

3D Point Clouds

Lecture 7 – Feature Detection

主讲人 黎嘉信

Aptiv 自动驾驶
新加坡国立大学 博士
清华大学 本科





1. Introduction to **Features/Keypoints**



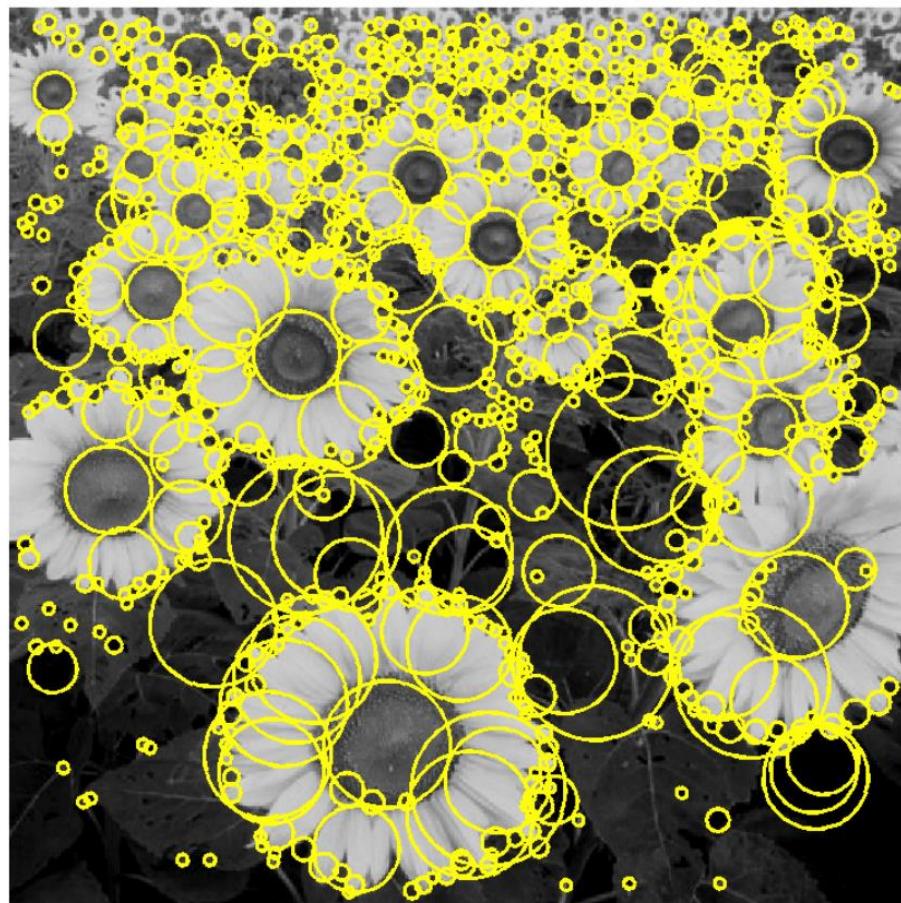
2. **Detector** – Classical and Modern



3. **Descriptor** – Classical and Modern

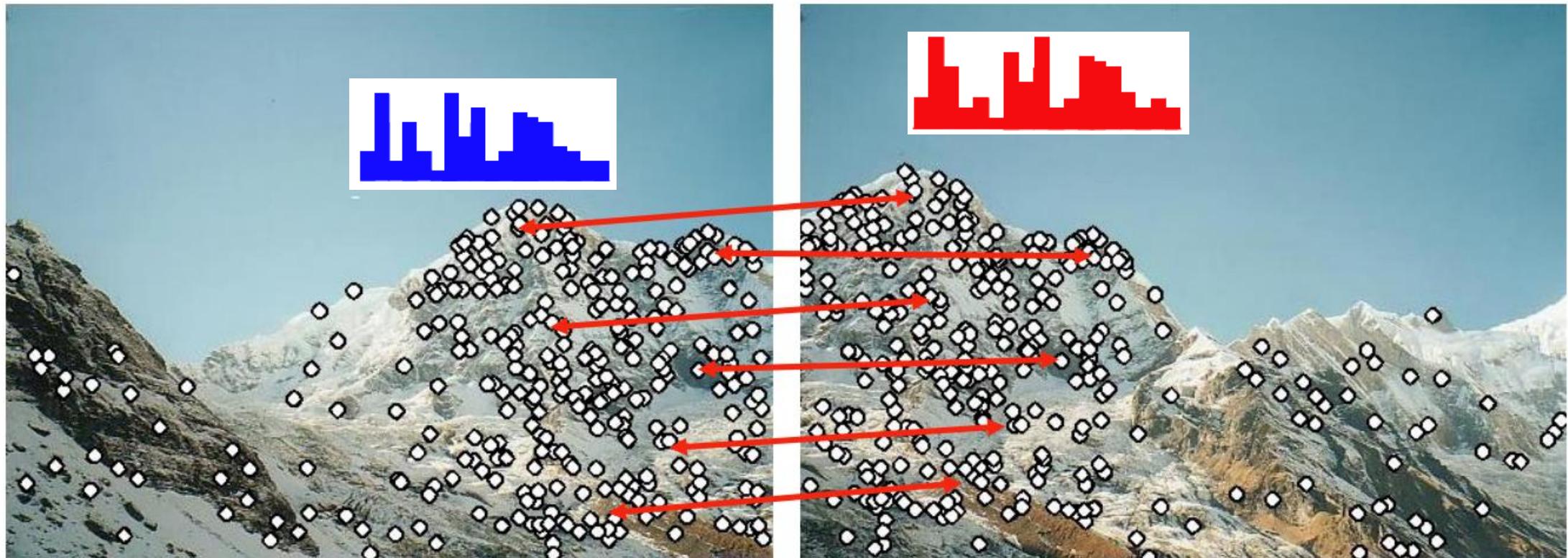


Image Features – Points / Blobs



Feature Detection: Identify the interest points

Source: Computer Vision, Raquel Urtasun



Feature Description: Extract a **vector** around the feature point to **descript it**.

Feature Matching: **Determine correspondence between descriptors**



Detection: Identity the interest points

Description: Extract a vector around the feature point to describe it.

