Dimensionality Reduction

☺☺☺☺☺☺

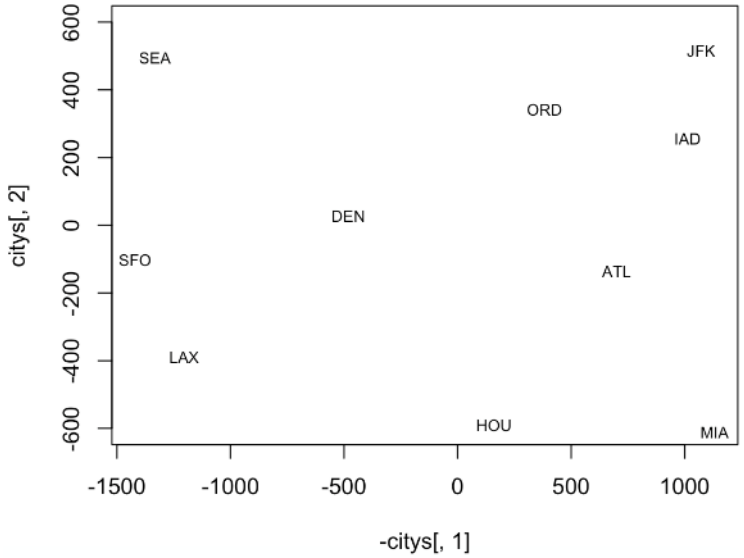
Multidimensional scaling

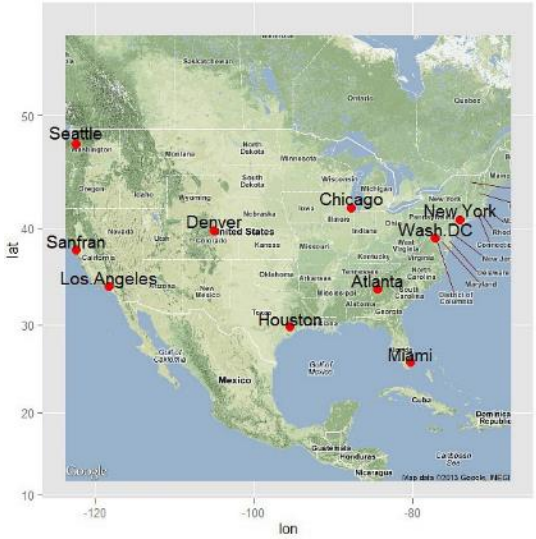
☺

Multidimensional Scaling， MDS

- MDS is a data analysis method which simplifies the research objects of multidimensional space to low-dimensional space for positioning, analysis and classification, while retaining the original relationship between objects.
- Applications: market research, stock market analysis, data compression……

	A	B	C	D	E	F	G	H	I	J	K
1		ATL	ORD	DEN	HOU	LAX	MIA	JFK	SFO	SEA	IAD
2	ATL	0	587	1212	701	1936	604	748	2139	2182	543
3	ORD	587	0	920	940	1745	1188	713	1858	1737	597
4	DEN	1212	920	0	879	831	1726	1631	949	1021	1494
5	HOU	701	940	879	0	1374	968	1420	1645	1891	1220
6	LAX	1936	1745	831	1374	0	2339	2451	347	959	2300
7	MIA	604	1188	1726	968	2339	0	1092	2594	2734	923
8	JFK	748	713	1631	1420	2451	1092	0	2571	2408	205
9	SFO	2139	1858	949	1645	347	2594	2571	0	678	2442
10	SEA	2182	1737	1021	1891	959	2734	2408	678	0	2329
11	IAD	543	597	1494	1220	2300	923	205	2442	2329	0

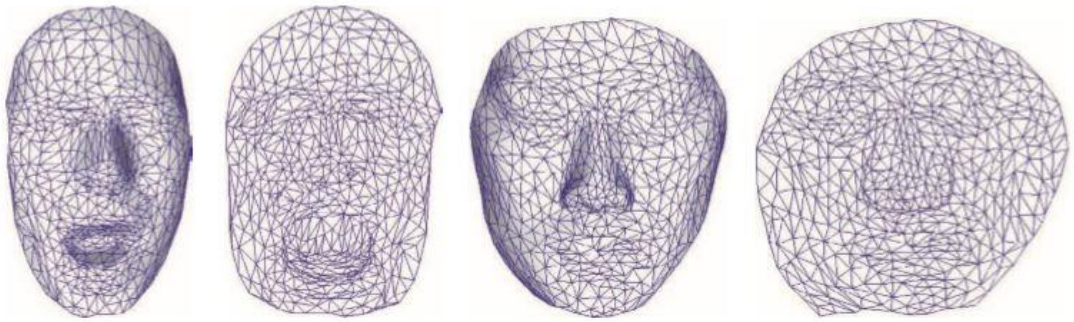




Processing of higher-dimensional surfaces

- A two dimensional surface in three dimensional space is transformed into a two dimensional plane, and the Euclidean distance between the planar vertices is approximately the same as that of the surface's geodesic.

- Convert a matrix $A_{3\times n}$ to $A'_{2\times n}$, hope $\forall i,j \in [1,N], d\left(A_{-i},A_{-j}\right) \approx \left(A'_{-i} - A'_{-j}\right)^2$, where $d(a,b)$ is the geodesic distance between points a and b on the original surface.



Original. 2D MDS. Original. 2D MDS.

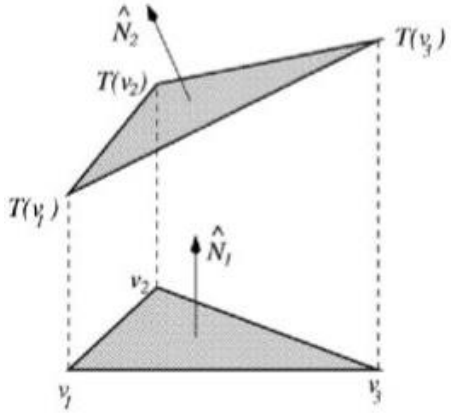
Concrete implementation

- First, we need to calculate the geodesic distance between two selected points of a given surface, which is a conditional extremum problem.

- Find a curve L under $G(x, y, z) = 0$: $L = \int_{x_0}^{x_1} \sqrt{1 + y'^2 + z'^2} dx$ that minimizes the equation $\Gamma = \begin{cases} y = y(x) \\ z = z(x) \end{cases}, (x_0 \leq x \leq x_1)$

- Fast marching algorithm similar to Dijkstra

- The Fast marching algorithm calculates the distance $T(b)$ from point b to origin a :
- Init: add a to the determined set V(initially empty), $T(a) = 0$, add the nearest point a_1 to V, and $T(a_1)$ is the length of the edge (a, a_1).
- Step 1: select the point v_3 that forms a triangle with two points in V and v_3 is not in V, calculate its $T(v_3)$ as shown in the right figure, and add all this v_3 to the minimum heap to be selected (update the T value if v_3 already exists).
- Step 2: if the heap is empty, the algorithm ends; otherwise, transfer the heap top v element to V, and $T(v)$ is the record T in the heap.
- Step 3： go to Step 1.



Concrete implementation

- Now we have the matrix E: $E_{ij} = d_{ij}^2$ composed of the geodesic distance $d_{ij} = d(x_i, x_j)$ between points. Then turn the question into:
- $X = \{x_1, x_2, \dots, x_n\}$ is an $n \times q$ matrix, n is the sample number, q is the data dimension, where each x_i is a row of the matrix X. But we only know the distance between two data, in terms of the matrix E

$$E_{ij} = d_{ij}^2 = \sum_{a=1}^{m} \left(x_{ia} - x_{ja}\right)^2, E = c1' + 1c' - 2XX', c_i = \sum_{a=1}^{m} x_{ia}^2$$

- Find the method of dimensionality reduction to minimize the change of each distance.

