HW2 Update

- Extend the deadline to 23:59pm, next Monday (Feb 21)
- Updated rubric:

	ICP (>)	Learning (>)
2pt	Beat baseline	Beat baseline
3pt	40%	20%
4pt	60%	50%
5pt	80%	80%

- Some basic generic tricks for network training are very useful (e.g., data normalization, increasing batch size, …)
- Fanbo will create a post about some generic tricks.

L13: Analysis by Intrinsics

Hao Su

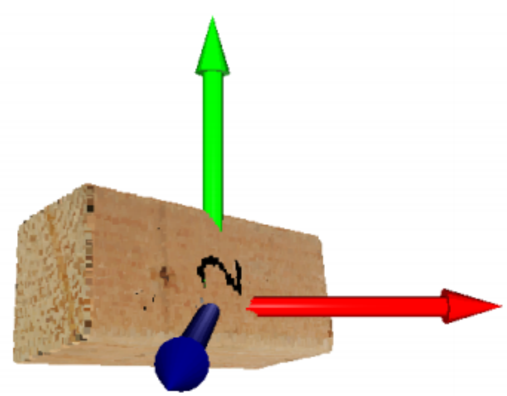
Thank Xiaoshuai Zhang for helping to prepare slides

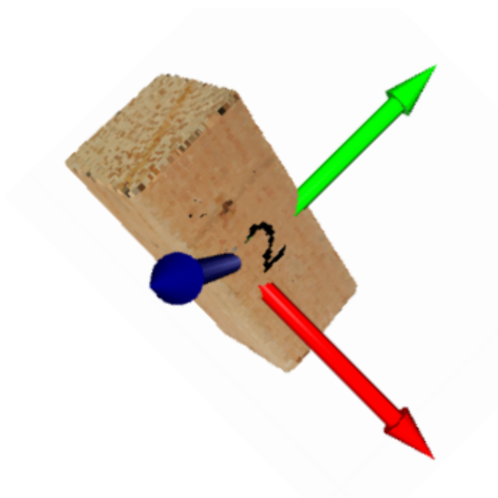
Extrinsics v.s. Intrinsics

- Extrinsics: cares about the embedding of a surface
  - Looking at the surface **outside** the surface

Extrinsics v.s. Intrinsics

- Extrinsics: cares about the embedding of a surface
  - Looking at the surface **outside** the surface
  - For example:
    - est. 6D pose of a rigid body





Extrinsics v.s. Intrinsics

- Extrinsics: cares about the embedding of a surface
  - Looking at the surface **outside** the surface
  - For example:
    - est. 6D pose of a rigid body
    - est. Articulation state of joints linking rigid bodies

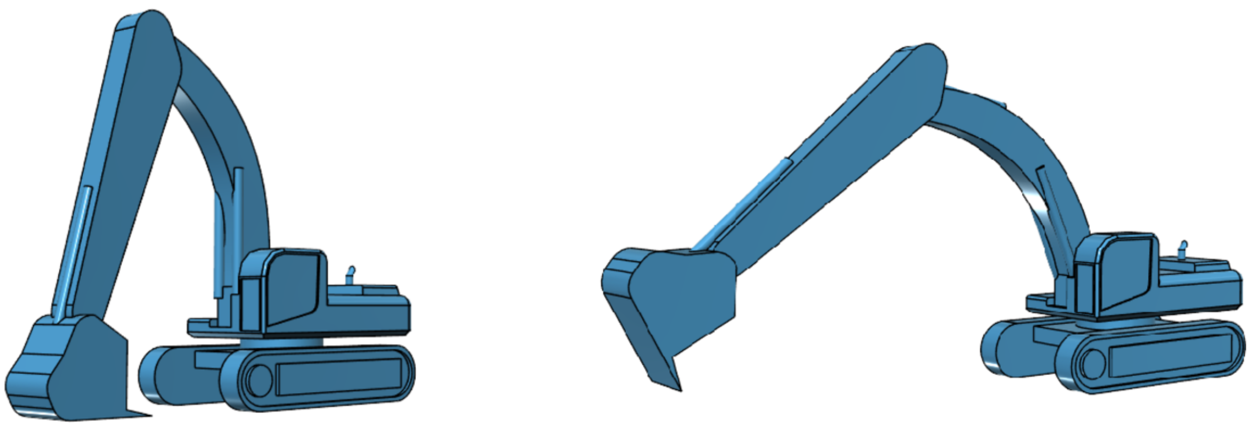
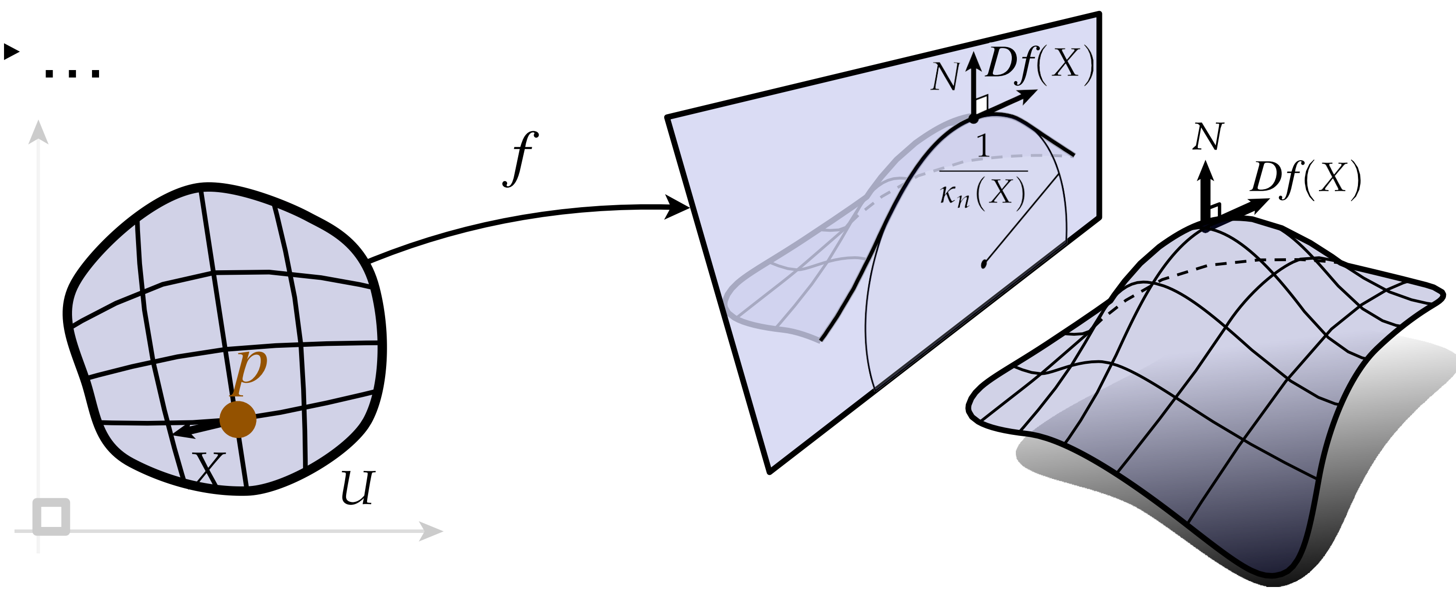


Image credit to Michael Spurr

Extrinsics v.s. Intrinsics

- Extrinsics: cares about the embedding of a surface
  - Looking at the surface **outside** the surface
  - For example:
    - est. 6D pose of a rigid body
    - est. Articulation state of joints linking rigid bodies
    - est. Normal curvature
    - …



Extrinsics v.s. Intrinsics

- Extrinsics: cares about the embedding of a surface
  - Looking at the surface **outside** the surface
  - For example:
    - est. 6D pose of a rigid body
    - est. Articulation state of joints linking rigid bodies
    - est. Normal curvature
    - …

- Intrinsics:
  - Looking at the surface **inside** the surface

Extrinsics v.s. Intrinsics

- Extrinsics: cares about the embedding of a surface
  - Looking at the surface **outside** the surface
  - For example:
    - est. 6D pose of a rigid body
    - est. Articulation state of joints linking rigid bodies
    - est. Normal curvature
    - …

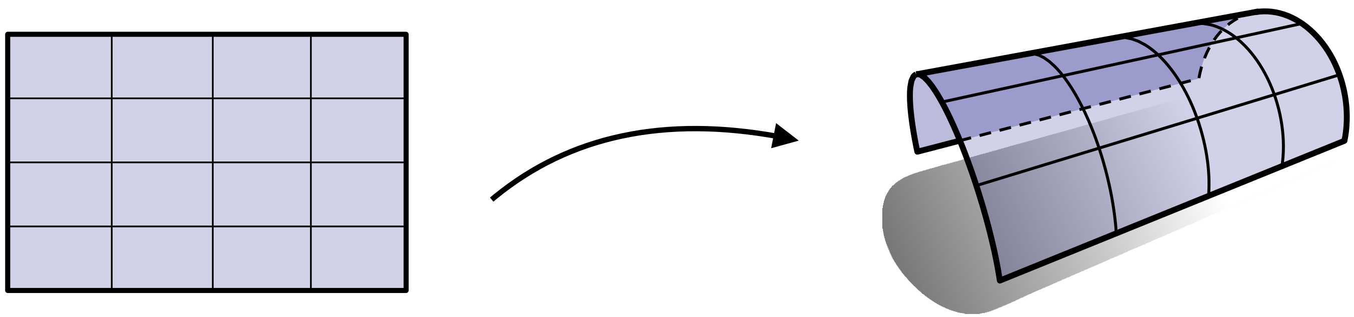
- Intrinsics:
  - Looking at the surface **inside** the surface
  - **Geodesic distance** uniquely determines shape intrinsics

Recall:

Gaussian and Mean Curvatures

Gaussian: $K := \kappa_1 \kappa_2$

Mean: $H := \frac{1}{2}(\kappa_1 + \kappa_2)$



$$K = 0 \quad H = 0 \quad \kappa_n(X) \equiv 0$$

$$K = 0 \quad H \neq 0 \quad \kappa_n(X) \not\equiv 0$$

Read by yourself

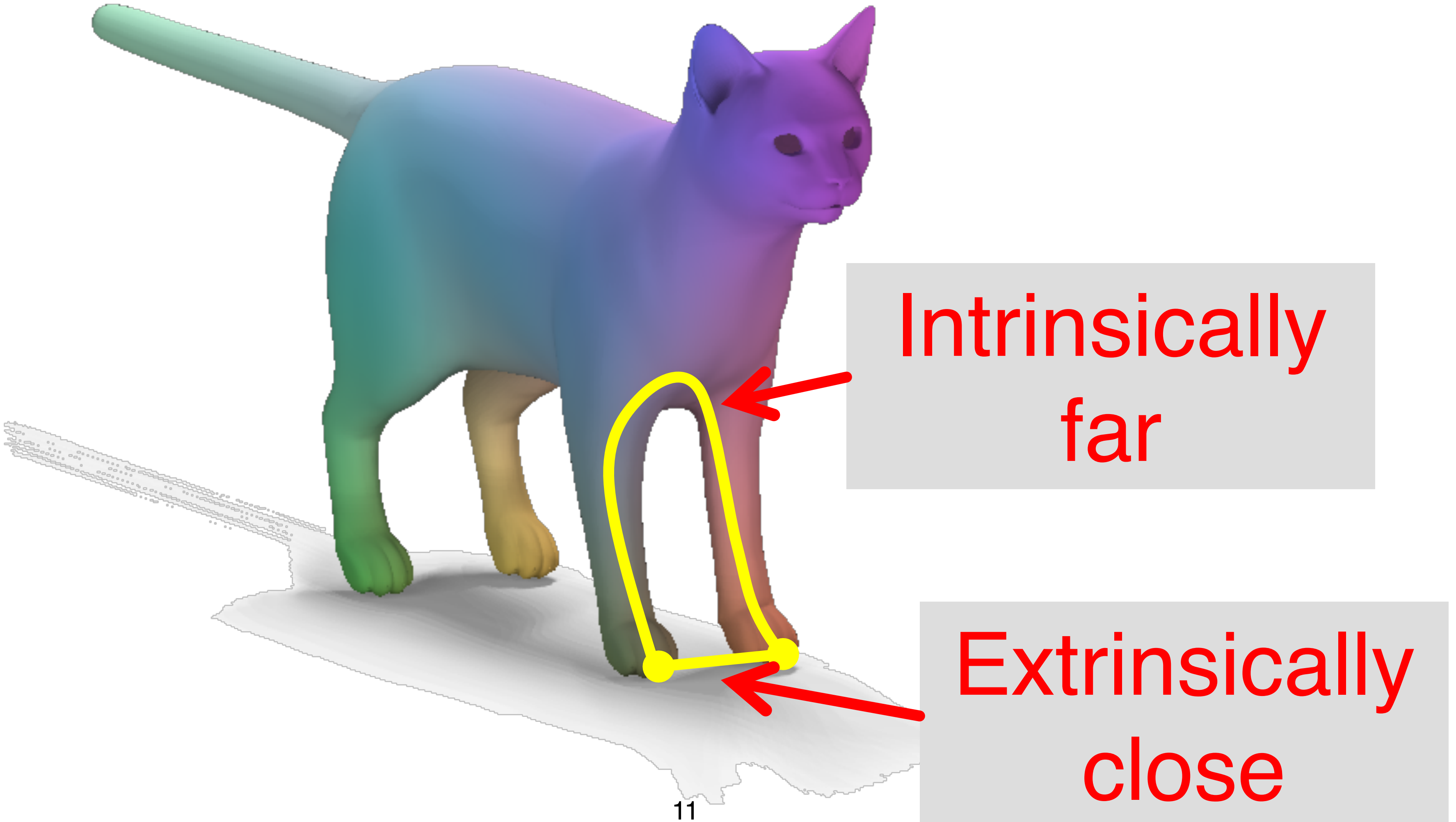
Recall: Theorema Egregium

The Gaussian curvature of an embedded smooth surface in $\mathbb{R}^3$ is invariant under the local isometries.

