Lecture 9
Introduction to Numerical Geometry

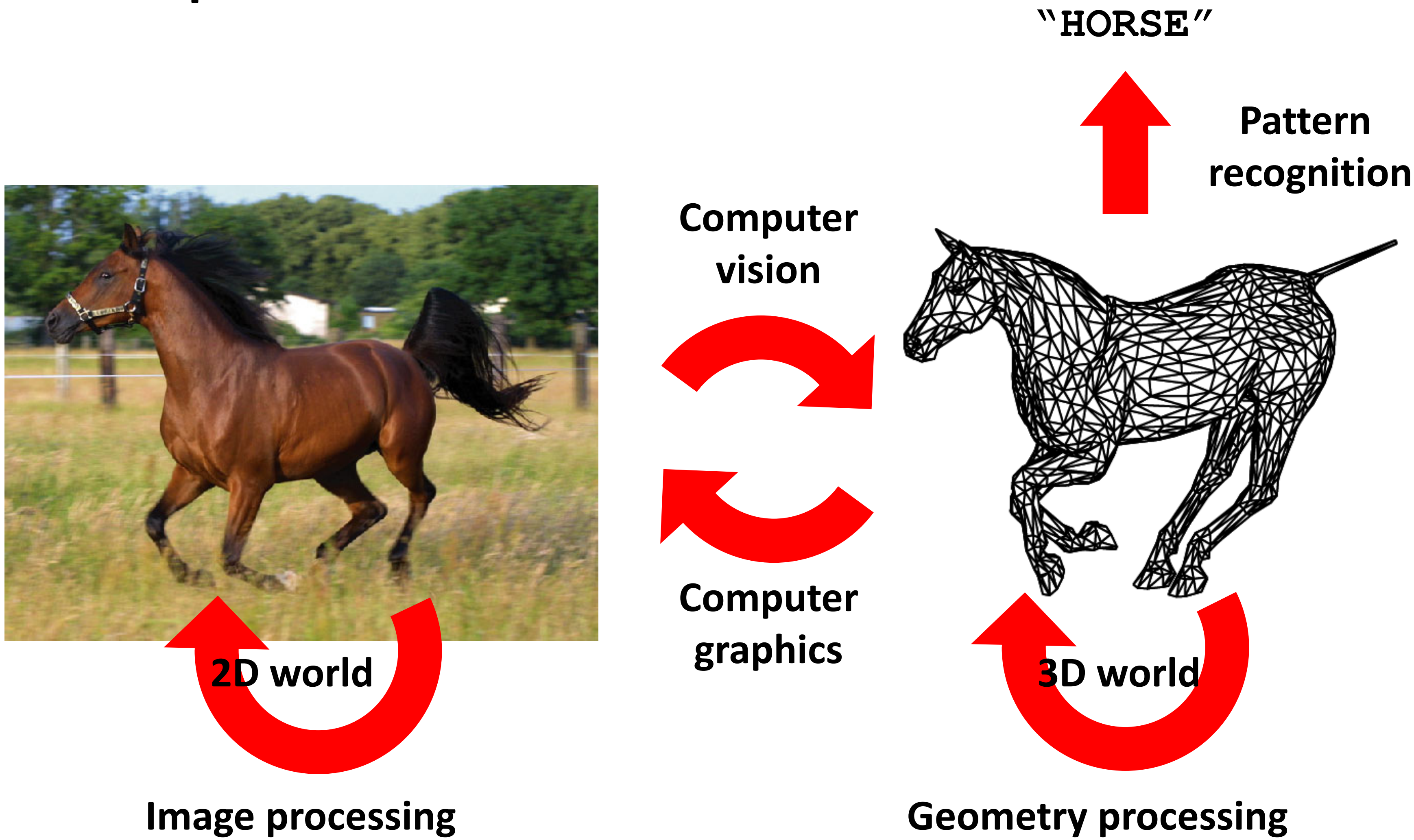
Lin ZHANG, PhD
School of Software Engineering
Tongji University
Fall 2024

Outline

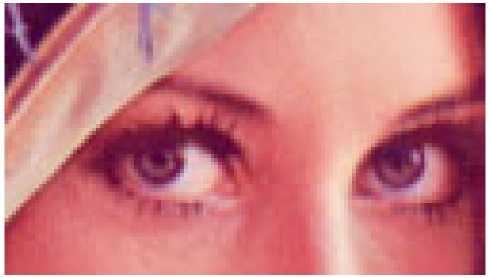
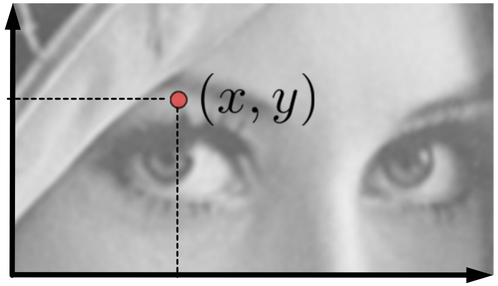
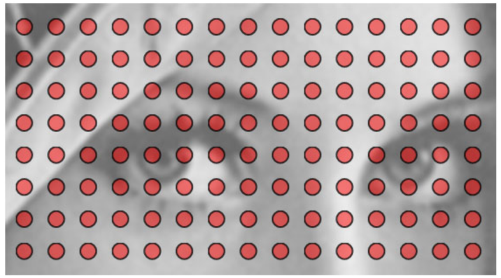
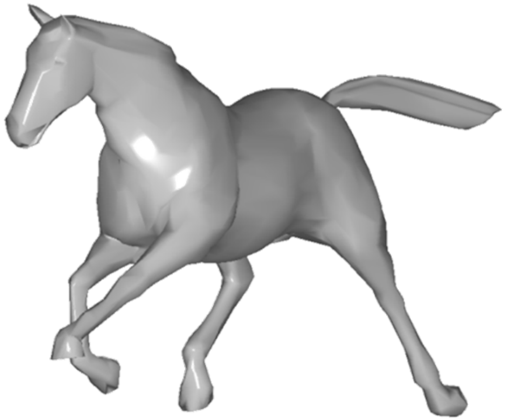
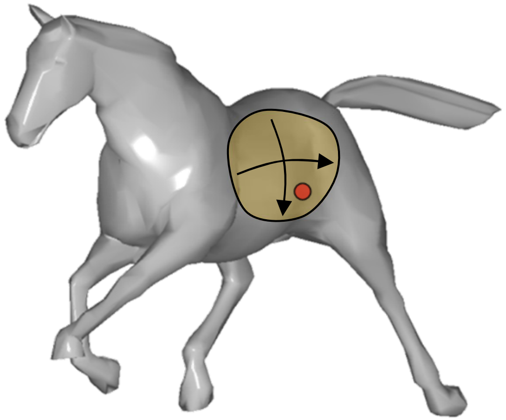
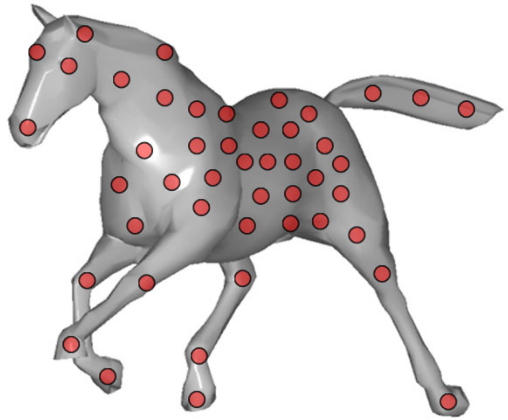
- Introduction
- Basic concepts in geometry
- Discrete geometry
  - Metric for discrete geometry
  - Sampling
- Rigid shape analysis
  - Euclidean isometries removal
  - ICP-based shape matching

Introduction

Landscape

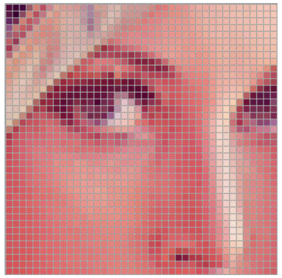


Shapes VS Images

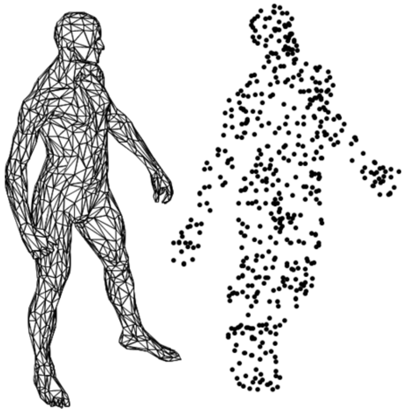
Geometry	**Parametrization**	**Sampling**
		
Euclidean (flat)	Global	Uniform Cartesian
		
Non-Euclidean (curved)	Local	“Uniform” is not well-defined

Shapes VS Images

Representation **Deformations**



Array of pixels

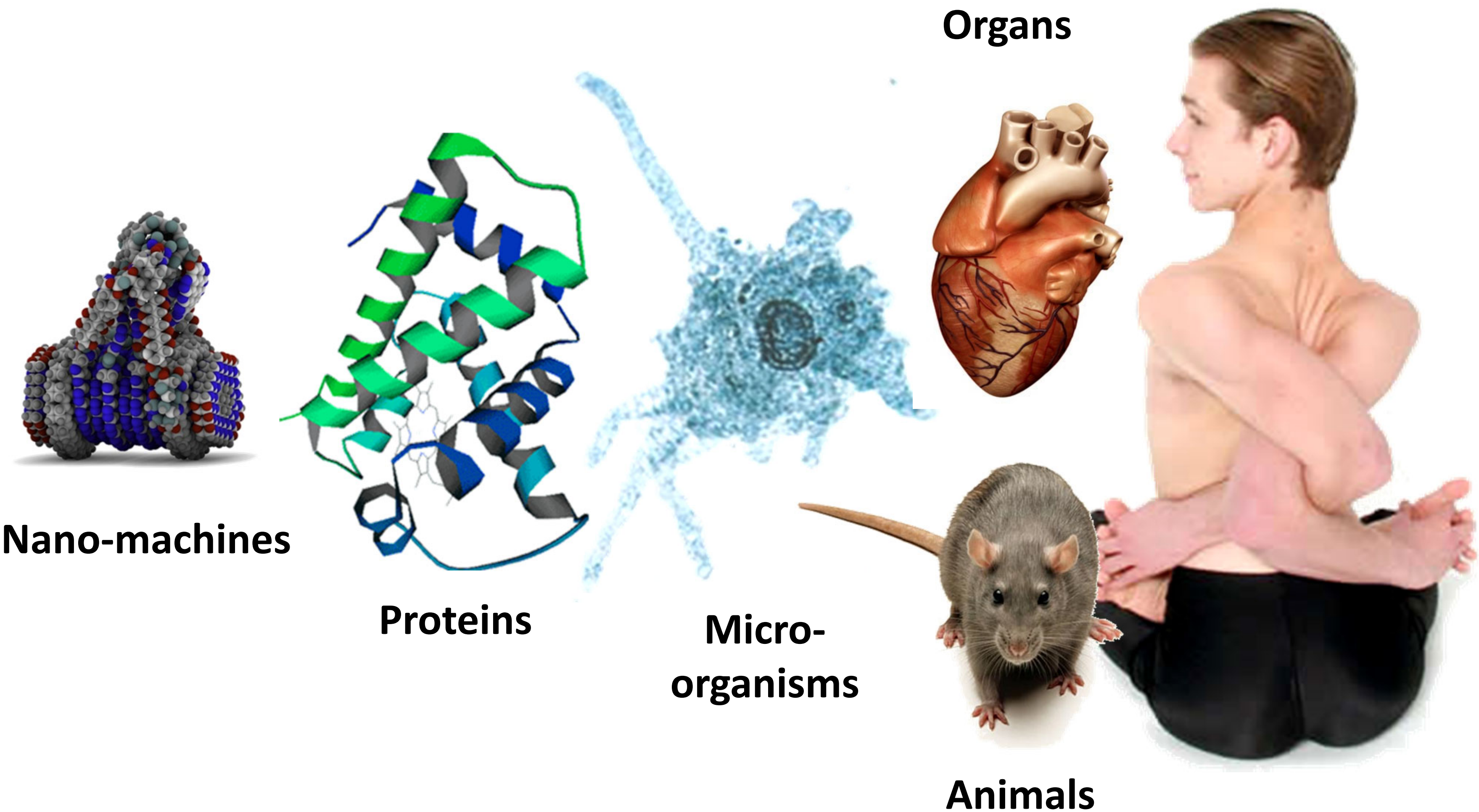


Cloud of points, mesh, etc, etc.

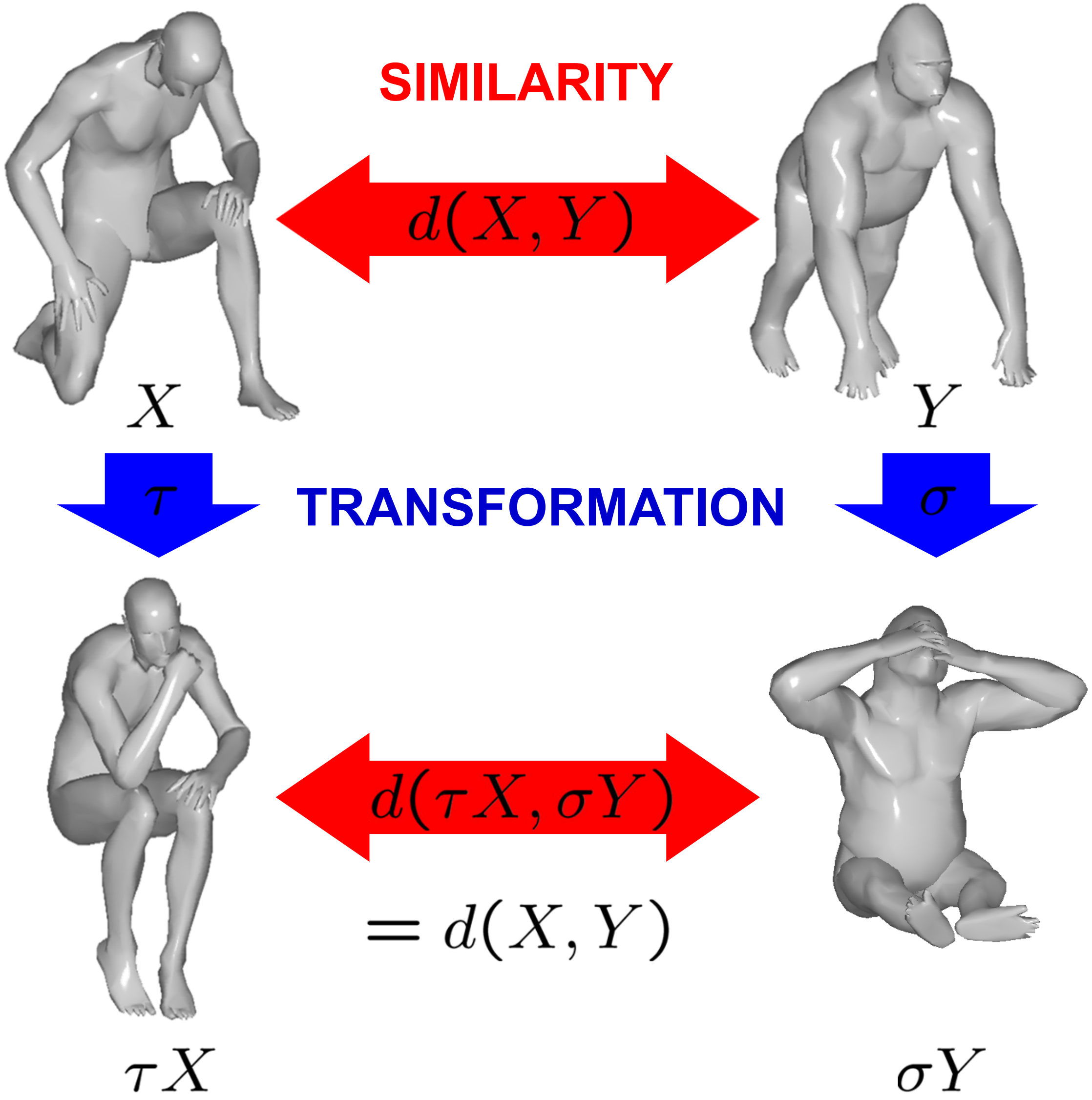
Rotation, affine, projective, etc.