PCA & Kernel PCA

☺☺

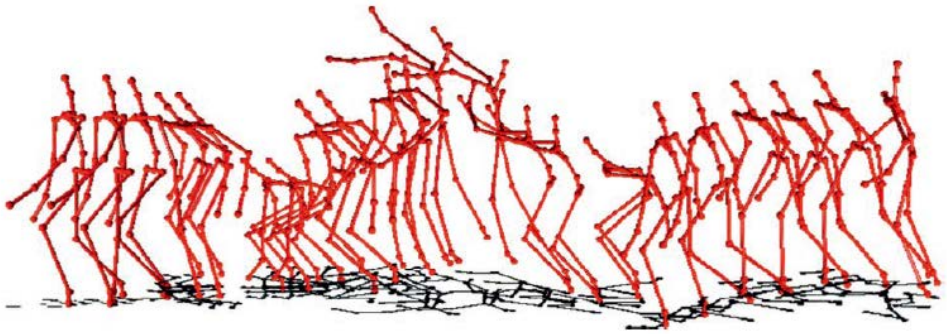
Linear dimensionality reduction - PCA

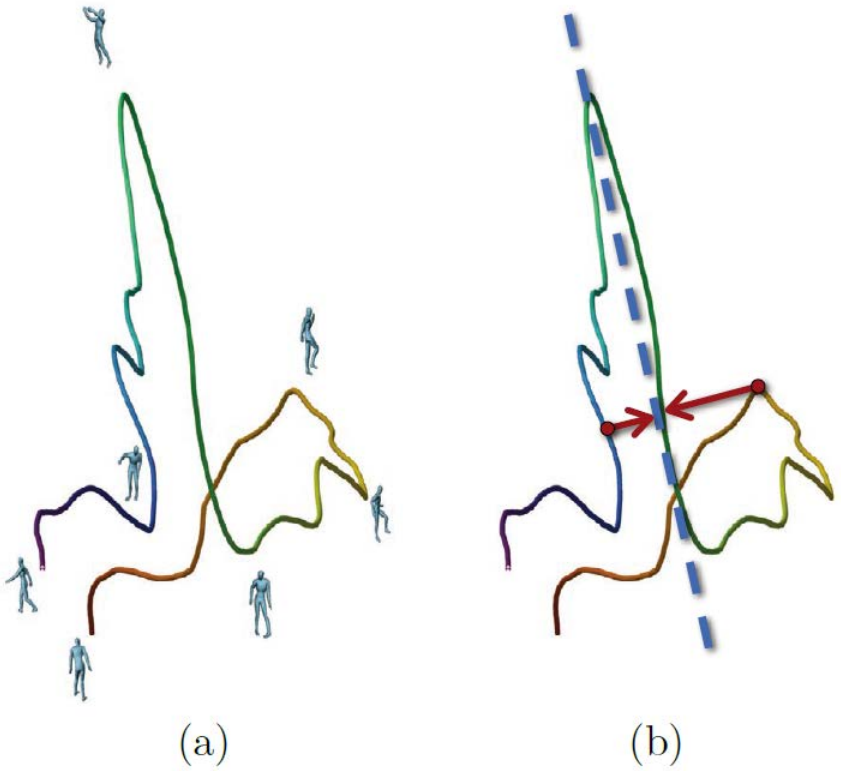
- $\overrightarrow{x_1}, \dots, \overrightarrow{x_n} \in \mathbb{R}^d, \;\; \overrightarrow{m} = \frac{1}{n}\sum_{i=1}^{n} \overrightarrow{x_i}, \;\; \overrightarrow{y_i} = \overrightarrow{x_i} - \overrightarrow{m}$
- $Y = \begin{bmatrix} \overrightarrow{y_1^T} \\ \vdots \\ \overrightarrow{y_n^T} \end{bmatrix}, \;\; S = Y^T Y = V \Lambda V^T$
- $\lambda_1 \geq \cdots \geq \lambda_d \; ; \; \overrightarrow{v_1}, \dots, \overrightarrow{v_d}$
- $\overrightarrow{x_i} = \overrightarrow{m} + \alpha_{i,1}\overrightarrow{v_1} + \cdots + \alpha_{i,d}\overrightarrow{v_d} \;\rightarrow \overrightarrow{x_i} = \overrightarrow{m} + \alpha_{i,1}\overrightarrow{v_1} + \cdots + \alpha_{i,d'}\overrightarrow{v_{d'}}$

Can we reduce the amount of calculation?

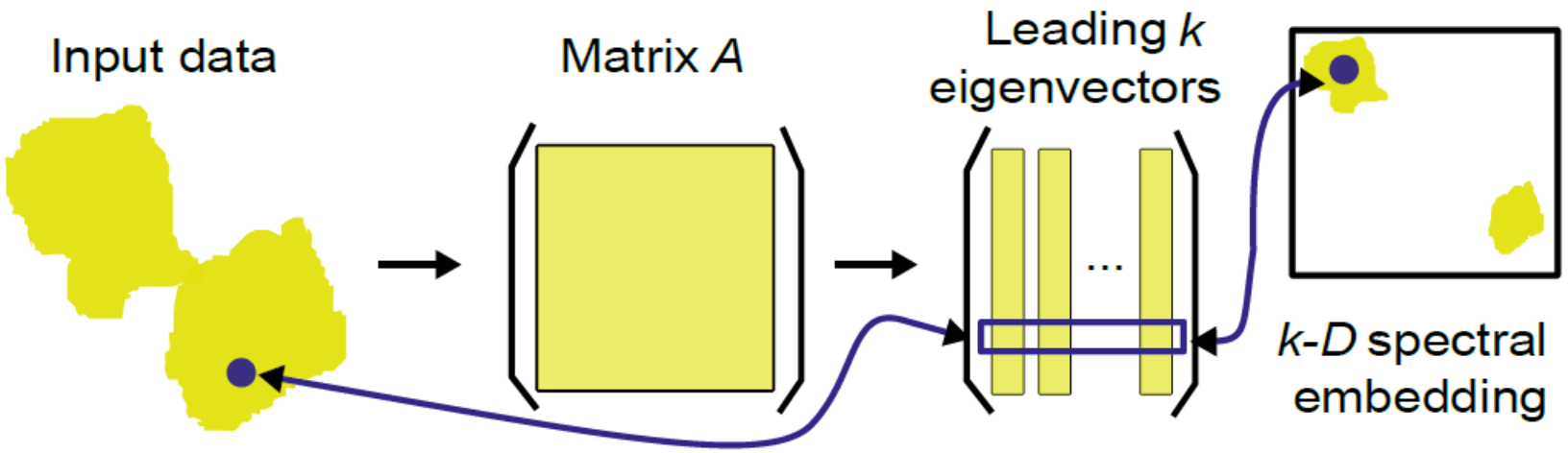
- If the dimension is too high, we may face trouble decomposing $S = Y^T Y = V \Lambda V^T \cdots$
- And if we have $d > n$, then we can apply a trick$\cdots$
- And this trick is really important$\cdots$

Limitations of PCA – Linearity





Non-linear methods



Kernel PCA – kernel function

- Kernel: $\Phi: \mathbb{R}^d \to \mathbb{R}^{\hat{d}}, x_i \longmapsto \Phi(x_i)$
- Φ does not need to be explicitly computed, only the dot product $\langle \Phi(x_i), \Phi(x_j) \rangle$ is needed and applied.
- $K_{n\times n} \coloneqq K_{ij} = \mathrm{K}(x_i, x_j)(1 \leq i, j \leq n)$

Kernel PCA – center the data

- $X_\Phi := \begin{bmatrix} \overrightarrow{\Phi(x_1)^T} \\ \vdots \\ \overrightarrow{\Phi(x_n)^T} \end{bmatrix}_{n \times \hat{\mathrm{d}}}$
- Yet, PCA operates on centered data.
- So an operator comes to rescue: $J = I_{n \times n} - 1_n$
- $Y_\Phi = JX_\Phi$ is centered.☺

Kernel PCA – apply the trick

- $Y_\Phi = JX_\Phi \Longrightarrow \bar{K} = K - 1_n K - K 1_n + 1_n K 1_n$
- So now we have $\bar{K} = Y_\Phi Y_\Phi^T$, but we need the eigenvalues and eigenvectors of $S = Y_\Phi^T Y_\Phi$.
- And the trick we emphasized before can be applied.
- ☺☺☺

Some useful kernels

- $\mathrm{K}(x_i, x_j) =$

- $(x_i \cdot x_j + 1)^p$ polynomial kernel
- $e^{-\frac{1}{2\sigma^2}(x_i - x_j)^2}$ Gaussian kernel
- $e^{-\gamma(x_i - x_j)^2}$ RBF kernel
- $\tanh(\eta x_i \cdot x_j + v)$ sigmoid kernel

Are these kernels really inner product?

$$e^{-\frac{1}{2\sigma^2}(x_i - x_j)^2} = e^{\frac{-x_i^2 - x_j^2}{2\sigma^2}} \sum_{k \geq 0} \frac{(2x_i \cdot x_j)^k}{k!}$$

$$= \phi(x_i)^T \phi(x_j)$$

Where, $\phi(x) = e^{-\frac{x^2}{2\sigma^2}}(1, \sqrt{\frac{2}{1!}}x, \sqrt{\frac{2^2}{2!}}x^2, \ldots)$

RBF kernel

$$\exp\left(-\frac{1}{2}\|\mathbf{x}-\mathbf{x}'\|^2\right) = \sum_{j=0}^{\infty} \frac{(\mathbf{x}^\top \mathbf{x}')^j}{j!} \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right) \exp\left(-\frac{1}{2}\|\mathbf{x}'\|^2\right)$$

$$= \sum_{j=0}^{\infty} \sum_{\sum n_i = j} \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right) \frac{x_1^{n_1} \cdots x_k^{n_k}}{\sqrt{n_1! \cdots n_k!}} \exp\left(-\frac{1}{2}\|\mathbf{x}'\|^2\right) \frac{{x'}_1^{n_1} \cdots {x'}_k^{n_k}}{\sqrt{n_1! \cdots n_k!}}$$

RBF kernel

- Gaussian kernel is used most commonly
- The effect of the parameter gamma

RBF SVM parameters¶

This example illustrates the effect of the parameters `gamma` and `C` of the Radial Basis Function (RBF) kernel SVM.

Intuitively, the `gamma` parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. The `gamma` parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors.

- from sklearn document

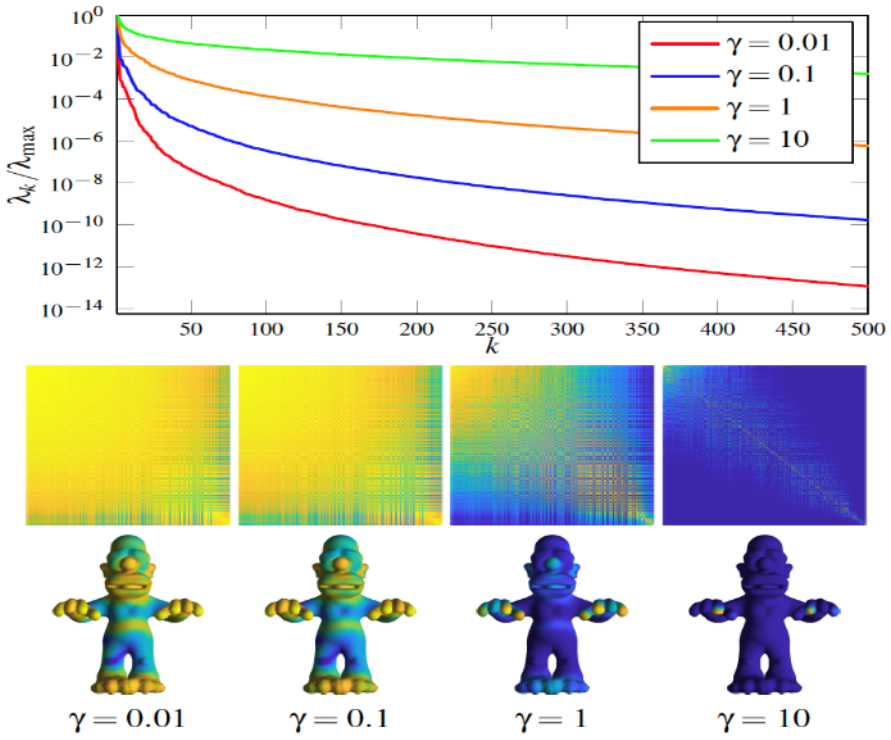
General case

- Theorem (Mercer's theorem, simplified)

- Suppose $\mathrm{K}(x, y) \in L^2(\mathbb{R}^d \times \mathbb{R}^d)$ is a positive definite kernel. Then, there exists a sequence of functions $\{\varphi_i(x)\}_{i=1}^{\infty}$ such that $\mathrm{K}(x, y) = \sum_{i=1}^{\infty} \varphi_i(x)\varphi_i(y)$, with convergence in the L^2 norm.