Gaussian Curvature is an Intrinsic Measurement of a Surface

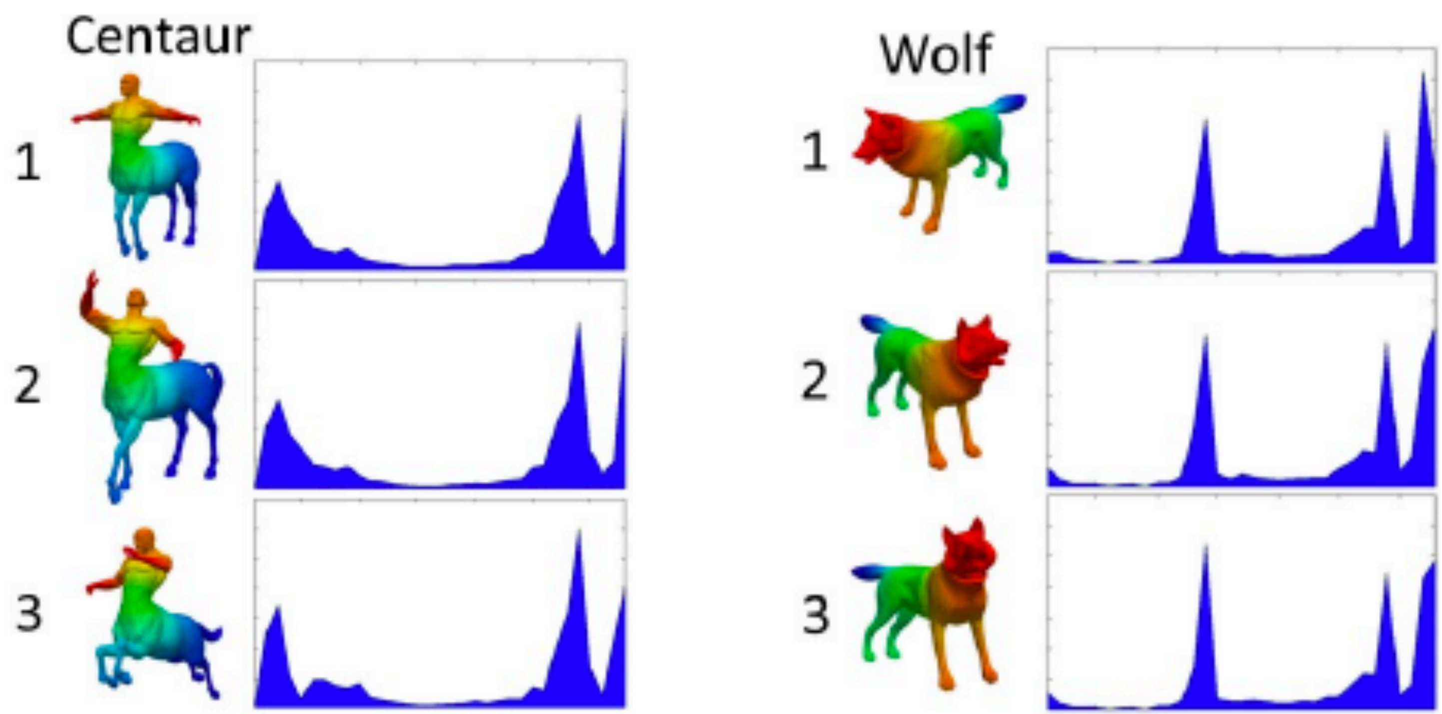
Read by yourself

Geodesic Distances



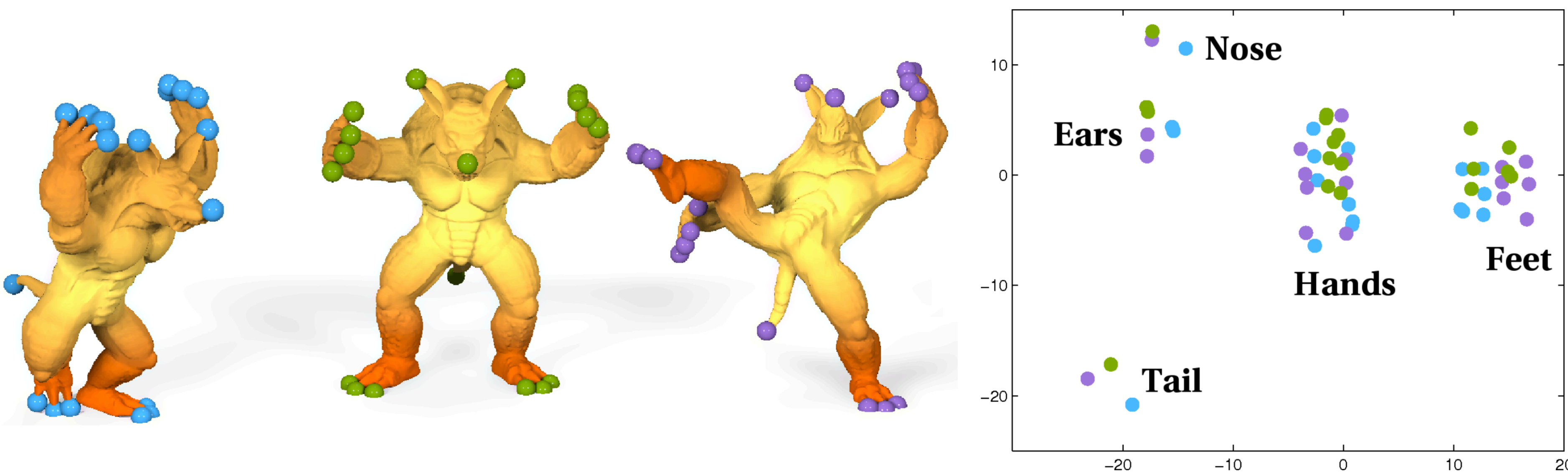
Applications: Shape Classification

- By classical method: Distribution of distances for point pairs randomly picked on the surface



Applications: Deformation-Invariant Point Feature

- By learning-based method: Extract some geodesic distance based features (intrinsic features)



Agenda

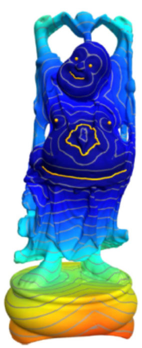
- Analytic Method for Computing Geodesics
- Learning to Predict Geodesics for Point Cloud
- Applications of Learning Point Cloud Geodesics
  - Normal Estimation
  - Mesh Reconstruction

Analytical Method for Computing Geodesics

Related Queries

Single source

Multi-source



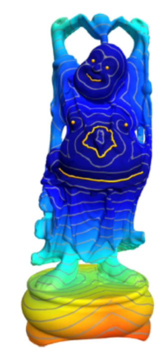
All-pairs

https://www.ceremade.dauphine.fr/~peyre/teaching/manifold/tp3.html http://www.sciencedirect.com/science/article/pii/S001044851100226

Related Queries

Single source

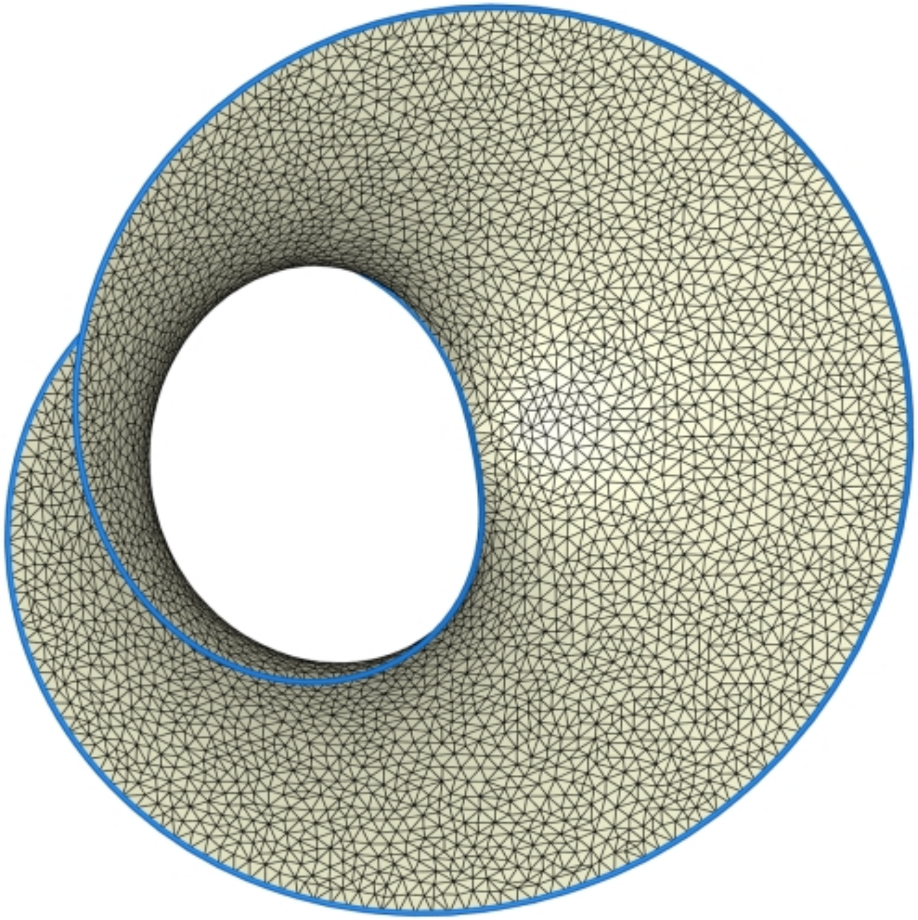
Multi-source



All-pairs

https://www.ceremade.dauphine.fr/~peyre/teaching/manifold/tp3.html http://www.sciencedirect.com/science/article/pii/S001044851100226

Geodesics on Meshes

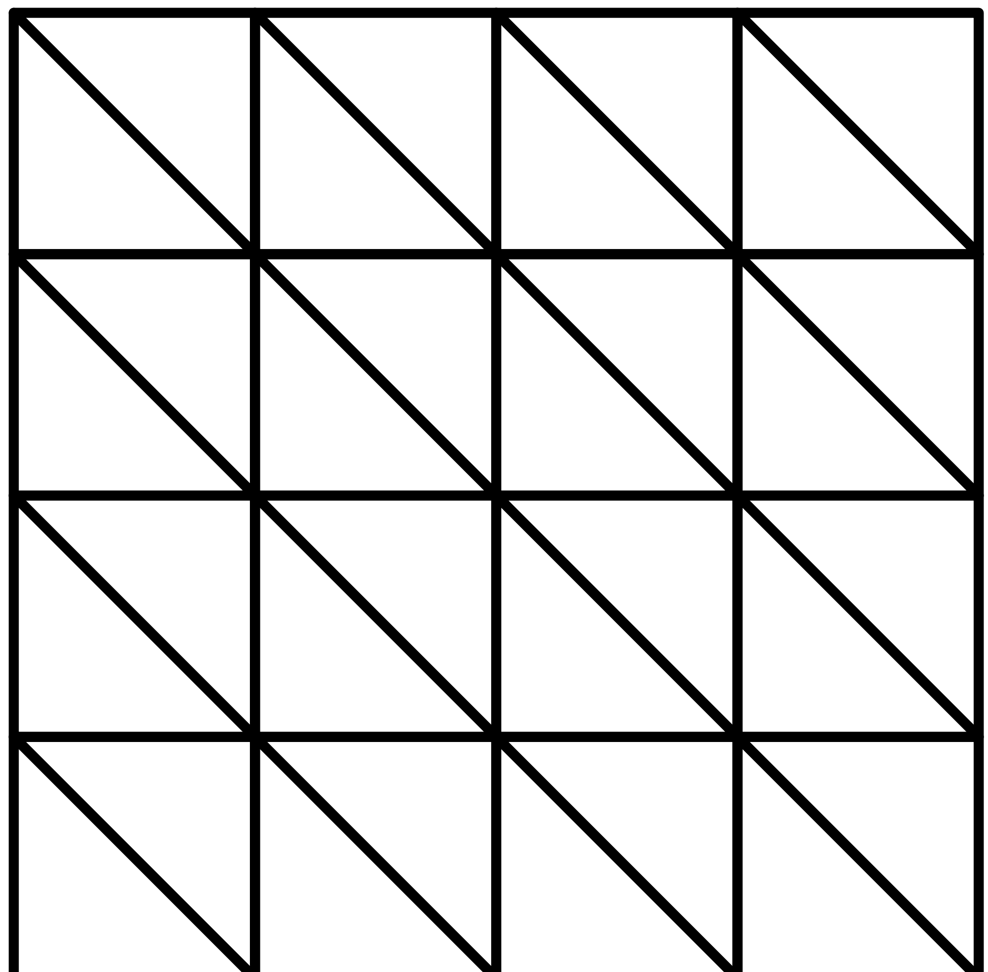


Approximate geodesics as paths along edges

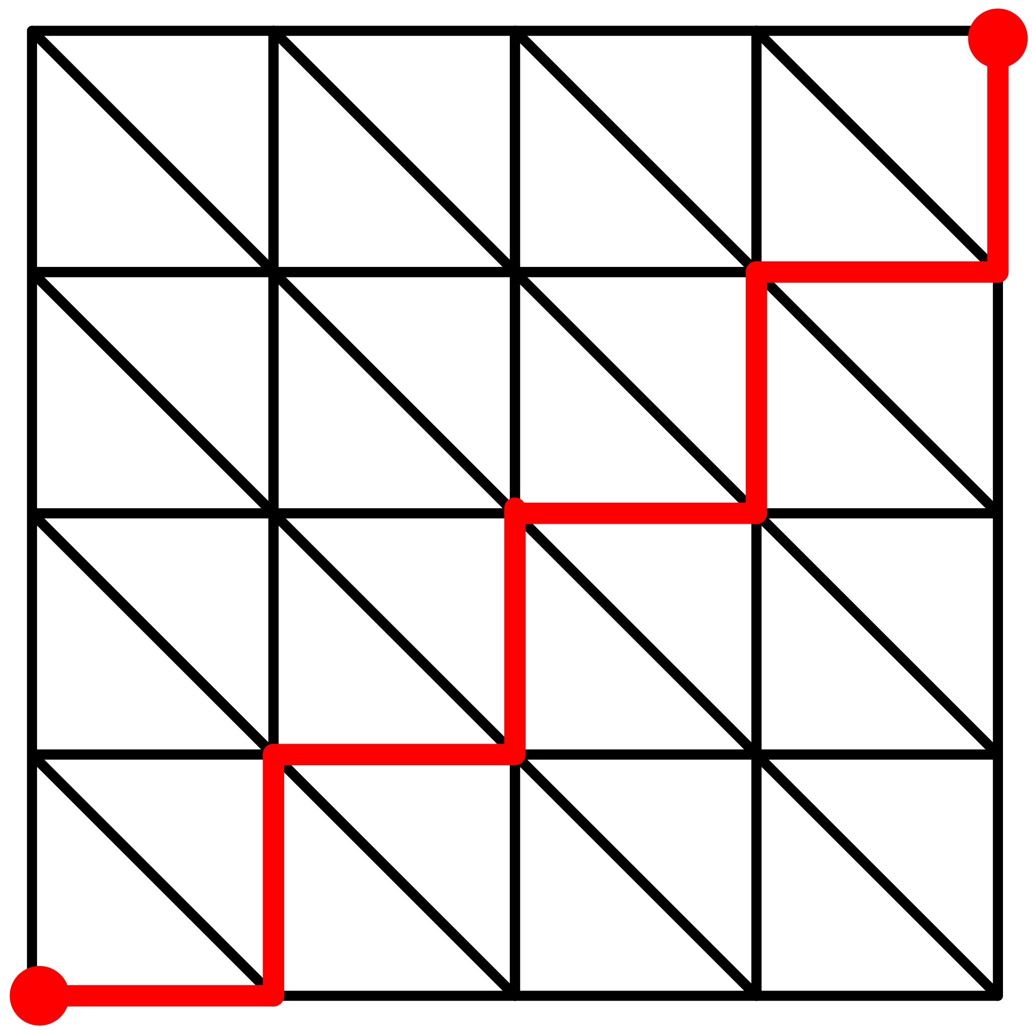
http://www.cse.ohio-state.edu/~tamaldey/isotopic.html

Meshes are graphs

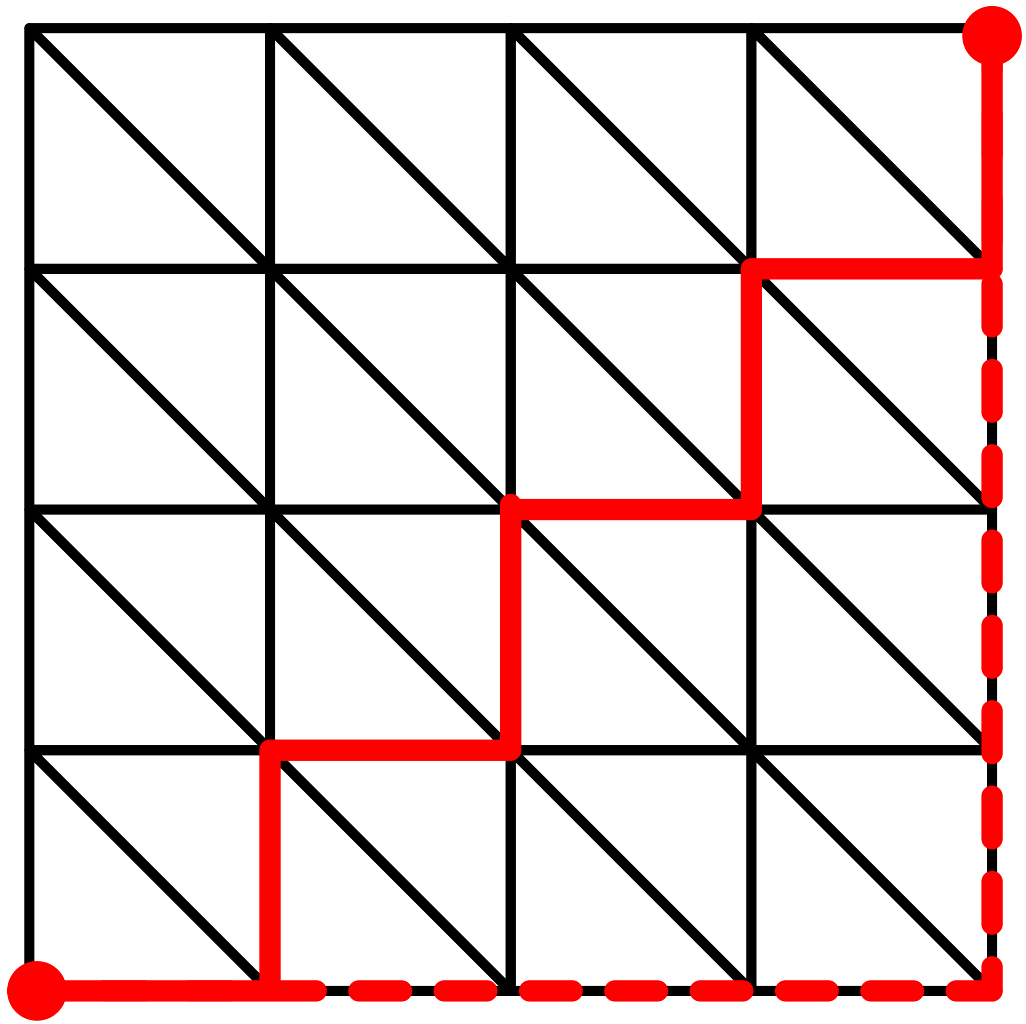
Can We Use Shortest Path Algorithms to Compute Geodesics?