Wealth of non-rigid deformations

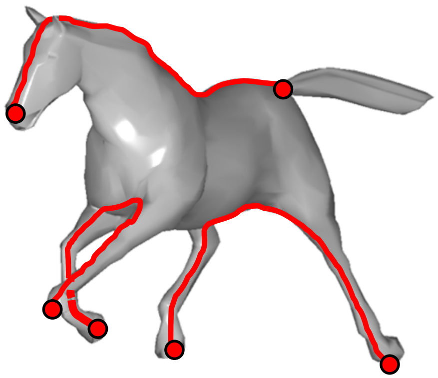
Non-rigid world from macro to nano



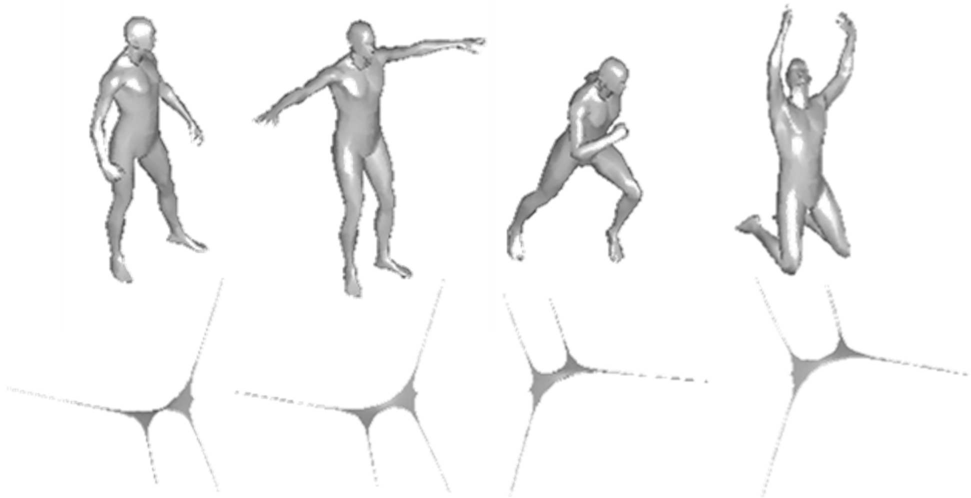
Invariant similarity



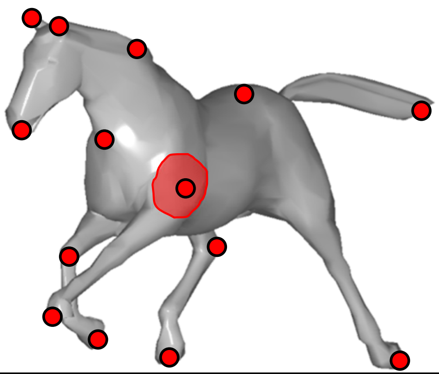
Topics



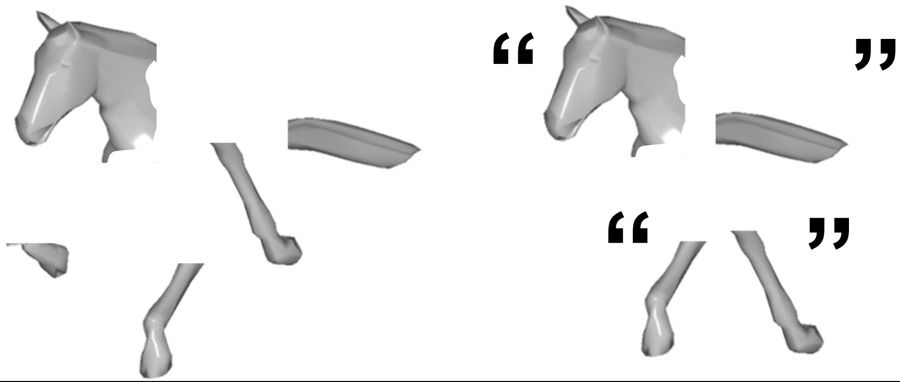
Metric spaces



Canonical forms



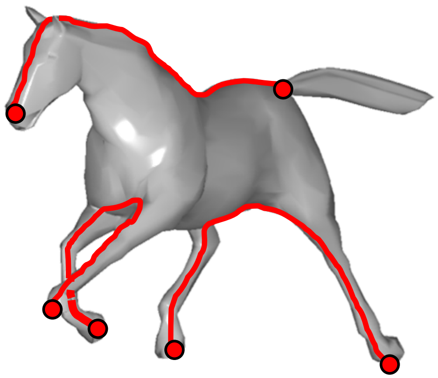
Local features



Geometric words & expressions

Shape Representation

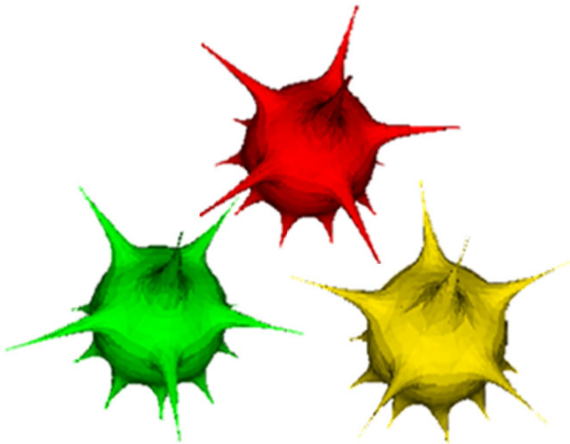
Tools



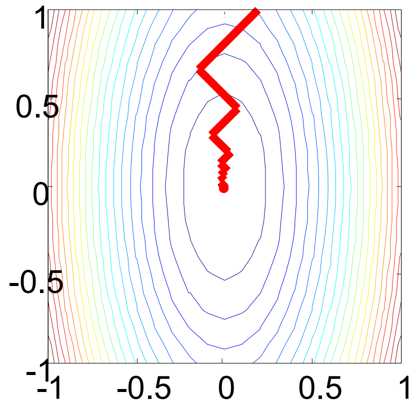
Metric geometry

Fast marching

Iterative closest point algorithms

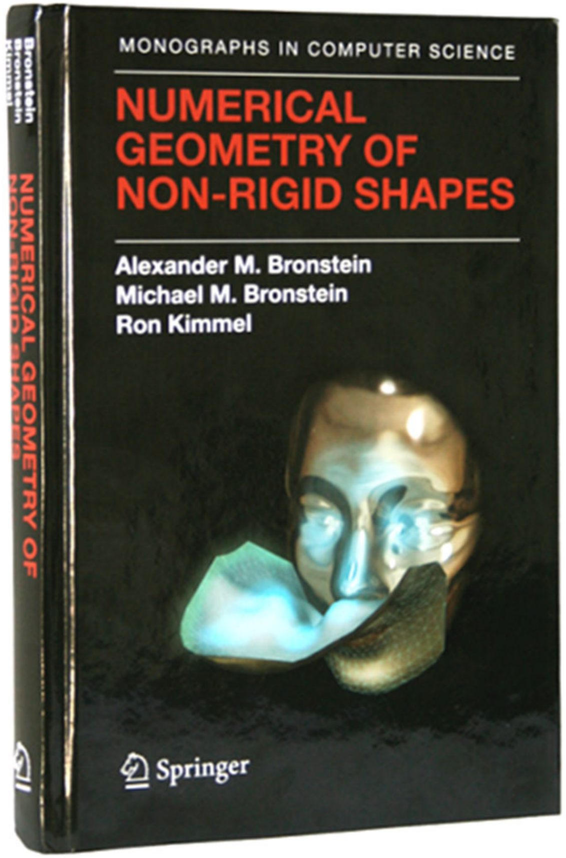


Multidimensional scaling



Convex optimization

Materials



A. M. Bronstein et al., Numerical geometry of non-rigid shapes, Springer 2008

Outline

- Introduction
- Basic concepts in geometry
- Discrete geometry
  - Metric for discrete geometry
  - Sampling
- Rigid shape analysis
  - Euclidean isometries removal
  - ICP-based shape matching

Distances

Euclidean

Manhattan

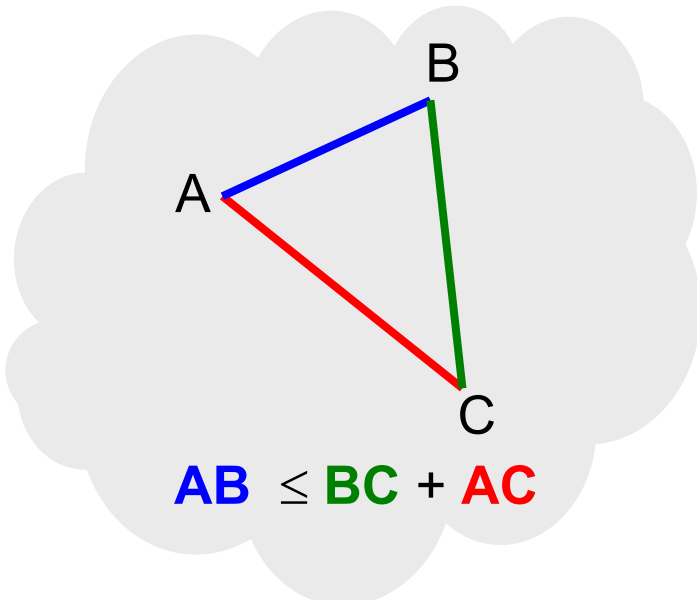
Geodesic

Metric

A function $d : X \times X \to \mathbb{R}$ satisfying for all $x_1, x_2, x_3 \in X$

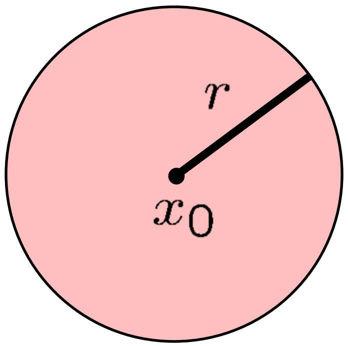
- **Non-negativity:** $d(x_1, x_2) \geq 0$
- **Indiscernability:** $d(x_1, x_2) = 0$ if and only if $x_1 = x_2$
- **Symmetry:** $d(x_1, x_2) = d(x_2, x_1)$
- **Triangle inequality:** $d(x_1, x_3) \leq d(x_1, x_2) + d(x_2, x_3)$

(X, d) is called a **metric space**



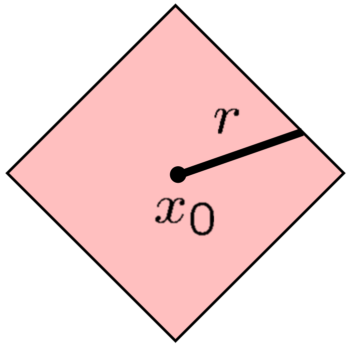
Metric balls

- **Open ball:** $B_r(x_0) = \{x \in X : d(x, x_0) < r\}$
- **Closed ball:** $\bar{B}_r(x_0) = \{x \in X : d(x, x_0) \leq r\}$



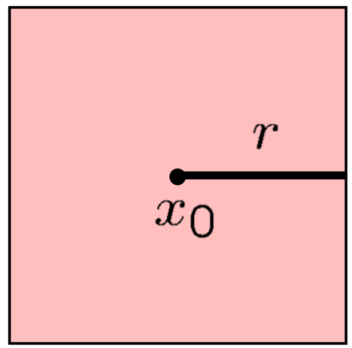
Euclidean ball

$$\|x - x_0\|_2 = \sqrt{\sum_k |x^k - x_0^k|^2} \leq r$$



L_1 ball

$$\|x - x_0\|_1 = \sum_k |x^k - x_0^k| \leq r$$

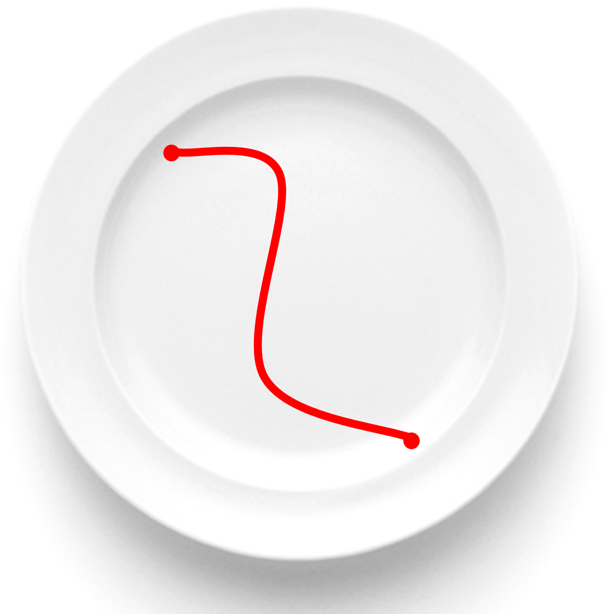


L_∞ ball

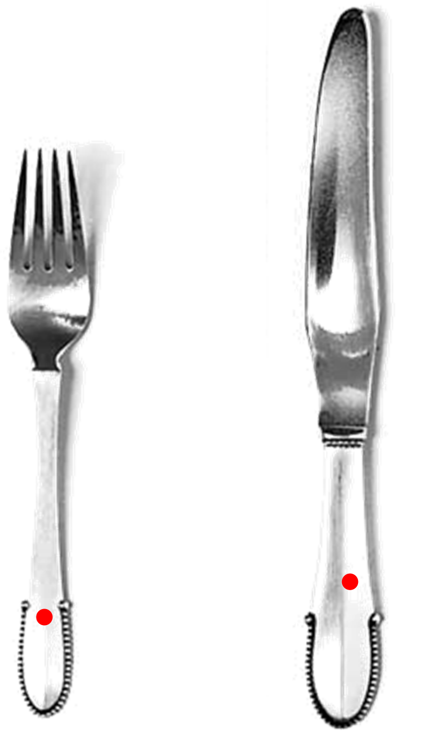
$$\|x - x_0\|_\infty = \max_k |x^k - x_0^k| \leq r$$

Connectivity

The space X is **connected** if it cannot be divided into two disjoint nonempty open sets, and **disconnected** otherwise