Kernel function maps

- Map surfaces to functions with kernel trick



Getting back…

☺

- In order to conduct PCA decomposition, it is necessary to mean X to zero. More generally, the new origin is set as the weighted average of samples:

$$\begin{aligned}\mathbf{X}_s &= \mathbf{X} - \mathbf{1}\mathbf{s}' \\ &= (\mathbf{I} - \mathbf{1}\mathbf{w}')\mathbf{X} \\ &= \mathbf{P}_w\mathbf{X}.\end{aligned}$$

$\mathbf{s}' = \mathbf{w}'\mathbf{X}$, where $\sum_{i=1}^{n} w_i = 1$, and, for every i, $w_i \geq 0$.

- When mean zero, take $\mathbf{w} = \frac{1}{n}\mathbf{1}$, $\mathbf{P}_w = \mathbf{J} = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}'$

$$\begin{aligned}\mathbf{B} &= -\frac{1}{2}J(\mathbf{c}\mathbf{1}' + \mathbf{1}\mathbf{c}' - 2XX')J' \\ &= -\frac{1}{2}JcO' - \frac{1}{2}Oc'J' + JXX'J' \\ &= JXX'J\end{aligned}$$

$$JX(JX)' = B = -\frac{1}{2}JEJ$$

- The result of inner product can be calculated by E, that is, Kernel PCA method can be used

- B is the inner product matrix in Kernel PCA, and the aforementioned method can be used for it. Let $\lambda_i$be the ith eigenvalue of B, and $v_i$be the ith row eigenvector, $i.e.$

$$B = [v_1', v_2', \ldots, v_n'] \wedge \begin{bmatrix} v_1 \\ \cdots \\ v_n \end{bmatrix} = Q \wedge Q'$$

Λ is eigenvalue diagonal matrix

- Then the column vectors lined up with the coordinate values of each X vector in

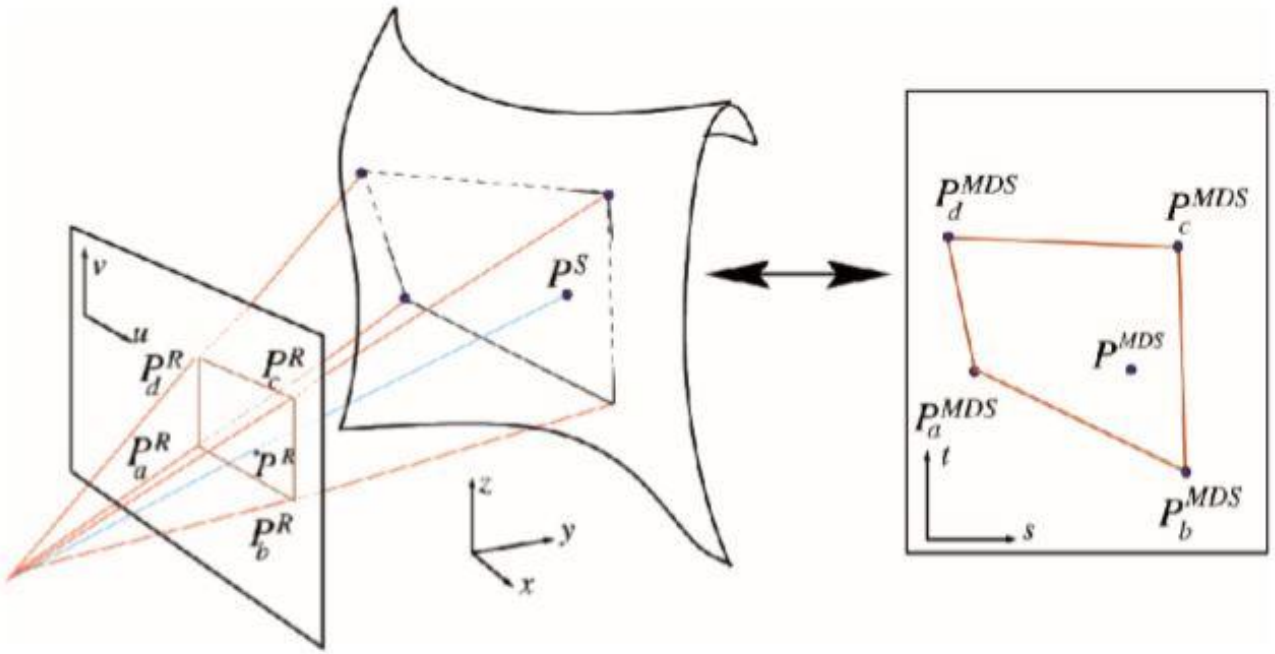
$$\hat{X}_i = Xv_i' = \frac{1}{\sqrt{\lambda_i}} Bv_i' = \frac{1}{\sqrt{\lambda_i}} Q \wedge Q' v_i' = \sqrt{\lambda_i}\, v_i'$$

$(\hat{X}_i)_j$ is the ith coordinate value of the j^{th}data

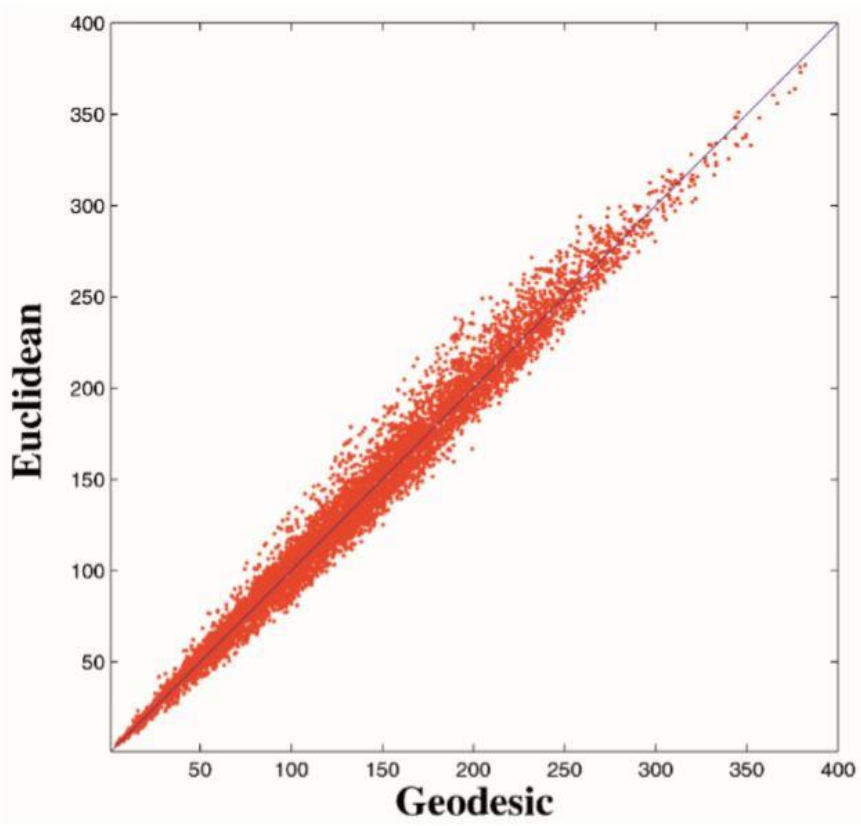
$$\left[\hat{X}_1, \hat{X}_2 \ldots \hat{X}_m\right] = [\sqrt{\lambda_1}\, v_1', \sqrt{\lambda_2}\, v_2' \ldots \sqrt{\lambda_m}\, v_m'] = Q_m \Lambda_m^{\frac{1}{2}}$$

- In specific problems, a small number of data points are generally selected for dimensionality reduction, and the remaining data points are mapped by the difference method

- Interpolation method: the position of (t, s) in the dimension reduction plane is determined according to the proportion of four points in (v, u) plane:



- The result obtained by flattening the face surface in 3d

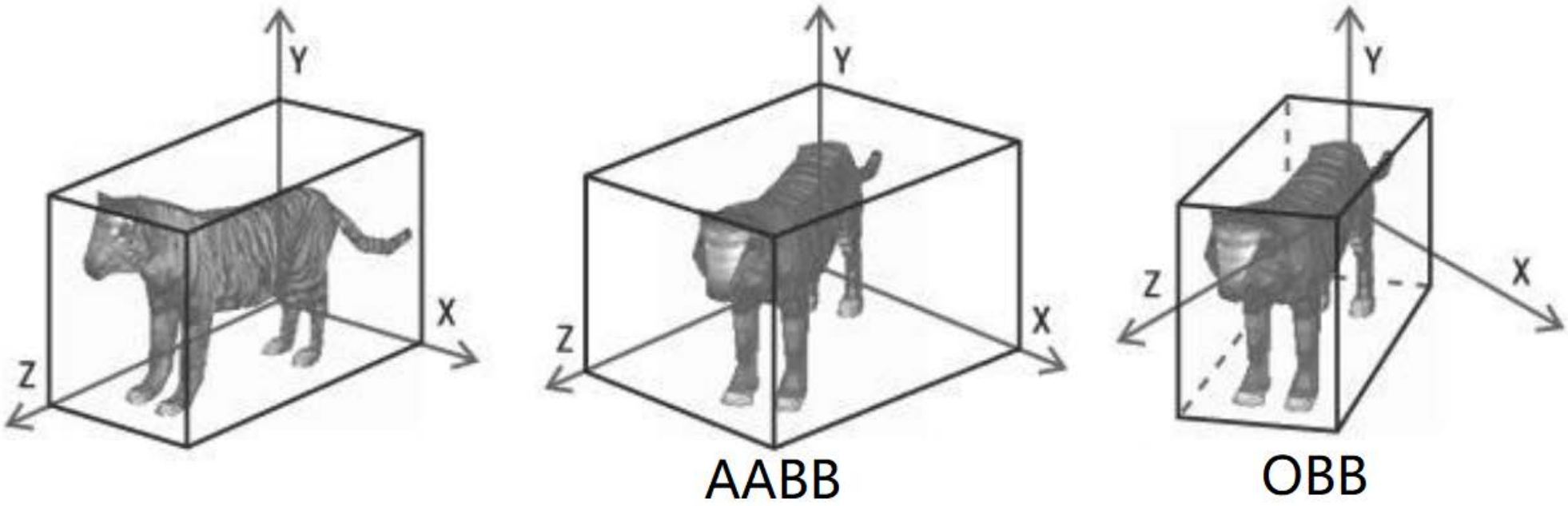


- Texture mapping based on the correspondence between plane and surface

OBB method

☺☺☺

Classification of bounding boxes



The AABB bounding box is aligned with the axis and is easy to generate, but it does not rotate with the object and does not perform well for more accurate collision detection. OBB bounding box is always generates a minimum rectangular bounding box along the principal component direction of the object, which can be rotated with the object for more accurate detection.

