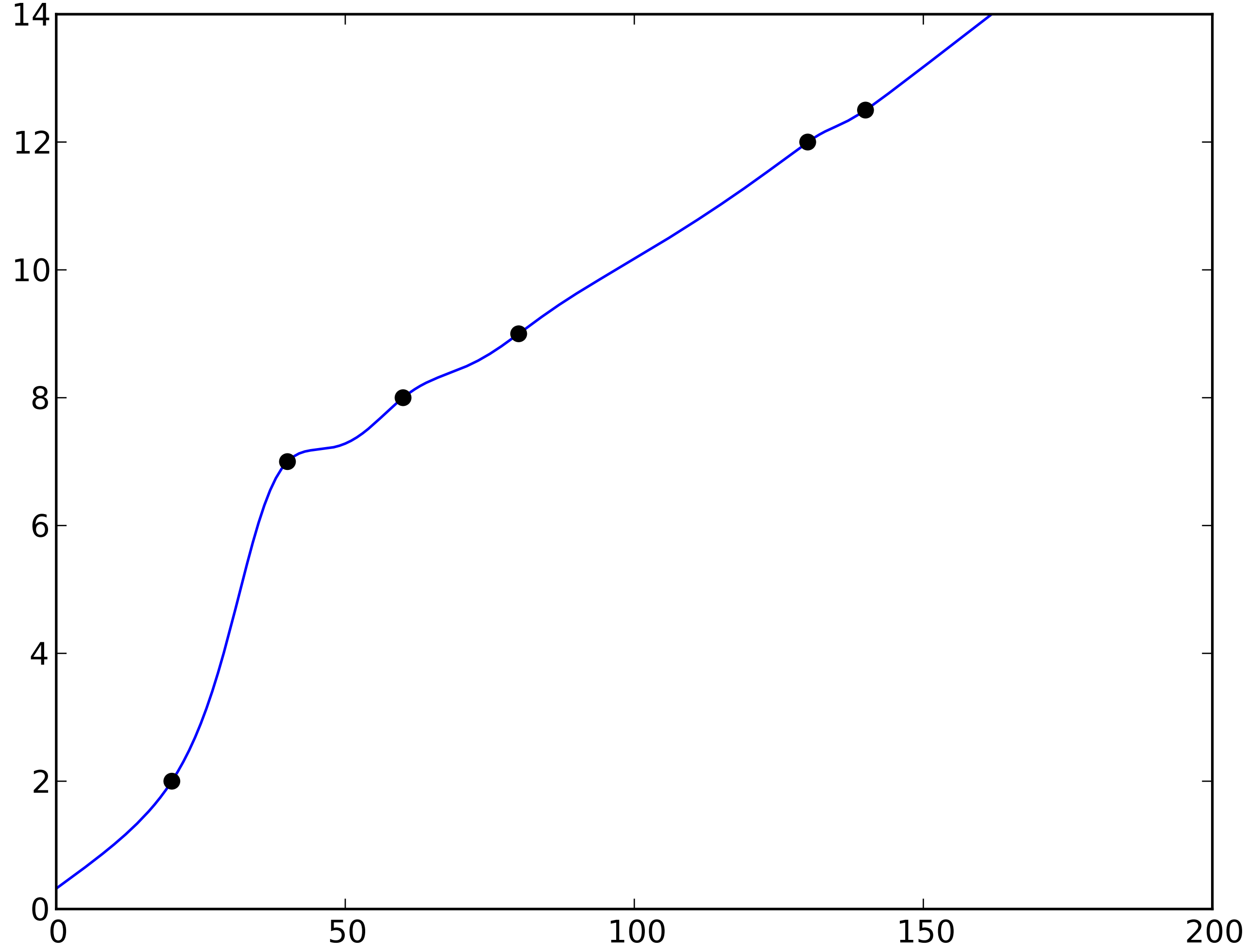
Comparison: Shepard's $p = 2$

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process



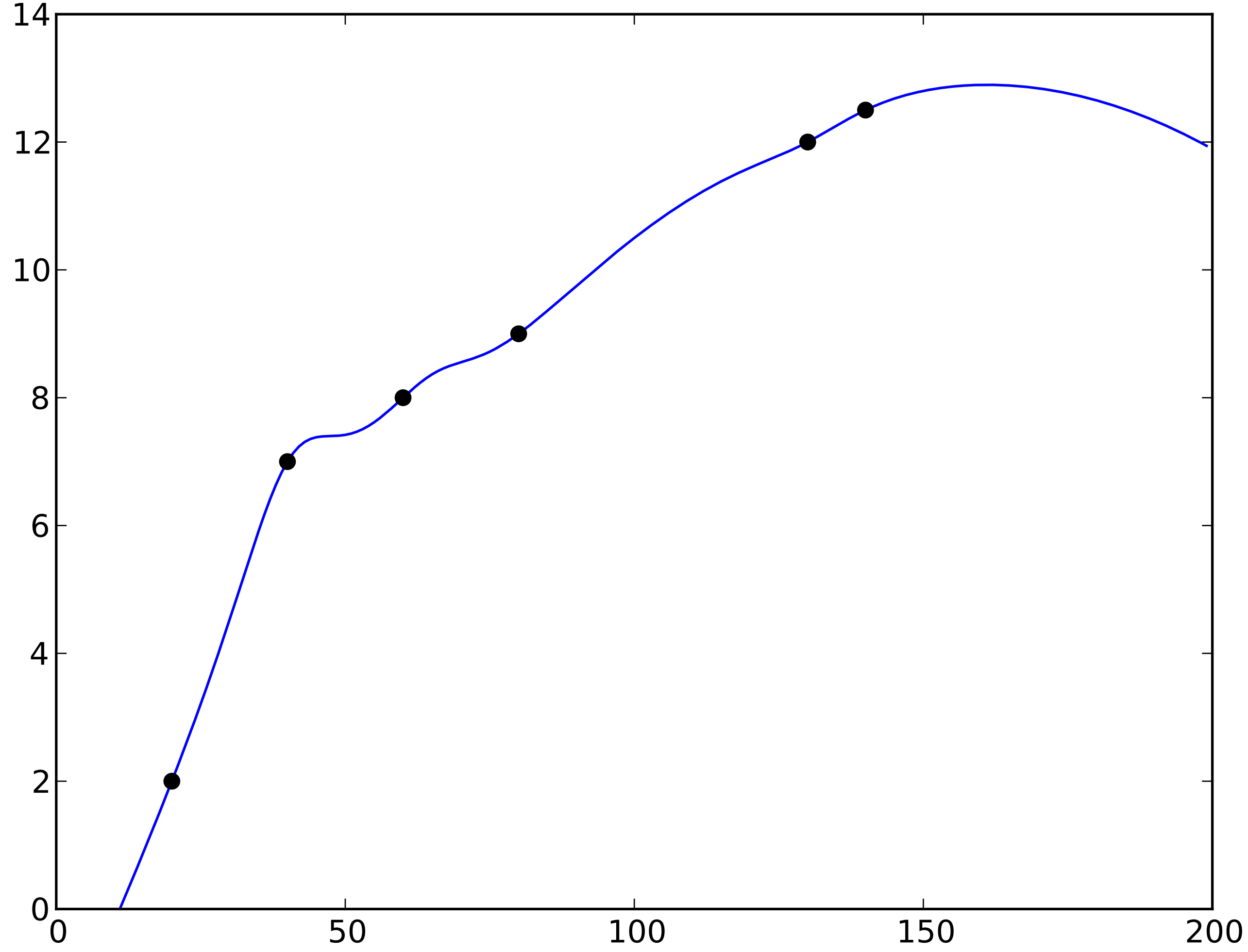
Comparison: Moving Least Squares, linear polynomial

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process



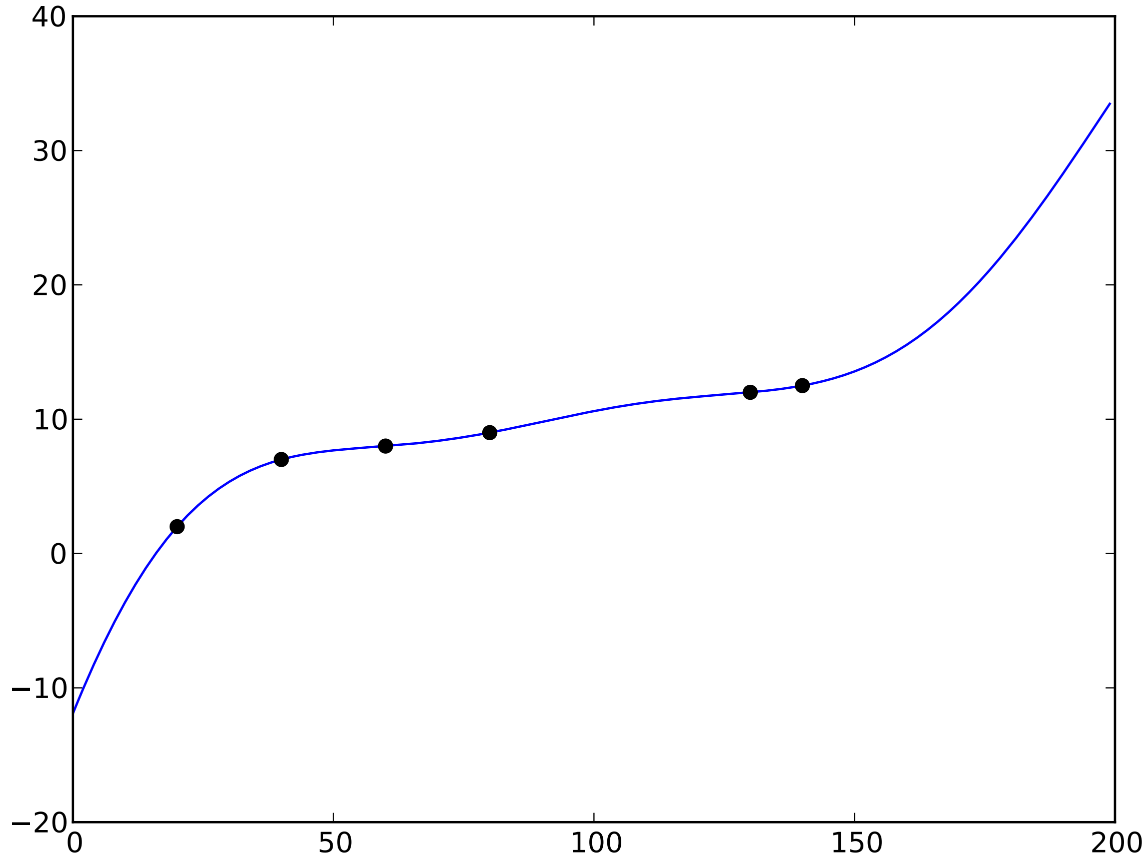
Comparison: Moving Least Squares, quadratic polynomial

