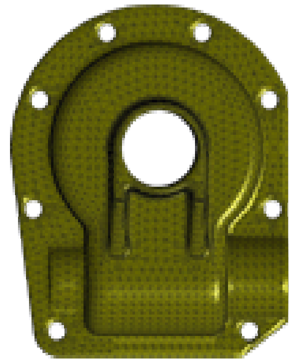
Methods for analysing discrete surfaces and their applications

Silvia Biasotti
Giuseppe Patané
IMATI-GE/CNR

Shape Processing & Analysis with Implicit Functions

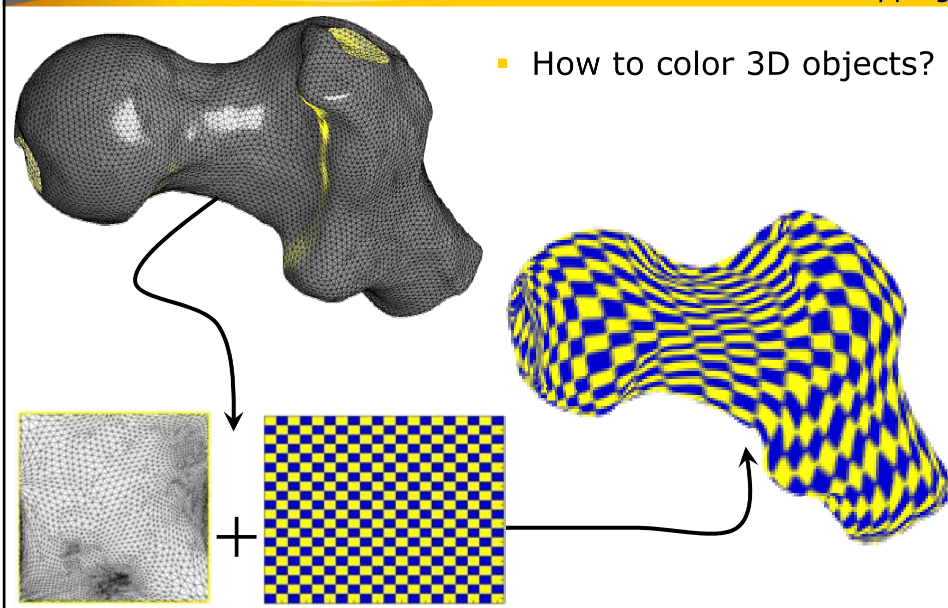
Remark & Motivations

- There is no single representation of surfaces that satisfies the needs of every problem in every application area.
  - Simple problems/queries such as genus evaluation and point location might be simpler if we use specific representations of the input surface (e.g., discrete vs implicit representations).
- Applications:
  - surface approximation;
  - texture mapping;
  - compression;
  - animation;
  - ….