Can We Use Shortest Path Algorithms to Compute Geodesics?



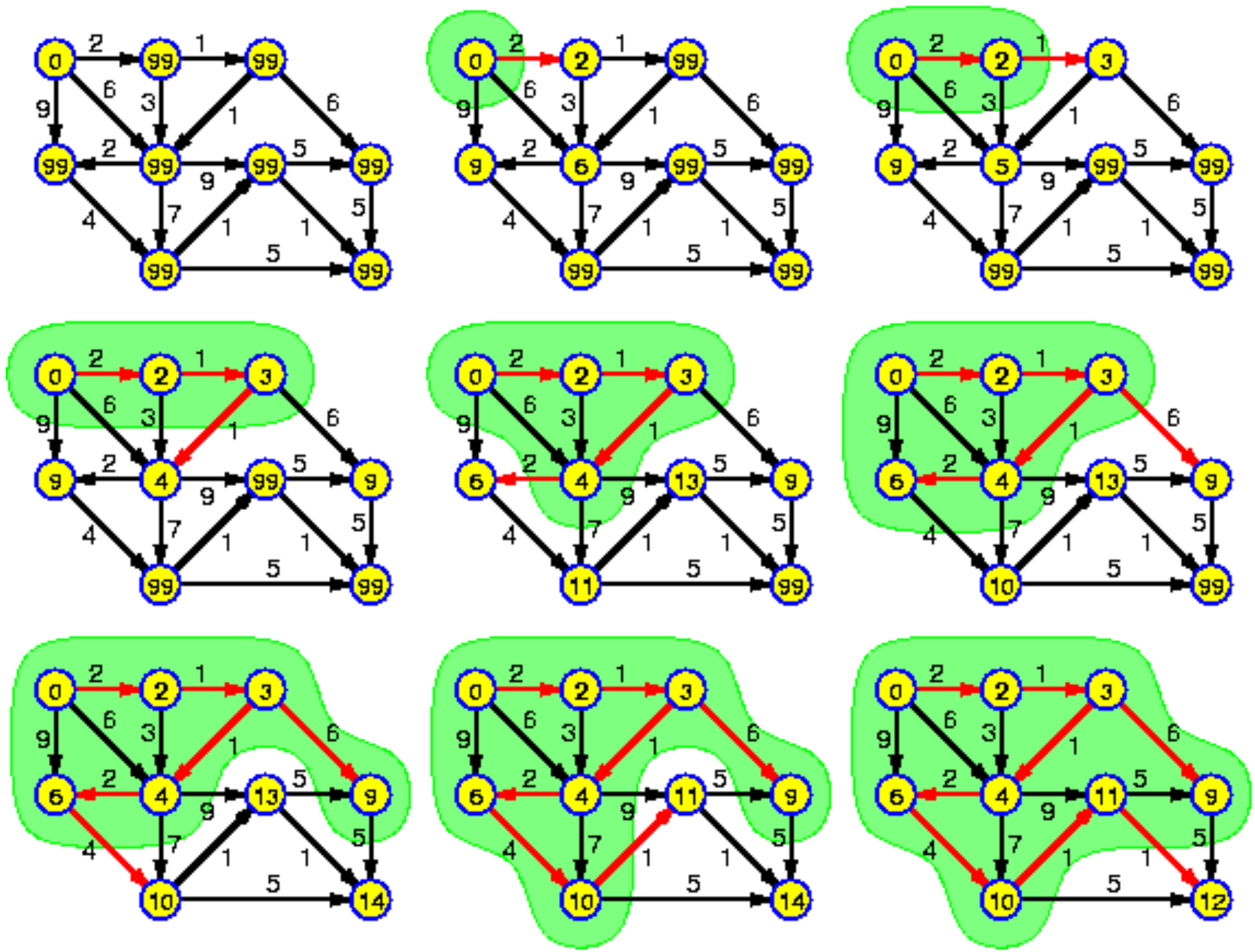
Can We Use Shortest Path Algorithms to Compute Geodesics?



Fast Marching Algorithm

Dijkstra's algorithm, modified to approximate geodesic distances.

Review of Dijkstra's Algorithm

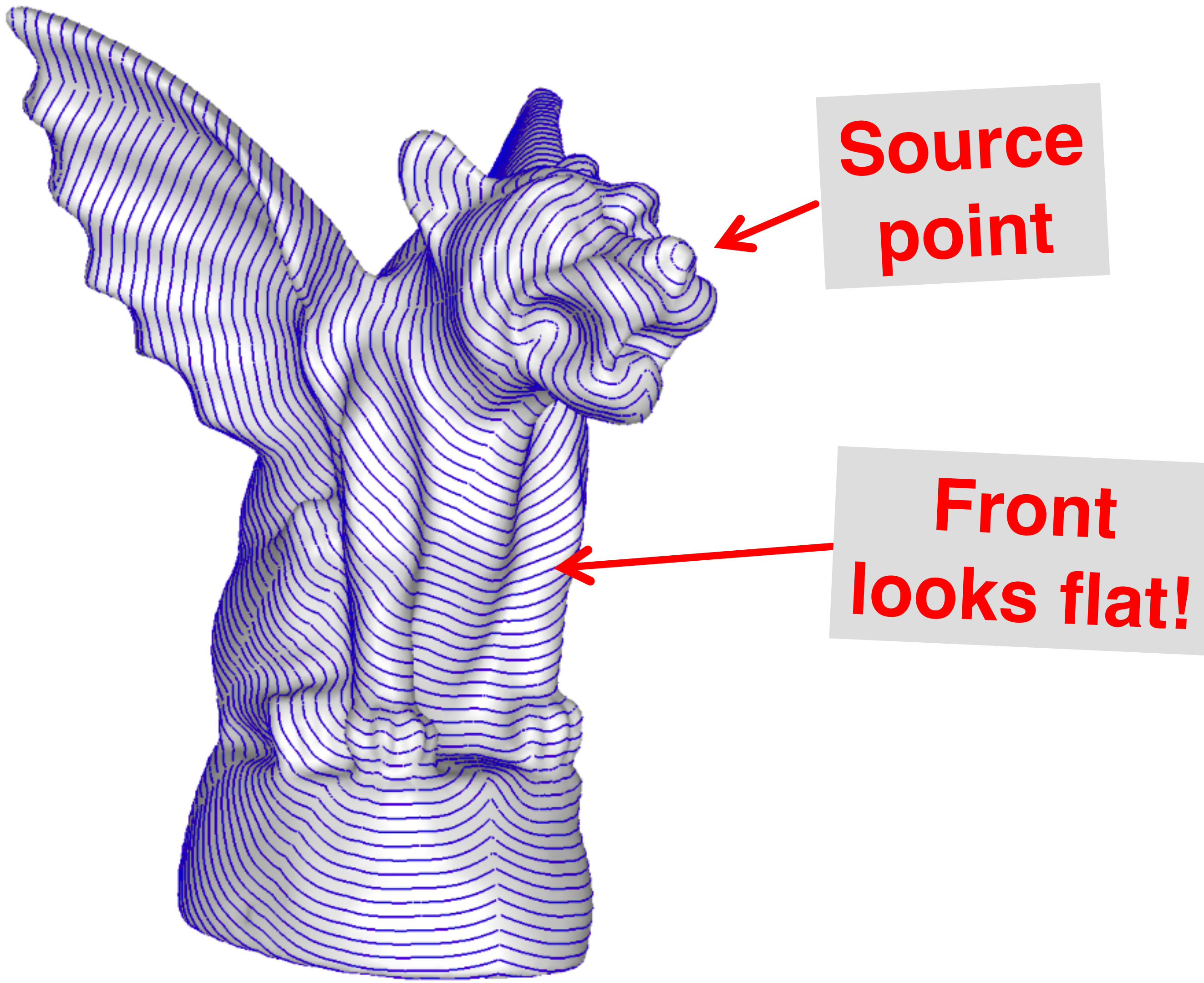


http://www.iekucukcay.com/wp-content/uploads/2011/09/dijkstra.gif

Key Idea of Dijkstra's Algorithm

- Maintain a frontier set of vertices with the shortest distance from the source
- Propagate to the neighborhood

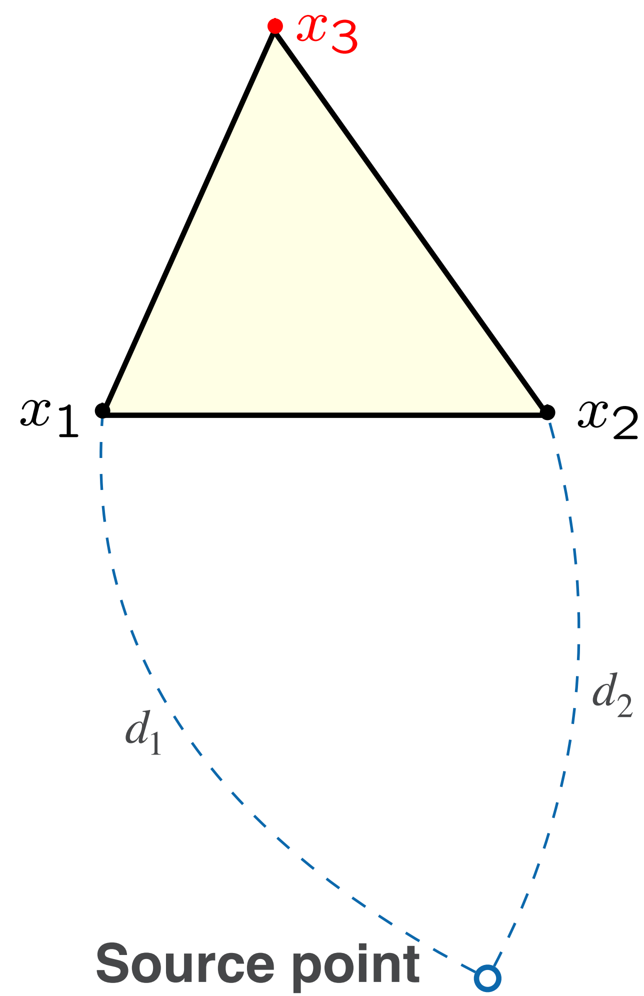
Fast Marching Algorithm



http://research.microsoft.com/en-us/um/people/hoppe/geodesics.pdf

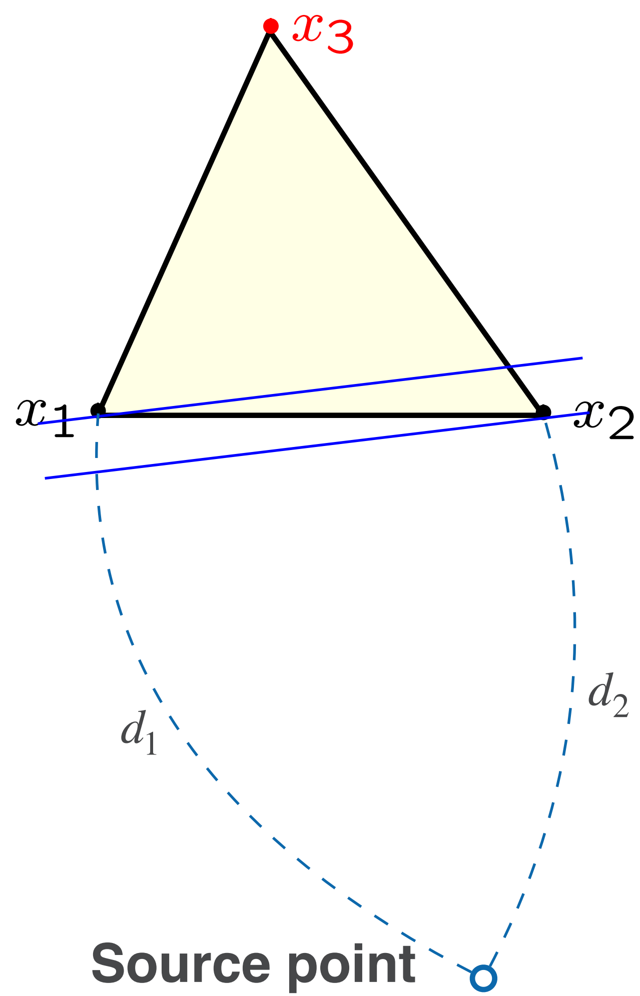
Fast Marching Algorithm

- At x_1 and x_2 stores the shortest paths d_1 and d_2
- Question: shortest path d_3 at x_3?