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process



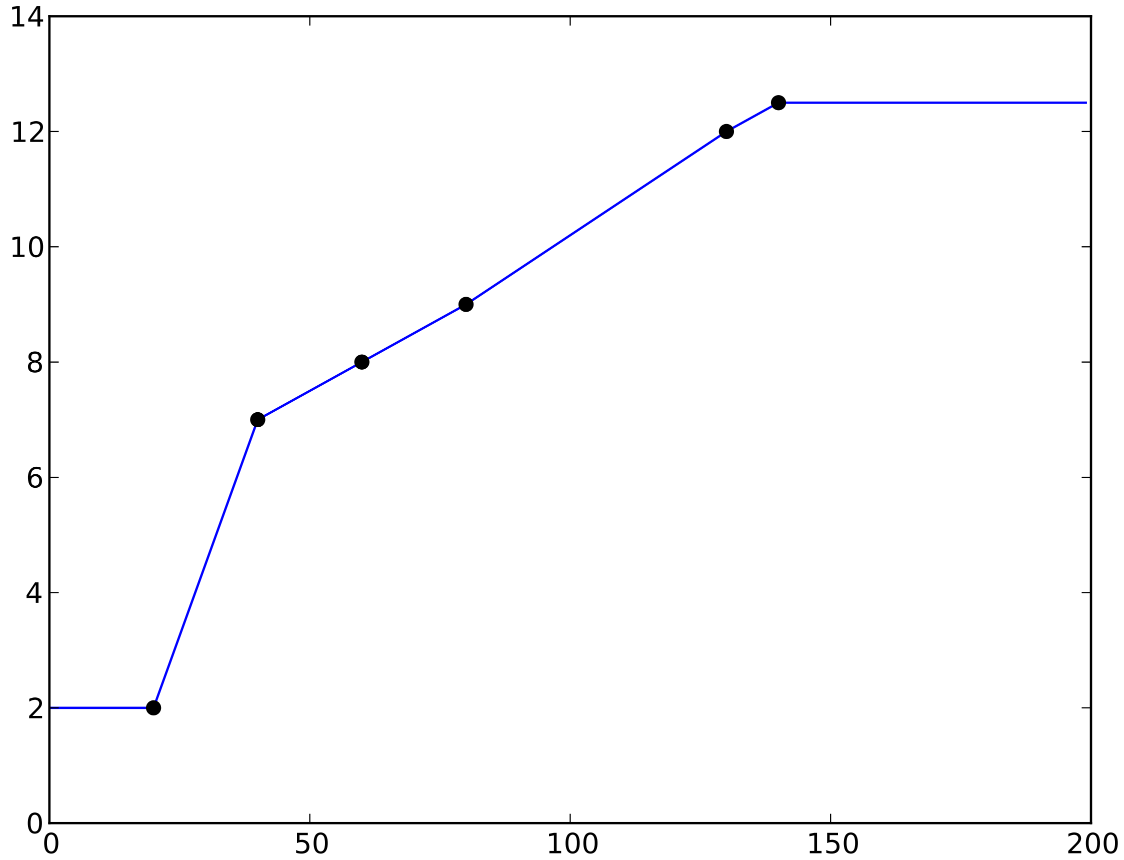
Comparison: GaussianProcess

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process



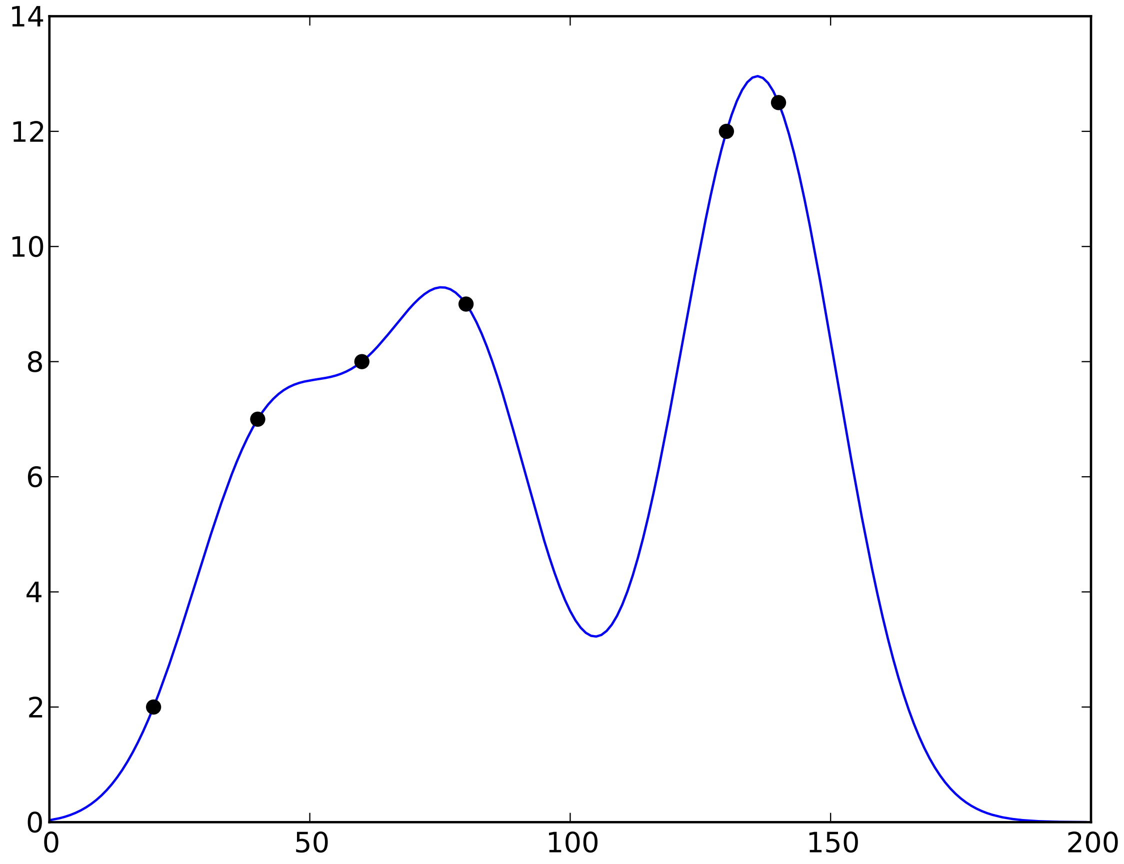
Comparison: Laplace

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process



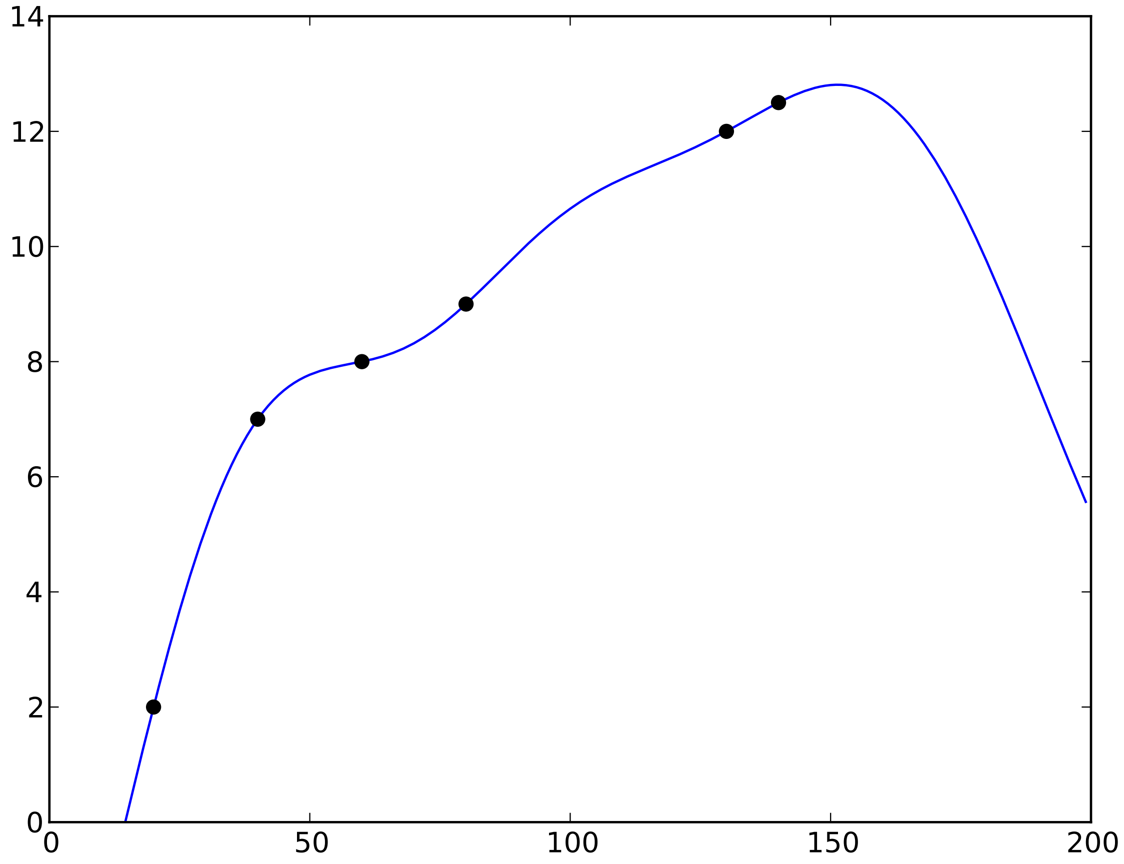
Comparison: RBF-Gauss