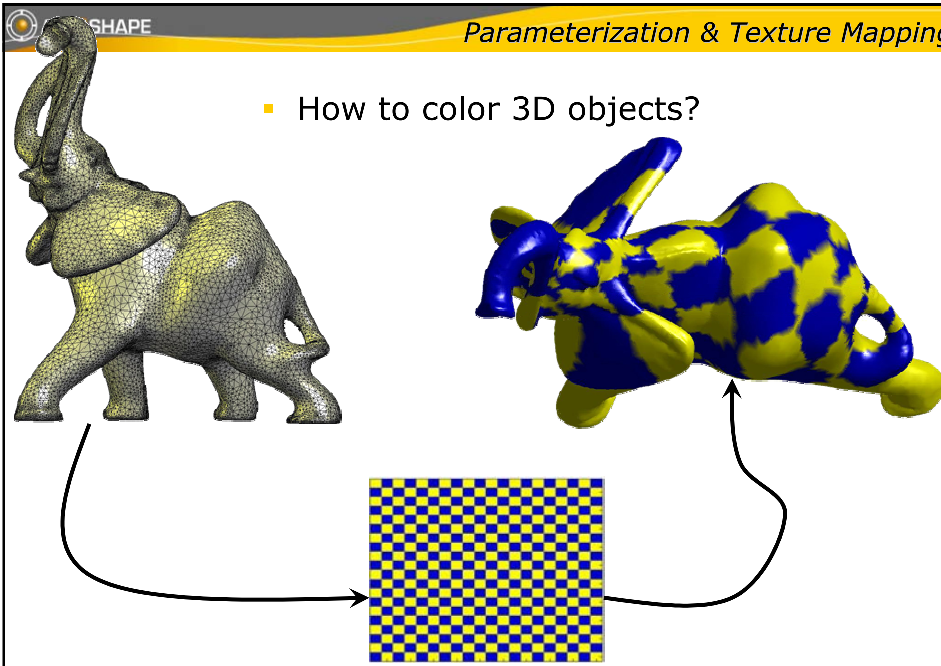
Parameterization & Texture Mapping

- How to color 3D objects?



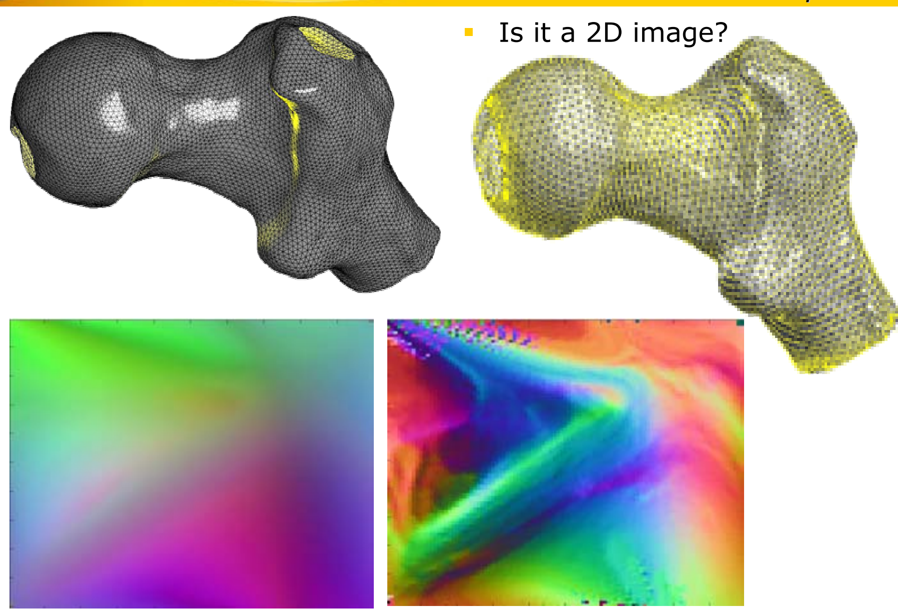
Parameterization & Texture Mapping

- How to color 3D objects?

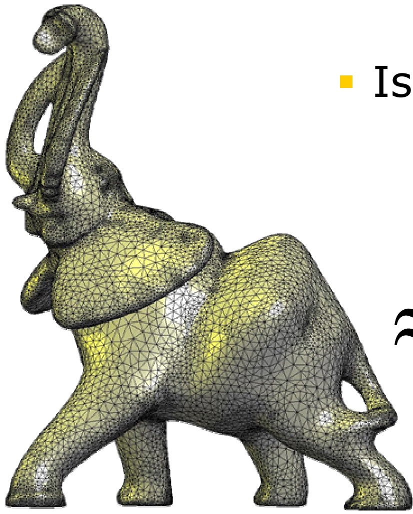


Parameterization & Compression

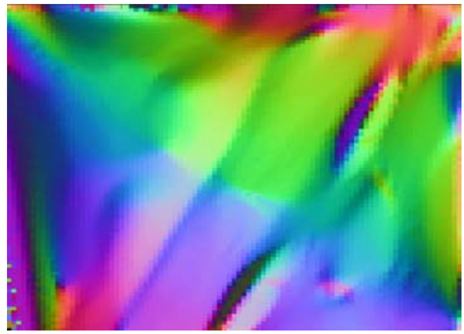
- Is it a 2D image?