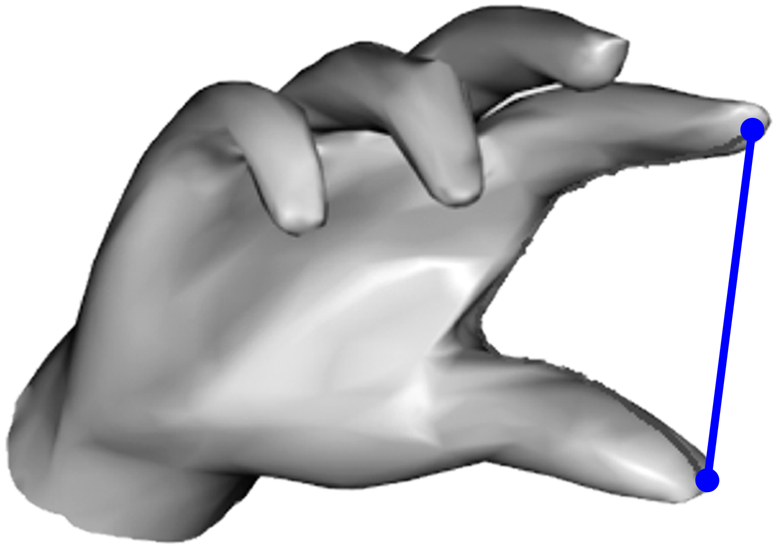
Connected



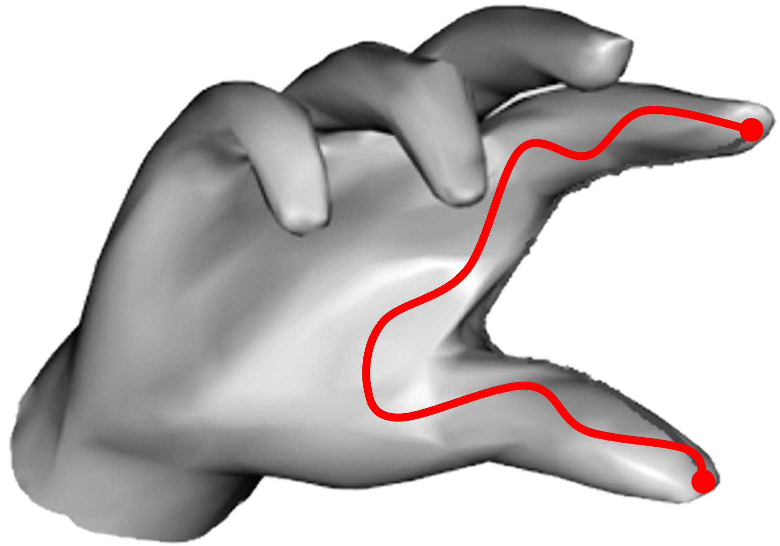
Disconnected

Stronger property: **path connectedness**

Examples of metrics



Euclidean



Path length

Homeomorphisms

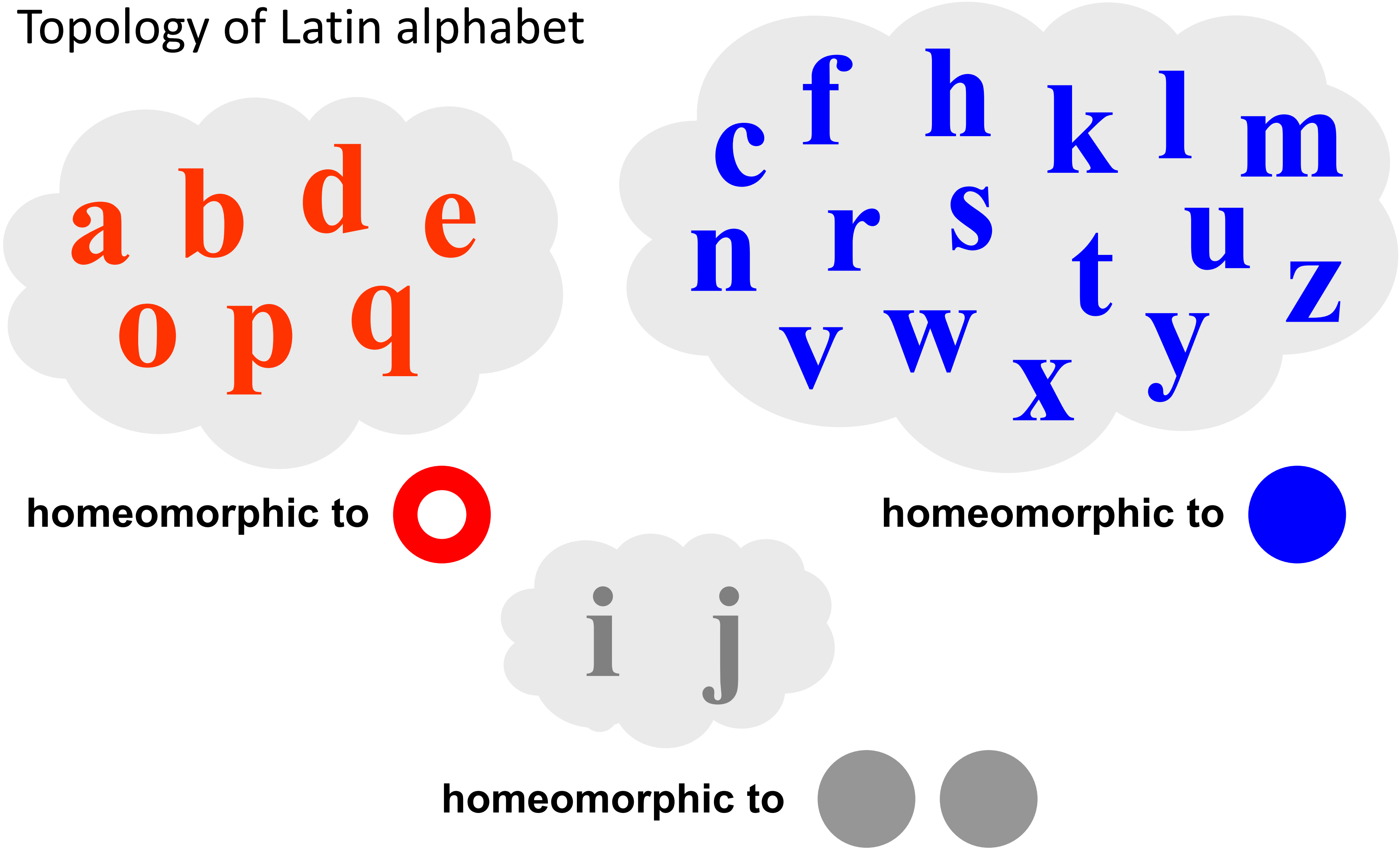
A **bijective** (one-to-one and onto) continuous function with a continuous inverse is called a **homeomorphism**

Homeomorphisms copy topology – homeomorphic spaces are **topologically equivalent**

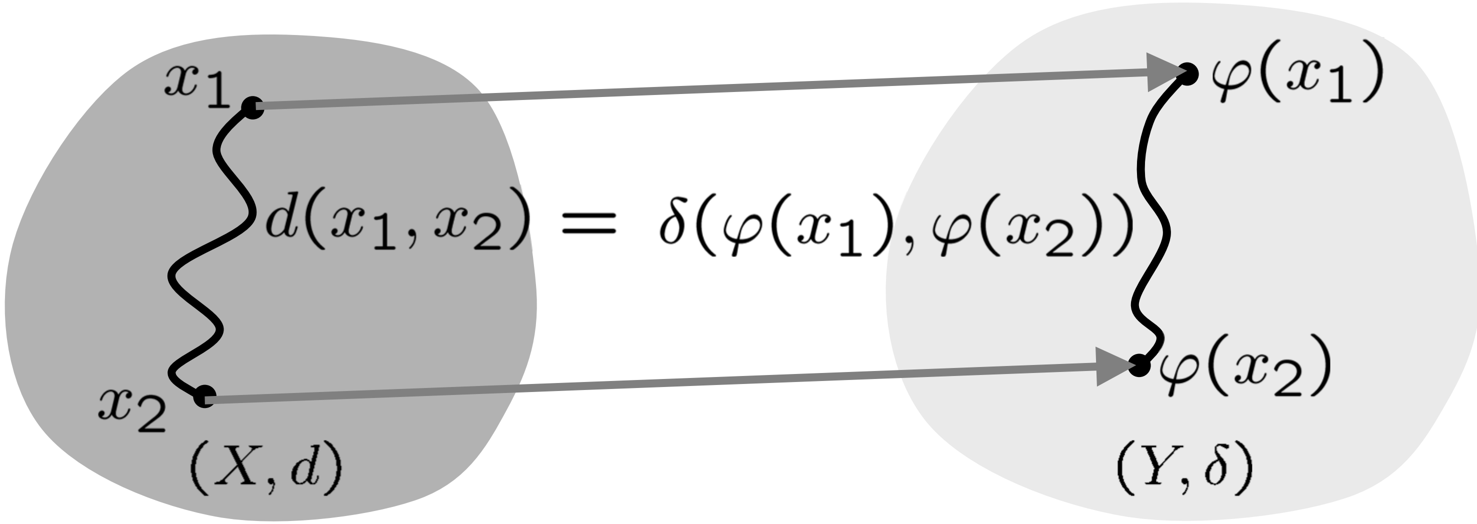
Torus and cup are homeomorphic

Homeomorphisms

Topology of Latin alphabet



Isometries



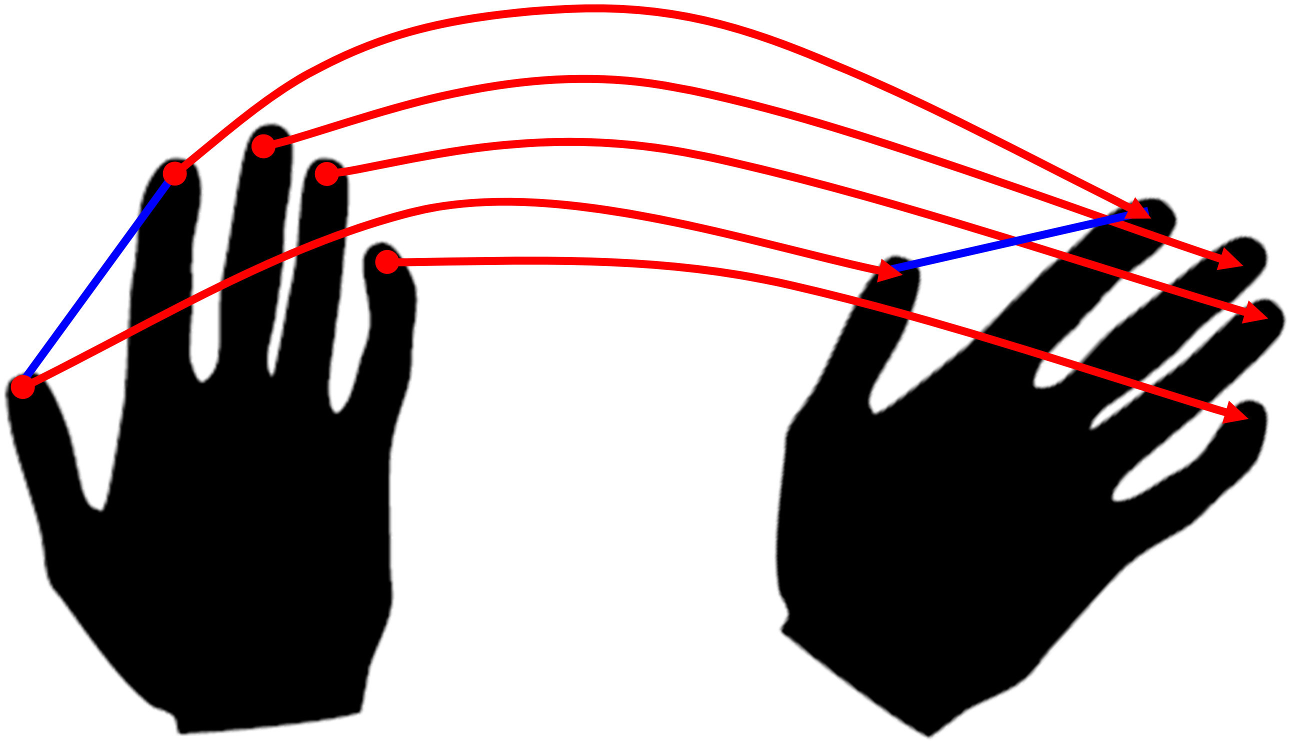
- Two metric spaces (X, d) and (Y, δ) are equivalent if there exists a **distance-preserving** map (**isometry**) $\varphi : (X, d) \rightarrow (Y, \delta)$ satisfying

$$\delta \circ (\varphi(x_1), \varphi(x_2)) = d(x_1, x_2)$$

- Such (X, d) and (Y, δ) are called **isometric**, denoted $(X, d) \sim (Y, \delta)$
- Isometries copy **metric geometries** – isometric spaces are equivalent from the point of view of metric geometry

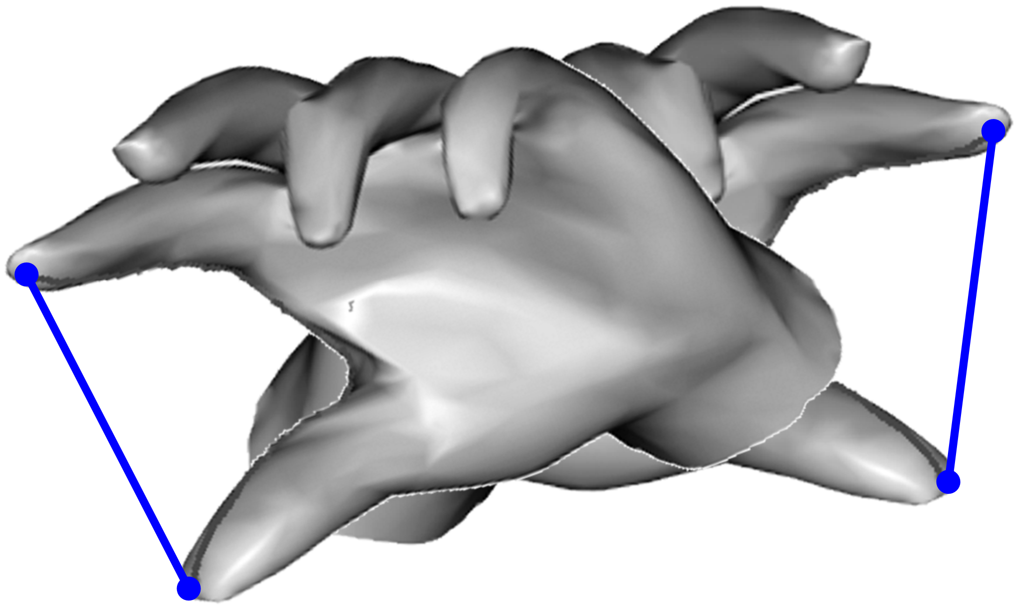
Isometries

Euclidean isometries



Isometries

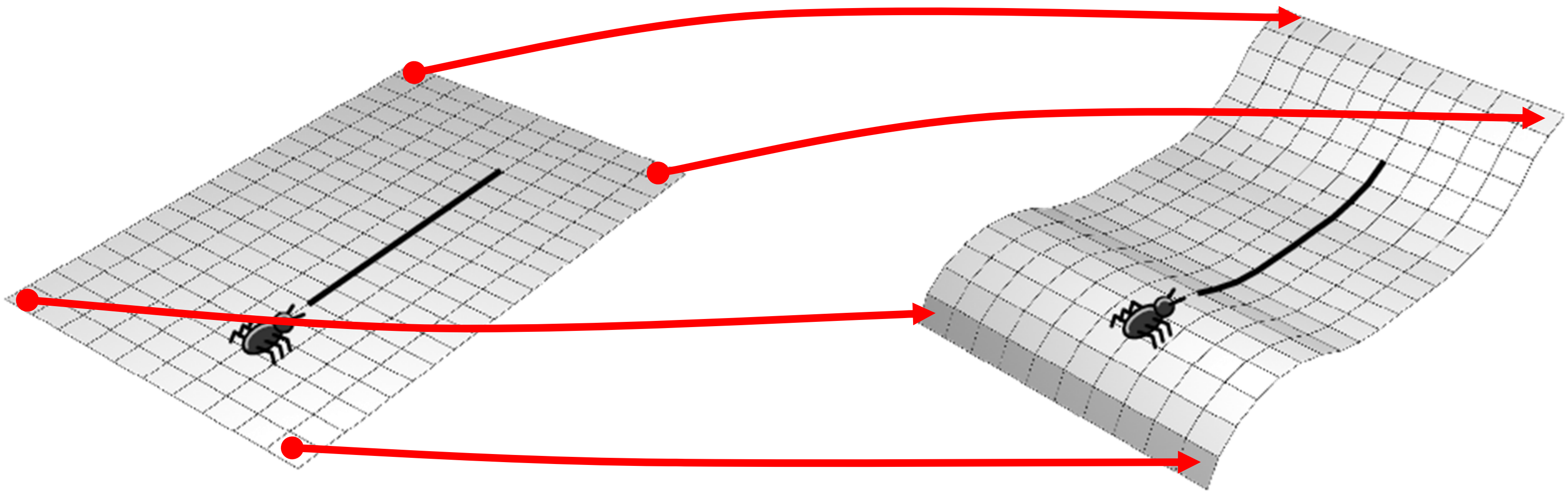
Euclidean isometries



Rotation **Translation** **Reflection**

Isometries

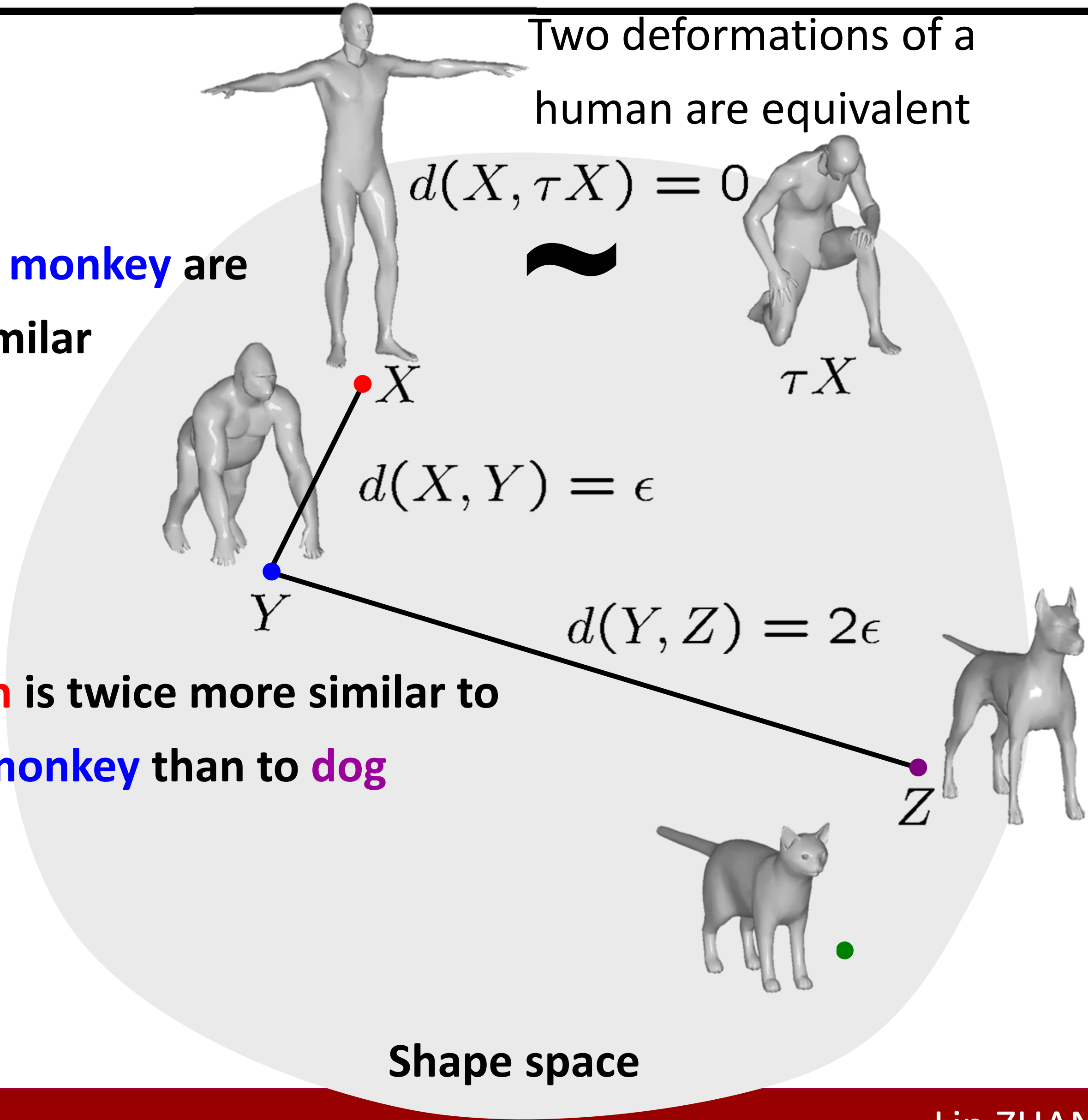
Geodesic isometries



Similarity as metric

Two deformations of a human are equivalent

Human and monkey are ε-similar



Human is twice more similar to monkey than to dog

Shape space

Outline

- Introduction
- Basic concepts in geometry
- Discrete geometry
  - Metric for discrete geometry
  - Sampling
- Rigid shape analysis
  - Euclidean isometries removal
  - ICP-based shape matching

Metric for discrete geometry

Discretization

Continuous world

- Surface X
- Metric d_X
- Topology

Discrete world

- Sampling

  $$X' = \{x_1, ..., x_N\} \subset X$$

- Discrete metric (matrix of distances) $D_X = (d_X(x_i, x_j))$
- Discrete topology (connectivity)

Metric for discrete geometry

How to compute the intrinsic metric?