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process



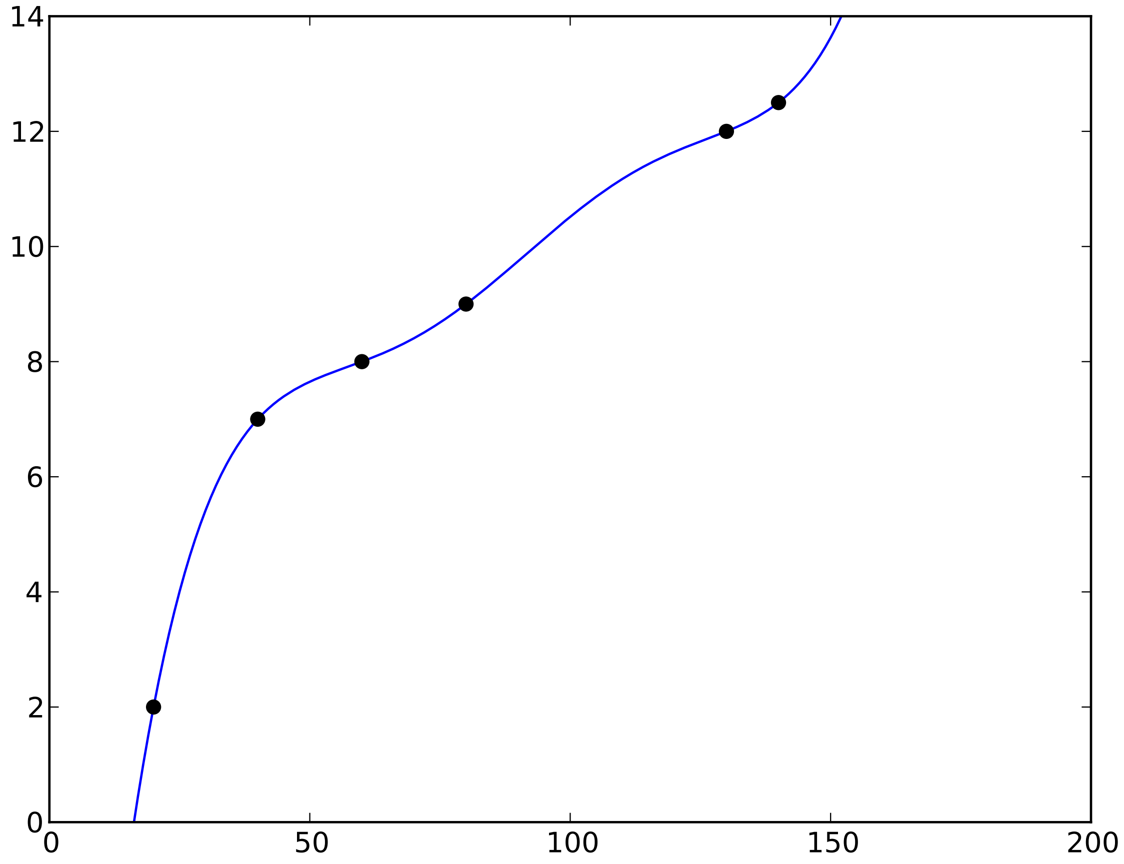
Comparison: RBF-Gauss

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process



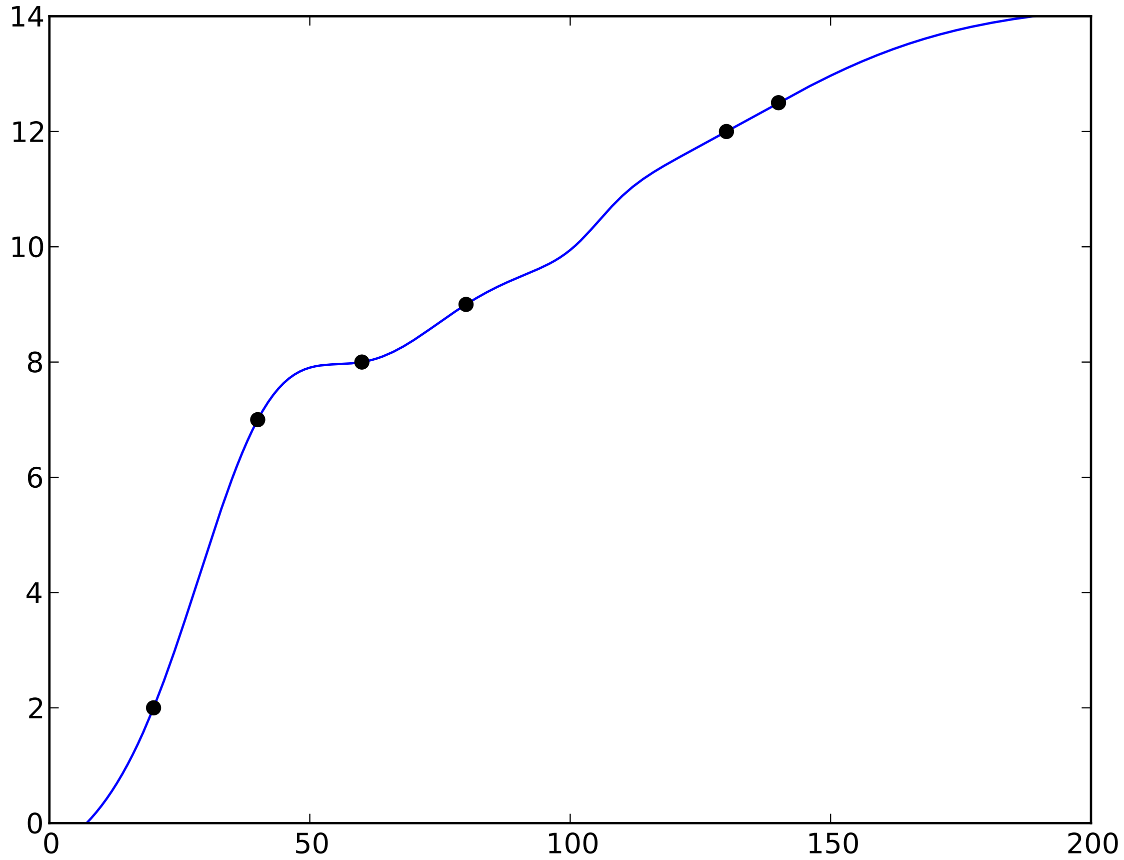
Comparison: RBF-Gauss

Euclidean invariant
Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process



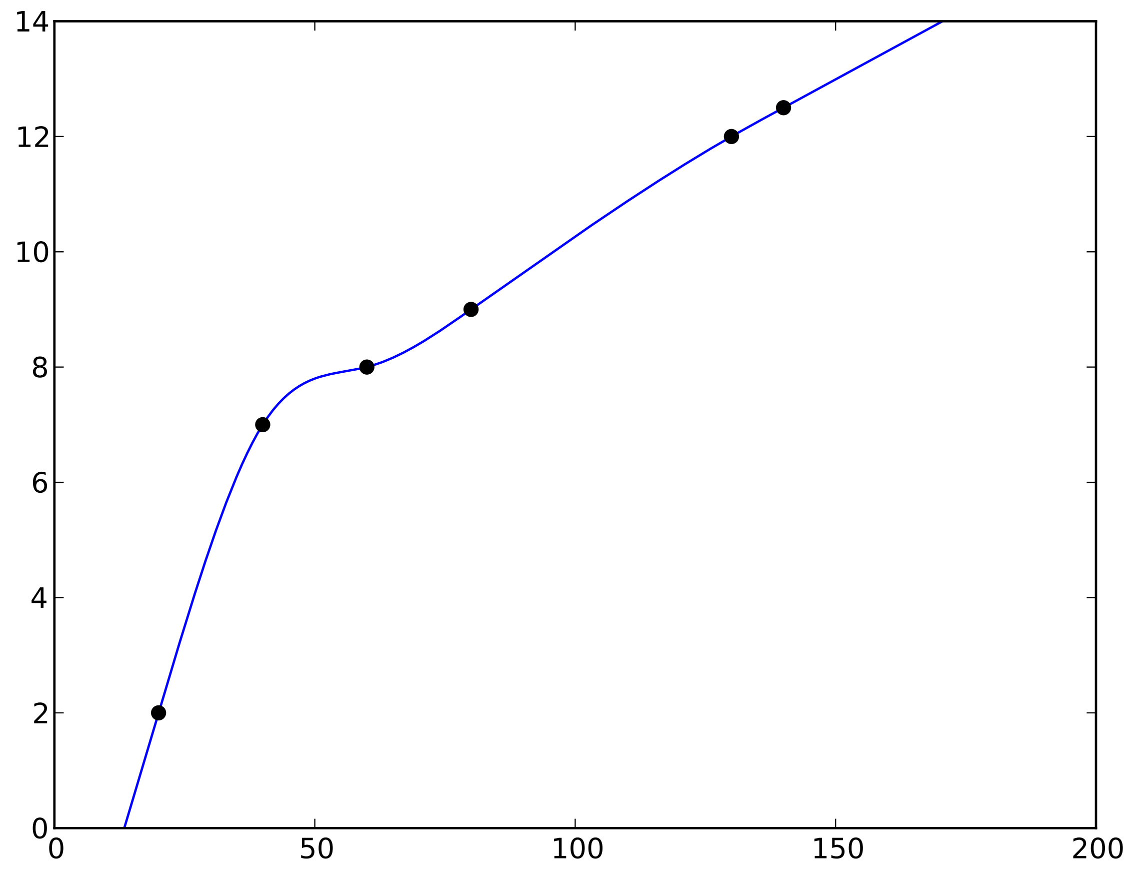
Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