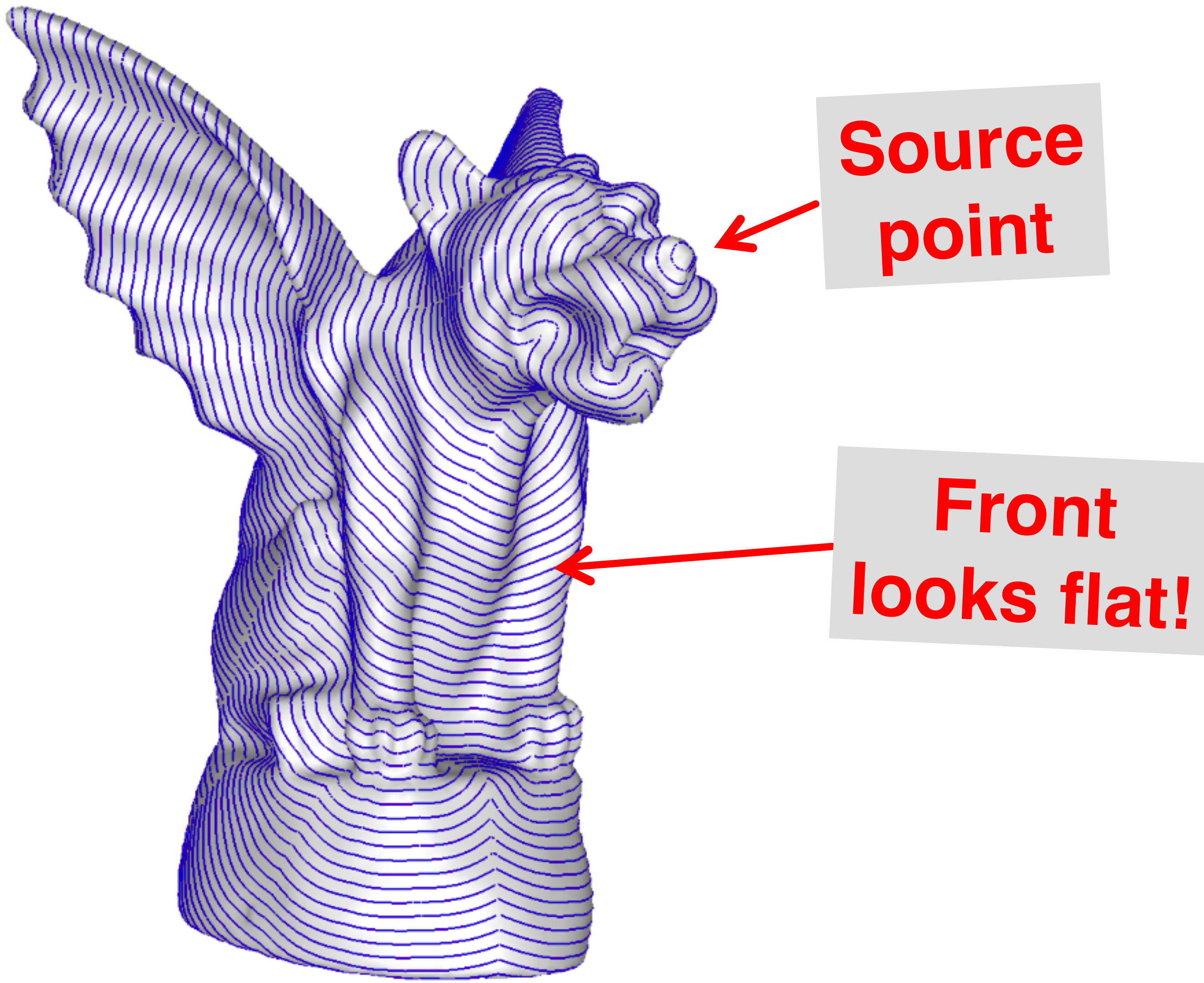


Fast Marching Algorithm

- Idea: Planar front approximation ($\because |x_1 x_2| \approx 0$)



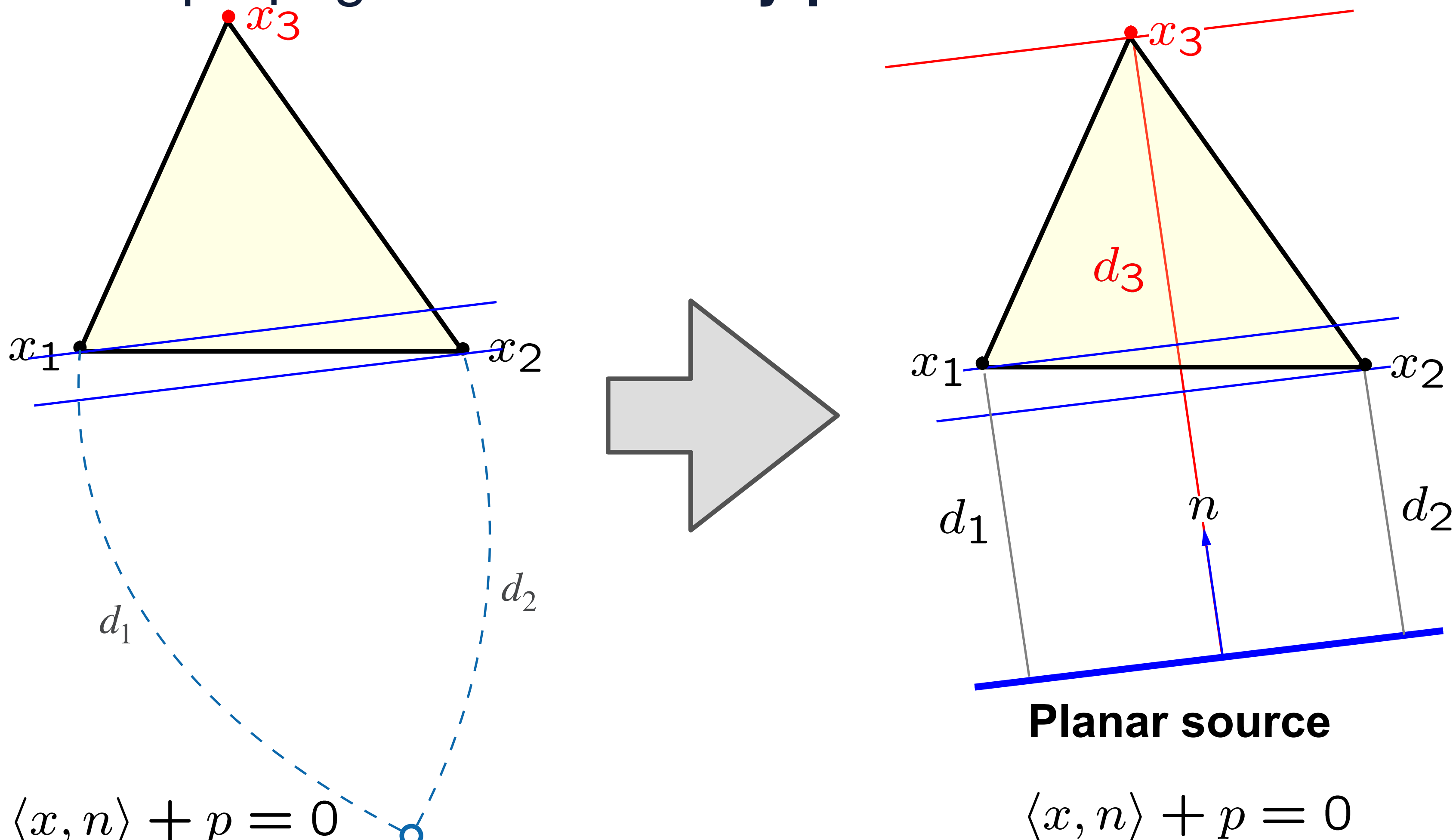
Trick: Every front looks flat!



http://research.microsoft.com/en-us/um/people/hoppe/geodesics.pdf

Fast Marching Algorithm

- Change of view: point source → planar source
- Front propagates from **every point** on the source



Fast Marching Update Steps

- Front hits x_1 at time d_1
- Hits x_2 at time d_2
- When does the front arrive x_3?

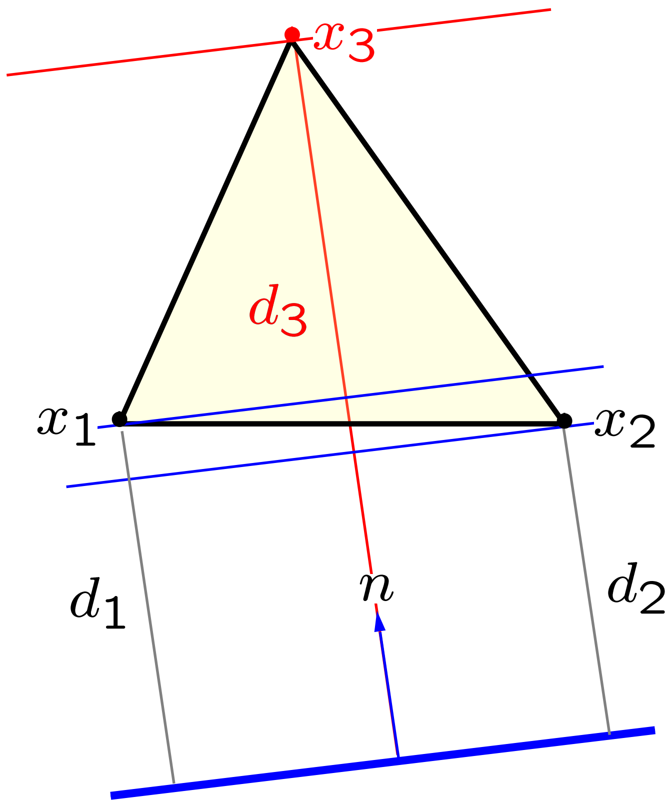
- Step 1: Model **wave front** propagation from a planar source

$$\langle x, n \rangle + p = 0$$

n : unit **propagation direction**

p : source **offset**

- Step 2: Compute the distance from x_3 to the plane



Planar source

$$\langle x, n \rangle + p = 0$$

Practical Implementation

Fast Exact and Approximate Geodesics on Meshes

Vitaly Surazhsky, University of Oslo
Tatiana Surazhsky, University of Oslo
Danil Kirsanov, Harvard University
Steven J. Gortler, Harvard University
Hugues Hoppe, Microsoft Research

Abstract

The computation of geodesic paths and distances on triangle meshes is a common operation in many computer graphics applications. We present several practical algorithms for computing such geodesics from a source point to one or all other points efficiently. First, we describe an implementation of the exact "single source, all destination" algorithm presented by Mitchell, Mount, and Papadimitriou (MMP). We show that the algorithm runs much faster in practice than suggested by worst case analysis. Next, we extend the algorithm with a merging operation to obtain computationally efficient and accurate approximations with bounded error. Finally, to compute the shortest path between two given points, we use a lower-bound property of our approximate geodesic algorithm to efficiently prune the frontier of the MMP algorithm, thereby obtaining an exact solution even more quickly.

Keywords: shortest path, geodesic distance.

1 Introduction

In this paper we present practical methods for computing both exact and approximate shortest (i.e. geodesic) paths on a triangle mesh. These geodesic paths typically cut across faces in the mesh and are therefore not found by the traditional graph-based Dijkstra algorithm for shortest paths.

The computation of geodesic paths is a common operation in many computer graphics applications. For example, parameterizing a mesh often involves cutting the mesh into one or more charts (e.g. [Krishnamurthy and Levoy 1
the result generally has less distorti
if the cuts are geodesic. Geodesic

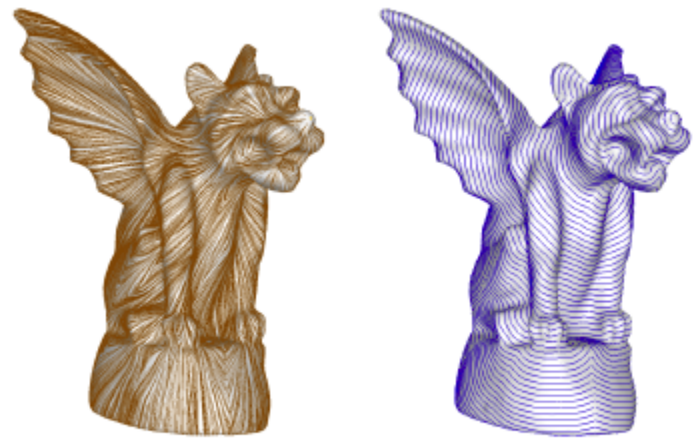


Figure 1: Geodesic paths from a source vertex, and isolines of the geodesic distance function.

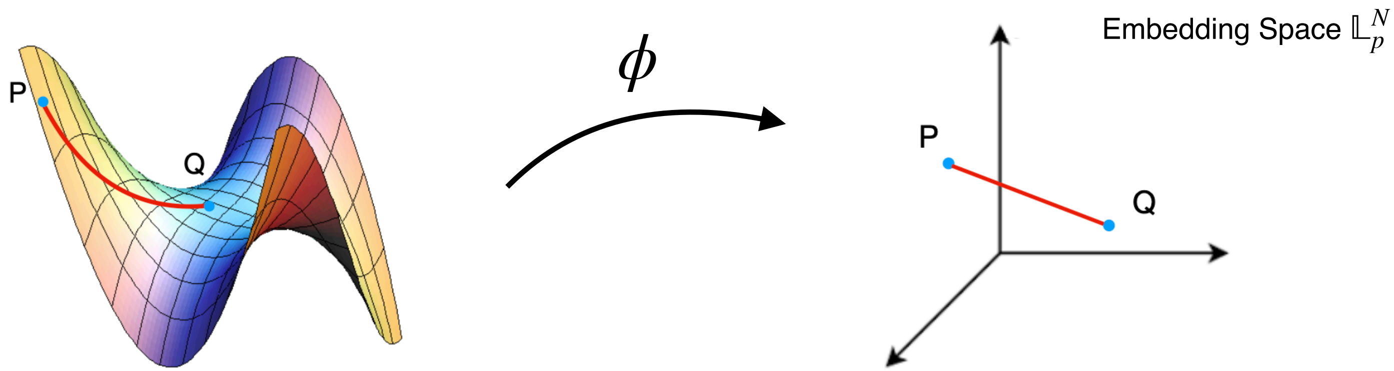
tance function over the edges, the implementation is actually practical even though, to our knowledge, it has never been done previously. We demonstrate that the algorithm's worst case running time of $O(n^2 \log n)$ is pessimistic, and that in practice, the algorithm runs in sub-quadratic time. For instance, we can compute the exact geodesic distance from a source point to all vertices of a 400K-triangle mesh in about one minute.

Approximation algorithm We extend the algorithm with a merging operation to obtain computationally efficient and accurate approximations with *bounded* error. In practice, the algorithm runs in

http://code.google.com/p/geodesic/

Learning to Predict Geodesics

Metric Learning Formulation

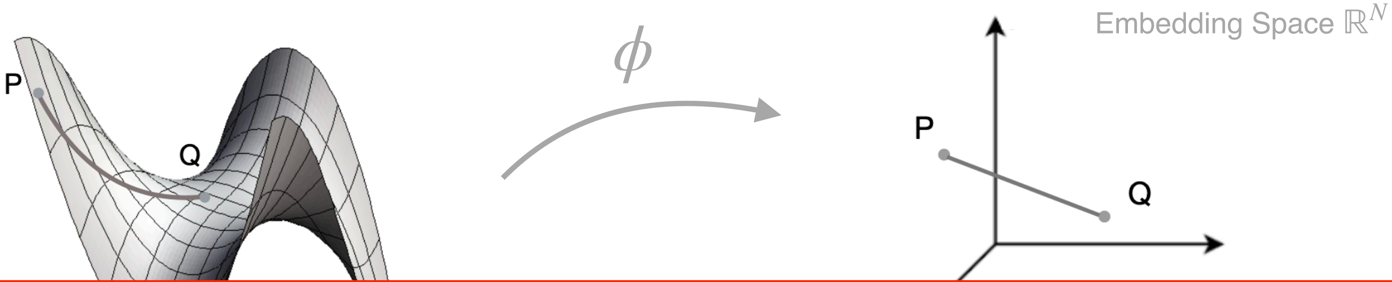


- Given points P, Q on the surface, what is $d_G(P, Q)$?
- Metric learning:

$$d_G(P, Q) \approx \|\phi(P) - \phi(Q)\|_p$$

- ϕ: embedding function (e.g., MLP)
- Usually, $\dim(\phi(\,\cdot\,)) \gg 3$ and $p = 2$

Metric Learning Formulation



Q: Could "$\approx$" become "$=$" with $\dim(\phi(\,\cdot\,)) < \infty$?