Matching: Determine correspondence between descriptors

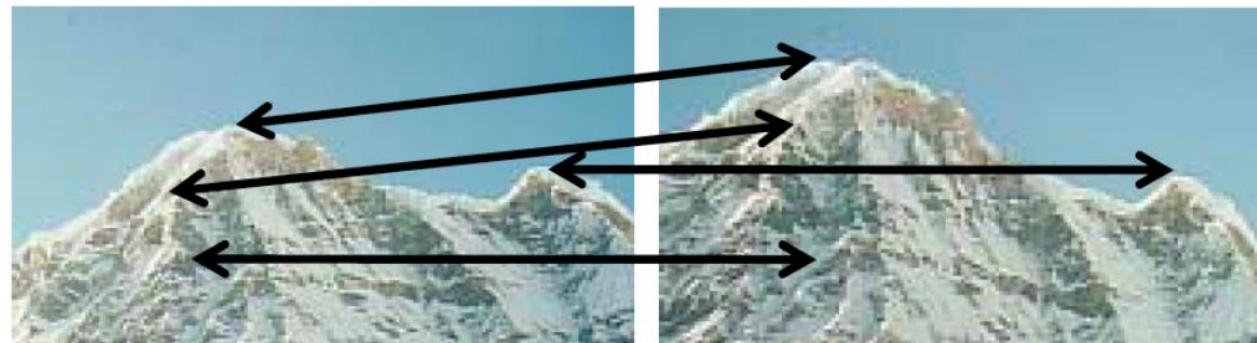
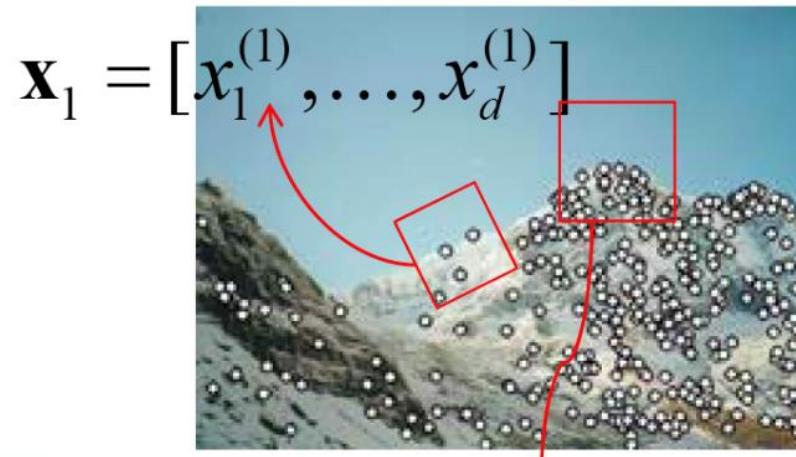
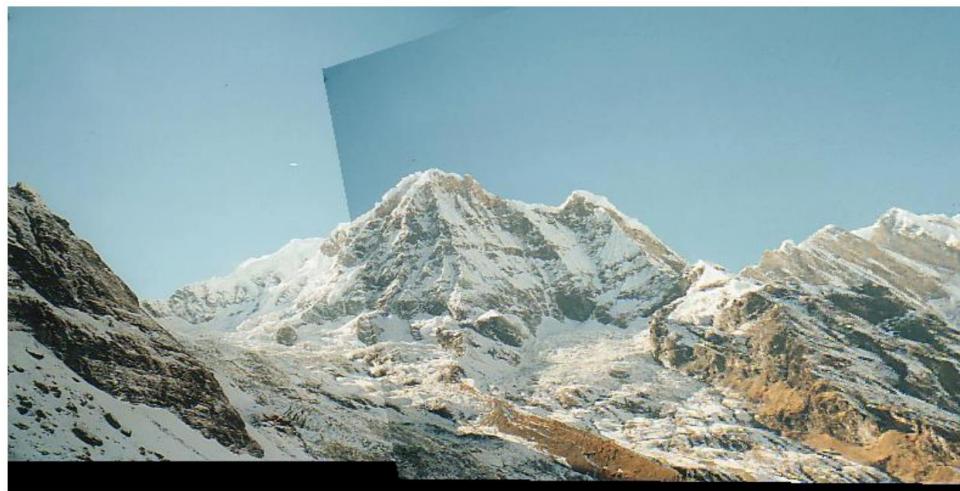
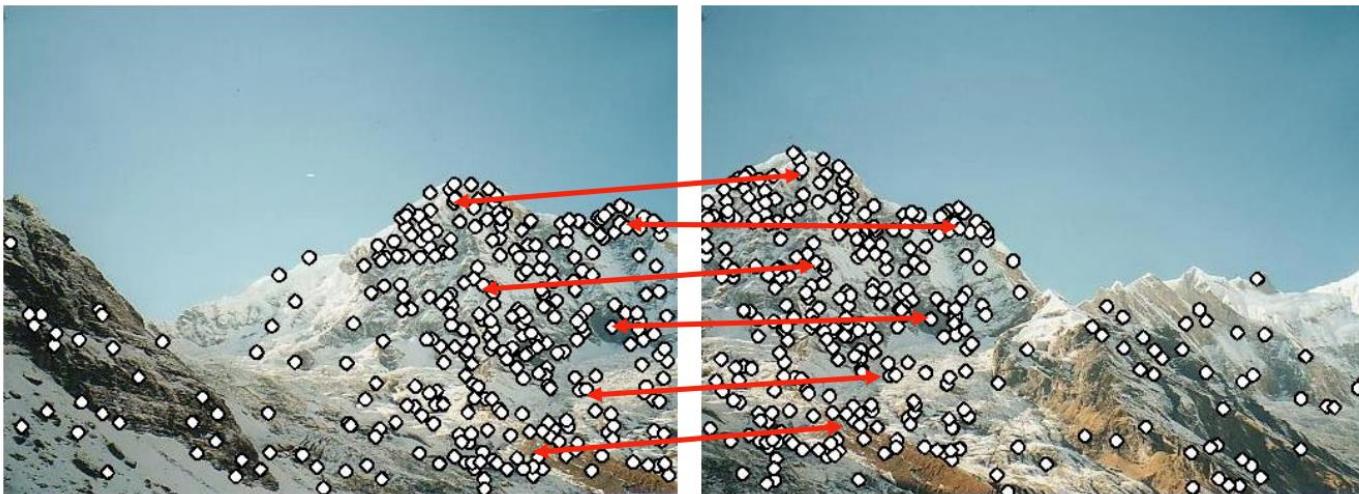




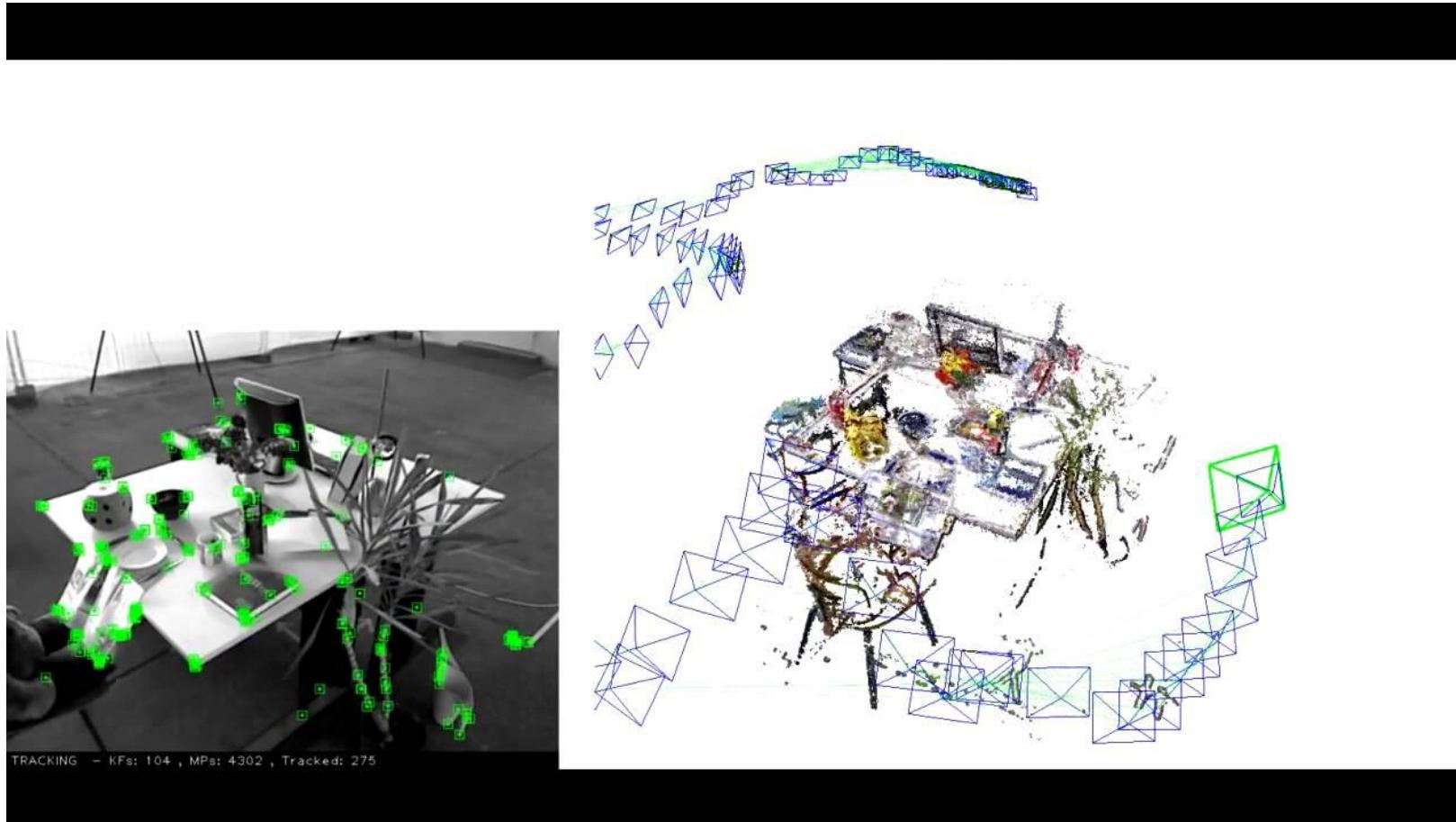
Image Features – Applications – Panorama



Source: Computer Vision, Raquel Urtasun



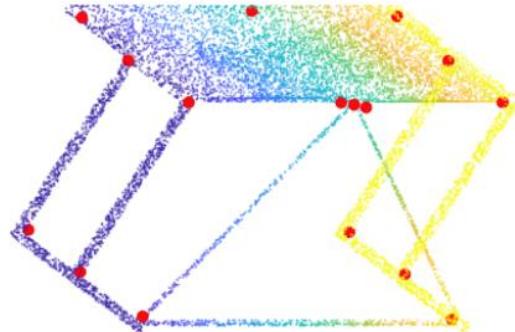
Image Features – Applications – SLAM (Simultaneous Localization and Mapping)



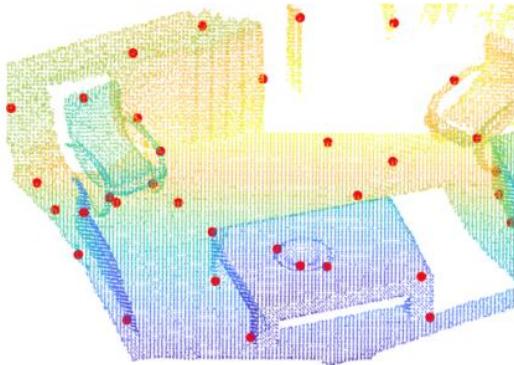
Source: ORB SLAM, <https://webdiis.unizar.es/~raulmur/orbslam/>