Comparison: Normalized RBF-Gauss



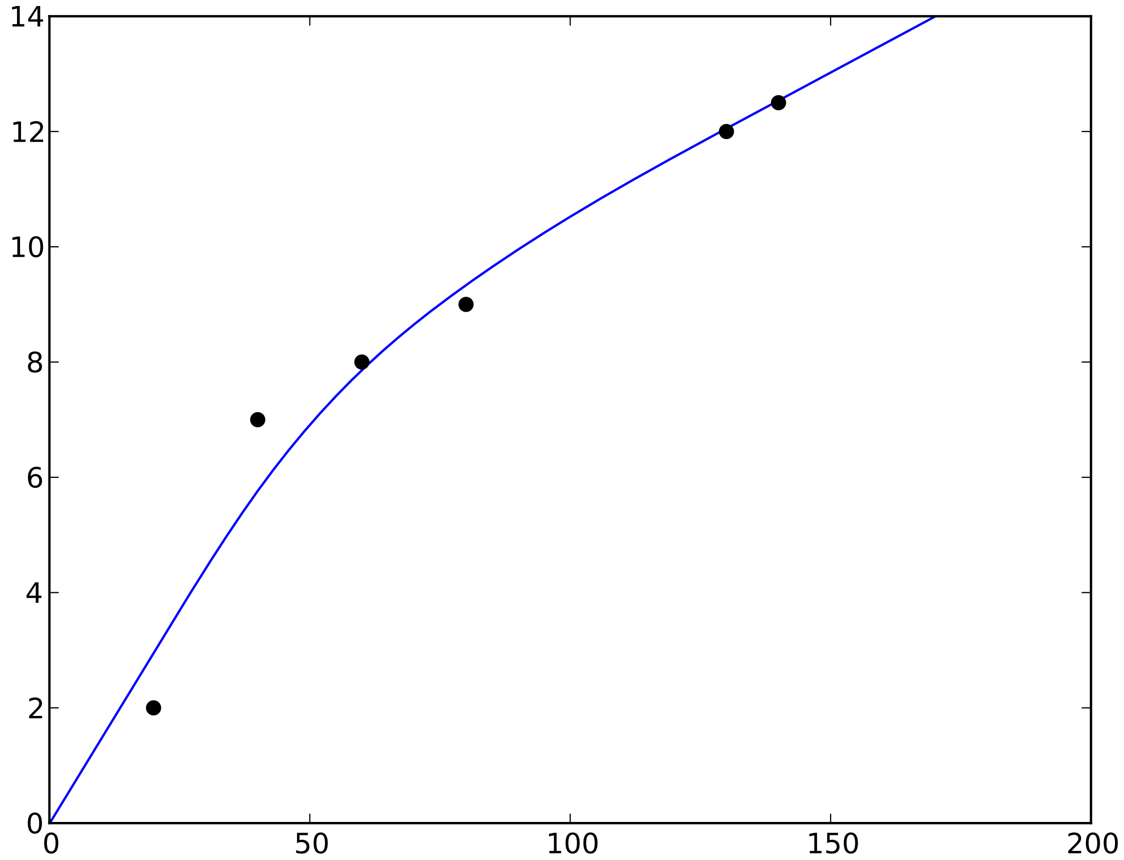
Comparison: Cubic (i.e. RBF-Thin plate)

Euclidean invariant
Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process



Comparison: Cubic + regularization

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process



Approximation rather than interpolation

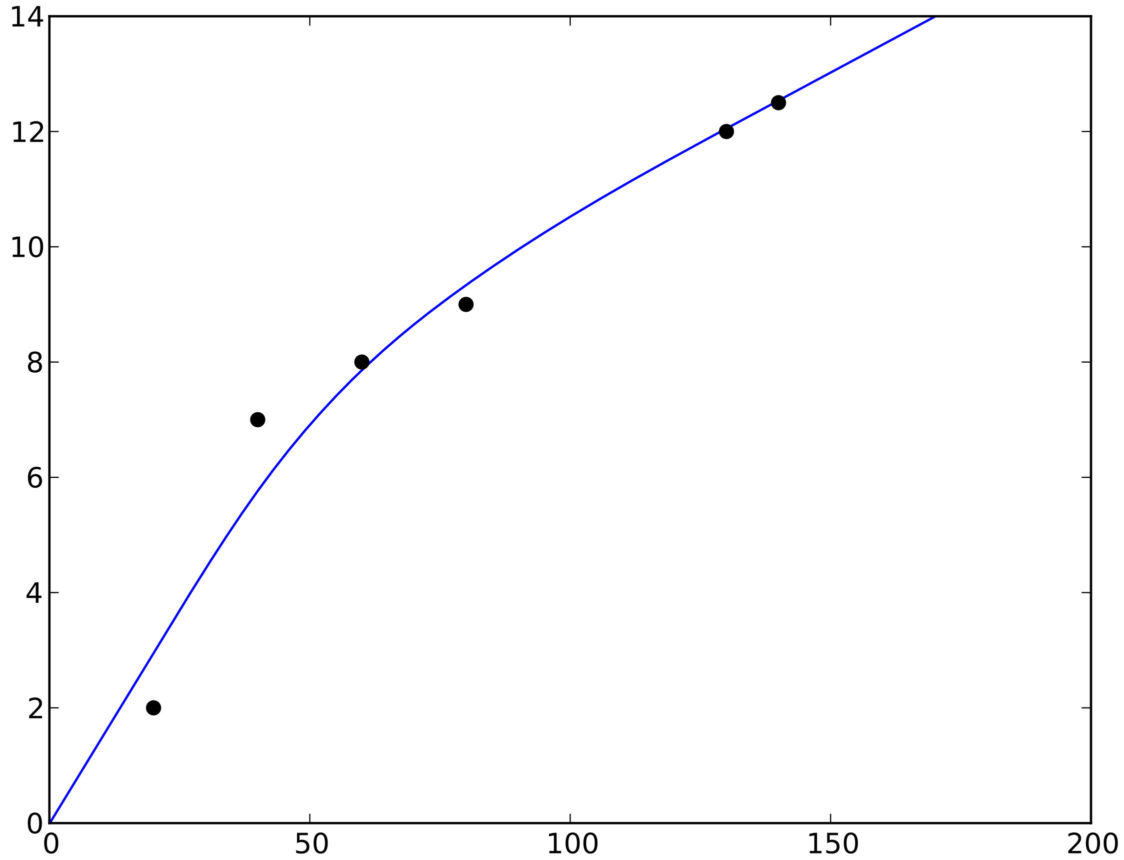
Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

Find $\mathbf{w}$ to minimize $(\mathbf{Rw} - \mathbf{b})^T(\mathbf{Rw} - \mathbf{b})$. If the training points are very close together, the corresponding columns of $\mathbf{R}$ are nearly parallel. Difficult to control if points are chosen by a user. Add a term to keep the weights small: $\mathbf{w}^T\mathbf{w}$.

$$
\begin{aligned}
\text{minimize} \qquad & (\mathbf{Rw} - \mathbf{b})^T(\mathbf{Rw} - \mathbf{b}) + \lambda \mathbf{w}^T\mathbf{w} \\
& \mathbf{R}^T(\mathbf{Rw} - \mathbf{b}) + 2\lambda\mathbf{w} = 0 \\
& \mathbf{R}^T\mathbf{Rw} + 2\lambda\mathbf{w} = \mathbf{R}^T\mathbf{b} \\
& (\mathbf{R}^T\mathbf{R} + 2\lambda\mathbf{I})\mathbf{w} = \mathbf{R}^T\mathbf{b} \\
& \mathbf{w} = (\mathbf{R}^T\mathbf{R} + 2\lambda\mathbf{I})^{-1}\mathbf{R}^T\mathbf{b}
\end{aligned}
$$

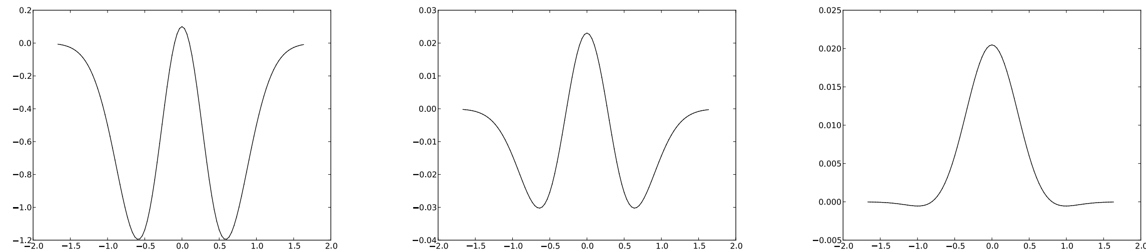
Comparison: Cubic + regularization

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

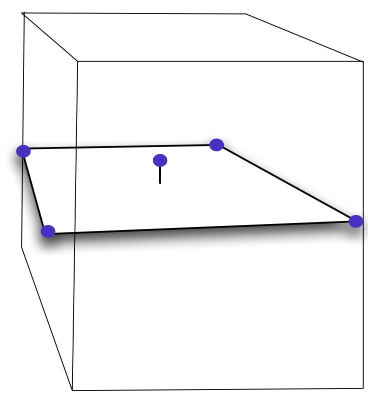


Regularization

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process



Ill-conditioning and regularization. The regularization parameter is 0, .01, and .1 respectively. (Vertical scale is changing).



Relation between Laplace,Thin-Plate, RBF

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

2D thin-plate interpolation

$$\hat{d}(\mathbf{p}) = \sum w_k R(\|\mathbf{p} - \mathbf{p}_k\|)$$

with $R(r) = r^2 \log(r)$.

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

Solving Thin plate interpolation

- if few known points: use RBF
- if many points use multigrid instead

- but Carr/Beatson et. al. (SIGGRAPH 01) use FMM for RBF with large numbers of points

Break

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

Relation between Laplace,Thin-Plate, RBF

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

2D thin-plate interpolation

$$\hat{d}(\mathbf{p}) = \sum w_k R(\|\mathbf{p} - \mathbf{p}_k\|)$$

with $R(r) = r^2 \log(r)$.

Where does $r^2 \log(r)$ come from??