- Metric learning:

$$d_G(P,Q) \approx \|\phi(P) - \phi(Q)\|_p$$

- ϕ: embedding function (e.g., MLP)
- Usually, $\dim(\phi(\,\cdot\,)) \gg 3$ and $p = 2$

Counter-Example

- Recall our $SO(2)$ example:
  - Distance preservation (geodesic v.s. ℓ_p) would preserve topological structure
  - Circle has different topology as line segments
  - So $SO(2)$ cannot be embedded into Euclidean space without distortion

- Some results of the embedding for general metric space to come

Some Result of Embedding *General Metrics* in ℓ_p

Distortion of Embedding

- Metrics:
  - $d(x, y) \geq 0$
  - $d(x, x) = 0$ and $\forall y \neq x, d(x, y) > 0$
  - $d(x, y) + d(y, z) \geq d(x, z)$

- Assume $\phi : S \mapsto T$ and $d_S(\cdot, \cdot)$ and $d_T(\cdot, \cdot)$
  - expansion$(f) = \max_{x,y} \dfrac{d_T(\phi(x), \phi(y))}{d_S(x, y)}$
  - contraciton$(f) = \max_{x,y} \dfrac{d_S(x, y)}{d_T(\phi(x), \phi(y))}$
  - distortion$(f) =$ expansion $\times$ contraction

A Well-Known Result

Theorem (Bourgain, 1985).
Let (X, d) be a metric space on n points. Then,

$$(X, d) \xhookrightarrow{O(\log n)} \ell_p^{O(\log^2 n)}$$

Remarks:

- **Any** **n-point** metric space (X, d) can be embedded in ℓ_p with distortion $O(\log n)$
- To accommodate more points, one needs higher dim space
- This bound can be shown tight (by worse case, w.r.t. distortion)

Influence

- We cannot expect that a finite (even infinite) dimensional ℓ_p space would approximate any metric
- To compromise, we usually require the approximation to have higher precision for closer points:
    - The geometry underlying closer points may be better embeddable (e.g., on the same tangent plane)
    - Far away points just need to be known far away
- Recall the hinge loss for point feature learning in the *bottom-up instance segmentation* lecture:

$$L_{ij} = \max(0, K - \|F_i - F_j\|)$$

Learning to Predict Geodesics

Example work: GeoNet

Geodesics for Point Clouds

- No connectivity information

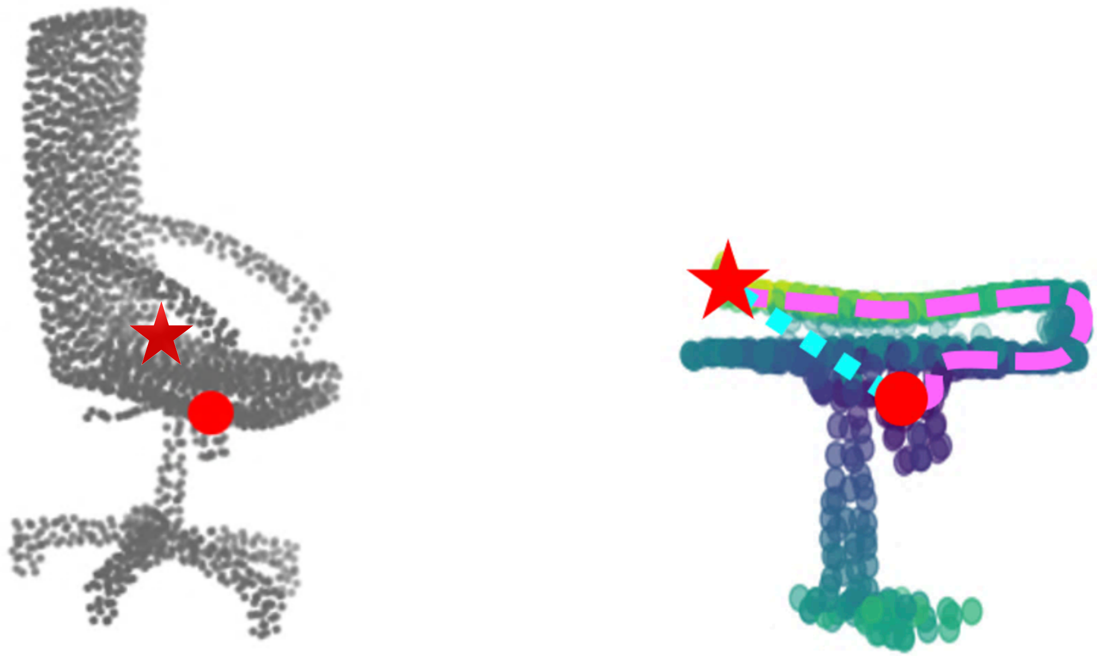


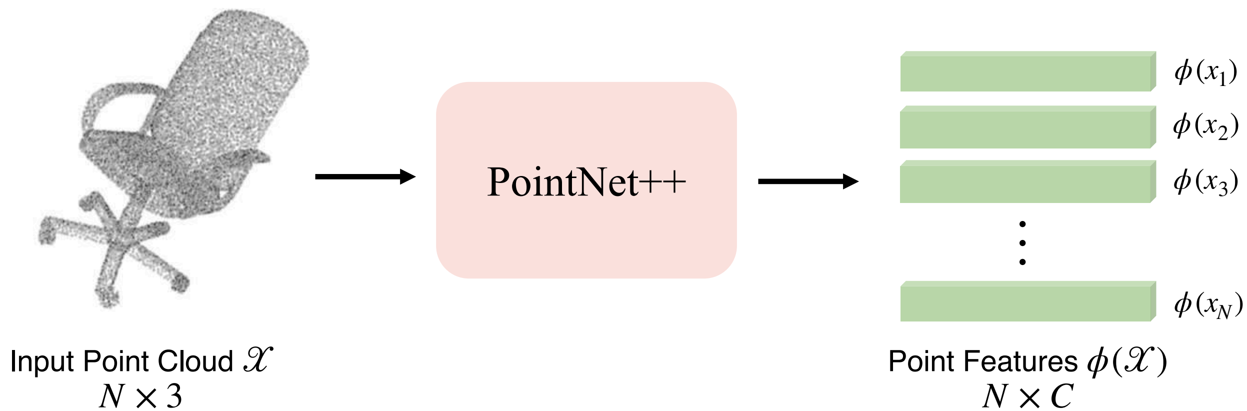
Figure from He, Tong, et al. "GeoNet: Deep geodesic networks for point cloud analysis." CVPR 2019 oral.

Task Formulation

- Input: Point Cloud $\mathscr{X} = \{x_i\}_{i=1}^{N}$
- Output: Geodesic distance to K Euclidean neighbors for each point

GeoNet: Geodesic Distance Regression

Step 1: Feature Extraction

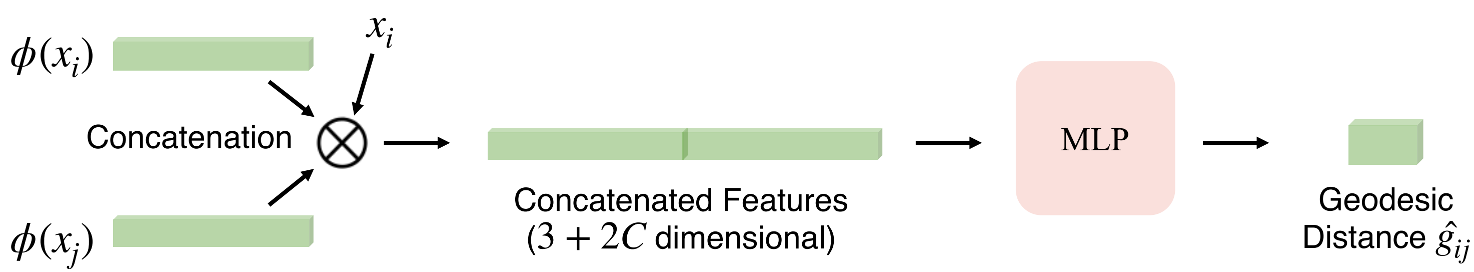


He, Tong, et al. "GeoNet: Deep geodesic networks for point cloud analysis." CVPR 2019.

GeoNet: Geodesic Distance Regression

Step 2: Metric Learning

For each x_i and its neighbor x_j:

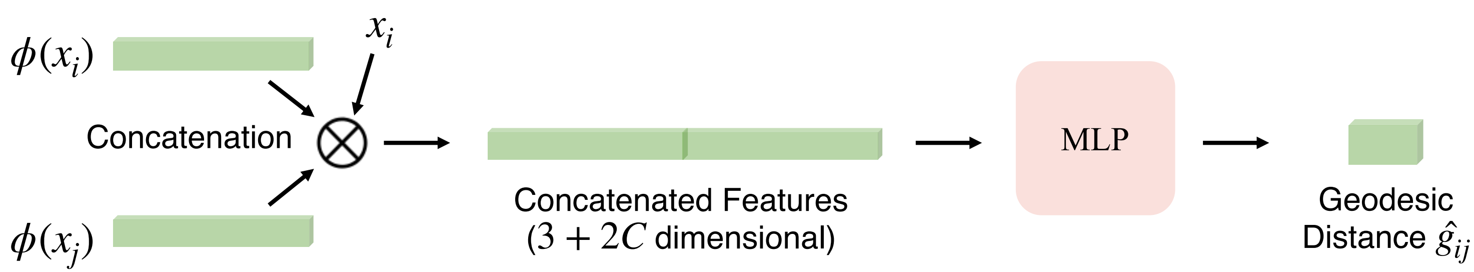


He, Tong, et al. "GeoNet: Deep geodesic networks for point cloud analysis." CVPR 2019.

GeoNet: Geodesic Distance Regression

Step 2: Metric Learning

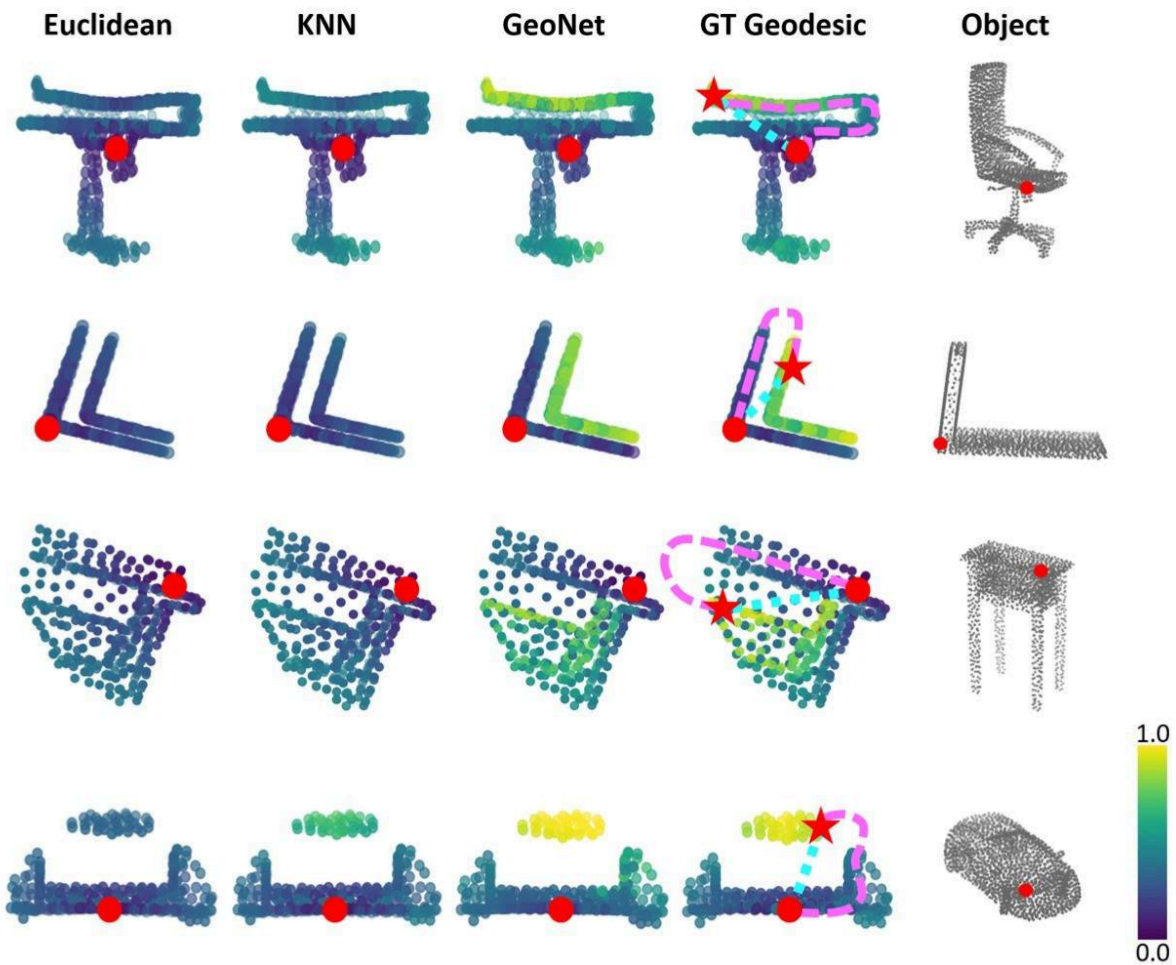
For each x_i and its neighbor x_j:



$$L_{geo} = \sum_{x_i \in \mathcal{X}} \sum_{x_j \in NN(x_i)} |g_{ij} - \hat{g}_{ij}|$$

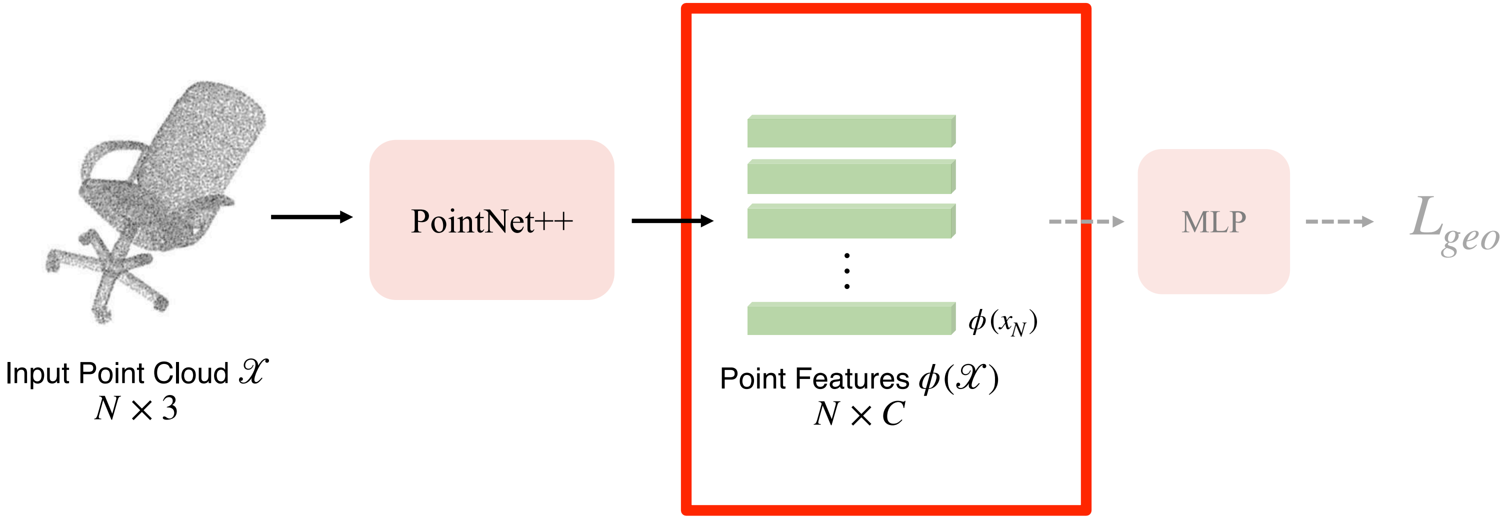
He, Tong, et al. "GeoNet: Deep geodesic networks for point cloud analysis." CVPR 2019.