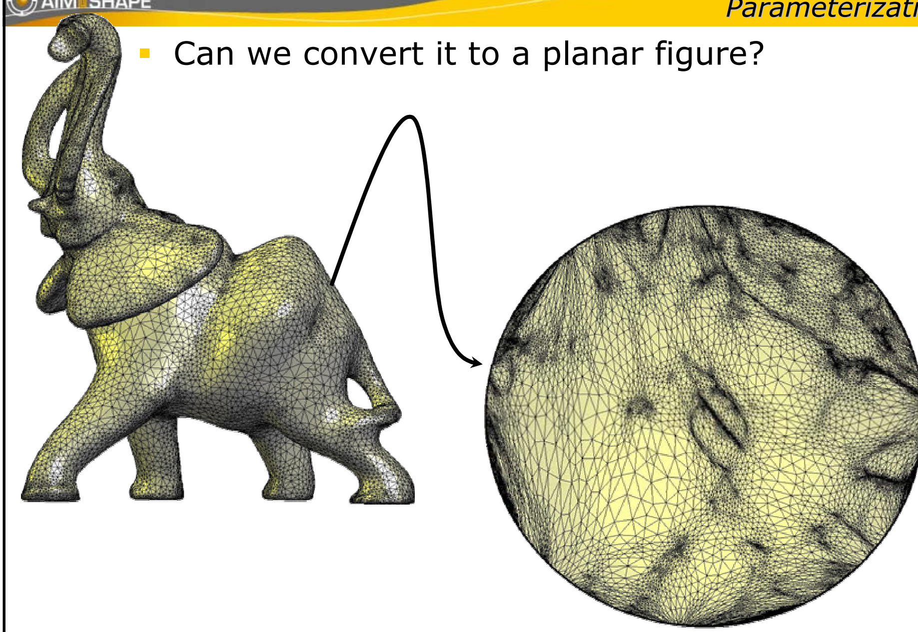
- Is it a 2D image?



≈



- Can we convert it to a planar figure?

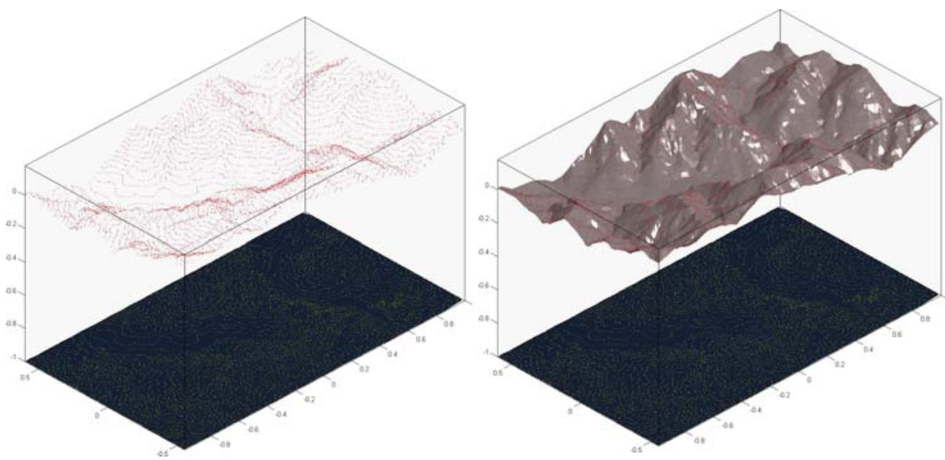


- Given a collection of k constrain points $\{p_1,\ldots,p_n\}$ that are scattered in 3D, together with scalar values at each of these points $\{h_1,\ldots,h_n\}$, construct a ***smooth surface that matches each constraint.***