- So far, we represented X itself.
- Our model of non-rigid shapes as metric spaces (X, d_X) involves the **intrinsic metric**

$$d_X(x, x') = \min_{\Gamma(x,x')} \int_{\Gamma} d\ell$$

- **Sampling** procedure requires d_X as well.
- We need a tool to **compute geodesic distances** on X.

Metric for discrete geometry

Shortest path problem

Metric for discrete geometry

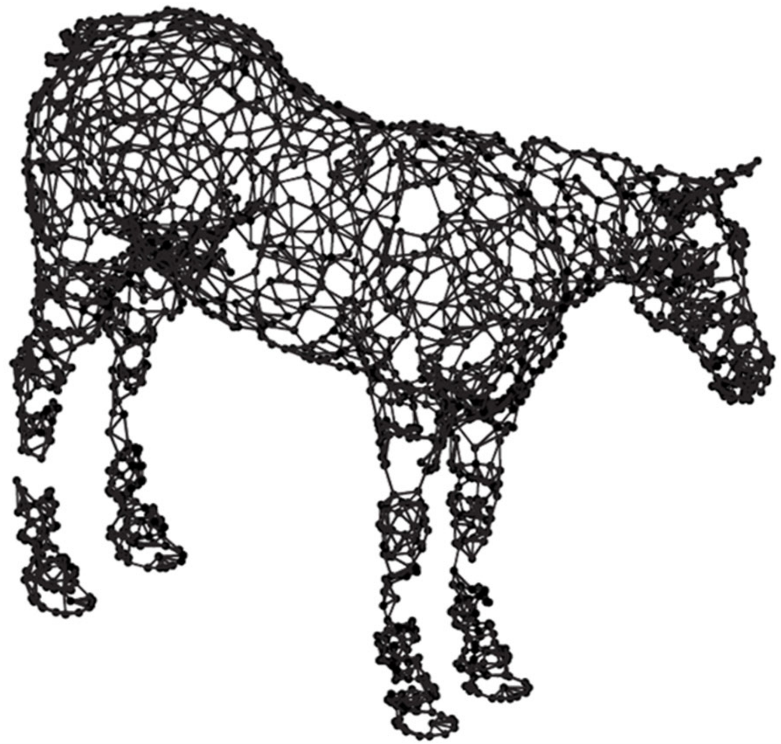
Shapes as graphs

- **Sample** the shape at N vertices $X = \{x_1, ..., x_N\}$.
- Represent shape as an **undirected graph**

$$G = (X, E)$$

- $E \subseteq X \times X$ set of **edges** representing **adjacent** vertices.
- Define **length function** $L : E \to \mathbb{R}$ measuring **local distances** as **Euclidean** ones,

$$L(x_i, x_j) = \|x_i - x_j\|_2$$



Metric for discrete geometry

Shapes as graphs

- **Path** between $x_i, x_j \in X$ is an **ordered set of connected edges**

$$\begin{aligned}\Gamma(x_i, x_j) &= \{e_1, e_2, ..., e_k\} \subset E \\ &= \{(x_{i_1}, x_{i_2}), (x_{i_2}, x_{i_3}), ..., (x_{i_{k-1}}, x_{i_k}), (x_{i_k}, x_{i_{k+1}})\}\end{aligned}$$

  where $x_{i_1} = x_i$ and $x_{i_{k+1}} = x_j$.

- **Path length** = sum of edge lengths

$$L(\Gamma(x_i, x_j)) = \sum_{n=1}^{k} L(e_n) = \sum_{n=1}^{k} L(x_{i_n}, x_{i_{n+1}})$$

Metric for discrete geometry

Geodesic distance

- **Shortest path** between $x_i, x_j \in X$

$$\Gamma^*(x_i, x_j) = \arg \min_{\Gamma(x_i, x_j)} L(\Gamma(x_i, x_j))$$

- **Length metric** in graph

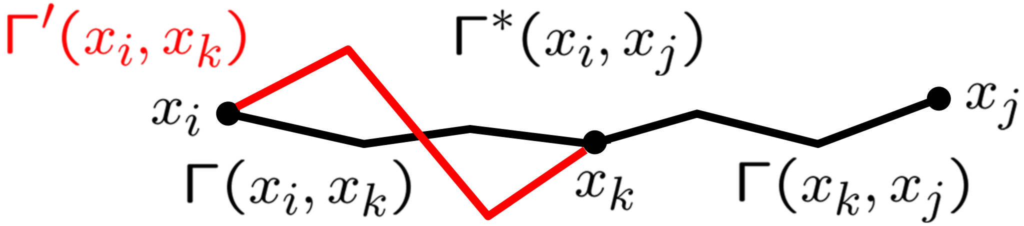
$$d_L(x_i, x_j) = \min_{\Gamma(x_i, x_j)} L(\Gamma(x_i, x_j))$$

- Approximates the **geodesic distance** $d_X \approx d_L$ on the shape.
- **Shortest path problem:** compute $\Gamma^*(x_i, x_j)$ and $d_L(x_i, x_j)$ between any $x_i, x_j \in X$.
- *Alternatively:* given a **source point** $x_0 \in X$, compute the **distance map** $d(x_i) = d_L(x_0, x_i)$.

Metric for discrete geometry

Bellman's principle of optimality

- Let $\Gamma^*(x_i, x_j)$ be **shortest path** between $x_i, x_j \in X$ and $x_k \in \Gamma^*(x_i, x_j)$ a point on the path.
- Then, $\Gamma(x_i, x_k)$ and $\Gamma(x_k, x_j)$ are **shortest sub-paths** between x_i, x_k , and x_k, x_j .



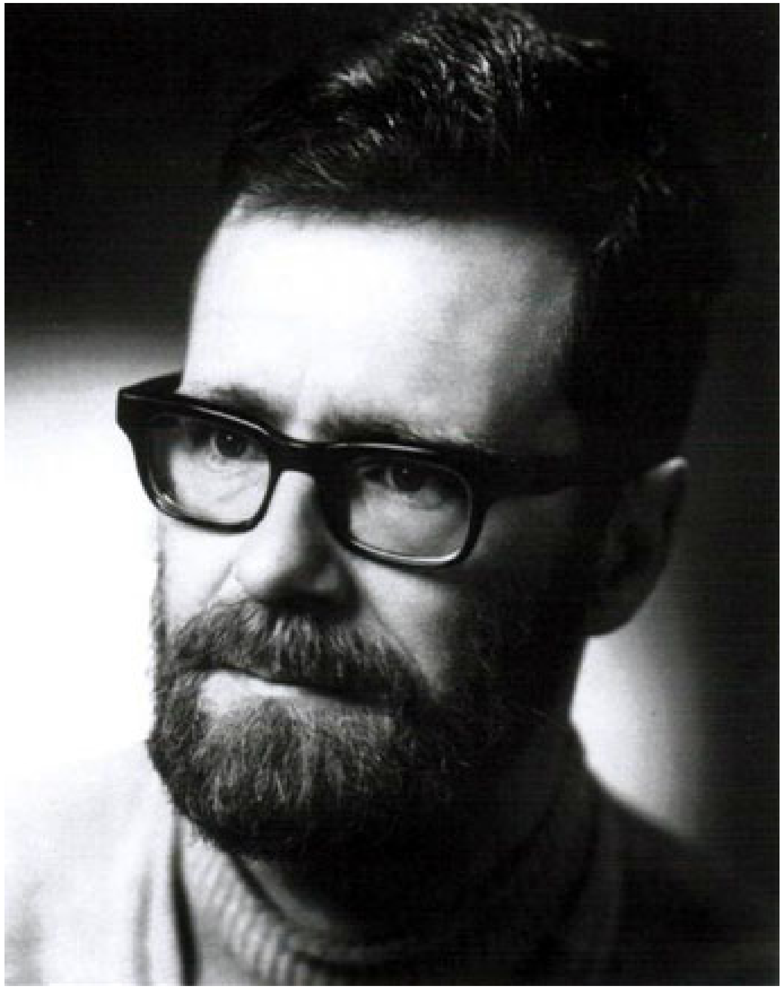
Richard Bellman
(1920-1984)

- Suppose there exists a **shorter** path $\Gamma'(x_i, x_k)$.

$$\begin{aligned} L(\Gamma'(x_i, x_j)) &= L(\Gamma'(x_i, x_k)) + L(\Gamma(x_k, x_j)) \\ &< L(\Gamma(x_i, x_k)) + L(\Gamma(x_k, x_j)) = L(\Gamma^*(x_i, x_j)) \end{aligned}$$

- **Contradiction** to $\Gamma^*(x_i, x_j)$ being shortest path.

Metric for discrete geometry



Edsger Wybe Dijkstra (*1930–2002*)

Metric for discrete geometry

Dijkstra's algorithm

- Initialize $d(x_0) = 0$ and $d(x_i) = \infty$ for the rest of the graph;
  Initialize **queue of unprocessed vertices** $Q = X$.
- While $Q \neq \emptyset$
  - Find vertex x with **smallest value** of d ,

    $$x = \arg\min_{x \in Q} d(x)$$

  - For each **unprocessed adjacent vertex** $x' \in \mathcal{N}(x) \cap Q$,

    $$d(x') = \min\{d(x'), d(x) + L(x, x')\}$$

  - **Remove** x from Q .
- Return **distance map** $d(x_i) = d_L(x_0, x_i)$.

Metric for discrete geometry

Troubles with the metric

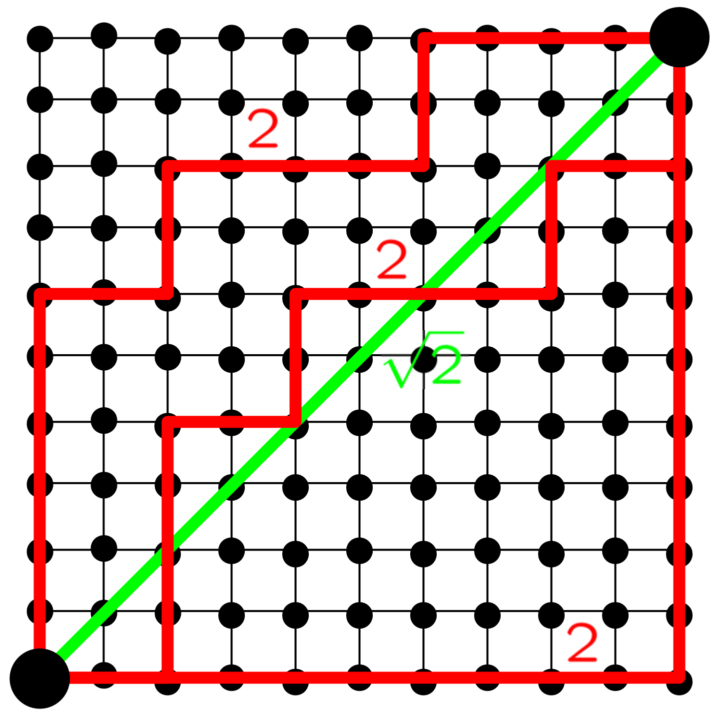
- Grid with **4-neighbor** connectivity.
- True **Euclidean distance**

$$d_{\mathbb{R}^2} = \sqrt{2}$$

- Shortest path in **graph** (**not unique**)

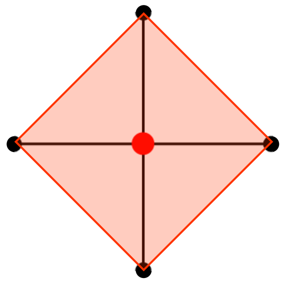
$$d_L = 2$$

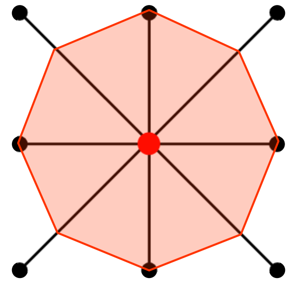
- Increasing **sampling density** does not help.

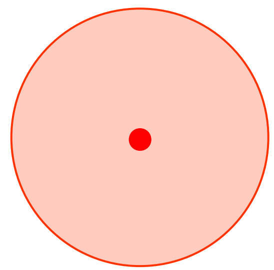


Metric for discrete geometry

Metrication error







4-neighbor topology | **8-neighbor** topology | Continuous $\mathbb{R}^2$

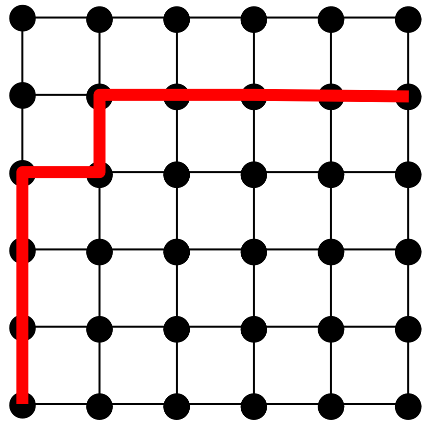
Manhattan distance | **Euclidean distance**

$$d_{L_1} = \sum_i |x_1^i - x_2^i|$$

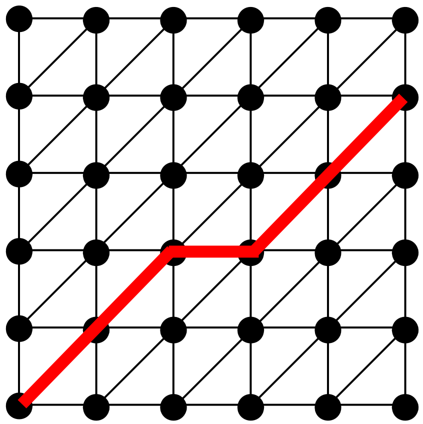
$$d_{L_2} = \sqrt{\sum_i (x_1^i - x_2^j)^2}$$

- **Graph representation** induces an **inconsistent metric**.
- Increasing **sampling size** does not make it consistent.
- Neither does increasing **connectivity**.