Geodesics Regression



Geodesics Induced Features

- Geodesics induced features are useful for downstream tasks



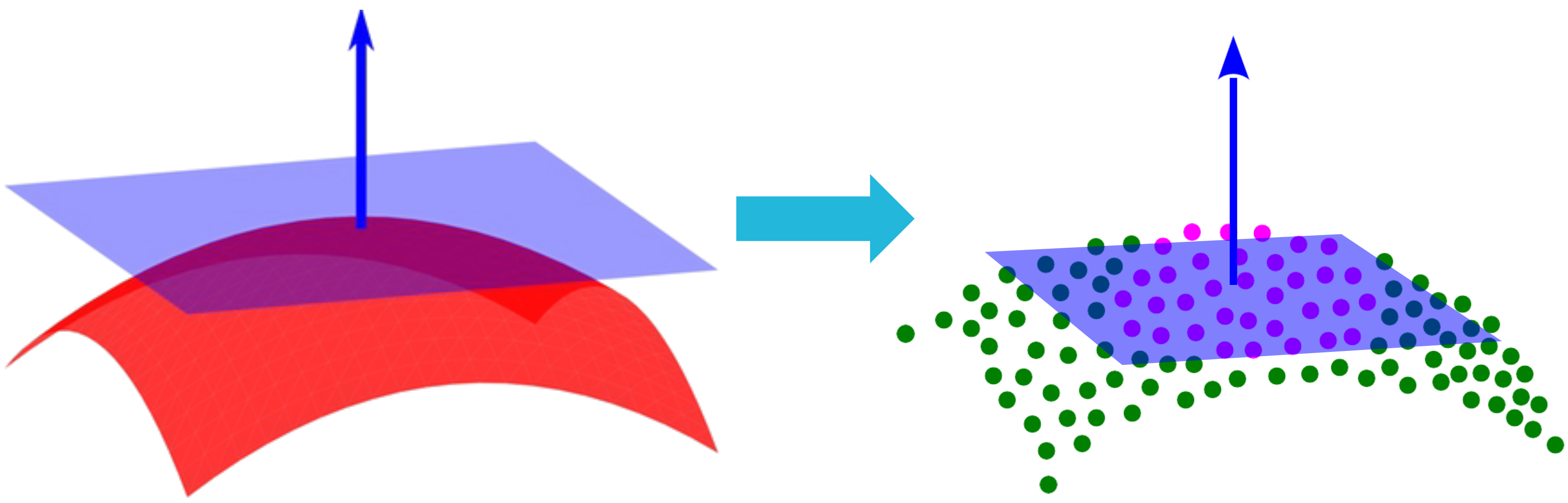
Applications

- Normal Estimation
- Mesh reconstruction

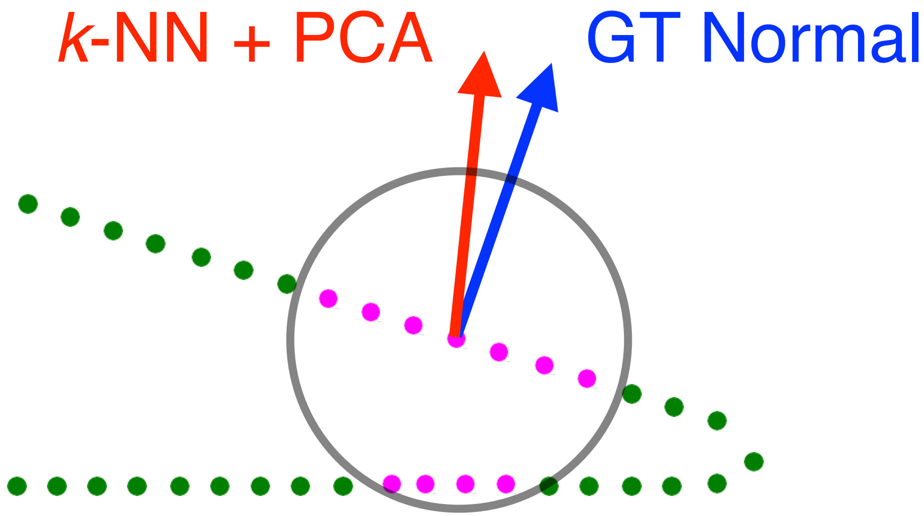
Recall:

Normal Estimation

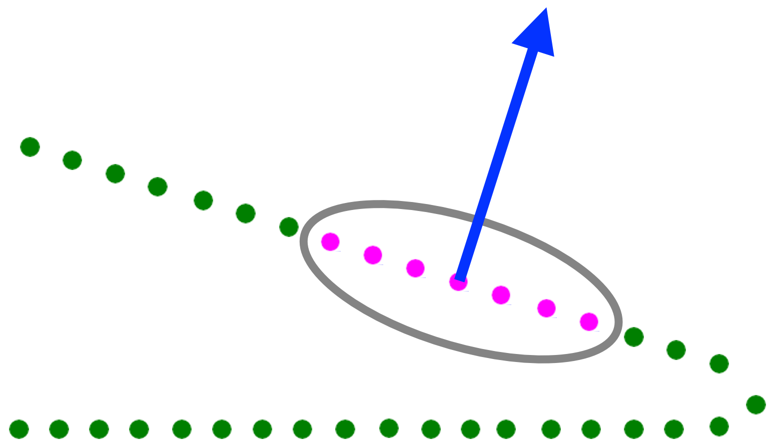
- Plane-fitting: find the plane that best fits the neighborhood of a point of interest
  - Let $M = \sum_i (x_i - \bar{x})(x_i - \bar{x})^T$ and $\bar{x} = \frac{1}{n}\sum_i x_i$,
  - n: the last principal component of M



Normal Estimation



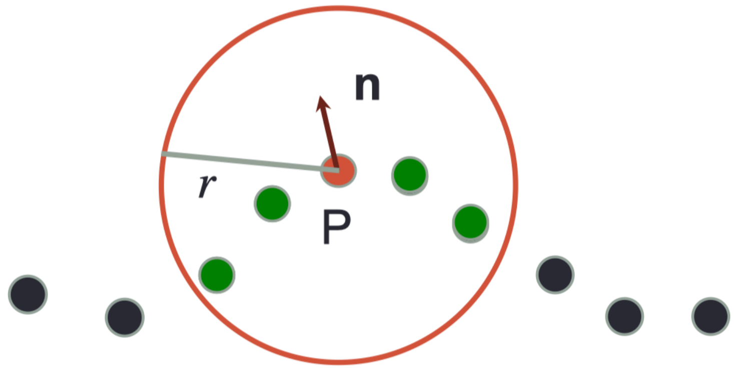
Normal Estimation



$d_G(u, v)$ is large although $d_E(u, v)$ is small!

$\Rightarrow$ Use geodesic neighborhoods for PCA

Normal Estimation



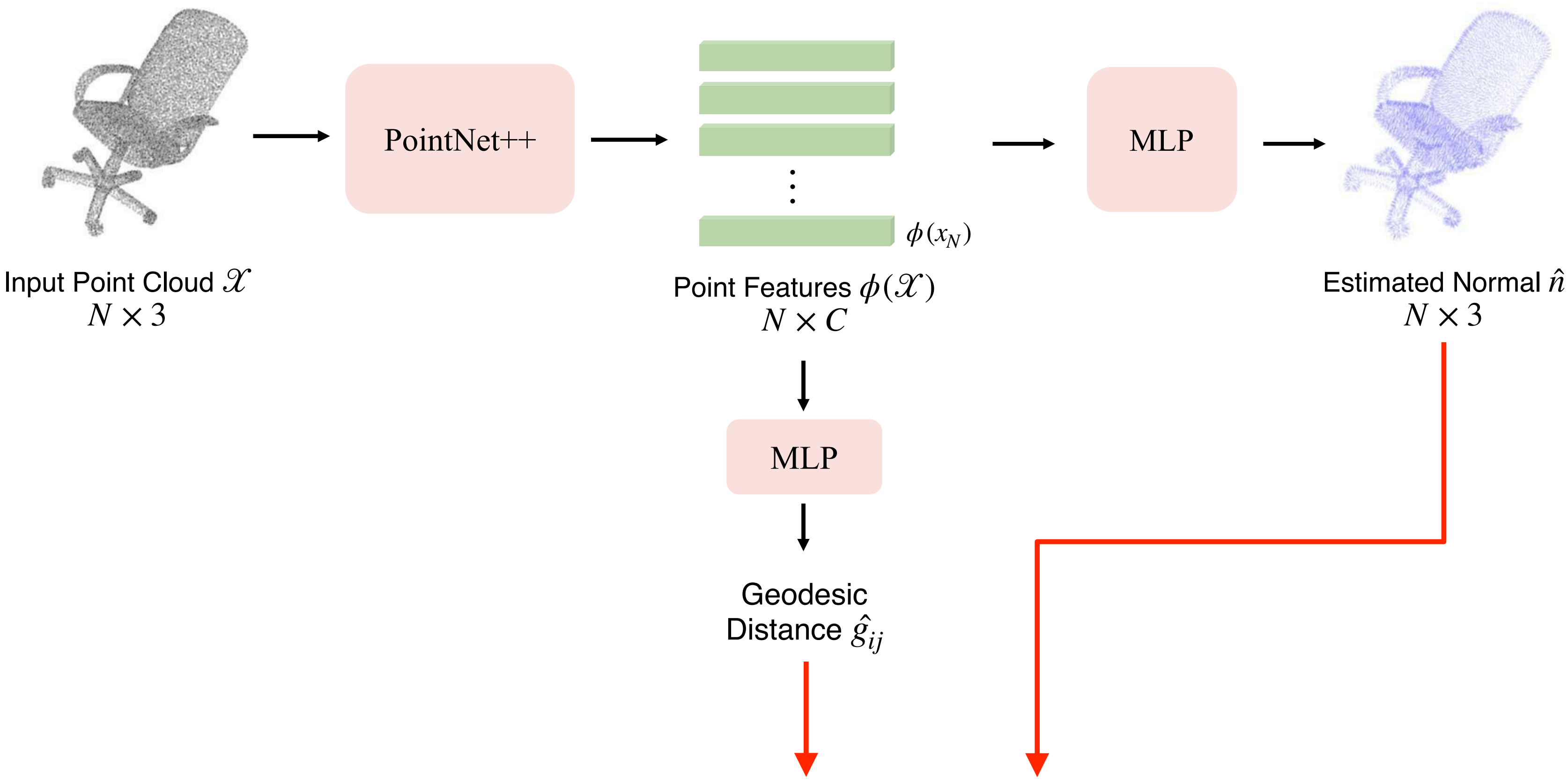
Critical parameter: number of neighbors *k*.
For unevenly sampled point cloud, we typically use all points inside a ball of radius *r*

Tricky to choose in practice!

Image courtesy of Maks Ovsjanikov
http://www.lix.polytechnique.fr/~maks/Verona_MPAM/Slides/lecture1.pdf

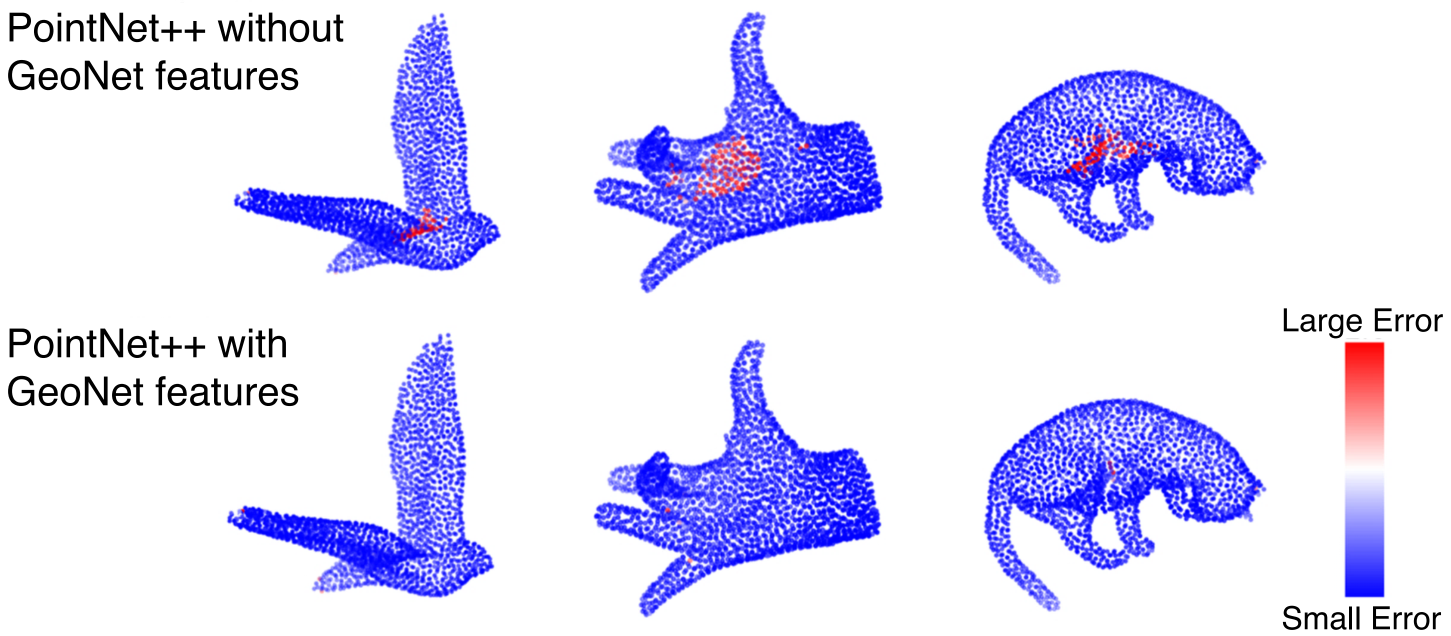
How about **predicting the normal**, without PCA and estimation of r or k?

Learning to Regress Normals

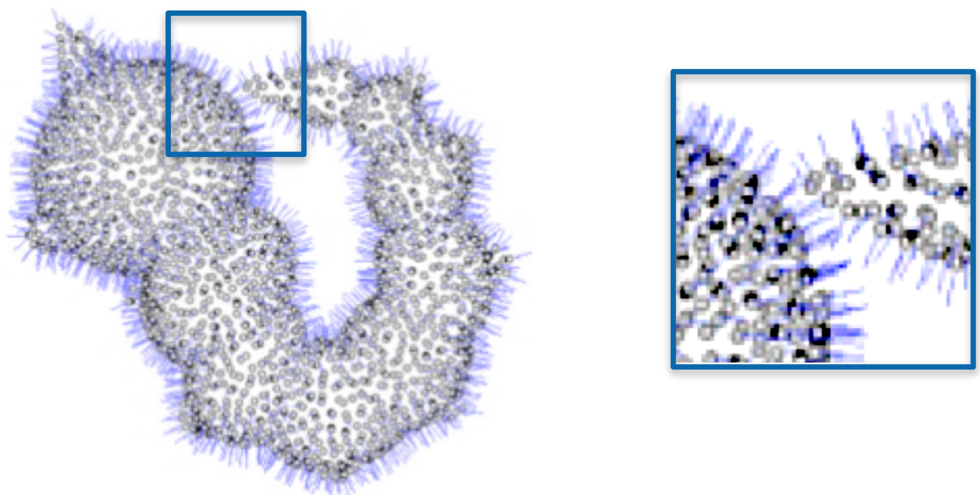


$$L_{total} = L_{geo} + L_{normal}$$

Error Patterns



Learning to Regress Normals



Advantage of Learning-based Normal Estimation

- Can be robust to point cloud sampling strategy
- Without the need to determine neighborhood size
- Get orientation consistent normals (in classical methods, this step is highly non-trivial)