$$f : R^3 \to R$$

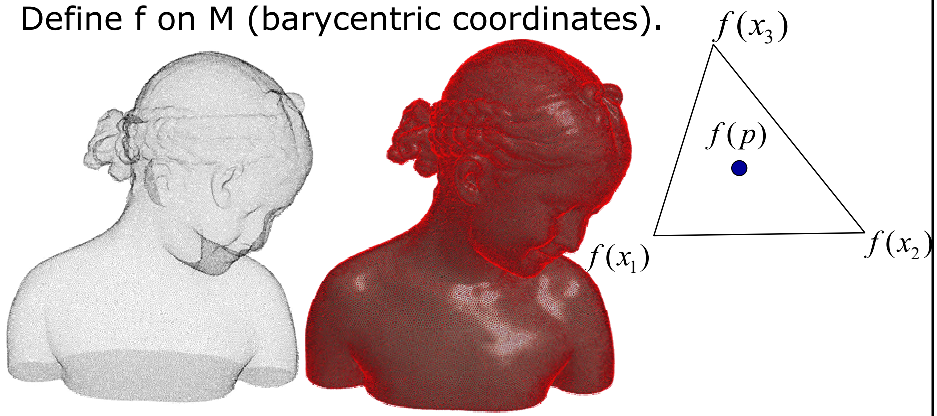
$$f(p_i) = h_i, i = 1,\ldots,n$$

$$\boxed{f \in C^k(R^3)} \qquad E(f) := \int_M \left[f_{xx}^2 + 2 f_{xy}^2 + f_{yy}^2 \right] dp$$



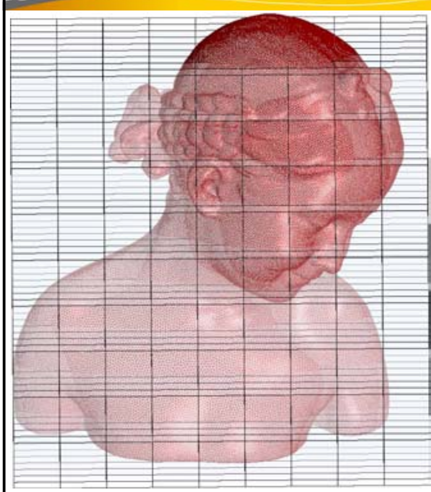
Piecewise linear approximation of M and f

- Triangulate the input dataset: (M,T)
- Define f on M (barycentric coordinates).

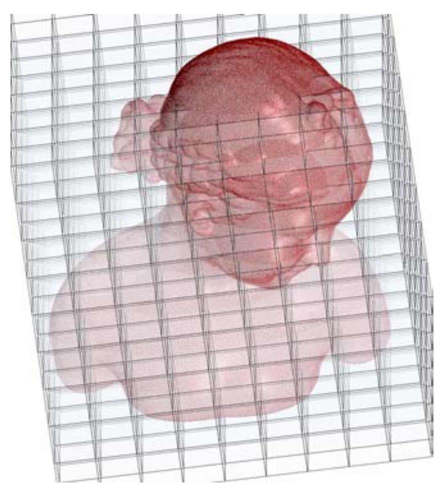


$$p = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 \Leftrightarrow f(p) = \lambda_1 f(x_1) + \lambda_2 f(x_2) + \lambda_3 f(x_3)$$
$$\lambda_i \geq 0, i = 1,2,3, \lambda_1 + \lambda_2 + \lambda_3 = 1$$

Volumetric approach



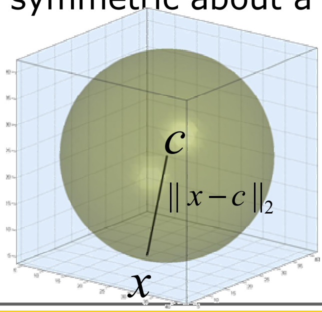
- Consider the volume "around" the input surface.



- Express the solution f in terms of Radial Basis Functions centered at the constrain locations.

$$f(p)=\sum_{i=1}^{n} a_i \varphi_i(p)$$

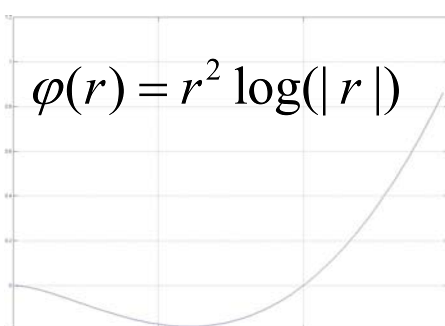
- Radial Basis Function: function which is radially symmetric about a single point (i.e., center).