Metric for discrete geometry

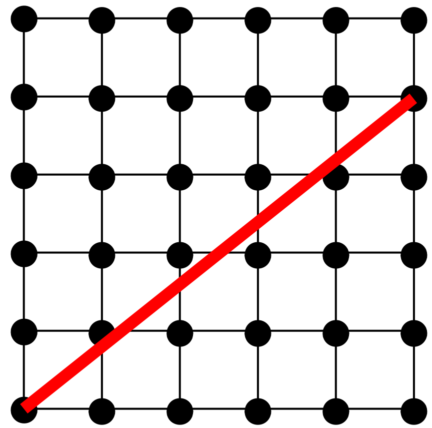


Discrete solution



- Stick to **graph** representation
- Change **connectivity**
- Consistency guaranteed under certain conditions

Continuous solution

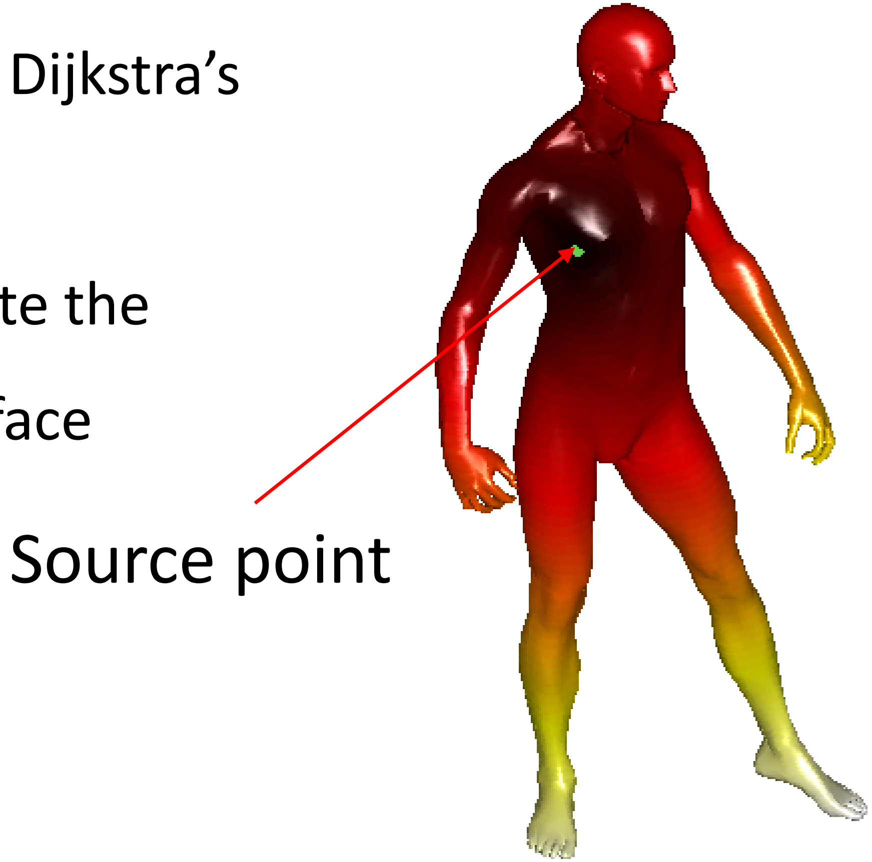


- Stick to given **sampling**
- Compute distance map on the **surface**
- **New algorithm!**

Metric for discrete geometry

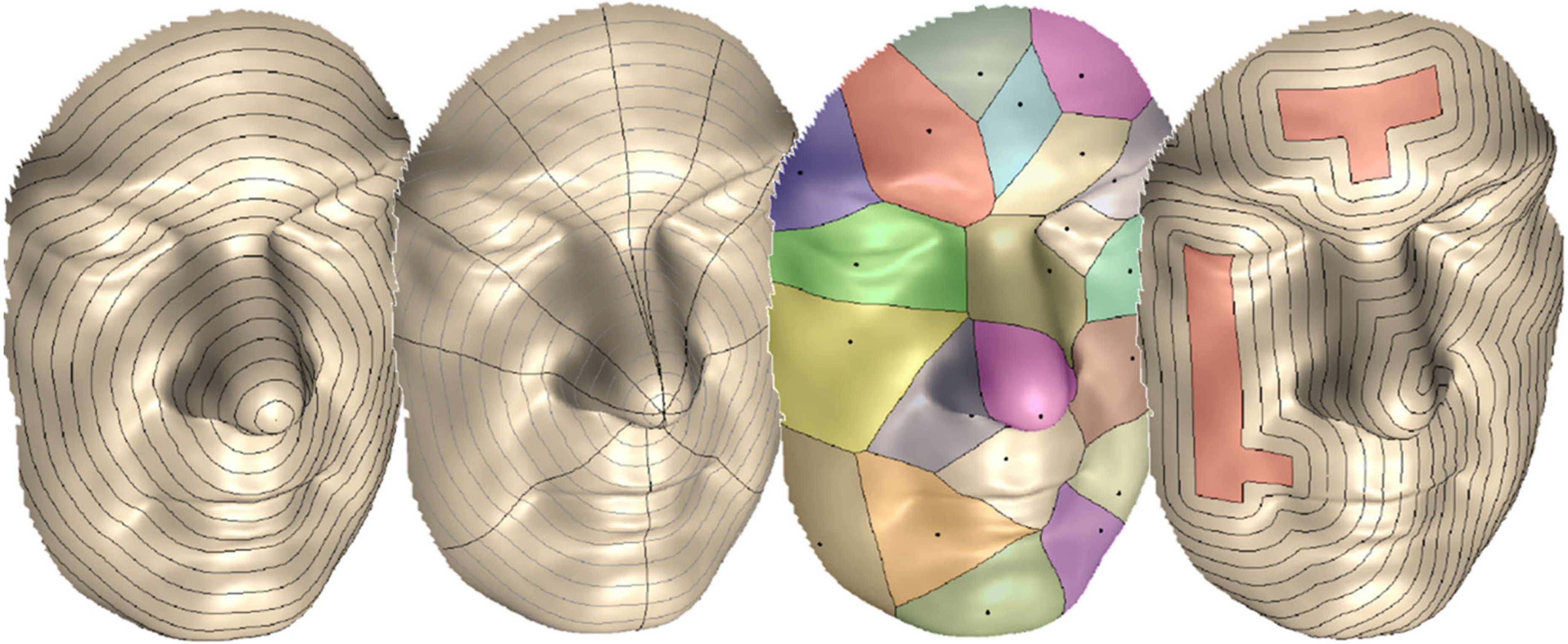
To solve the above issue, we can use ***fast marching methods***

- A continuous variant of Dijkstra's algorithm
- Consistently approximate the intrinsic metric on the surface



Metric for discrete geometry

Usages of fast marching



Geodesic distances

Minimal geodesics

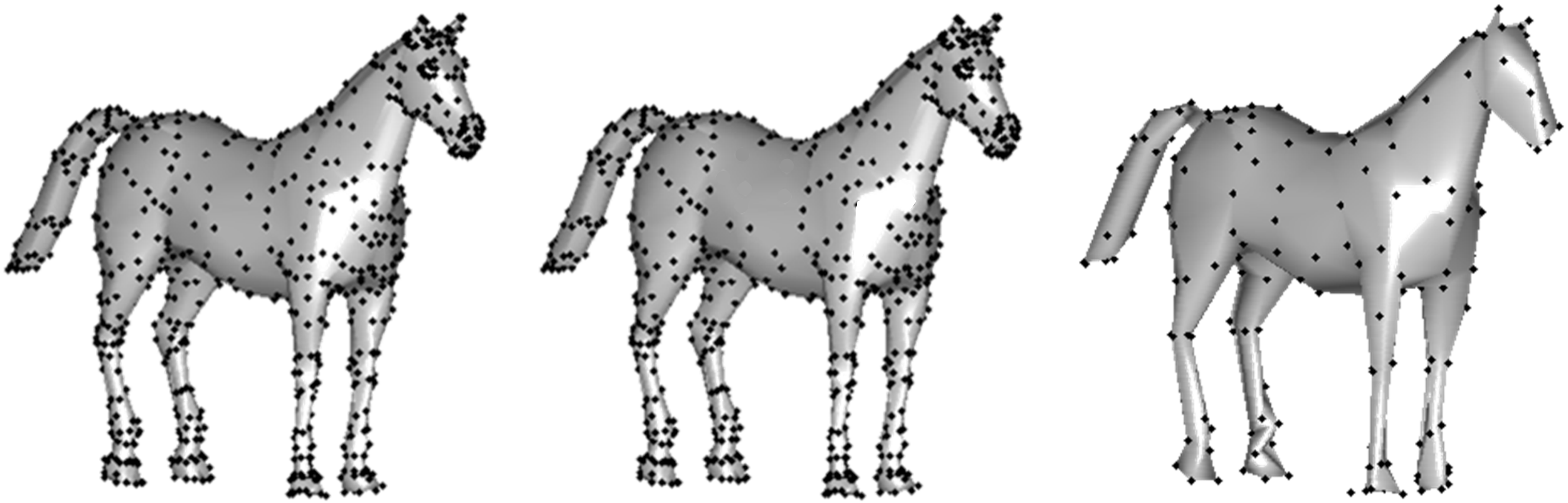
Voronoi tessellation & sampling

Offset curves

Outline

- Introduction
- Basic concepts in geometry
- Discrete geometry
  - Metric for discrete geometry
  - Sampling
- Rigid shape analysis
  - Euclidean isometries removal
  - ICP-based shape matching

How good is a sampling?



Sampling density

- How to quantify **density** of sampling?
- X' is an r-**covering** of X if

$$\bigcup_{x_i \in X'} B_r(x_i) = X$$

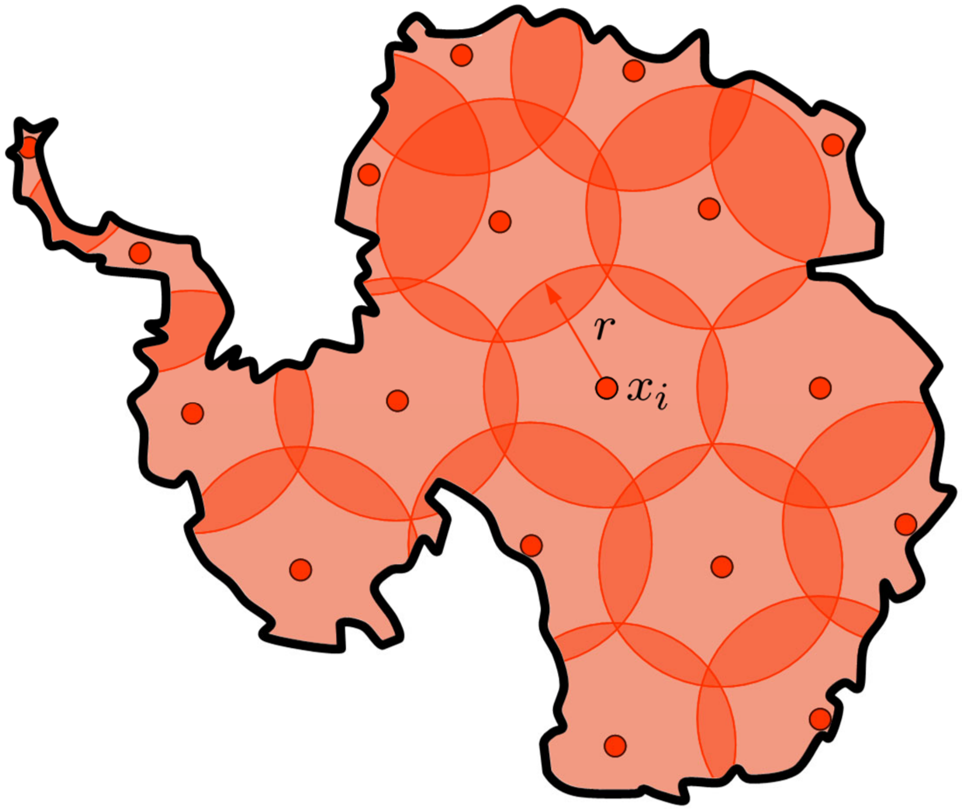
Alternatively:

$$d_X(x, X') \leq r$$

for all $x \in X$, where

$$d_X(x, X') = \inf_{x_i \in X'} d_X(x, x_i)$$

is the **point-to-set distance**.

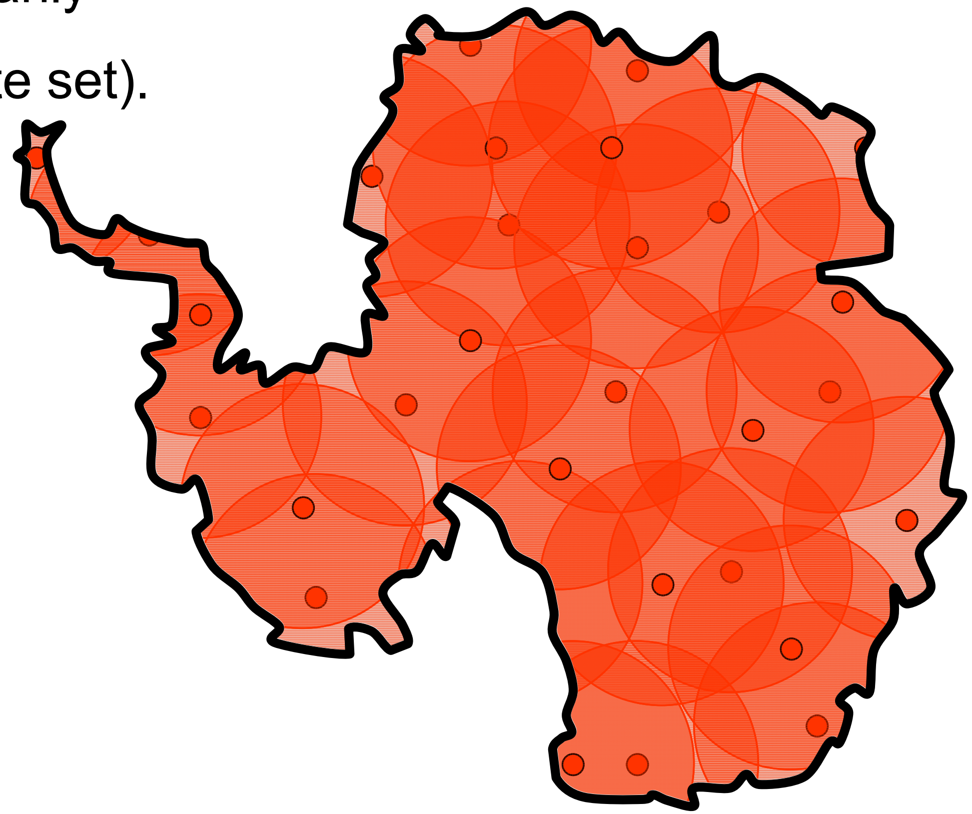


Sampling efficiency

- Are all points **necessary**?
- An r-covering may be unnecessarily dense (may even not be a discrete set).
- Quantify how well the samples are **separated**.
- X' is r'**-separated** if

$$d_{X'}(x_i, x_j) \geq r'$$

for all $x_i, x_j \in X'$.
- For $r' > 0$, an r'-separated set is **finite** if X is **compact**.

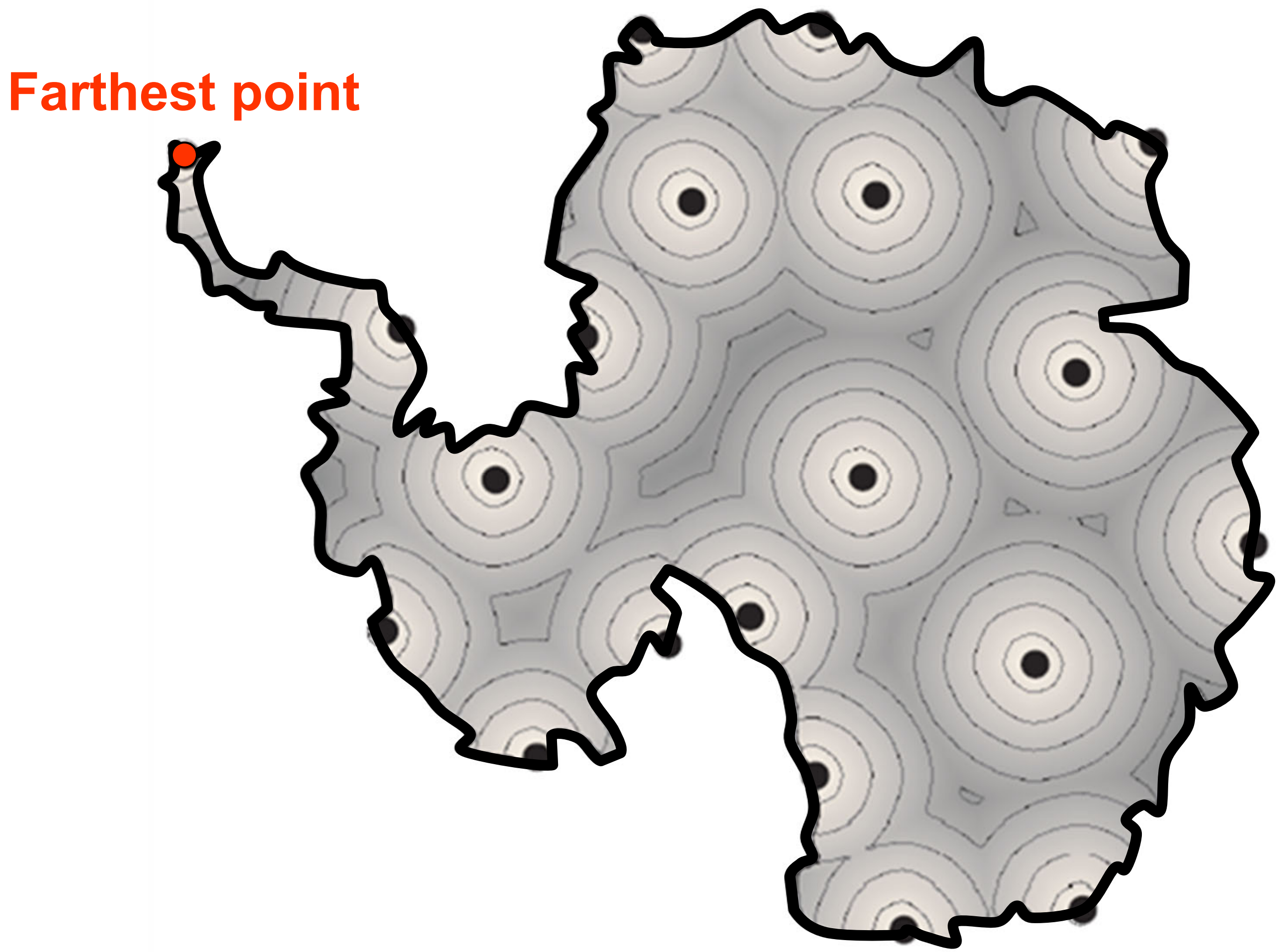


Also an *r*-covering!