Applications

- Normal Estimation
- Mesh Reconstruction

Mesh Reconstruction

- Input: Point Cloud
- Output: Polygon Mesh

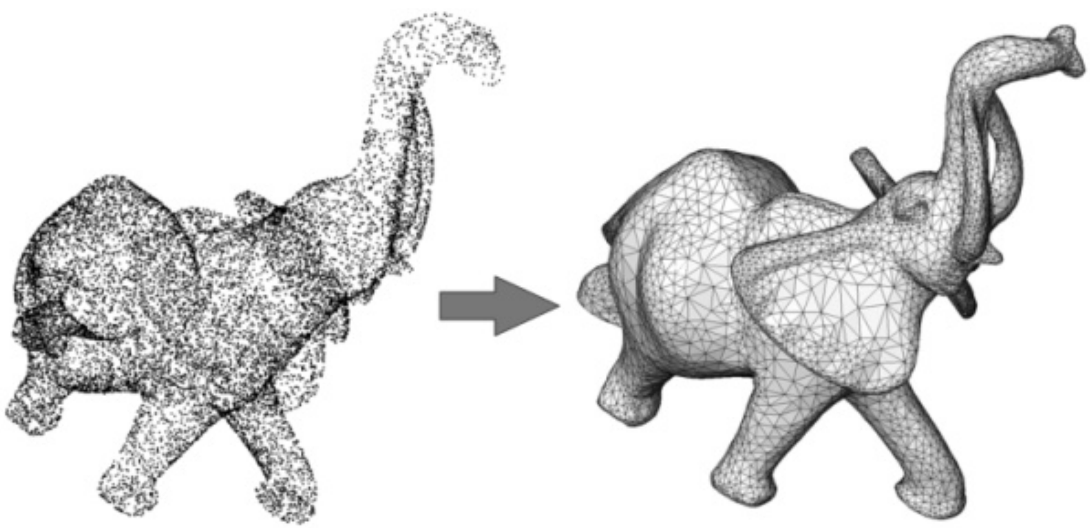


Figure courtesy of Pierre Alliez, Laurent Saboret, Gaël Guennebaud

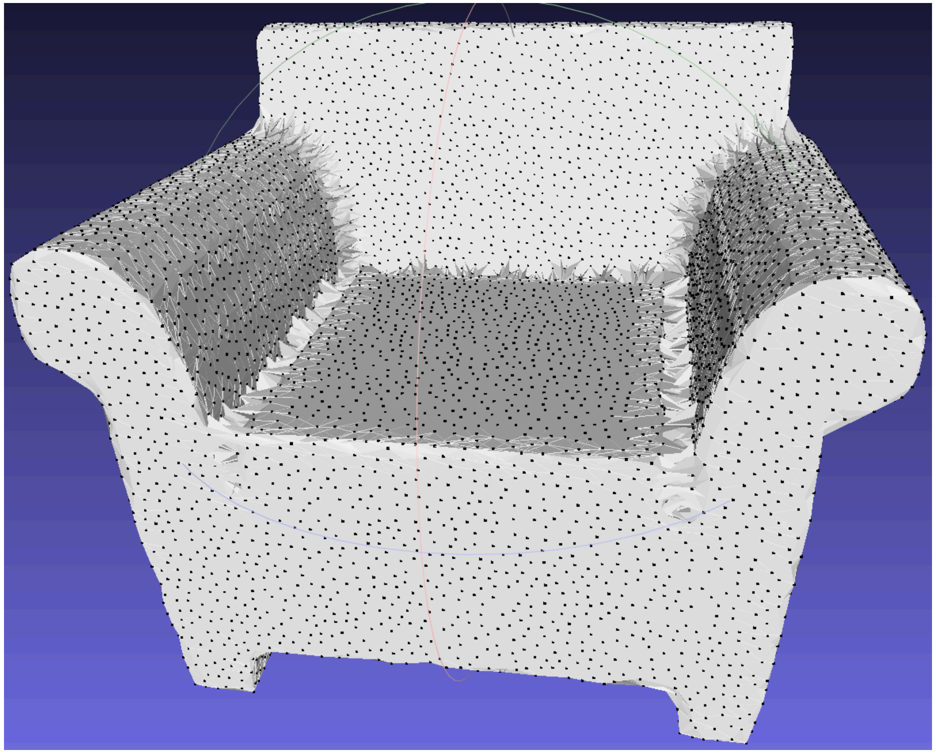
A Simple K-NN Approach

- For each point x_i we try to greedily add all triangles formed by x_i and its neighbors:

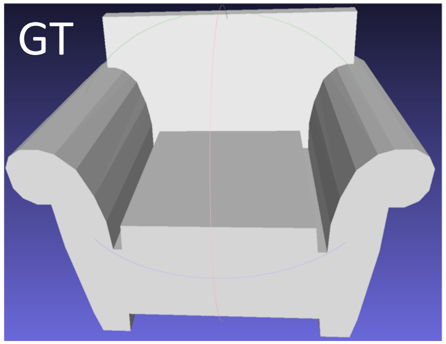
```
χ = {x_i}^n is the input point cloud
S = { } is the output mesh triangles

for i = 1 to n:
    find K nearest neighbors {p_l}_{l=1}^K of x_i:
    for j = 1 to K:
        for k = j + 1 to K:
            triangle △ = (x_i, N_j, N_k)
            if △ is not in S and S is manifold after adding △:
                add △ to S
```

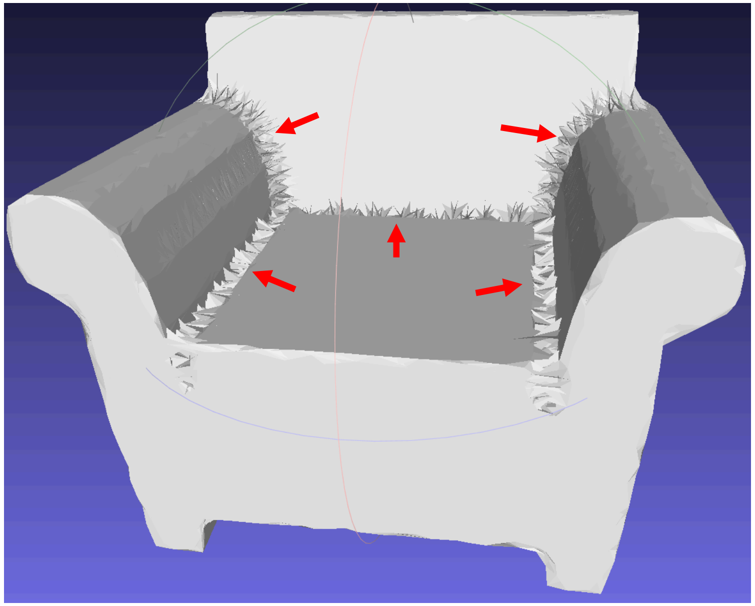
Results not bad…

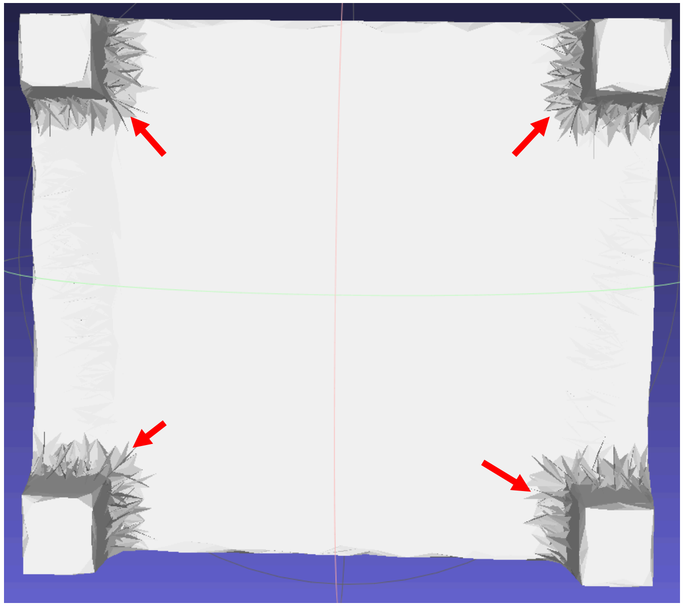


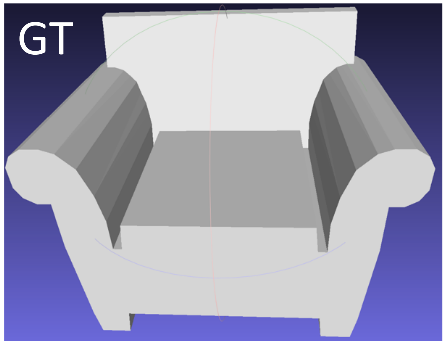
$K = 10$



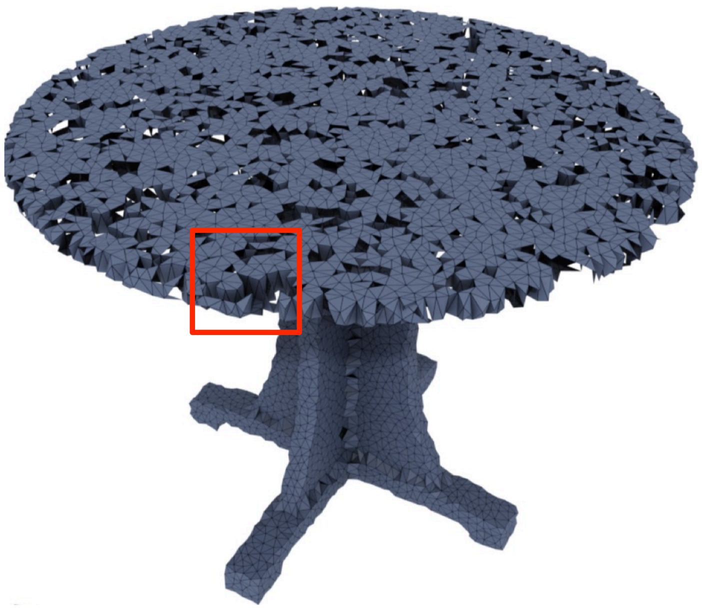
But has issues at boundaries

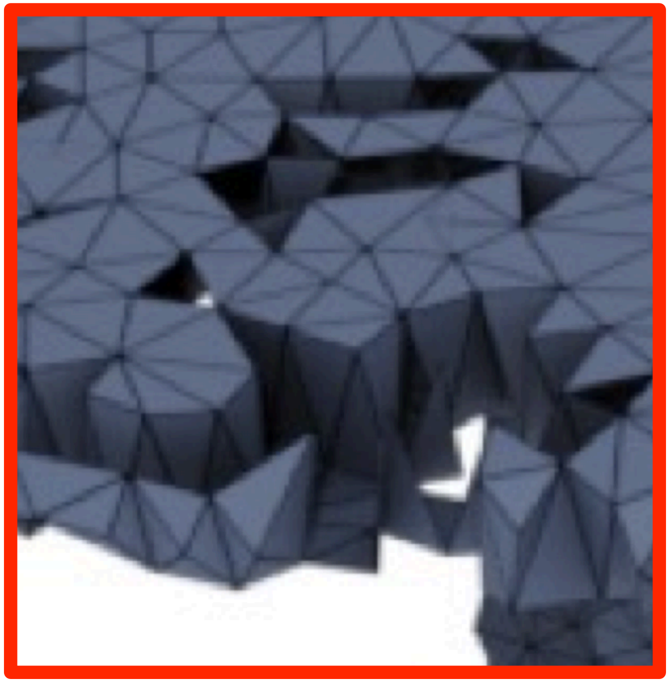






and some bad triangles





- Triangles formed by connecting the upper surface and the bottom surface of the tabletop (intrinsically far)

and some bad triangles

Q: Do we have a certificate of good triangles?

- Triangles formed by connecting the upper surface and the bottom surface of the tabletop (intrinsically far)

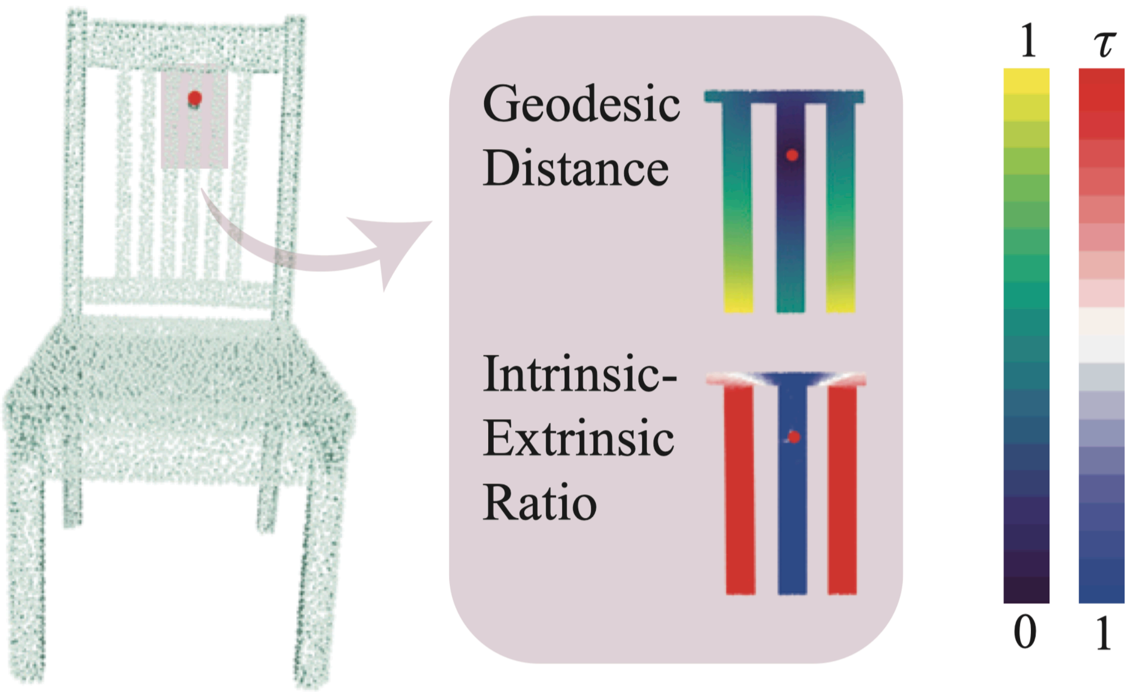
Intrinsic-Extrinsic Ratio

Given two vertices u, v on the surface, the intrinsic-extrinsic ratio is defined as:

$$\text{IER}(u, v) = \frac{d_G(u, v)}{d_E(u, v)}$$

Meshing Point Clouds with Predicted Intrinsic-Extrinsic Ratio Guidance, Liu et al, ECCV2020

Visualization



Meshing Point Clouds with Predicted Intrinsic-Extrinsic Ratio Guidance, Liu et al, ECCV2020

Intrinsic-Extrinsic Ratio for Triangles

For a triangle $\triangle\, uvw$:

$$\text{IER}(\triangle uvw) = \frac{d_G(u,v) + d_G(v,w) + d_G(w,u)}{d_E(u,v) + d_E(v,w) + d_E(w,u)}$$

If point u, v, w is **close enough**, IER($\triangle uvw) \leq 1 + \varepsilon$
$\Rightarrow$ (very likely) Triangle is on the surface

Meshing Point Clouds with Predicted Intrinsic-Extrinsic Ratio Guidance, Liu et al, ECCV2020

Intrinsic-Extrinsic Ratio for Triangles

For a triangle $\triangle\, uvw$:

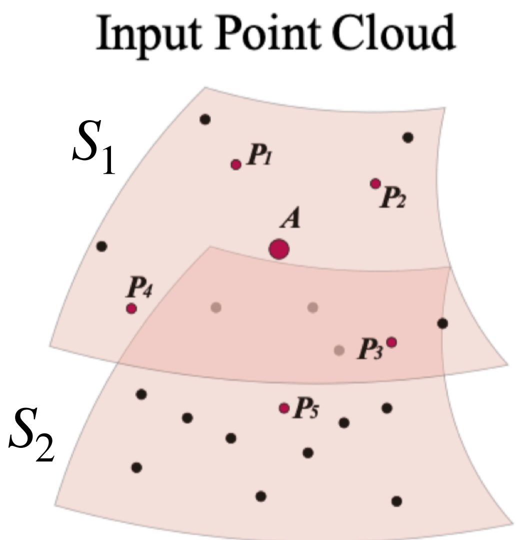
$$\mathrm{IER}(\triangle uvw) = \frac{d_G(u,v) + d_G(v,w) + d_G(w,u)}{d_E(u,v) + d_E(v,w) + d_E(w,u)}$$

If point u, v, w is **close enough**, $\boxed{\mathrm{IER}(\triangle uvw) \leq 1 + \varepsilon}$
$\Rightarrow$ (very likely) Triangle is on the surface

We learn to predict this quantity!