$$\varphi_c(x) := \exp(-\| x-c \|_2)$$

- It is possible to choose the RBFs in such a way that they will automatically solve differential equations such as $$E(f) := \int_M \left[f_{xx}^2 + 2 f_{xy}^2 + f_{yy}^2 \right] dp$$ subject to constrains located at their centers, i.e.,

$$f(p_i)=h_i, i=1,\ldots,n$$



Bi-harmonic kernel

- Using the appropriate RBFs, we can write the interpolation function as:

$$f(p)=\sum_{i=1}^{n} a_i \varphi(p-p_i)+\boxed{P(x)}$$

unknowns

locations of the constraints

degree one polynomial

$$h_i = f(p_i)=\sum_{j=1}^{n} a_j \varphi(p_i-p_j)+P(p_i)$$

$$\varphi_{ij} := \varphi(p_i - p_j),\ p_i := (p_i^x, p_i^y, p_i^z)$$

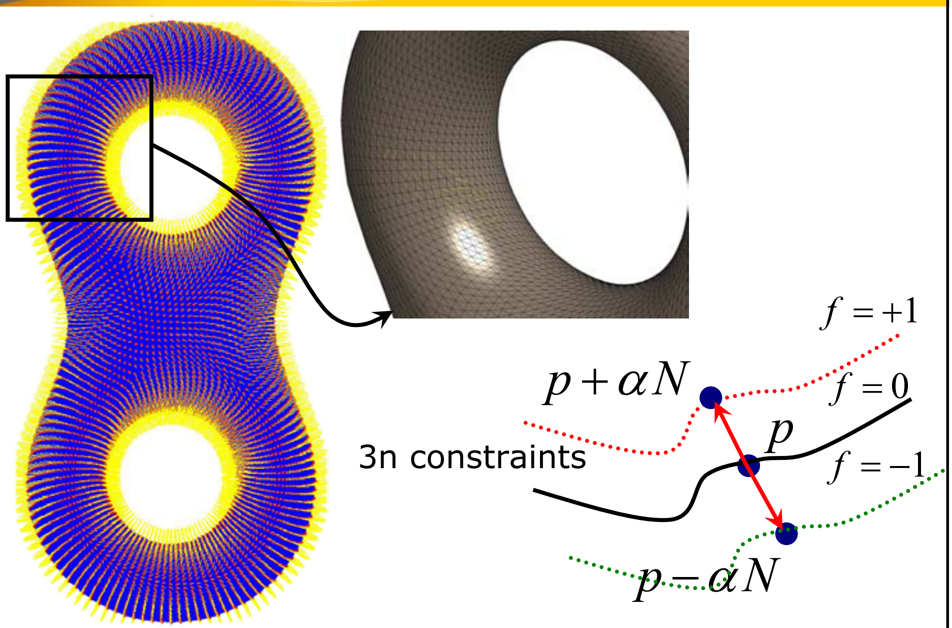
$$\begin{bmatrix} \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1n} & 1 & p_1^x & p_1^y & p_1^z \\ \varphi_{21} & \varphi_{22} & \cdots & \varphi_{2n} & 1 & p_2^x & p_2^y & p_2^z \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots \\ \varphi_{n1} & \varphi_{n2} & \cdots & \varphi_{nn} & 1 & p_n^x & p_n^y & p_n^z \\ 1 & 1 & \ldots & 1 & 0 & 0 & 0 & 0 \\ p_1^x & p_2^x & \ldots & p_n^x & 0 & 0 & 0 & 0 \\ p_1^y & p_2^y & \ldots & p_n^y & 0 & 0 & 0 & 0 \\ p_1^z & p_2^z & \cdots & p_n^z & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \\ a_{n+1} \\ a_{n+2} \\ a_{n+3} \\ a_{n+4} \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_n \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

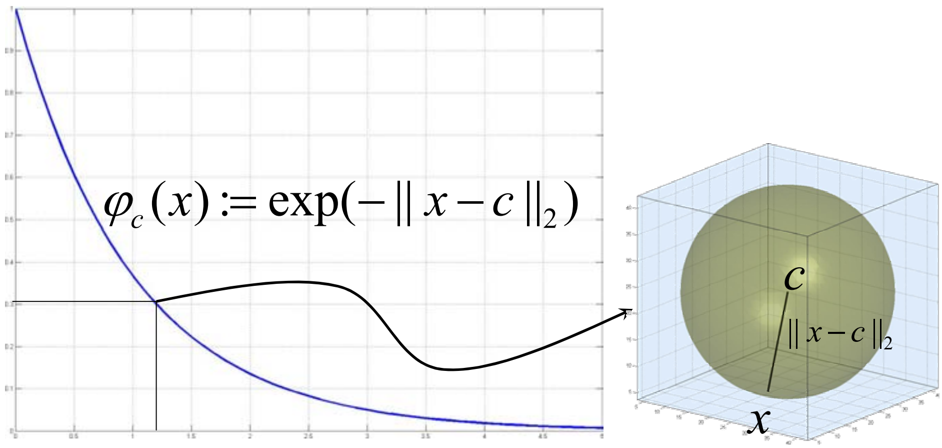
Computational cost and storage requirements

