Point Processing & Modeling

Sources cours de Jean-Marc Thiery :
https://perso.telecom-paristech.fr/jthiery/
Roi Pooran ETH

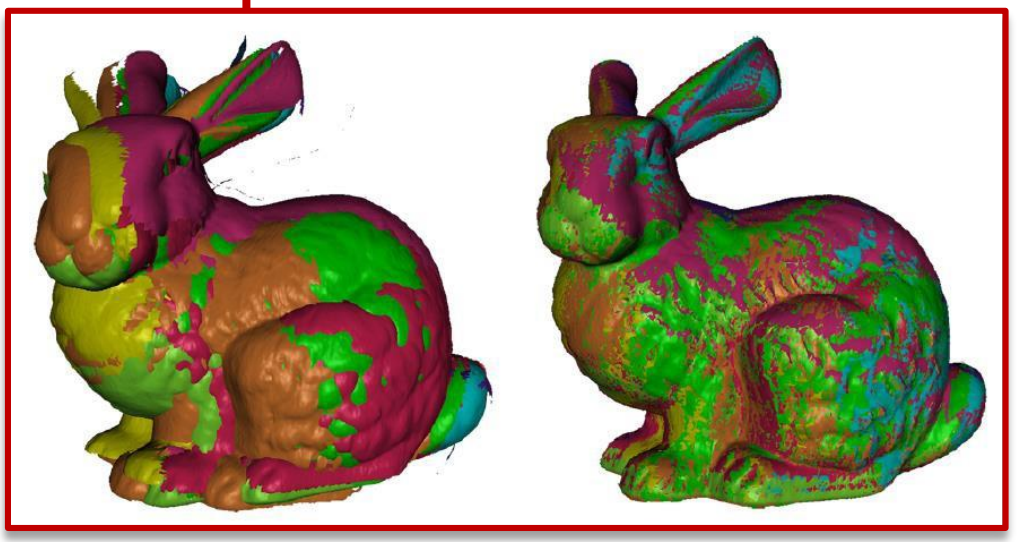
Geometry Acquisition Pipeline

Scanning:
results in range images

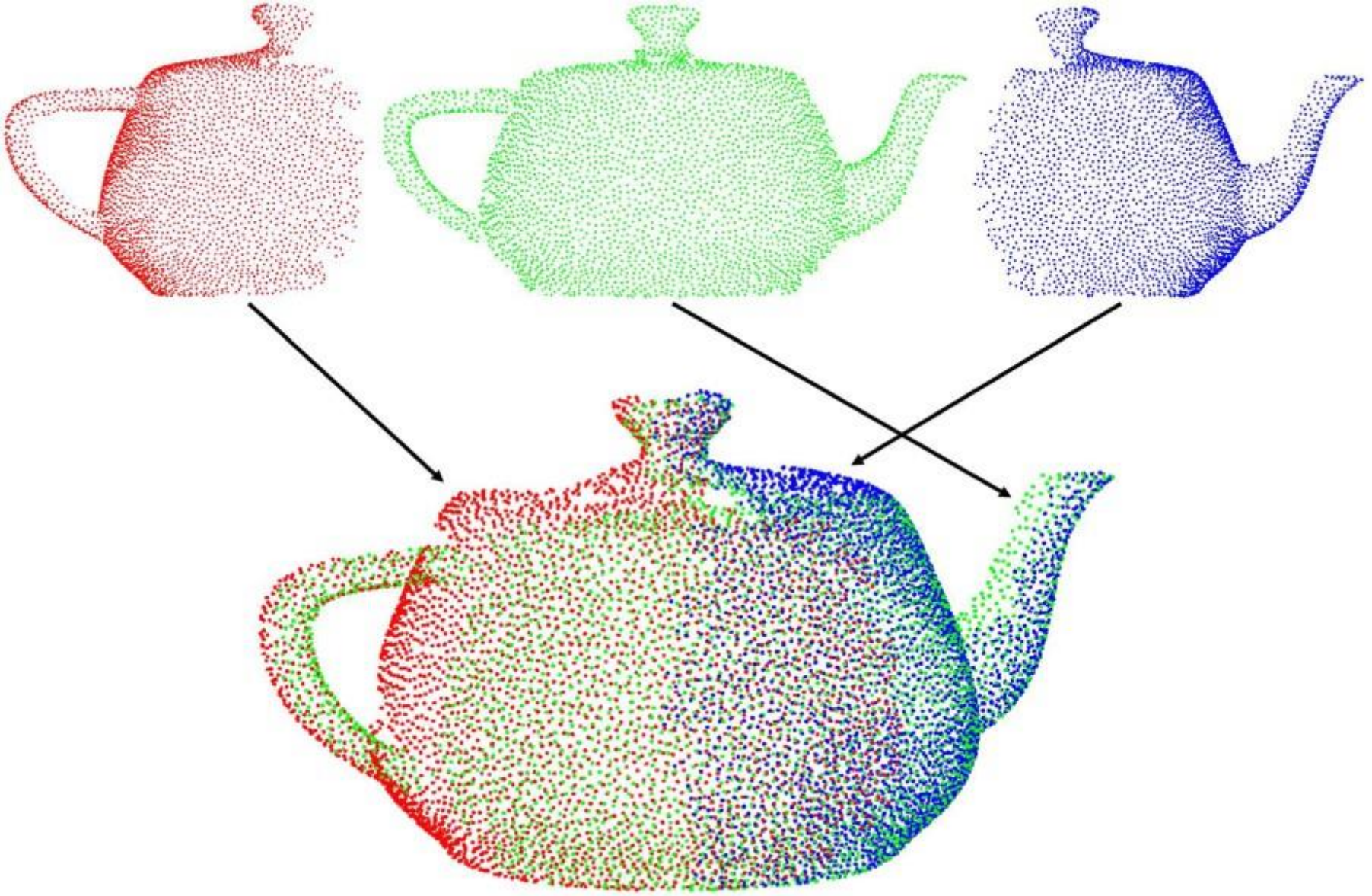
Registration:
bring all range images to one coordinate system

Reconstruction:
Integration of scans into a single mesh

Postprocess:
- Topological and geometric filtering
- Remeshing
- Compression

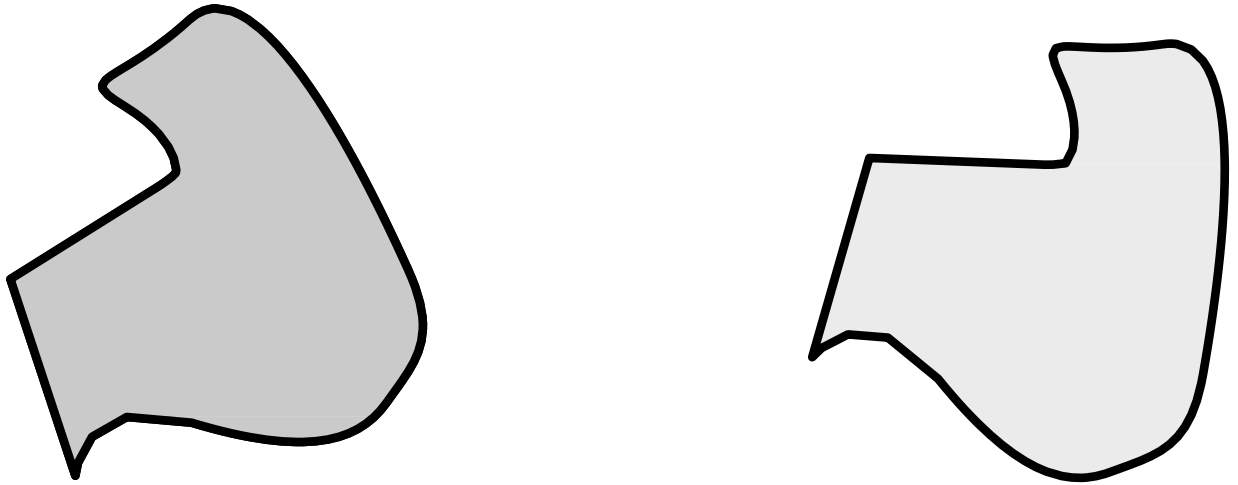


Pourquoi ?



Problem Statement

M_1 M_2

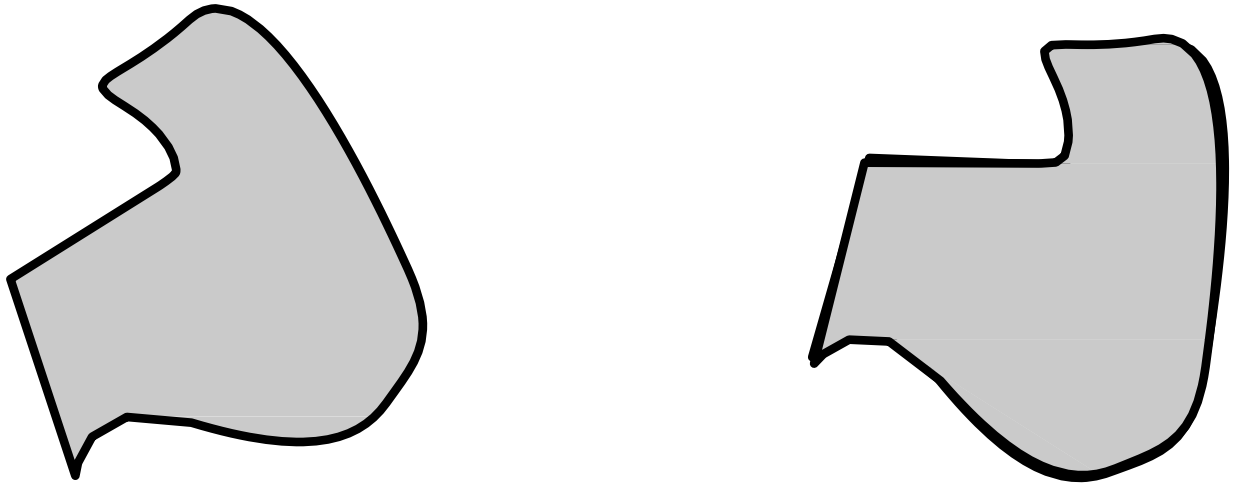


$$M_1 \approx T(M_2)$$

T: Translation + Rotation

Problem Statement

M_1 M_2

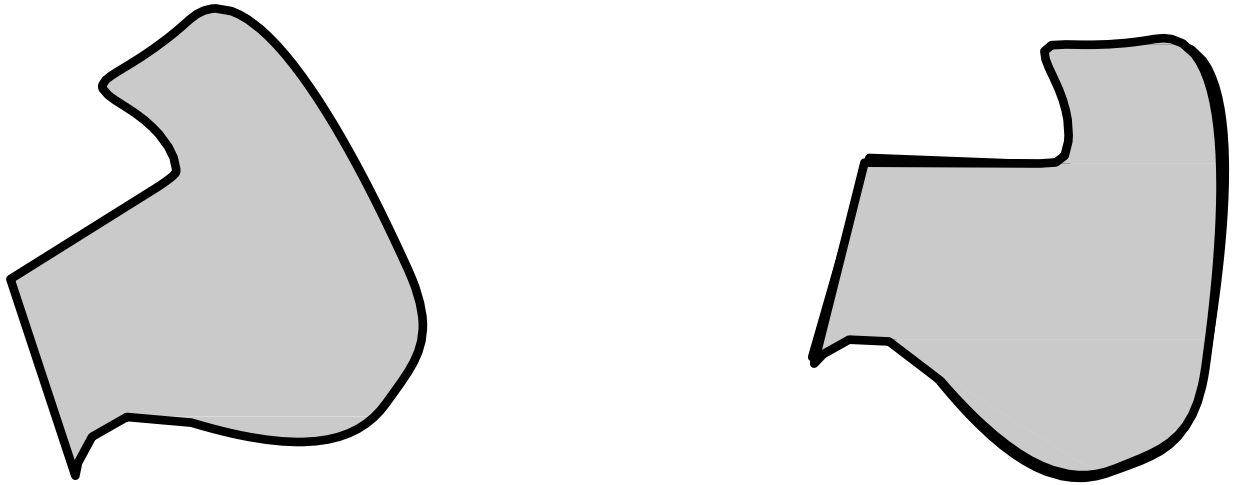


$$M_1 \approx T(M_2)$$

T: Translation + Rotation

Problem Statement

M_1 M_2



Given $M_1, \ldots, M_n$ find $T_2, \ldots, T_n$ such that

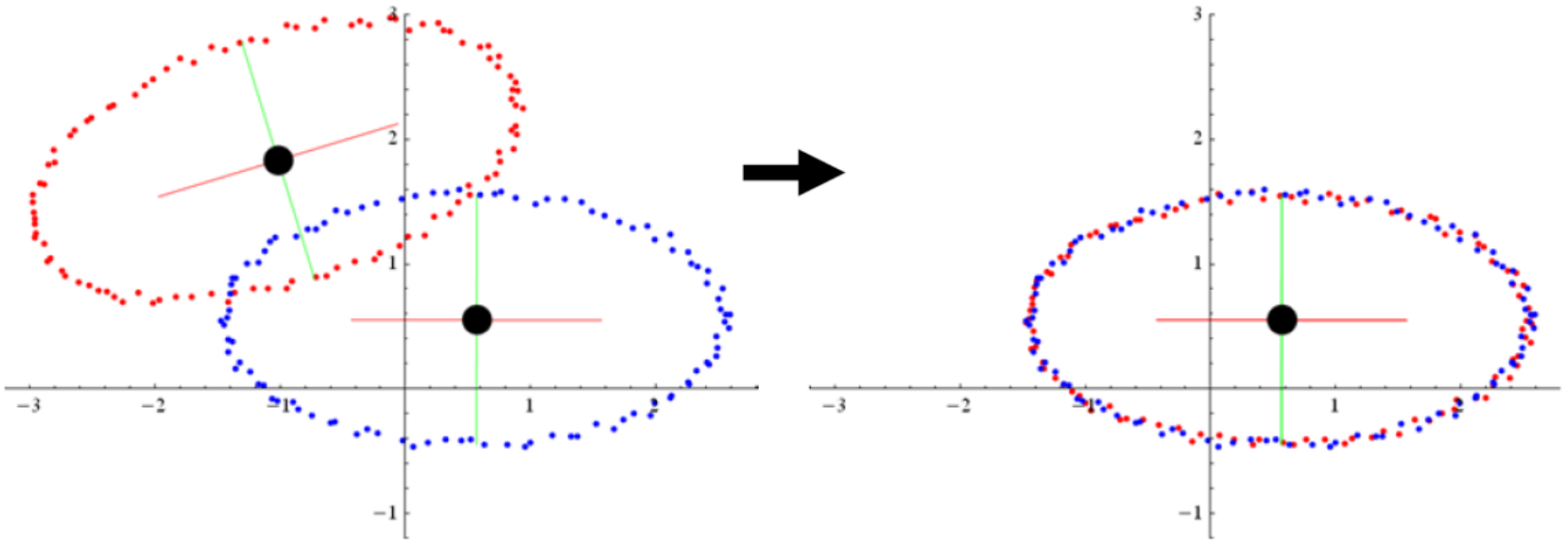
$$M_1 \approx T_2(M_2) \approx \cdots \approx T_n(M_n)$$

Methods

- Method 1: Principal component analysis (PCA)
  - Aligning principal directions
- Method 2: Singular value decomposition (SVD)
  - Optimal alignment given prior knowledge of correspondence
- Method 3: Iterative closest point (ICP)
  - An iterative SVD algorithm that computes correspondences as it goes

PCA

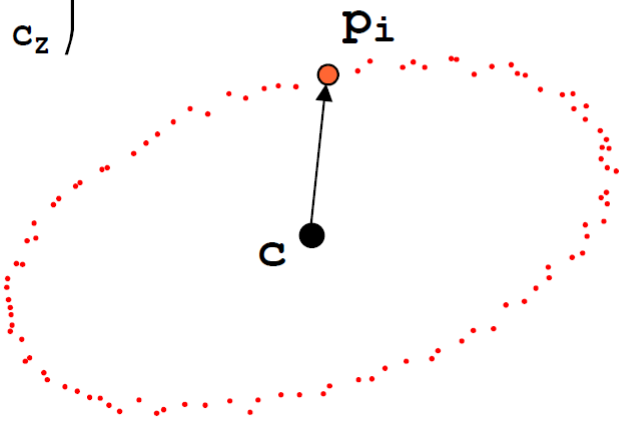
- Compute a shape-aware coordinate system for each model
  - Origin: Centroid of all points
  - Axes: Directions in which the model varies most or least
- Transform the source to align its origin/axes with the target



https://www.youtube.com/watch?v=uV5hmpzmWsU

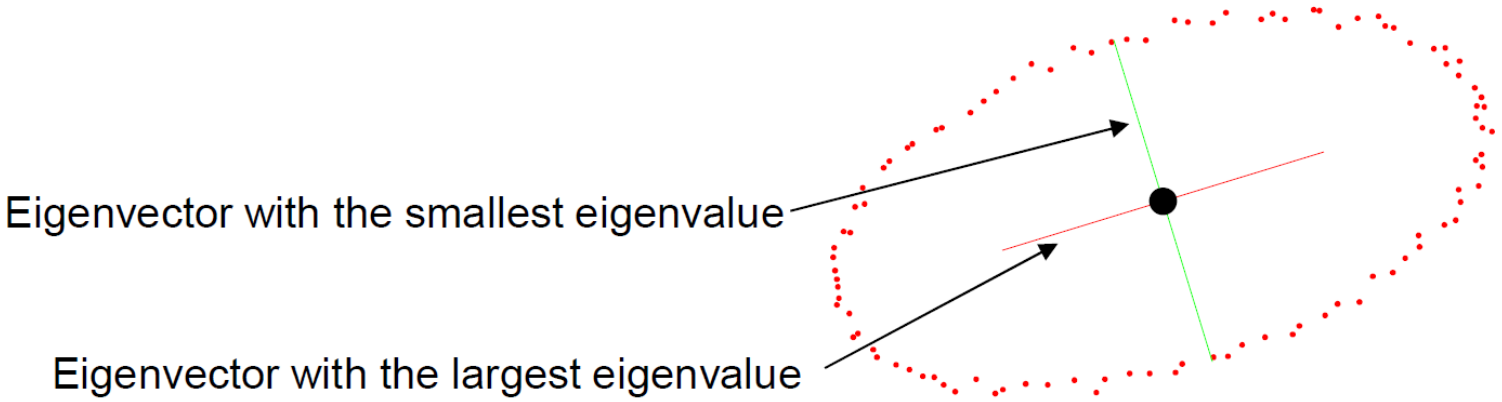
PCA

- Computing axes: Principal Component Analysis (PCA)
  - Consider a set of points $p_1,\ldots,p_n$ with centroid location c
    - Construct matrix P whose *i*-th column is vector $p_i - c$
      - 2D (2 by n): $P = \begin{pmatrix} p1_x - c_x & p2_x - c_x & \cdots & pn_x - c_x \\ p1_y - c_y & p2_y - c_y & \cdots & pn_y - c_y \end{pmatrix}$
      - 3D (3 by n): $P = \begin{pmatrix} p1_x - c_x & p2_x - c_x & \cdots & pn_x - c_x \\ p1_y - c_y & p2_y - c_y & \cdots & pn_y - c_y \\ p1_z - c_z & p2_z - c_z & \cdots & pn_z - c_z \end{pmatrix}$
    - Build the covariance matrix: $M = P \cdot P^T$
      - 2D: a 2 by 2 matrix
      - 3D: a 3 by 3 matrix

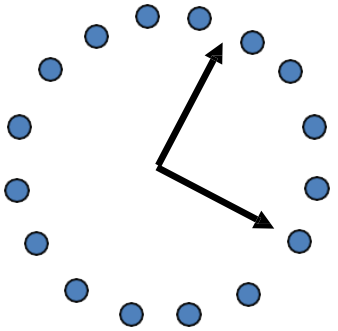


PCA

- Computing axes: Principal Component Analysis (PCA)
  - Eigenvectors of the covariance matrix represent principal directions of shape variation
    - The eigenvectors are un-singed and orthogonal (2 in 2D; 3 in 3D)
  - Eigenvalues indicate amount of variation along each eigenvector
    - Eigenvector with largest (smallest) eigenvalue is the direction where the model shape varies the most (least)



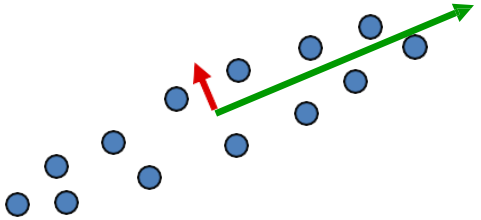
Principal Components



- There's no preferable direction
- S looks like this:

$$\mathbf{S} = \mathbf{V} \begin{pmatrix} \lambda & \\ & \lambda \end{pmatrix} \mathbf{V}^T$$

- Any vector is an eigenvector



- There's a clear preferable direction
- S looks like this:

$$\mathbf{S} = \mathbf{V} \begin{pmatrix} \lambda & \\ & \mu \end{pmatrix} \mathbf{V}^T$$

- μ is close to zero, much smaller than λ

PCA

- PCA-based alignment
  - Let c_S,c_T be centroids of source and target.
  - First, translate source to align c_S with c_T:

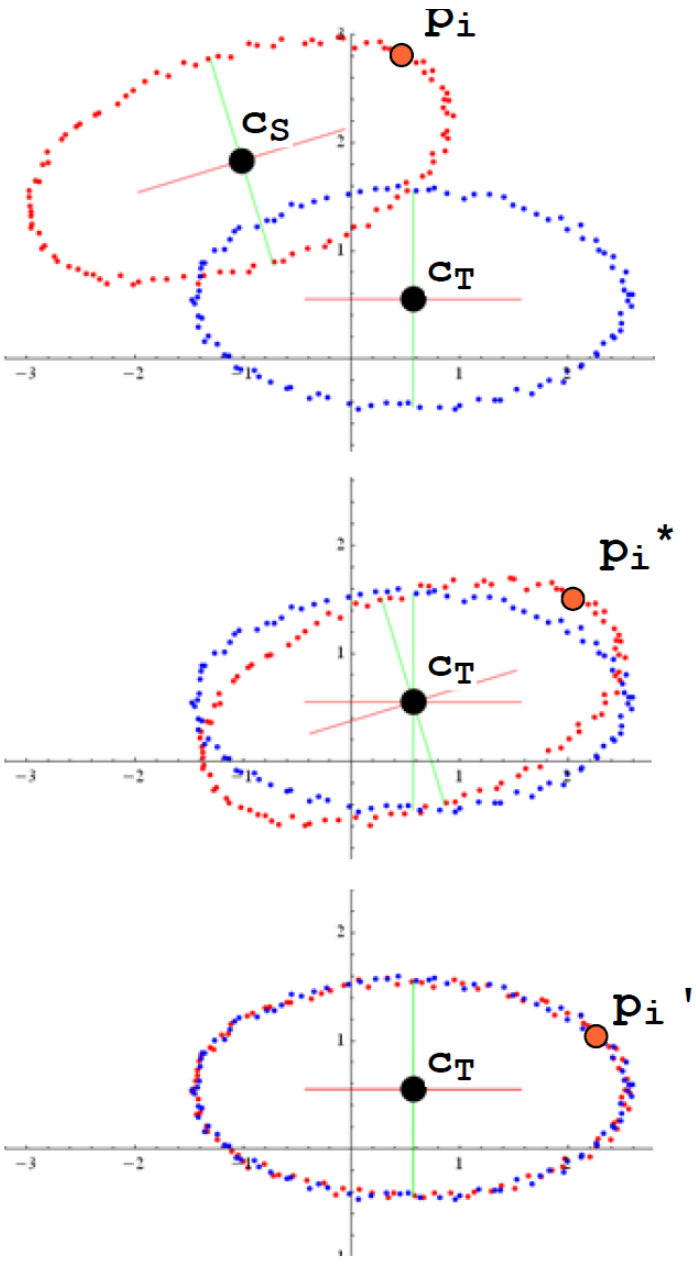
$$p_i^* = p_i + (c_T - c_S)$$

  - Next, find rotation R that aligns two sets of PCA axes, and rotate source around c_T:

$$p_i' = c_T + R \cdot (p_i^* - c_T)$$

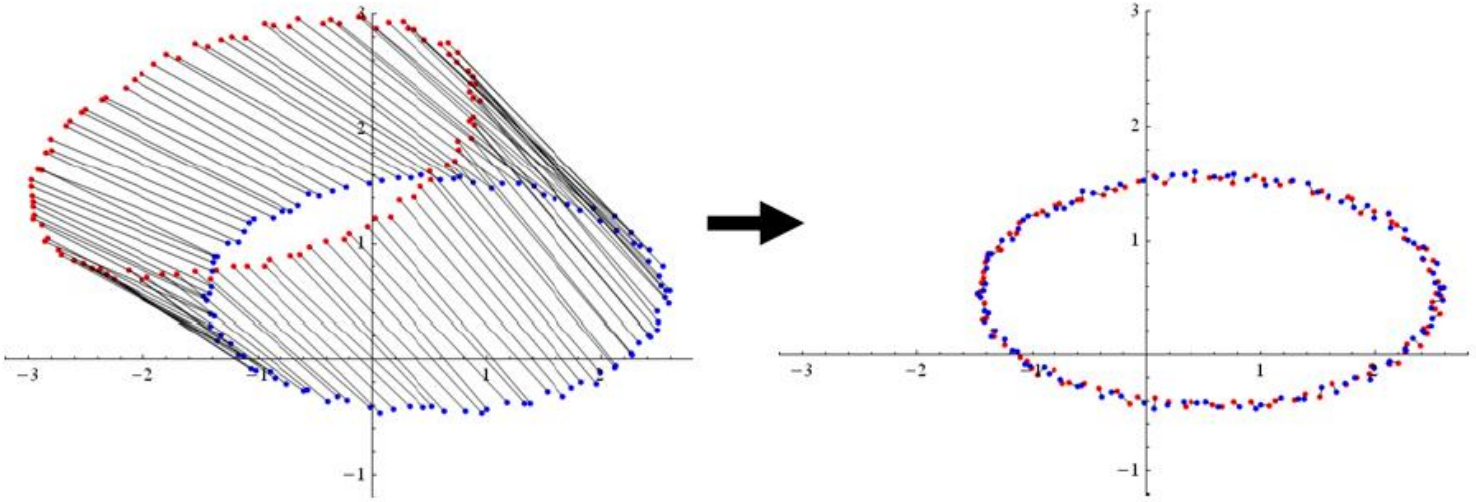
  - Combined:

$$p_i' = c_T + R \cdot (p_i - c_S)$$



SVD

Optimal si les deux pointsets sont rigoureusement les mêmes...



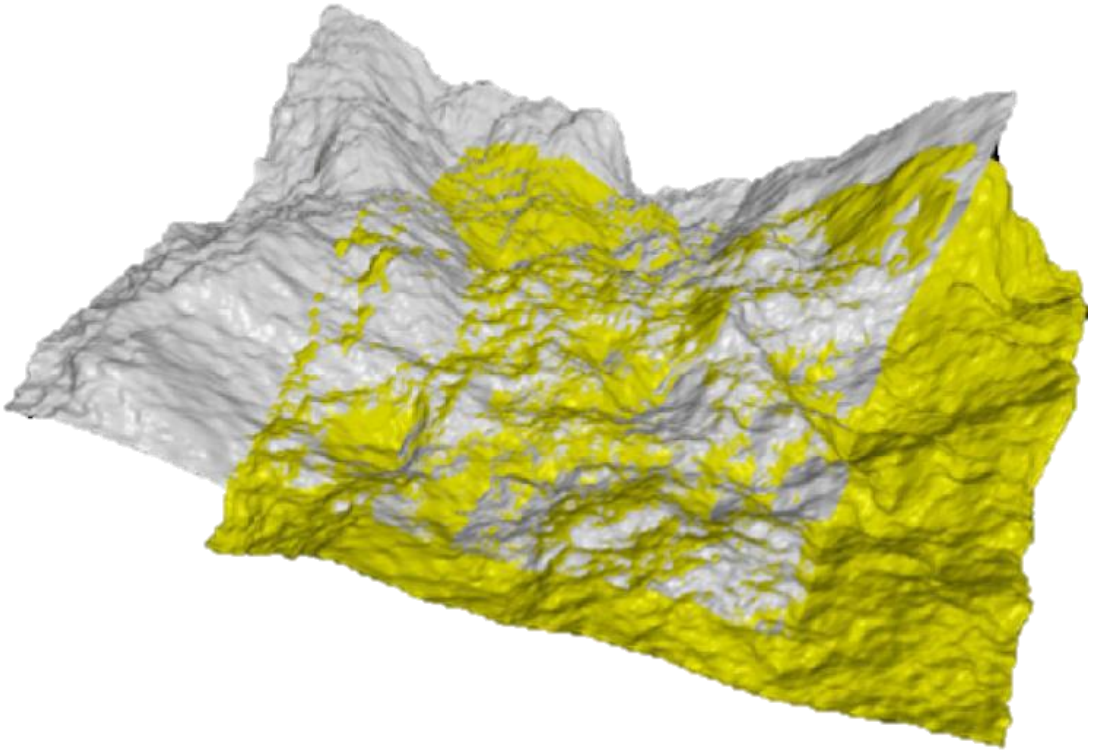
Correspondances

Combien de points sont nécessaires pour trouver la bonne transformation ?

Le premier problème est de trouver les paires

$$\mathbf{p}_1 \rightarrow \mathbf{q}_1$$
$$\mathbf{p}_2 \rightarrow \mathbf{q}_2$$
$$\mathbf{p}_3 \rightarrow \mathbf{q}_3$$

$$R\mathbf{p}_i + t \approx \mathbf{q}_i$$



ICP: Iterative Closest Point

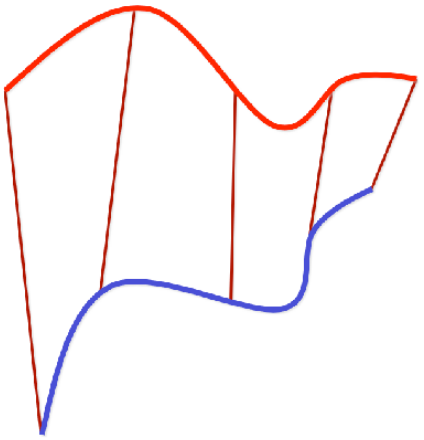
Intuition:

Correct correspondences ⇒ problem solved!

Idea:

(1) Find correspondences

(2) Use them to find a transformation



ICP: Iterative Closest Point

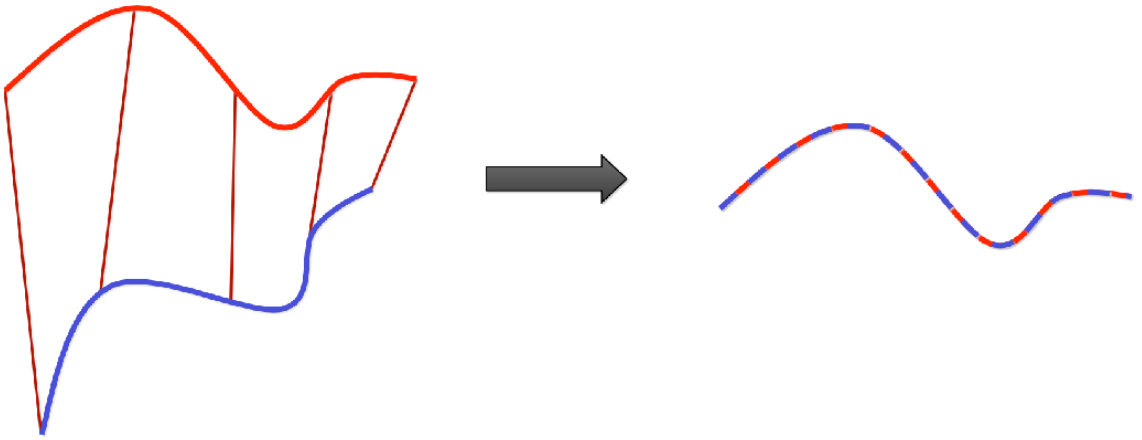
Intuition:

Correct correspondences ⇒ problem solved!

Idea:

(1) Find correspondences

(2) Use them to find a transformation



ICP: Iterative Closest Point

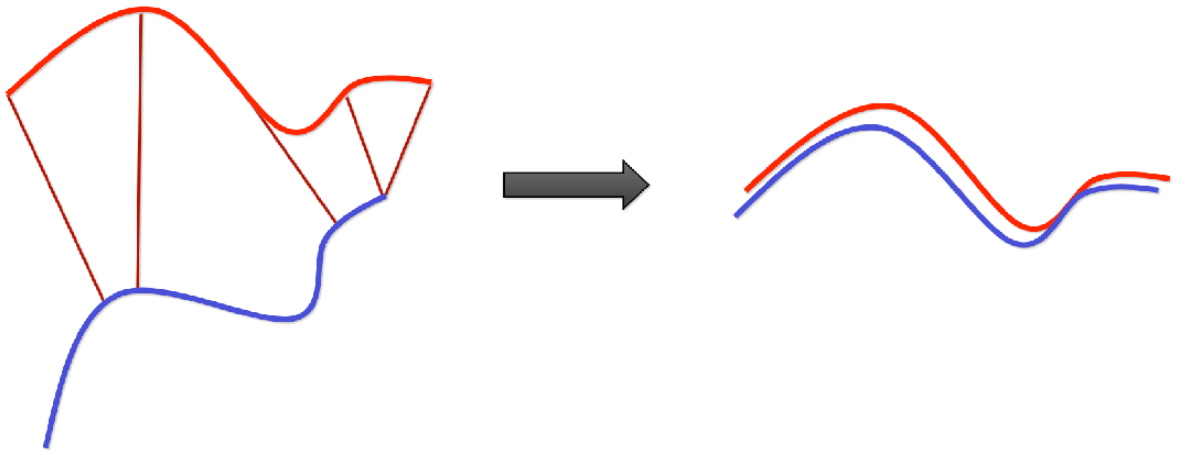
Intuition:

Correct correspondences ⇒ problem solved!

Idea:

(1) Find correspondences

(2) Use them to find a transformation



ICP: Iterative Closest Point

Intuition:

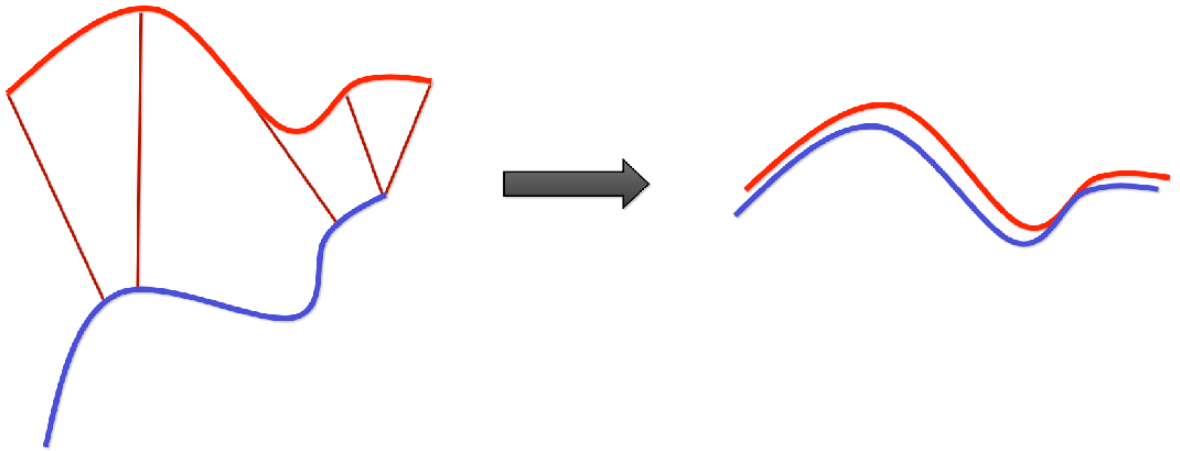
Correct correspondences ⇒ problem solved!

Idea:

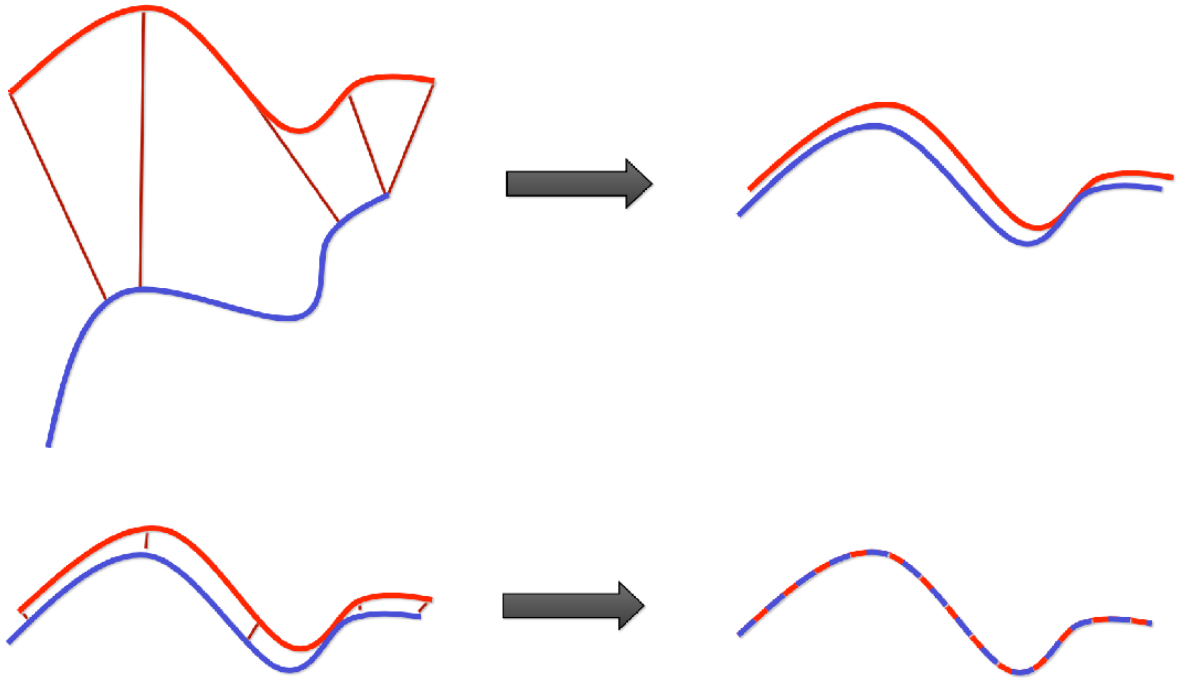
Iterate

(1) Find correspondences

(2) Use them to find a transformation



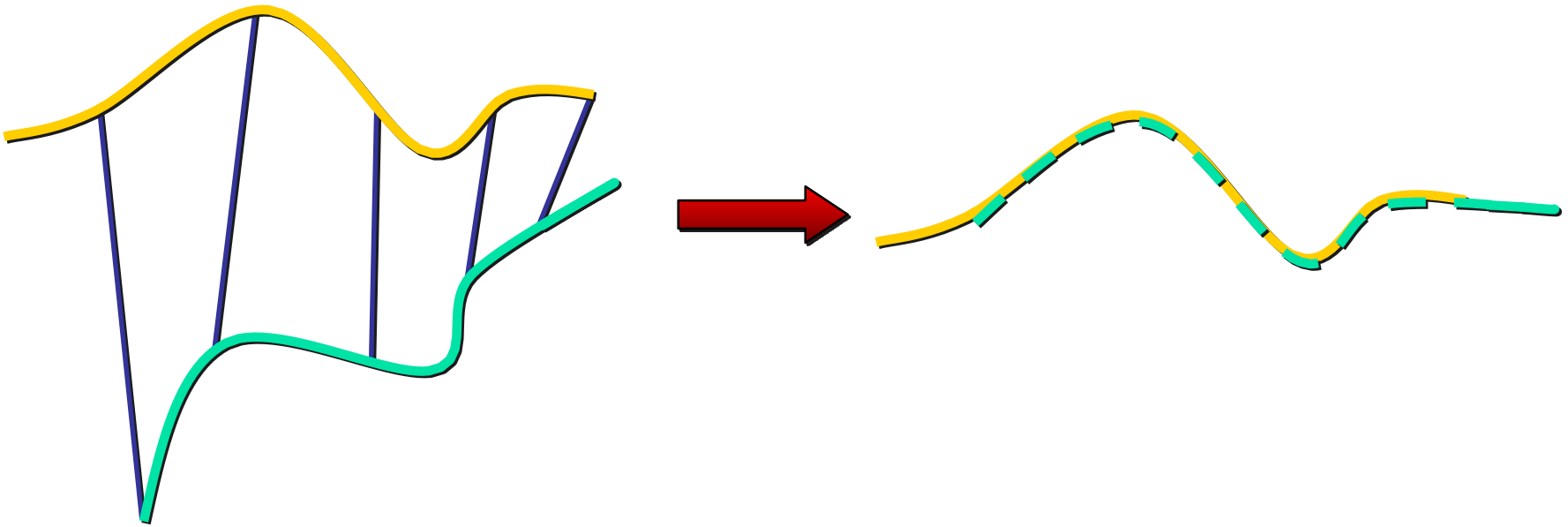
ICP: Iterative Closest Point



This algorithm converges to the correct solution
if the starting scans are “close enough”

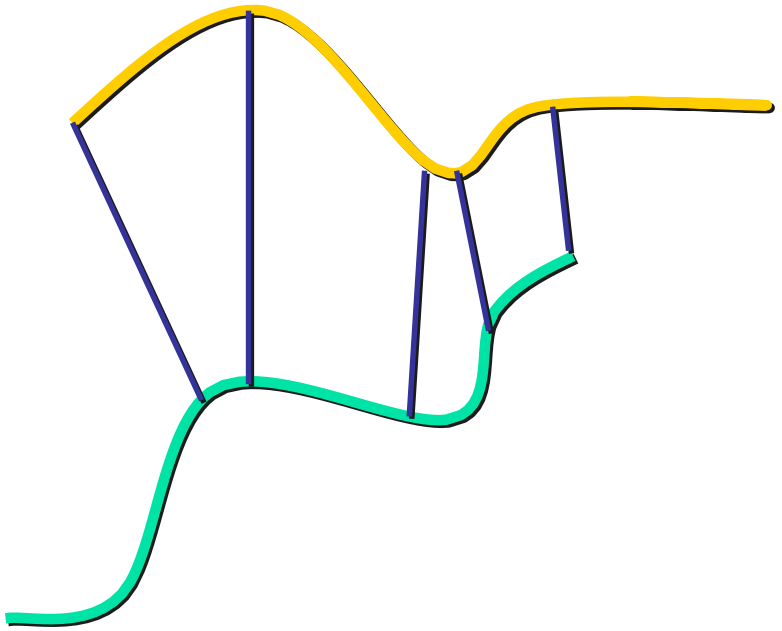
ICP : Iterative Closest Points

Si on connaît une correspondance entre deux pointsets, on peut les aligner (trouver une rotation et une translation pour aligner un sur l'autre).