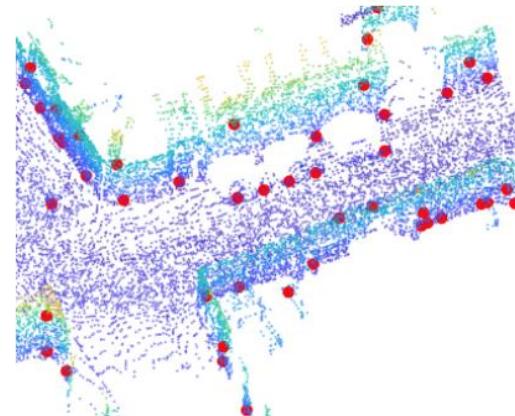
Point Cloud Features



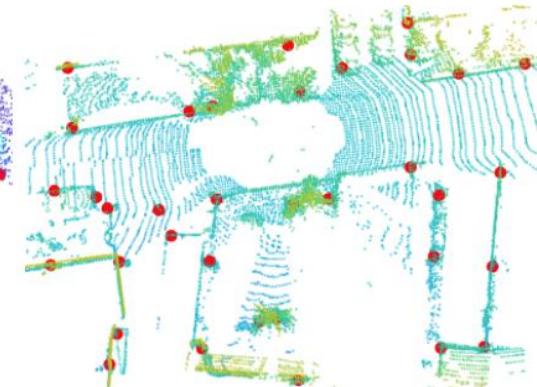
CAD Model



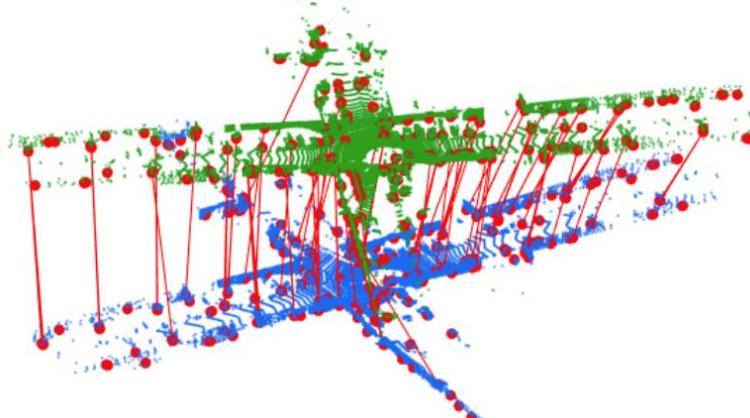
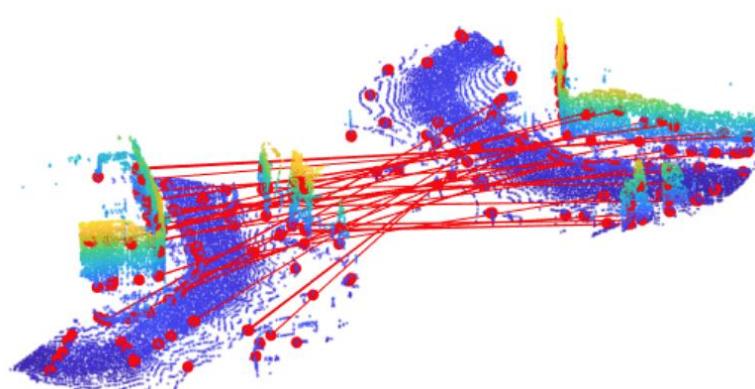
RGB-D