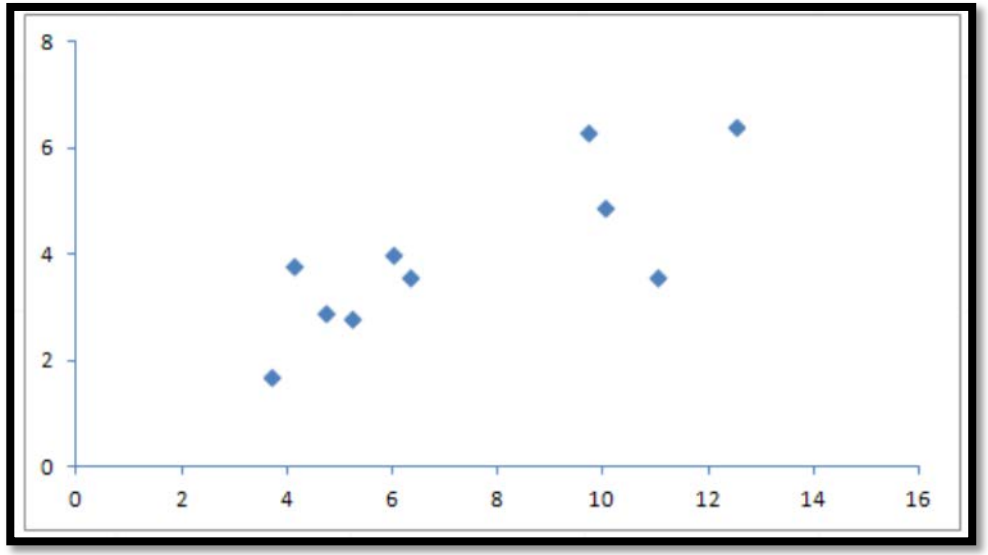
Calculating process

$$cov(X_i, X_j) = E[(X_i - \mu_i)(X_j - \mu_j)]$$

covariance matrix $A = \begin{bmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(x,y) & cov(y,y) & cov(y,z) \\ cov(x,z) & cov(y,z) & cov(z,z) \end{bmatrix}$

- The covariance matrix is a real symmetric matrix, so it can be diagonalized.
- The eigenvectors of the covariance matrix represent the direction of the OBB bounding box.
- The OBB bounding box axes should follow the direction of the eigenvector corresponding to the maximum eigenvalue.

- (3.7, 1.7), (4.1, 3.8), (4.7, 2.9), (5.2, 2.8), (6.0, 4.0), (6.3, 3.6), (9.7, 6.3), (10.0, 4.9), (11.0, 3.6), (12.5, 6.4)



Step 1. Calculate the covariance matrix

$$A = \begin{bmatrix} cov(x,x) & cov(x,y) \\ cov(x,y) & cov(y,y) \end{bmatrix} = \begin{bmatrix} 9.0836 & 3.365 \\ 3.365 & 2.016 \end{bmatrix}$$

Step 2. diagonalize A

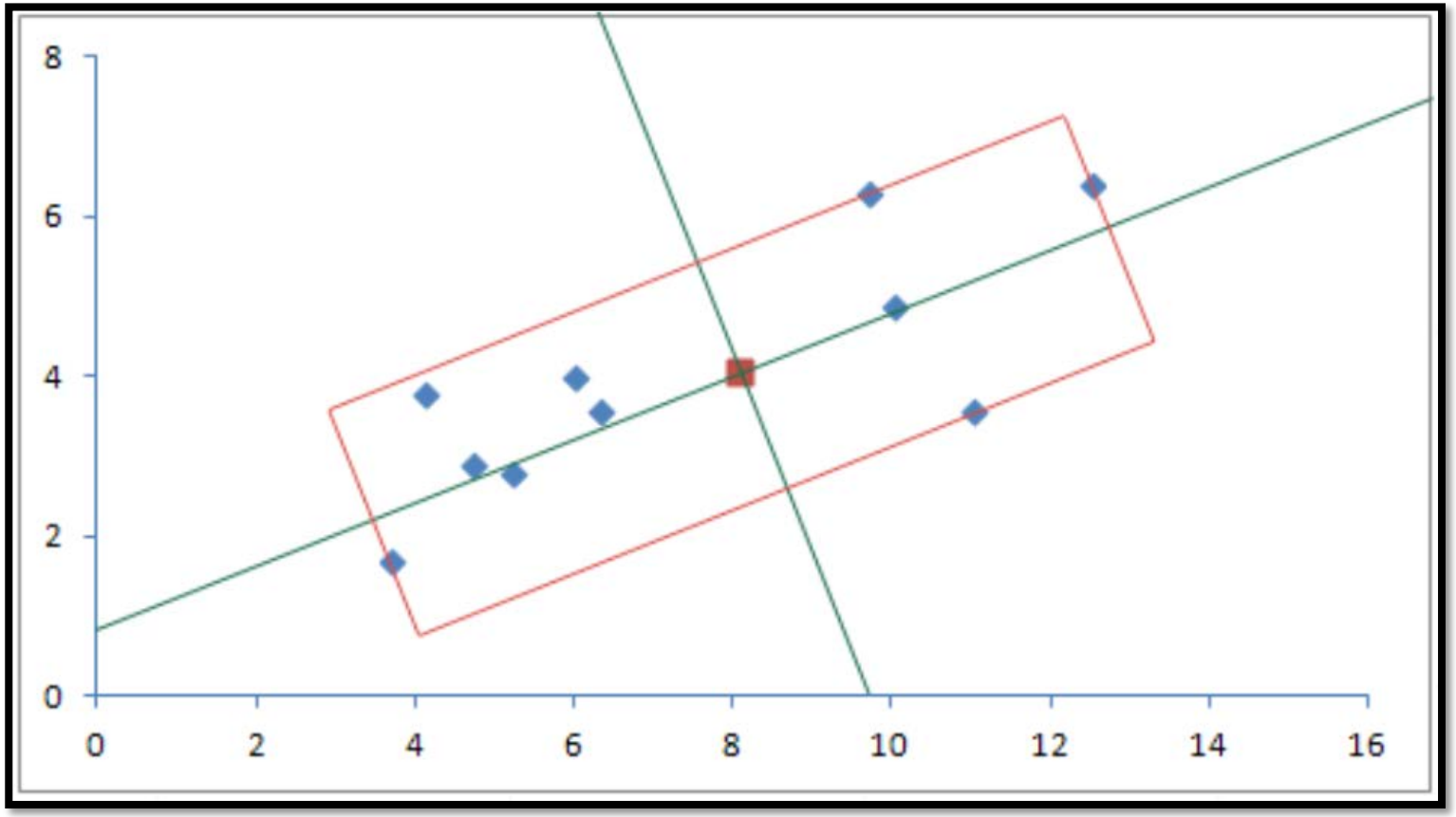
$$A = \begin{bmatrix} 0.928491 & 0.371355 \\ -0.371355 & 0.928491 \end{bmatrix} \begin{bmatrix} 10.4294 & 0.0 \\ 0.0 & 0.670152 \end{bmatrix} \begin{bmatrix} 0.928491 & -0.371355 \\ 0.371355 & 0.928491 \end{bmatrix}$$

Step 3. Take the eigenvector as the direction of OBB

$$\vec{v}_1 = \begin{bmatrix} 0.928491 \\ 0.371355 \end{bmatrix}, \vec{v}_2 = \begin{bmatrix} -0.371355 \\ 0.928491 \end{bmatrix}$$

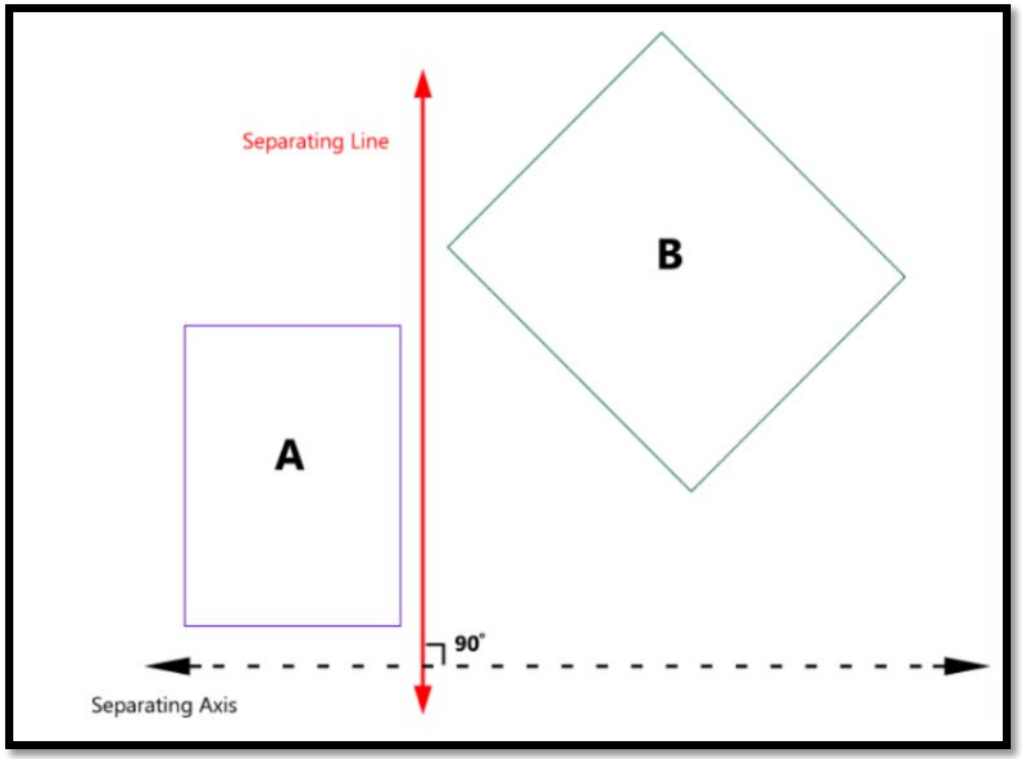
Two directions of the bounding box

$$\vec{v}_1 = \begin{bmatrix} 0.928491 \\ 0.371355 \end{bmatrix}, \vec{v}_2 = \begin{bmatrix} -0.371355 \\ 0.928491 \end{bmatrix}$$