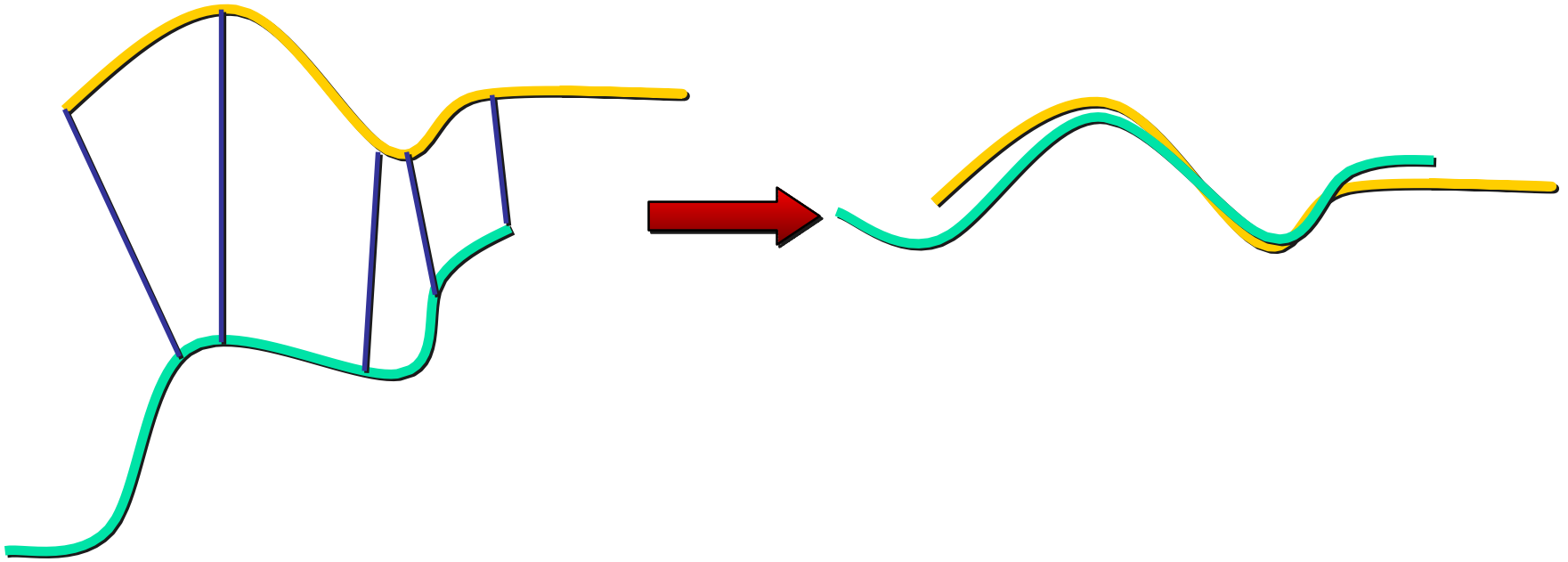
ICP : Iterative Closest Points

Si les deux pointsets sont relativement proches, alors on peut dire que le point le plus proche dans le pointset vert est un bonne correspondance avec un point dans le pointset jaune.



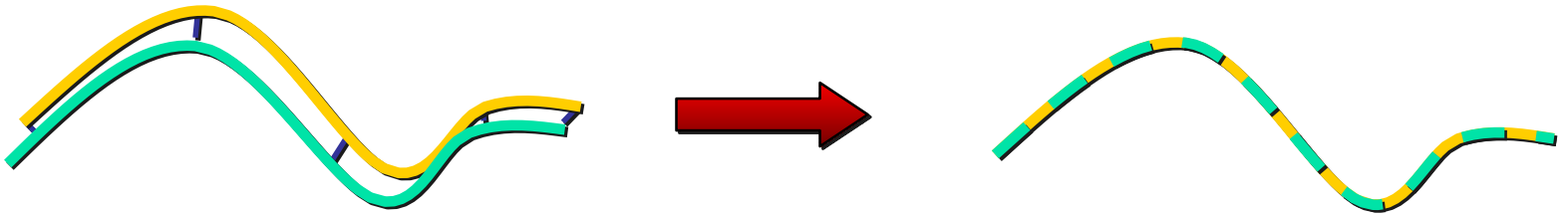
ICP : Iterative Closest Points

Ce ne sera pas parfait, mais l’alignement sera meilleur



ICP : Iterative Closest Points

Et en iterant, cela finit par converger (si il y a suffisamment d’overlap dans les deux pointsets, et que l’alignement initial n’est pas non plus trop mauvais)



Basic Algorithm

Select (e.g., 1000) random points

Match each to closest point on other scan

Reject pairs with distance too big

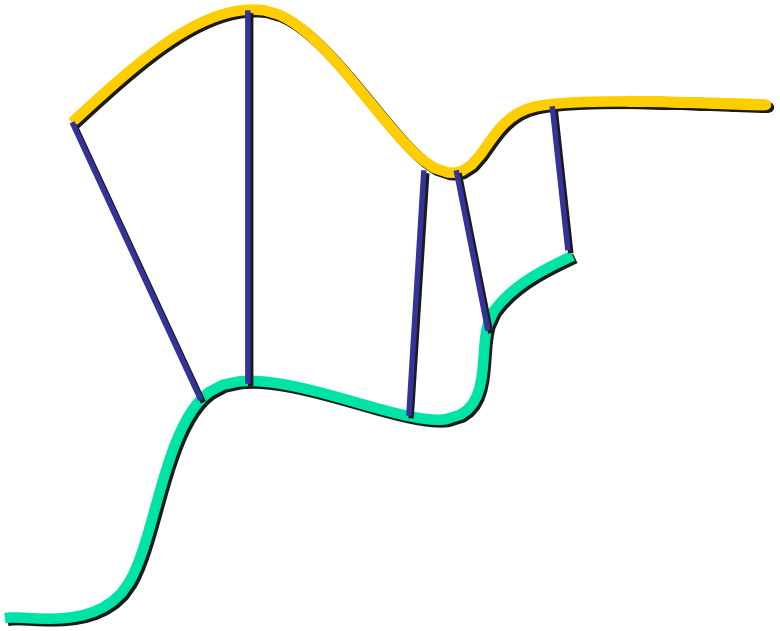
Minimize

$$E := \sum_i (R\mathbf{p}_i + t - \mathbf{q}_i)^2$$

closed form solution in:
http://dl.acm.org/citation.cfm?id=250160

Correspondance

1) Trouver, pour chaque point $p_i \in P$ le point le plus proche dans $q_i \in Q$ et enregistrer la correspondance : $\varphi(p_i) = q_i$

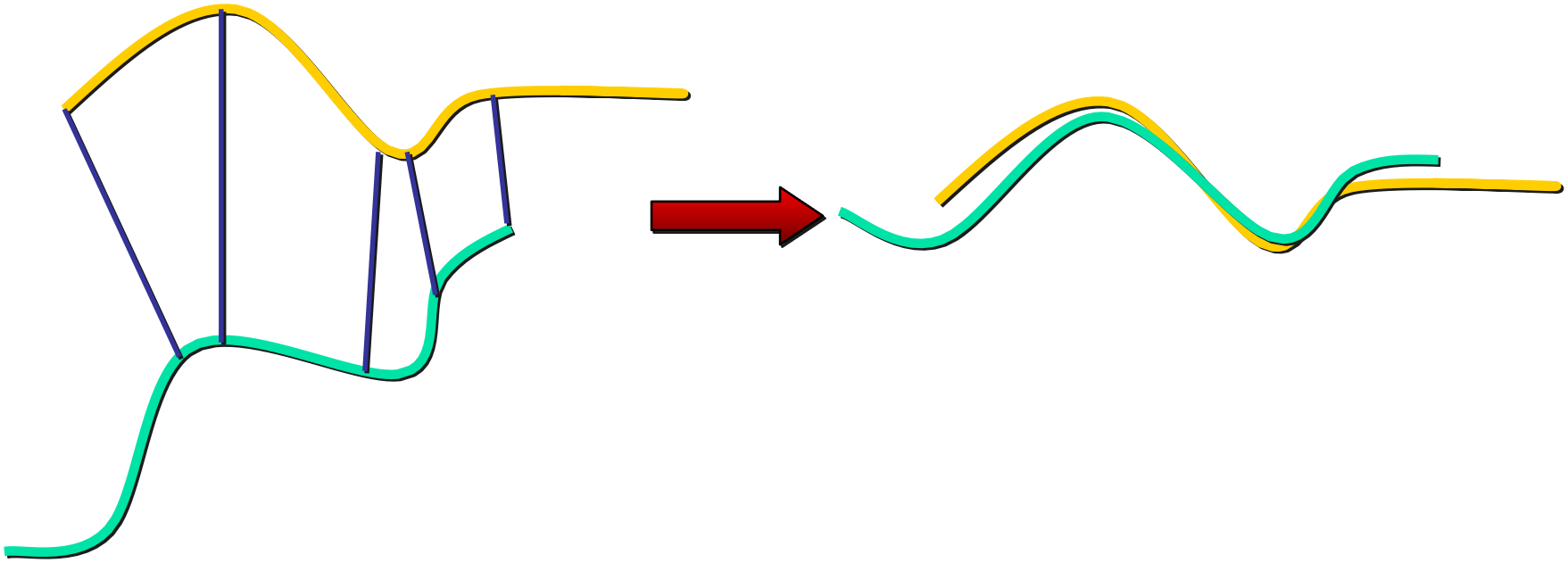


Trivial :
plus proche voisin (kd-tree par ex)

Alignement

2) Étant donnée une correspondance, trouver la transformation rigide (R,t) qui aligne au mieux les points correspondants

$$(R,t)=argmin \sum_i w_i \|R.p_i+t-q_i\|^2$$



Alignement

2) Étant donnée une correspondance, trouver la transformation rigide (R,t) qui aligne au mieux les points correspondants

$$(R,t)=argmin \sum_i w_i \|R.p_i+t-q_i\|^2$$

Théorème : la transformation optimale aligne les centroides.

$p=\sum_i w_i p_i / \sum_i w_i$

$q=\sum_i w_i q_i / \sum_i w_i$

centroides

$p_i'=p_i-p$

$q_i'=q_i-q$

recentrés

$\sum_i w_i p_i'=0$

$\sum_i w_i q_i'=0$

Alignement

2) Étant donnée une correspondance, trouver la transformation rigide (R,t) qui aligne au mieux les points correspondants

$$(R,t)=argmin \sum_i w_i \|R.p_i+t-q_i\|^2$$

Théorème : la transformation optimale aligne les centroides.

$$p=\sum_i w_i p_i / \sum_i w_i$$
$$q=\sum_i w_i q_i / \sum_i w_i$$

centroides

$$p_i'=p_i-p$$
$$q_i'=q_i-q$$

recentrés

$$\sum_i w_i p_i'=0$$
$$\sum_i w_i q_i'=0$$

$$\|R.p_i+t-q_i\|^2=\|(R.p_i'-q_i')+(R.p+t-q)\|^2$$
$$=\|(R.p_i'-q_i')\|^2+\|(R.p+t-q)\|^2+2(R.p_i'-q_i')^T.(R.p+t-q)$$

Alignement

2) Étant donnée une correspondance, trouver la transformation rigide (R,t) qui aligne au mieux les points correspondants

$$(R,t)=argmin \sum_i w_i \|R.p_i+t-q_i\|^2$$

Théorème : la transformation optimale aligne les centroides.

$$p=\sum_i w_i p_i / \sum_i w_i \qquad q=\sum_i w_i q_i / \sum_i w_i$$

centroides

$$p_i'=p_i-p \qquad q_i'=q_i-q$$

recentrés

$$\sum_i w_i p_i'=0 \qquad \sum_i w_i q_i'=0$$

$$\|R.p_i+t-q_i\|^2=\|(R.p_i'-q_i')+(R.p+t-q)\|^2$$
$$=\|(R.p_i'-q_i')\|^2+\|(R.p+t-q)\|^2+2(R.p_i'-q_i')^T.(R.p+t-q)$$

$$\sum_i w_i\|R.p_i+t-q_i\|^2=\sum_i w_i(\|(R.p_i'-q_i')\|^2+\|(R.p+t-q)\|^2)+2(R.(\sum_i w_i p_i')-(\sum_i w_i q_i'))^T.(R.p+t-q)$$
$$=\sum_i w_i(\|(R.p_i'-q_i')\|^2+\|(R.p+t-q)\|^2)$$

Alignement

$$(R,t)=argmin \sum_i w_i(\|(R.p_i'-q_i')\|^2+\|(R.p+t-q)\|^2)$$

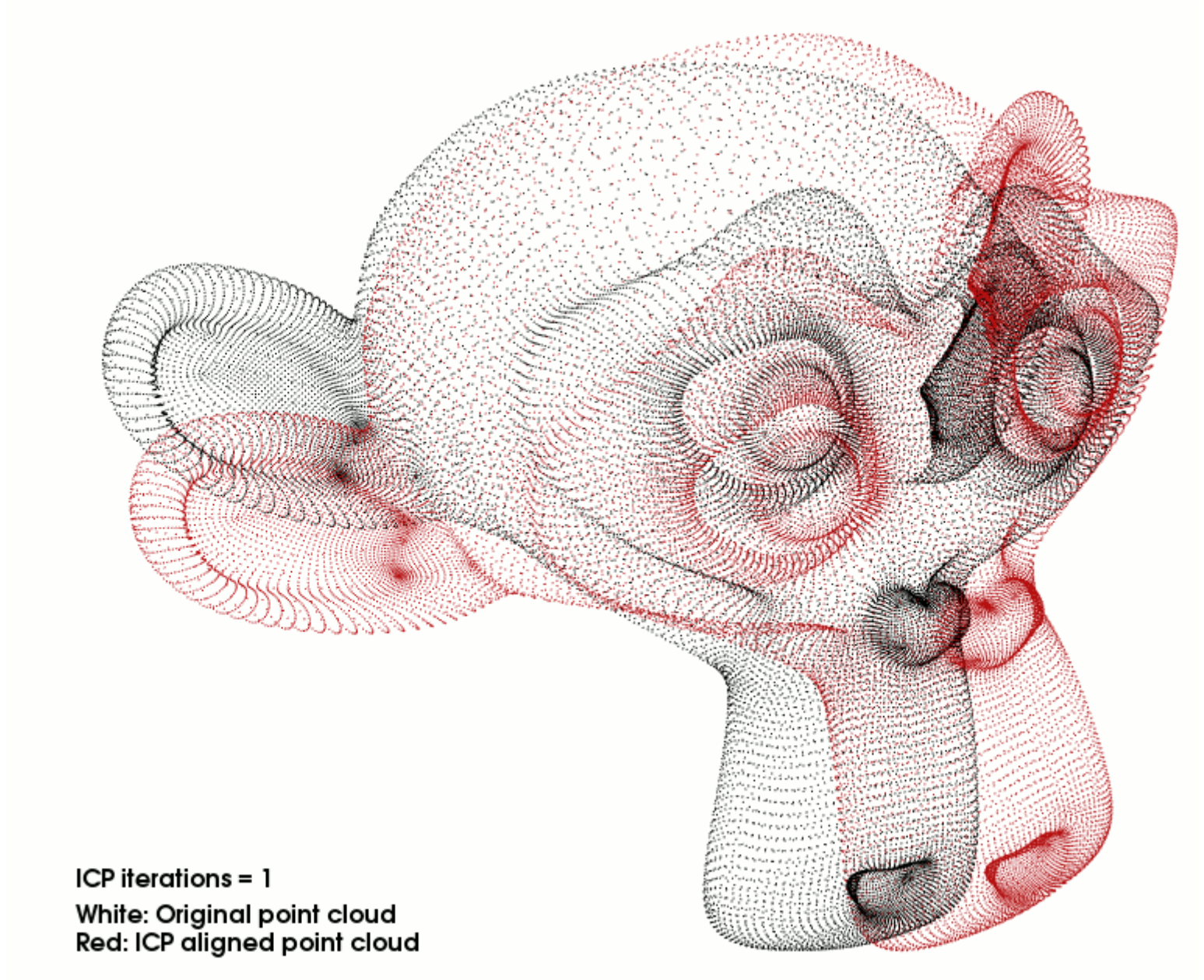
avec $p=\sum_i w_i p_i / \sum_i w_i$ $q=\sum_i w_i q_i / \sum_i w_i$ centroides, $p_i'=p_i-p$ $q_i'=q_i-q$ recentrés

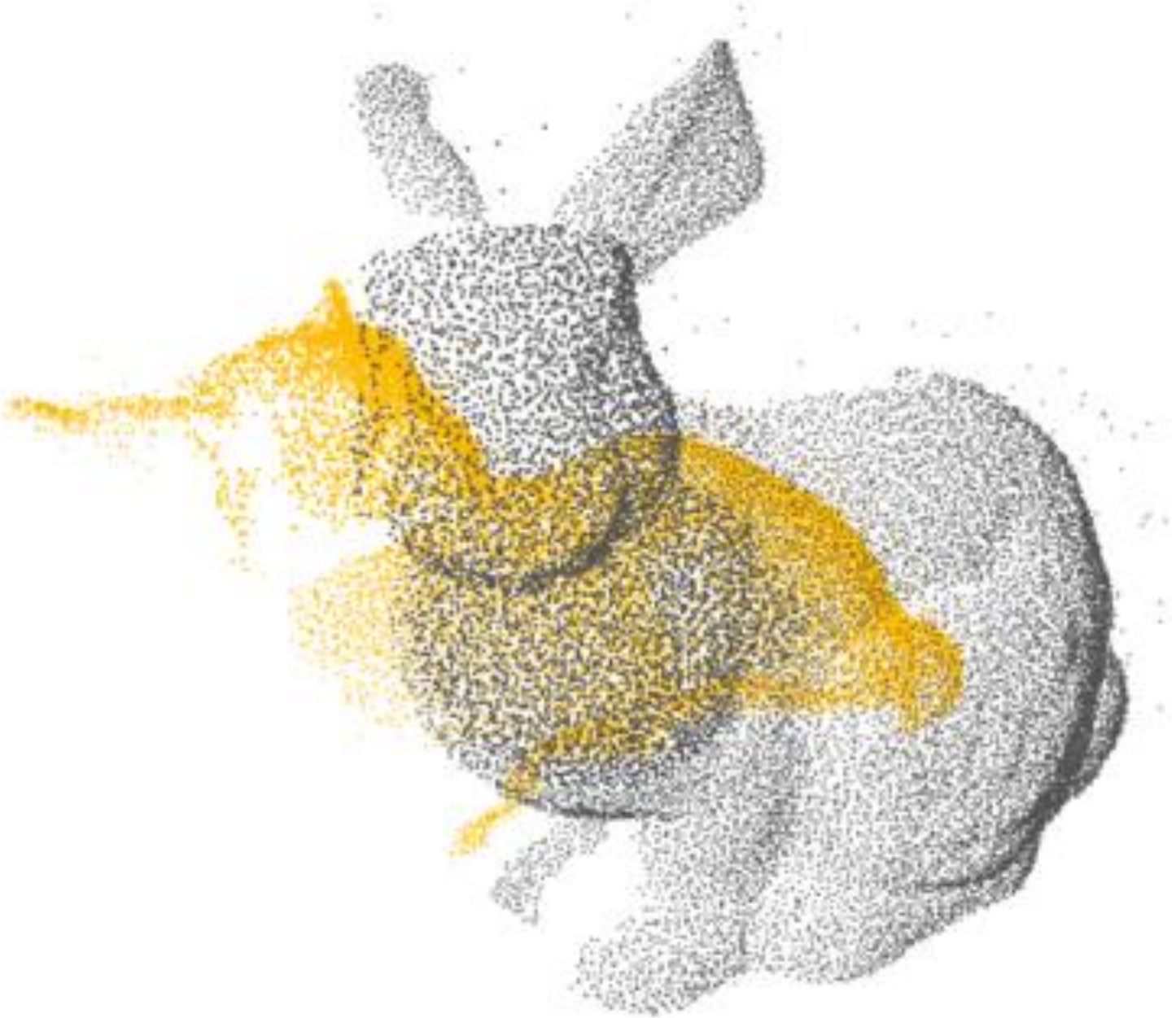
Procédure :

Trouver R qui minimise : $\sum_i w_i\|(R.p_i'-q_i')\|^2$ (Problème de Procustes)

$$\begin{cases} C=\sum_i w_i q_i'.p_i'^T \\ C=U.S.V^T \quad \text{(décomposition en valeurs singulières)} \\ R=U.diag(1,1,sign(U.V^T)).V^T \end{cases}$$

Puis t est donné par : $t=q-R.p$


ICP iterations = 1
White: Original point cloud
Red: ICP aligned point cloud

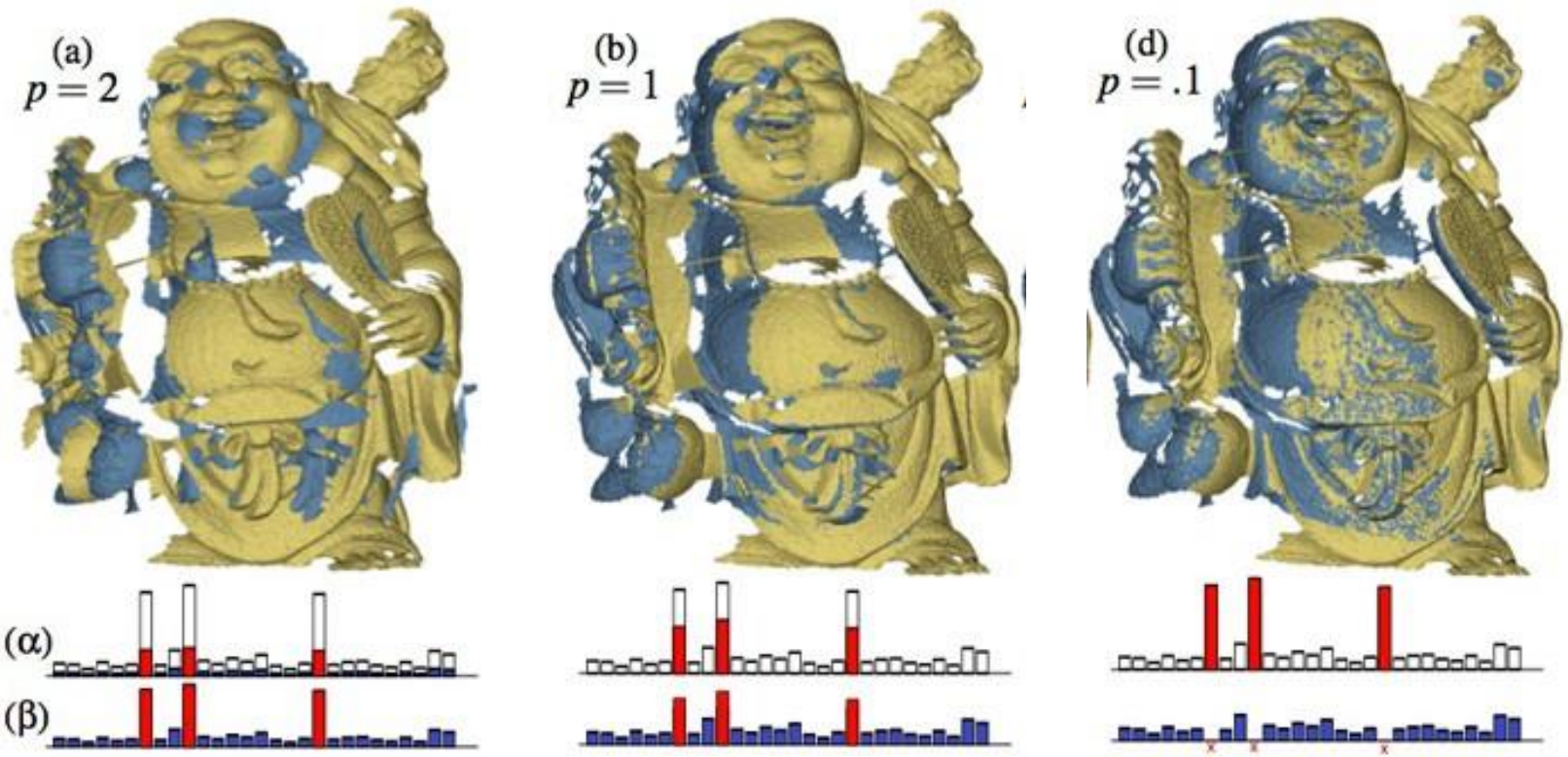


Variantes

$$(R,t)=argmin \sum_i w_i \|R.p_i+t-q_i\|^2$$

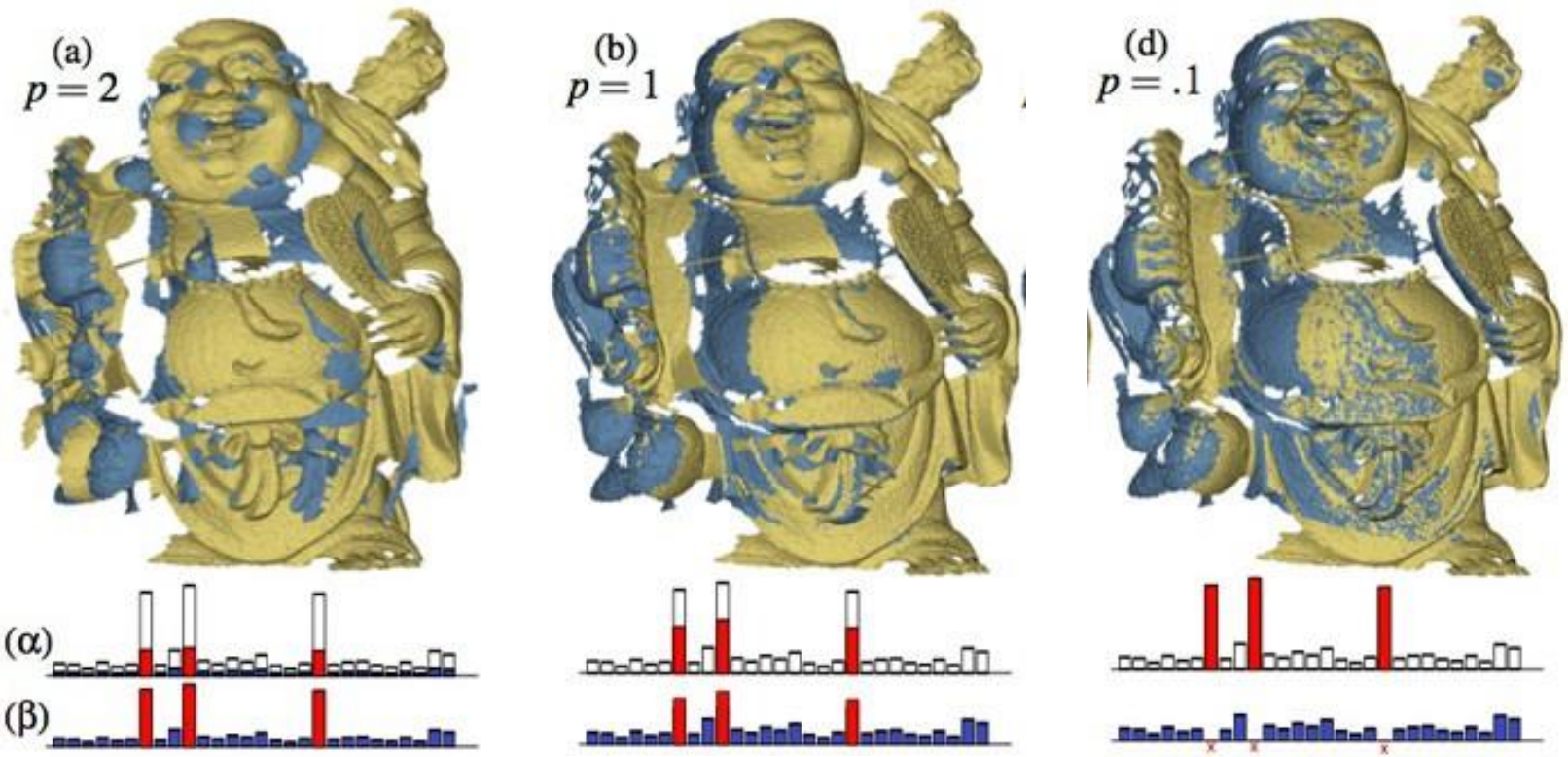
- Quels sont des poids pertinents ?
- Doit-on considérer tous les points de P ?
- Doit-on associer un point de Q nécessairement ?
- Peut-on aligner P sur Q ET Q sur P en même temps ?
- En présence d'outliers, peut-on ignorer certains points ?
  - RANSAC
  - Normes insensibles aux outliers : $(R,t)=argmin \sum_i w_i \|R.p_i+t-q_i\|^p$

Application : « Sparse » ICP



[Bouaziz et al. 2013] : Sparse Iterative Closest Point

Vidéo démo



https://www.youtube.com/watch?v=ii2vHBwlmo8

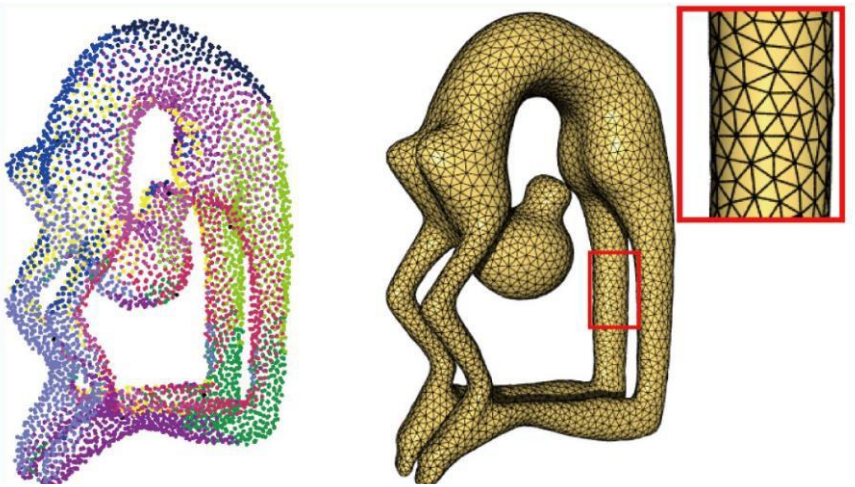
Geometry Acquisition Pipeline

Scanning:
results in range images

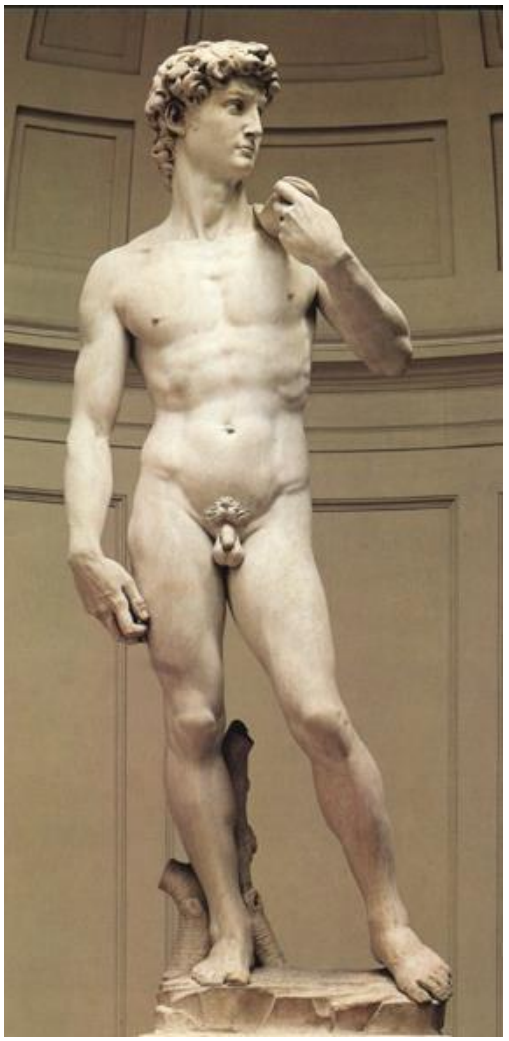
Registration:
bring all range images to one coordinate system

Reconstruction:
Integration of scans into a single mesh

Postprocess:
- Topological and geometric filtering
- Remeshing
- Compression



Digital Michelangelo Project

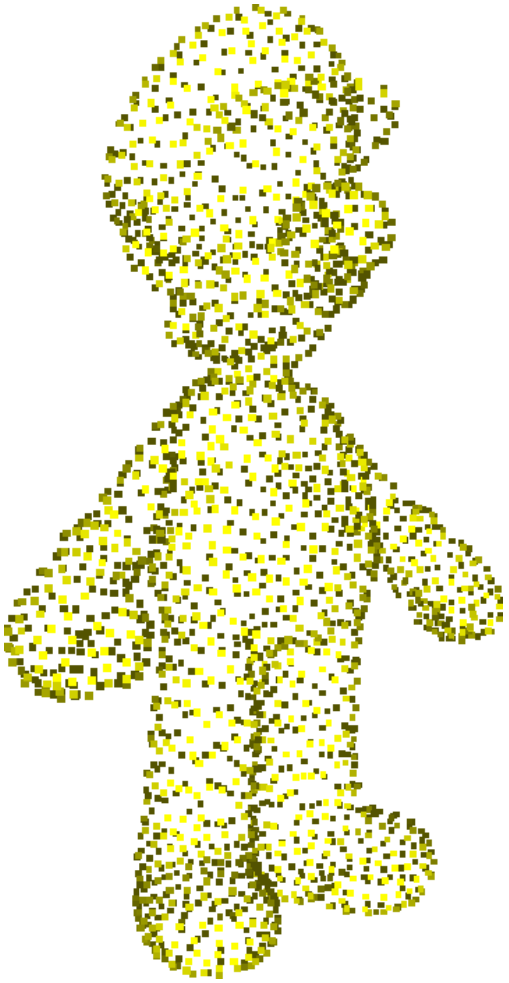


1G sample points → 8M triangles

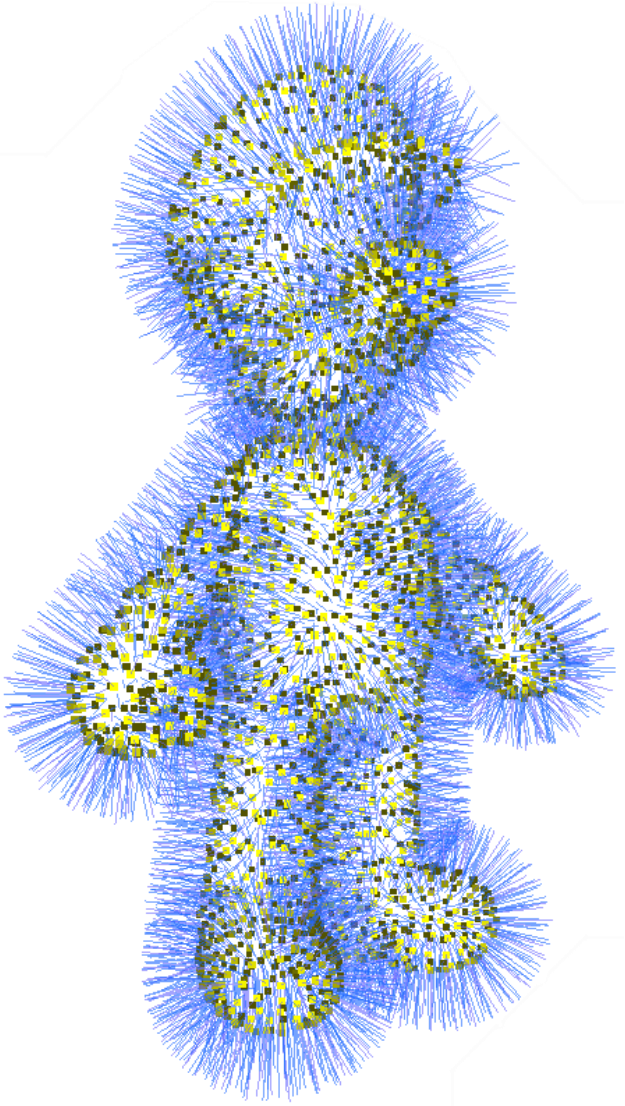
4G sample points → 8M triangles

Input to Reconstruction Process

Point cloud

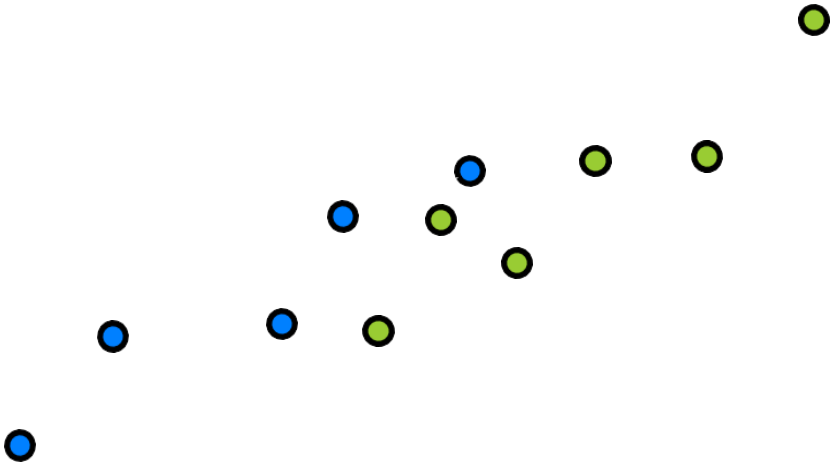


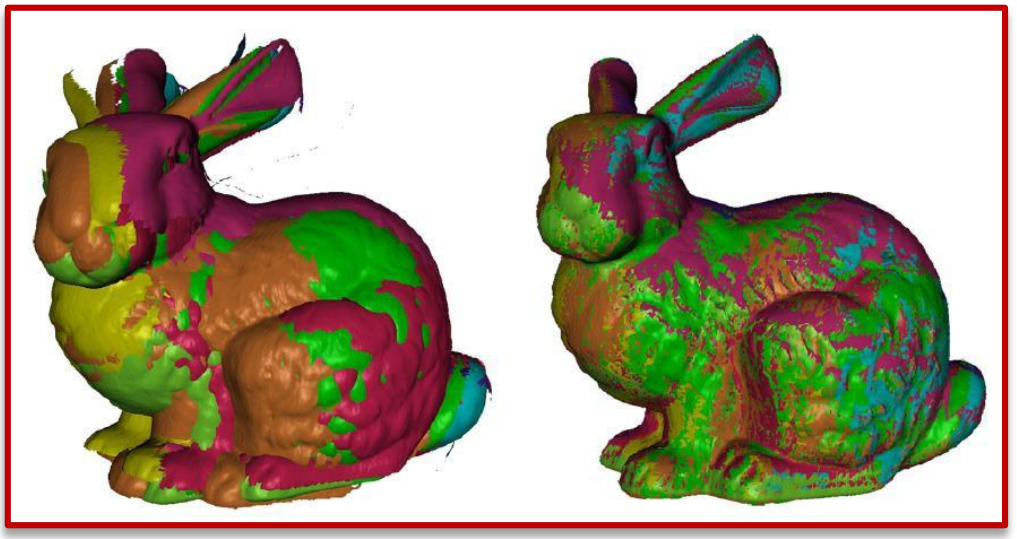
Oriented
Point cloud



How to Connect the Dots?

Explicit reconstruction:
Connect sample points by triangles

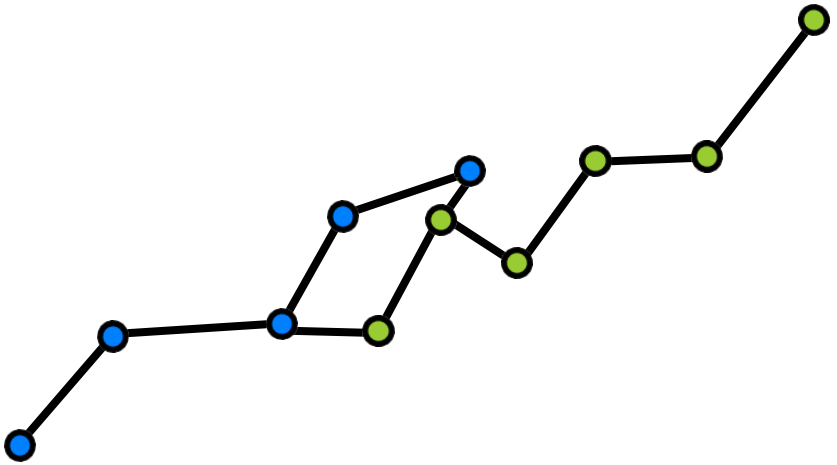


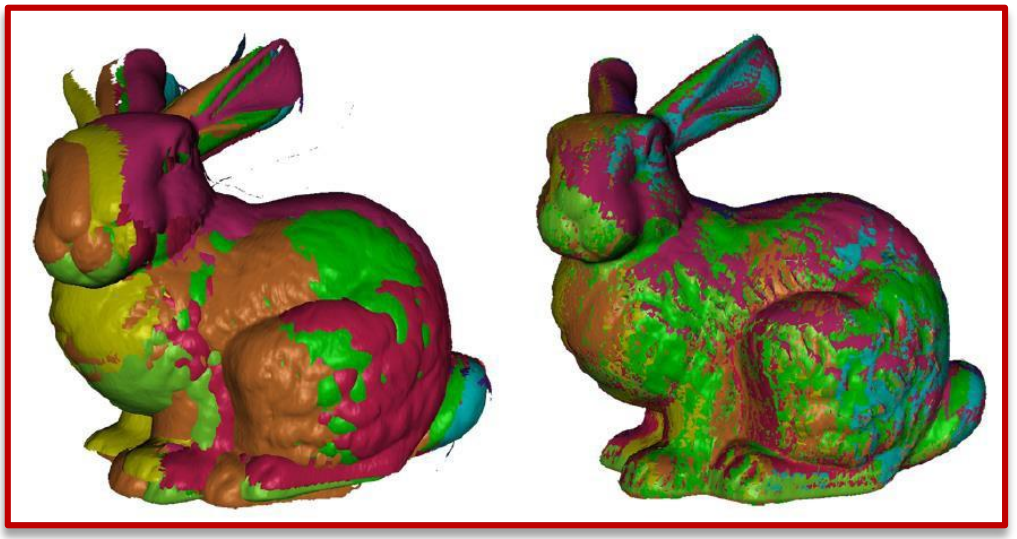


"Zippered Polygon Meshes from Range Images", Greg Turk and Marc Levoy, ACM SIGGRAPH 1994

How to Connect the Dots?