- Given n constraints:
  - Storage: full symmetric coefficient matrix $O(n^2)$;
  - Computational cost: direct solver $O(n^3)$;
  - n approx. 5.000 constraints represents a bottle neck for computation and storage.

$$
\begin{bmatrix}
\varphi_{11} & \varphi_{12} & \cdots & \varphi_{1n} & 1 & c_1^x & c_1^y & c_1^z \\
\varphi_{21} & \varphi_{22} & \cdots & \varphi_{2n} & 1 & c_2^x & c_2^y & c_2^z \\
\vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots \\
\varphi_{n1} & \varphi_{n2} & \cdots & \varphi_{nn} & 1 & c_n^x & c_n^y & c_n^z \\
1 & 1 & \ldots & 1 & 0 & 0 & 0 & 0 \\
c_1^x & c_2^x & \ldots & c_n^x & 0 & 0 & 0 & 0 \\
c_1^y & c_2^y & \ldots & c_n^y & 0 & 0 & 0 & 0 \\
c_1^z & c_2^z & \cdots & c_n^z & 0 & 0 & 0 & 0
\end{bmatrix}
$$

Constraint definitions



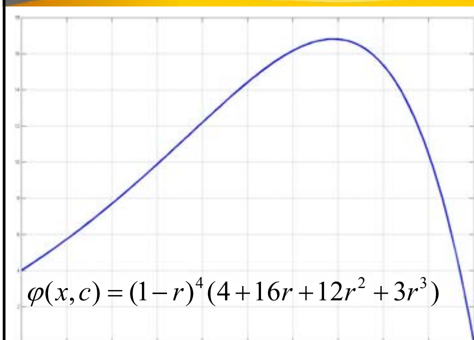


$$\varphi_c(x) := \exp(-\| x - c \|_2)$$

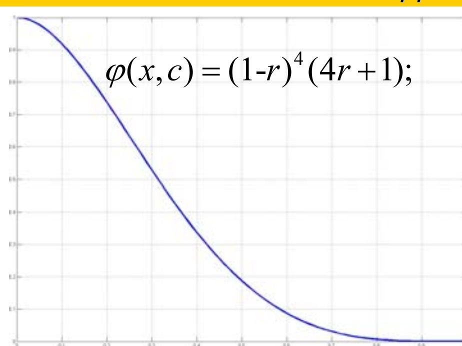
$$\| x - c \|_2$$

$$\sup(\varphi_c) = \mathbb{R}$$

AIM@SHAPE
Radial Basis Functions: Local vs Global Support



$$\varphi(x,c) = (1-r)^4 (4 + 16r + 12r^2 + 3r^3)$$



$$\varphi(x,c) = (1\text{-}r)^4 (4r + 1);$$

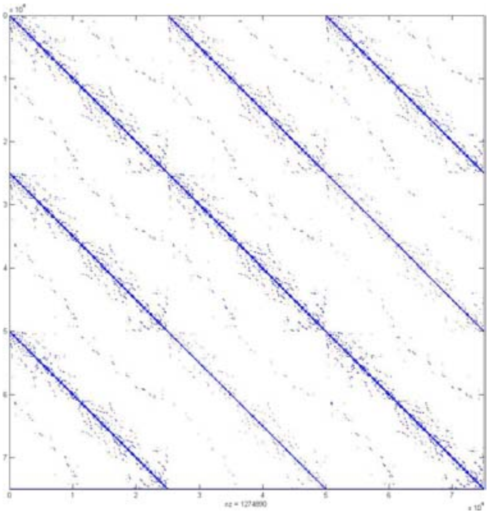
$$\varphi^*(x,c) = \begin{cases} \varphi(x,c) & = \| x - c \|_2 \leq R \\ 0 & \text{altrimenti} \end{cases} \qquad r := \| x - c \|_2$$