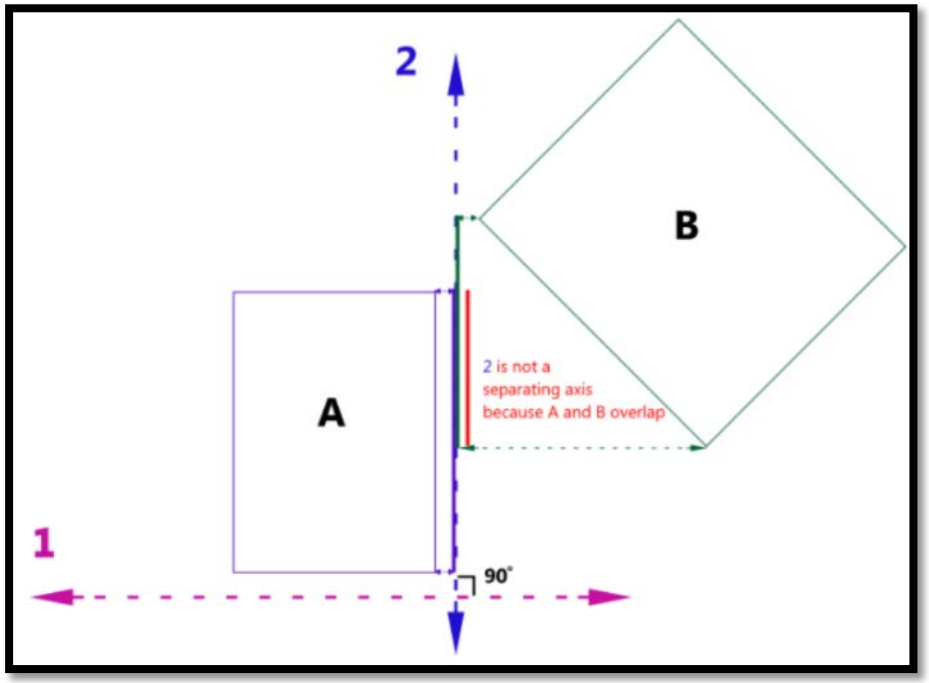
OBB and collision detection

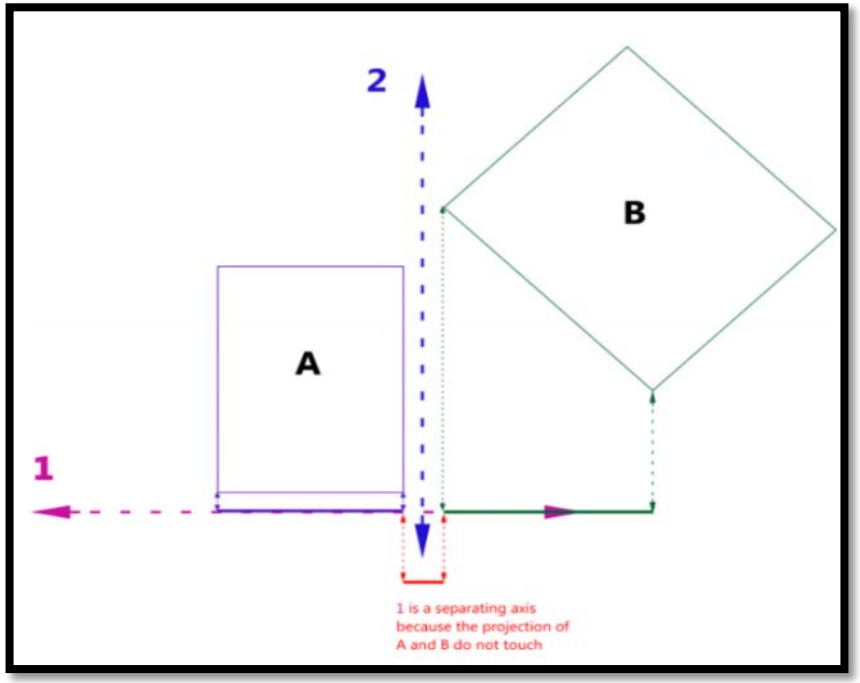
- Two convex hull polygons do not intersect, iif there is a line (Separating Axis) , their projections on that line do not intersect.



OBB and collision detection

- To find the separation axis, try only on the sides of two rectangles (four directions in total).
- In the figure below, 2 is not the split axis and 1 is the split axis

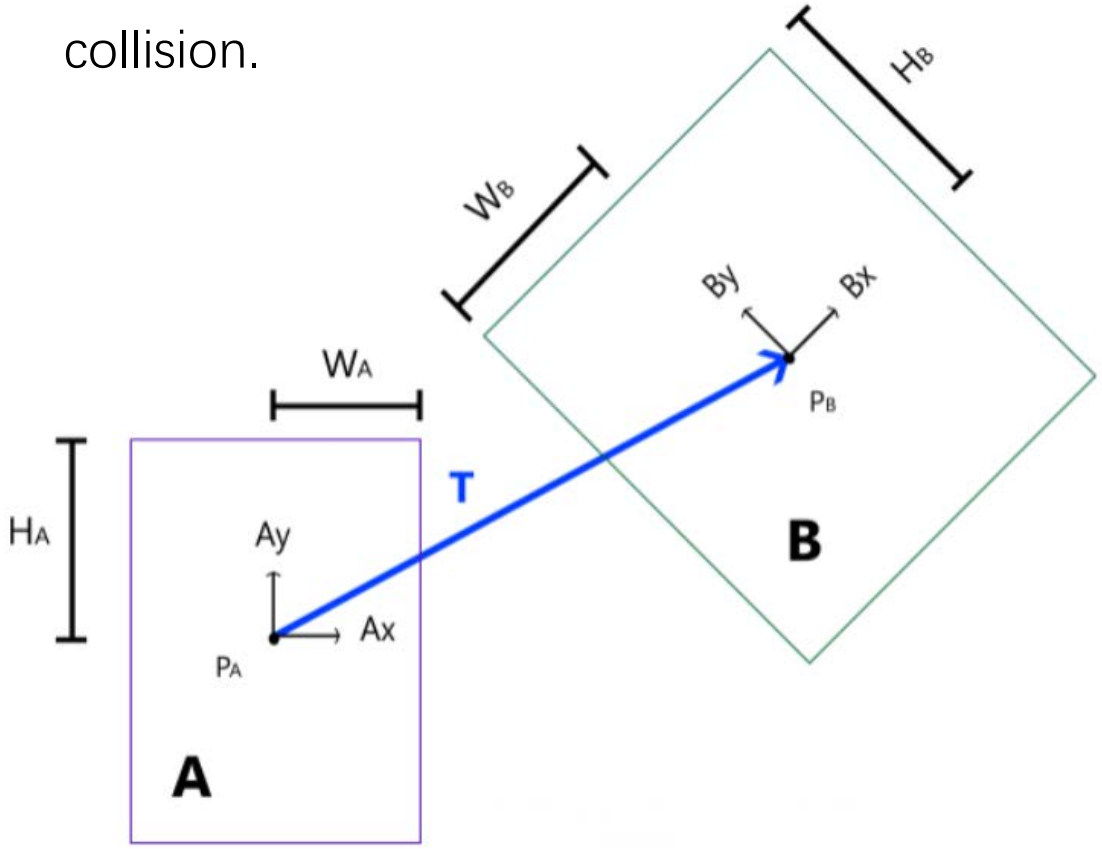


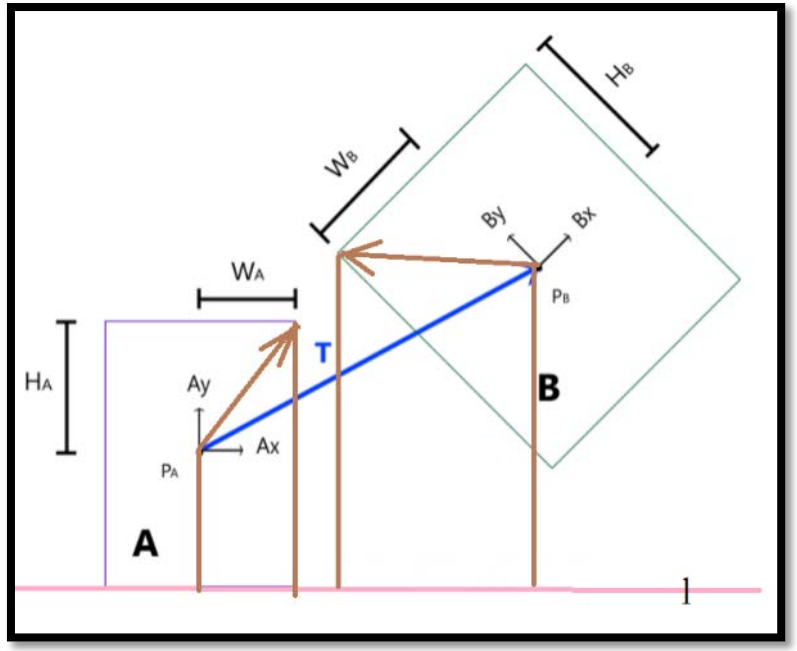


OBB and collision detection

$$|T \bullet L| > |(W_A Ax) \bullet L| + |(H_A Ay) \bullet L| + |(W_B Bx) \bullet L| + |(H_B By) \bullet L|$$

- Try the four values of L, if there is one that satisfies the condition, then there is no collision.