Explicit reconstruction:
Connect sample points by triangles

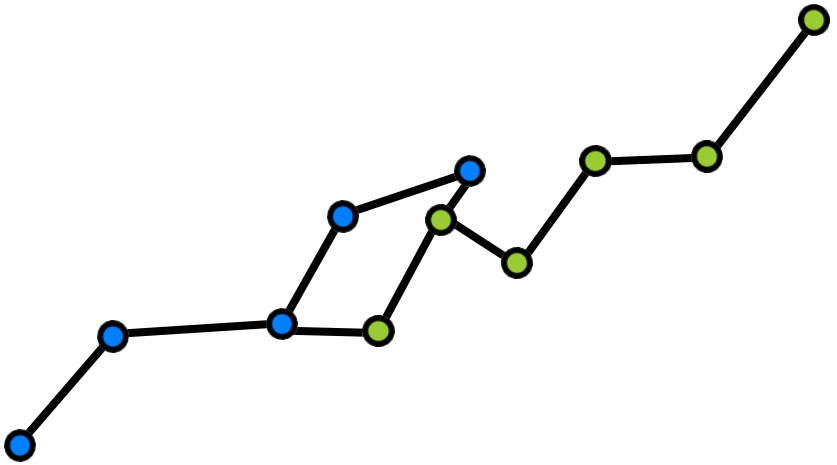




"Zippered Polygon Meshes from Range Images", Greg Turk and Marc Levoy, ACM SIGGRAPH 1994

How to Connect the Dots?

Explicit reconstruction:
Connect sample points by triangles



Problems:

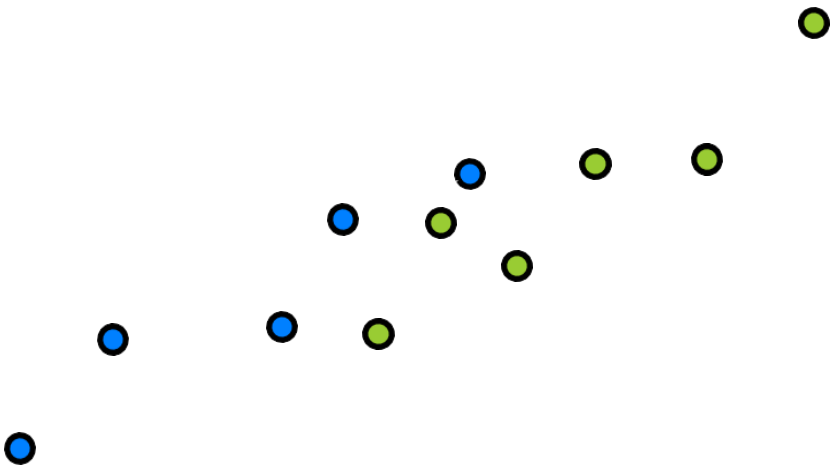
- Bad for noisy or misaligned data
- Can lead to holes or non-manifold situations

How to Connect the Dots?

Implicit reconstruction:

estimate a signed distance function (SDF)

extract zero set

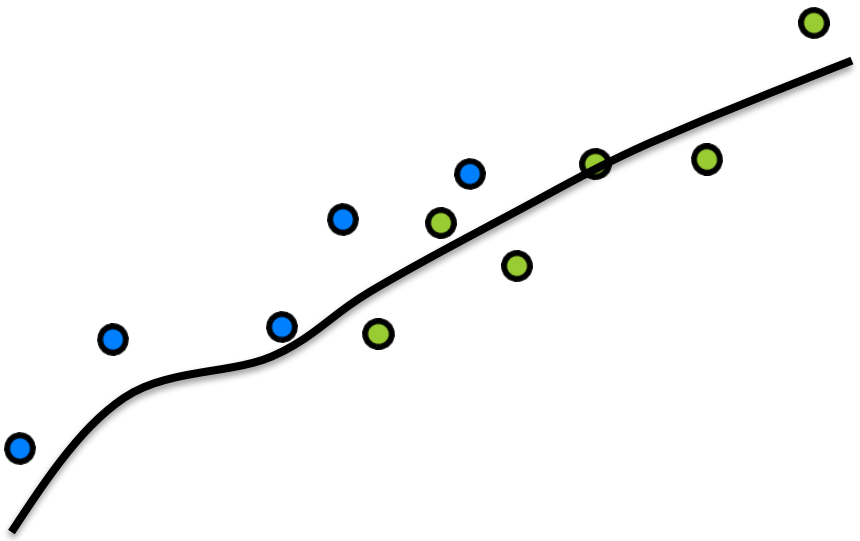


How to Connect the Dots?

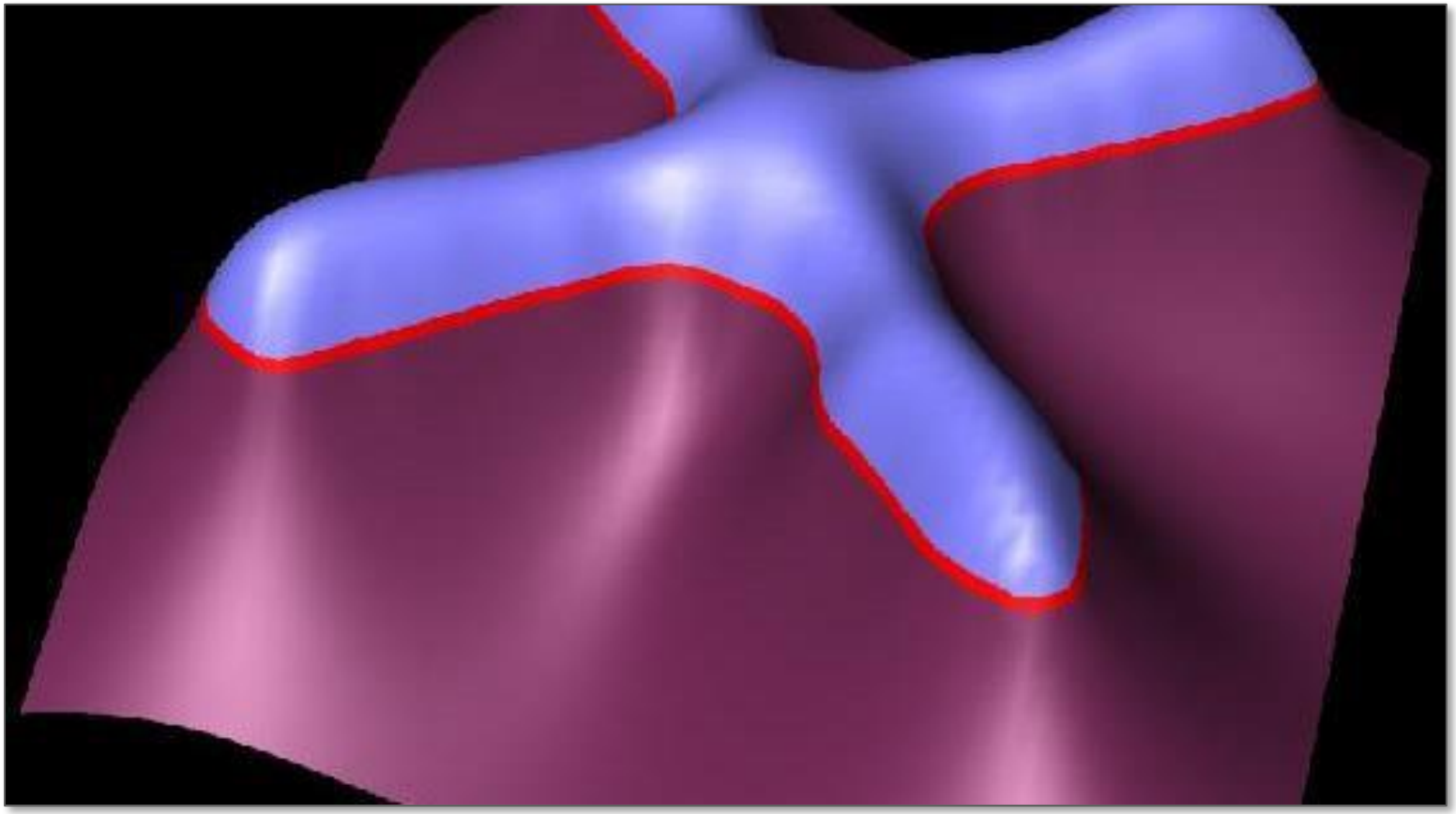
Implicit reconstruction:

estimate a signed distance function (SDF)

extract zero set



Implicit Curves and Surfaces



Surfaces implicites

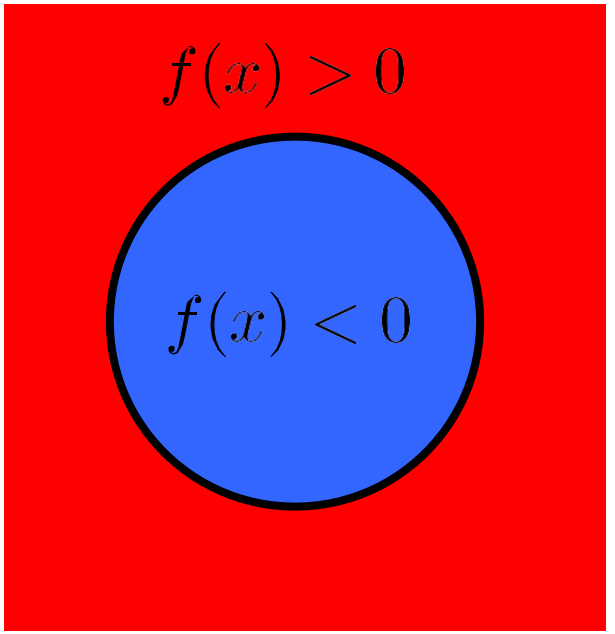
- Zero set d'une fonction scalaire
  - Courbe en 2D : $S = \{x \in \mathbb{R}^2 | f(x) = 0\}$
  - Surface en 3D : $S = \{x \in \mathbb{R}^3 | f(x) = 0\}$

- Partitionnement de l'espace

$\{x \in \mathbb{R}^m | f(x) > 0\}$ Extérieur

$\{x \in \mathbb{R}^m | f(x) = 0\}$ Surface/courbe

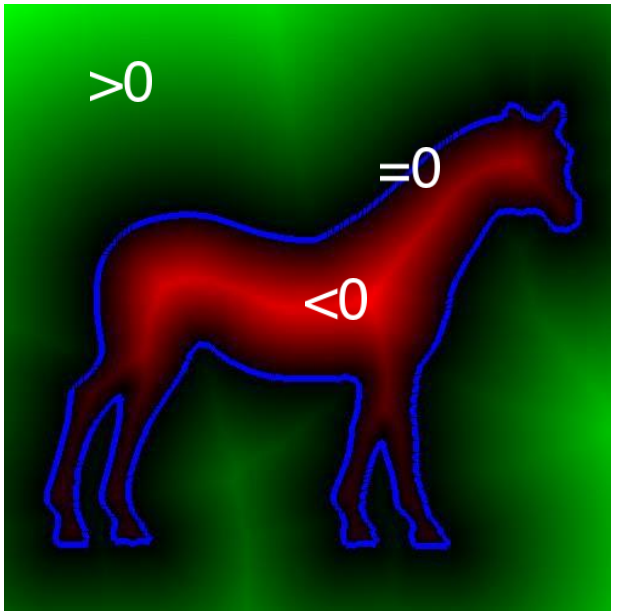
$\{x \in \mathbb{R}^m | f(x) < 0\}$ Intérieur



Surfaces implicites

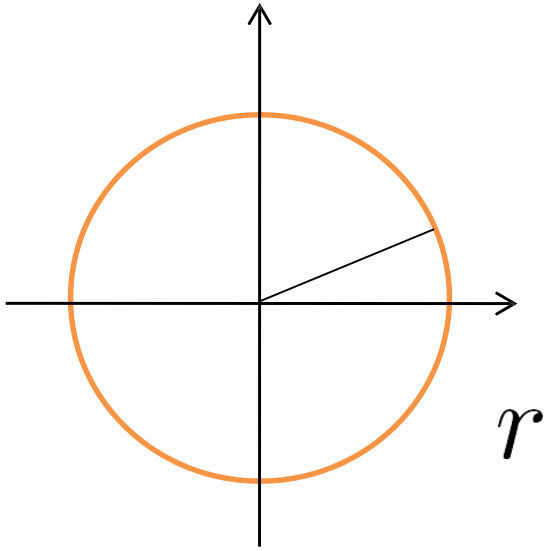
- Zero set d'une fonction scalaire $f : \mathbb{R}^m \rightarrow \mathbb{R}$
  - Courbe en 2D : $S = \{x \in \mathbb{R}^2 | f(x) = 0\}$
  - Surface en 3D : $S = \{x \in \mathbb{R}^3 | f(x) = 0\}$

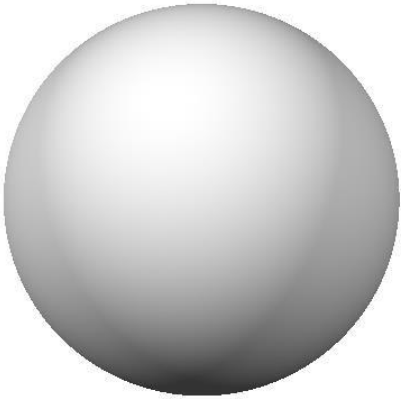
- Ensemble de niveau zéro d'une fonction de distance signée



Surfaces implicites

- Cercle et sphère implicite





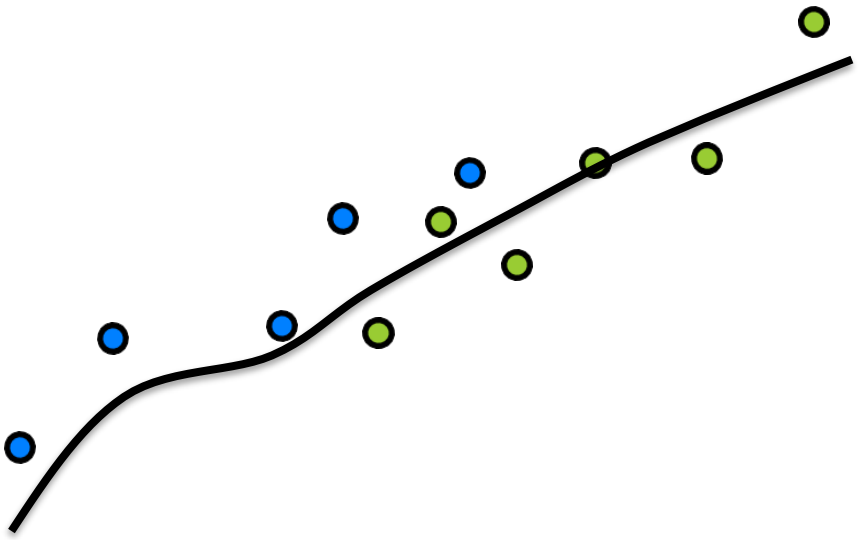
$$f(x,y) = x^2 + y^2 - r^2 \qquad f(x,y,z) = x^2 + y^2 + z^2 - r^2$$

Comment connecter les points ?

Reconstruction implicite :

éstimer une fonction de distance signée (SDF)

extraire le zero set

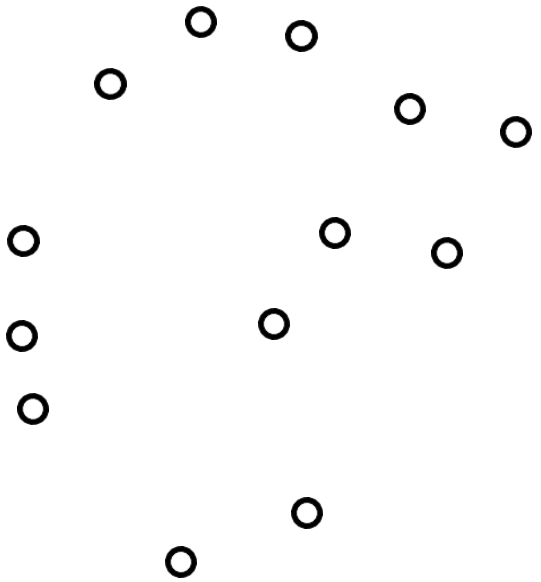


Comment connecter les points ?

Reconstruction implicite :

éstimer une fonction de distance signée (SDF)

extraire le zero set

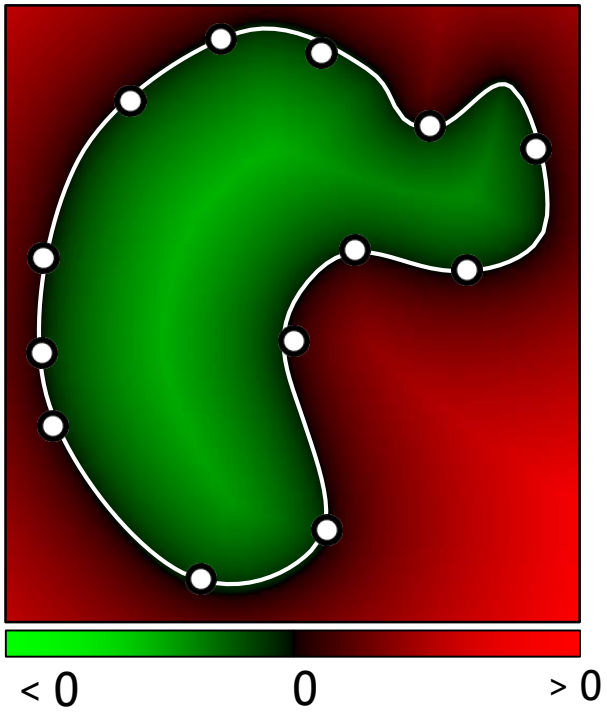


Comment connecter les points ?

Reconstruction implicite :

éstimer une fonction de distance signée (SDF)

extraire le zero set

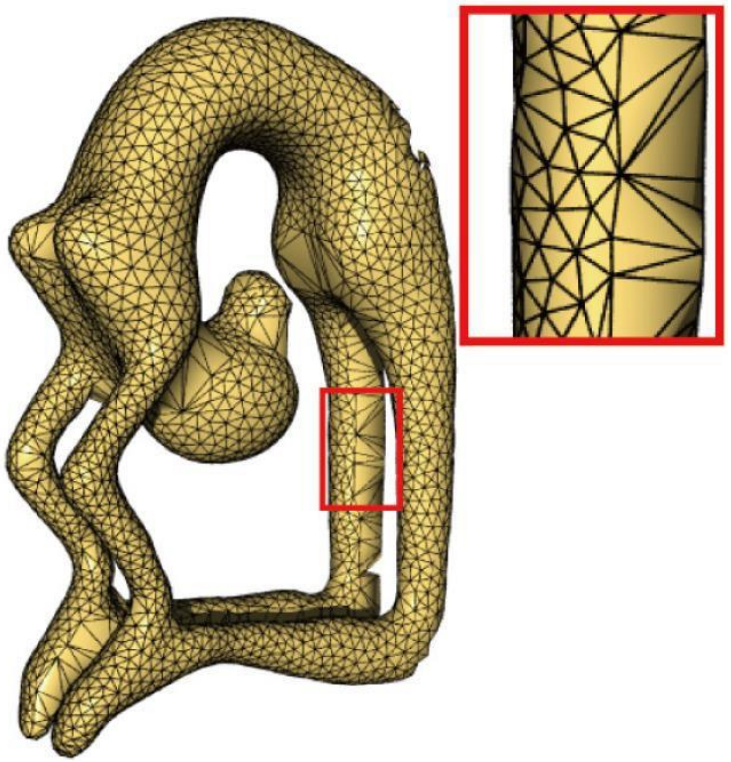


Avantages :

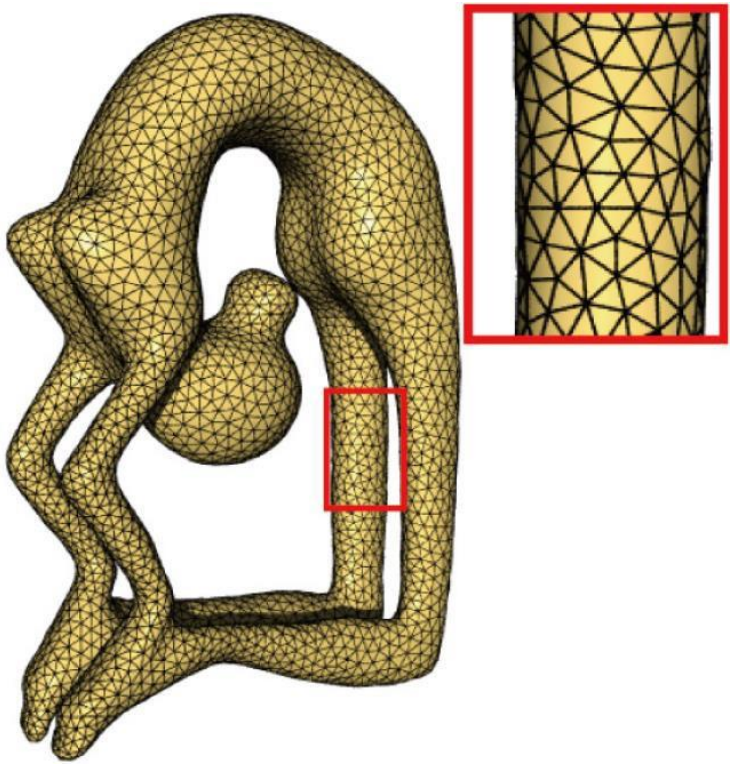
- Approximation des points d'entrée
- Watertight manifold par construction

Implicite vs. Explicite

Input



Explicit

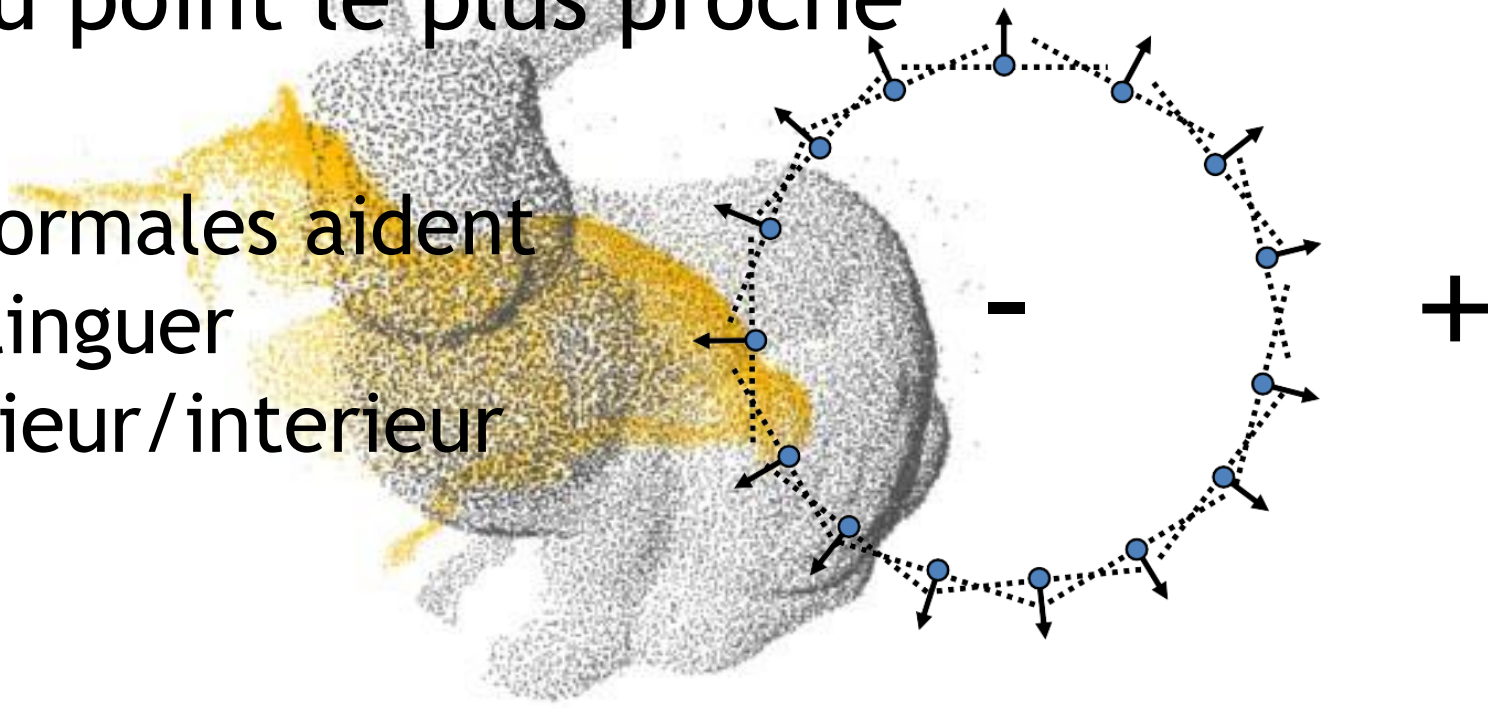


Implicit

SDF from Points and Normals

Calculer la distance signée au plan tangent du point le plus proche

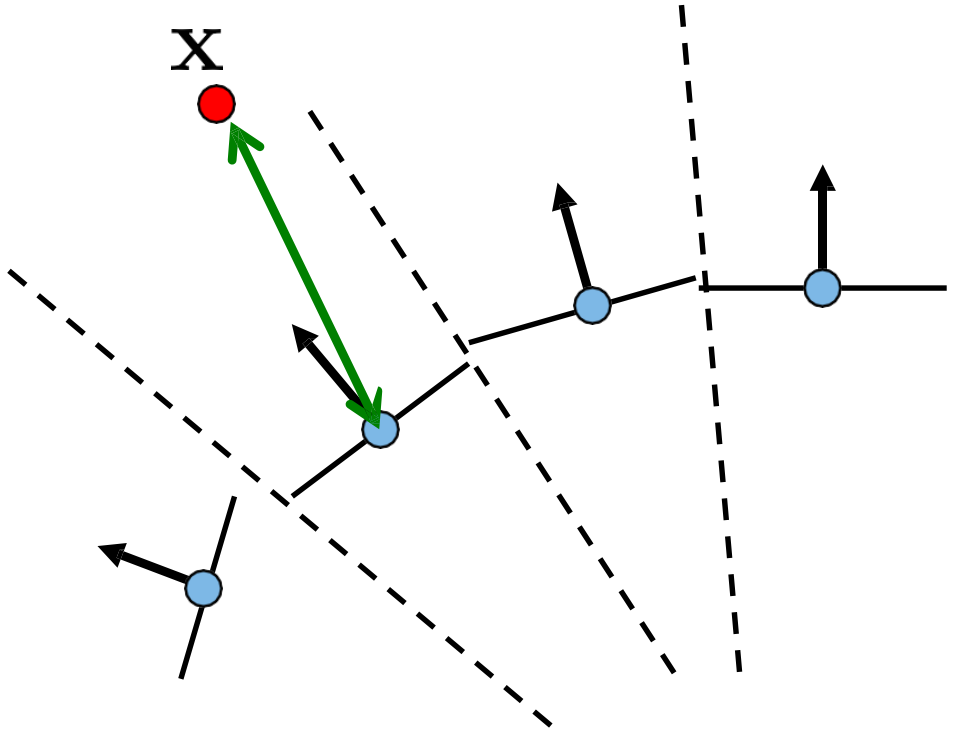
Les normales aident à distinguer exterieur/interieur



"Surface reconstruction from unorganized points", Hoppe et al., ACM SIGGRAPH 1992
http://research.microsoft.com/en-us/um/people/hoppe/proj/recon/

SDF from Points and Normals

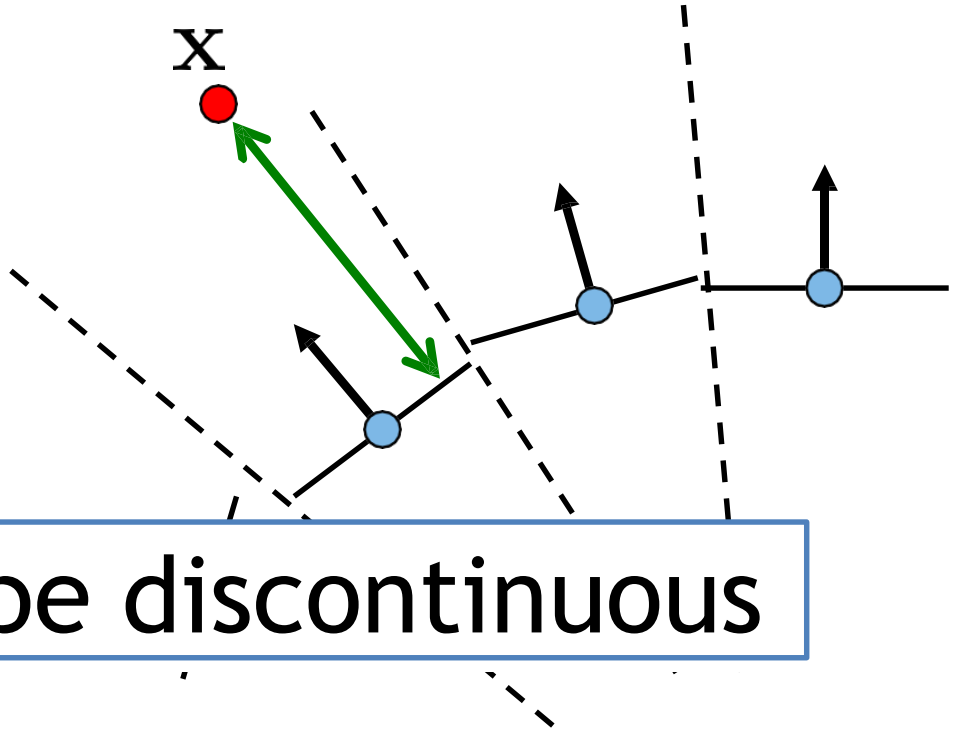
Compute signed distance to the tangent plane of the closest point



SDF from Points and Normals

Compute signed distance to the tangent plane of the closest point

Problème ?



The function will be discontinuous

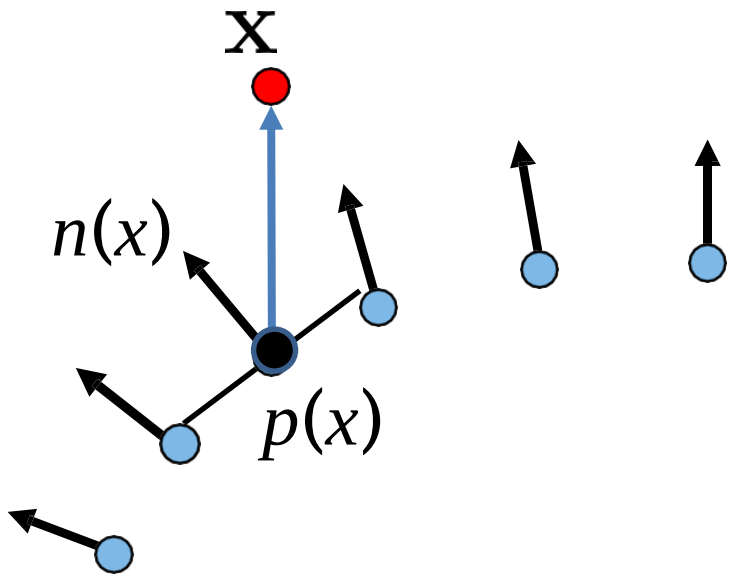
Note: The Hoppe92 paper computes the tangent planes slightly differently (by PCA on k-nearest-neighbors of each data point, see next class), but the consequences are still the same.

Utilisation des MLS

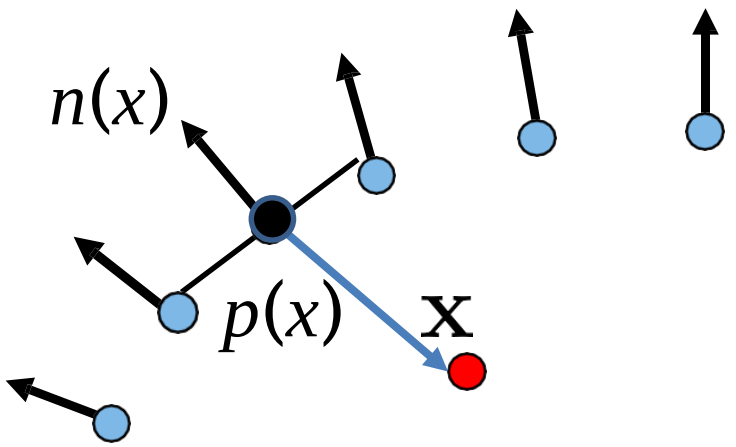
- On peut definir une fonction implicite

$$f(x) = (x - p(x))^T . n(x)$$

$p(x)$: projection sur la surface MLS



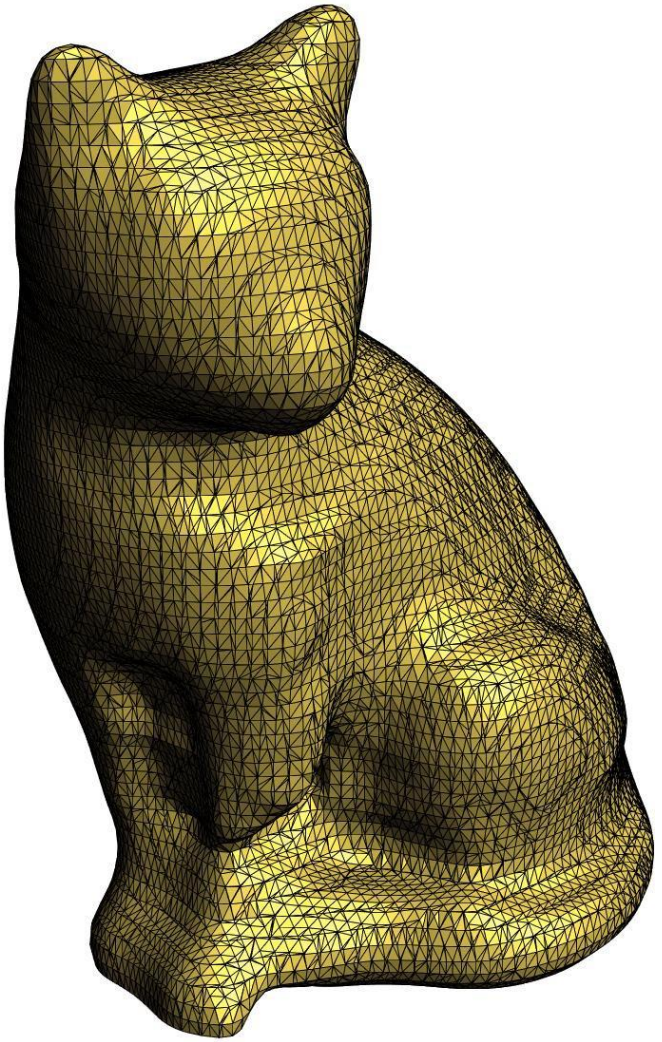
$f(x) > 0$



$f(x) < 0$

(la surface est alors le « 0-set » de cette fonction).

Exemple : Reconstruction



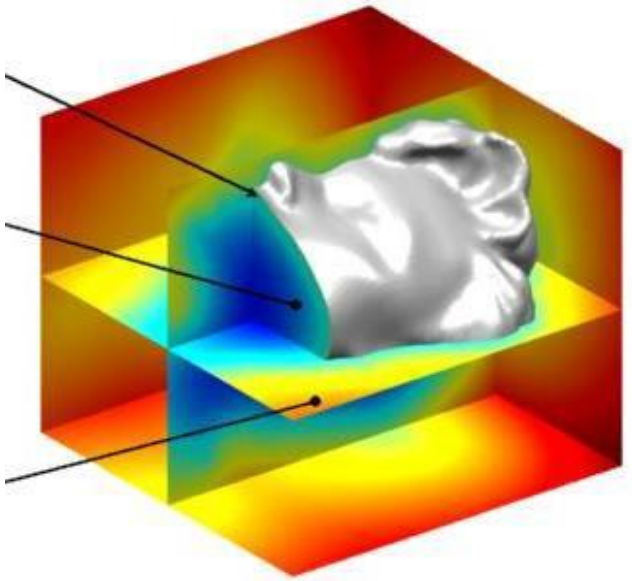
Extraire la Surface

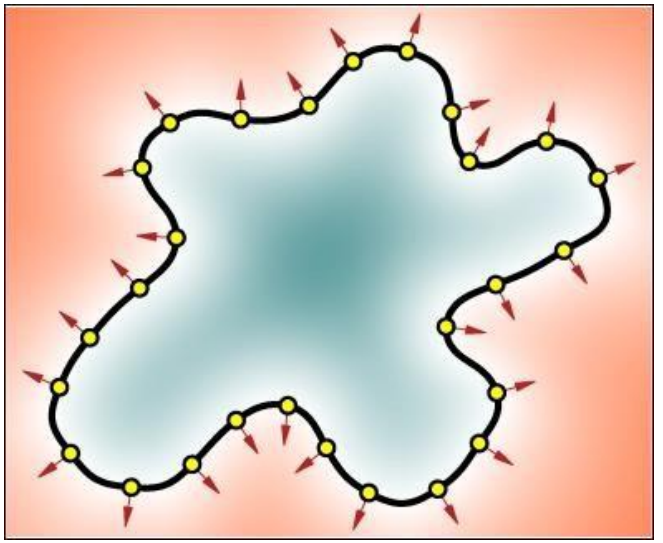
Comment extraire la surface d'un ensemble de niveaux ?