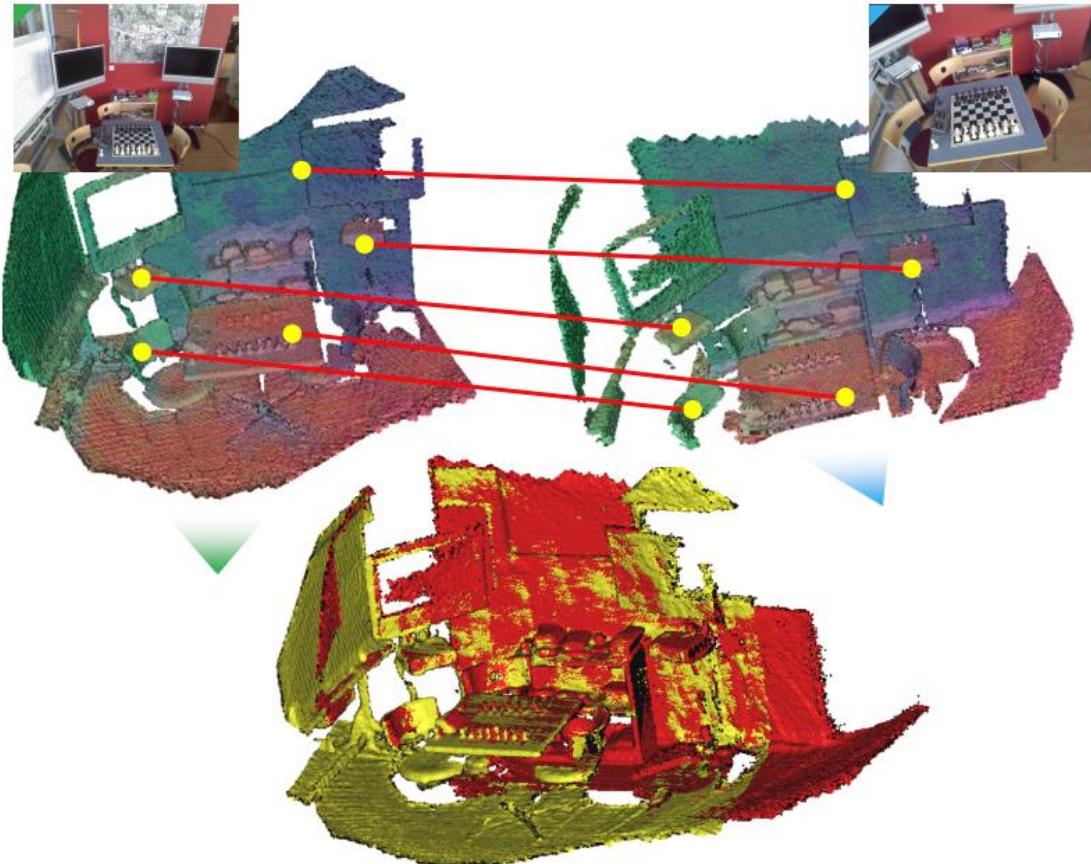
Oxford RobotCar



KITTI



Description and Matching

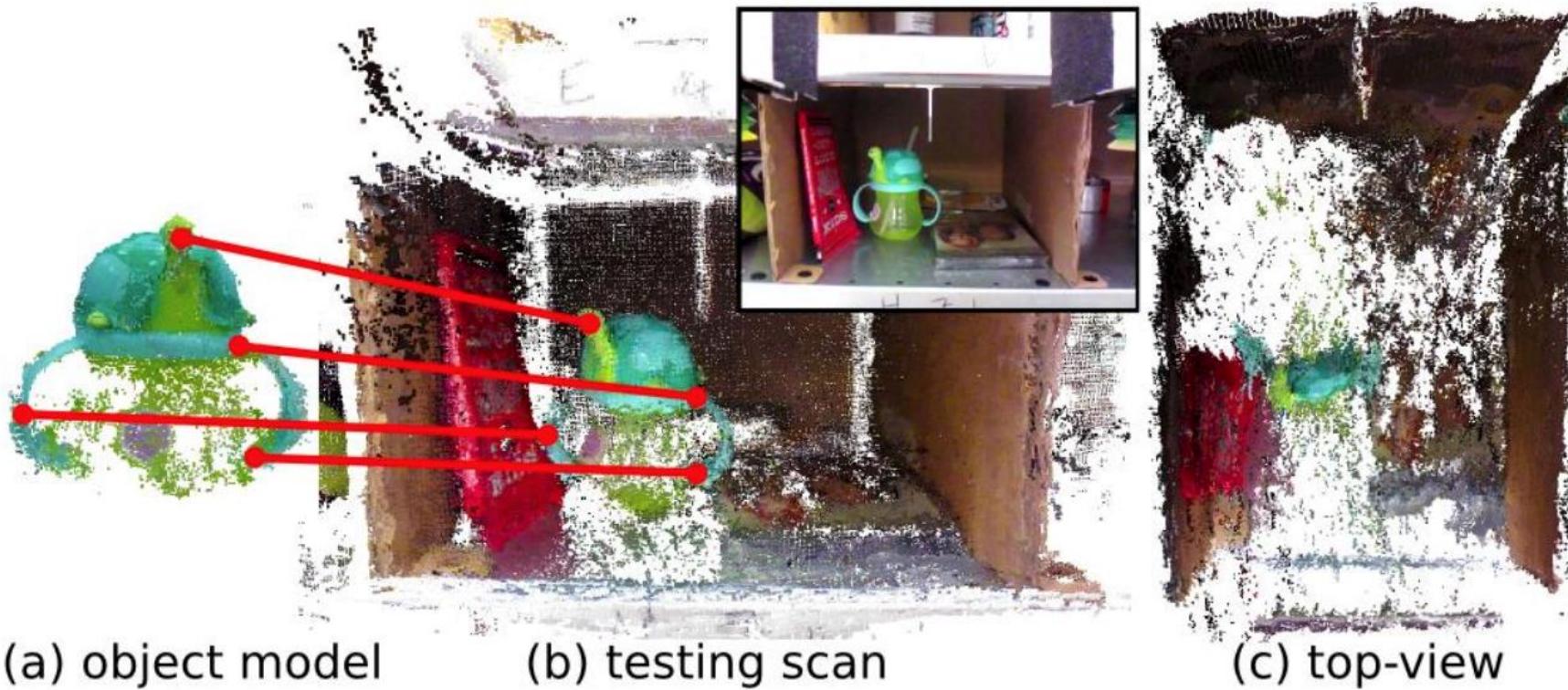


Registration

- Find a transform to align two point clouds
- A transform consists of
 - Rotation R
 - Translation t
- Method 1 - **Iterative Closest Point (ICP)?**
 - ICP requires proper **initial guess**
 - Low overlapping ratio 需要足够多的重合率, 不然很难配准
- Method 2 – **Detect and match features**
 - **No initialization required**
 - **Works for low overlapping ratio**



Point Cloud Features – Object 6D Pose



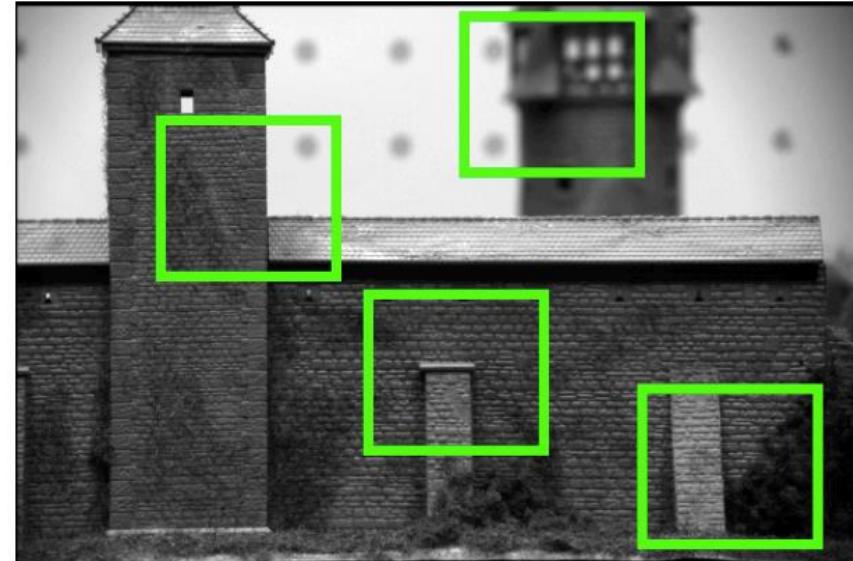
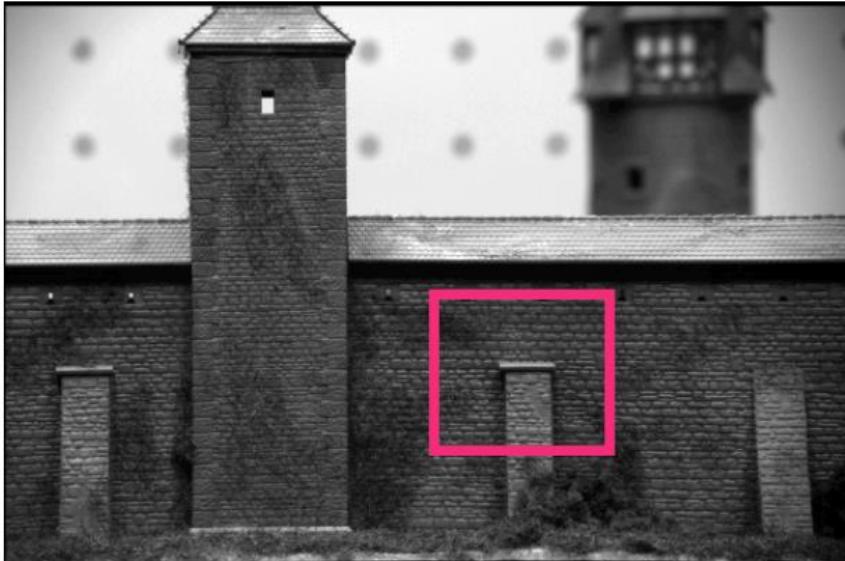
- In fact, it is still registration



- Borrow ideas from **image features**
 - **Harris**
 - SUSAN (Small Univalue Segment Assimilating Nucleus)
 - **SIFT (Scale-Invariant Feature Transform)**
- Native in 3D geometry
 - **ISS (Intrinsic Shape Signatures)**
- Deep learning
 - Very few methods available



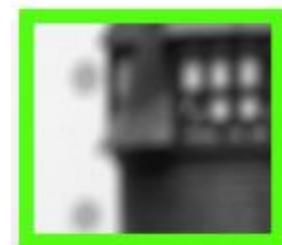
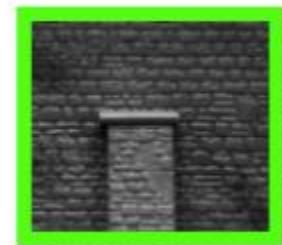
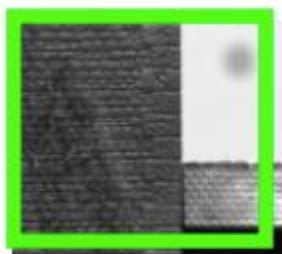
Harris Detector – Patch Matching



- Find the most similar patch in the second image.