Relation between Laplace,Thin-Plate, RBF

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

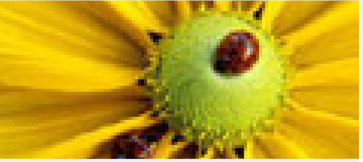
the "roughness penalizing" formulation,

$$\min_{\mathbf{f}} \int \|\nabla \mathbf{f}\|^2 d\mathbf{x}$$

The RBF solution

$$f(\mathbf{p}) = \sum w_k R(\mathbf{p} - \mathbf{p}_k)$$

R is essentially $(\nabla^2)^{-1}$.



Where does the rbf kernel come from?

Euclidean invariant
Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

Fit an unknown function f to the data y_k, regularized by minimizing a smoothness term.

$$\min_f = \sum (f_k - y_k)^2 + \lambda \int ||Pf||^2$$

e.g. $$||Pf||^2 = \int \left(\frac{d^2 f}{dx^2} \right)^2 dx$$

Variational derivative w.r.t. f leads to a differential equation

$$P^T P f(x) = \frac{1}{\lambda} \sum (f(x) - y_k) \delta(x - x_k)$$

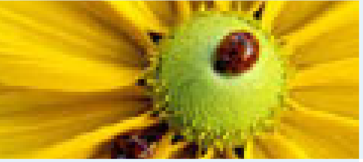
Where does the rbf kernel come from?

Euclidean invariant
Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

Solve linear differential equation by finding Green's function of the differential operator, convolving it with the RHS (works only for a linear operator). Schematically,

$$
\begin{aligned}
Lf &= rhs && L \text{ is the operator } P^T P, \\
& && \text{rhs is the data fidelity} \\
f &= g \star rhs && \text{f obtained by convolving g} \star \text{rhs} \\
L(g \star rhs) &= rhs && \\
Lg &= \delta && \text{choosing rhs} = \delta
\end{aligned}
$$

g is the "convolutional inverse" of L.



Where does the rbf kernel come from?

Euclidean invariant
Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

$$Lg = \delta$$

This is easier to solve in the Fourier domain, where convolution becomes multplication. The transform of δ is a constant, so in the Fourier domain g is the reciprocal of $L = P^T P$.

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

Where does the rbf kernel come from?

Fit an unknown function f to the data y_k, regularized by minimizing a smoothness term.

$$\min_f = \sum (f_k - y_k)^2 + \lambda ||Pf||^2$$

e.g. $$||Pf||^2 = \int \left(\frac{d^2 f}{dx^2}\right)^2 dx$$

A similar discrete version.

$$\min_{\mathbf{f}} = (\mathbf{f} - \mathbf{y})^T \mathbf{S}^T \mathbf{S} (\mathbf{f} - \mathbf{y}) + \lambda \mathbf{f}^T \mathbf{P}^T \mathbf{P} \mathbf{f}$$

Where does the rbf kernel come from?

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

(continued) A similar discrete version.

$$\min_{\mathbf{f}} = (\mathbf{f}-\mathbf{y})^T\mathbf{S}^T\mathbf{S}(\mathbf{f}-\mathbf{y}) + \lambda \mathbf{f}^T\mathbf{P}^T\mathbf{P}\mathbf{f}$$

- simplifying assumptions: uniform sampling, 1 dimension
- $\mathbf{S}$ is a diagonal "selection matrix" with 1s and 0s
- $\mathbf{P}$ is a diagonal-constant matrix that encodes the discrete form of the roughness operator, e.g.

$$\begin{bmatrix} -2, 1, 0, 0, \ldots \\ 1, -2, 1, 0, \ldots \\ 0, 1, -2, 1, \ldots \end{bmatrix}$$

Euclidean invariant
Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

Where does the rbf kernel come from?

Note $\mathbf{S}^T\mathbf{S} = \mathbf{S}$ because diagonal

Take the derivative with respect to the vector $\mathbf{f}$,

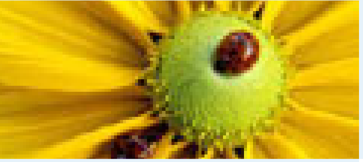
$$2\mathbf{S}(\mathbf{f} - \mathbf{y}) + \lambda 2\mathbf{P}^T\mathbf{P}\mathbf{f} = 0$$

$$\mathbf{P}^T\mathbf{P}\mathbf{f} = -\frac{1}{\lambda}\mathbf{S}(\mathbf{f} - \mathbf{y})$$

Multiply by $\mathbf{G}$, being the inverse of $\mathbf{P}^T\mathbf{P}$:

$$\mathbf{f} = \mathbf{G}\mathbf{P}^T\mathbf{P}\mathbf{f} = -\frac{1}{\lambda}\mathbf{G}\mathbf{S}(\mathbf{f} - \mathbf{y})$$

So the RBF kernel "comes from" $\mathbf{G} = (\mathbf{P}^T\mathbf{P})^{-1}$.



Where does kernel come from: Discrete/Continuous

Euclidean invariant Shepard Interpolation
Comparison: Shepard's $p = 1$
Comparison: Shepard's $p = 2$
Comparison: Shepard's $p = 5$
Kernel smoothing
Foley and Nielsen
Moving Least Squares
Moving Least Squares
Moving Least Squares
MLS = Shepard's when $m = 0$
Moving Least Squares
Moving Least Squares
Moving Least Squares
Natural Neighbor Interpolation
Natural Neighbor Interpolation
Notation
Gaussian Process regression
Gaussian Process

(Discrete version) RBF kernel is $\mathbf{G} = (\mathbf{P}^T\mathbf{P})^{-1}$.
Take SVD

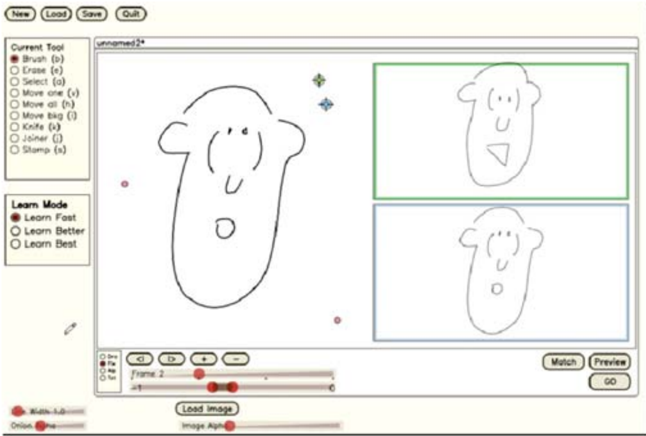
$$\mathbf{P} = \mathbf{U}\mathbf{D}\mathbf{V}^T \Rightarrow \mathbf{P}^T\mathbf{P} = \mathbf{V}\mathbf{D}^2\mathbf{V}^T$$

The inverse of $\mathbf{V}\mathbf{D}^2\mathbf{V}^T$ is $\mathbf{V}\mathbf{D}^{-2}\mathbf{V}^T$.

- eigenvectors of a circulant matrix are sinusoids,
- and $\mathbf{P}$ is diagonal-constant (toeplitz), or nearly circulant.
- So $\mathbf{V}\mathbf{D}^{-2}\mathbf{V}^T$ is approximately the same as taking the Fourier transform and then the reciprocal (remembering that $\mathbf{D}$ are the singular values of $\mathbf{P}$ not $\mathbf{P}^T\mathbf{P}$)

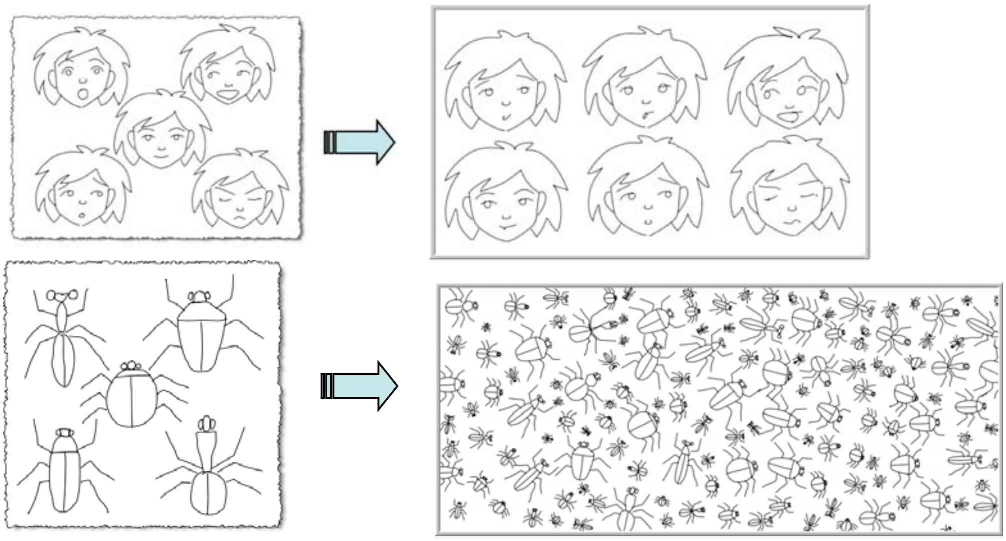
Learning Doodle by Example

Application Examples



Problem Definition

- Given *N* similar input drawings (doodles)
- Construct more doodles that resemble the *N* inputs
- Example results



Proposed Solution

- Construct a space of doodles defined by the inputs
  - Use results from machine learning and statistics:
    - Consider drawings as sample points in some space.
    - Similar doodles should be located nearby in the space.
    - We wish to fit (or learn) a continuous function over the space that "explains" the examples as well.
- We call the result a "Latent Doodle Space"(LDS).

‣ See the paper in EUROGRAPHICS06 (by Baxter and Anjyo) for details.

Two Main Challenges

To build a latent doodle space (LDS):

- Find correspondences between two line drawings.
  - Hard problem. No perfect solution.
    - ‣ Do best possible, but still must have a good UI.
- Generate the space of similar drawings.
  - Use Bayesian techniques and statistical methods to improve results.

How to construct the LDS?