$F(\mathbf{x}) = 0 \rightarrow$ surface

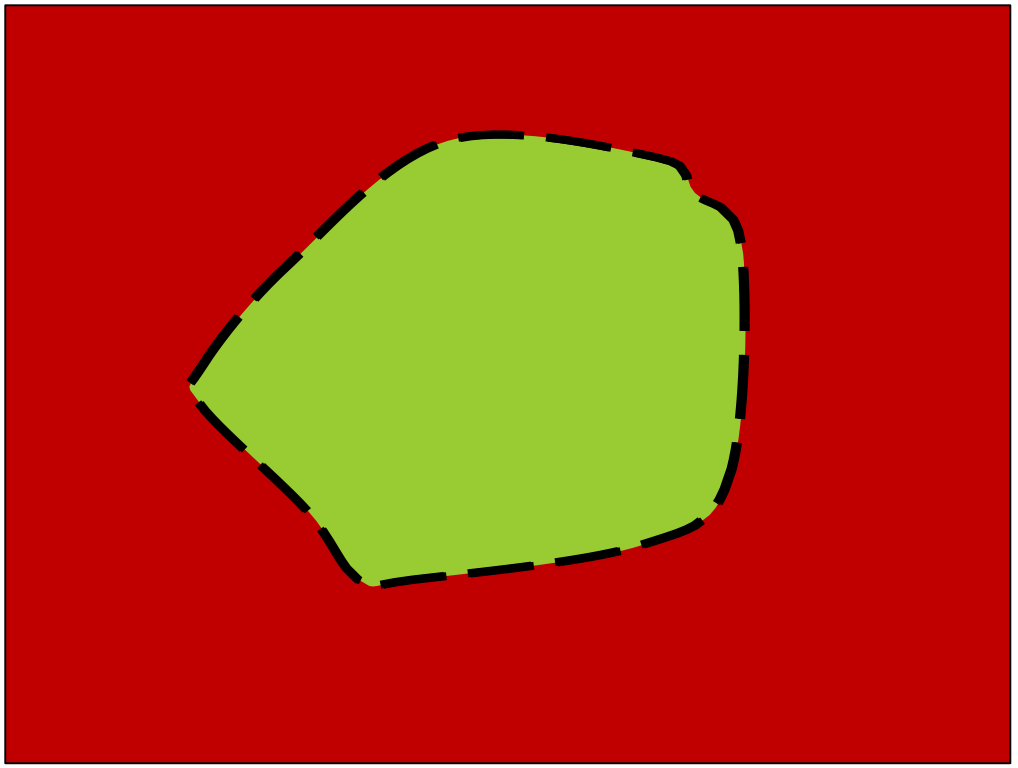
$F(\mathbf{x}) < 0 \rightarrow$ inside

$F(\mathbf{x}) > 0 \rightarrow$ outside

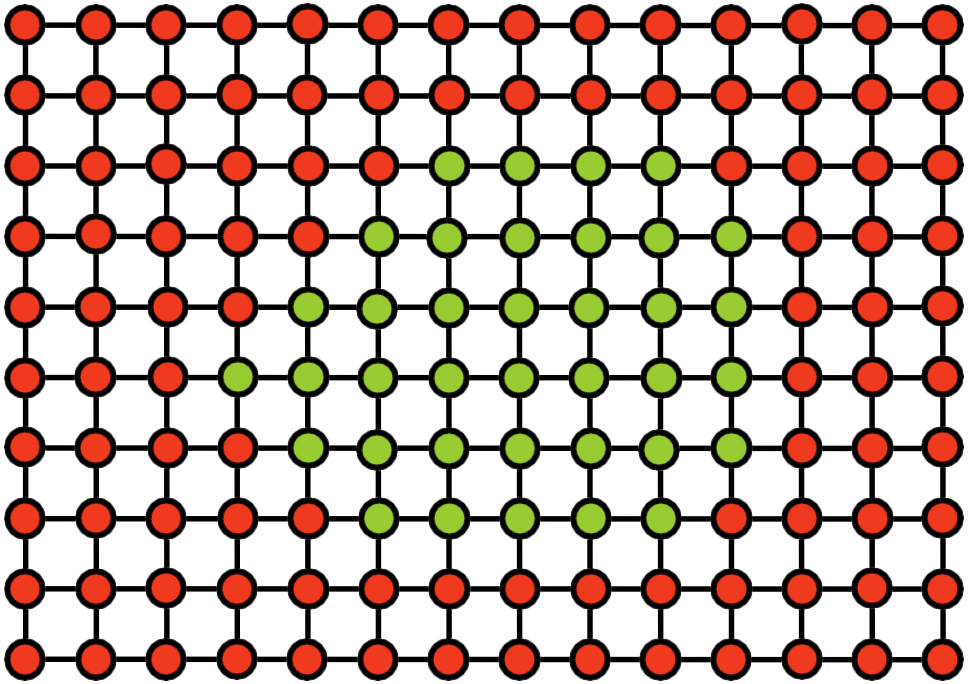


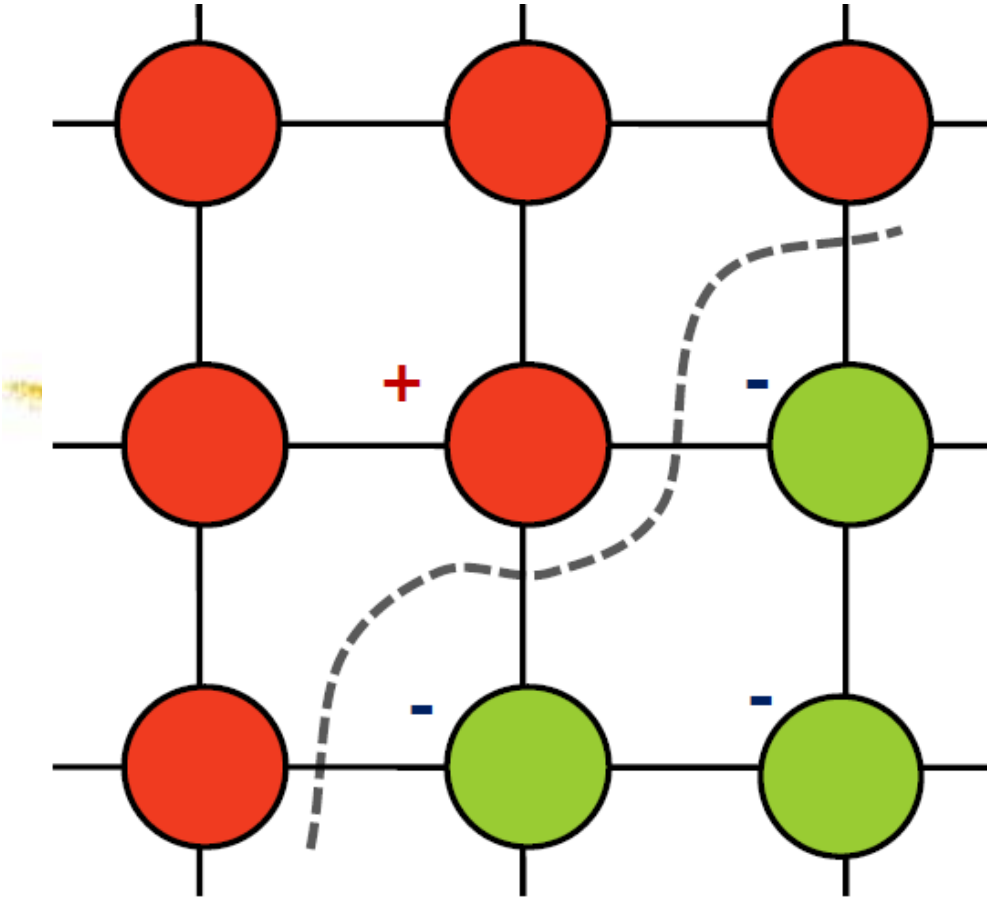


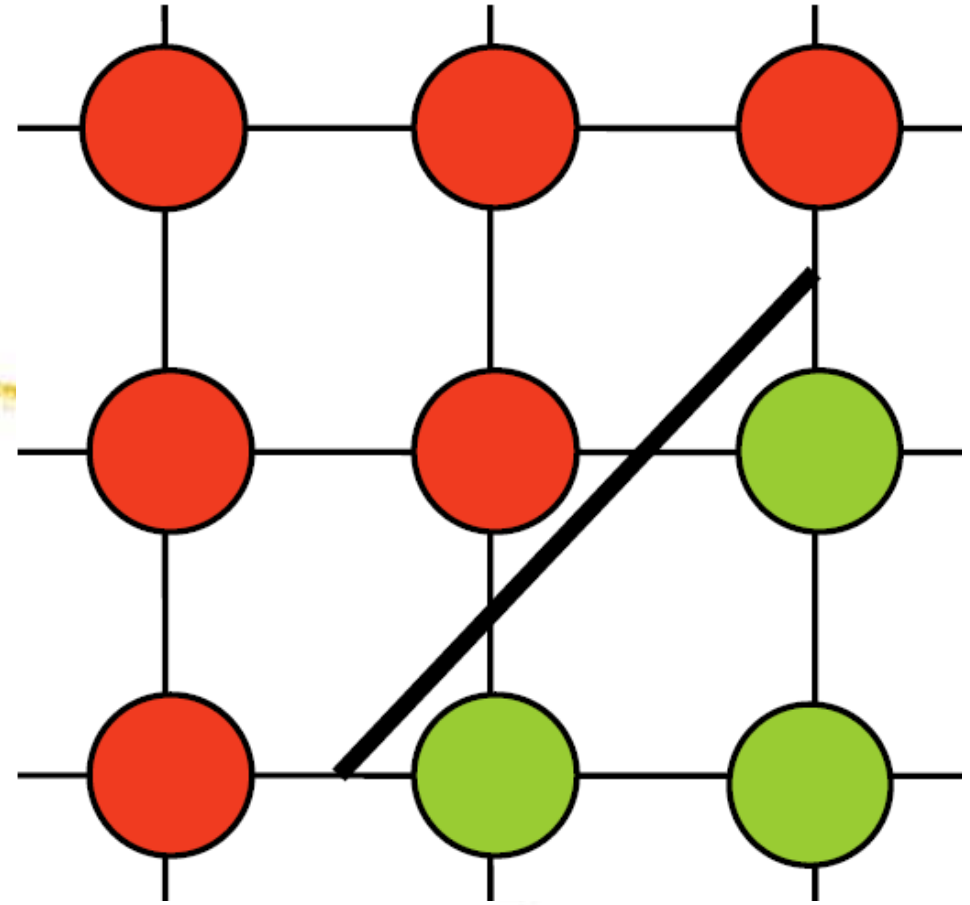
Échantillonner la SDF



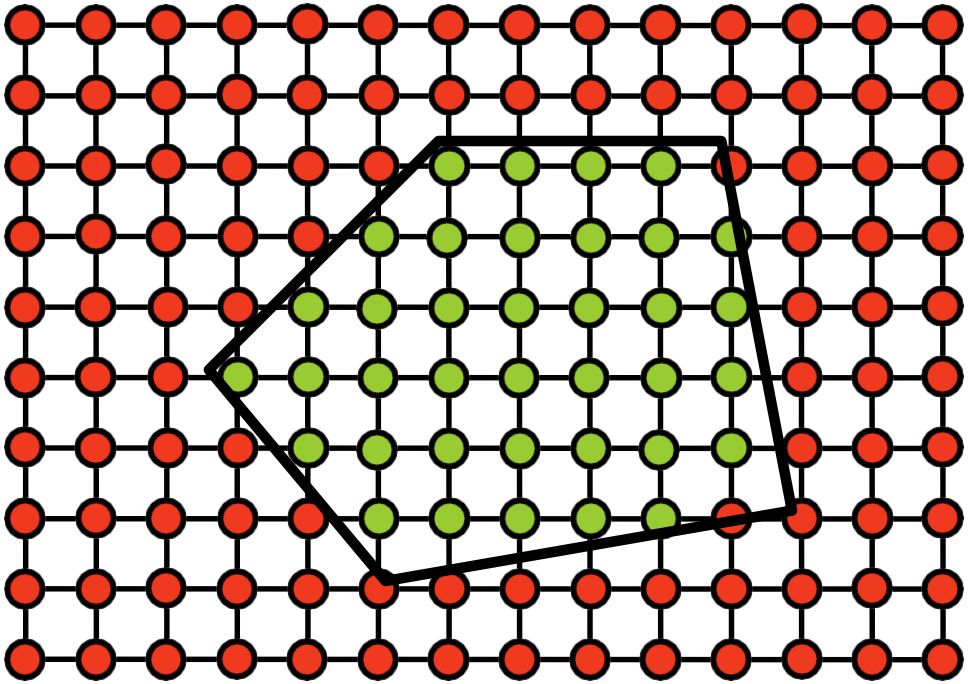
Échantillonner la SDF





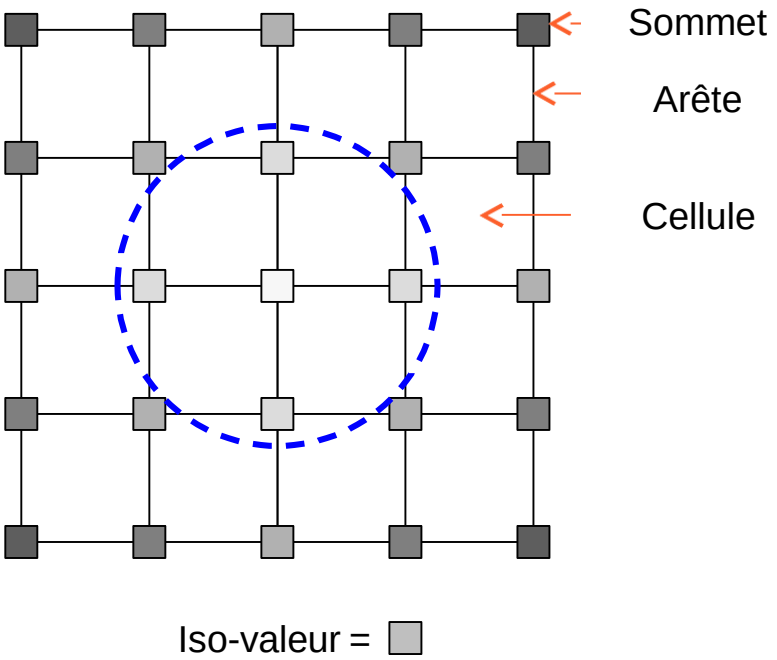


Échantillonner la SDF



Extraction de surface a partir d'une fonction implicite sur une grille

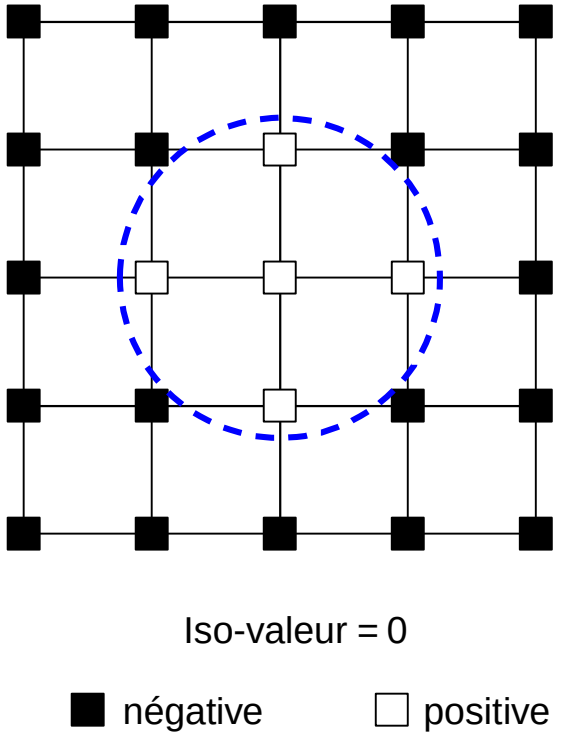
- Input : une fonction continue f définie partout dans l'espace
- On extrait la surface correspondant au « 0-set » de f
- On définira une fonction implicite a partir d'un pointset. On pourra donc extraire la surface correspondante



Ressources extérieures utilisées :
http://www.cs.wustl.edu/~taoju/cse554/index.htm#lectures

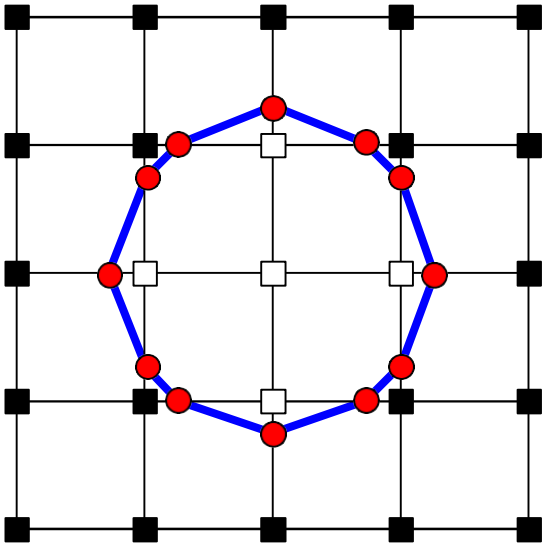
« Marching cubes »

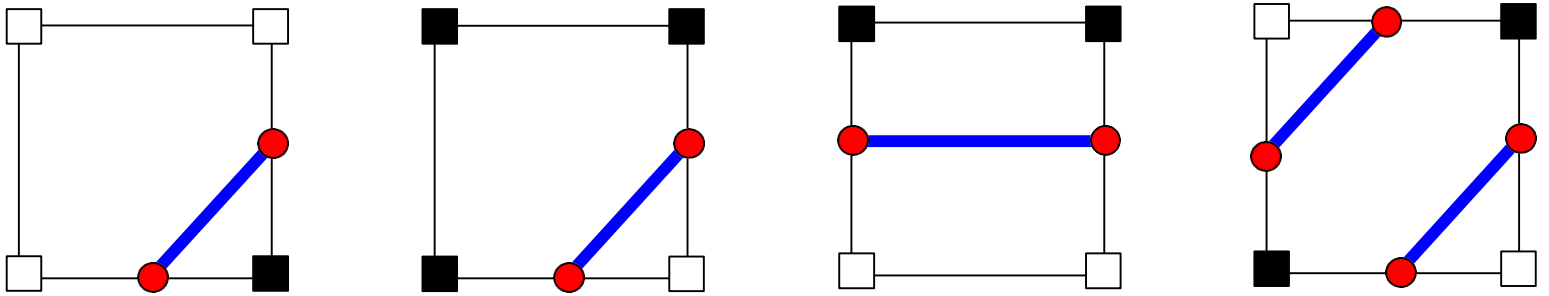
- Étiqueter les sommets comme « intérieur » ou « extérieur »



« Marching cubes »

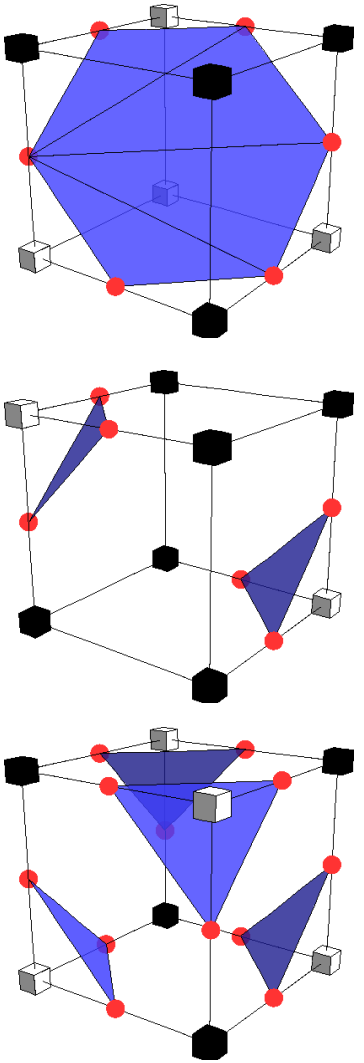
- Étiqueter les sommets comme « intérieur » ou « extérieur »
- Sur les arêtes changeant de signe, créer un sommet.
- Connecter les arêtes (en procédant cellule par cellule).





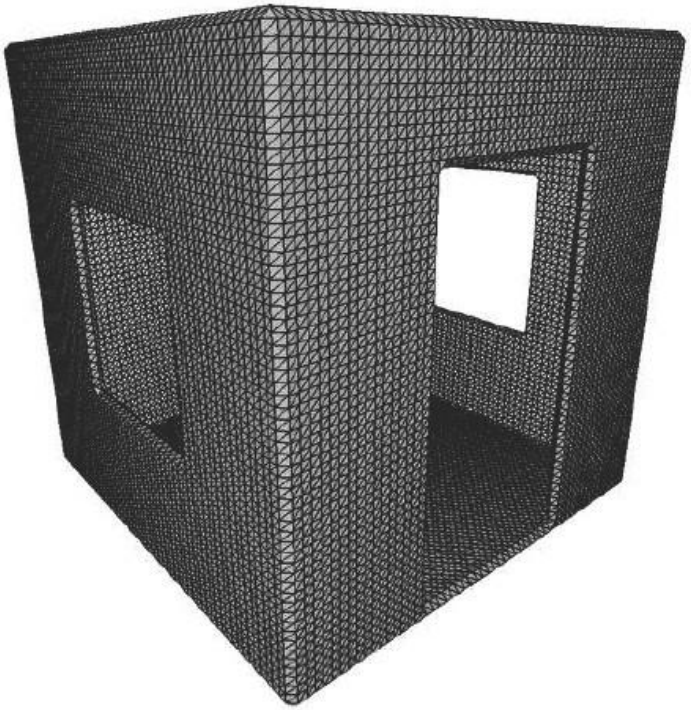
« Marching cubes »

- En 3D : $2^8 = 256$ combinaisons
- Look-up table
- Fastidieux a implémenter
- Code open source disponible
- Toujours manifold comme sortie
- Géométrie de basse qualité



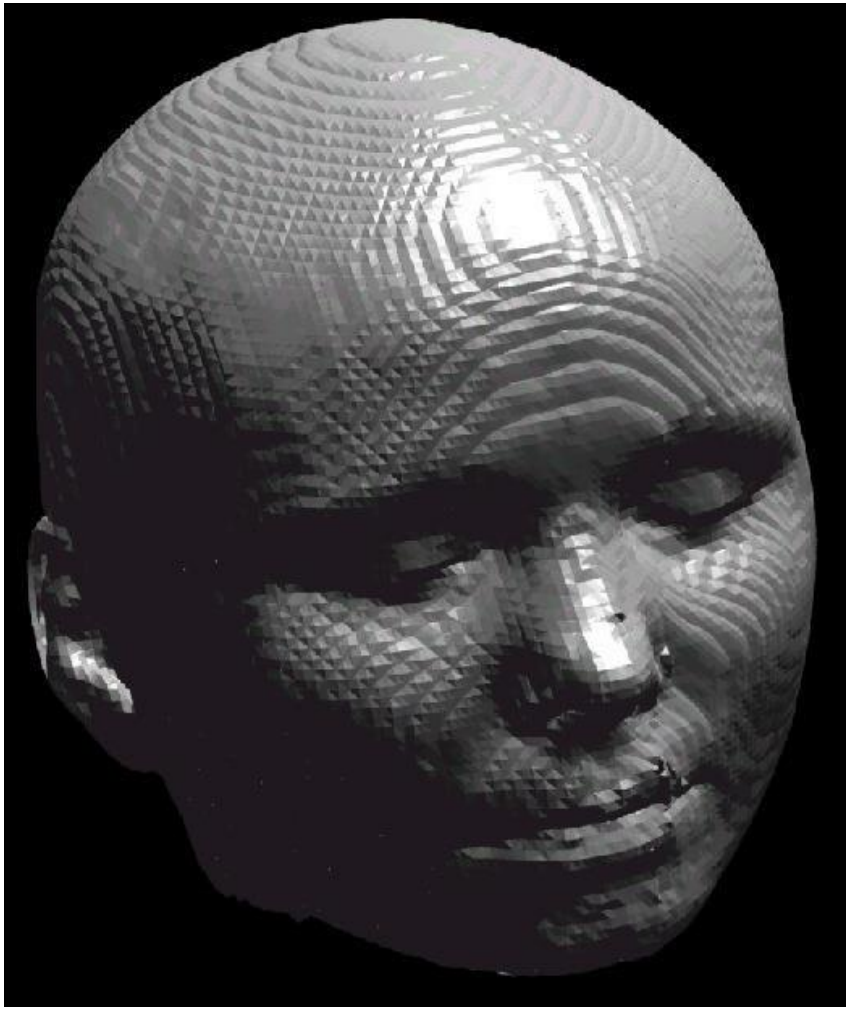
Marching Cubes – Problèmes

- Grille non adaptative
- Beaucoup de primitives pour représenter les petites caractéristiques



Images from: “Dual Marching Cubes: Primal Contouring of Dual Grids” by Schaeffer et al.

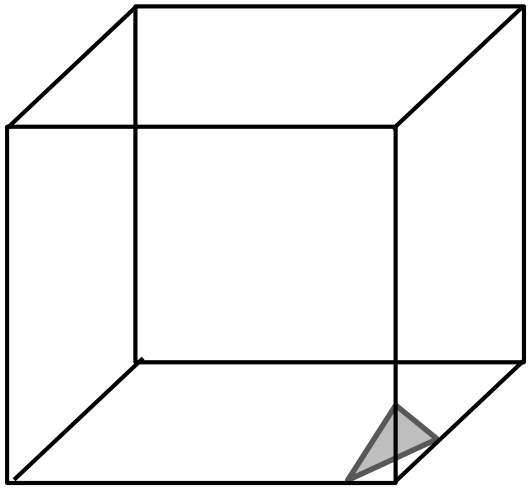
Marching Cubes – Problèmes

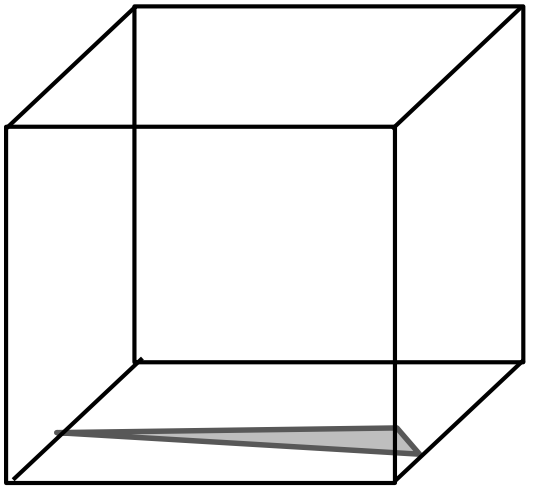


Marching Cubes – Problèmes

Problèmes d'arêtes courtes

- Surface intersecte le cube près d'un coin, le petit triangle résultant ne contribue pas beaucoup au maillage (aire réduite)
- Si l'intersection est proche d'une arête du cube, les triangles sont mal formés (mauvais aspect ratio)

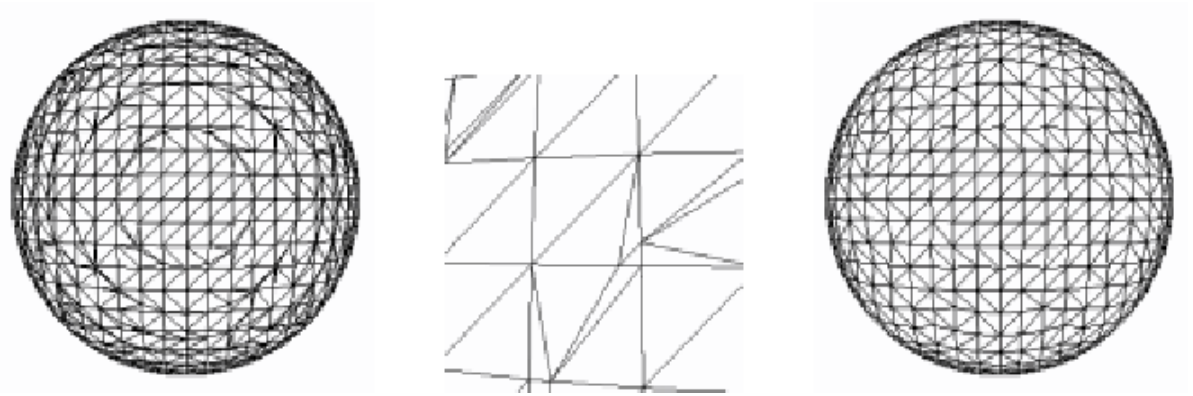




Grid Snapping

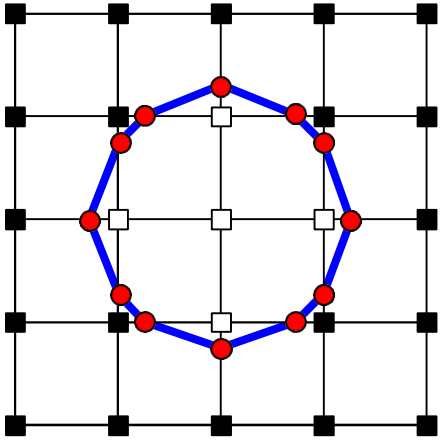
Solution : fixer un seuil sur les distances entre les sommets créés les coins du cube quand < d_{snap} le sommet est ramené au coin

Si plus d'un sommet d'un triangle sont ramenés au même point, le triangle est ignoré

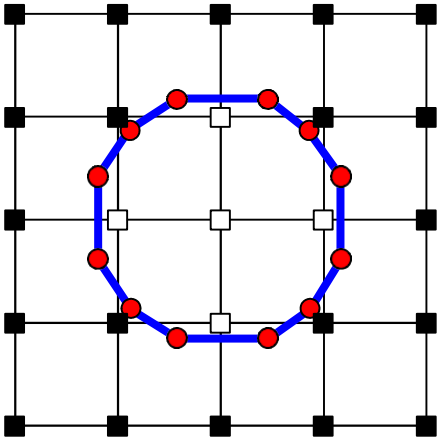


Méthodes primales et duales

Primales : nouveaux sommets sur les aretes de la grille (par ex, Marching cubes)

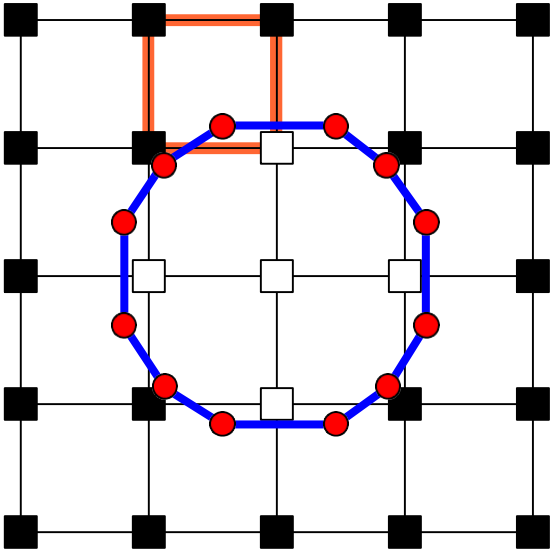


Duales : nouveaux sommets dans les cellules de la grille



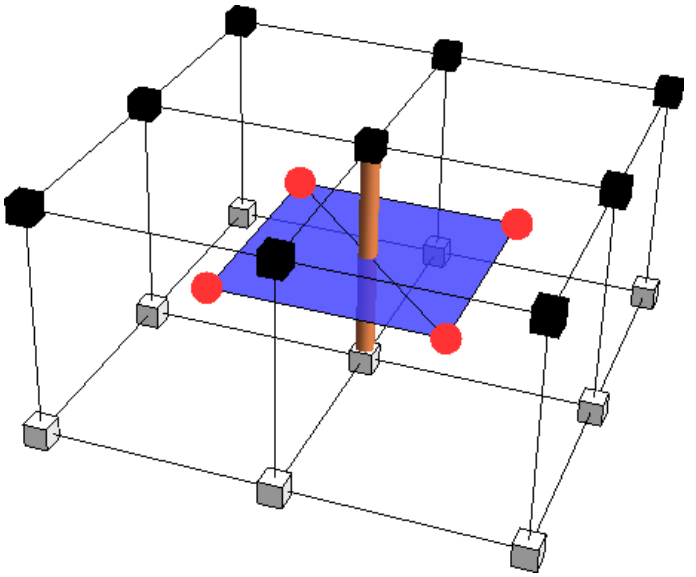
Méthodes duales (dual contouring)

- Pour chaque cellule avec un changement de signe: créer un sommet
- Pour chaque arête avec un changement de signe : connecter les sommets des cellules séparées par l’arête



Dual contouring (3D)

- Pour chaque cellule avec un changement de signe : créer un sommet
- Pour chaque arête avec un changement de signe : connecter les sommets des cellules séparées par l’arête
- Pas besoin de look-up table



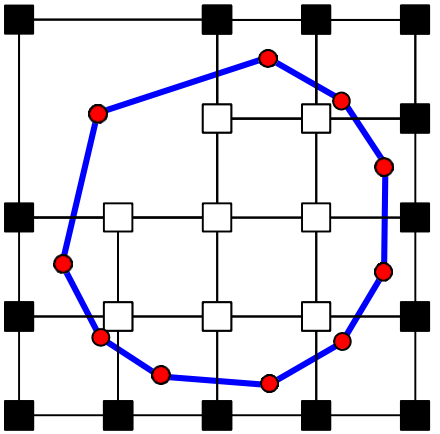
Comparaison

Marching Cubes

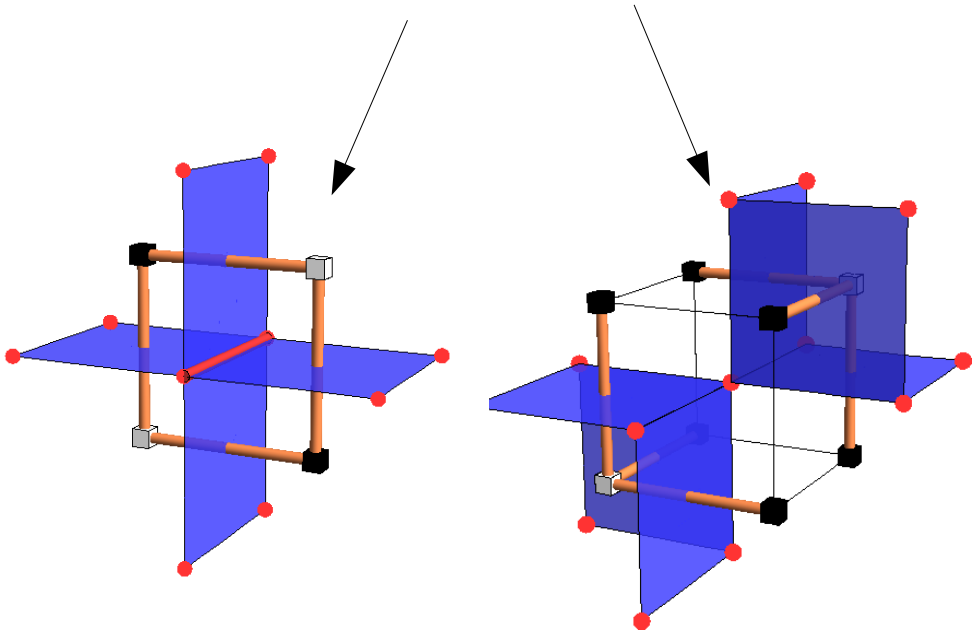
Dual Contouring

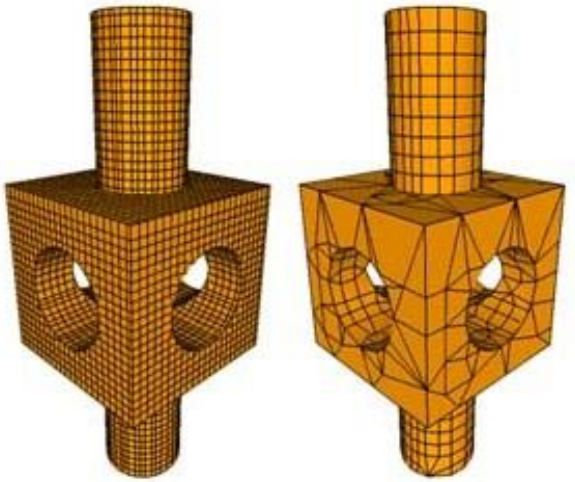
Dual contouring (3D)

- Plusieurs manières possibles de définir la position d'un sommet
- Meilleure qualité avec Dual Contouring
- Dual Contouring peut être réalisé sur une structure non-uniforme
- **La sortie peut être non-manifold**



Dual contouring sur un quadtree



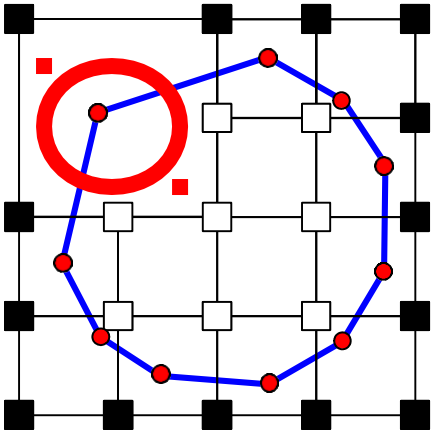


MC sur une grille uniforme

DC sur un octree

Dual contouring (3D)

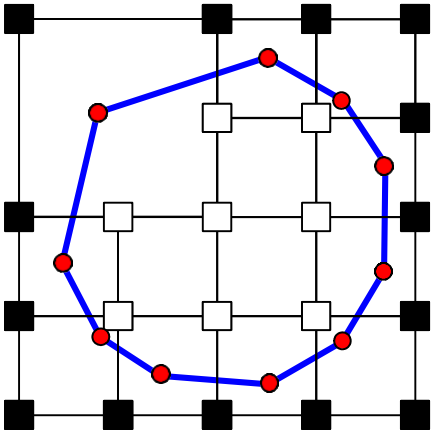
- Choix raisonnables pour la position du sommet inséré dans la cellule :
  - Centre de la cellule
  - Projection sur la surface MLS
  - … ?



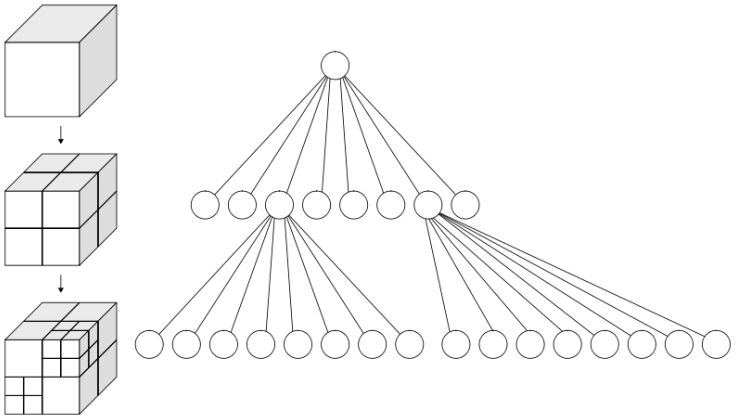
Dual contouring sur un quadtree

Dual contouring (3D)

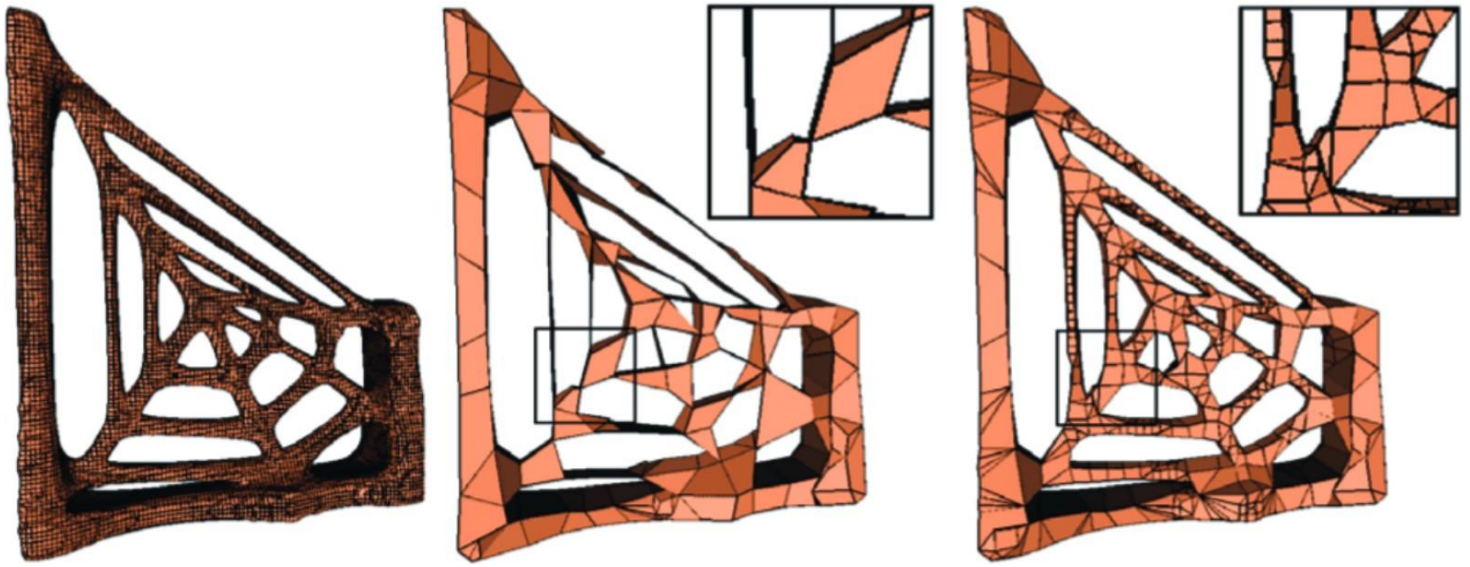
- Stratégie raisonnable pour l'extraction sur un octree :
  - Initialiser à un niveau minimum (par ex, 5)
  - Itérativement, subdiviser une cellule si la fonction implicite à ses coins change de signe. S'arrêter à une profondeur maximale (par ex, 10)
- Pour les arêtes :
  - Partir des nœuds les plus profonds pour tester les arêtes qui changent de signe.
  - Trouver les 3 cellules adjacentes, et insérer le morceau de surface correspondant.
- Difficile à coder.



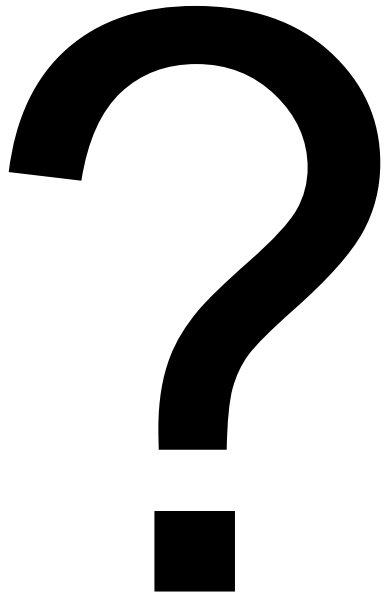
Dual contouring sur un quadtree



Extensions : manifold dual contouring (3D)

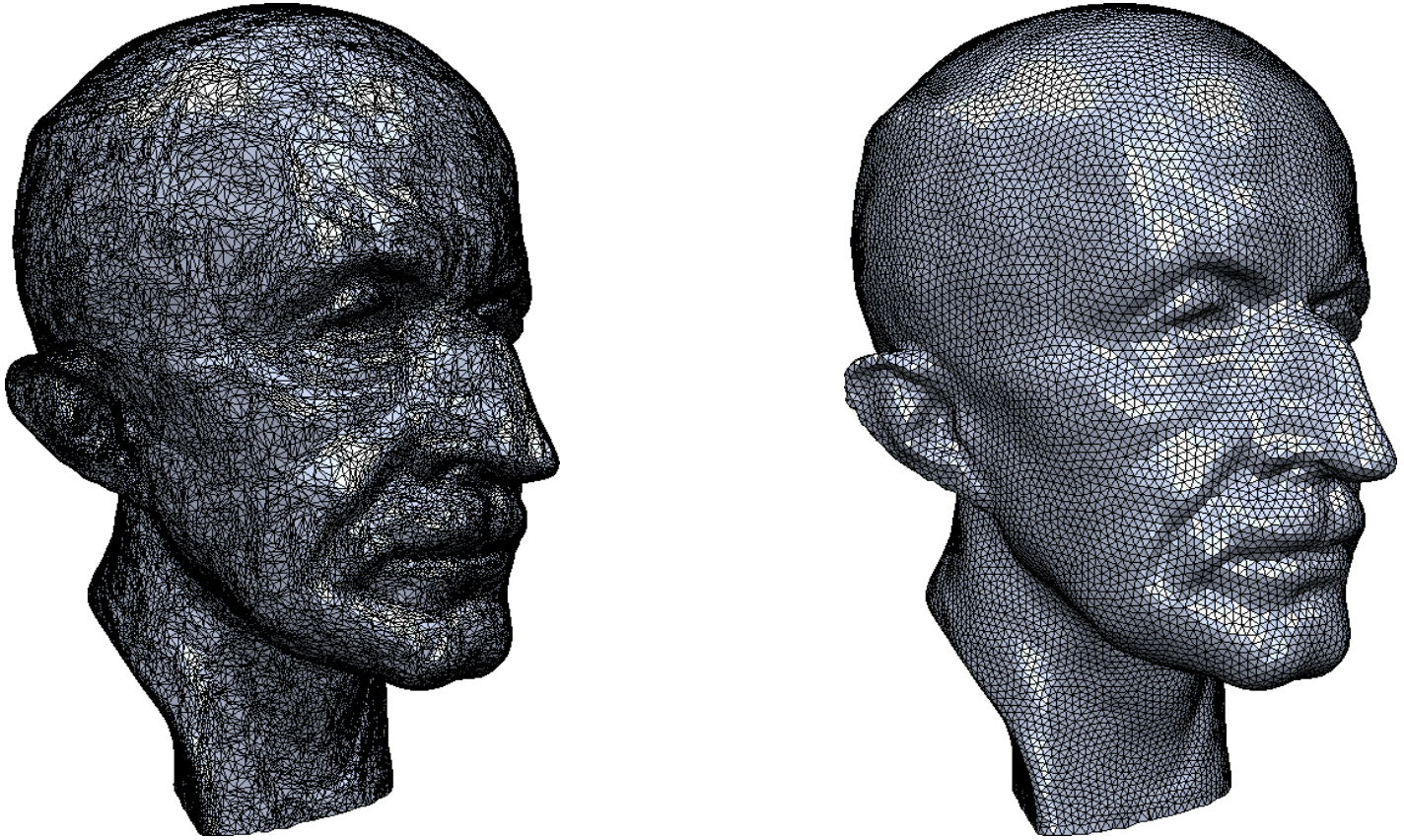


Questions ?