$$\sup(\varphi_c) = [0, R]$$

imati
Meeting Place and Date
ist

Compactly supported RBFs

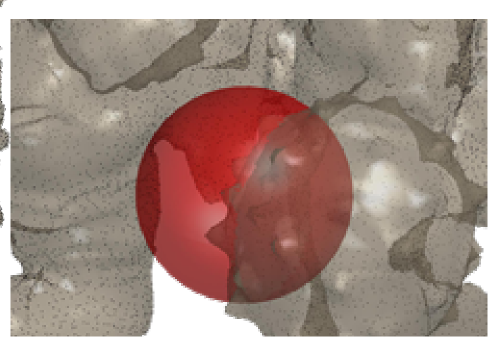
- The coefficient matrix becomes sparse → lower storage requirement and computational cost.



Kd-Tree Search

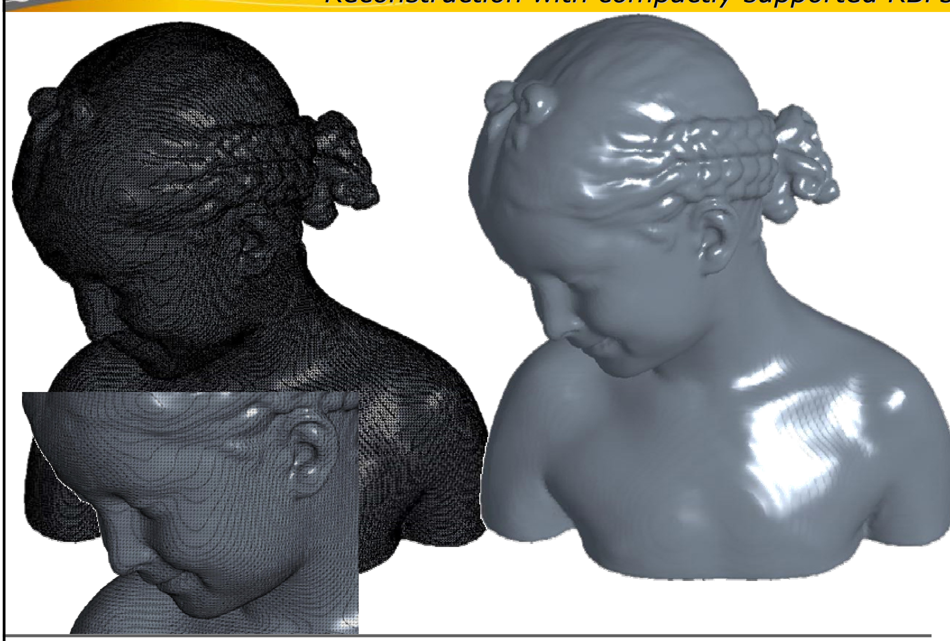
- We need a fast algorithm for point location: kd-Tree requires O(n log(n))-time.

$$x : \| x - c \|_2 \leq r$$

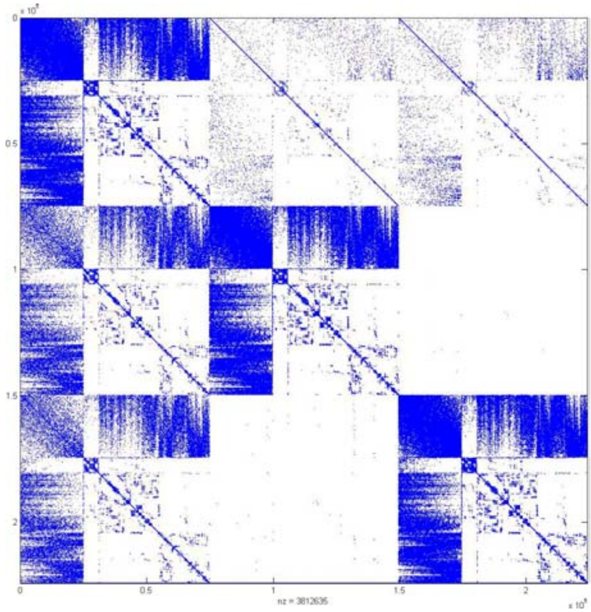


Reconstruction with compactly supported RBFs

Reconstruction with compactly supported RBFs



Reconstruction with compactly supported RBFs



Sparsification

- Given several approximations/representations (e.g., over-complete bases) of f

$$f(p) = \sum_{i=1}^{n} a_i \varphi_i(p)$$

We want to select a sub-basis such that

$$f^*(p) = \sum_{i \in I} a_i \varphi_i(p), I \subseteq \{1, \ldots, n\}$$

Is the best compromise between:

- Sparsity;
- Approximation accuracy;
- Smoothness;
- Feasible computational cost and storage.