Farthest point sampling

- Good sampling has to be **dense** and **efficient** at the same time.
- Find a r-**separated** and r-**covering** X' of X.
- Achieved using **farthest point sampling**.

Farthest point sampling



Farthest point sampling

- Start with some $X' = \{x_1 \in X\}$.
- Determine **sampling radius**

$$r = \max_{x \in X} d_X(x, X')$$

- If $r \leq r_{\text{target}}$ **stop**.
- Find the **farthest point** from X

$$x' = \arg\max_{x \in X} d_X(x, X')$$

- **Add** x' to X'

Farthest point sampling

- Outcome: r-separated r-covering of X.
- Produces sampling with **progressively increasing** density.
- A **greedy algorithm**: previously added points remain in X'.
- There might be another r-separated r-covering containing less points.
- In practice used to **sub-sample** a densely sampled shape.
- Straightforward time complexity: $\mathcal{O}(MN)$

  M number of points in dense sampling, N number of points in X'.
- Using **efficient data structures** can be reduced to $\mathcal{O}(N \log M)$.

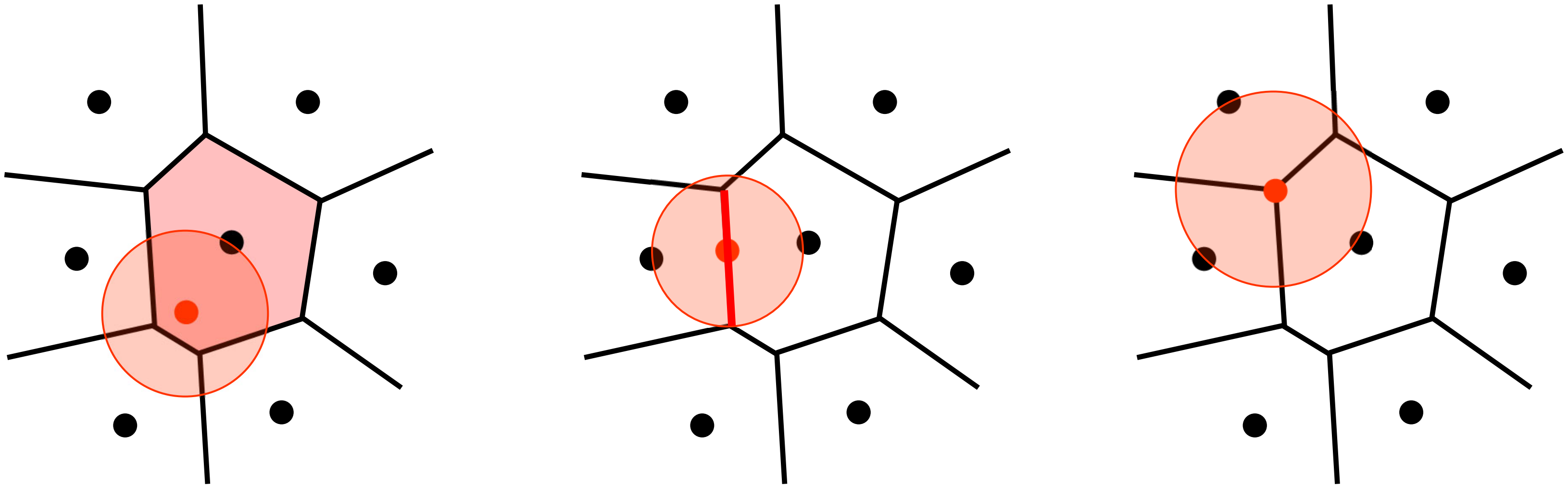
Sampling as representation

- Sampling **represents** a region on X as a single point $x_i \in X'$.
- Region of points on X **closer** to x_i than to any other x_j

$$V_i(X') = \{x \in X : d_X(x, x_i) < d_X(x, x_j), x_{j \neq i} \in X'\}$$

- **Voronoi region** (Dirichlet or Voronoi-Dirichlet region, Thiessen polytope or polygon, Wigner-Seitz zone, domain of action).

Voronoi decomposition



Voronoi region **Voronoi edge** **Voronoi vertex**

- A point $x \in X$ can belong to one of the following
  - **Voronoi region** V_i (x is closer to x_i than to any other x_j).
  - **Voronoi edge** $V_{ij} = \overline{V}_i \cap \overline{V}_j$ (x is **equidistant** from x_i and x_j).
  - **Voronoi vertex** $V_{ijk} = \overline{V}_i \cap \overline{V}_j \cap \overline{V}_k$ (x is equidistant from three points x_i, x_j, x_k).

Voronoi decomposition

Voronoi decomposition

- Voronoi regions are **disjoint**.
- Their closure

$$\bigcup_i \overline{V}_i = X$$

covers the entire X.

- Cutting X along Voronoi edges produces a collection of **tiles** $\{V_i\}$.
- The tiles are **topological disks** (are homeomorphic to a disk).

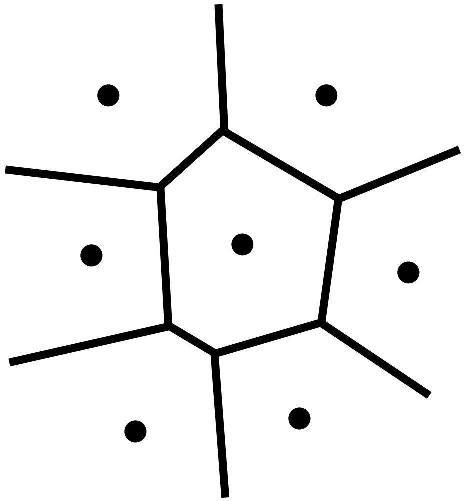

Voronoi tessellations in Nature

Delaunay tessellation

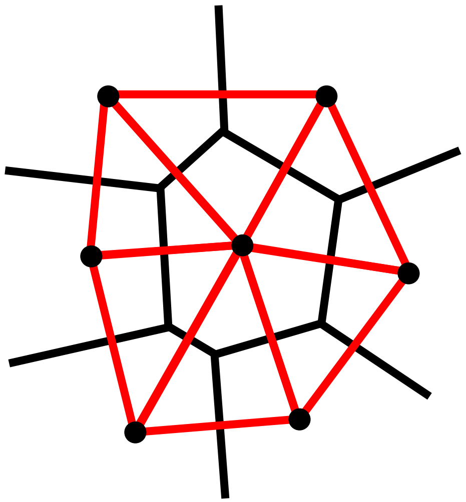
Define connectivity as follows: a pair of points whose Voronoi cells are adjacent are connected

The obtained connectivity graph is **dual** to the Voronoi diagram and is called **Delaunay tesselation**

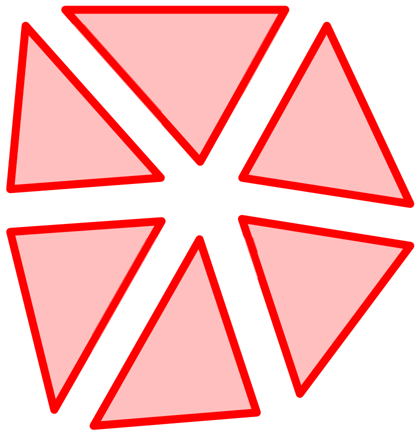
Boris Delaunay (1890-1980)



Voronoi regions



Connectivity

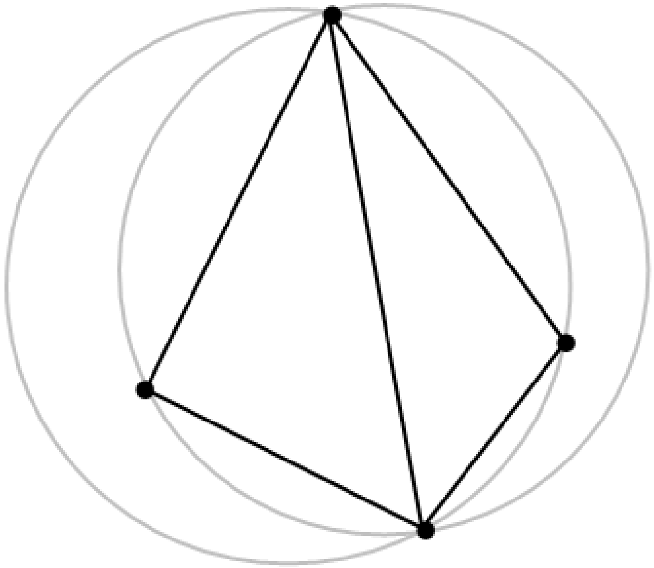


Delaunay tesselation

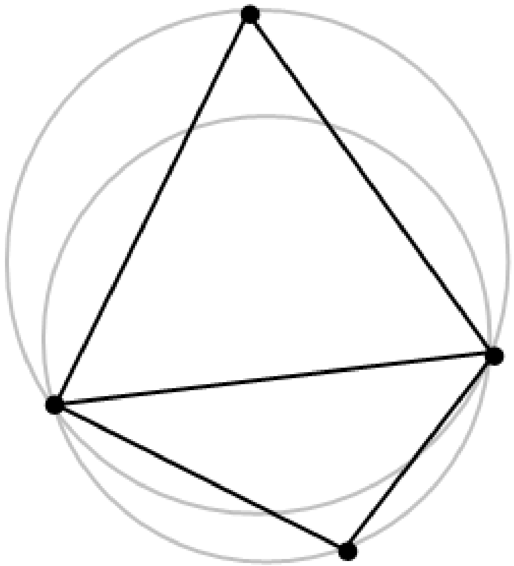
Delaunay tessellation

- For a set P of points in the (d-dimensional) Euclidean space, a Delaunay triangulation is a triangulation DT(P) such that no point in P is inside the circumhypersphere of any simplex in DT(P)
- It is known that there exists a unique Delaunay triangulation for P if P is a set of points in general position
- In the plane, the Delaunay triangulation maximizes the minimum angle

Delaunay tessellation

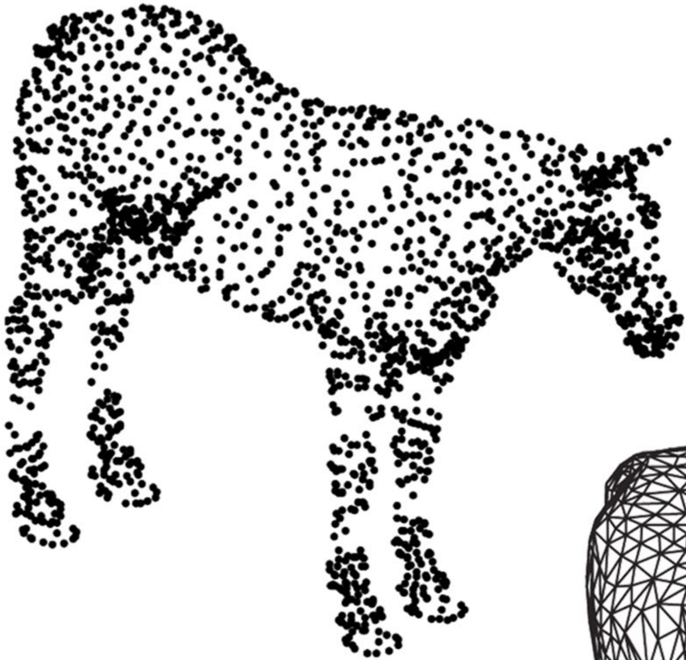


This triangulation does not meet the Delaunay condition (the circumcircles contain more than three points)

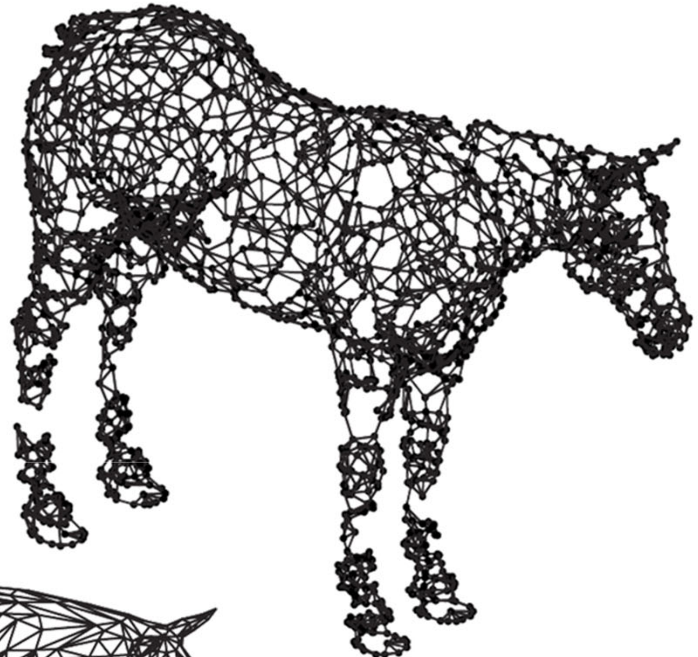


Flipping the common edge produces a Delaunay triangulation for the four points

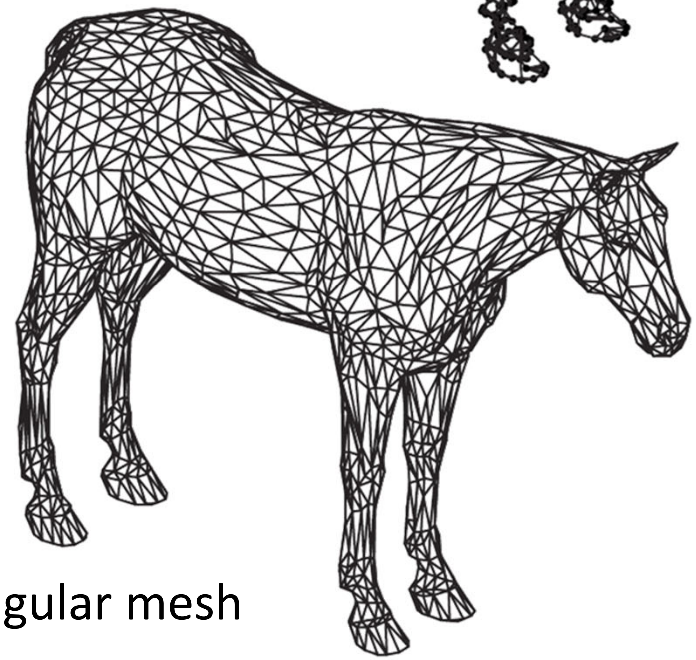
Shape representation



Cloud of points



Graph



Triangular mesh

Triangular meshes

A structure of the form (I, E, T) consisting of

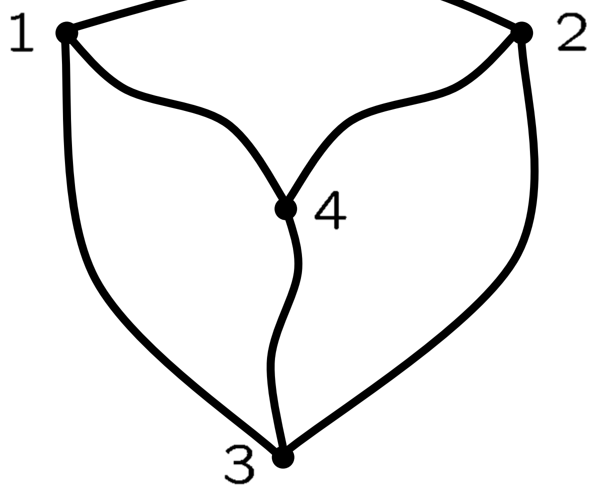
- **Vertices** $I = \{1, ..., N\}$
- **Edges** $E = \{(i, j) \in I \times I : x_j \in \mathcal{N}(x_i)\}$
- **Faces** $T = \{(i, j, k) \in I \times I \times I : (i, j), (i, k), (k, j) \in E\}$