To generate the space of similar drawings (LDS):

- Two main tasks :
  - Finding latent coordinate system ⇐ PCA
  - Interpolating within LDS ⇐ RBF
- Three options in the EG06 paper: PCA+RBF, PCA+GP, GPLVM

➡ This talk focuses on the first strategy: PCA+ RBF

Dimension reduction by PCA

- A line drawing (doodle) has the structure:
  - line drawing l_k - stroke s_p - linear segments a_v^p
- After establishing correspondences among line drawings:
  - Construct the data matrix X (with zero-mean):

(x, y) → $a^1{}_0$

stroke s_1 — *stroke* s_2

$$\begin{array}{l} \textit{line drawing } l_1 \rightarrow \\ \textit{line drawing } l_2 \rightarrow \end{array} \begin{pmatrix} a^1{}_0 & a^1{}_1 & \cdots & a^1{}_{s_1} & b^1{}_0 & \cdots & b^1{}_{s_2} & \cdots \\ a^2{}_0 & \cdots & & a^2{}_{s_1} & b^2{}_0 & \cdots & b^2{}_{s_2} & \\ \vdots & & & & & & & \end{pmatrix} = X$$

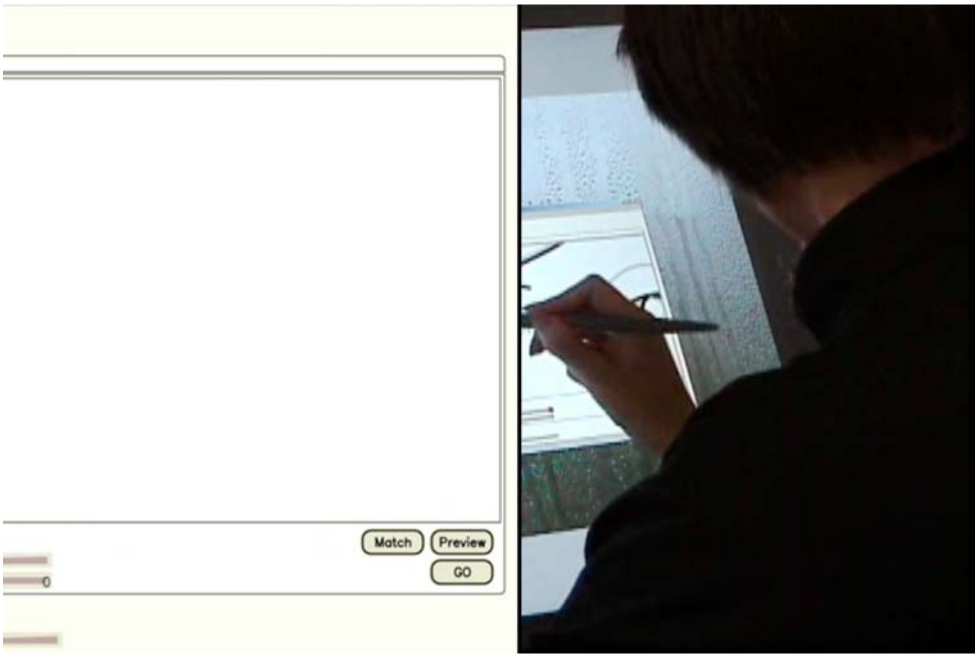
  - Applying PCA: Eigen decomposition of the covariance matrix $X^T \cdot X$

2-d LDS and RBF

- Make the LDS 2-dim by taking the eigenvectors with the first two largest eigenvalues.
- Interpolate each of s, t (scalar) values of the drawing samples by *thin plate spline*:
  - space dimension n = 2 and smoothness m = 2 (explain later!)
  - The spline function is: $\phi(r) = r^2 \log r$ where $r := \|(s,t)\| = \sqrt{s^2 + t^2}$
  - The interpolant is of the form:

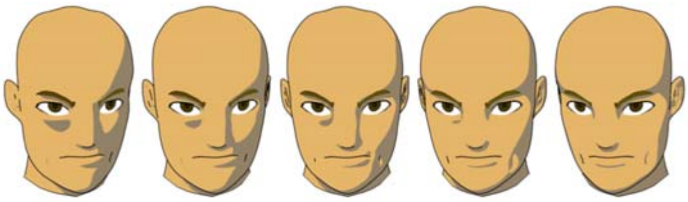
$$f(s,t) := \sum_i w_i \phi(s - s_i, t - t_i) + as + bt + c$$

Demo



Locally Controllable Stylized Shading

Application Examples



Background: Cartoon shading

- Based on thresholded **N•L** shading model

N•L intensity distribution

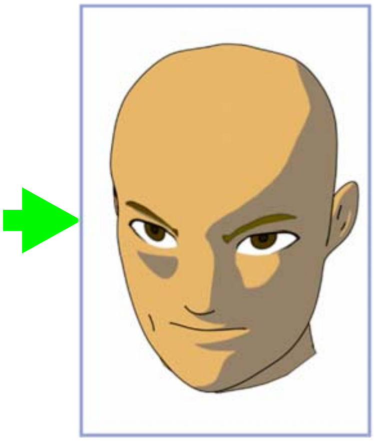
Motivation: Fake but expressive shading


With conventional shader


Artists want neat shading

Our Goal


conventional

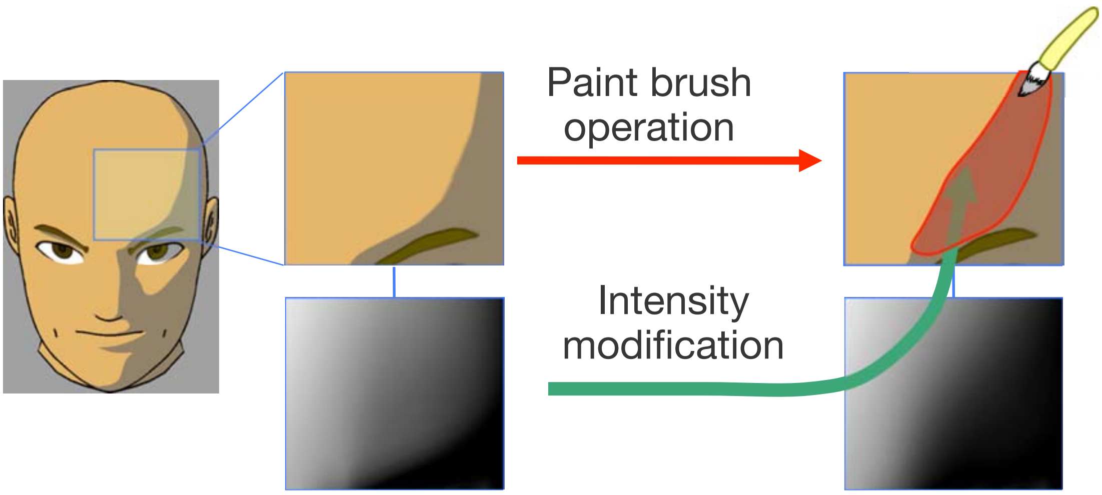

desired

- Make toon shader artist-friendly
  - Edit undesirable shaded area
  - Add artistic light and shade to original 3D lighting

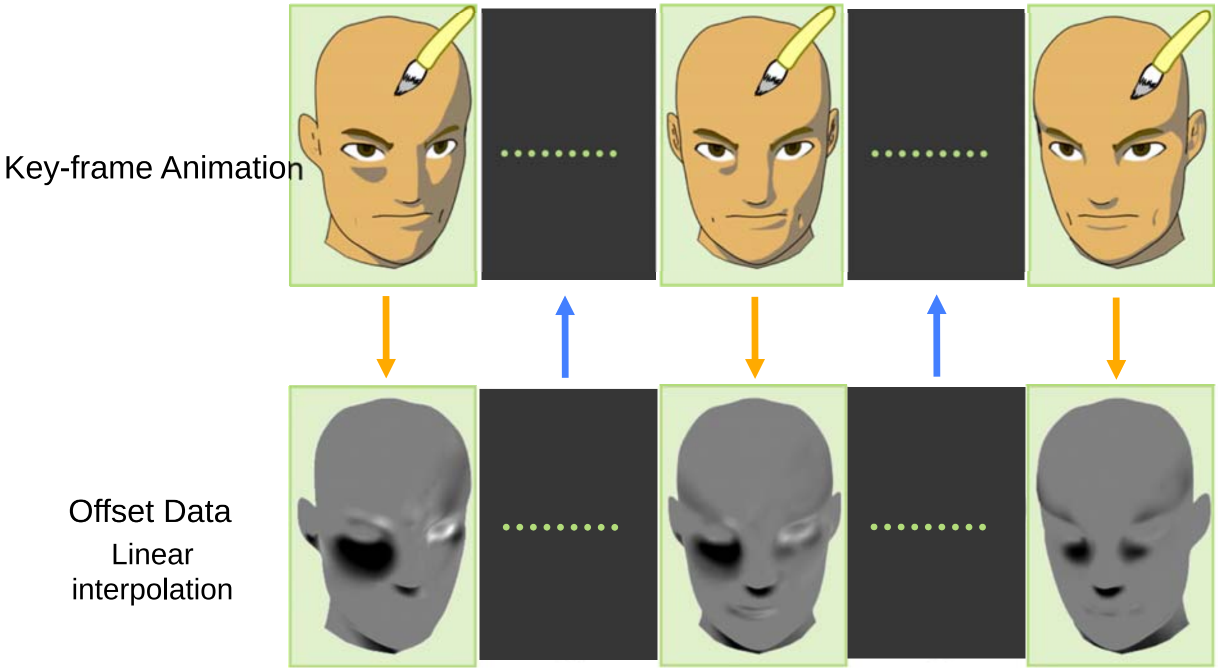
Video demonstration

The shade painter (our approach)