Sparsification

- Main approaches:
  - (I) Support Vector Machines (1995 - Vapnik/Cortes);
  - (II) Tikhonov Regularization (1977 – Poggio/Bertero);
  - It has been proven that (I) and (II) are EQUIVALENT (1998 – Girosi).

- Main ideas:
  - *Reproducing Kernel Hilbert Spaces*

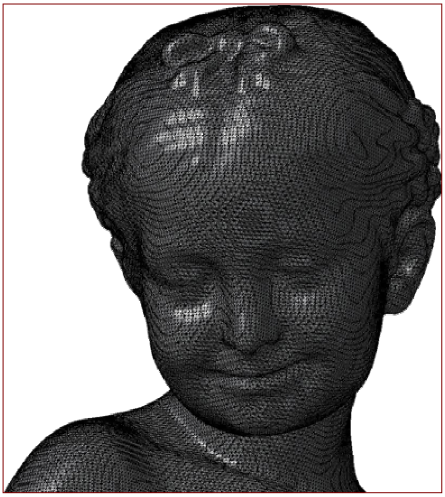
$$\min_{f^\star \in \mathcal{H}} \left\{ \frac{1}{2} \| f - \sum_{i=1}^{N} a_i K(x, x_i) \|^2_{\mathcal{H}} + \epsilon \|a\|_{l_1} \right\}$$

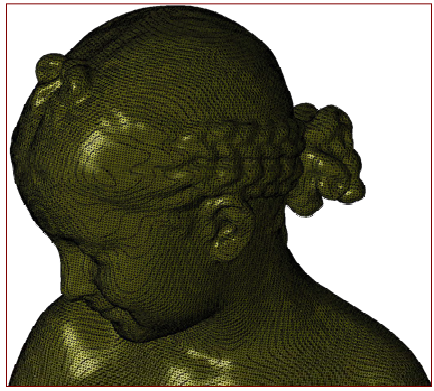
$$h(x) =< h(y), K(x,y) >_{\mathcal{H}}, \qquad \forall h, \qquad \forall x, y \in \mathbb{R}^d,$$

Sparsification examples

Sparsification examples

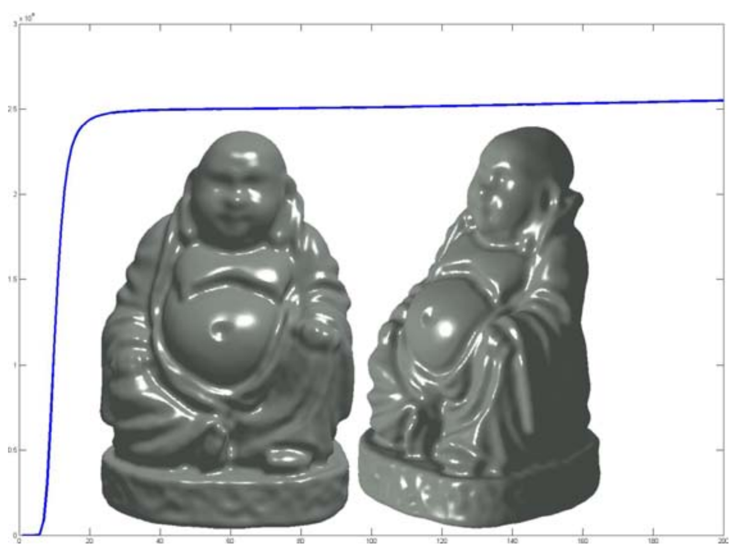
- 240K input centers → 62% selected centers





Sparsification examples

- Idea: iterative approach & multi-level sparsification.



Implicit modeling