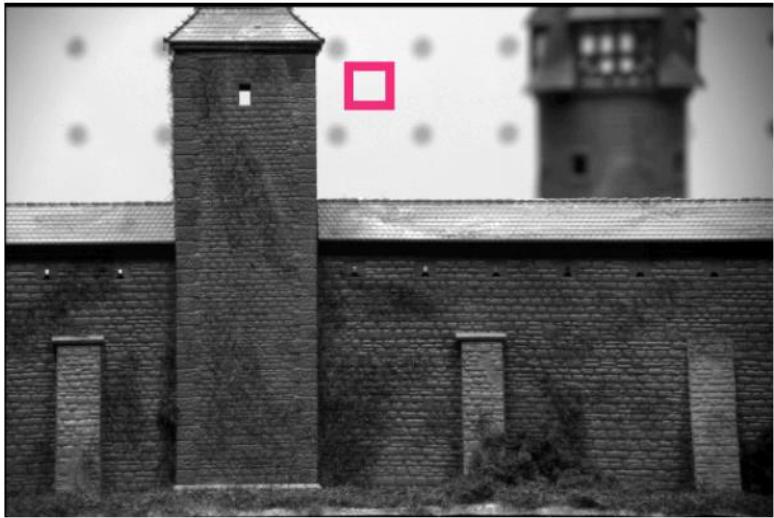


?
=





Harris Detector – Patch Matching



animals

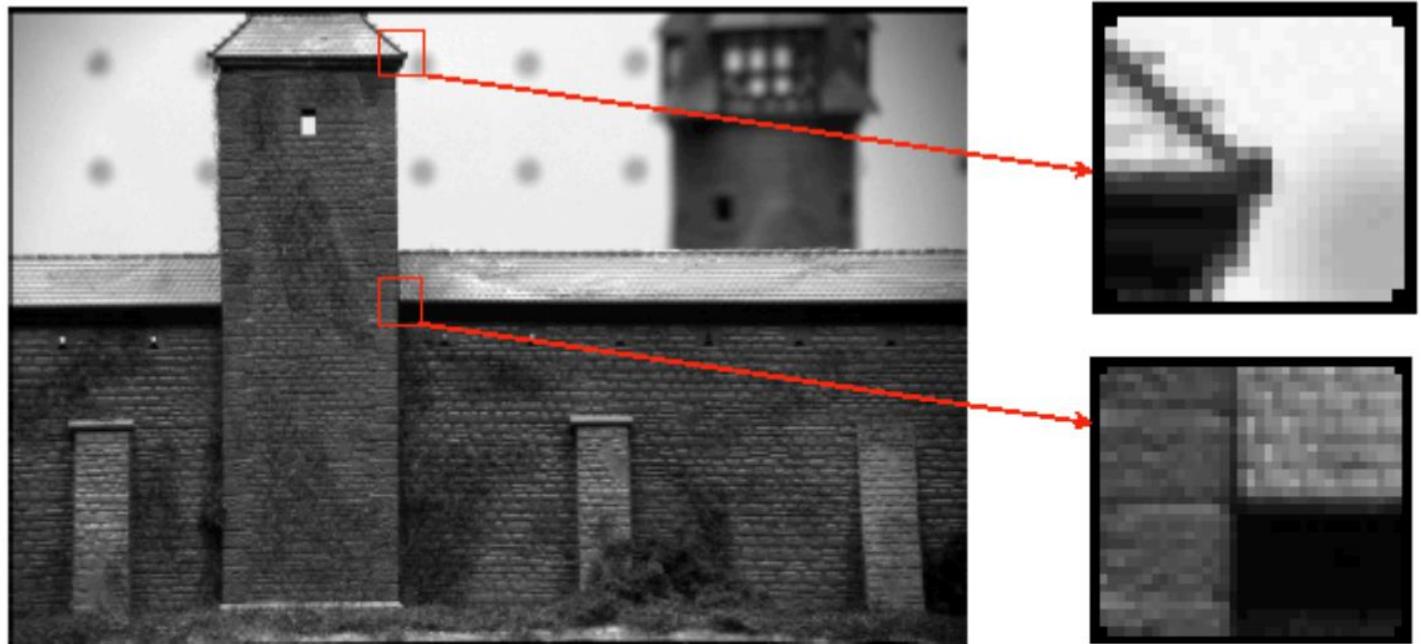
- Not all **patches** are **created equal**. Some are more equal than others.
-- Animal Farm, George Orwell





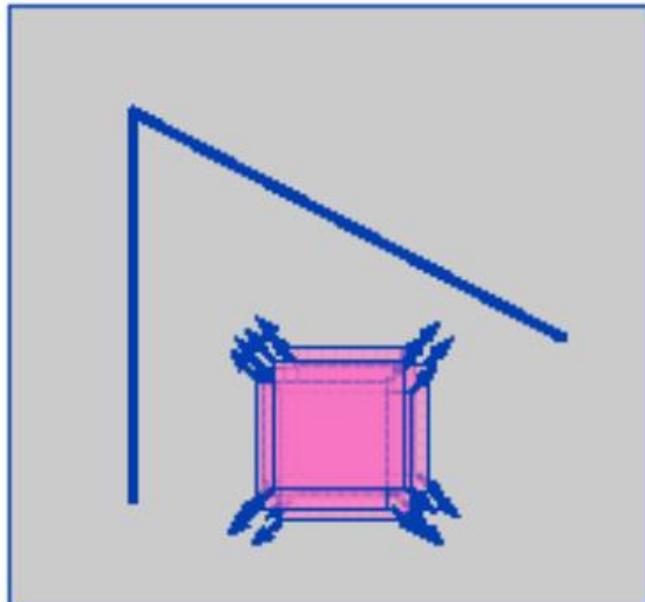
Harris Detector – Corner Points

- What are corners?
 - Junctions of contours
- Corners are distinctive.
 - Large variations in neighborhood
 - Stable to viewpoint change
 - Good features!

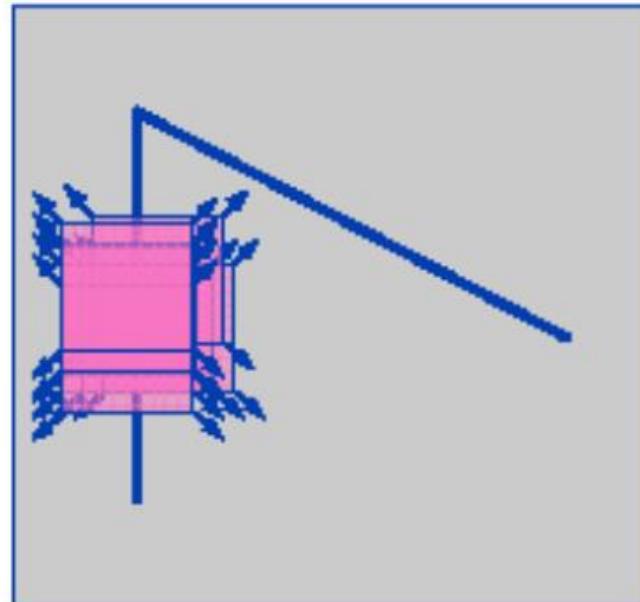




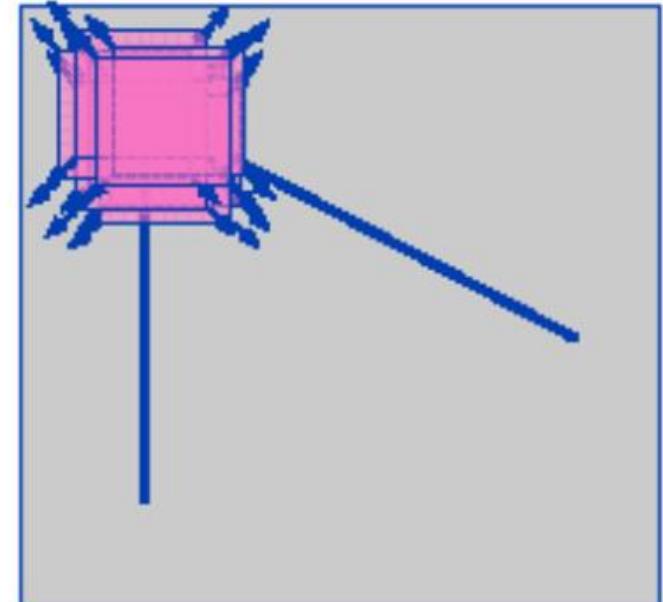
Corner – Basic Idea



“flat”:
no change in all directions



“edge”:
no change along the edge
direction



“corner”:
**significant change in all
directions**



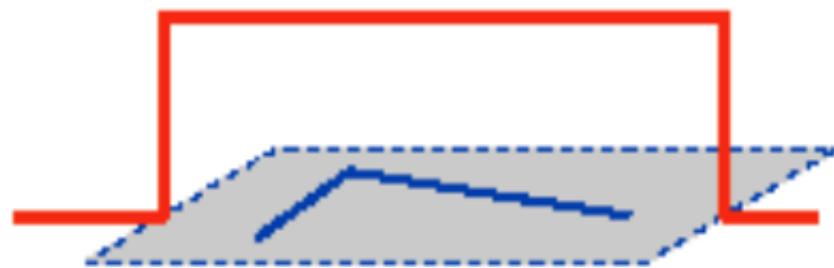
- Given an image I , consider a patch $x, y \in \Omega$
- Shift the patch by $[u, v]$, the intensity change is:

$$E(u, v) = \sum_{x, y \in \Omega} w(x, y)[I(x + u, y + v) - I(x, y)]^2$$

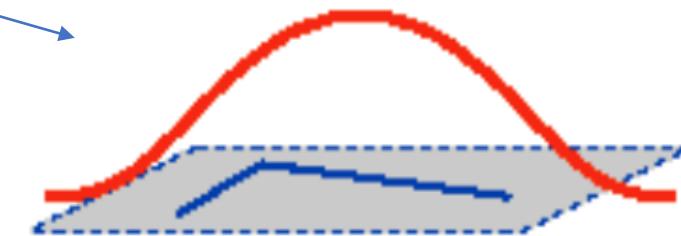
Windows
function

Shifted
intensity

Intensity



1 in window, 0 outside



Gaussian



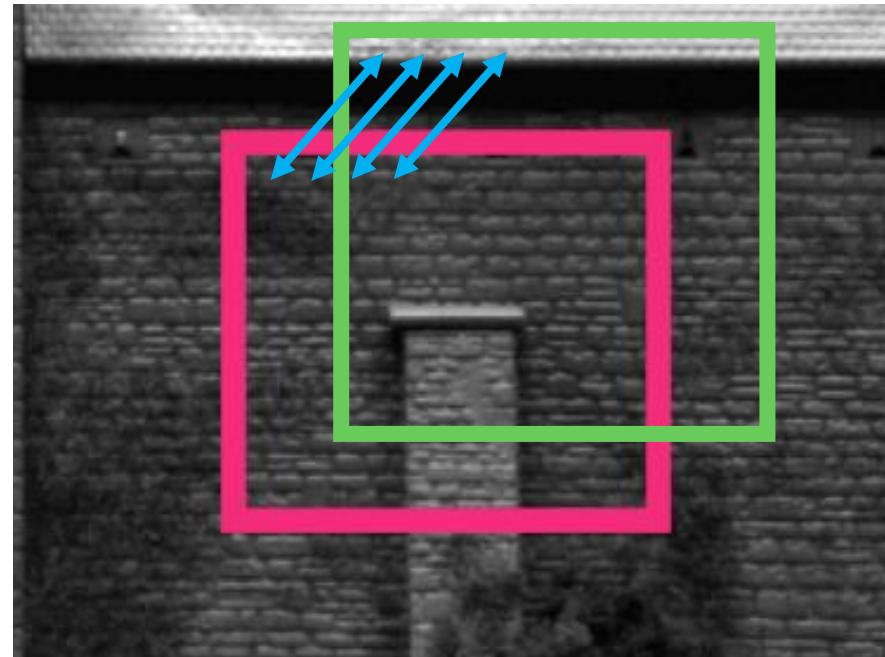
$$E(u, v) = \sum_{x,y \in \Omega} w(x, y)[I(x + u, y + v) - I(x, y)]^2$$

u, v is fixed

x, y is loop over the patch

How to select u, v ?