Non-rigid Surfaces Matching

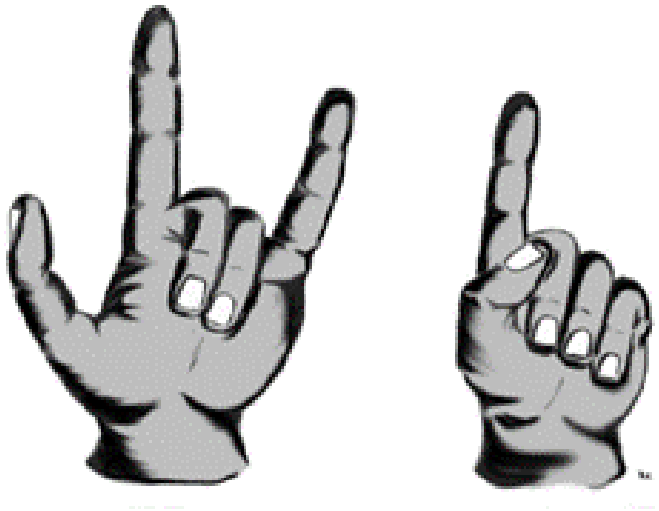
☺☺☺☺

Task Description

- Shape retrieval
- Shape correspondence
- Different poses of a deformable object

Geodesic Distance

- Use geodesic distance to describe the intrinsic structure of the surface.
- Alg. fast marching on triangulated domains: compute the geodesic distance between n points out of N points in $O(nN)$

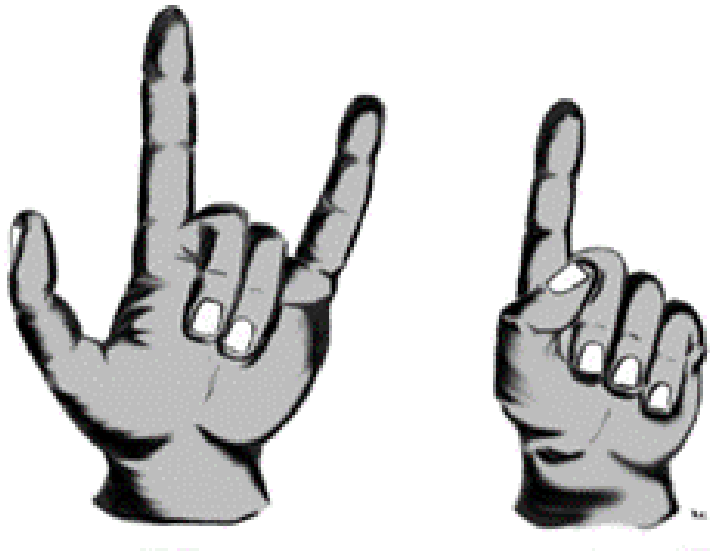


Triangulation

- Triangulate the surface
- Choose a small subset of vertexes (“far” from each other)
- Voronoi sampling
- Obtain an $n \times n$ matrix of geodesic distance

MDS Mapping

- Map the vertexes into an m-dimension Euclid space
- The geodesic distance between two points equals to the Euclid distance in the new space
- Obtain the “signature surface”



Matching

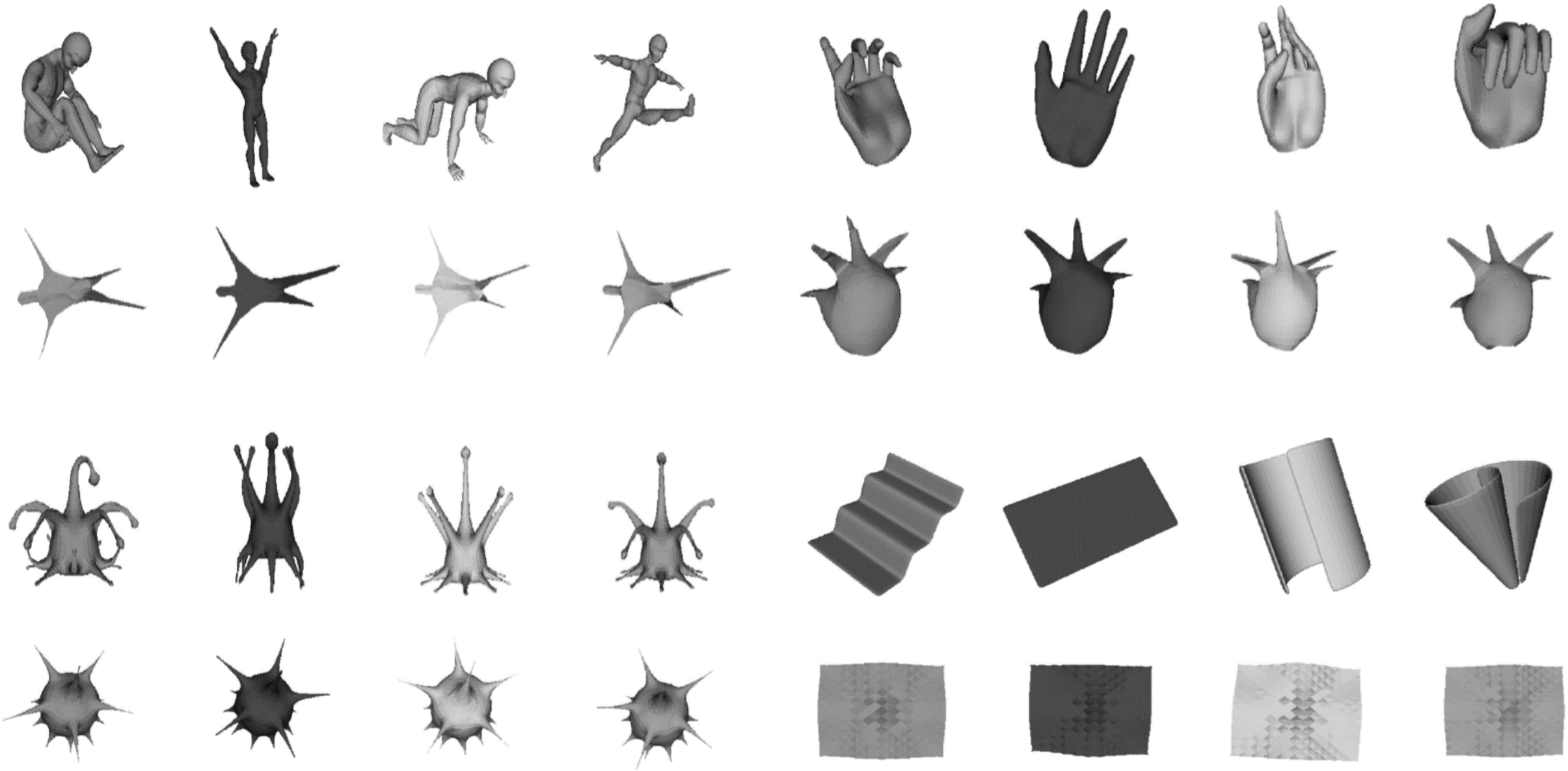
- What we have done:
- matching nonrigid surface $\rightarrow$ matching rigid surface

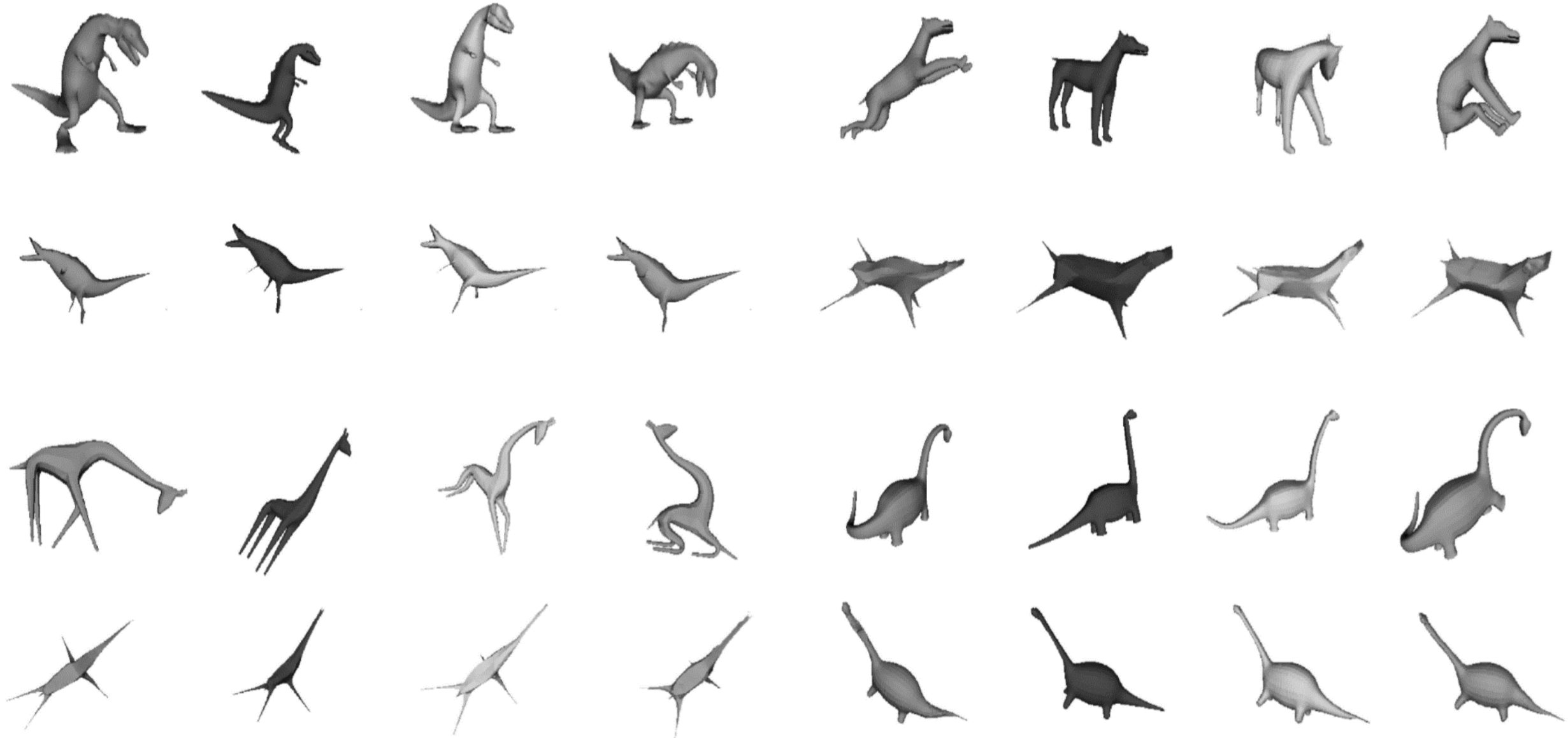
- Let M_i be a vector of the first few moments of the surface S_i, then the moments-distance matrix, D_M, is defined as
- $[D_M]_{ij} = \left|\left|M_i - M_j\right|\right|_2^2$