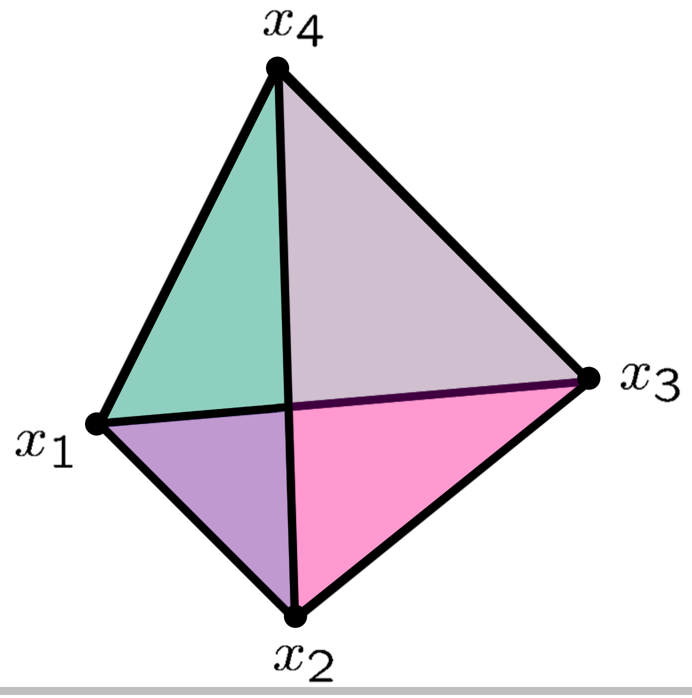
is called a **triangular mesh**

The mesh is a purely **topological** object and does not contain any geometric properties

The faces can be represented as an $N_F \times 3$ matrix of indices, where each row is a vector of the form $t_k = (t_k^1, t_k^2, t_k^3)$, $t_k^i \in I$ and $k = 1, ..., N_F$

Example of triangular mesh





Vertices	1	2	3	4
Edges	$(1,2)$	$(1,3)$	$(1,4)$	
	$(4,2)$	$(4,3)$	$(2,3)$	
Faces	$(2,4,3)$	$(1,4,2)$		
	$(3,4,1)$	$(2,3,1)$		

Topological

Coordinates	$(0.5, 0.86, 0)$
	$(0, 0, 0)$
	$(1, 0, 0)$
	$(0.5, 0.28, 0.86)$

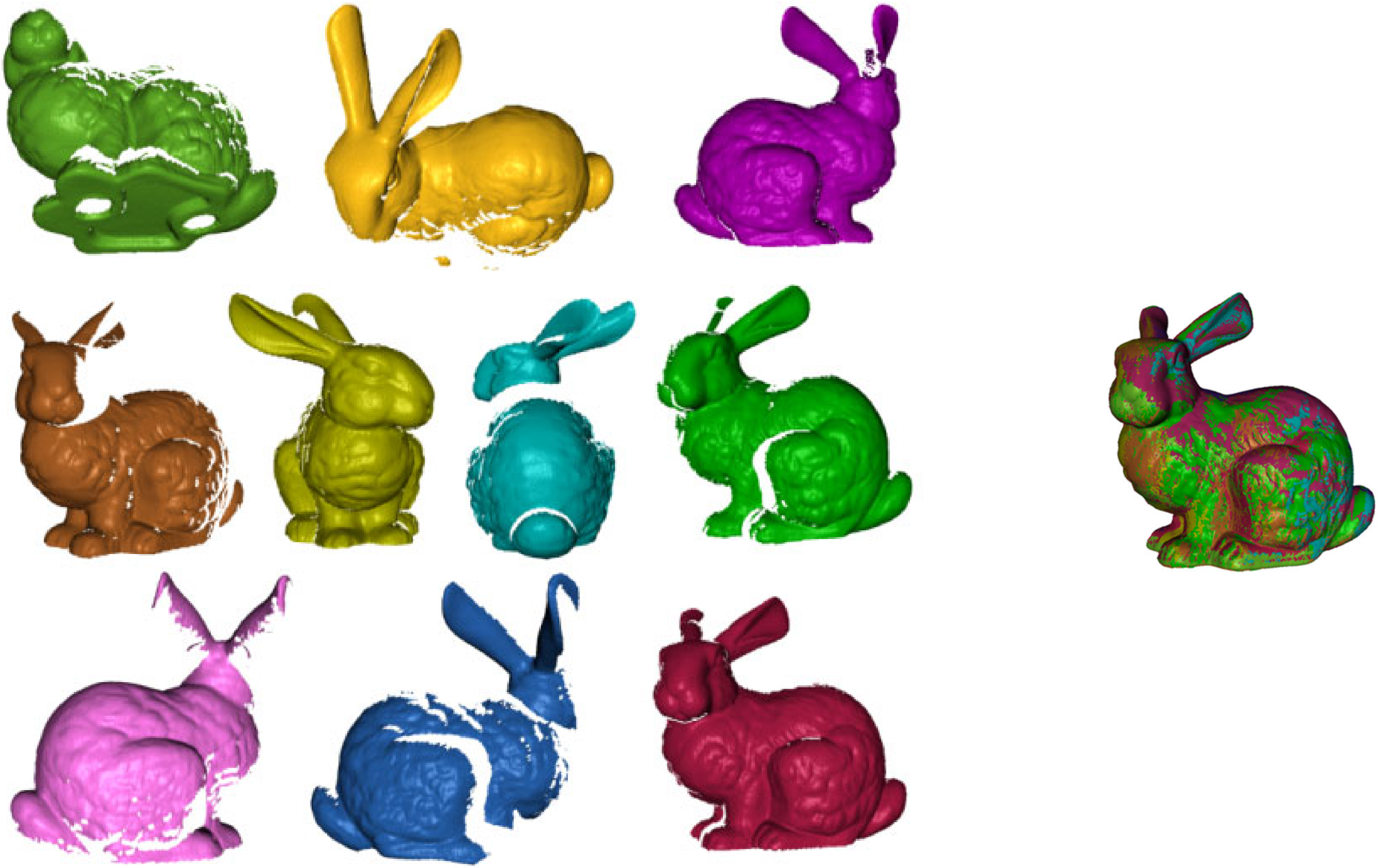
Geometric

Outline

- Introduction
- Basic concepts in geometry
- Discrete geometry
  - Metric for discrete geometry
  - Sampling
- Rigid shape analysis
  - Euclidean isometries removal
  - ICP-based shape matching

A fairy tale shape similarity problem

Extrinsic shape similarity



Extrinsic shape similarity

- Given two shapes X and Y, find the degree of their **incongruence**.
- Compare X and Y as subsets of the Euclidean space $\mathbb{R}^3$.
- Invariance to rigid motion: **rotation**, **translation**, (**reflection**):

$$x' = Rx + t$$

- R is a rotation matrix, $R^\top R = I$
- t is a translation vector

How to get rid of Euclidean isometries?

- How to remove translation and rotation ambiguity?
- Find some “canonical” placement of the shape X in $\mathbb{R}^3$?
- **Extrinsic centroid** (**center of mass**, or **center of gravity**):

$$x_0 = \frac{\int_X x dx}{\int_X dx}$$

- Set $t = -x_0$ to resolve translation ambiguity.
- Three degrees of freedom remaining…

How to get rid of Euclidean isometries?

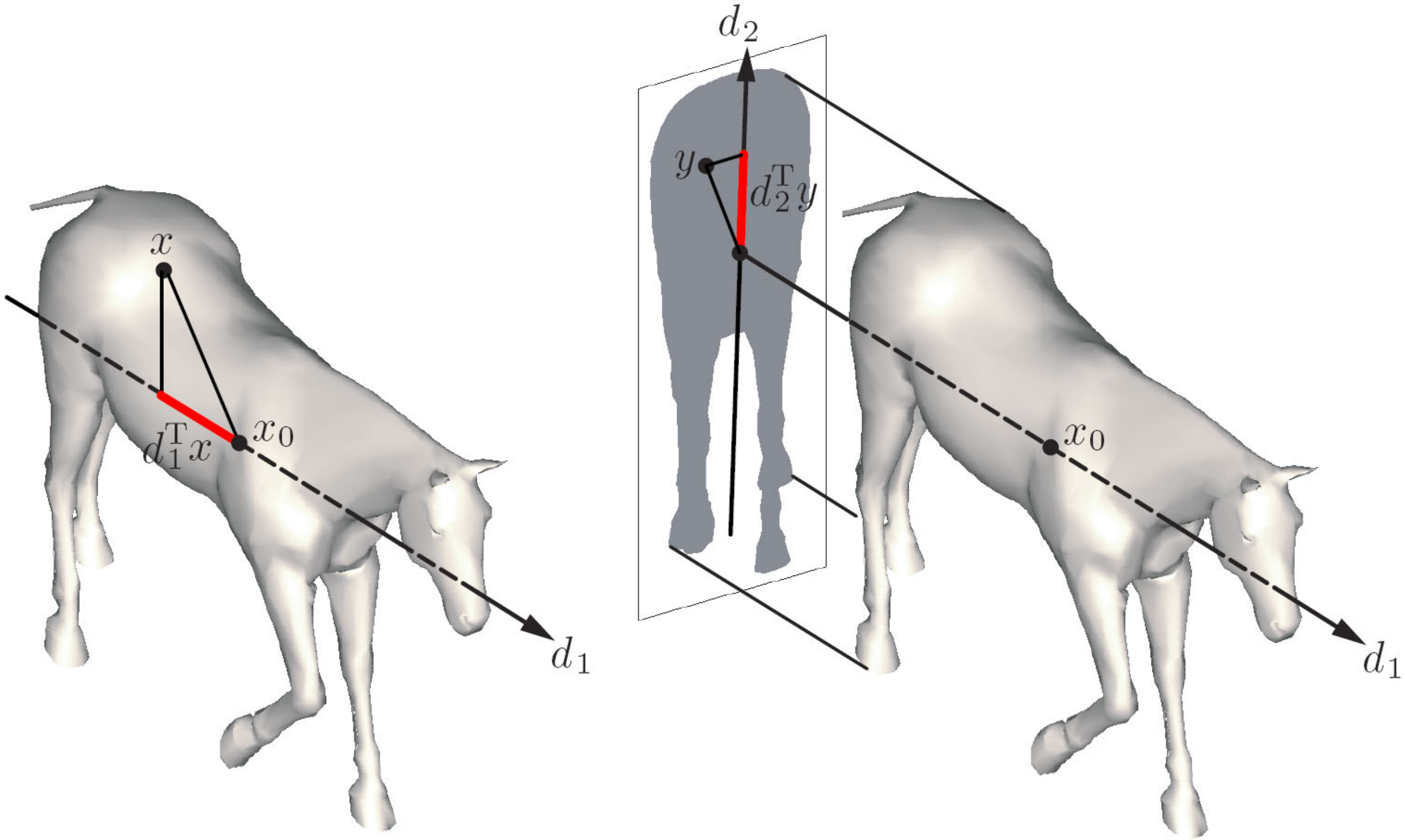
- Find the direction d_1 in which the surface has **maximum extent**.
- Maximize **variance** of projection of X onto d_1

$$
\begin{aligned}
d_1 &= \arg \max_{d_1 : \|d_1\|_2 = 1} \int_X (d^\top x)^2 dx \\
&= \arg \max_{d_1 : \|d_1\|_2 = 1} d_1^\top \left(\int_X x x^\top dx \right) d_1 \\
&= \arg \max_{d_1 : \|d_1\|_2 = 1} d_1^\top \Sigma_X d_1
\end{aligned}
$$

- Σ_X is the **covariance matrix**
- d_1 is the first **principal direction**

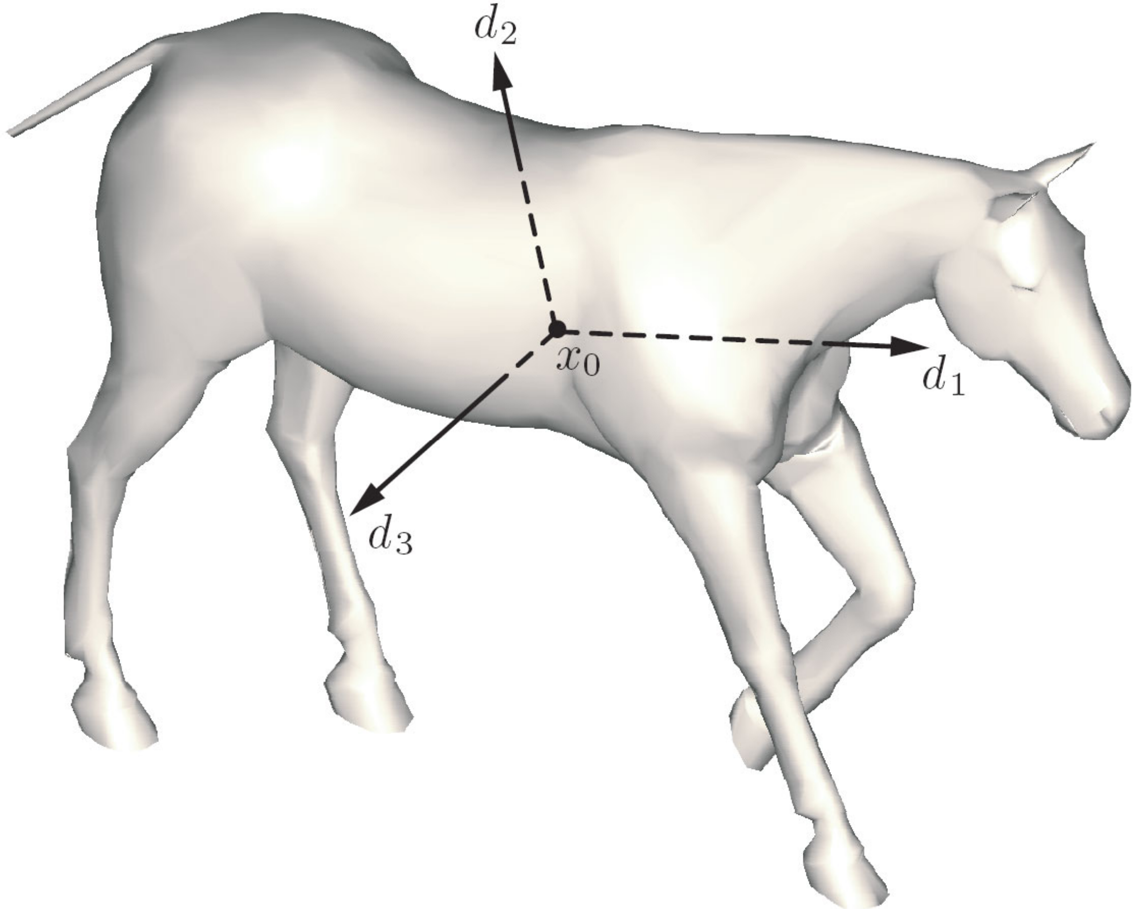
How to get rid of Euclidean isometries?

- Project X on the plane orthogonal to d_1.
- Repeat the process to find second and third principal directions d_2, d_3.



How to get rid of Euclidean isometries?

Canonical basis



- $d_1 \perp d_2 \perp d_3$ span a canonical orthogonal basis for X in $\mathbb{R}^3$.

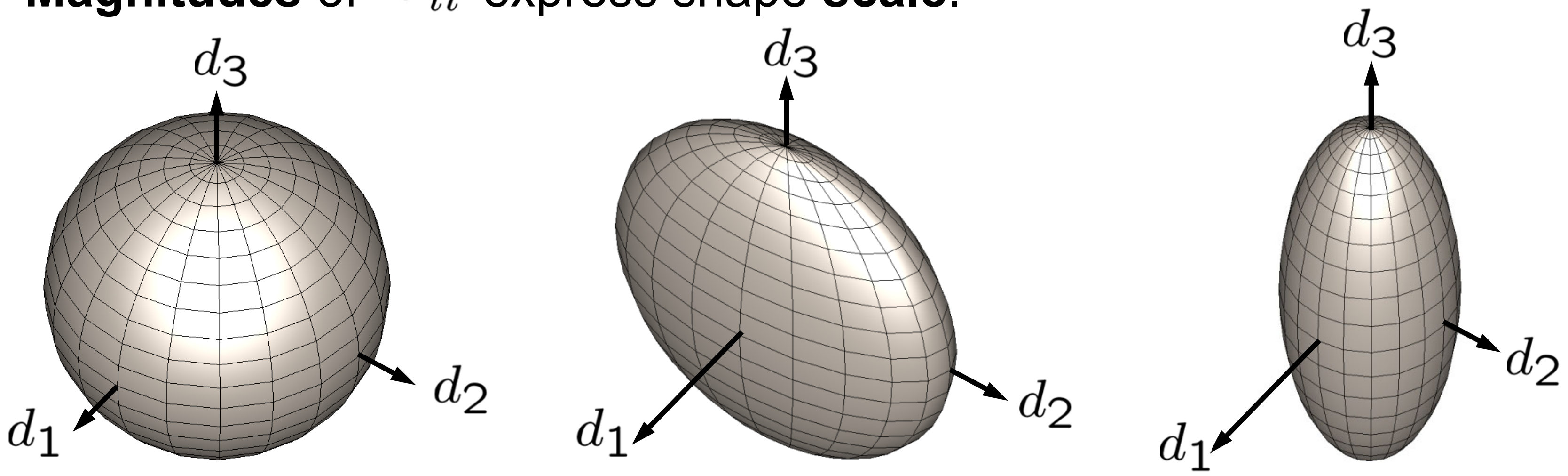
How to get rid of Euclidean isometries?