Meshing Point Clouds with Predicted Intrinsic-Extrinsic Ratio Guidance, Liu et al, ECCV2020

Meshing Point Clouds with Intrinsic-Extrinsic Ratio Guidance

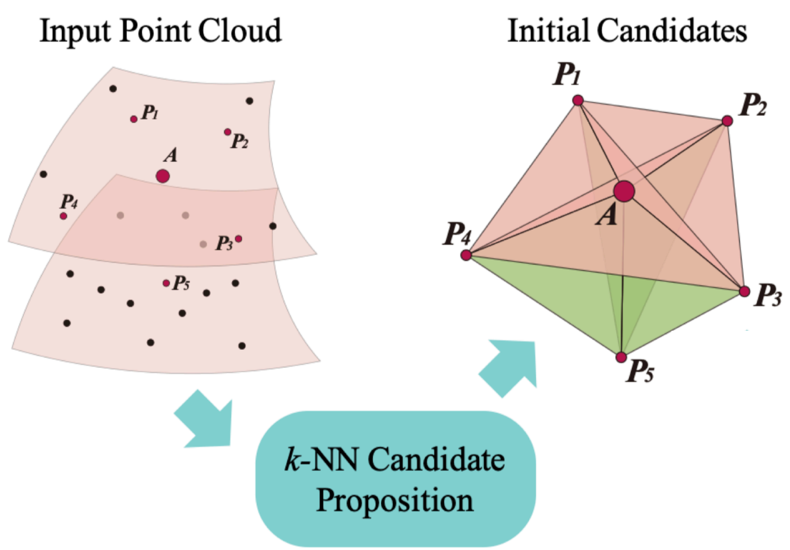


Case Study:

- Input point cloud is sampled from two closed thin surfaces S_1 and S_2
- $\{P_i\}$ are Euclidean nearest neighbors of A
  - $\{A, P_1, P_2, P_3, P_4\} \subset S_1$
  - $P_5 \in S_1$

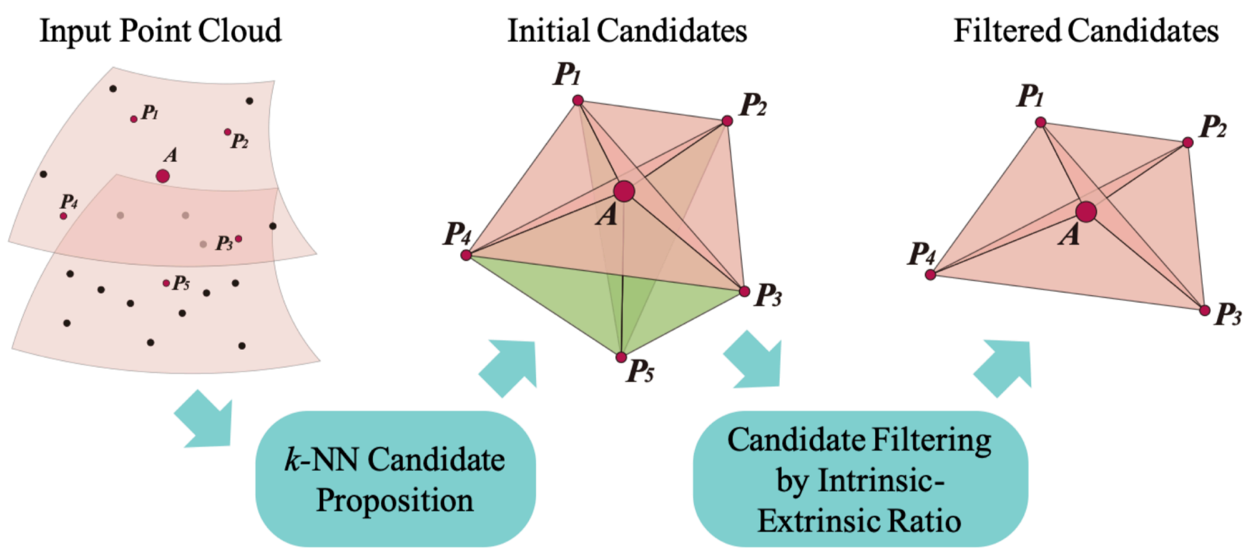
Meshing Point Clouds with Predicted Intrinsic-Extrinsic Ratio Guidance, Liu et al, ECCV2020

Meshing Point Clouds with Intrinsic-Extrinsic Ratio Guidance



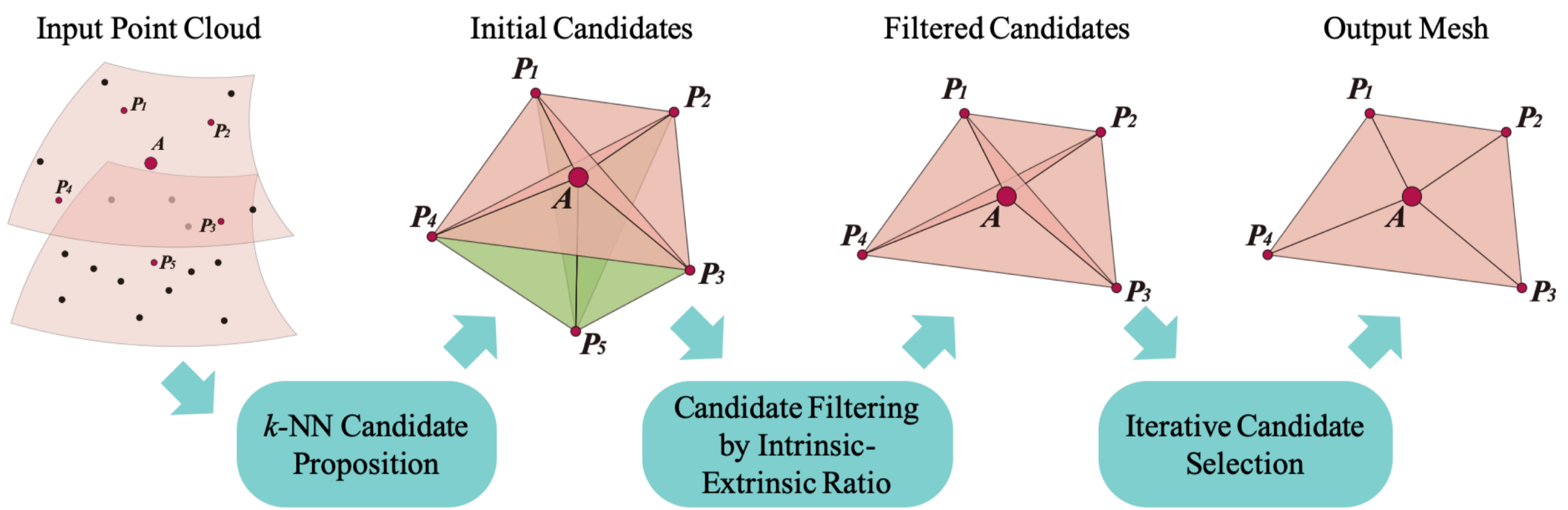
Meshing Point Clouds with Predicted Intrinsic-Extrinsic Ratio Guidance, Liu et al, ECCV2020

Meshing Point Clouds with Intrinsic-Extrinsic Ratio Guidance



Meshing Point Clouds with Predicted Intrinsic-Extrinsic Ratio Guidance, Liu et al, ECCV2020

Meshing Point Clouds with Intrinsic-Extrinsic Ratio Guidance

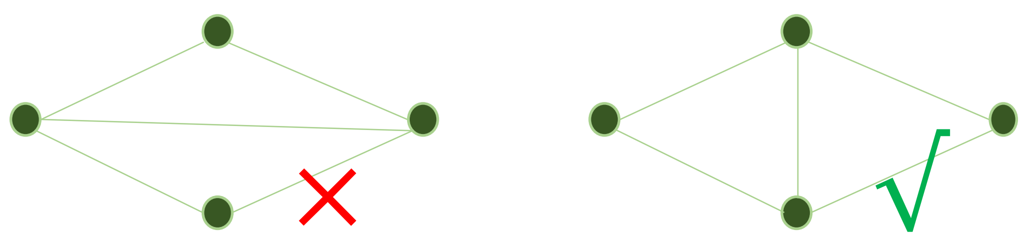


Meshing Point Clouds with Predicted Intrinsic-Extrinsic Ratio Guidance, Liu et al, ECCV2020

Iterative Candidate Selection

- The added triangle should not break manifold properties
  - No edge has more than two incident faces
  - No intersection between triangles

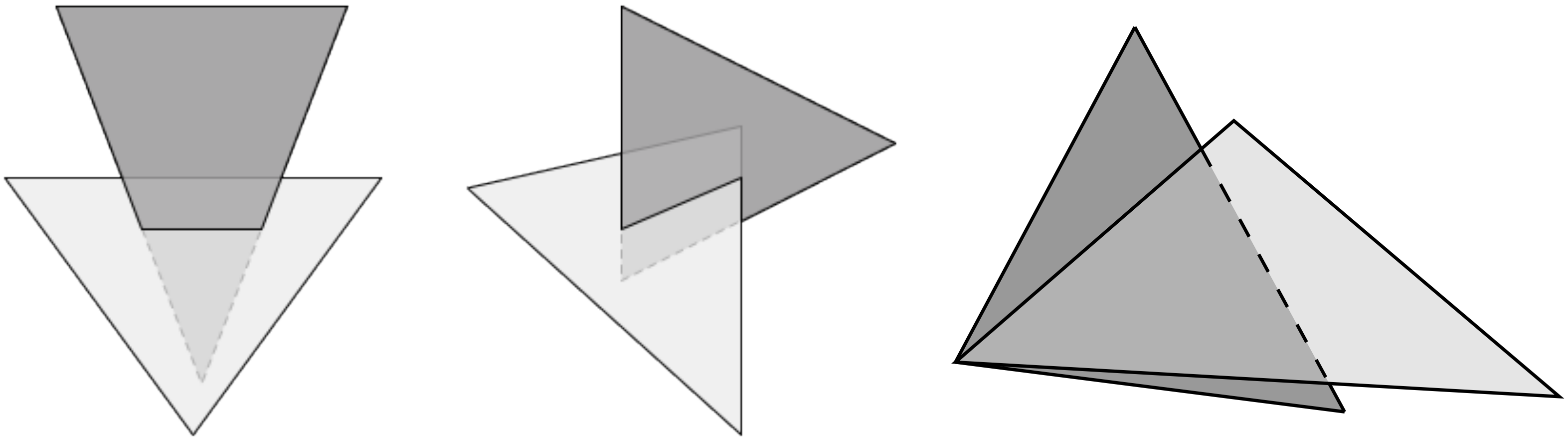
- Prefer equilateral triangles:



Measured by the ratio of longest edge to shortest edge

Meshing Point Clouds with Predicted Intrinsic-Extrinsic Ratio Guidance, Liu et al, ECCV2020

Triangle Collision Detection



Guigue P, Devillers O. Fast and robust triangle-triangle overlap test using orientation predicates. Journal of Graphics Tools. 2003 Jan 1;8(1):25-32.

Code: https://github.com/erich666/jgt-code/tree/master/Volume_08/Number_1/Guigue2003

Meshing Point Clouds with Predicted Intrinsic-Extrinsic Ratio Guidance, Liu et al, ECCV2020

Read by yourself

Meshing Results

k-NN

k-NN
with IER

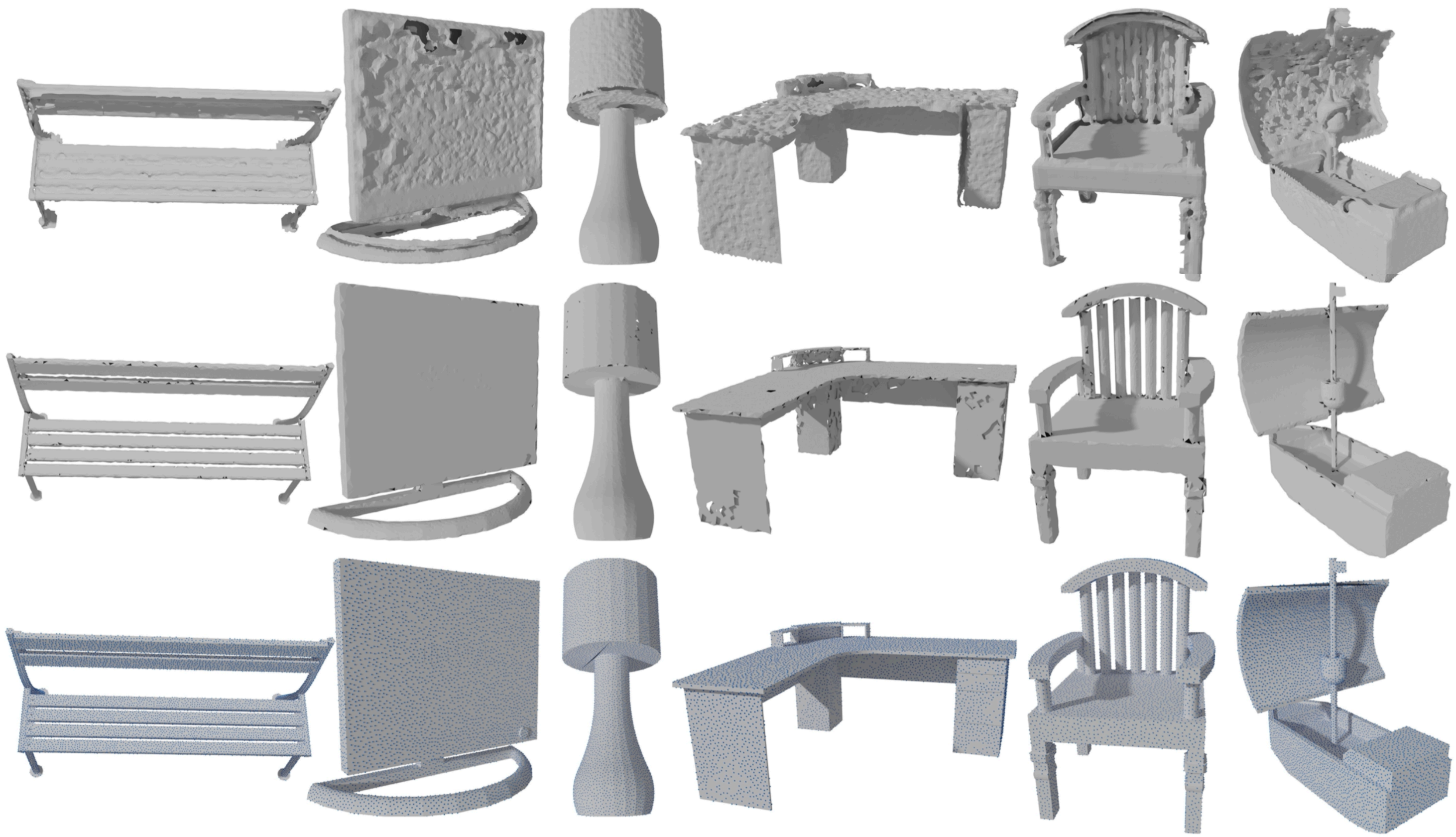
Meshing Point Clouds with Predicted Intrinsic-Extrinsic Ratio Guidance, Liu et al, ECCV2020

Robustness

Poisson

k-NN
with IER

Ground-
Truth



Meshing Point Clouds with Predicted Intrinsic-Extrinsic Ratio Guidance, Liu et al, ECCV2020