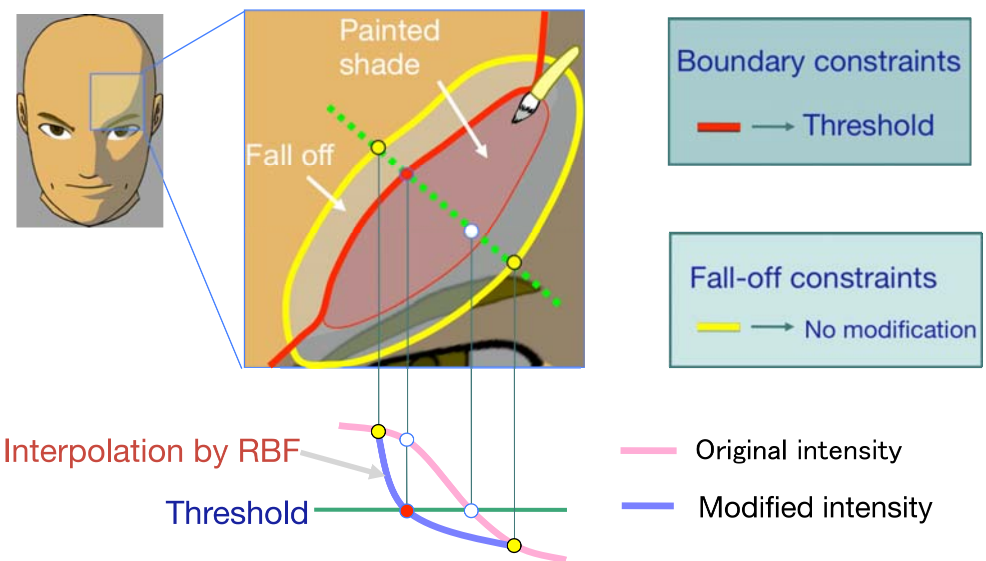
- The main idea:
  Modify intensity according to painted strokes



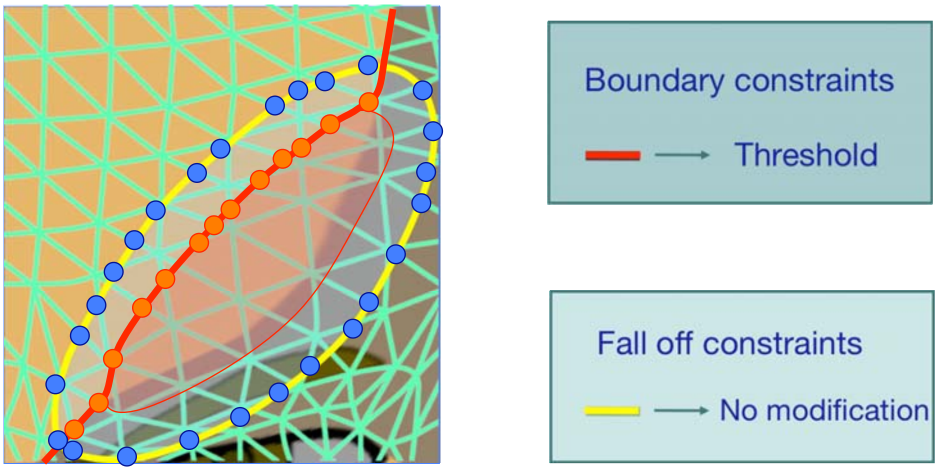
Keyframing



Intensity modification by painting



RBF interpolation



- The discretized constraints for the RBF are specified for the points ● and ●.

The unknown values

- We employ the following interpolation function:

$$f(x) = \sum_{i=1}^{l} w_i \phi(x - \mathrm{c}_i) + P(x)$$

where $x = (x_1, x_2, x_3)$, and $\phi(x) := \|x\| \equiv \sqrt{x_1^{\,2} + x_2^{\,2} + x_3^{\,2}}$; $P(x)$ is a linear polynomial of x_1, x_2, x_3; c_i means the constraint points (● and ●); l is the number of all the constraint points.

- We want to determine the weights $\{w_k\}$ and the four coefficients of $P(x)$

  → Totally $l + 4$ unknown values.

For the given values $h_{j\ (1 \le j \le l)}$, we have:

$$\sum_{i=1}^{l} w_i \phi(\mathrm{c}_j - \mathrm{c}_i) + P(\mathrm{c}_j) = h_j, \quad for\ 1 \le j \le l.$$

The unknown values

- We further add the following condition:

for any linear polynomial $\sum_{i=1}^{l} w_i Q(\mathrm{c}_i) = 0.$

→ Taking Q = $1, x_1, x_2,$ and x_3 , we know that the above condition means:

$$\sum_{i=1}^{l} w_i = 0 \qquad \sum_{i=1}^{l} w_i c_{ij} = 0, \quad (j = 1, 2, 3)$$

The linear equation for RBF

- By putting $P(x) = p_0 + p_1 x_1 + p_2 x_2 + p_3 x_3$ and $\phi_{ij} := \phi(\mathrm{c}_i - \mathrm{c}_j)$, we have the following *linear* equation:

$$\begin{pmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1l} & 1 & c_{11} & c_{12} & c_{13} \\ \phi_{21} & \ddots & & & \vdots & \vdots & \ddots & \vdots \\ \vdots & & & & & & & \\ \phi_{l1} & & & \phi_{ll} & 1 & c_{l1} & c_{l2} & c_{l3} \\ 1 & 1 & \cdots & 1 & 0 & 0 & 0 & 0 \\ c_{11} & & \cdots & c_{l1} & \vdots & \ddots & & \vdots \\ c_{12} & & \cdots & c_{l2} & \vdots & & \ddots & \vdots \\ c_{13} & & \cdots & c_{l3} & 0 & \cdots & \cdots & 0 \end{pmatrix} \begin{pmatrix} w_1 \\ \\ \\ w_l \\ p_0 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} h_1 \\ \vdots \\ h_l \\ 0 \\ \vdots \\ \vdots \\ 0 \end{pmatrix}$$

Locally Controllable Stylized Shading

System Overview

Locally Controllable Stylized Shading

Example Animations

Functional Analysis, RBF and RKHS connections

RBF and RKHS

Recall the RBFs in our examples

Background
Differential equation for TPS
RBF as TPS
Regularization problem

In our examples:

- The RBF interpolants can be expressed in such a form that:

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

▸ Latent doodle space uses TPS: $G(\boldsymbol{x}) = \|\boldsymbol{x}\|^2 \boldsymbol{log}\|\boldsymbol{x}\|$

- followed by linear polynomial $p(\boldsymbol{x}) = p(x_1,x_2) \equiv c_0 + c_1x_1 + c_2x_2$

▸ The shade painter employs $G(\boldsymbol{x}) = \|\boldsymbol{x}\| = \sqrt{x_1^{\,2} + x_2^{\,2} + x_3^{\,2}}$ and p is linear:

$$p(\boldsymbol{x}) = p(x_1,x_2,x_3) \equiv p_0 + p_1x_1 + p_2x_2 + p_3x_3 \; .$$

✓ PSD applications use Gaussian RBF with no polynomial term ($p(\boldsymbol{x}) \equiv 0$).

In the shade painter case

- The shade painter uses $G(\boldsymbol{x}) = \|\boldsymbol{x}\| = \sqrt{x_1^{\,2} + x_2^{\,2} + x_3^{\,2}}$ and the interpolant:

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

- Point constraints: $f_k = \sum_{i=1}^{N} \alpha_i G(\boldsymbol{x}_k - \boldsymbol{x}_i) + p(\boldsymbol{x}_k) \quad (k = 1, ..., \mathrm{N})$

- Vanishing moments: $\sum_{i=1}^{N} \alpha_i = 0. \quad \sum_{i=1}^{N} \alpha_i x_{i\,j} = 0, \quad j = 1,2,3.$

  where $\boldsymbol{x}_i = (x_{i1}, x_{i2}, x_{i3})^T$.

- We have (N+4) linear equations for (N+4) unknown values.

  ➡ Get the values of $\{\alpha_i\}$ and the four coefficients of p by solving the linear equation system!

So why?

- Where do RBFs come from?
- Why vanishing moment condition?
- Where is Gaussian RBF?

➡ Functional analysis might be helpful to answer them.

Thin Plate Spline Revisited

Background
Differential equation for TPS
RBF as TPS
Regularization problem

Background

- Let $\Omega \; in \; R^2$. Find a function $\varphi\,(x)$ defined on Ω that minimizes the following energy:

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

- Where can we find the solution ?

  - As a necessary condition, it should belong to:

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

where

$$L^2(\Omega) = \left\{ \varphi : \Omega \to R \cup \{\pm\infty\} \mid \int_\Omega |\varphi(\boldsymbol{x})|^2 \, dx_1 dx_2 < \infty \right\}.$$

Differential equation for TPS

- Cauchy's idea: Let $f = \mathbf{arg}\, F$. Then consider $G(t) = F(\boldsymbol{f} + t\,\boldsymbol{g})$, where

  $t \in R, \boldsymbol{g}$ is a smooth function with compact support:

$$\boldsymbol{g}(\boldsymbol{x}) = 0, \; \textit{if} \; \boldsymbol{x} \in \partial\Omega.$$

  - $G(t)$ is then a quadratic function of t :

$$G(t) = t^2 \int \left({g_{x_1 x_1}}^2 + 2{g_{x_1 x_2}}^2 + {g_{x_2 x_2}}^2 \right) d\boldsymbol{x}$$
$$+ \, t \int \left(2 f_{x_1 x_1} g_{x_1 x_1} + 4 f_{x_1 x_2} g_{x_1 x_2} + 2 f_{x_2 x_2} g_{x_2 x_2} \right) d\boldsymbol{x} + \text{(const)}.$$

  - and it must satisfy $G'(0) = 0$.