Matching

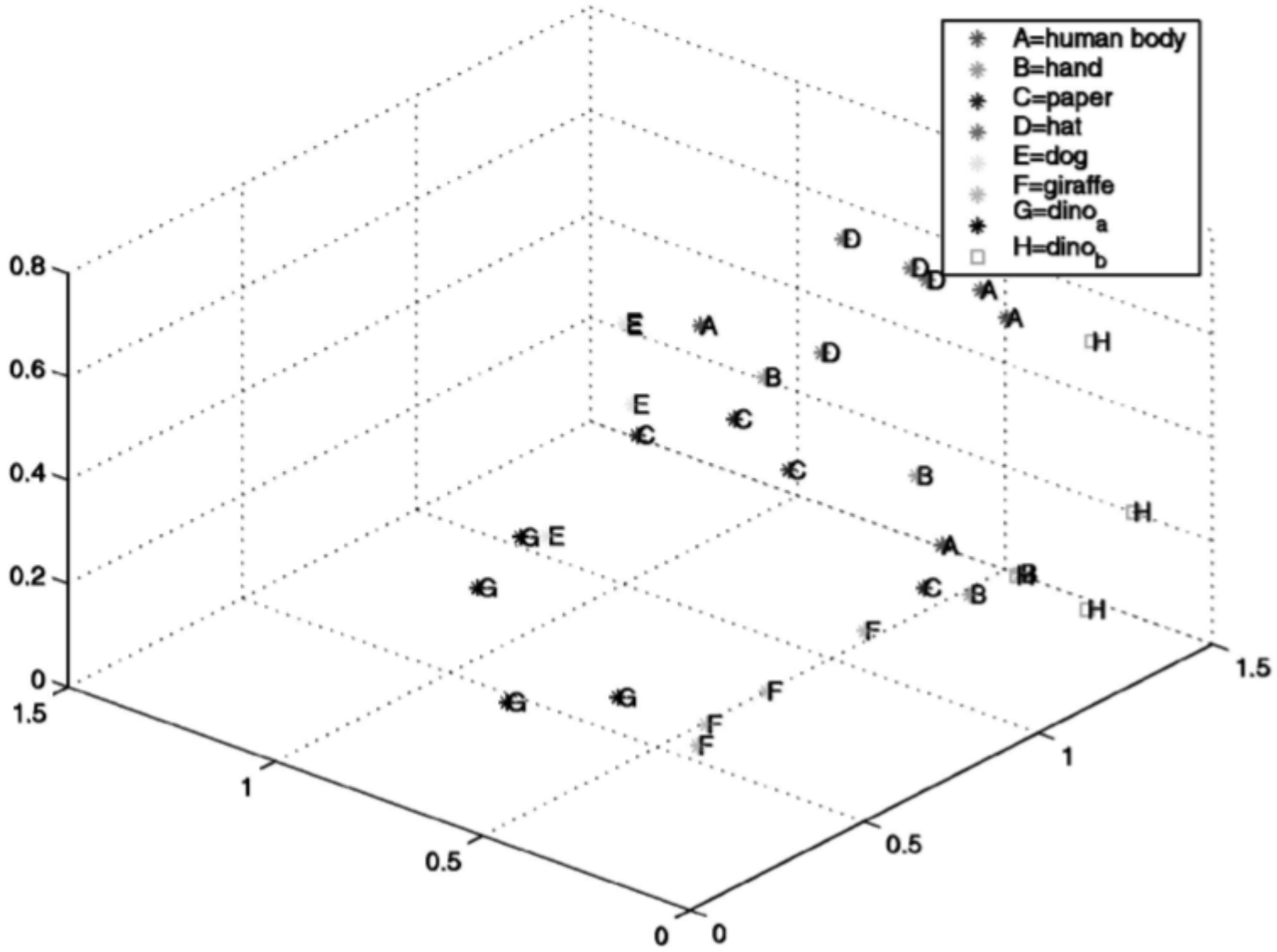
- Now D_M describe the pairwise dissimilarity between surfaces
- Apply the MDS again!
- Each surface $\rightarrow$ a point in 3-dimension Euclid space
- Apply K-means, K-nearest Neighbor

Results

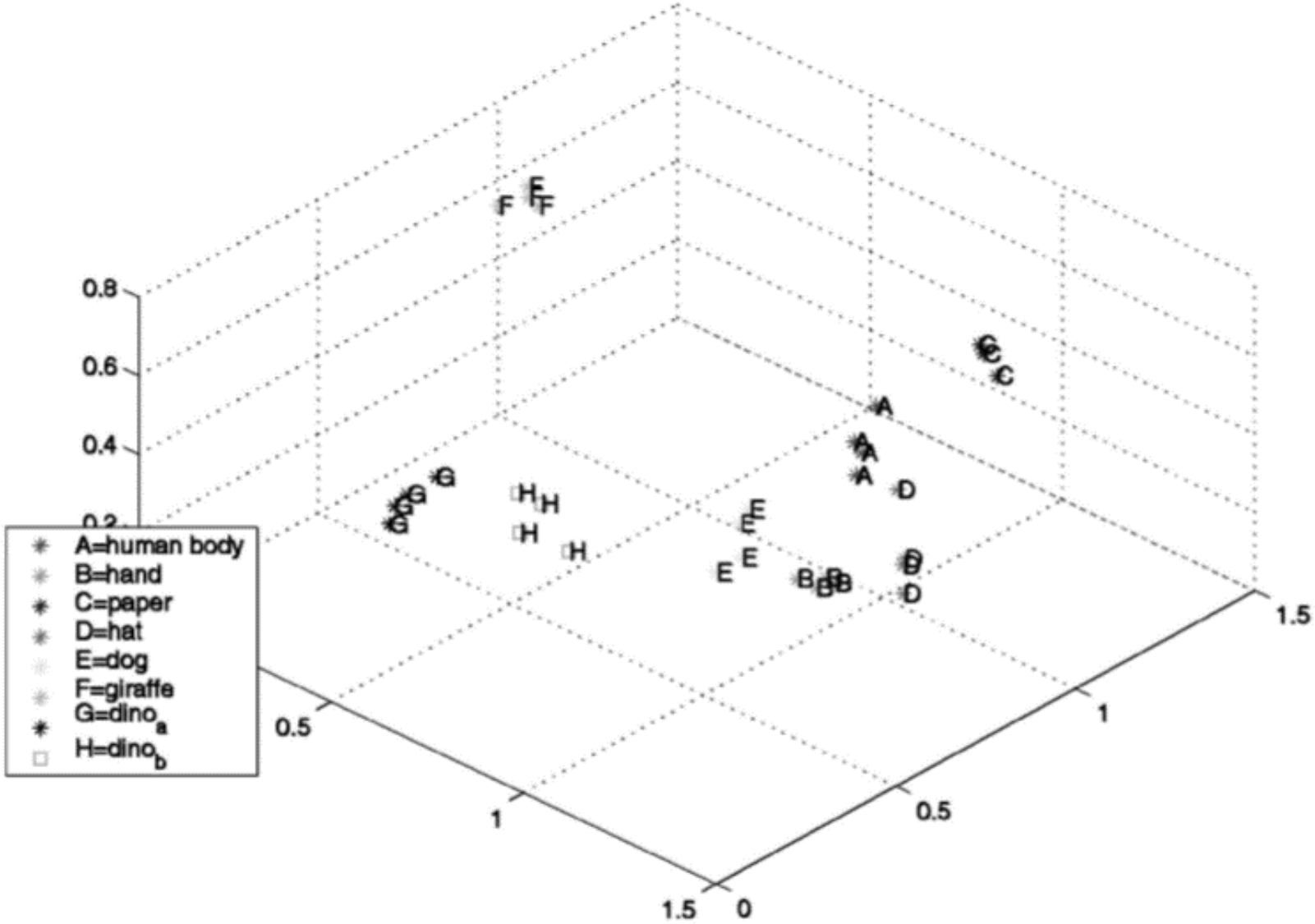




- Apply moment-based clustering on the original surface…

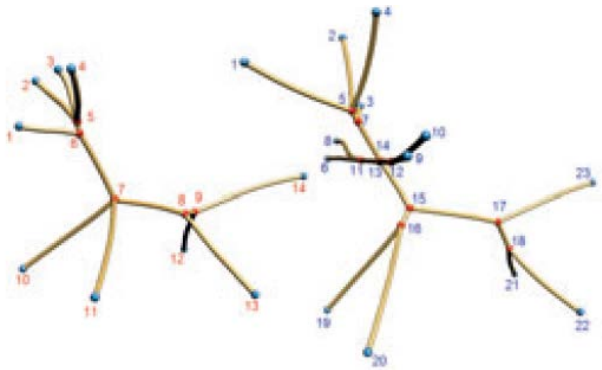


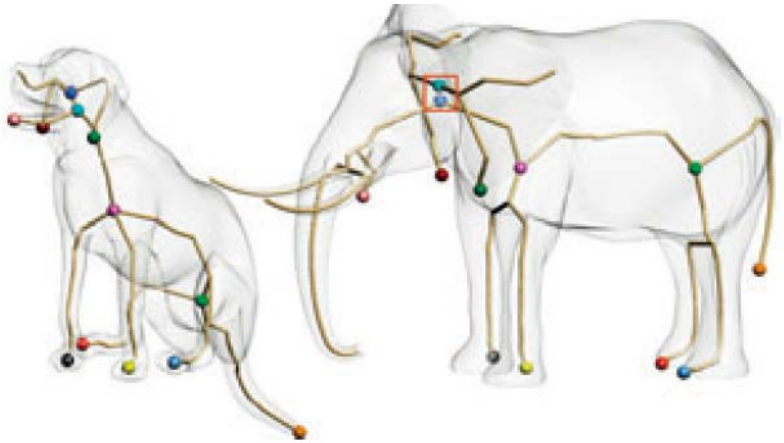
- Apply moment-based clustering on the original surface…