Remeshing

Given a 3D mesh, find a “better” discrete representation of the underlying surface

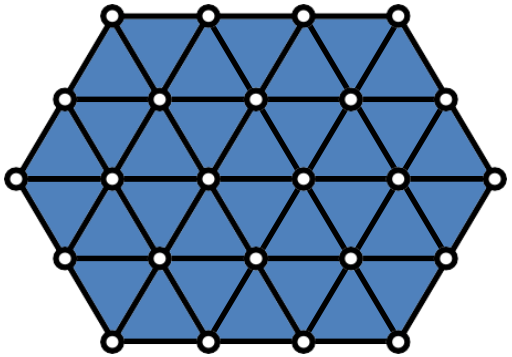


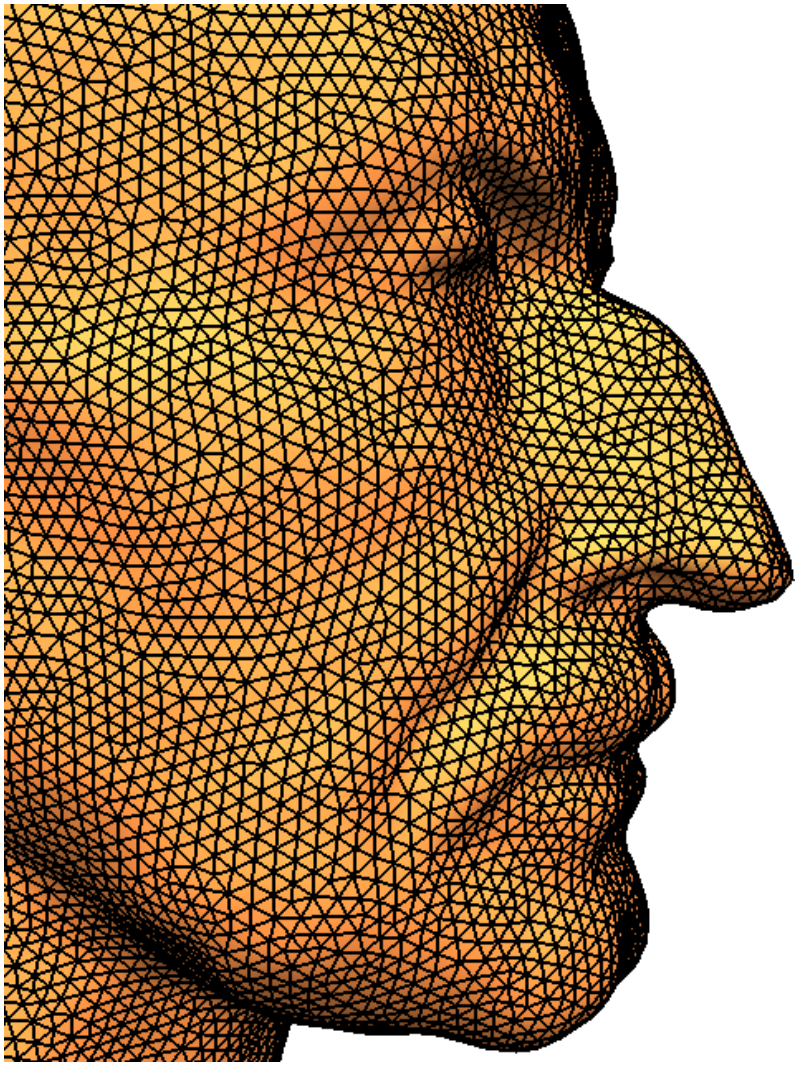
What is a good mesh?

Equal edge lengths

Equilateral triangles

Valence close to 6





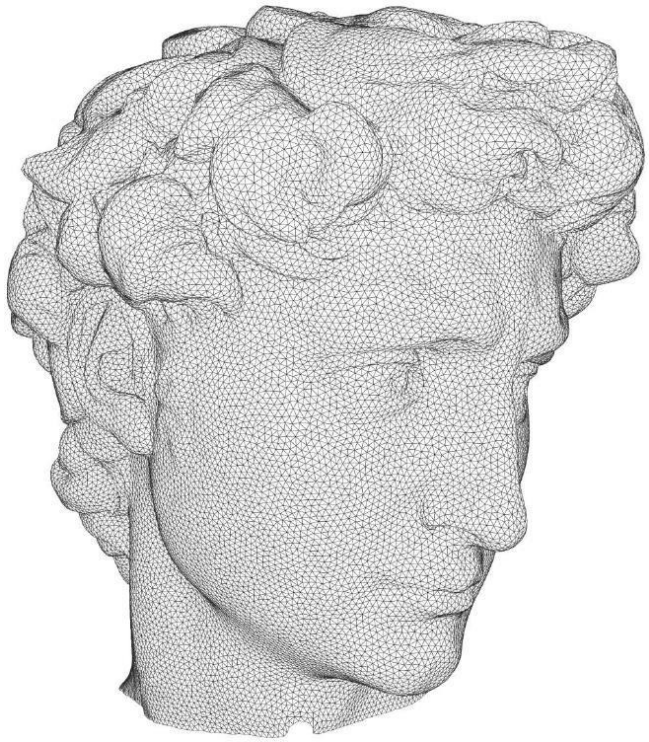
What is a good mesh?

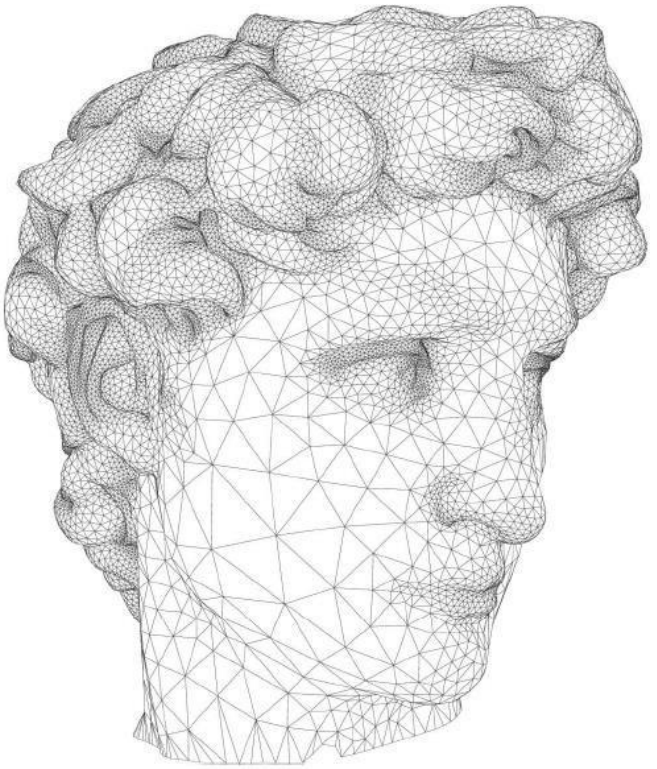
Equal edge lengths

Equilateral triangles

Valence close to 6

Uniform vs. adaptive sampling





What is a good mesh?

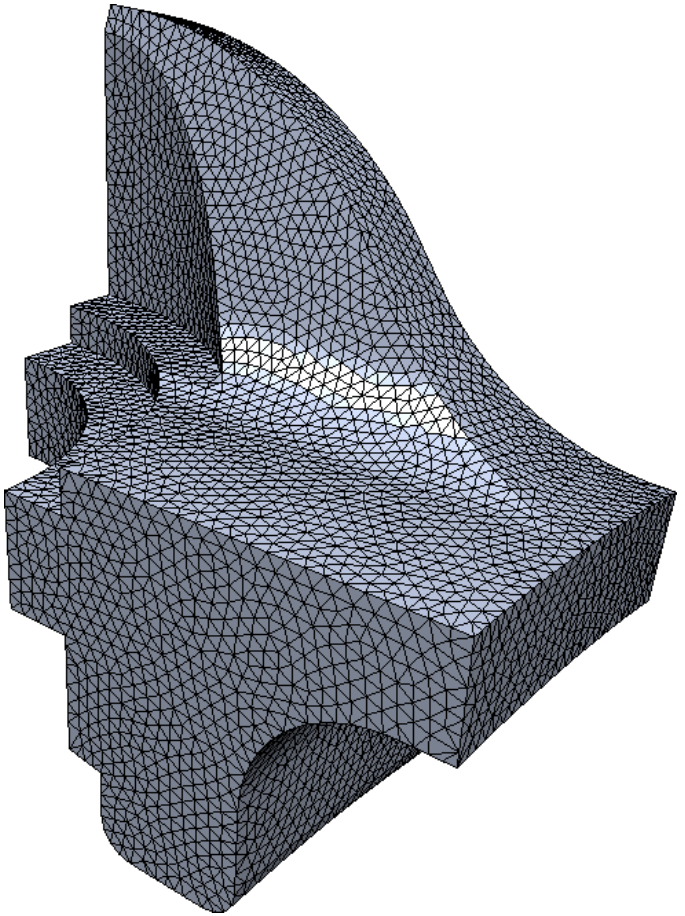
Equal edge lengths

Equilateral triangles

Valence close to 6

Uniform vs. adaptive sampling

Feature preservation



What is a good mesh?

Equal edge lengths

Equilateral triangles

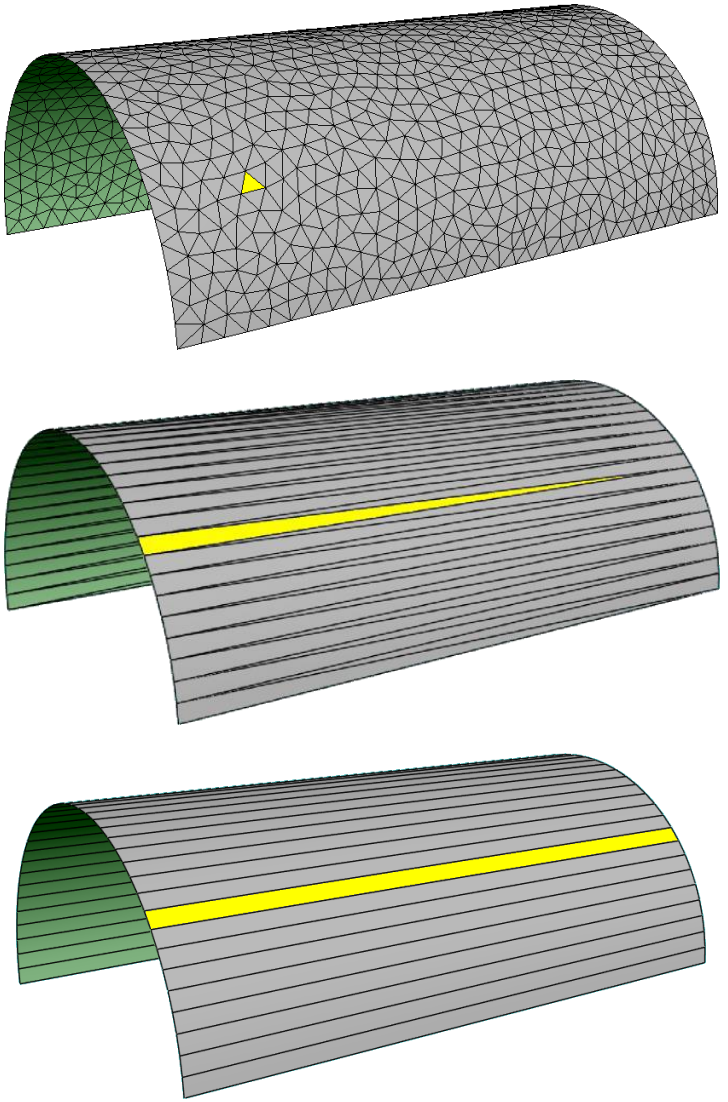
Valence close to 6

Uniform vs. adaptive sampling

Feature preservation

Alignment to curvature lines

Isotropic vs. anisotropic



What is a good mesh?

Equal edge lengths

Equilateral triangles

Valence close to 6

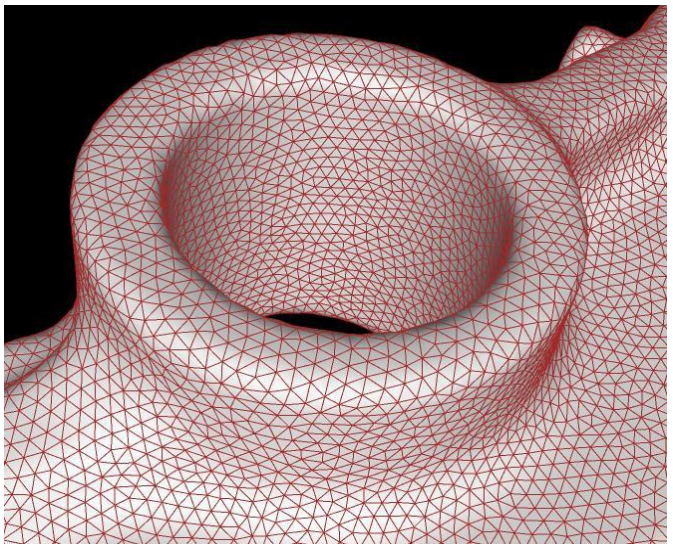
Uniform vs. adaptive sampling

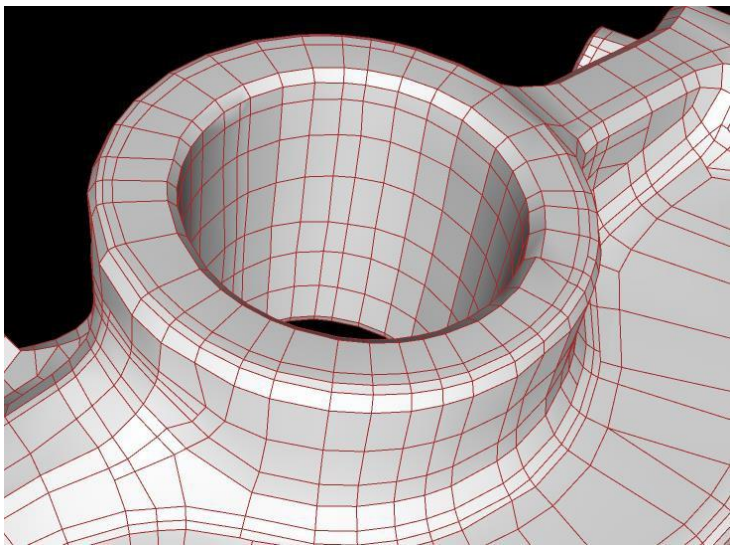
Feature preservation

Alignment to curvature lines

Isotropic vs. anisotropic

Triangles vs. quads

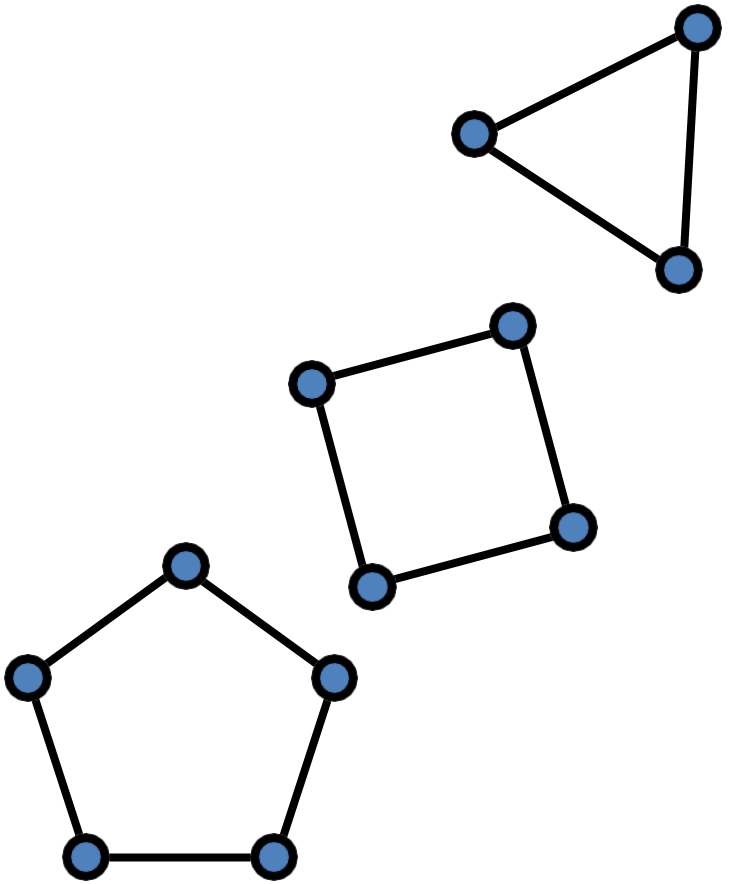




Local Structure

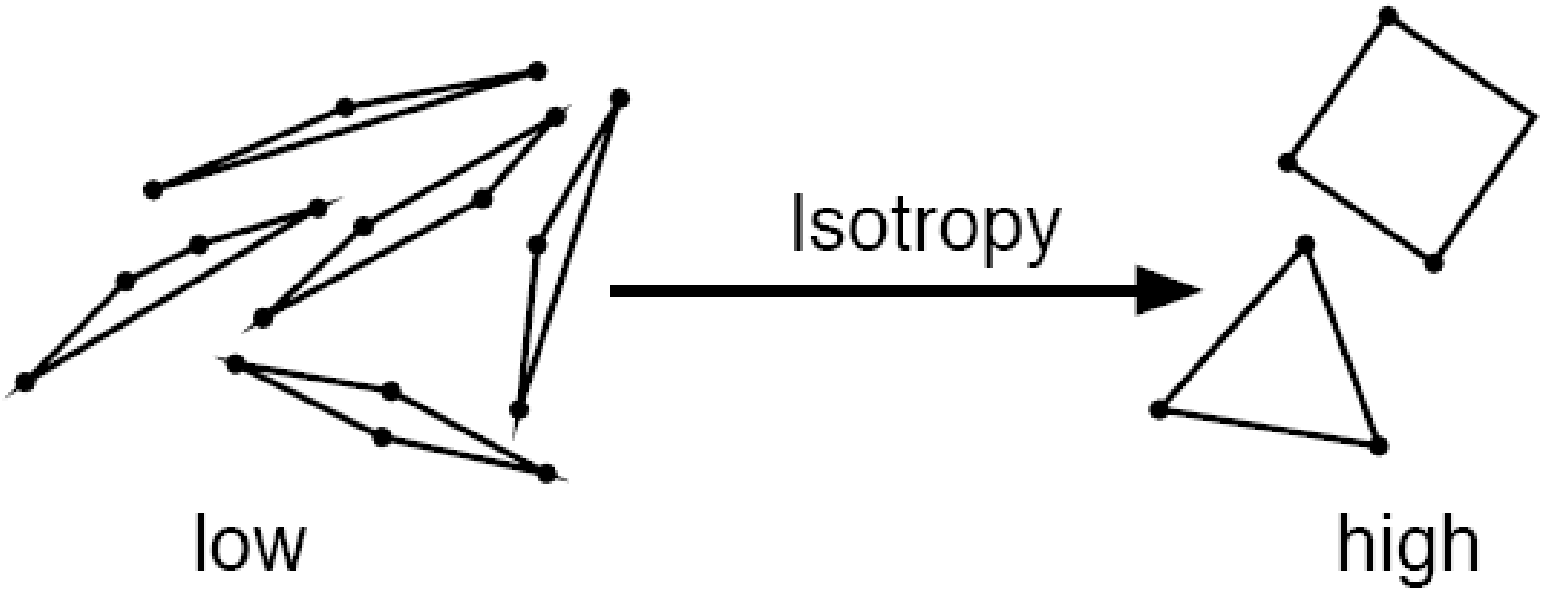
Element **type**

- Triangle
- Quadrangle
- Polygon



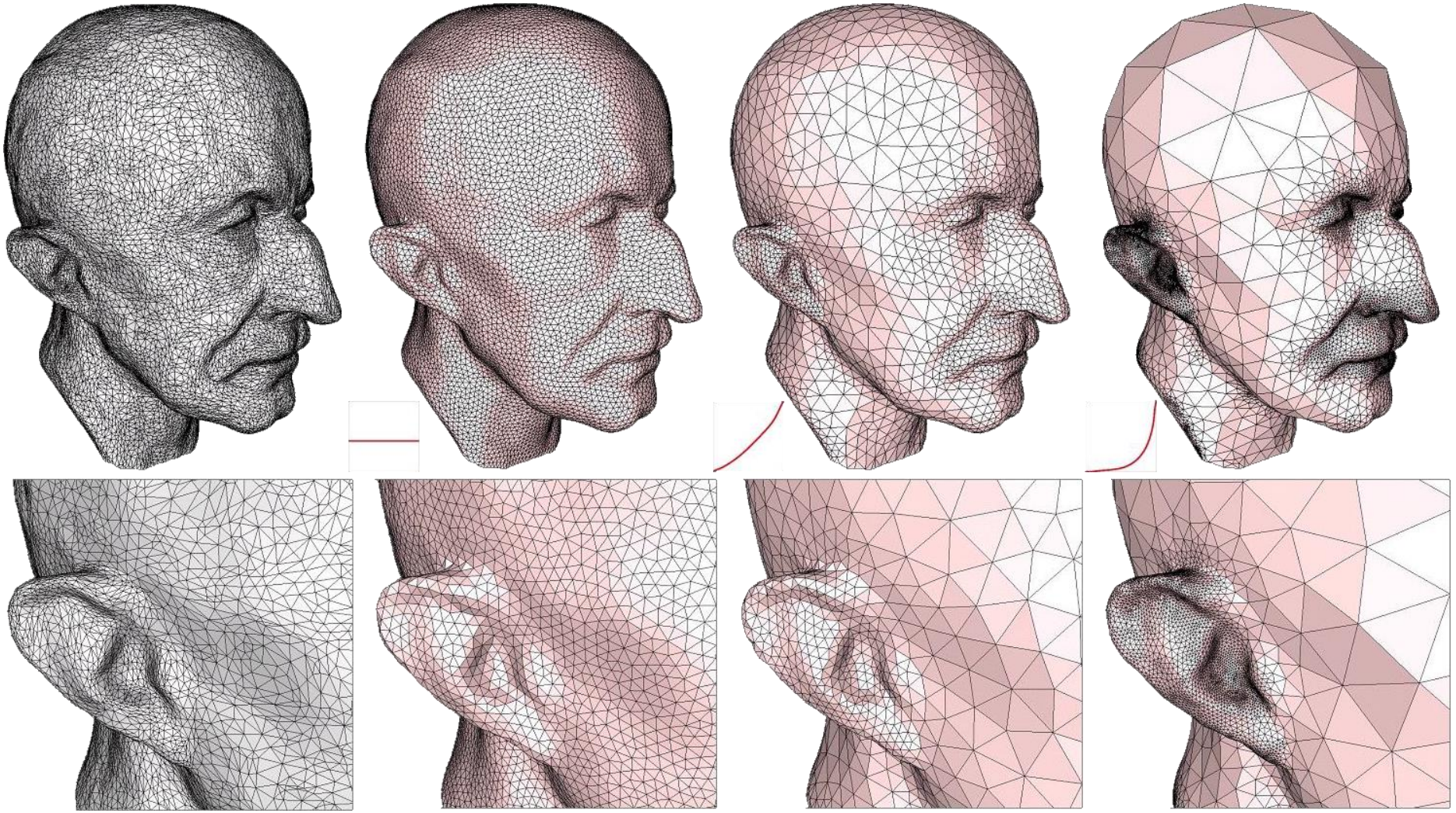
Local Structure

Element **shape** (isotropy vs anisotropy)



Local Structure

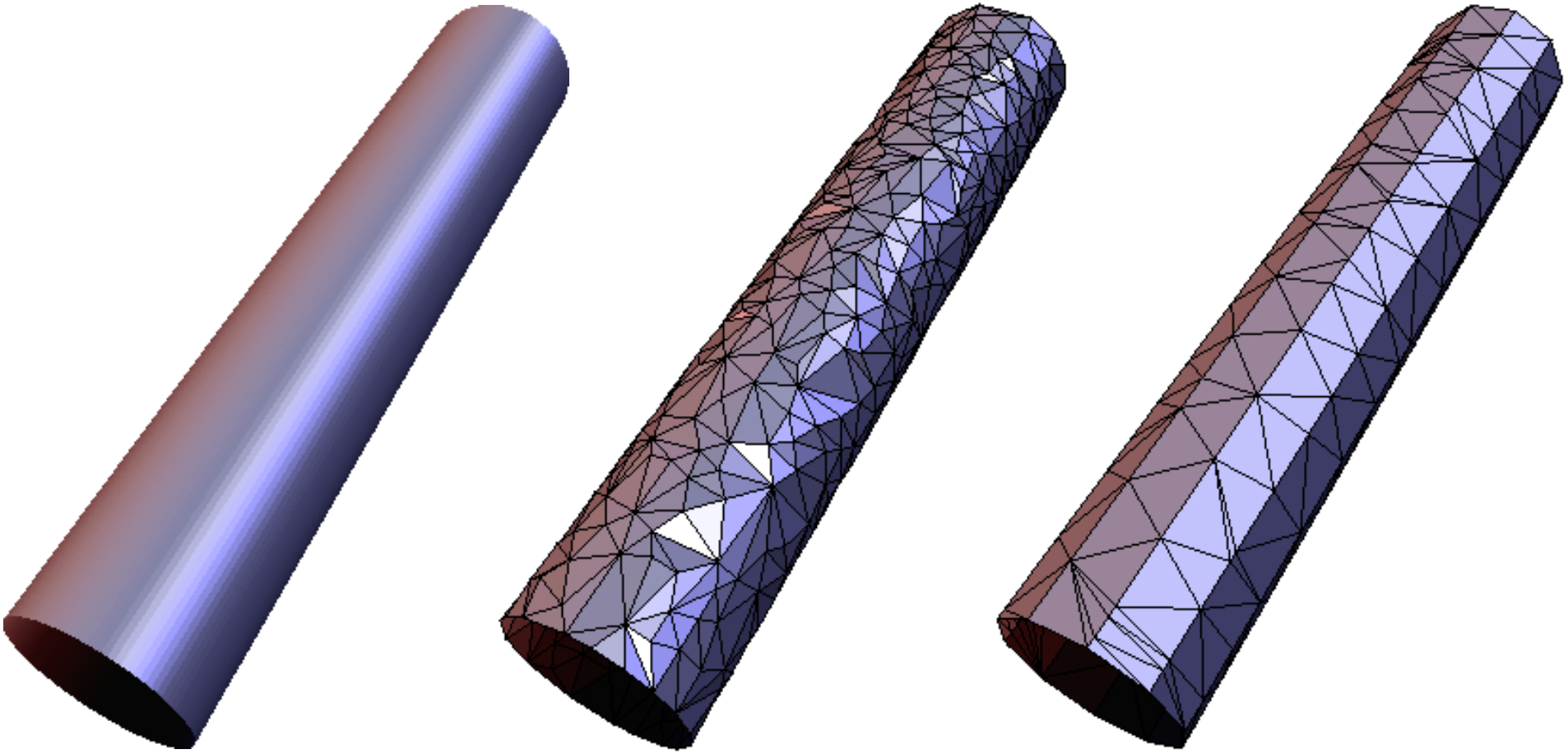
Element distribution (sizing, grading)



Input Uniform Adaptive

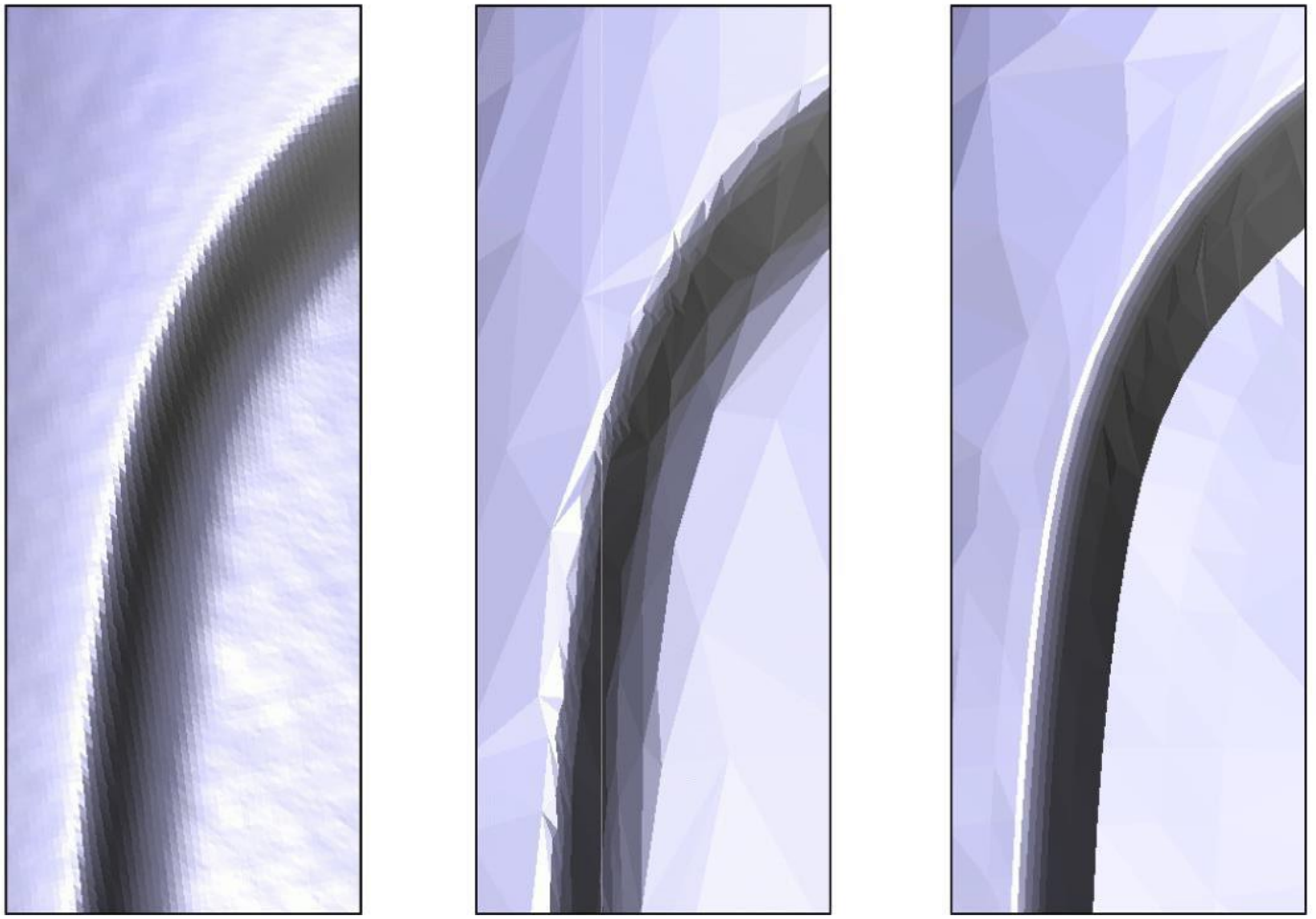
Local Structure

Element orientation



Local Structure

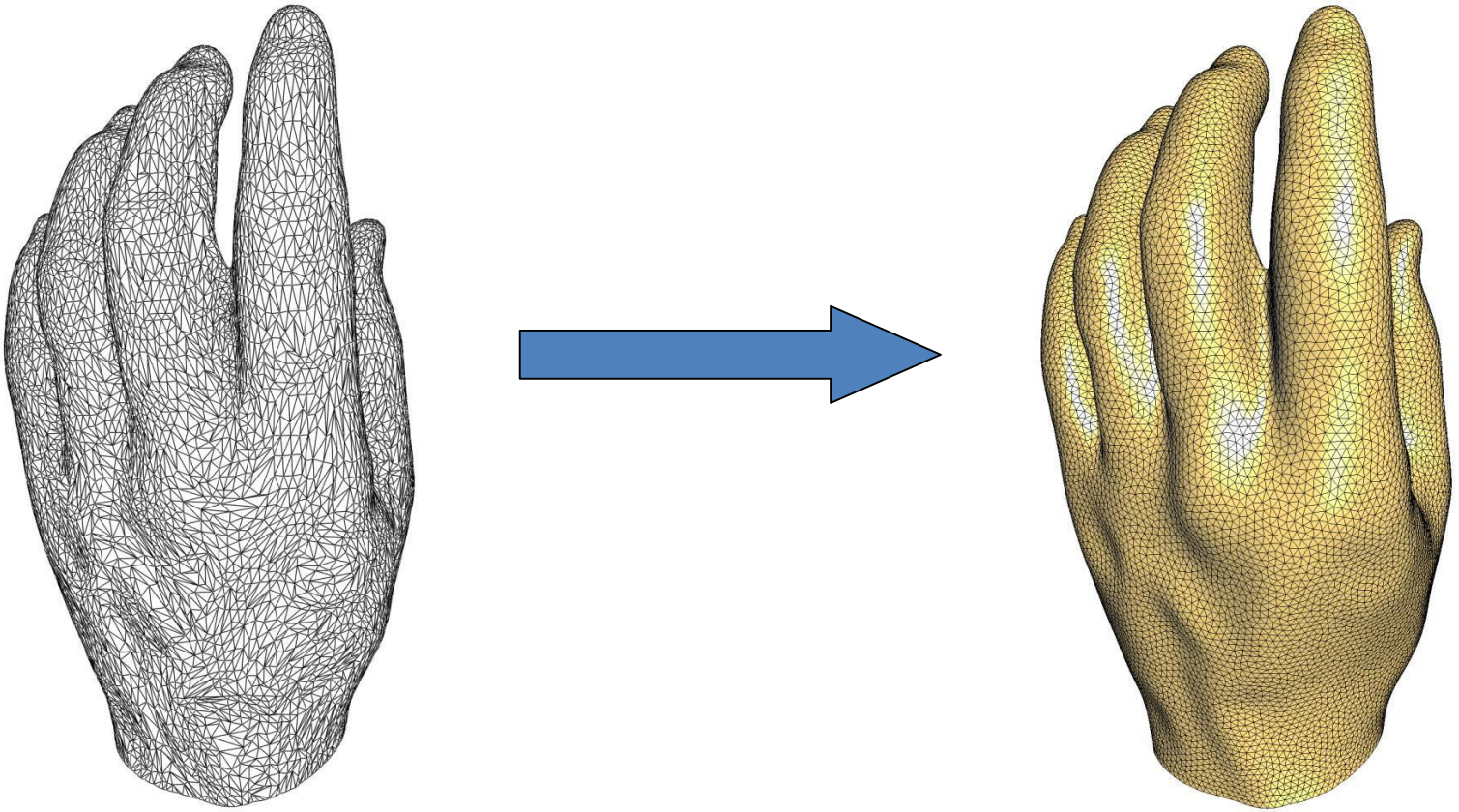
Element orientation



Isotropic Remeshing

Well-shaped elements

for processing & simulation (numerical stability & efficiency)



Direct Surface Remeshing

[Botsch et al. ’04]

Avoid global parameterization

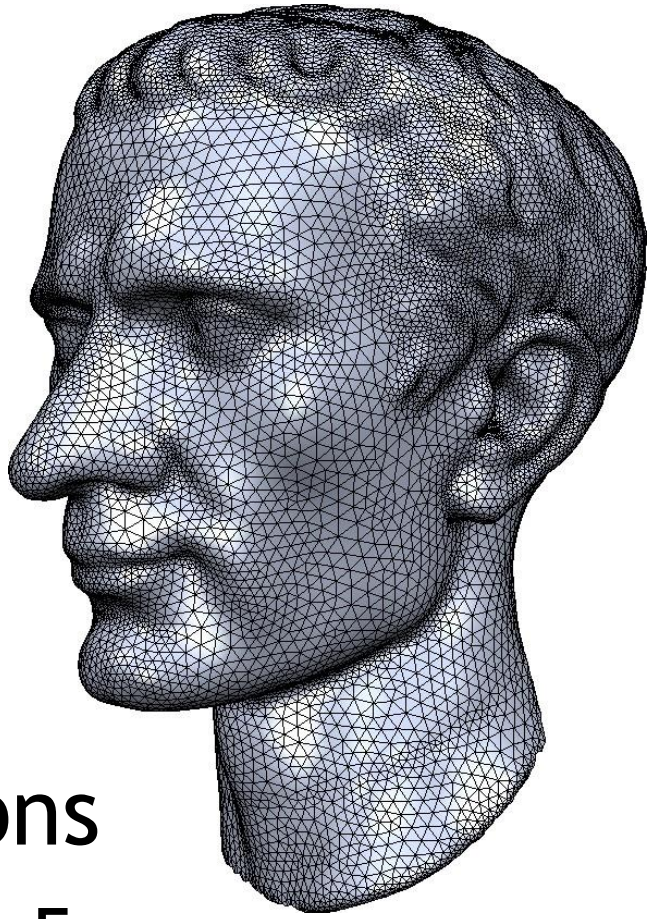
- Numerically very sensitive
- Topological restrictions

Avoid local parameterizations

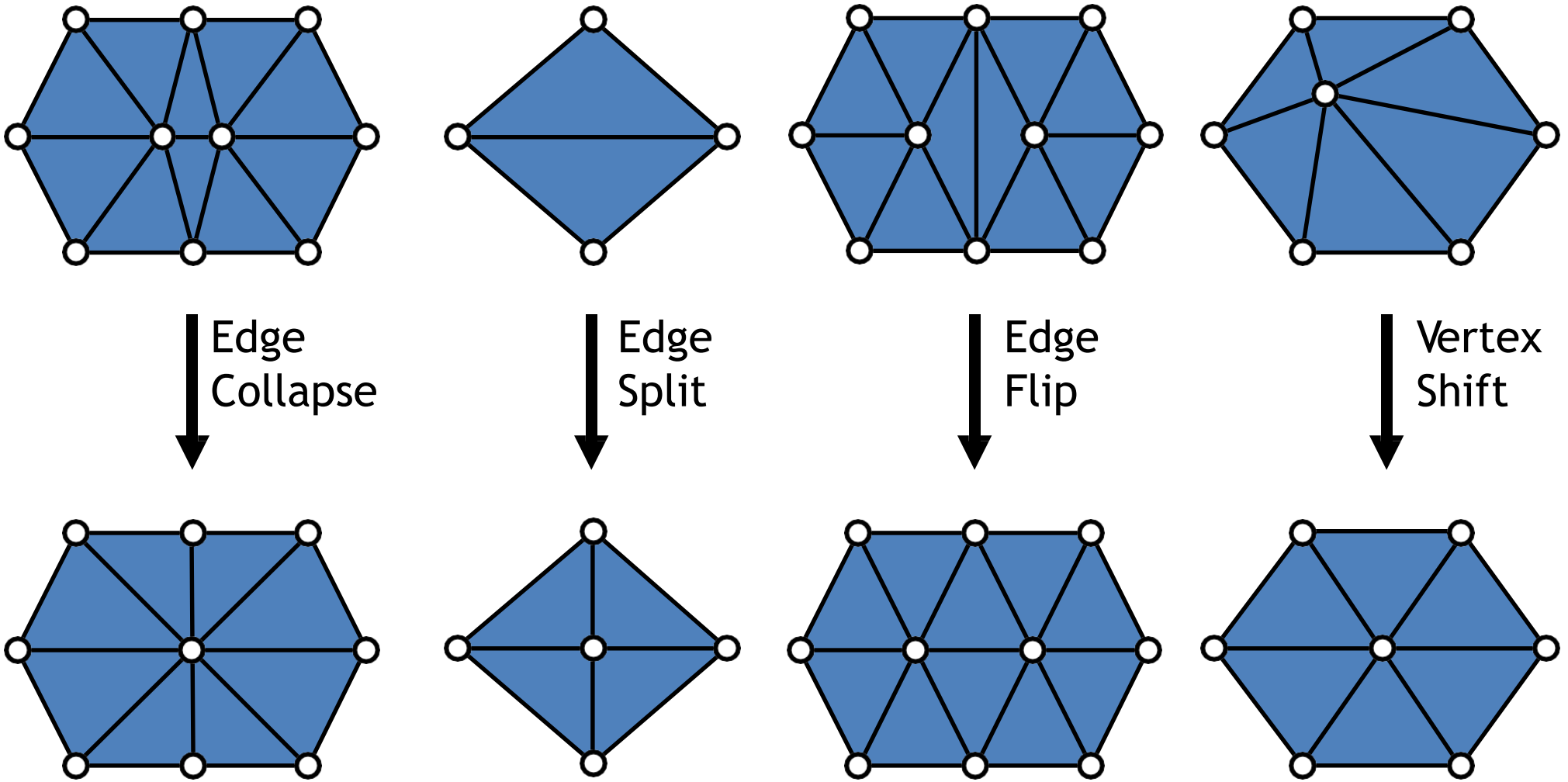
- Expensive computations

Use local operators & projections

- Resampling of 100k triangles in < 5s



Local Remeshing Operators



Isotropic Remeshing

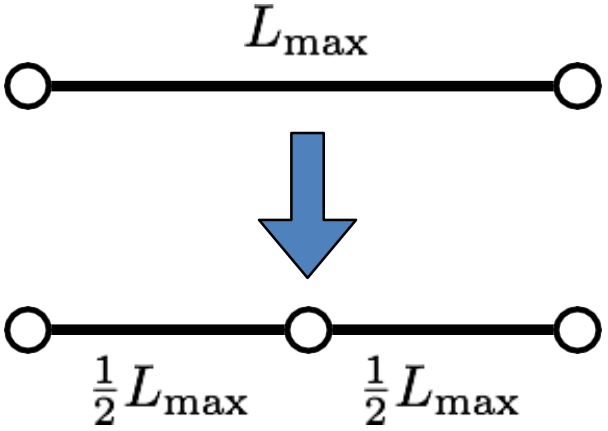
Specify target edge length L

Compute edge length range $[L_{min}, L_{max}]$

Iterate:

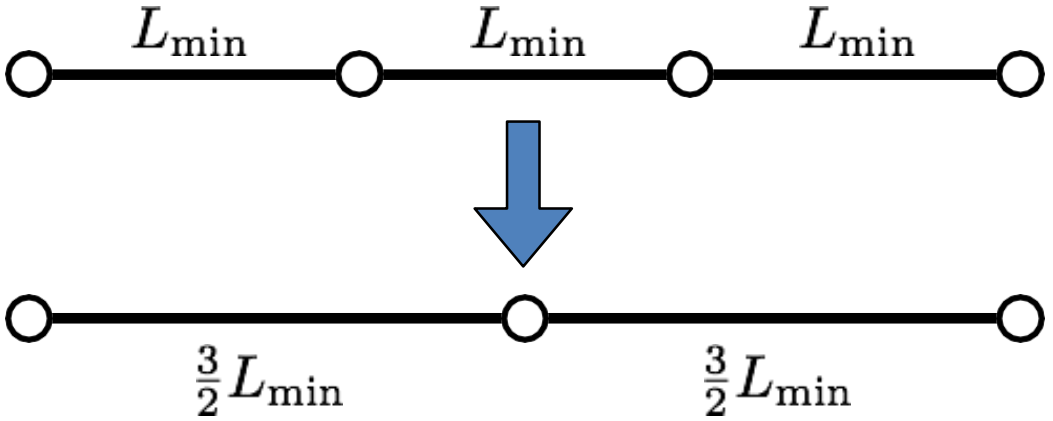
1. Split edges longer than L_{max}
2. Collapse edges shorter than L_{min}
3. Flip edges to get closer to valence 6
4. Vertex shift by tangential relaxation
5. Project vertices onto reference mesh

Edge Collapse / Split



$$|L_{\max} - L| = \left|\frac{1}{2}L_{\max} - L\right|$$

$$\Rightarrow L_{\max} = \frac{4}{3}L$$



$$|L_{\min} - L| = \left|\frac{3}{2}L_{\min} - L\right|$$

$$\Rightarrow L_{\min} = \frac{4}{5}L$$

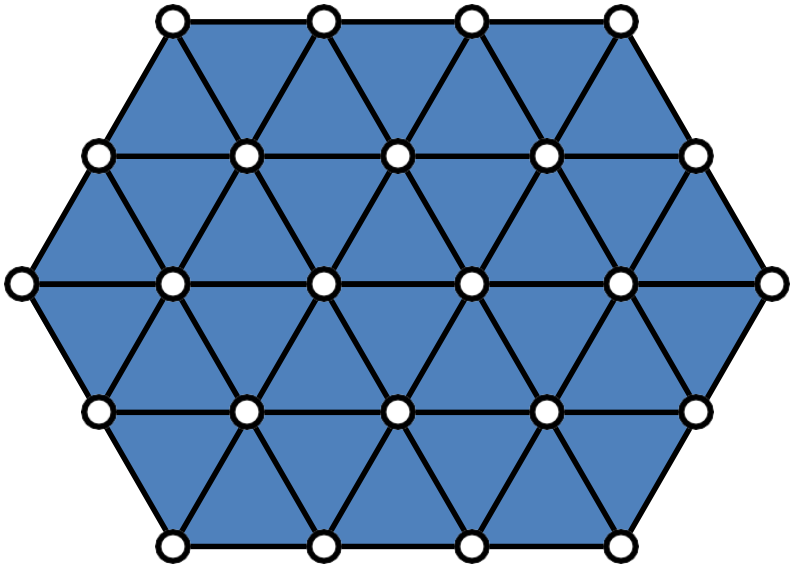
Edge Flip

Improve valences

- Avg. valence is 6 (Euler)
- Reduce variation

Optimal valence is

- 6 for interior vertices
- 4 for boundary vertices



Edge Flip

Improve valences

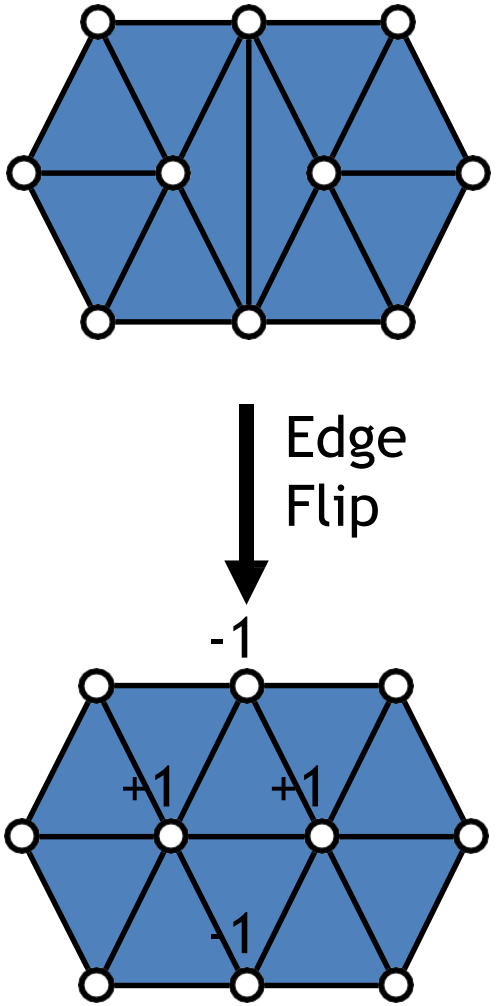
- Avg. valence is 6 (Euler)
- Reduce variation

Optimal valence is

- 6 for interior vertices
- 4 for boundary vertices

Minimize valence excess

$$\sum_{i=1}^{4} (\text{valence}(v_i) - \text{opt\_valence}(v_i))^2$$

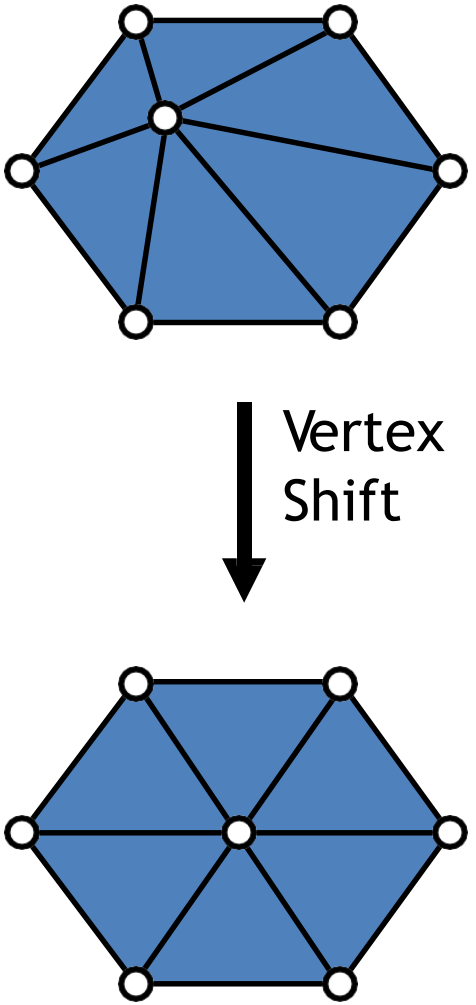


Vertex Shift

Local “spring” relaxation

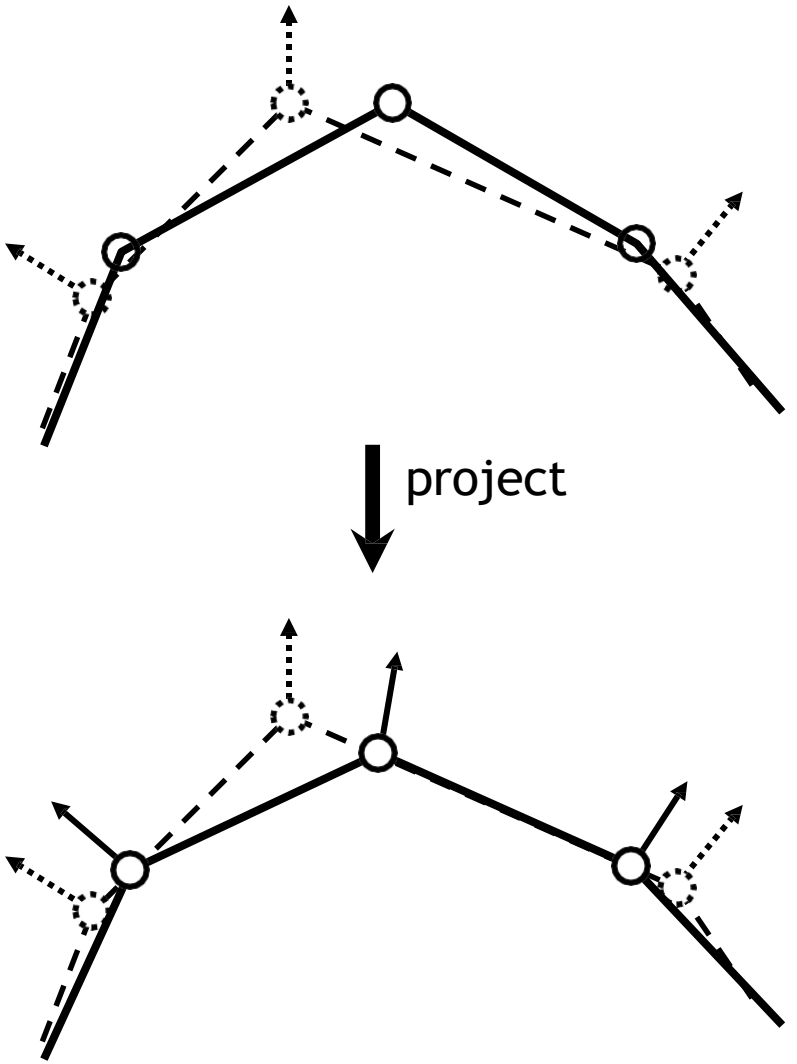
- Uniform Laplacian smoothing
- Bary-center of one-ring neighbors

$$\mathbf{c}_i = \frac{1}{\text{valence}(v_i)} \sum_{j \in N(v_i)} \mathbf{p}_j$$

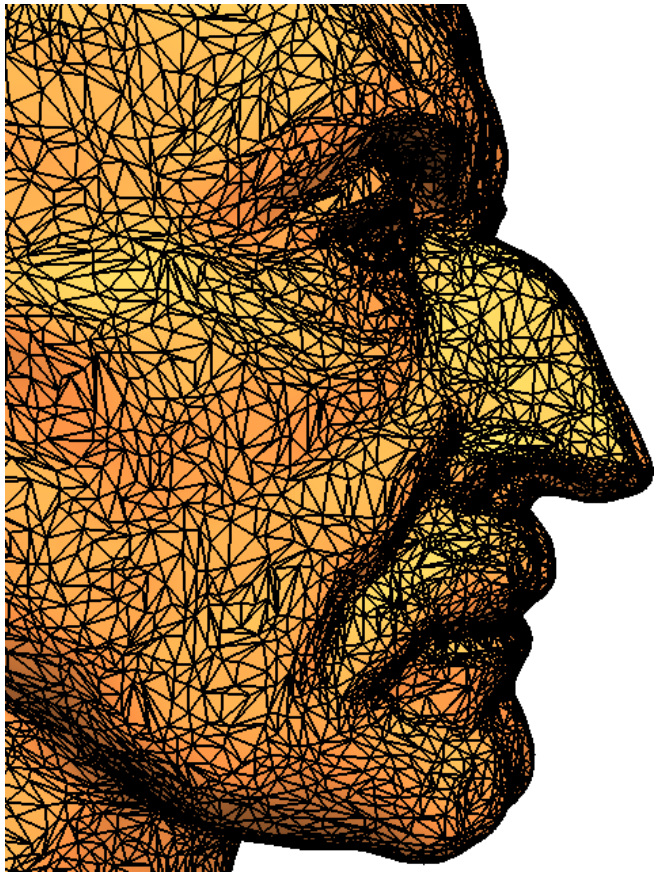


Vertex Projection

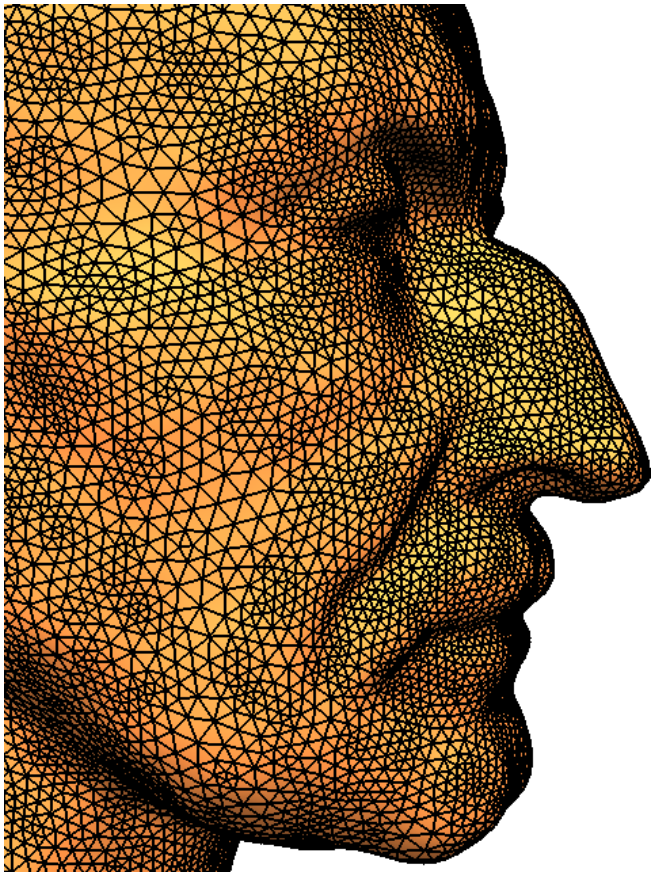
- Project vertices onto original reference mesh
- Assign position & interpolated normal



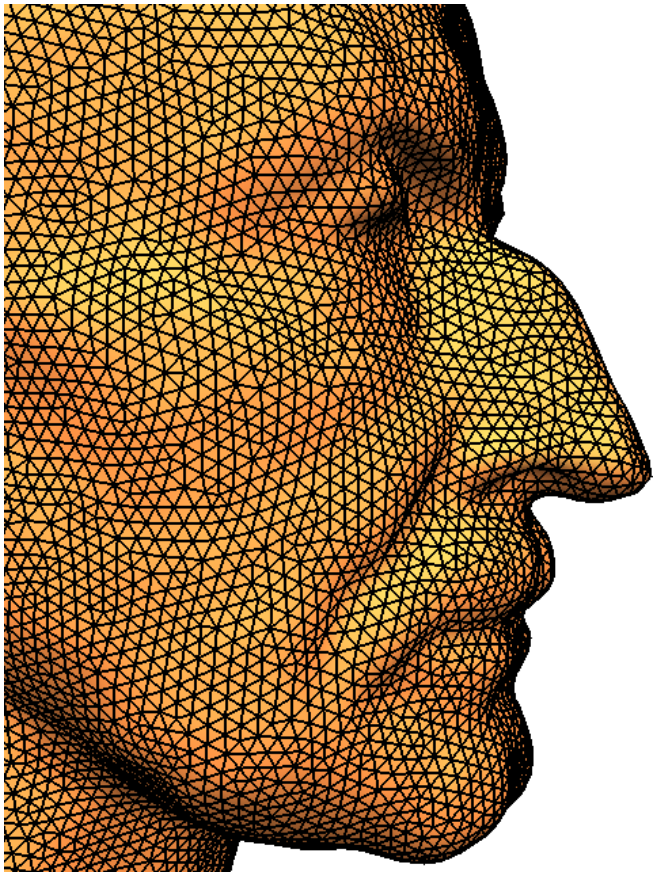
Remeshing Results



Original

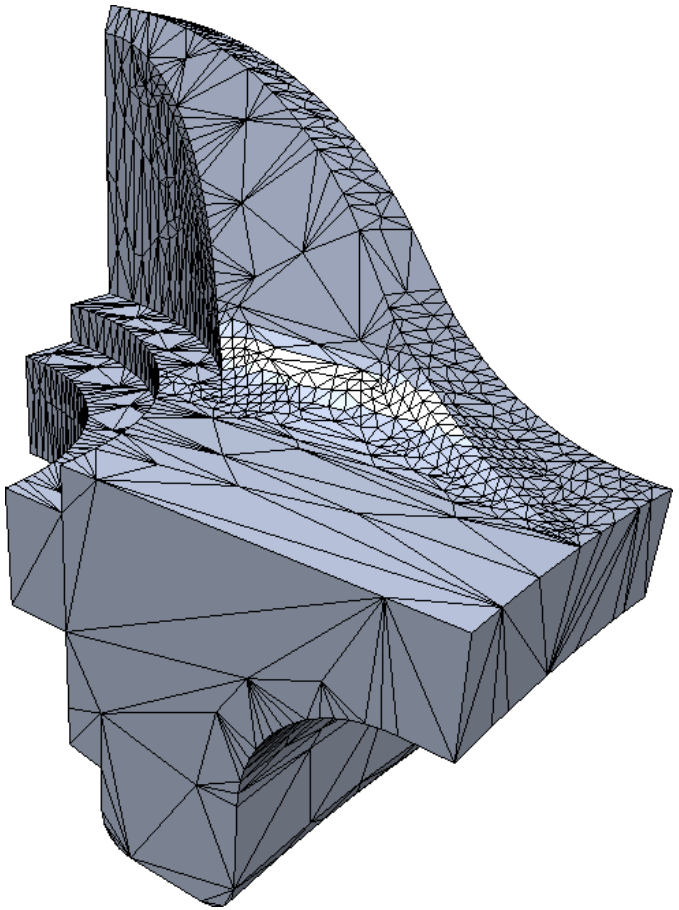


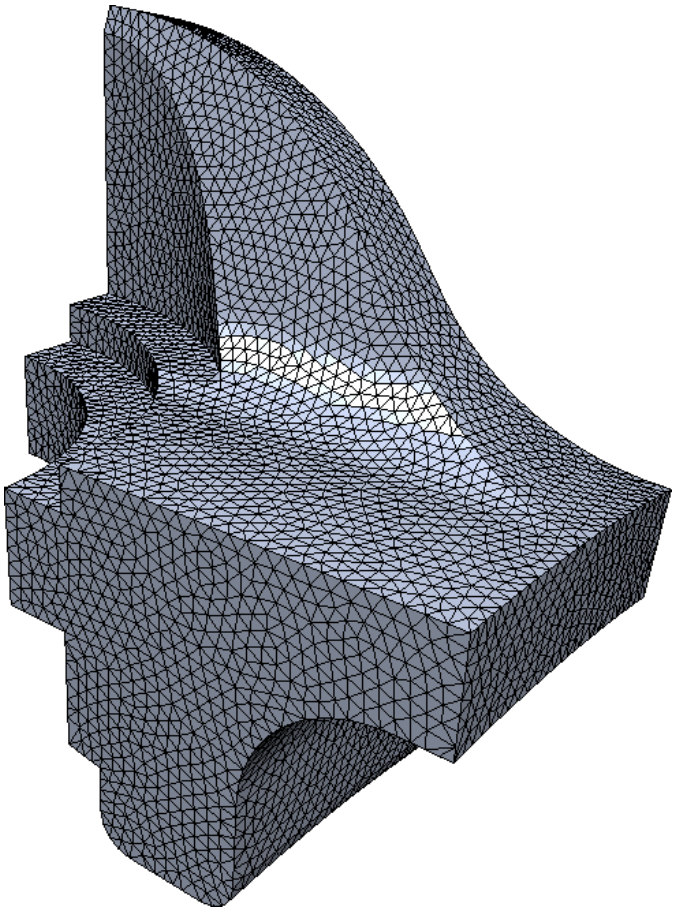
$\left(\frac{1}{2}, 2\right)$



$\left(\frac{4}{5}, \frac{4}{3}\right)$

Feature Preservation?





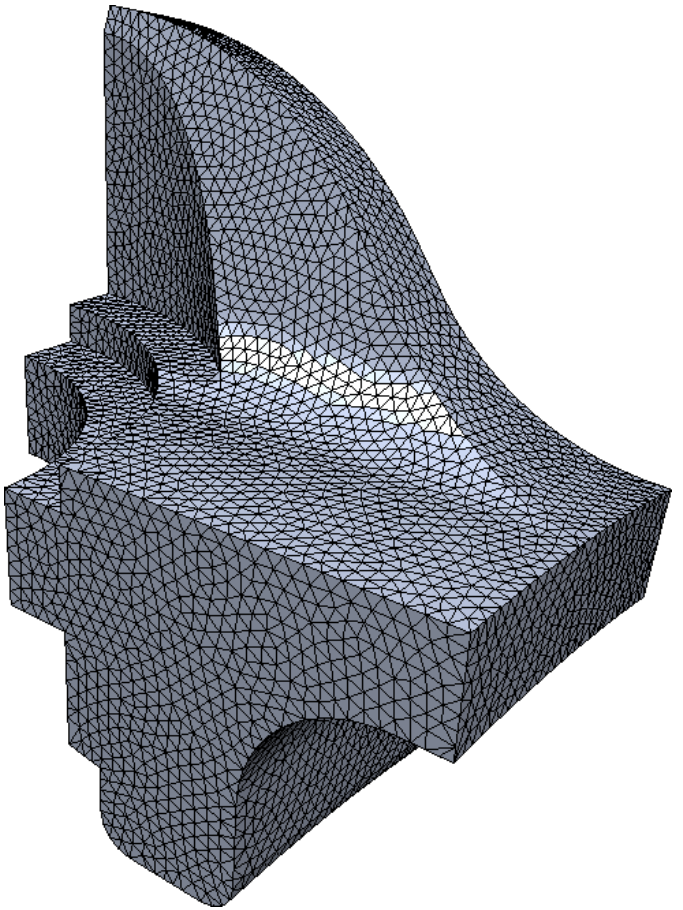
Feature Preservation

Define features

- Sharp edges
- Material boundaries

Adjust local operators

- Don’t move corners
- Collapse only along features
- Don’t flip feature edges
- Project to feature curves

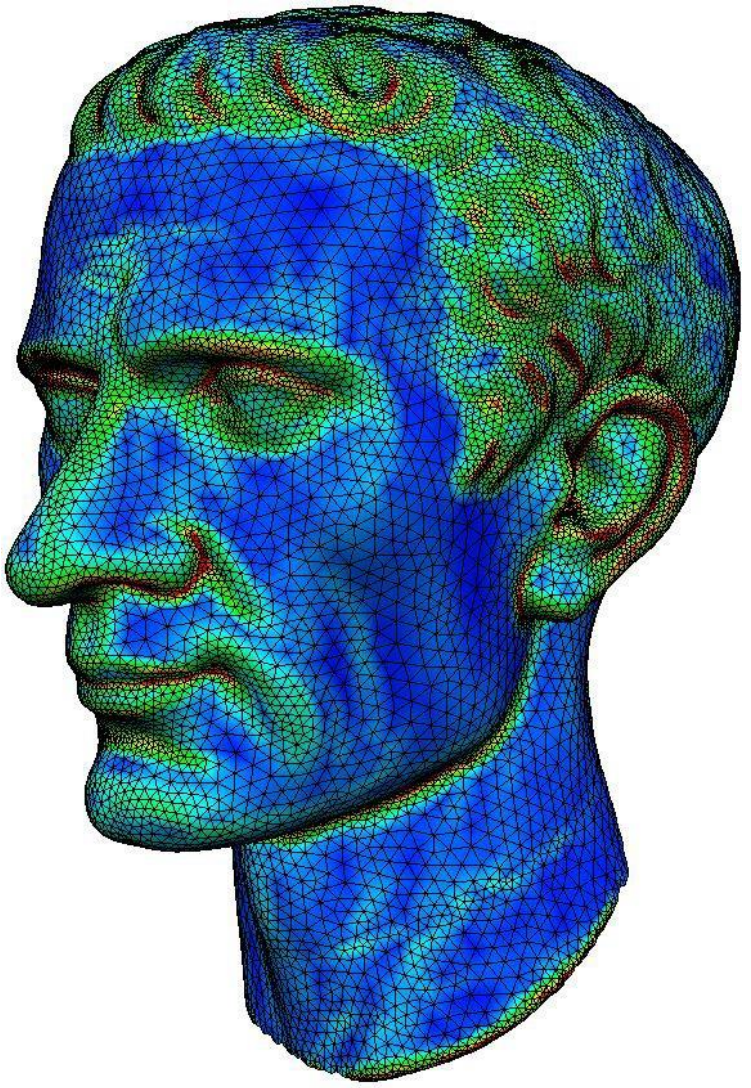


Adaptive Remeshing

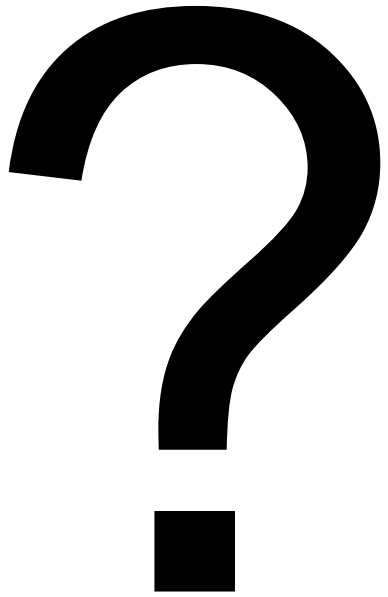
Precompute max. curvature on reference mesh

Target edge length locally determined by curvature

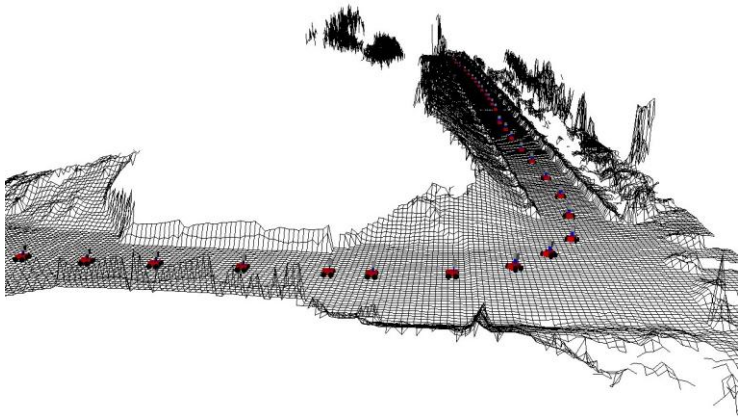
Adjust split / collapse criteria

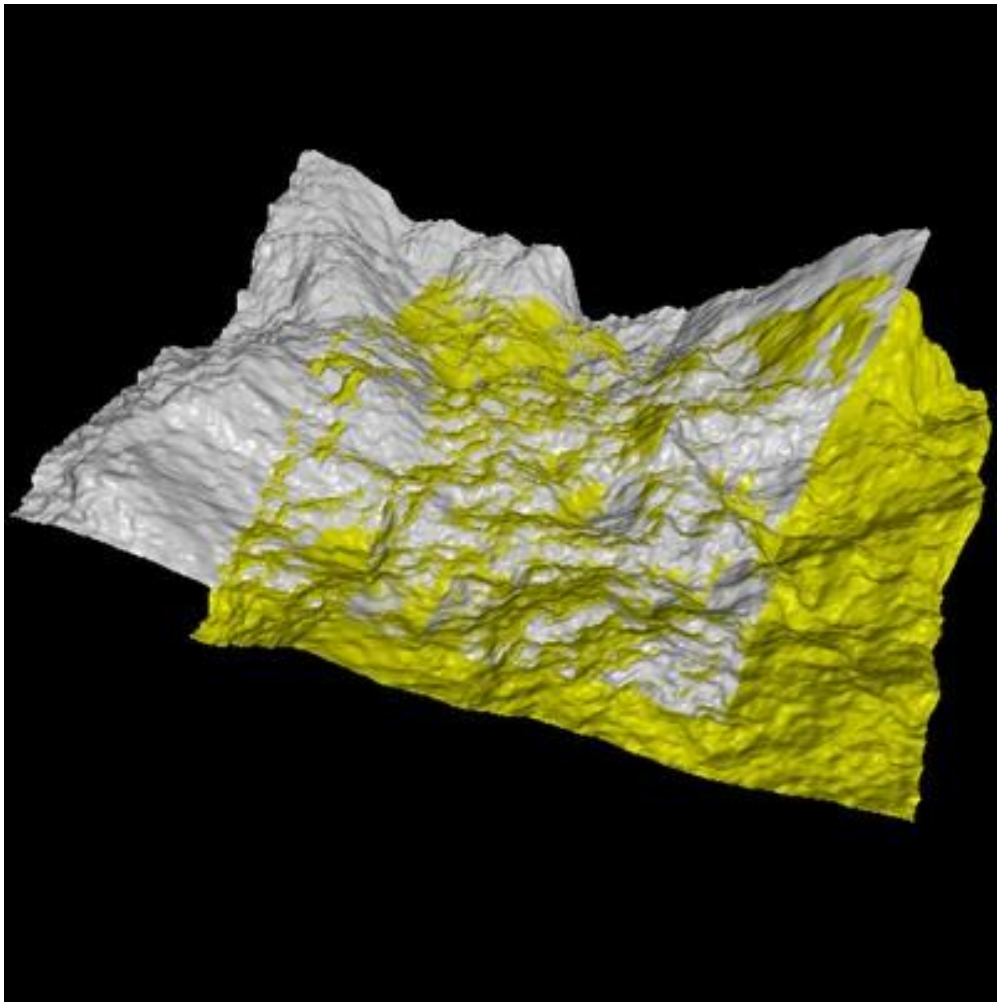


Questions ?



Motivation





Goal: Find local transformation to align points

The Problem

- Given two corresponding point sets:

$$X = \{x_1, ..., x_{N_x}\}$$

$$P = \{p_1, ..., p_{N_p}\}$$

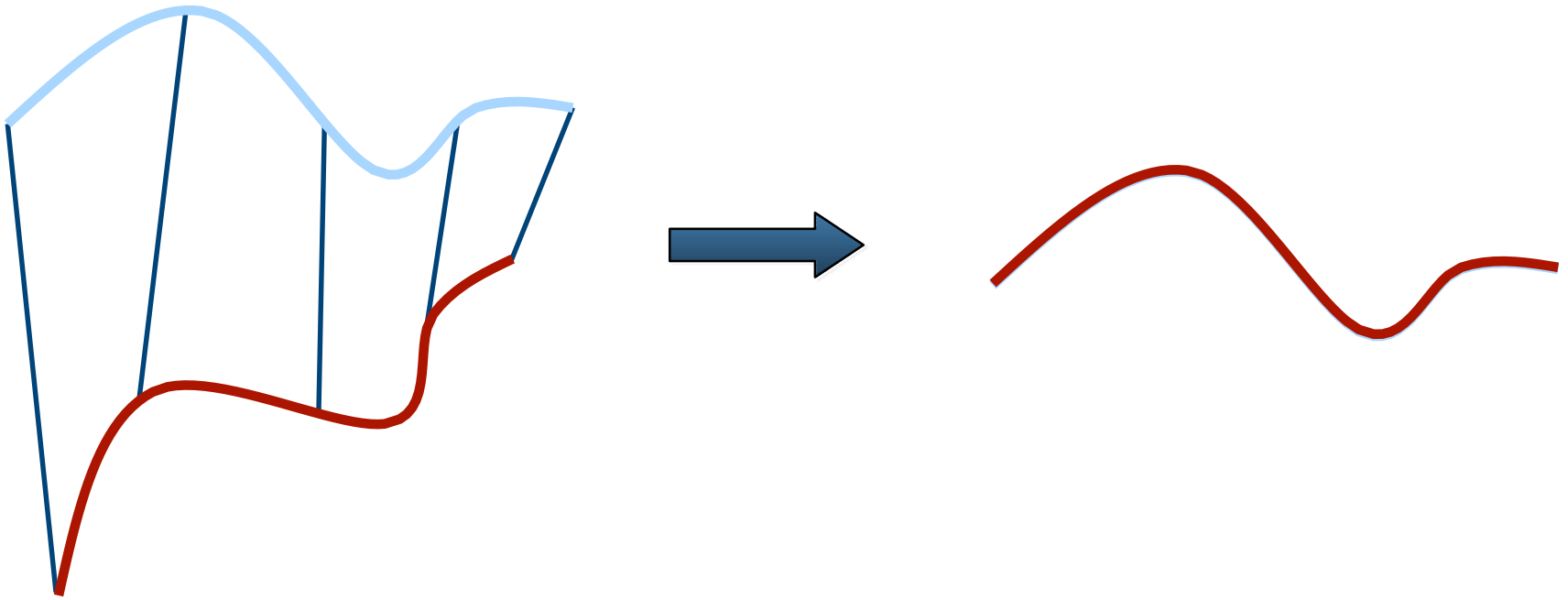
- Wanted: Translation *t* and rotation *R* that minimize the sum of the squared error:

$$E(R,t) = \frac{1}{N_p} \sum_{i=1}^{N_p} ||x_i - Rp_i - t||^2$$

Where x_i and p_i are corresponding points

Key Idea

- If the correct correspondences are known, the correct relative rotation/translation can be calculated in **closed form**



Center of Mass

$$\mu_x = \frac{1}{N_x}\sum_{i=1}^{N_x} x_i \quad \text{and} \quad \mu_p = \frac{1}{N_p}\sum_{i=1}^{N_p} p_i$$

are the centers of mass of the two point sets

Idea:

- Subtract the corresponding center of mass from every point in the two point sets before calculating the transformation
- The resulting point sets are:

$$X' = \{x_i - \mu_x\} = \{x'_i\}$$
$$P' = \{p_i - \mu_p\} = \{p'_i\}$$

and

Singular Value Decomposition

Let $W = \sum_{i=1}^{N_p} x'_i p'^T_i$

denote the singular value decomposition (SVD) of W by:

$$W = U \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} V^T$$

where $U, V \in \mathbb{R}^{3\times3}$ are unitary, and

$\sigma_1 \geq \sigma_2 \geq \sigma_3$ are the singular values of W

SVD

Theorem (without proof):

If rank(*W*) = 3, the optimal solution of E(*R*,*t*) is unique and is given by:

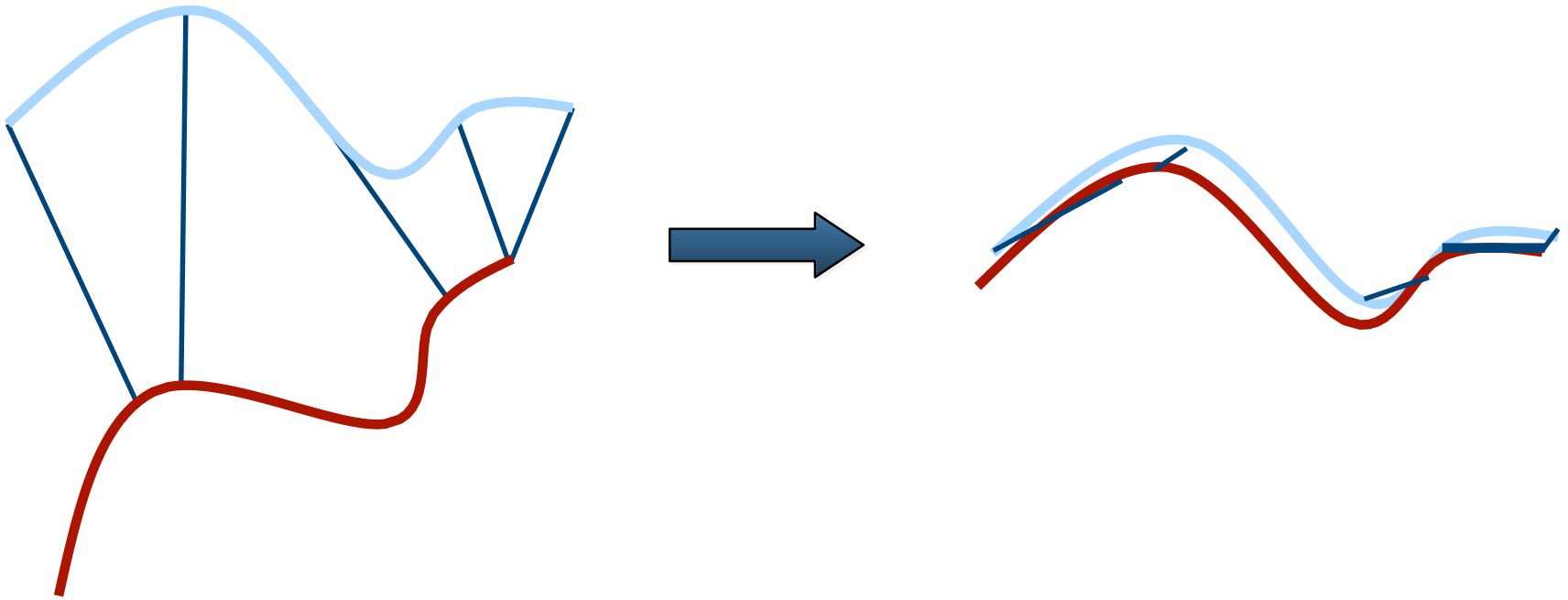
$$R = UV^T$$
$$t = \mu_x - R\mu_p$$

The minimal value of error function at (*R*,*t*) is:

$$E(R,t) = \sum_{i=1}^{N_p} (||x_i'||^2 + ||y_i'||^2) - 2(\sigma_1 + \sigma_2 + \sigma_3)$$

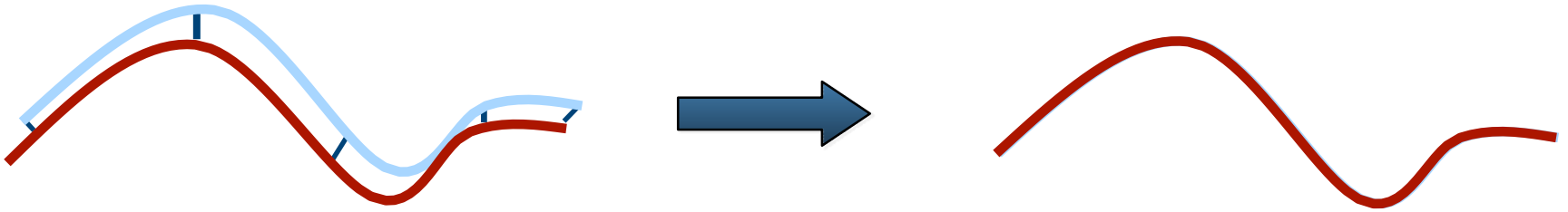
ICP with Unknown Data Association

- If the correct correspondences are **not known**, it is generally impossible to determine the optimal relative rotation/ translation in one step



Iterative Closest Point (ICP) Algorithm

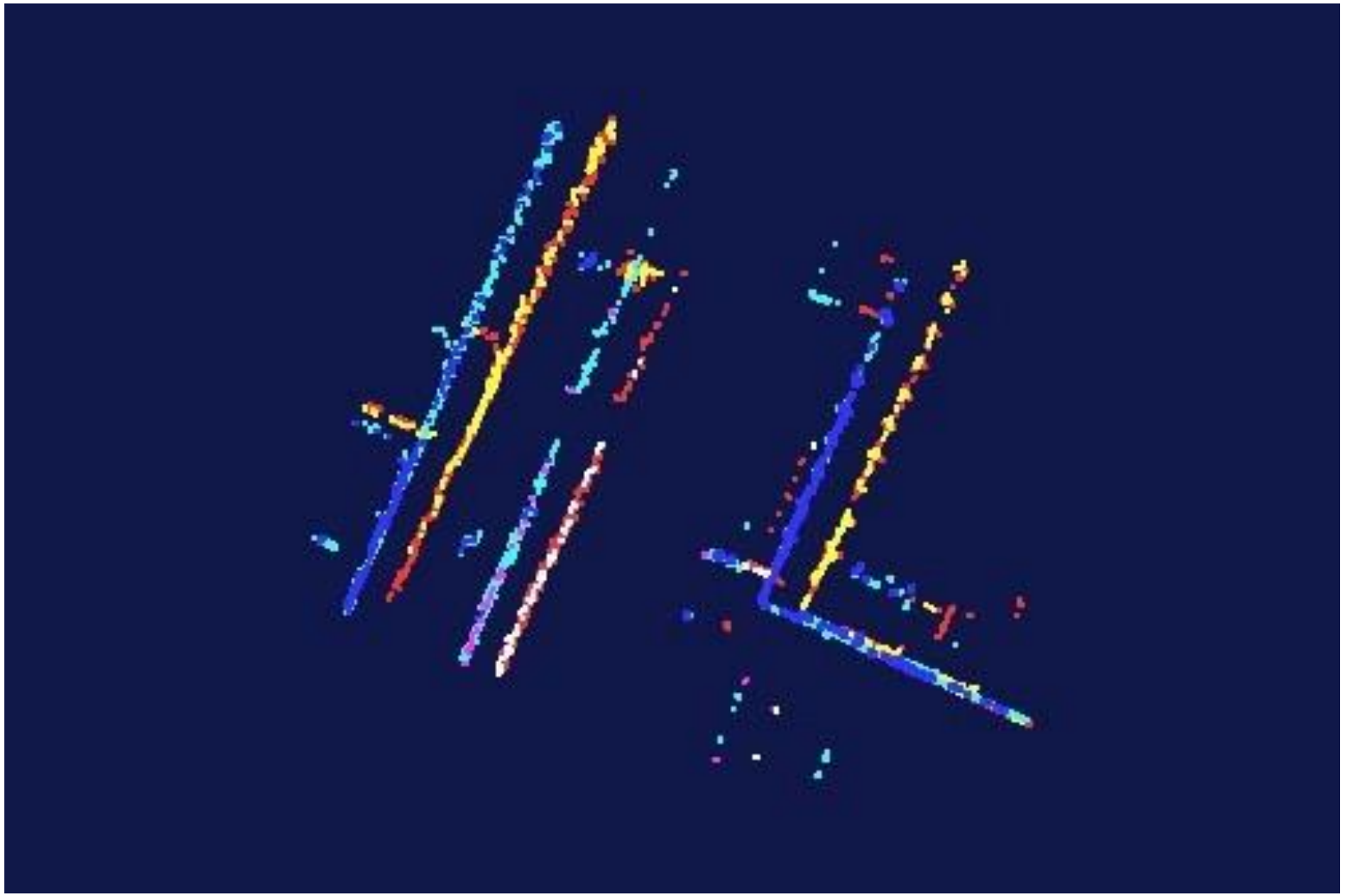
- Idea: Iterate to find alignment
- Iterative Closest Points
  [Besl & McKay 92]
- Converges if starting positions are "close enough"