As small as possible \rightarrow gradient





$$f(x + u, y + v) = f(x, y) + uf_x(x, y) + vf_y(x, y)$$

First order partial derivatives

$$+ \frac{1}{2!} [u^2 f_{xx}(x, y) + uv f_{xy}(x, y) + v^2 f_{yy}(x, y)]$$

Second order partial derivatives

$$+ \frac{1}{3!} [u^3 f_{xxx}(x, y) + u^2 v f_{xxy}(x, y) + uv^2 f_{xyy}(x, y) + v^3 f_{yyy}(x, y)]$$

Third order partial derivatives

+ ... (*higher order terms*)

First order approximation: $f(x + u, y + v) \approx f(x, y) + uf_x(x, y) + vf_y(x, y)$



Harris Corner – Derivation

- Intensity change for a patch when shifted $[u, v]$ is represented by

$$\sum_{x,y \in \Omega} w(x, y) [I(x + u, y + v) - I(x, y)]^2$$

- Let's assume $w(x, y)$ is the binary windows function, i.e, $w(x, y) = 1, \forall x, y \in \Omega$

$$E(u, v) = \sum_{x,y \in \Omega} (I(x + u, y + v) - I(x, y))^2$$



- Intensity change is given by

$$E(u, v) \approx [u \ v] M \begin{bmatrix} u \\ v \end{bmatrix}, \quad M = \sum_{x,y \in \Omega} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

- M is the covariance matrix of the image gradient

- Denote intensity gradient for location $[x_i, y_i]$ as $I_i = [I_{x_i}, I_{y_i}]^T \in \mathbb{R}^2$
- Covariance matrix $M = \sum_i I_i I_i^T$
- However,
 - What does it actually mean?
 - How can we get feature points?