Electors Voting for Fast Automatic Shape Correspondence

☺☺☺☺☺



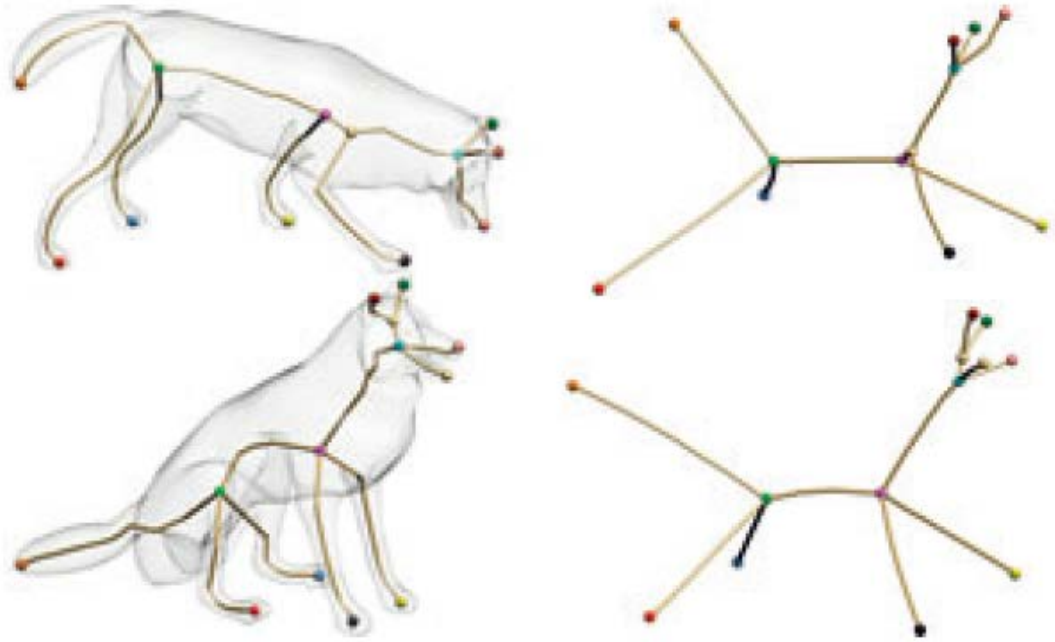


Challenges

- Semantically similar but possibly geometrically very different
- Finding an exact subgraph isomorphism is an NP-complete problem

Electors Selection

- Spatial Configuration
- Alleviate the influence of different poses by performing a spatial embedding using MDS



Electors Selection

- Pruning 1： Node-Centricity $c_p = \frac{1}{|P|} \sum_{p_i \in P} geo(p, p_i)$
  - $geo(p, p_i)$ is the geodesic distance between p and p_i along skeletal paths. Larger values of c_p means p is further away from the center of the shape.
  - Reject (p_k, q_k) if their centricity difference is larger than a threshold

$$\frac{|c_{p_k} - c_{q_k}|}{(c_{p_k} + c_{q_k})/2} > \varepsilon_C$$

Electors Selection

- Pruning 2： Path-Length and Path-Radius
  - We reject (p_k, q_k) if either one of the following is true

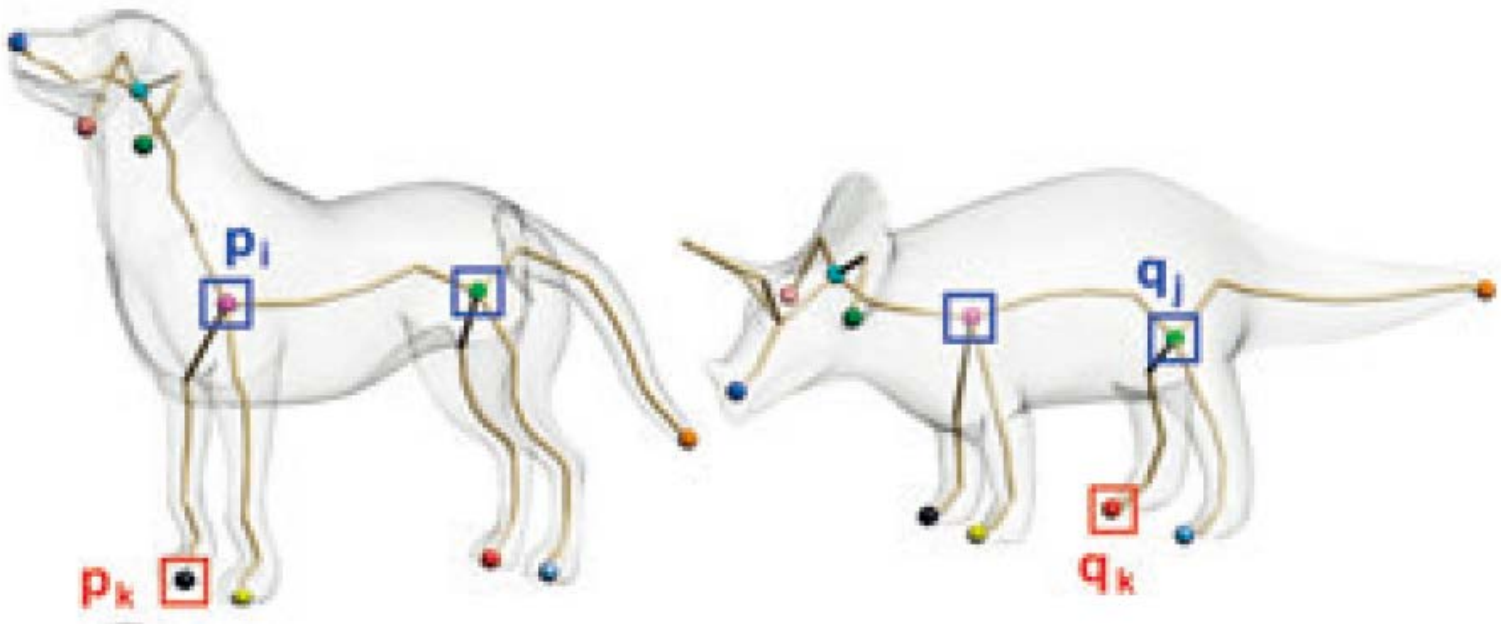
$$\frac{|geo(p_k,p_j)-geo(q_k,q_j)|}{(geo(p_k,p_j)+geo(q_k,q_j))/2} > \varepsilon_C,\ \forall(p_j,q_j)\in\Omega,$$

$$\frac{|rad(p_k,p_j)-rad(q_k,q_j)|}{(rad(p_k,p_j)+rad(q_k,q_j))/2} > \varepsilon_C,\ \forall(p_j,q_j)\in\Omega,$$

  - $geo(\cdot,\ \cdot)$ and $rad(\cdot,\ \cdot)$ are the geodesic distance and the average path radius of two feature nodes along skeletal paths.
  - normalize to range $[0, 1]$ ， size invariant
  - pose invariant

Electors Selection

- Pruning 3: Topology Consistency
  - To maintain similar local topological structures we reject (p_k, q_k).



Voting Process

- s_{ij} is the total occurrences of the pair (p_i, q_j) in all electors
- a higher score has a higher probability of being a correct matched feature pair

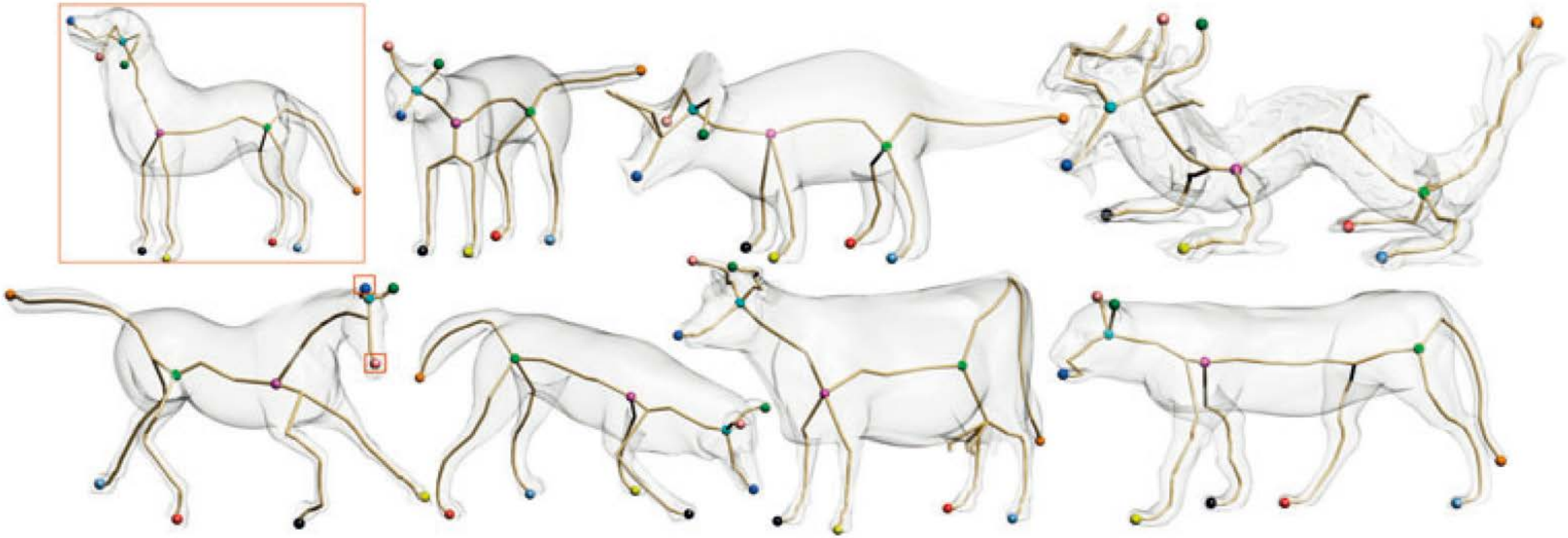
$$s_{ij} = \sum_k \Phi_{i,j,k}, \quad \Phi_{i,j,k} = \begin{cases} 1 & \text{if } (p_i, q_j) \in \Omega_k, \\ 0 & \text{if } (p_i, q_j) \notin \Omega_k. \end{cases}$$

Form the Final Correspondence

- All the feature pairs are first sorted by their scores, and the pair $(p_i, q_j)$with the highest score is iteratively added to Ω^* satisfying the following conditions:

  - ◇ **1-1 mapping**: Neither p_i nor q_j is already in Ω^*;
  - ◇ **Topology consistency**: Same as the T3 pruning test, i.e., the closest junction nodes of p_i and q_j that are already in Ω^* must match.

Results



Thank you!

☺☺☺☺☺☺