Basic ICP Algorithm

- Determine corresponding points
- Compute rotation R, translation t via SVD
- Apply R and t to the points of the set to be registered
- Compute the error $E(R,t)$
- If error decreased and error > threshold
  - Repeat these steps
  - Stop and output final alignment, otherwise

ICP Example



ICP Variants

Variants on the following stages of ICP have been proposed:

1. Point subsets (from one or both point sets)
2. Weighting the correspondences
3. Data association
4. Rejecting certain (outlier) point pairs

Performance of Variants

- Various aspects of performance:
  - Speed
  - Stability (local minima)
  - Tolerance wrt. noise and outliers
  - Basin of convergence
    (maximum initial misalignment)

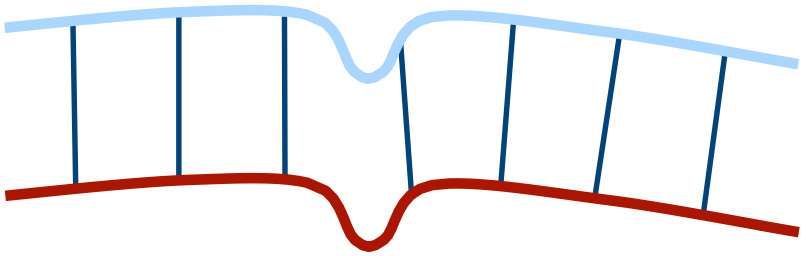
ICP Variants

1. Point subsets (from one or both point sets)
2. Weighting the correspondences
3. Data association
4. Rejecting certain (outlier) point pairs

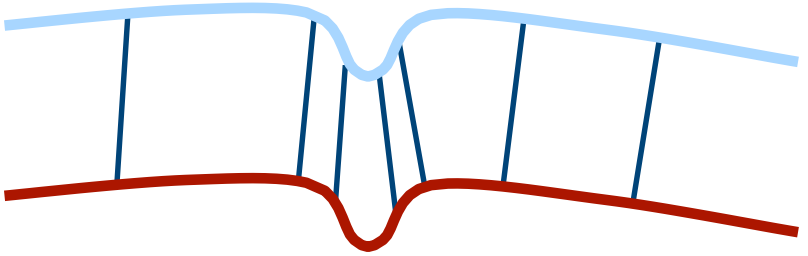
Selecting Source Points

- Use all points
- Uniform sub-sampling
- Random sampling
- Feature based sampling
- Normal-space sampling
  (Ensure that samples have normals distributed as uniformly as possible)

Normal-Space Sampling



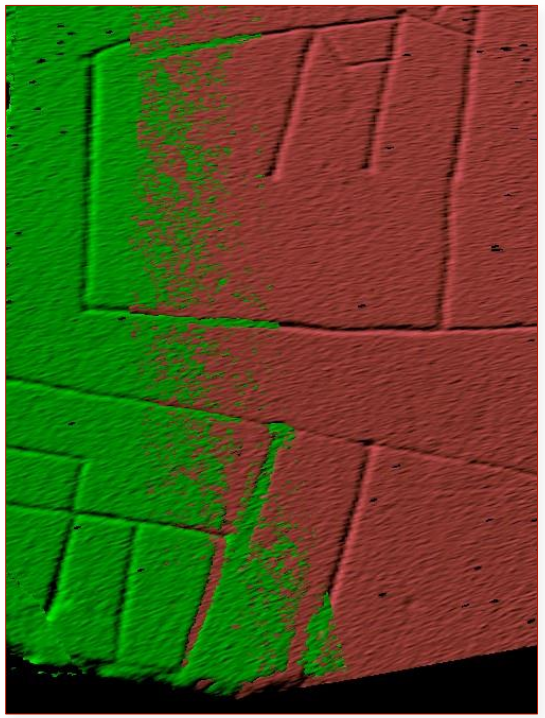
uniform sampling



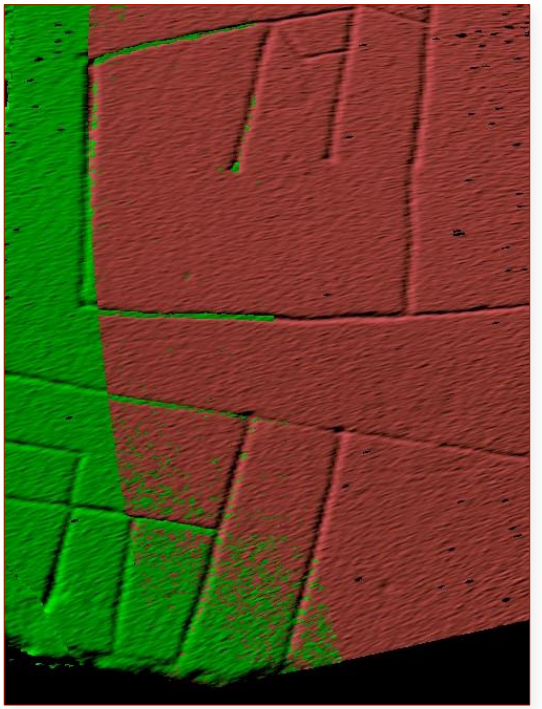
normal-space sampling

Comparison

- Normal-space sampling better for mostly smooth areas with sparse features
  [Rusinkiewicz et al., 01]



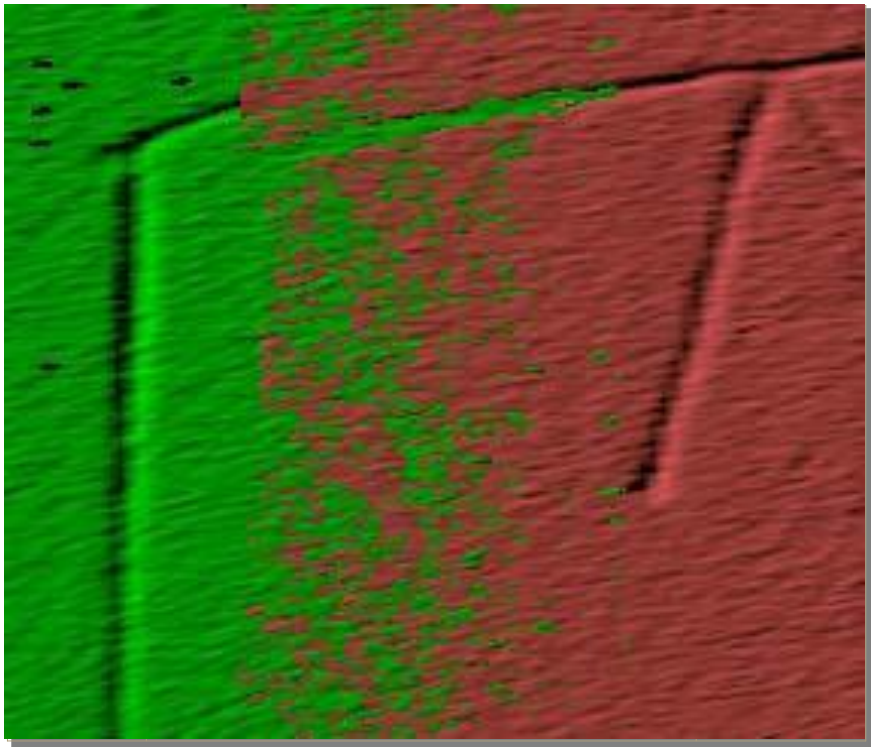
Random sampling



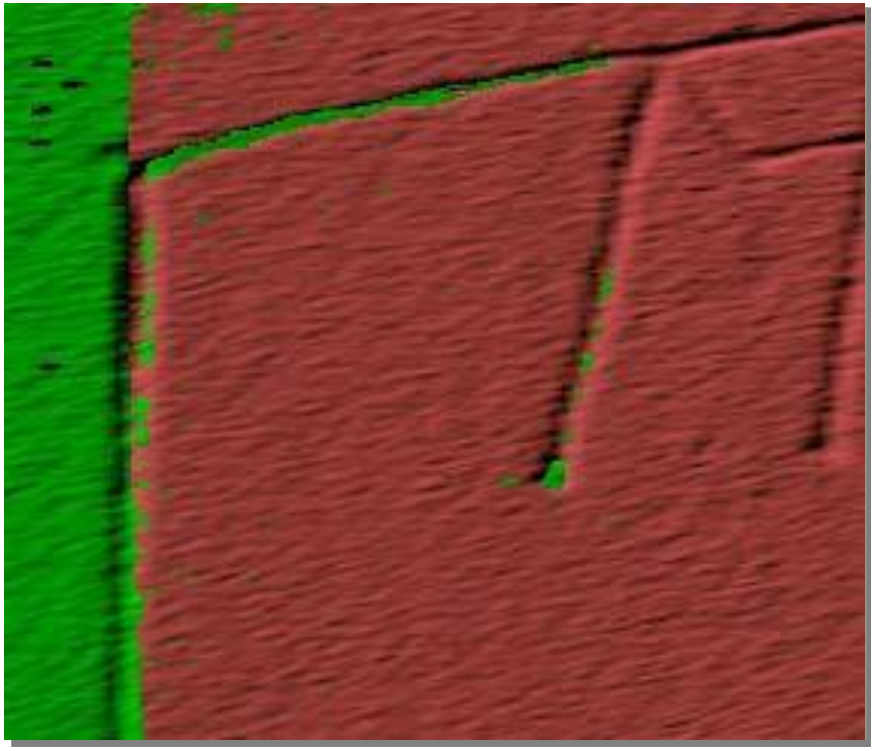
Normal-space sampling

Comparison

- Normal-space sampling better for mostly smooth areas with sparse features
  [Rusinkiewicz et al., 01]



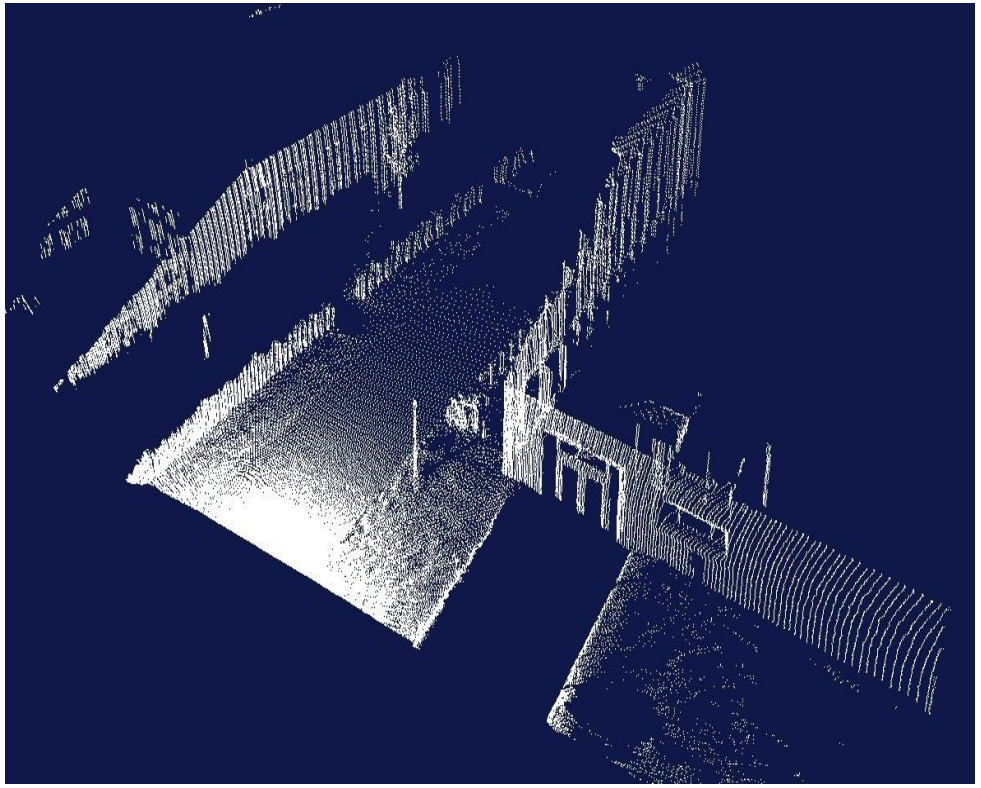
Random sampling



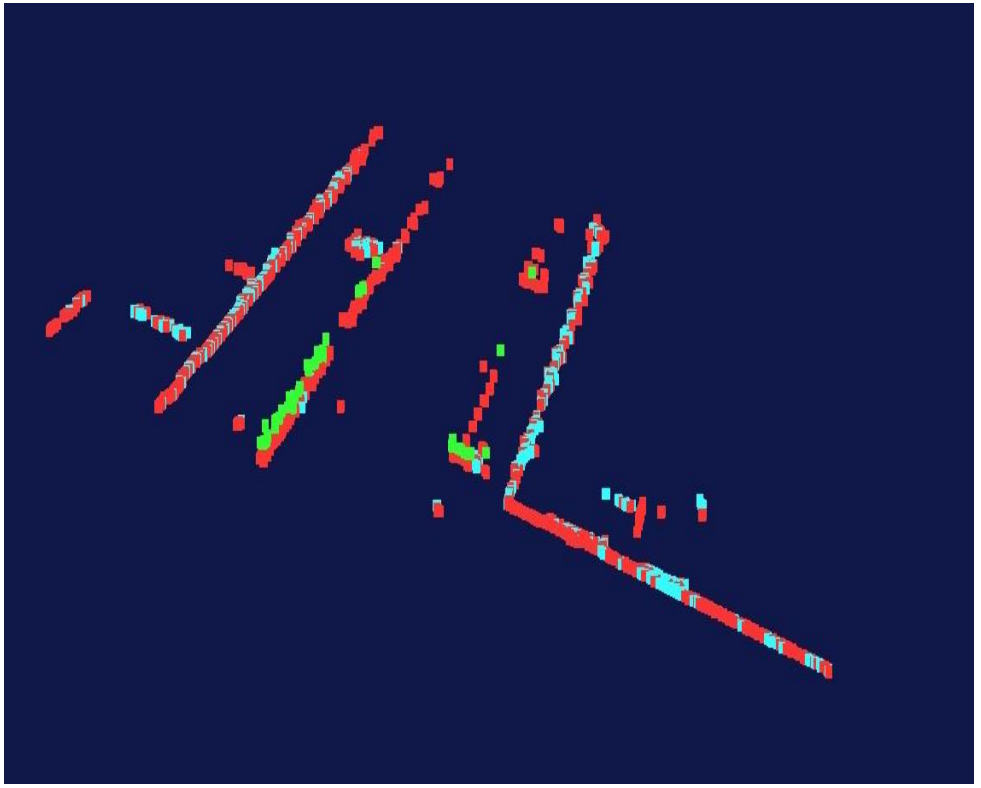
Normal-space sampling

Feature-Based Sampling

- Try to find “important” points
- Decreases the number of correspondences to find
- Higher efficiency and higher accuracy
- Requires preprocessing

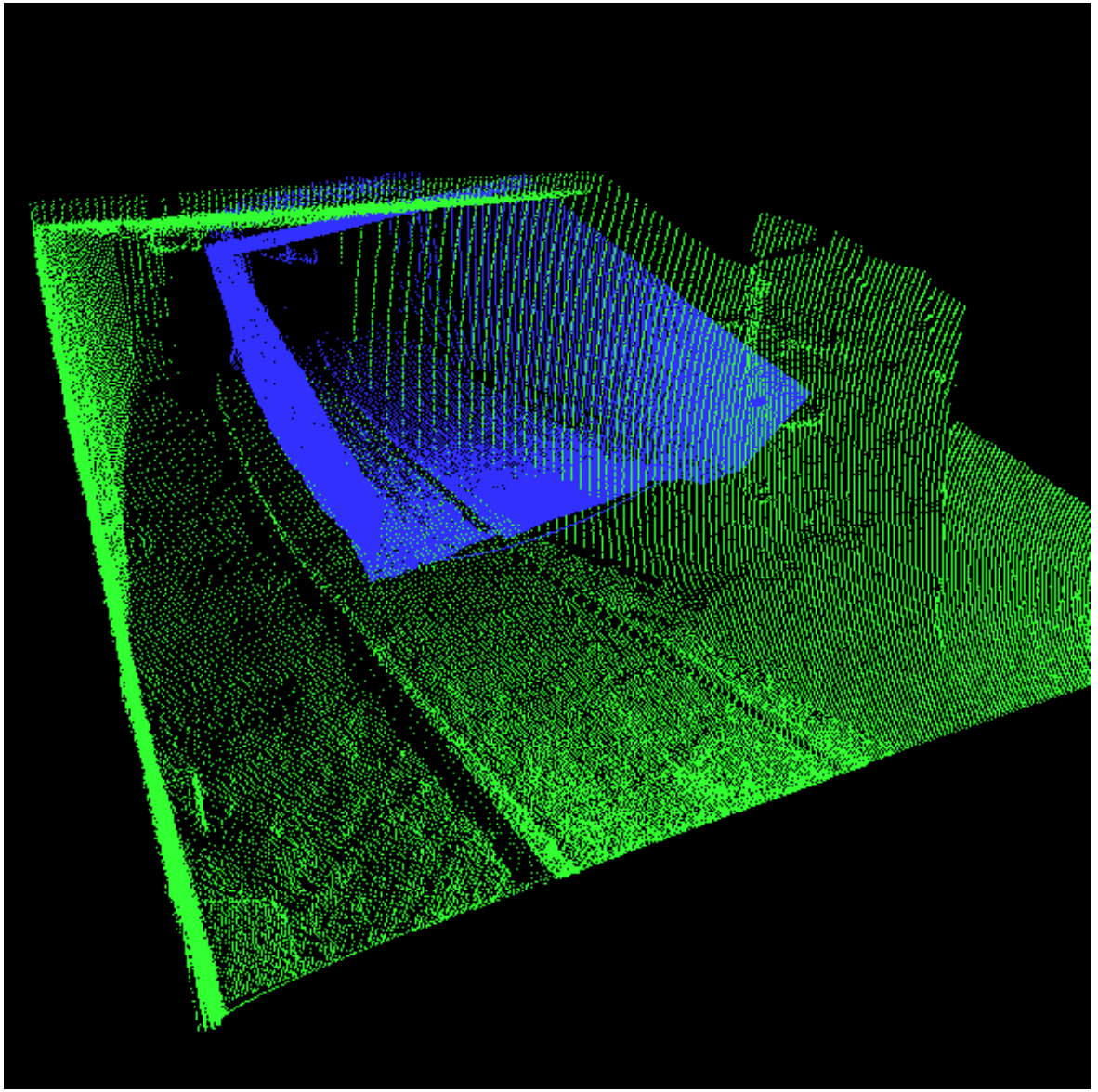


3D Scan (~200.000 Points)



Extracted Features (~5.000 Points)

ICP Application (With Uniform Sampling)



[Nuechter et al., 04]

ICP Variants

1. Point subsets (from one or both point sets)
2. Weighting the correspondences
3. Data association
4. Rejecting certain (outlier) point pairs

Weighting

- Select a set of points for each set
- Match the selected points of the two sets
- **Weight the corresponding pairs**
- E.g., assign lower weights for points with higher point-point distances
- Determine transformation that minimizes the error function

ICP Variants

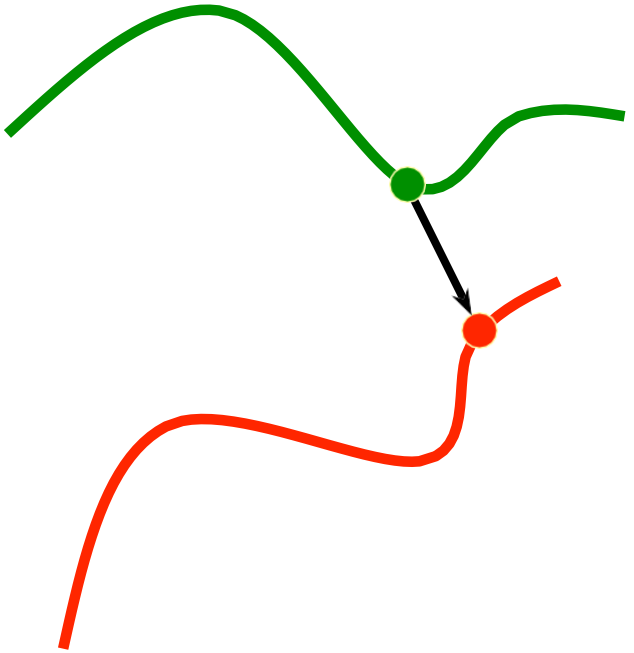
1. Point subsets (from one or both point sets)
2. Weighting the correspondences
3. Data association
4. Rejecting certain (outlier) point pairs

Data Association

- Has greatest effect on convergence and speed
- Matching methods:
  - Closest point
  - Normal shooting
  - Closest compatible point
  - Projection-based

Closest-Point Matching

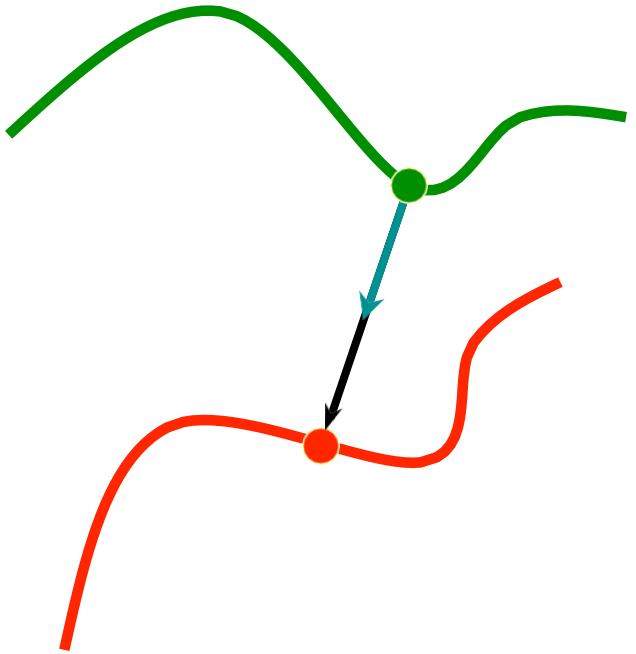
- Find closest point in other the point set (using kd-trees)



Generally stable, but slow convergence and requires preprocessing

Normal Shooting

- Project along normal, intersect other point set



Slightly better convergence results than closest point for smooth structures, worse for noisy or complex structures

Closest Compatible Point

- Improves the two previous variants by considering the **compatibility** of the points
- Only match compatible points
- Compatibility can be based on
  - Normals
  - Colors
  - Curvature
  - Higher-order derivatives
  - Other local features

Point-to-Plane Error Metric

- Minimize the sum of the squared distance between a point and the tangent plane at its correspondence point [Chen & Medioni 91]

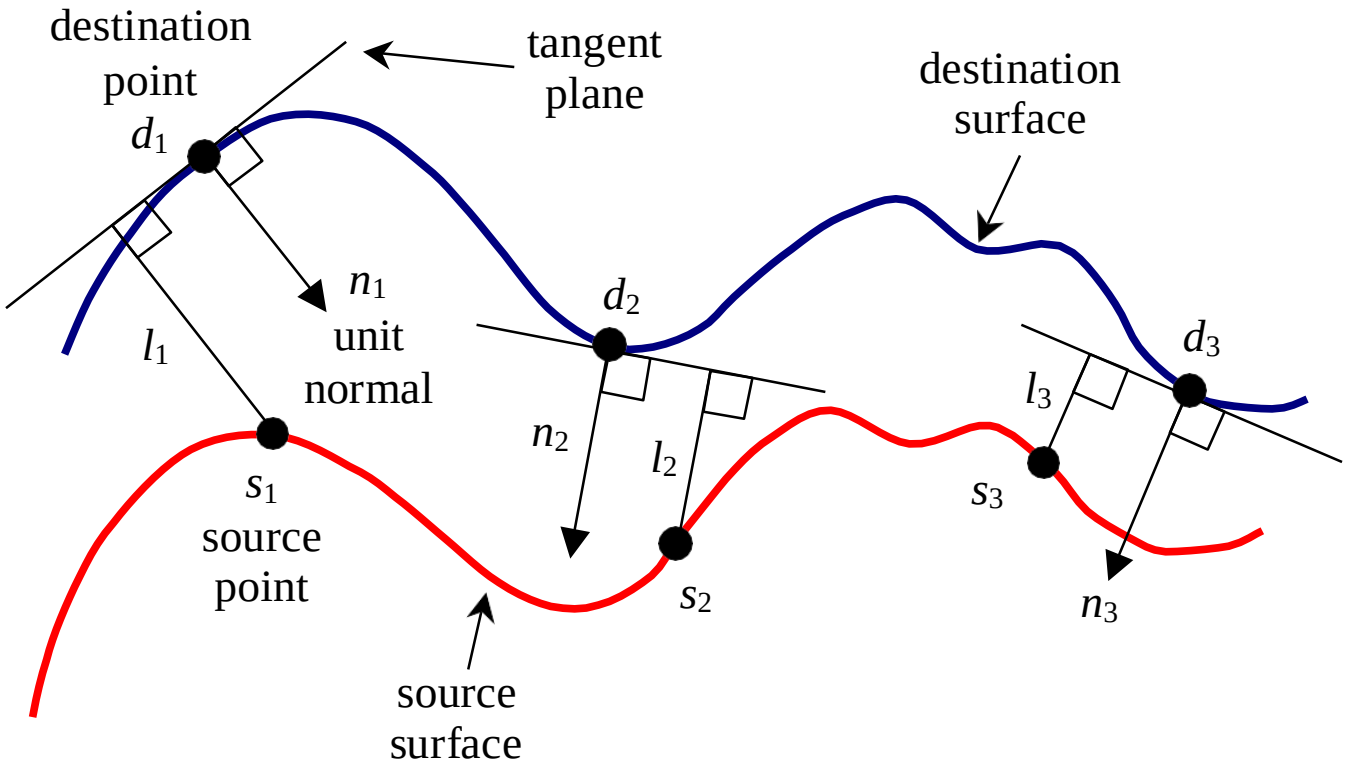


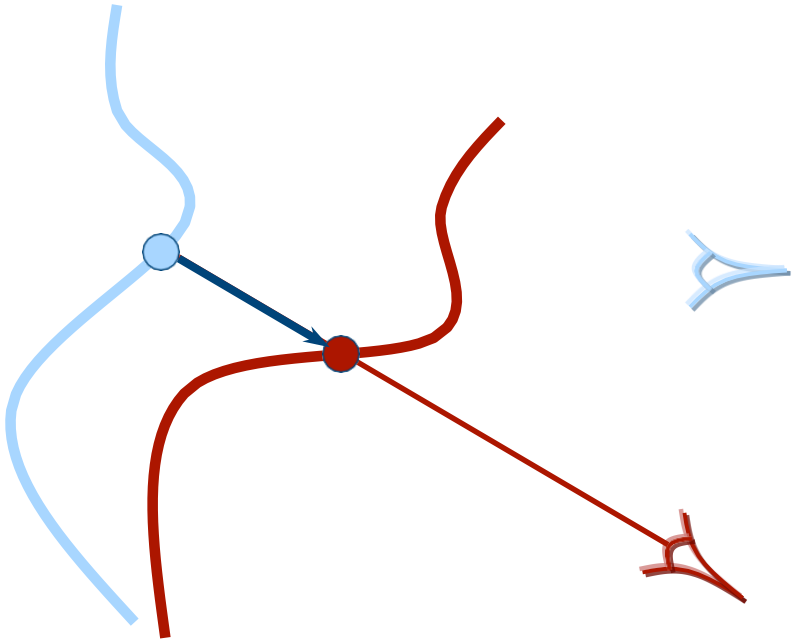
image from [Low 04]

Point-to-Plane Error Metric

- Solved using standard nonlinear least squares methods (e.g., Levenberg-Marquardt method [Press92]).
- Each iteration generally slower than the point-to-point version, however, often significantly better convergence rates [Rusinkiewicz01]
- Using point-to-plane distance instead of point-to-point lets flat regions slide along each other [Chen & Medioni 91]

Projection

- Finding the closest point is the most expensive stage of the ICP algorithm
- Idea: Simplified nearest neighbor search
- For range images, one can project the points according to the view-point [Blais 95]



Projection-Based Matching

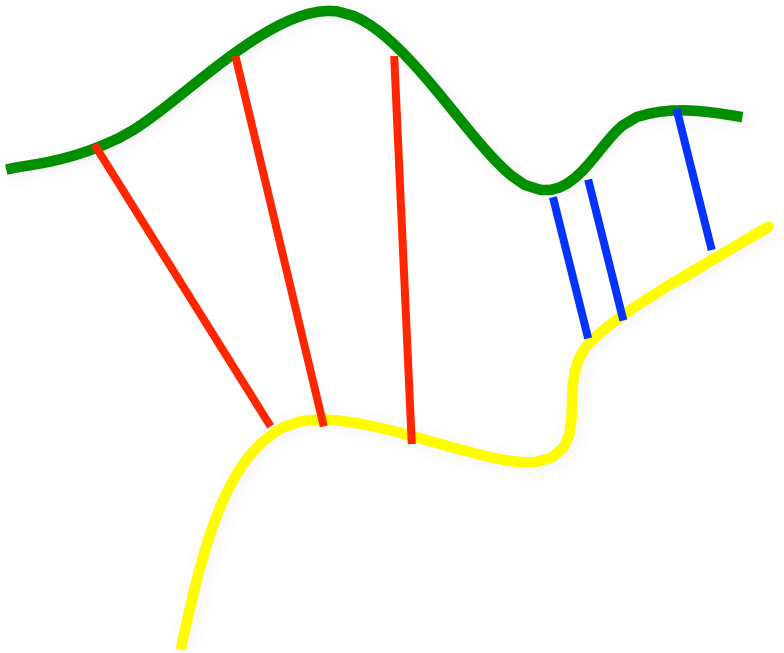
- Constant time
- Does not require precomputing a special data structure
- Requires point-to-plane error metric
- Slightly worse alignments per iteration

ICP Variants

1. Point subsets (from one or both point sets)
2. Weighting the correspondences
3. Data association
4. Rejecting certain (outlier) point pairs

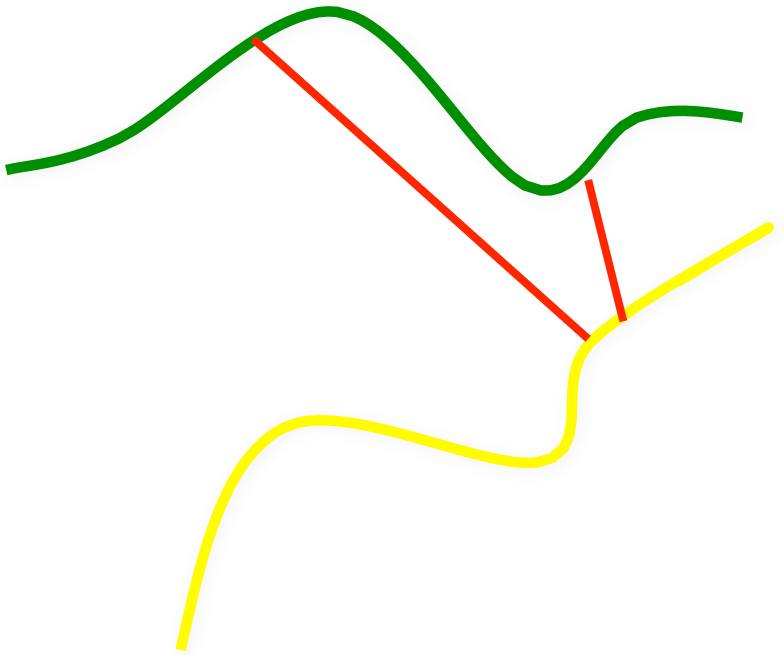
Rejecting (Outlier) Point Pairs

- Corresponding points with point to point distance higher than a given threshold



Rejecting (Outlier) Point Pairs

- Corresponding points with point to point distance higher than a given threshold
- Rejection of pairs that are not consistent with their neighboring pairs [Dorai 98]



Rejecting (Outlier) Point Pairs

- Corresponding points with point to point distance higher than a given threshold
- Rejection of pairs that are not consistent with their neighboring pairs [Dorai 98]
- Sort all correspondences with respect to their error and delete the worst *t*%, Trimmed ICP (TrICP) [Chetverikov et al. 02]
  - *t* is used to estimate the overlap
  - Problem: Knowledge about the overlap is necessary or has to be estimated

Summary: ICP Algorithm

- Potentially sample Points
- Determine corresponding points
- Potentially weight / reject pairs
- Compute rotation R, translation t (e.g. SVD)
- Apply R and t to all points of the set to be registered
- Compute the error $E(R,t)$
- If error decreased and error > threshold
  - Repeat to determine correspondences etc.
  - Stop and output final alignment, otherwise

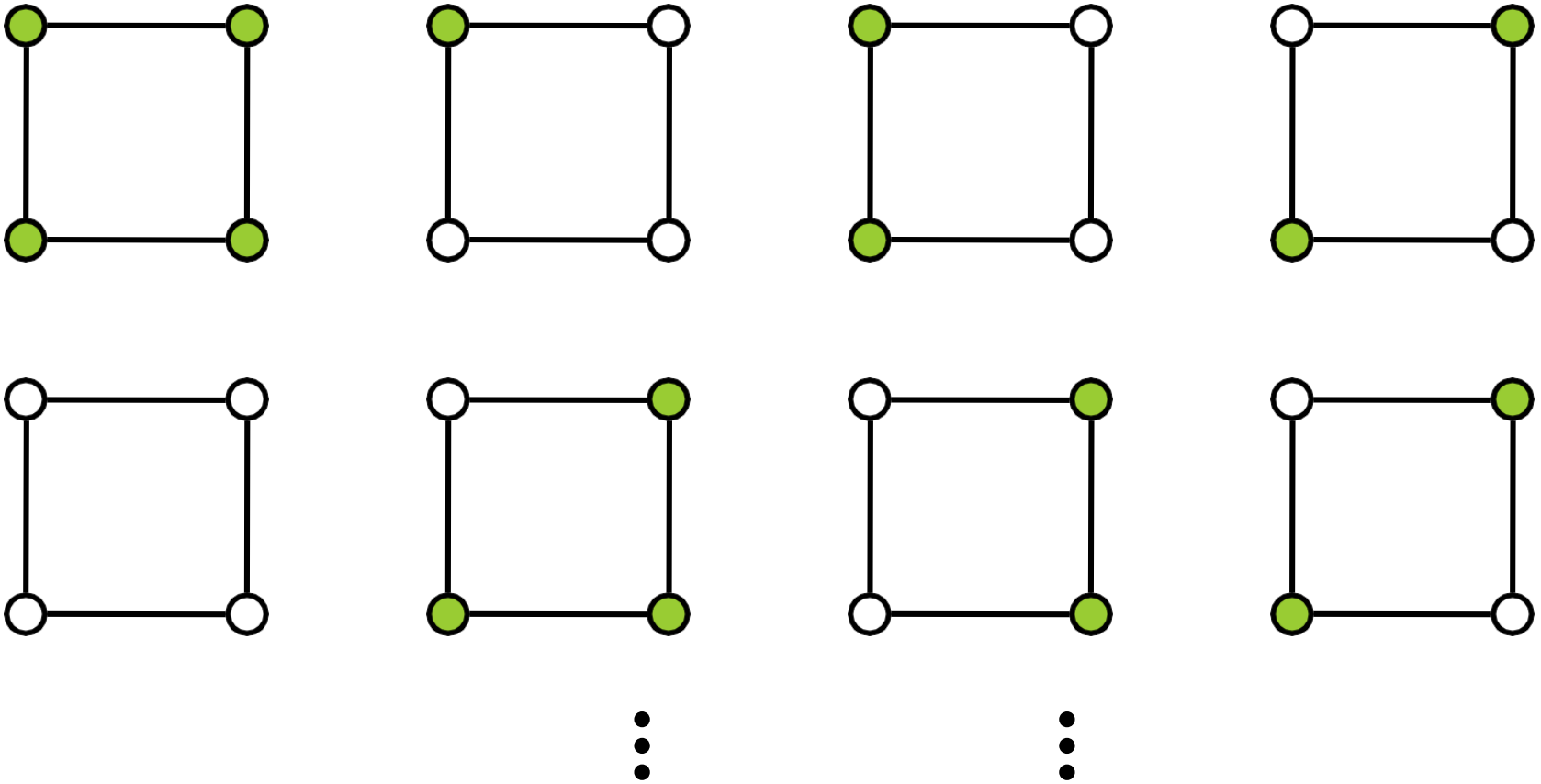
ICP Summary

- ICP is a powerful algorithm for calculating the displacement between scans
- The major problem is to determine the correct data associations
- Convergence speed depends on point matchings
- Given the correct data associations, the transformation can be computed efficiently using SVD

Marching Squares

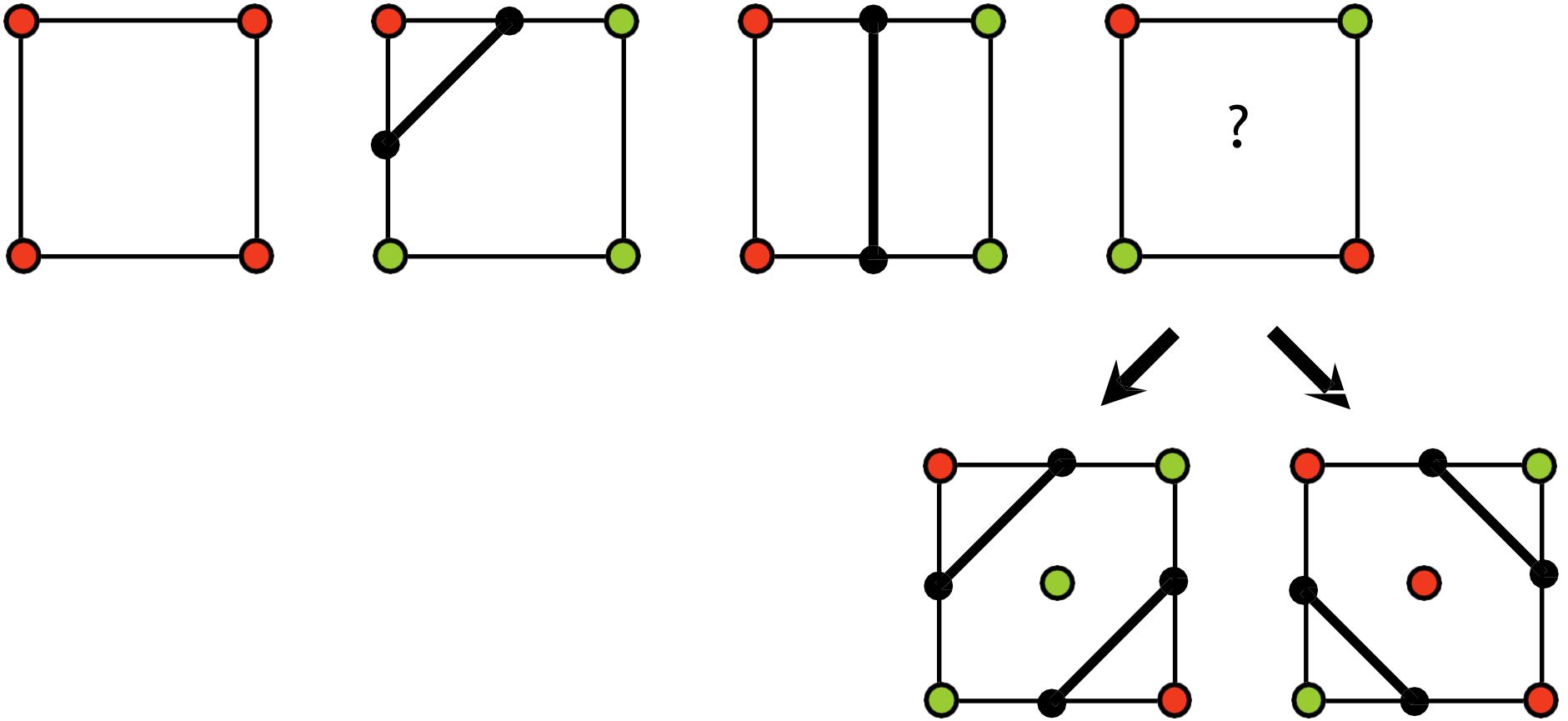
16 configurations différentes configurations en 2D