2D Derivatives

- Linear edge

- $I_i = [I_{x_i}, 0]^T$ Intensity 的一阶导

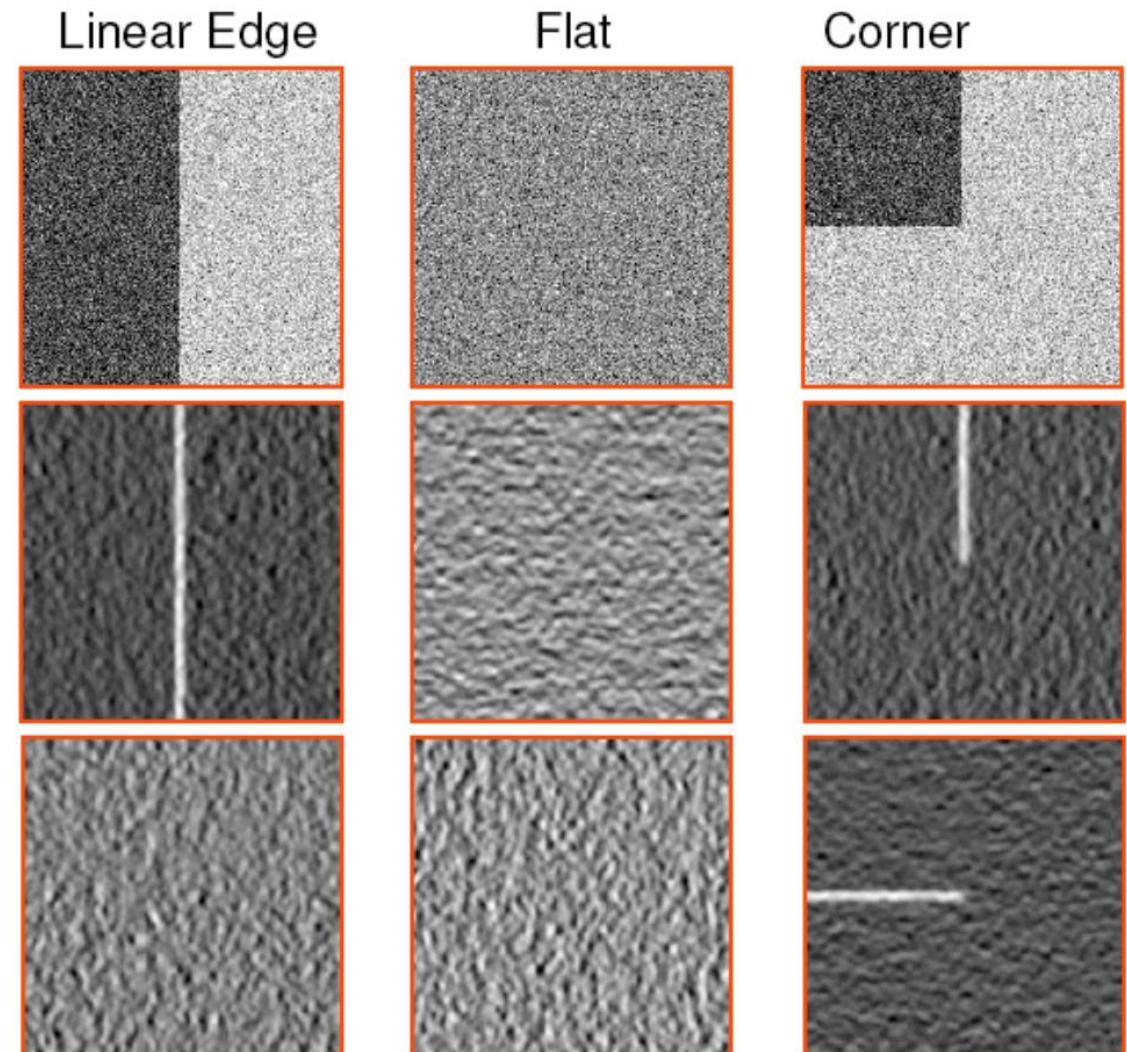
- Flat

- $I_i = [0,0]^T$

- Corner

- $I_i = [I_{x_i}, I_{y_i}]^T$

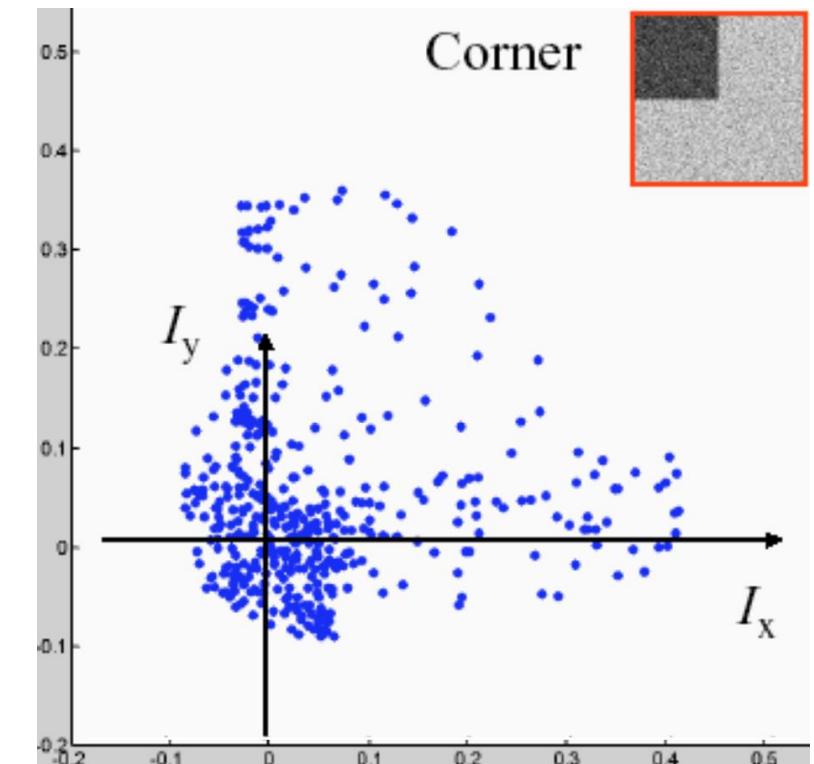
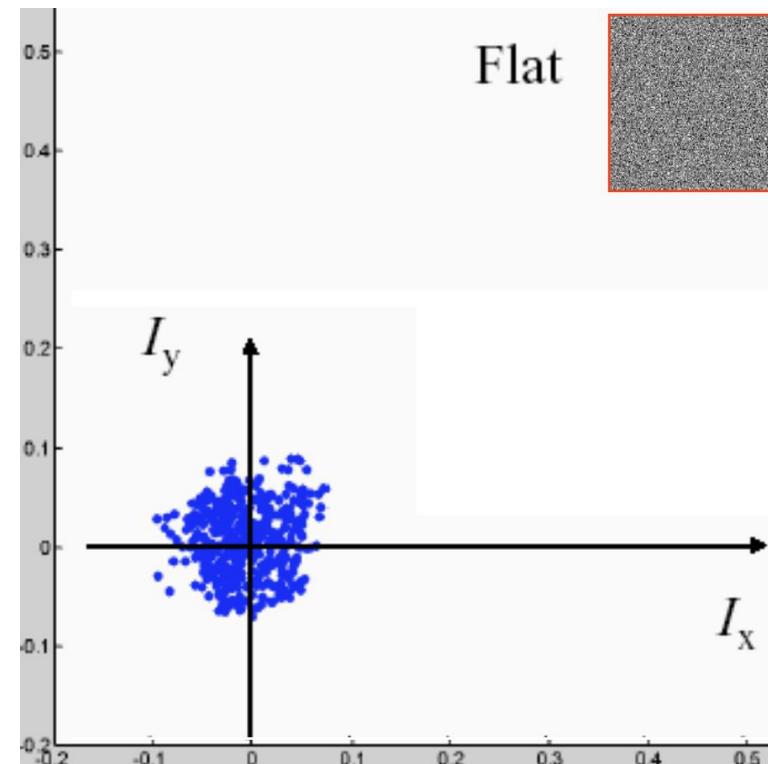
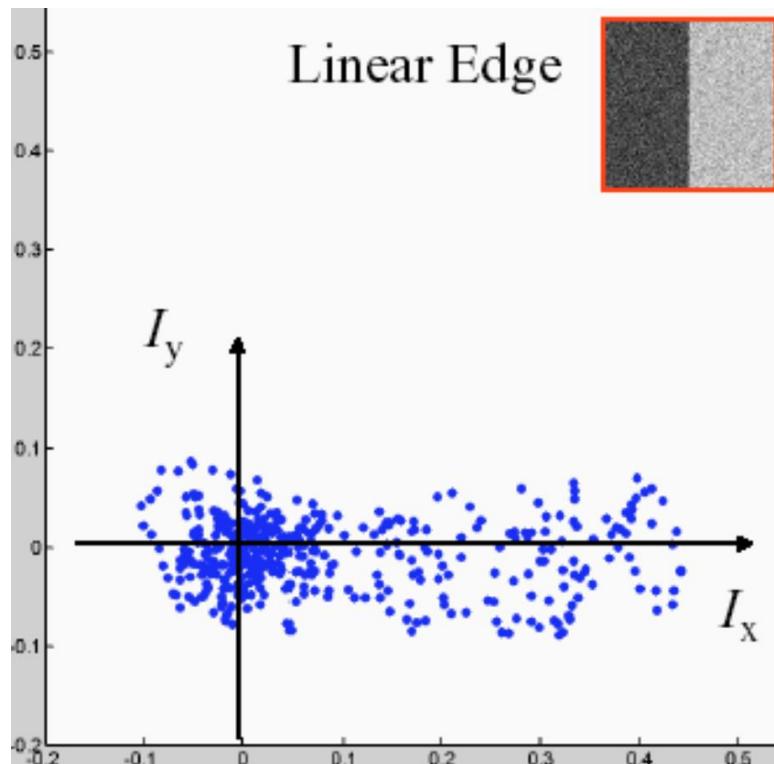
Y derivative X derivative Input image patch





Gradient I_i as points

- Plot the gradient $I_i = [I_{x_i}, I_{y_i}]^T$ for three case: Linear Edge, Flat, Corner



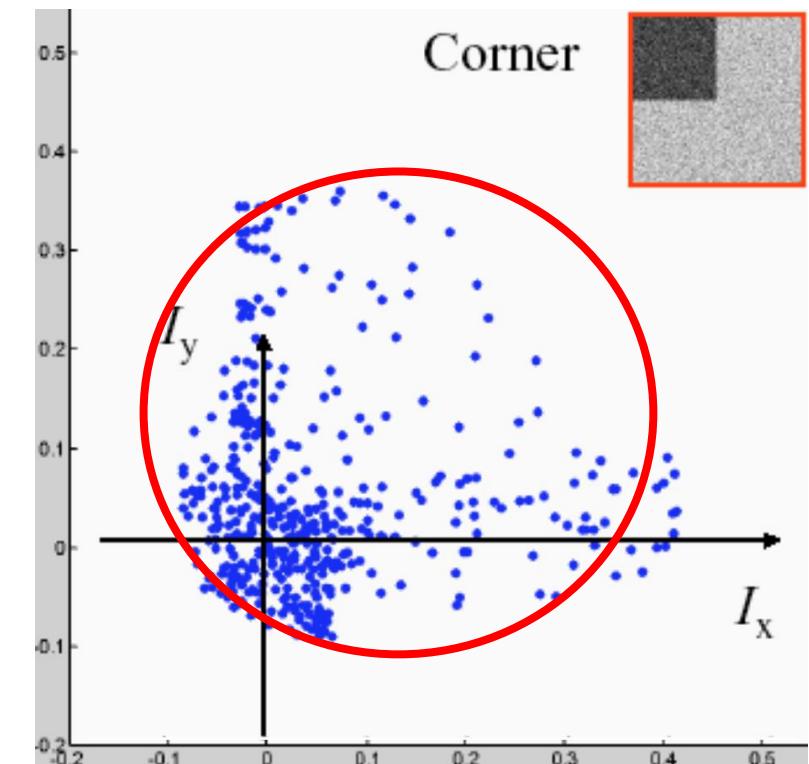
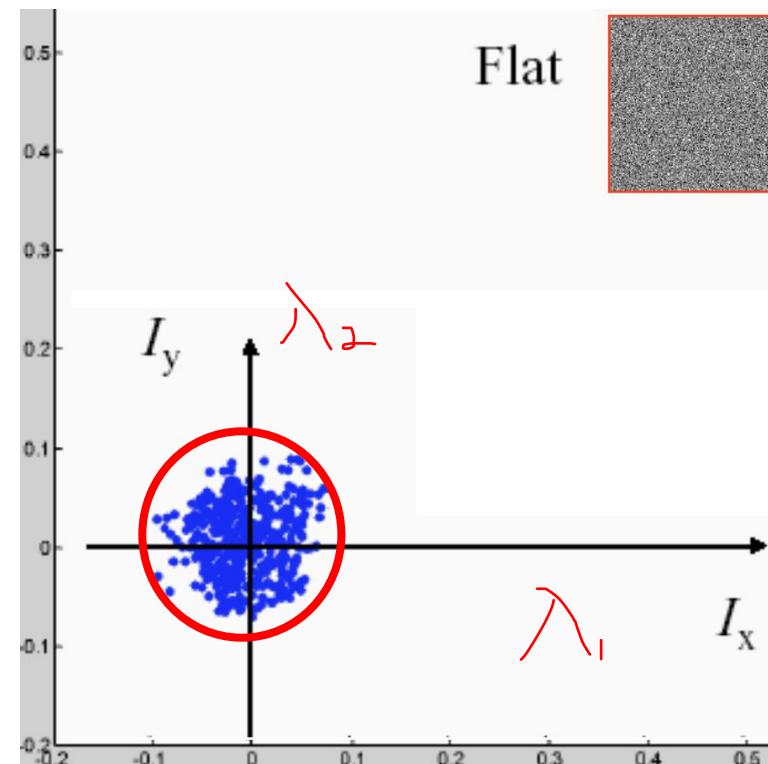
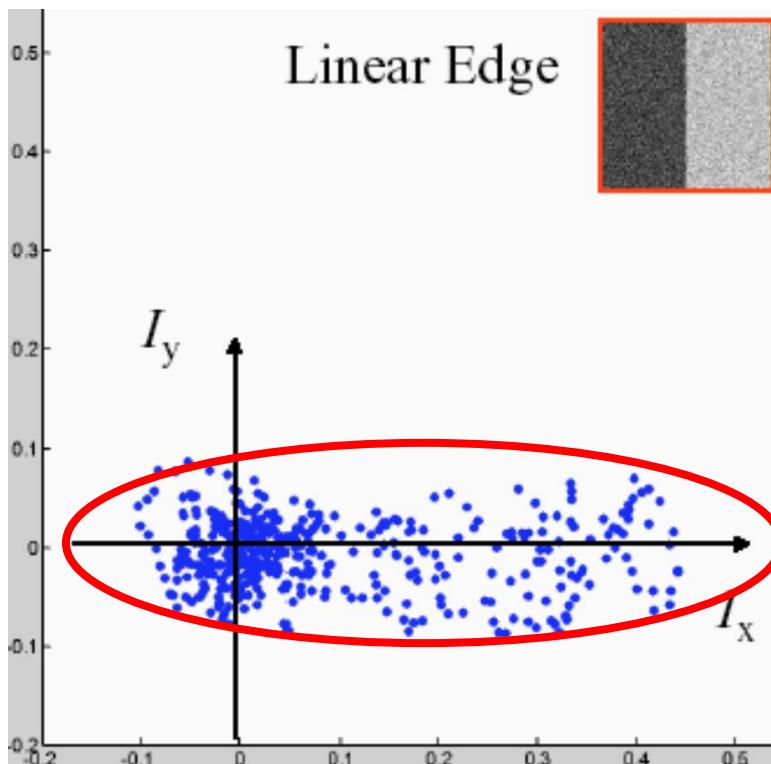