Differential equation for TPS

- Cauchy's idea (cont.):
  - $G'(0) = 0$

$$\Rightarrow 0 = \int \left(f_{x_1x_1} g_{x_1x_1} + 2 f_{x_1x_2} g_{x_1x_2} + f_{x_2x_2} g_{x_2x_2} \right) d\boldsymbol{x}$$

integration by parts (twice!)

$$\Rightarrow = -\int \left(f_{x_1x_1x_1} g_{x_1} + 2 f_{x_1x_2x_1} g_{x_2} + f_{x_2x_2x_2} g_{x_2} \right) d\boldsymbol{x}$$

$$= \int \left(f_{x_1x_1x_1x_1} + 2 f_{x_1x_2x_1x_2} + f_{x_2x_2x_2x_2} \right) g \, d\boldsymbol{x}$$

$$= \int \left(\Delta^2 f \right) \cdot g \, d\boldsymbol{x}$$

  - Since g is arbitrary, we have: $\Delta^2 \boldsymbol{f} \equiv \left(\frac{\partial^2}{\partial x_1{}^2} + \frac{\partial^2}{\partial x_2{}^2} \right)^2 \boldsymbol{f} = 0.$

RBF as TPS

- We'll use Green's function $\phi(\boldsymbol{x})$ associated with the differential operator Δ^2:

$$\Delta^2 \phi(\boldsymbol{x}) = \delta(\boldsymbol{x}) \longrightarrow \phi(\boldsymbol{x}) = \|\boldsymbol{x}\|^2 \boldsymbol{log} \|\boldsymbol{x}\|$$

  - $\delta(\boldsymbol{x})$ is the Dirac delta function.

The Regularization Problem with TPS

- We treat a simple case where $\Omega = \boldsymbol{R}^2$.

- For given data $(\boldsymbol{x}_i, f_i) \in \boldsymbol{R}^2 \times \boldsymbol{R}$ $(i = 1, 2, ..., \mathrm{N})$, find a solution $\boldsymbol{f}$:

$$\min_{\boldsymbol{f}} \left\{ \sum_{i=1}^{\mathrm{N}} \left(f_i - \boldsymbol{f}(\boldsymbol{x}_i) \right)^2 + \lambda F(\boldsymbol{f}) \right\}$$

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

  - λ is also given, called the *regularization parameter*.

- Where can we find the solution? $B_2^2(\Omega)$?:

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

Regularization Problem in Function Space

Problem statement

Generalizing the TPS regularization problem

- Let $\Omega = \boldsymbol{R}^n$. Generalize the TPS regularization problem into higher dimensions and, instead of F, with:

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

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

  - With this definition, $F = J_2^2$.

- For given data $(\boldsymbol{x}_i, f_i) \in \boldsymbol{R}^n \times \boldsymbol{R}$ $(i = 1, 2, \ldots, \mathrm{N})$, find a solution $\boldsymbol{f}$:

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

- We need to decide where we find the solution - "Function Space".

Function Space

- A function space is a totality of functions that share common properties.
- Examples:

1. $\mathcal{P}_m := \{P(x) | P(x)$ is a polynomial of at most m-th order $\}$.
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. $L^p(\Omega) := \{f : \Omega \to R \cup \{\pm\infty\} | \int |f(x)|^p dx < \infty\}$, where p is a positive number.

Regularization Problem in $\boldsymbol{B}_m^n$

- For given data $(\boldsymbol{x}_i, f_i) \in \boldsymbol{R}^n \times \boldsymbol{R}$ $(i = 1, 2, ..., \mathrm{N})$, find a solution $\boldsymbol{f}$:

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

- The function space where we want to find the solution is:

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

  - Simple generalization of TPS, where we set $m = n = 2$ and $J_2^2(\boldsymbol{f}) = F(\boldsymbol{f})$.

Role of the parameters

- n: dimension of variables
  - $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T \in \boldsymbol{R}^n$.
- m: degree of smoothness of the solution
  - J_m^n includes up to m-th order derivatives.
- λ: regularization parameter
  - Specifies the trade-off between minimization of the first term $\sum_{i=1}^{\mathrm{N}} (f_i - \boldsymbol{f}(\boldsymbol{x}_i))^2$ and smoothness of the solution enforced by $\boldsymbol{J}_m^n$.

Solution

- If $2\mathrm{m} - \mathrm{n} > 0$, the solution is then given by:

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

  - $G(\boldsymbol{x}) = \begin{cases} |\boldsymbol{x}|^{2m-n} \log |\boldsymbol{x}| & \text{if } 2m-n \text{ is an even integer,} \\ |\boldsymbol{x}|^{2m-n} & \text{otherwise,} \end{cases}$
  - p is a polynomial $\in \mathcal{P}_{m-1}$.

- We need to get the weights $\{\alpha_i\}$ and the coefficients of the polynomial.
  - The *vanishing moment* condition is then satisfied:

$$\sum_{k=1}^{N} \alpha_k Q(\boldsymbol{x}_k) = 0, \quad \textit{for all } Q \in \mathcal{P}_{m-1}$$

In our examples:

- The RBF interpolants are given by:

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

  - $G(\boldsymbol{x}) = \begin{cases} |\boldsymbol{x}|^{2m-n} \log |\boldsymbol{x}| & \text{if } 2m-n \text{ is an even integer,} \\ |\boldsymbol{x}|^{2m-n} & \text{otherwise,} \end{cases}$
  - p is a polynomial $\in \mathcal{P}_{m-1}$.

  - Latent doodle space uses TPS, where m = n = 2 and p is a linear polynomial.
  - The shade painter deals with the case where m = 2, n = 3, and p is linear.
- The regularization problem in $\boldsymbol{B}_m^n$ does not explain the PSD cases.

Functional Analysis and RKHS

What is functional analysis?
RBF and RKHS

What is functional analysis?

- Mathematical theory of functions, differential equations.
- Deals with "generalization" of function, derivatives, ...
- Usually it's a different thing from what we want to know...
- Function space is a concept of infinite dimensional geometry.
  - Ex: $\boldsymbol{R}^n \Rightarrow \quad l^2 \quad \Rightarrow \quad L^2.$
- See the detailed discussions in our course notes.

Basic properties of J_m^n

- $J_m^n(f) = (-1)^m \langle f, \Delta^m f \rangle_{L^2}$.
  - This formula can be obtained through integration by parts.

    ➡ Recall the technique of "integration by parts" in the TPS case.

- $J_m^n(f) = 0 \;\Leftrightarrow\; f \in \mathcal{P}_{m-1}$

➡ We therefore have: $\boldsymbol{B}_m^n = \boldsymbol{H}_m^n \oplus \mathcal{P}_{m-1}$.

- $\boldsymbol{H}_m^n$ is called a Reproducing Kernel Hilbert space.

RKHS and RBF

- The RBF interpolants are given by:

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

  - The first term belongs to RKHS $\boldsymbol{H}_m^n$.

- Roughly speaking, an RKHS is a function space spanned by the finite sum of { $G(\boldsymbol{x} - \boldsymbol{c}_k)$ }. Particularly in solving the regularization problem, we can take $\boldsymbol{c}_i = \boldsymbol{x}_i$ for $1 \le i \le \mathrm{N}$ (The representer theorem).

- $\boldsymbol{H}_m^n$ is a normed space with $\|f\|^2_{\boldsymbol{H}_m^n} = J_m^n(f)$, which also means that

$$\langle f, g \rangle_{\boldsymbol{H}_m^n} := \sum_{\alpha_1+\alpha_2+\cdots+\alpha_n=m} \frac{m!}{\alpha_1!\alpha_2!\cdots\alpha_n!} \langle D^\alpha f, D^\alpha g \rangle_{L^2}$$

The kernel

- In our situations (examples), if we set $K(\boldsymbol{x},\boldsymbol{y}) := G(\boldsymbol{x}-\boldsymbol{y})$, then $\boldsymbol{K}$ has the following properties:

  - $\boldsymbol{K}$ is a symmetric, positive semi-definite function.

  ➡ This gives an alternative definition of RKHS.

  ✓ $\boldsymbol{K}$ is called the kernel function of RKHS.

- G is characterized by $\Delta^m G(\boldsymbol{x}) = \delta(\boldsymbol{x})$.
  - This yields that
  - $$G(\boldsymbol{x}) = \begin{cases} |\boldsymbol{x}|^{2m-n}\log|\boldsymbol{x}| & \text{if } 2m-n \text{ is an even integer,} \\ |\boldsymbol{x}|^{2m-n} & \text{otherwise} \end{cases}.$$

The vanishing moment condition

- This condition follows from the fact that $\boldsymbol{B}_m^n = \boldsymbol{H}_m^n \oplus \mathcal{P}_{m-1}$.

  - We note that, for any $\boldsymbol{f}$ and $\boldsymbol{g} \in \boldsymbol{B}_m^n$, $\langle \boldsymbol{f}, \boldsymbol{g}\rangle_{\mathrm{B}_m^n} = (-1)^m \langle \Delta^m \boldsymbol{f}, \boldsymbol{g}\rangle_{L^2}$.

  - Substitute $\boldsymbol{f} = \sum_{i=1}^{N} \alpha_i G(\boldsymbol{x}-\boldsymbol{x}_i)$ and $Q(\boldsymbol{x}) \in \mathcal{P}_{m-1}$. We thus have:

$$0 = \left\langle \sum_{i=1}^{N} \alpha_i G(\boldsymbol{x}-\boldsymbol{x}_i), Q(\boldsymbol{x}) \right\rangle_{B_m^n} = \sum_{i=1}^{N} \alpha_i \langle \Delta^m G(\boldsymbol{x}-\boldsymbol{x}_i), Q(\boldsymbol{x})\rangle_{L^2}$$

$$= \sum_{i=1}^{N} \alpha_i \langle \delta(\boldsymbol{x}-\boldsymbol{x}_i), Q(\boldsymbol{x})\rangle_{L^2} = \sum_{i=1}^{N} \alpha_i Q(\boldsymbol{x}_i).$$

The Gaussian RBF

- Instead of J_m^n, consider $\sum_{m\geq 0} a_m J_m^n$ with $a_m = \frac{\sigma^{2m}}{m!2^m}\,(\sigma > 0)$.
- The Green's function of the differential operator $\sum_{m\geq 0} (-1)^m \Delta^m$ is:

$$G(\boldsymbol{x}) = c\ exp(-\frac{\|\boldsymbol{x}\|^2}{2\sigma^2}).$$

- $\boldsymbol{f} = \sum_{i=1}^{N} \alpha_i G(\boldsymbol{x} - \boldsymbol{x}_i)$ is the solution of the regularization problem (We don't need a polynomial term this time).

So why?

- Where do RBFs come from?
- Why vanishing moment condition?
- Where is Gaussian RBF?

➡ Functional analysis should be helpful to answer them.