4 classes (rotation, reflection, negation)



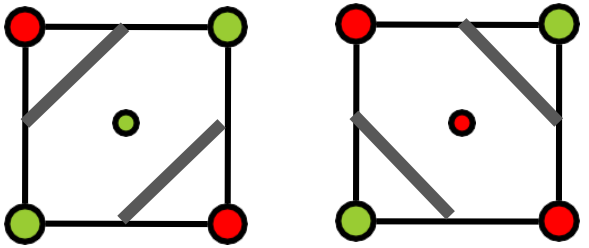
Tessellation in 2D

4 classes (rotation, reflection, negation)

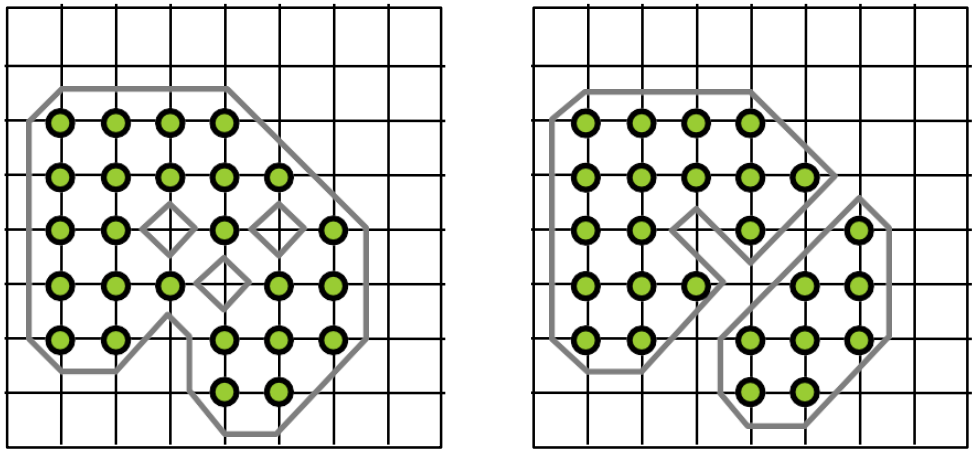


Tessellation in 2D

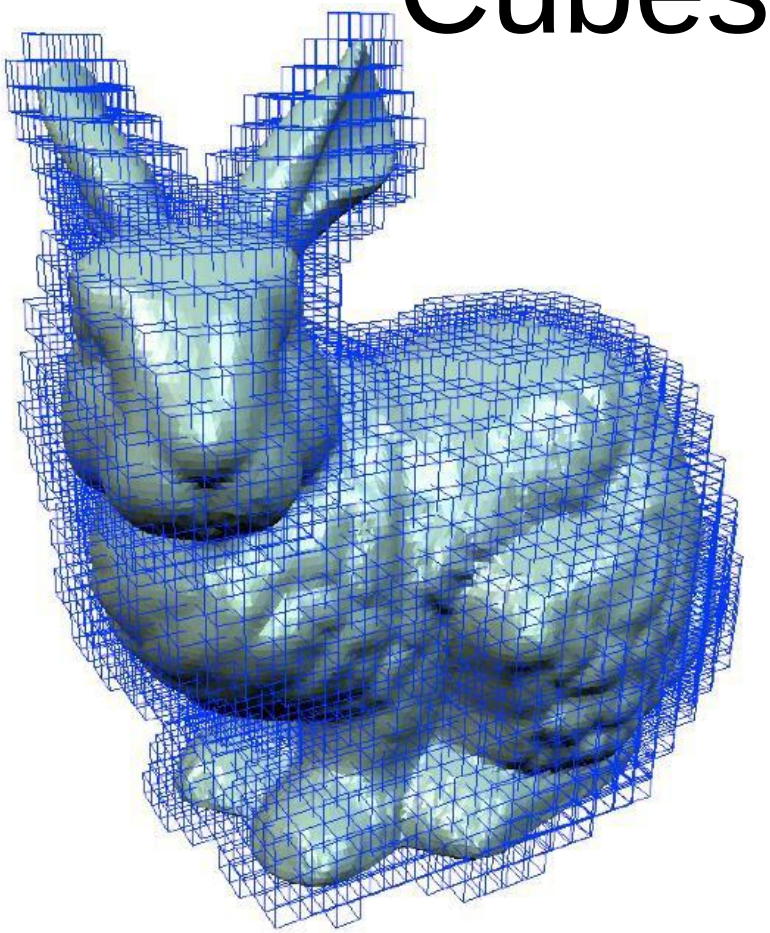
Case 4 is ambiguious:

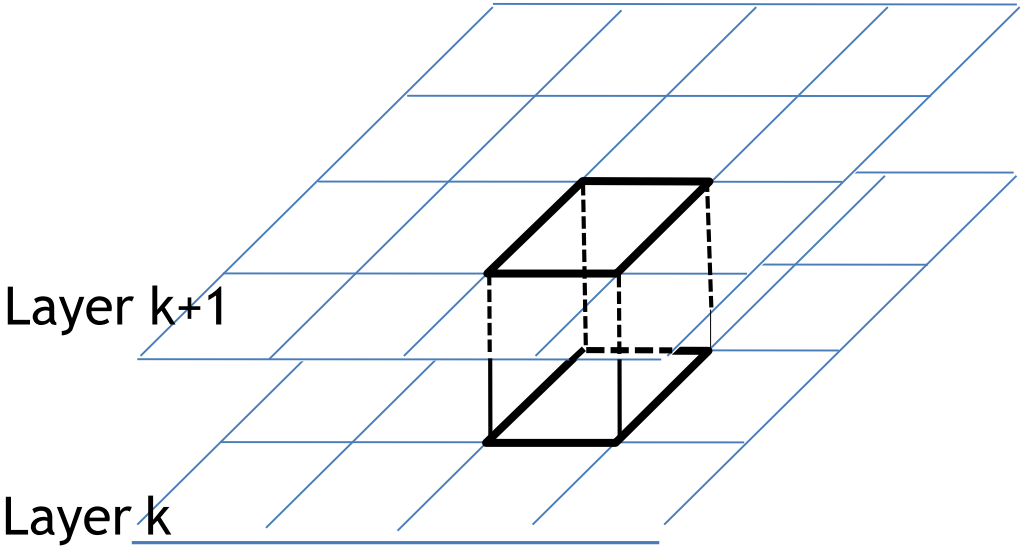


Always pick consistently



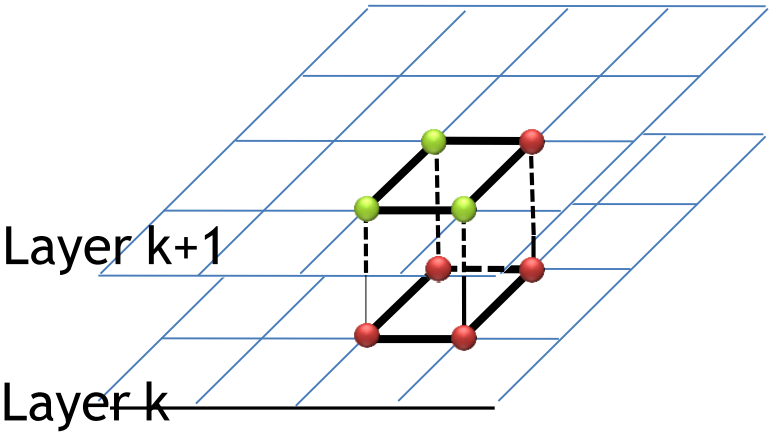
3D: Marching Cubes





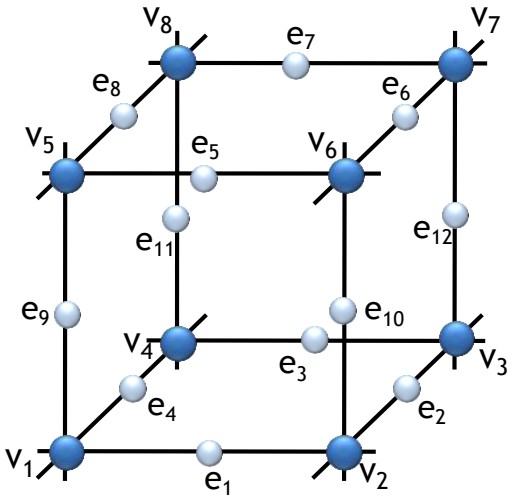
Marching Cubes

- Marching Cubes (Lorensen and Cline 1987)
  1. Load 4 layers of the grid into memory
  2. Create a cube whose vertices lie on the two middle layers
  3. Classify the vertices of the cube according to the implicit function (inside, outside or on the surface)

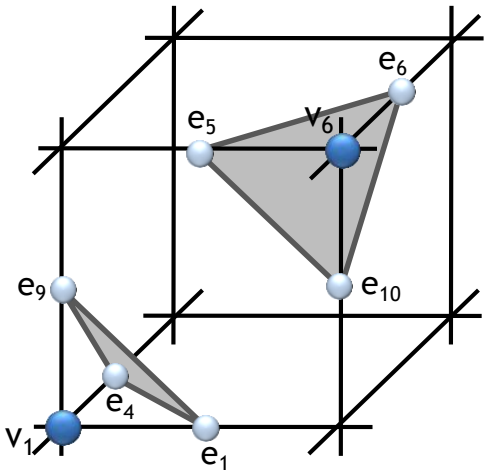


Marching Cubes

4. Compute case index. We have $2^8 = 256$ cases (0/1 for each of the eight vertices) – can store as 8 bit (1 byte) index.



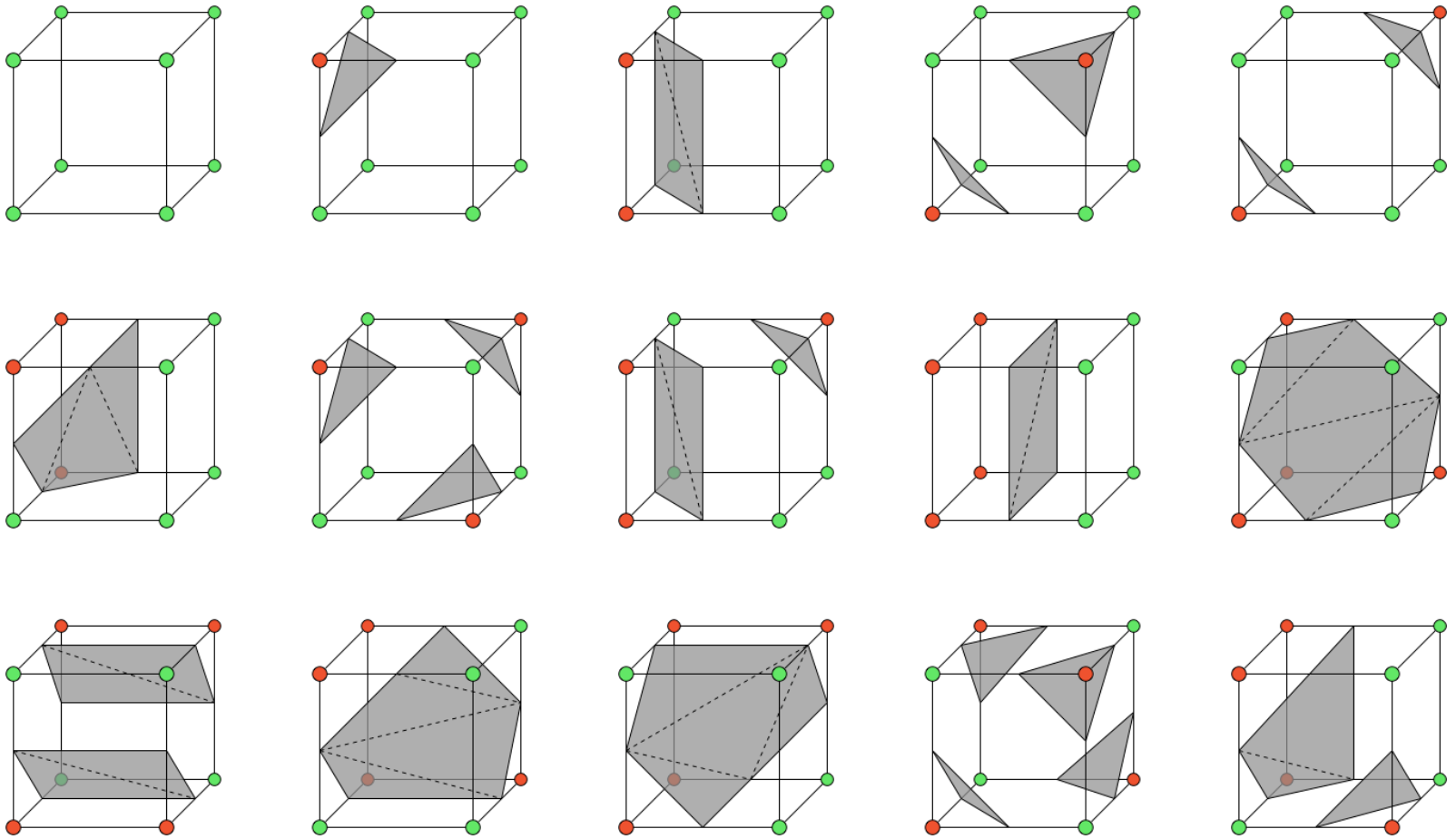
index = | v_1 | v_2 | v_3 | v_4 | v_5 | v_6 | v_7 | v_8 |



index = | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | = 33

Marching Cubes

- Unique cases (by rotation, reflection and negation)

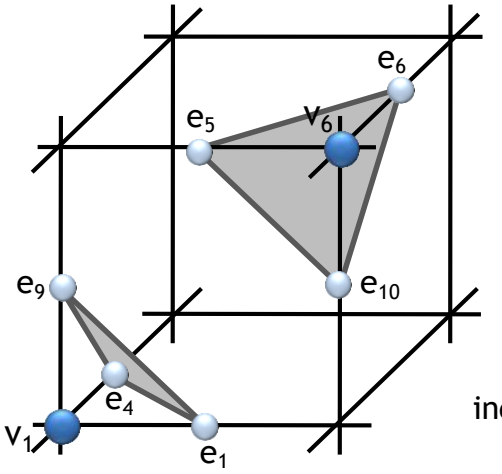


Tessellation

3D – Marching Cubes

5. Using the case index, retrieve the connectivity in the look-up table

- Example: the entry for index 33 in the look-up table indicates that the cut edges are e_1; e_4; e_5; e_6; e_9 and e_{10} ; the output triangles are (e_1; e_9; e_4) and (e_5; e_{10}; e_6).



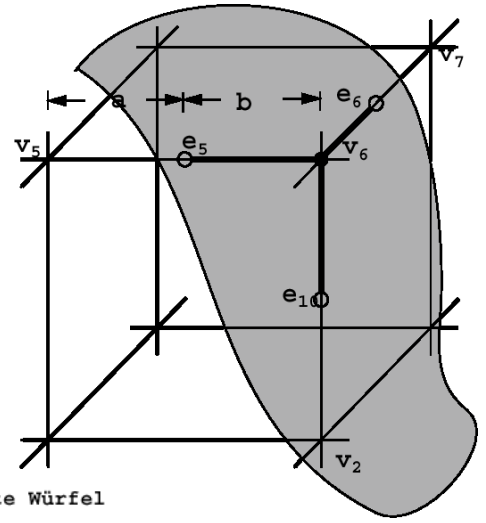
index = 0 0 1 0 0 0 0 1 = 33

Marching Cubes

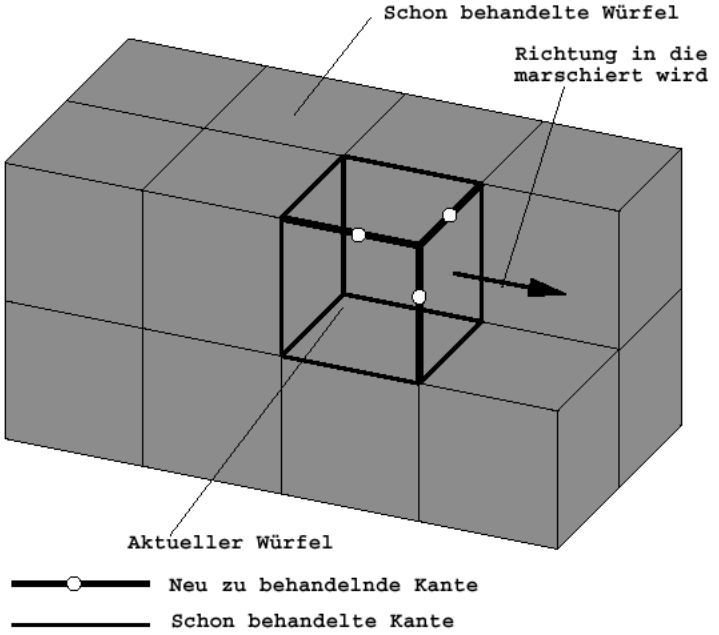
6. Calculer la position des sommets par interpolation

$$\mathbf{v}_s = t\mathbf{v}_a + (1-t)\mathbf{v}_b$$

$$t = \frac{F(\mathbf{v}_b)}{F(\mathbf{v}_b) - F(\mathbf{v}_a)}$$

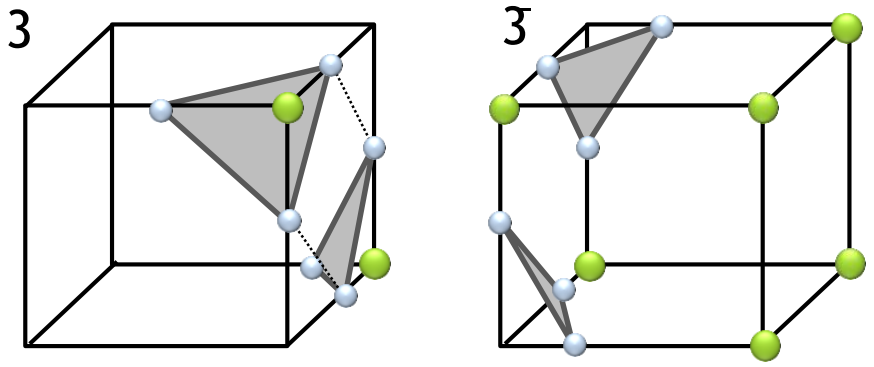


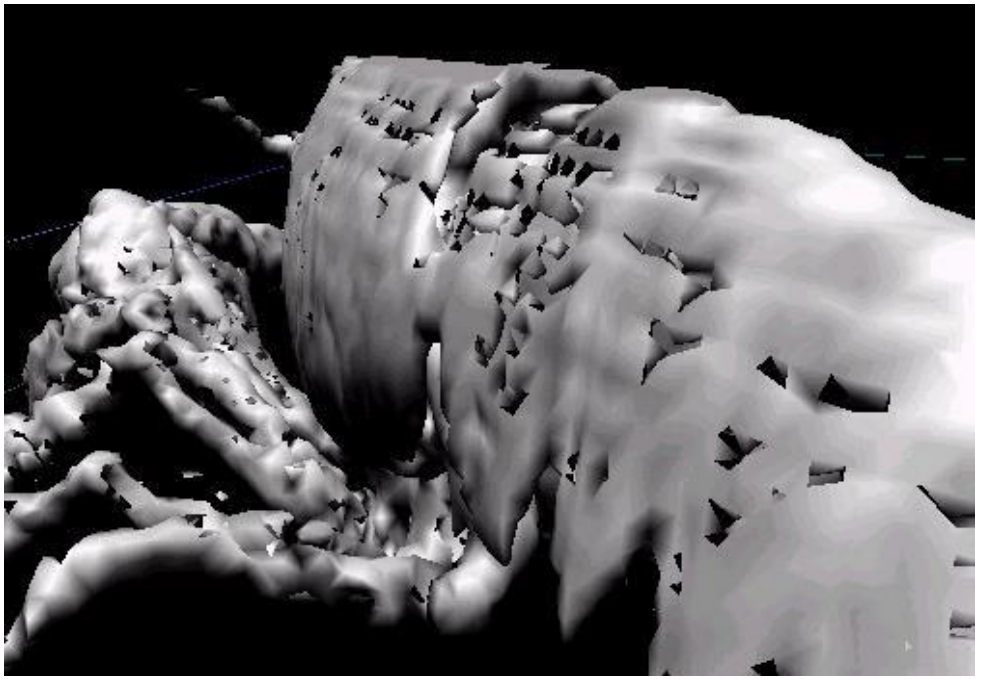
7. Passer au cube suivant



Marching Cubes – Problems

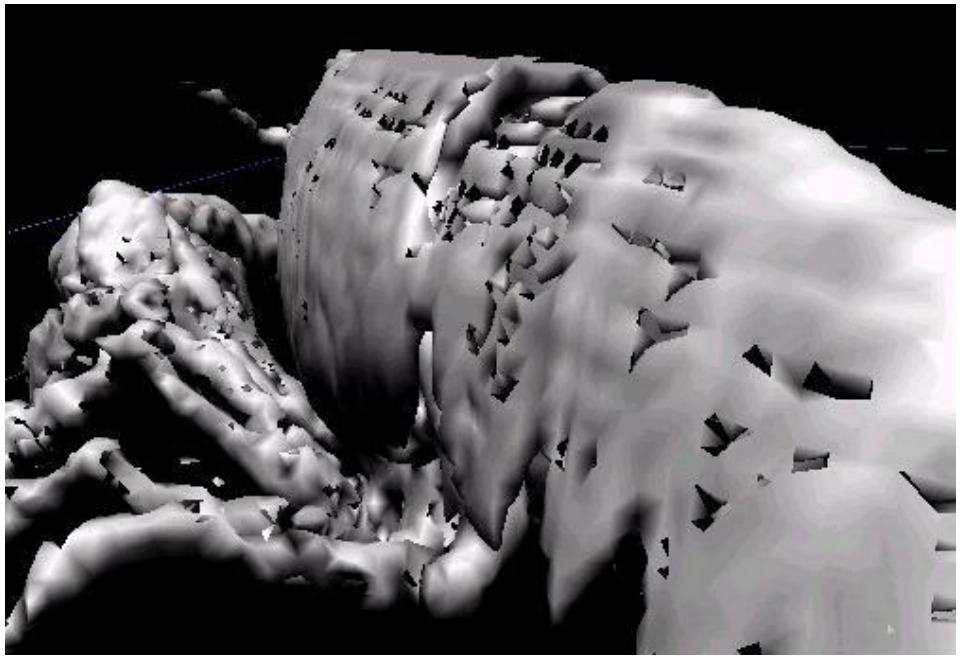
- Have to make consistent choices for neighboring cubes – otherwise get holes



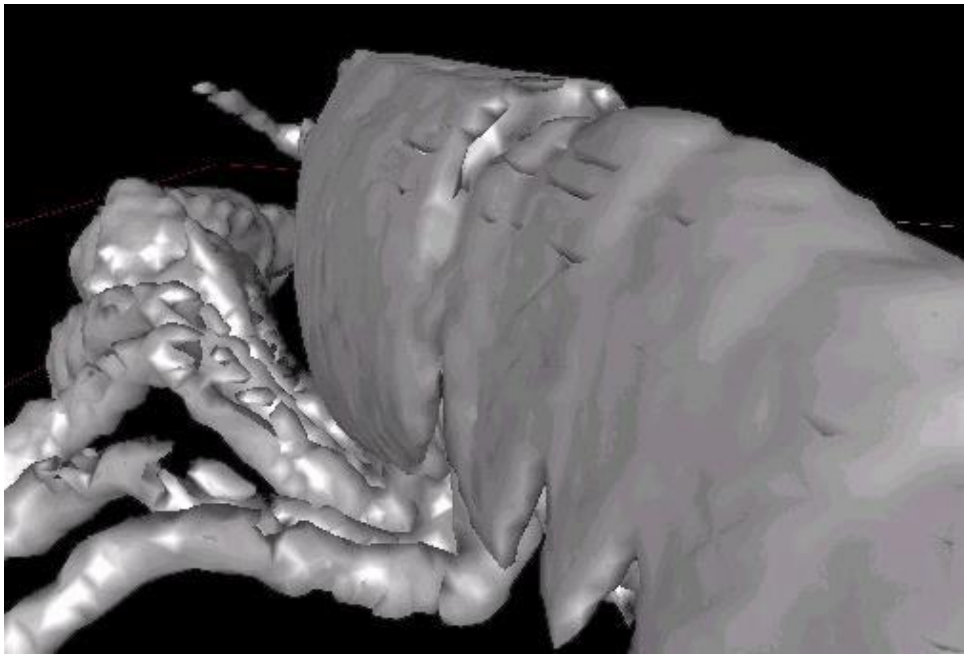


Marching Cubes – Problèmes

- Résolution des ambiguïtés



Ambiguity



No Ambiguity

Alternating Direction Method of Multipliers (ADMM)

Problème:

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c \end{array}$$

Lagrangien augmenté :

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + (\rho/2)\|Ax + Bz - c\|_2^2$$

Étapes:

$$\begin{aligned} x^{k+1} &:= \operatorname*{argmin}_x L_\rho(x, z^k, y^k) \\ z^{k+1} &:= \operatorname*{argmin}_z L_\rho(x^{k+1}, z, y^k) \\ y^{k+1} &:= y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \end{aligned}$$

Requiert une leçon, mais vous pouvez me croire...

ADMM formulation de Sparse ICP

$$(R,t)=argmin\sum_i \|R.p_i+t-q_i\|^p$$ est reformulé :

$$(R,t)=argmin\sum_i \|z_i\|^p \quad s.t. \quad z_i=R.p_i+t-q_i$$

Lagrangien augmenté : $$L=\sum_i \|z_i\|^p+y_i^T\cdot(R.p_i+t-q_i-z_i)+\rho/2\|R.p_i+t-q_i-z_i\|^2$$

ADMM formulation de Sparse ICP

$$(R,t)=argmin \sum_i \|R.p_i+t-q_i\|^p$$ est reformulé :

$$(R,t)=argmin \sum_i \|z_i\|^p \quad s.t. \quad z_i=R.p_i+t-q_i$$

Lagrangien augmenté : $L=\sum_i \|z_i\|^p+y_i^T\cdot\underbrace{(R.p_i+t-q_i-z_i)}_{:=b}+\rho/2\|R.p_i+t-q_i-z_i\|^2$

Itérer:

a) z = argmin L. Vous pouvez vérifier que : $L=\sum_i \|z_i\|^p+\rho/2\,\|z_i-(b+y_i/\rho)\|^2+const$

→ Solution analytique indépendante (« shrink operateur », voir papier)

ADMM formulation de Sparse ICP

$$(R,t)=argmin \sum_i \|R.p_i+t-q_i\|^p$$ est reformulé :

$$(R,t)=argmin \sum_i \|z_i\|^p \quad s.t. \quad z_i=R.p_i+t-q_i$$

Lagrangien augmenté : $L=\sum_i \|z_i\|^p+y_i^T\cdot\underbrace{(R.p_i+t-q_i-z_i)}_{:=b}+\rho/2\|R.p_i+t-q_i-z_i\|^2$

Itérer:

a) z = argmin L. Vous pouvez vérifier que : $L=\sum_i \|z_i\|^p+\rho/2\,\|z_i-(b+y_i/\rho)\|^2+const$
→ Solution analytique indépendante (« shrink operateur », voir papier)

b) (R,t)= argmin L. N'implique que des termes quadratiques → Procustes classique

ADMM formulation de Sparse ICP

$$(R,t)=argmin \sum_i \|R.p_i+t-q_i\|^p$$ est reformulé :

$$(R,t)=argmin \sum_i \|z_i\|^p \quad s.t. \quad z_i=R.p_i+t-q_i$$

Lagrangien augmenté : $$L=\sum_i \|z_i\|^p+y_i^T\cdot\underbrace{(R.p_i+t-q_i-z_i)}_{:=b}+\rho/2\|R.p_i+t-q_i-z_i\|^2$$

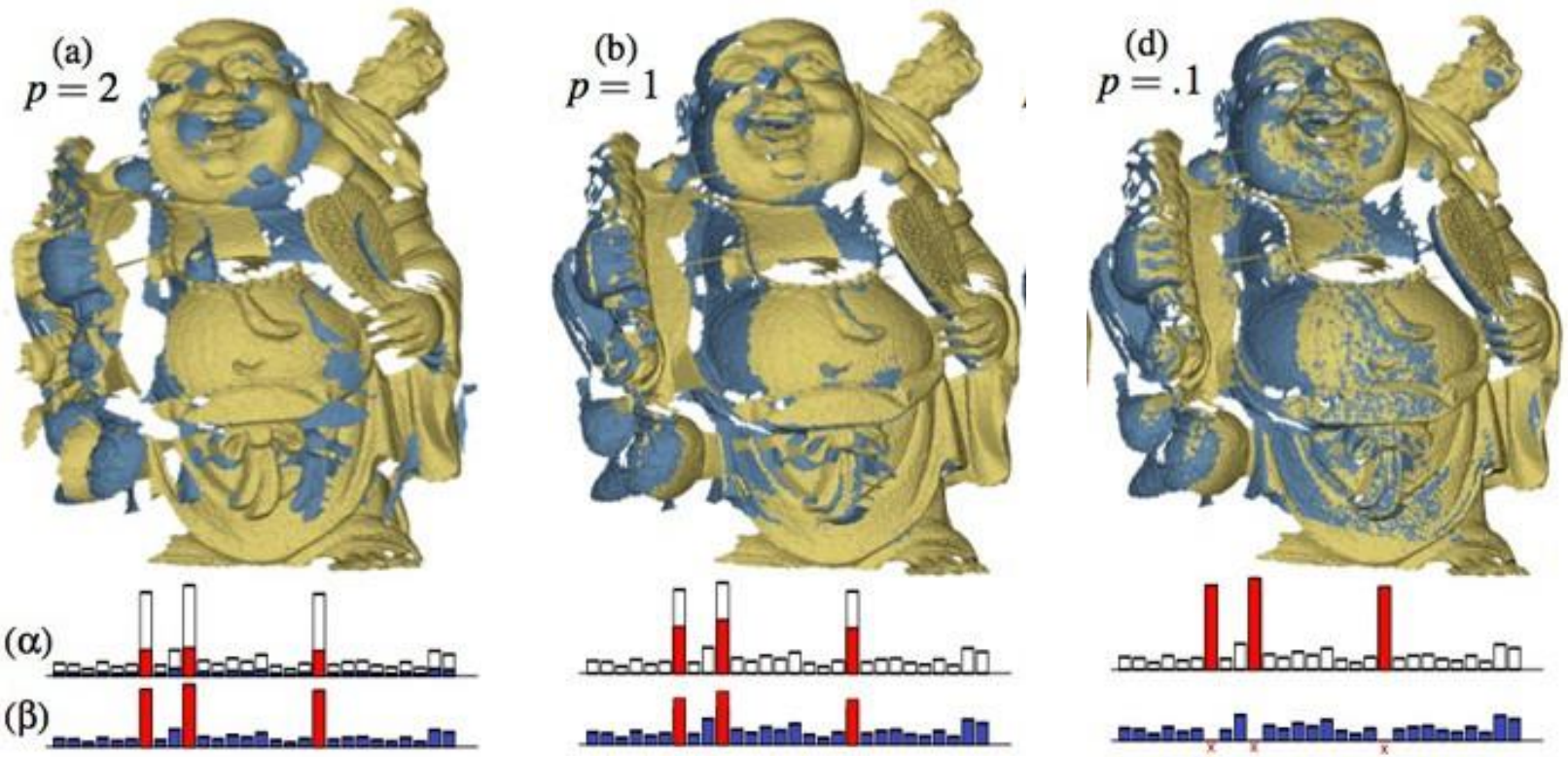
Itérer:

a) z = argmin L. Vous pouvez vérifier que : $L=\sum_i \|z_i\|^p+\rho/2\,\|z_i-(b+y_i/\rho)\|^2+const$
→ Solution analytique indépendante (« shrink operateur », voir papier)

b) (R,t)= argmin L. N'implique que des termes quadratiques → Procustes classique

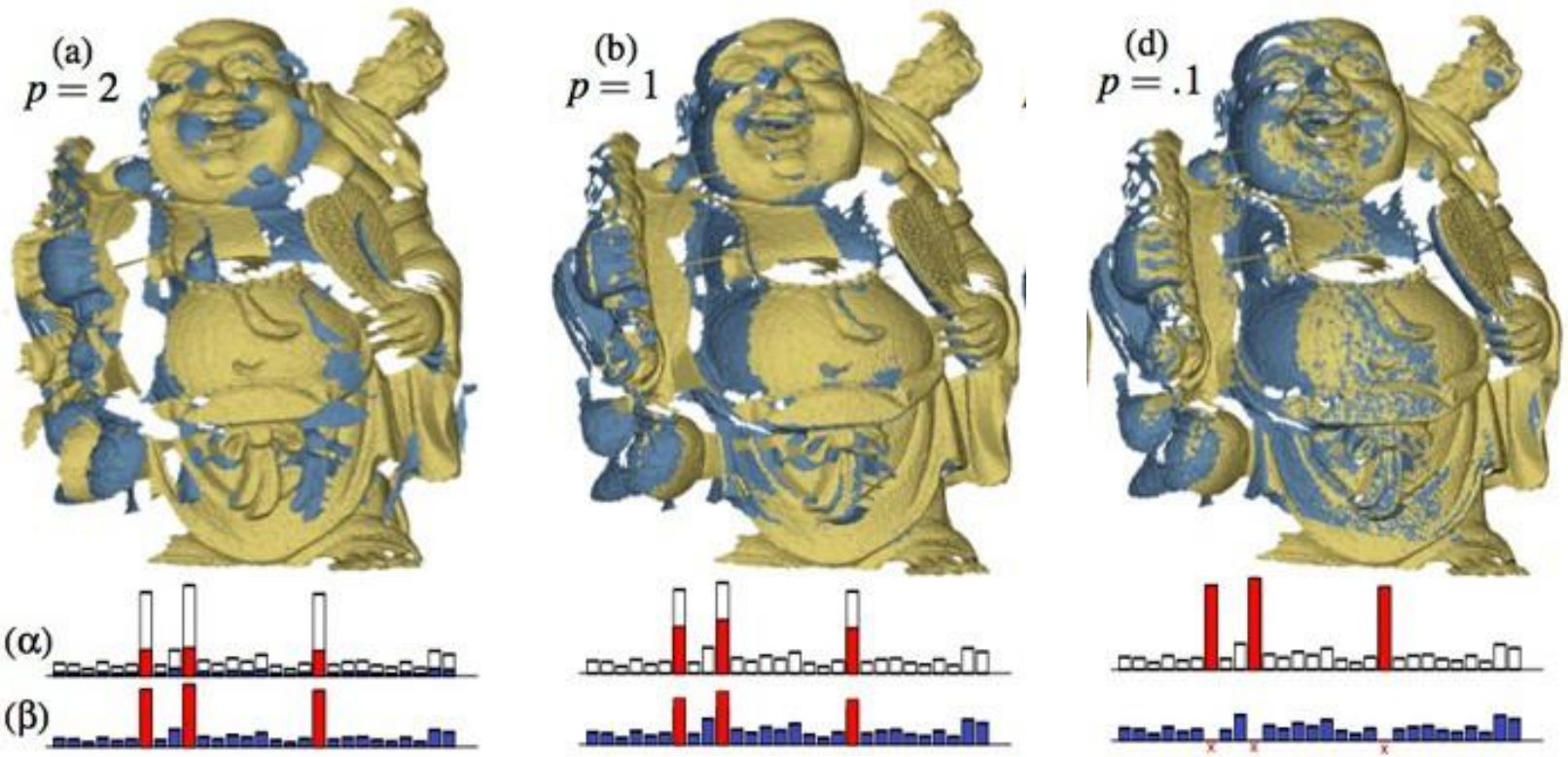
c) $y_i:=y_i+\rho(R.p_i+t-q_i-z_i)$

Application : « Sparse » ICP



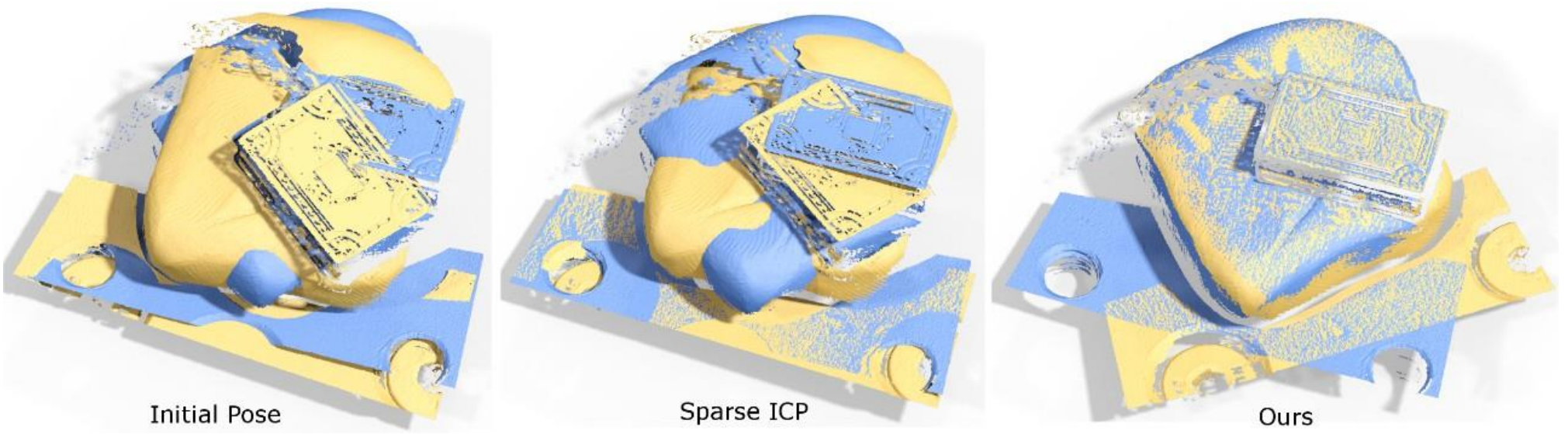
[Bouaziz et al. 2013] : Sparse Iterative Closest Point

Vidéo démo



https://www.youtube.com/watch?v=ii2vHBwlmo8

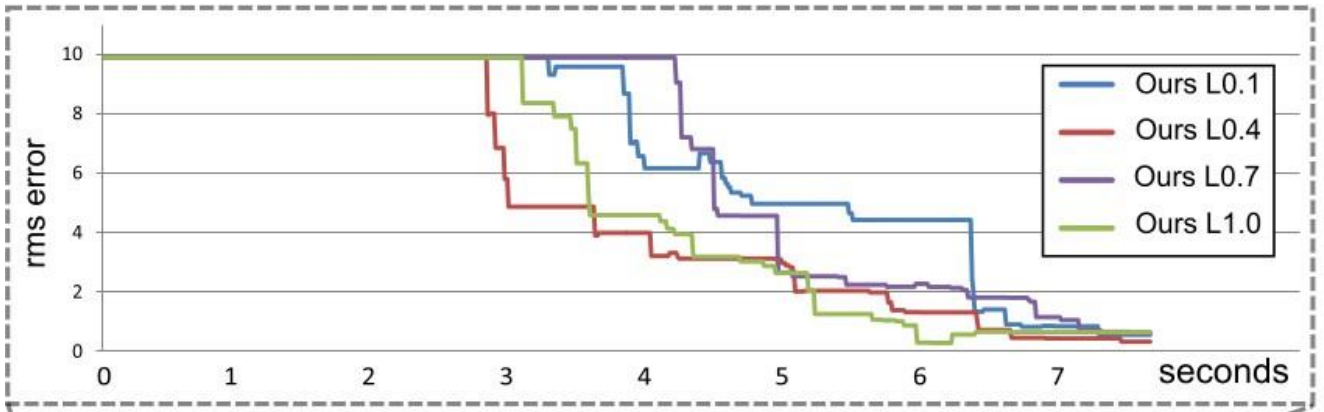
Sparse ICP : optimisation



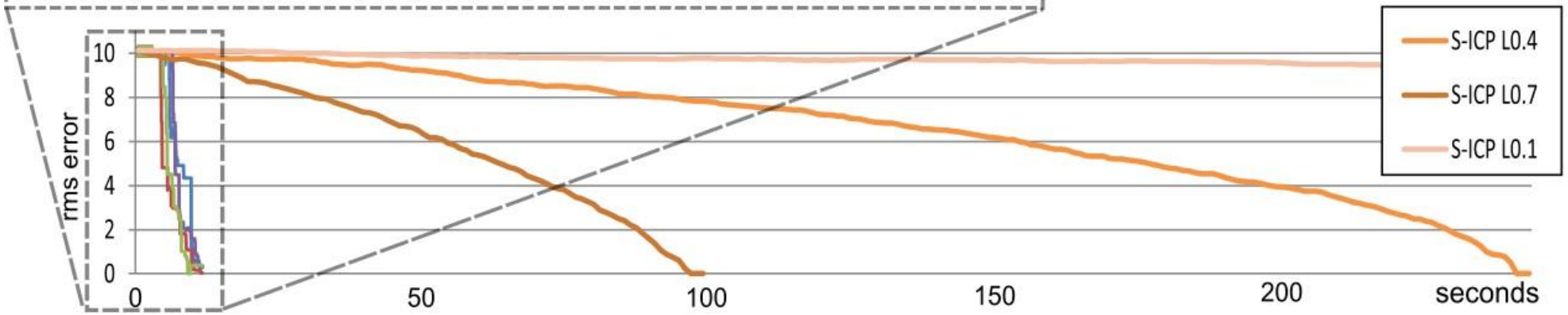
- ADMM est un couteau suisse, MAIS :
- Minima locaux
- Il y a souvent mieux !
- Comme la: Combiner ADMM avec un simple recuit simulé

[Mavridis et al. 2015] : Efficient Sparse ICP

Efficient sparse ICP



Norm	S-ICP	Ours	Speedup
$\ell_{1.0}$	37.6	5.2	7.2×
$\ell_{0.7}$	99.9	5.6	17.8×
$\ell_{0.4}$	246.8	6.6	37.4×
$\ell_{0.1}$	2646.3	7.5	352.8×



[Mavridis et al. 2015] : Efficient Sparse ICP