- Direction maximizing $d_1^\top \Sigma_X d_1$ = **largest eigenvector** of Σ_X.
- d_2 and d_3 correspond to the second and third eigenvectors of Σ_X.
- Σ_X admits **unitary diagonalization** $\Sigma_X = U^\top \Lambda U$.

  where $U = \begin{pmatrix} d_1^T \\ d_2^T \\ d_3^T \end{pmatrix}$.
- **Principal component analysis** (PCA), or **Karhunen-Loéve transform** (KLT), or **Hotelling transform**.

Second-order geometric moments

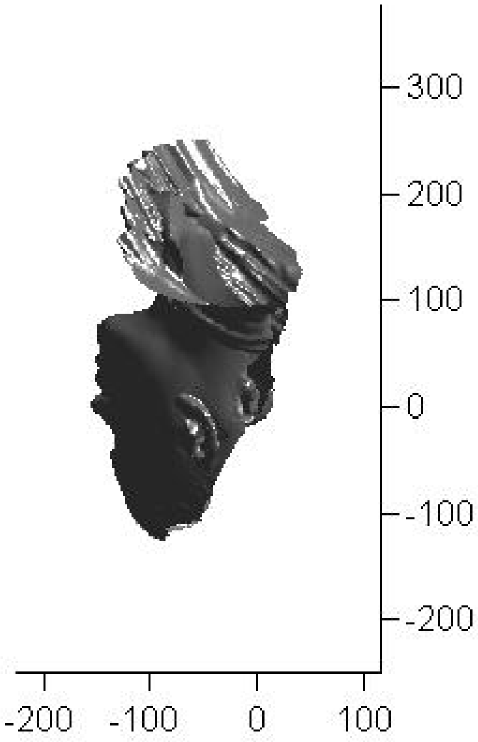
- **Eigenvalues** of Σ_X are **second-order moments** σ_{ii} of X.
- **Second-order geometric moments** of X : $\sigma_{ij} = \int_X x^i x^j dx$
- In the canonical basis, **mixed moments** σ_{ij} vanish.
- **Ratio** $\sigma_{11} : \sigma_{22} : \sigma_{33}$ describe **eccentricity** of X .
- **Magnitudes** of σ_{ii} express shape **scale**.



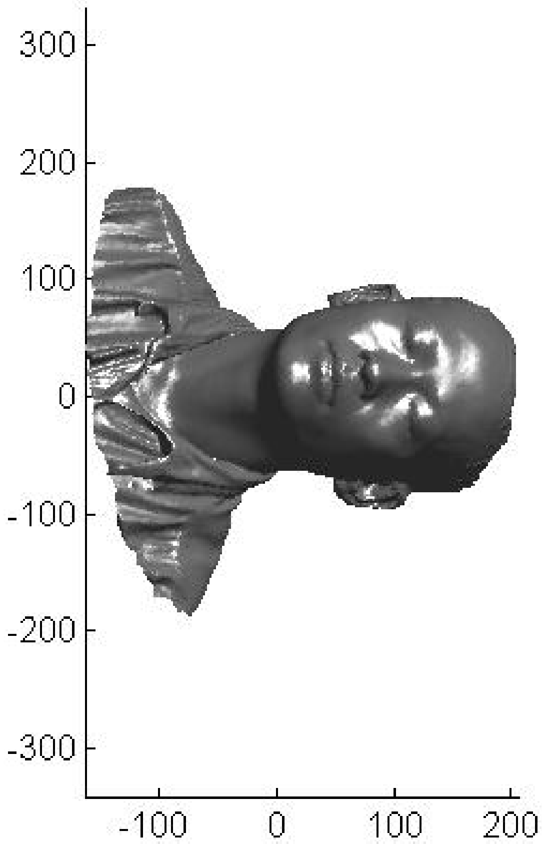
$\sigma_{11} \approx \sigma_{22} \approx \sigma_{33}$ $\sigma_{11} \ll \sigma_{22} \approx \sigma_{33}$ $\sigma_{11} \approx \sigma_{22} \ll \sigma_{33}$

How to get rid of Euclidean isometries?

Examples



Without self-alignment



With self-alignment by using PCA

Outline

- Introduction
- Basic concepts in geometry
- Discrete geometry
  - Metric for discrete geometry
  - Sampling
- Rigid shape analysis
  - Euclidean isometries removal
  - ICP-based shape matching

Iterative closest point (ICP) algorithms

- Given two point sets $\{m_i\}_{i=1}^{N}$ and $\{n_j\}_{j=1}^{M}$, find the best motion (s,R,t) bringing $\left\{sR(n_j)+t\right\}$ **as close as possible** to $\{m_i\}_{i=1}^{N}$:

$$d_{ICP}\left(\{m_i\},\{n_j\}\right)=\min_{s,R,t} d\left(\left\{sR(n_j)+t\right\},\{m_i\}\right)$$

- $d\left(\left\{sR(n_j)+t\right\},\{m_i\}\right)$ is some **shape-to-shape distance**.
- **Minimum** = extrinsic dissimilarity of $\{m_i\}_{i=1}^{N}$ and $\{n_j\}_{j=1}^{M}$.
- **Minimizer** = best alignment between $\{m_i\}_{i=1}^{N}$ and $\{n_j\}_{j=1}^{M}$.
- ICP is a **family of algorithms** differing in
  - The choice of the **shape-to-shape distance**.
  - The choice of the **numerical minimization** algorithm.

Iterative closest point (ICP) algorithms

$[s,R,T]$ = ICP ($\{m_i\}_{i=1}^{N}$, $\{n_j\}_{j=1}^{M}$) (suppose N<M)

calculate the point correspondences $\{m_i, n_i\}_{i=1}^{N}$ (closest point)

calculate the error: $\Sigma^2 = \sum_{i=1}^{N} (m_i - n_i)^2$

While not convergent

Evaluate s, R and T according to the pairs $\{m_i, n_i\}_{i=1}^{N}$

Apply s, R and T to $\{n_j\}$ to get $\{n_j^{'}\}$

Let $\{n_j\} = \{n_j^{'}\}$

Re-calculate the point correspondences $\{m_i, n_i\}_{i=1}^{N}$

re-calculate the error: $\Sigma^2 = \sum_{i=1}^{N} (m_i - n_i)^2$

End

Return s, R, T

Iterative closest point (ICP) algorithms

$[s,R,T]$ = ICP ($\{m_i\}_{i=1}^{N}$, $\{n_j\}_{j=1}^{M}$) (suppose N<M)

calculate the point correspondences $\{m_i, n_i\}_{i=1}^{N}$ (closest point)

calculate the error: $\Sigma^2 = \sum_{i=1}^{N}(m_i - n_i)^2$

Can be efficiently computed by using Delaunay triangulation

While not convergent

Evaluate s, R and T according to the pairs $\{m_i, n_i\}_{i=1}^{N}$

Apply s, R and T to $\{n_j\}$ to get $\{n_j^{'}\}$

Let $\{n_j\} = \{n_j^{'}\}$

Re-calculate the point correspondences $\{m_i, n_i\}_{i=1}^{N}$

re-calculate the error: $\Sigma^2 = \sum_{i=1}^{N}(m_i - n_i)^2$

End

Return s, R, T

Iterative closest point (ICP) algorithms

$[s,R,T]$ = ICP ($\{m_i\}_{i=1}^{N}$, $\{n_j\}_{j=1}^{M}$) (suppose $N<M$)

calculate the point correspondences $\{m_i, n_i\}_{i=1}^{N}$ (closest point)

calculate the error: $\Sigma^2 = \sum_{i=1}^{N} (m_i - n_i)^2$

While not convergent

Evaluate s, R and T according to the pairs $\{m_i, n_i\}_{i=1}^{N}$ How?

Apply s, R and T to $\{n_j\}$ to get $\{n_j^{'}\}$

Let $\{n_j\} = \{n_j^{'}\}$

Re-calculate the point correspondences $\{m_i, n_i\}_{i=1}^{N}$

re-calculate the error: $\Sigma^2 = \sum_{i=1}^{N} (m_i - n_i)^2$

End

Return s, R, T

Iterative closest point (ICP) algorithms

Problem definition:

Given a set of point correspondence pairs $\{m_i, n_i\}_{i=1}^{N}$, how to evaluate s, R and T to minimize

$$\Sigma^2 = \sum_{i=1}^{N} \left\| m_i - \left(sR(n_i) + T \right) \right\|^2$$

Iterative closest point (ICP) algorithms

We assume that there is a similarity transform between point sets $\{m_i\}_{i=1}^{N}$ *and* $\{n_i\}_{i=1}^{N}$

Find s, R and T to minimize

Note: *R* is an orthogonal matrix.

$$\Sigma^2 = \sum_{i=1}^{N} e_i^2 = \sum_{i=1}^{N} \left\| m_i - \left(sR(n_i) + T \right) \right\|^2 \quad (1)$$

Let

$$\overline{m} = \frac{1}{N} \sum_{i=1}^{N} m_i, \overline{n} = \frac{1}{N} \sum_{i=1}^{N} n_i, m_i^{'} = m_i - \overline{m}, n_i^{'} = n_i - \overline{n}$$

Note that: $\sum_{i=1}^{N} m_i^{'} = \mathbf{0}, \sum_{i=1}^{N} n_i^{'} = \mathbf{0}$

Iterative closest point (ICP) algorithms

Then:

$$e_i = m_i - sR(n_i) - T = m_i^{'} + \overline{m} - sR(n_i^{'} + \overline{n}) - T = m_i^{'} + \overline{m} - sR(n_i^{'}) - sR(\overline{n}) - T$$

$$= m_i^{'} - sR(n_i^{'}) - \left(T - \overline{m} + sR(\overline{n})\right) = m_i^{'} - sR(n_i^{'}) - e_0$$

$e_0 = T - \overline{m} + sR(\overline{n})$ is independent from $\{m_i^{'}, n_i^{'}\}$

(1) can be rewritten as:

$$\Sigma^2 = \sum_{i=1}^{N} e_i^2 = \sum_{i=1}^{N} \left\| m_i^{'} - sR(n_i^{'}) - e_0 \right\|^2 = \sum_{i=1}^{N} \left\| m_i^{'} - sR(n_i^{'}) \right\|^2 - 2e_0 \cdot \sum_{i=1}^{N} \left(m_i^{'} - sR(n_i^{'}) \right) + Ne_0^2$$

$$= \sum_{i=1}^{N} \left\| m_i^{'} - sR(n_i^{'}) \right\|^2 - 2e_0 \cdot \sum_{i=1}^{N} \left(m_i^{'} \right) + 2e_0 \cdot \sum_{i=1}^{N} \left(sR(n_i^{'}) \right) + Ne_0^2$$

$$= \sum_{i=1}^{N} \left\| m_i^{'} - sR(n_i^{'}) \right\|^2 + Ne_0^2$$

Variables are separated and can be minimized separately.

$$e_0^2 = 0 \Leftrightarrow T = \overline{m} - sR(\overline{n})$$

If we have s and R, T can be determined.

Iterative closest point (ICP) algorithms

Then the problem simplifies to: how to minimize

$$\Sigma^2 = \sum_{i=1}^{N} \left\| m_i^{'} - sR(n_i^{'}) \right\|^2$$

Consider its geometric meaning here.

We revise the error item as a symmetrical one:

$$\Sigma^2 = \sum_{i=1}^{N} \left\| \frac{1}{\sqrt{s}} m_i^{'} - \sqrt{s} R(n_i^{'}) \right\|^2 = \frac{1}{s} \sum_{i=1}^{N} \left\| m_i^{'} \right\|^2 - 2 \sum_{i=1}^{N} m_i^{'} \cdot R(n_i^{'}) + s \sum_{i=1}^{N} \left\| R(n_i^{'}) \right\|^2$$

$$= \frac{1}{s} \sum_{i=1}^{N} \left\| m_i^{'} \right\|^2 - 2 \sum_{i=1}^{N} m_i^{'} \cdot R(n_i^{'}) + s \sum_{i=1}^{N} \left\| n_i^{'} \right\|^2$$

P D Q

Variables are separated.

$$\Sigma^2 = \frac{1}{s} P - 2D + sQ = \left(\sqrt{s}\sqrt{Q} - \frac{1}{\sqrt{s}} \sqrt{P} \right)^2 + 2(\sqrt{PQ} - D)$$

Thus,

Iterative closest point (ICP) algorithms

$$\left(\sqrt{s}\sqrt{Q}-\frac{1}{\sqrt{s}}\sqrt{P}\right)^2=0\Leftrightarrow s=\sqrt{\frac{P}{Q}}=\sqrt{\frac{\sum_{i=1}^{N}\left\|m_i^{'}\right\|^2}{\sum_{i=1}^{N}\left\|n_i^{'}\right\|^2}}$$

Determined!.

Then the problem simplifies to: how to maximize

$$D=\sum_{i=1}^{N}m_i^{'}\cdot R(n_i^{'})$$

Note that: D is a real number.

$$D=\sum_{i=1}^{N}m_i^{'}\cdot Rn_i^{'}=\sum_{i=1}^{N}\left(m_i^{'}\right)^T Rn_i^{'}=trace\left(\sum_{i=1}^{N}Rn_i^{'}\left(m_i^{'}\right)^T\right)=trace\left(RH\right)$$

$$H\equiv\sum_{i=1}^{N}n_i^{'}\left(m_i^{'}\right)^T$$

Now we are looking for an orthogonal matrix *R* to maximize the trace of *RH*.

Iterative closest point (ICP) algorithms

Lemma

For any positive semi-definite matrix C and any orthogonal matrix *B*:

$$trace(C) \geq trace(BC)$$

Proof:

From the positive definite property of *C*, $\exists A, C = AA^T$

where A is a non-singular matrix.

Let a_i be the *i*th column of A. Then

$$trace(BAA^T) = trace(A^T BA) = \sum_i a_i^T (Ba_i)$$

According to Schwarz inequality: $|< x, y >| \leq \|x\| \|y\|$

$$a_i^T (Ba_i) \leq \|a_i^T\| \|Ba_i\| = \sqrt{(a_i^T a_i)(a_i^T B^T Ba_i)} = a_i^T a_i$$

Hence,

$trace(BAA^T) \leq \sum_i a_i^T a_i = trace(AA^T)$ that is, $trace(BC) \leq trace(C)$

Iterative closest point (ICP) algorithms

Consider the SVD of $H \equiv \sum_{i=1}^{N} n_i^{'} \left(m_i^{'} \right)^T \quad H = U \Lambda V^T$

According to the property of SVD, U and V are orthogonal matrices, and Λ is a diagonal matrix with nonnegative elements.

Now let $X = VU^T$

Note that: X is orthogonal.

We have $XH = VU^T U \Lambda V^T = V \Lambda V^T$ which is positive semi-definite.

Thus, from the lemma, we know: for any orthogonal matrix *B*

$$trace(XH) \geq trace(BXH)$$

for any orthogonal matrix Ψ

$$trace(XH) \geq trace(\Psi H)$$

It's time to go back to our objective now…

R should be X

Iterative closest point (ICP) algorithms

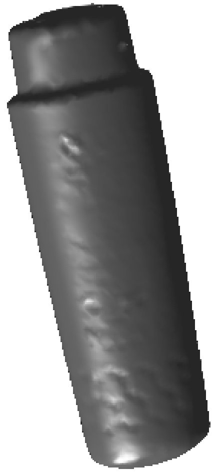
Now, *s*, *R* and *T* are all determined.

$$H \equiv \sum_{i=1}^{N} n_i^{'} \left(m_i^{'} \right)^T = U \Lambda V^T$$

$$R = VU^T \qquad s = \sqrt{\frac{\sum_{i=1}^{N} \left\| m_i^{'} \right\|^2}{\sum_{i=1}^{N} \left\| n_i^{'} \right\|^2}} \qquad T = \overline{m} - sR\left(\overline{n}\right)$$

ICP Matching—An Example


bottle1


bottle2


ac1


ac2

bottle1~bottle2: 0.8131

ac1~ac2: 0.8939

bottle1~ac1: 9.8462

bottle1~ac2: 10.3231

bottle2~ac1: 7.9172

bottle2~ac2: 10.3362


The End