Gradient I_i as points

协方差矩阵代表了数据点在空间中是怎么分布的

缩放

- We can distinguish these 3 cases by covariance matrix M
- The eigenvalues of M are λ_1, λ_2





Feature Detection by Eigenvalues

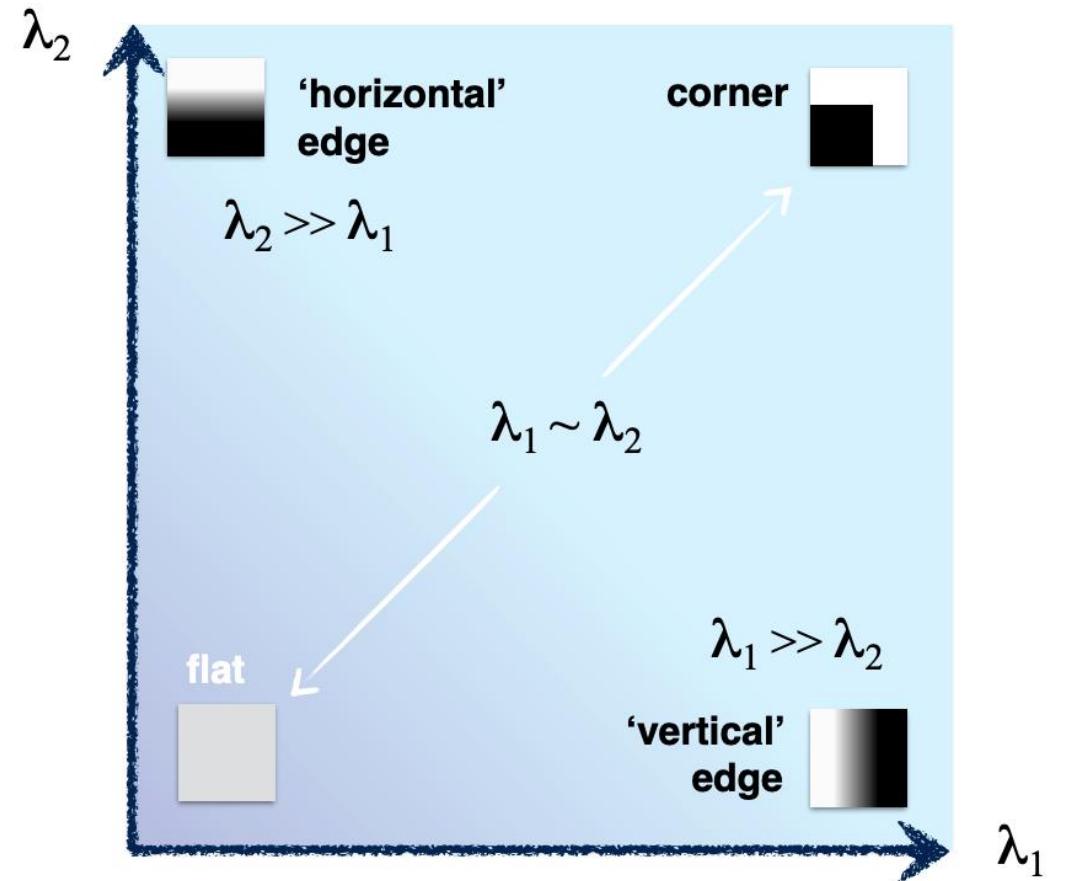
- The intensity change is

$$E(u, v) \approx [u \quad v] M \begin{bmatrix} u \\ v \end{bmatrix}, \quad M = \sum_{x, y \in \Omega} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

- We want **corner points**

- Large gradient along x $\rightarrow I_i$ spreads along x
- Large gradient along y $\rightarrow I_i$ spreads along y

- Corner points – both λ_1, λ_2 are large





Response Function R

- Harris & Stephens (1988): A Combined Corner and Edge Detector

$$R = \det M - k(\text{trace } M)^2,$$

$$\det M = \lambda_1 \lambda_2$$

$$\text{trace } M = \lambda_1 + \lambda_2$$

$k \in [0.04, 0.06]$ (empirically determined *constant*)

- Kanade & Tomasi (1994): Good Features to Track

$$R = \min(\lambda_1, \lambda_2)$$

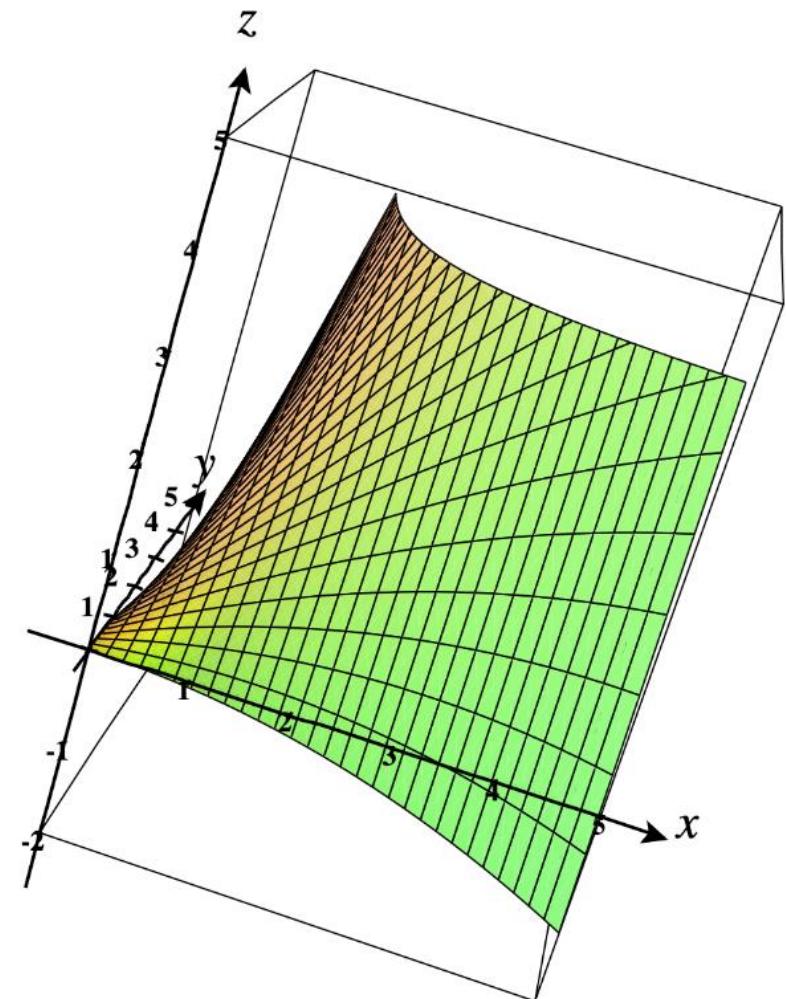
- Nobel (1998): Finding Corners.

$$R = \frac{\det M}{\text{trace}(M) + \epsilon}$$



Response Function R

- Set a threshold τ for R
 - Corner: $R \geq \tau$
- Harris Detector Algorithm
 - Compute R for every pixel in the image
 - Store result as a "response image"
 - Normalized the "response image" to e.g. $[0, 1]$
 - Empirically set a τ
 - Non-Maximum Suppression (NMS)



Plot of Harris response $R = \det M - k(\text{trace } M)^2$



Harris Corner Detector – Input Image

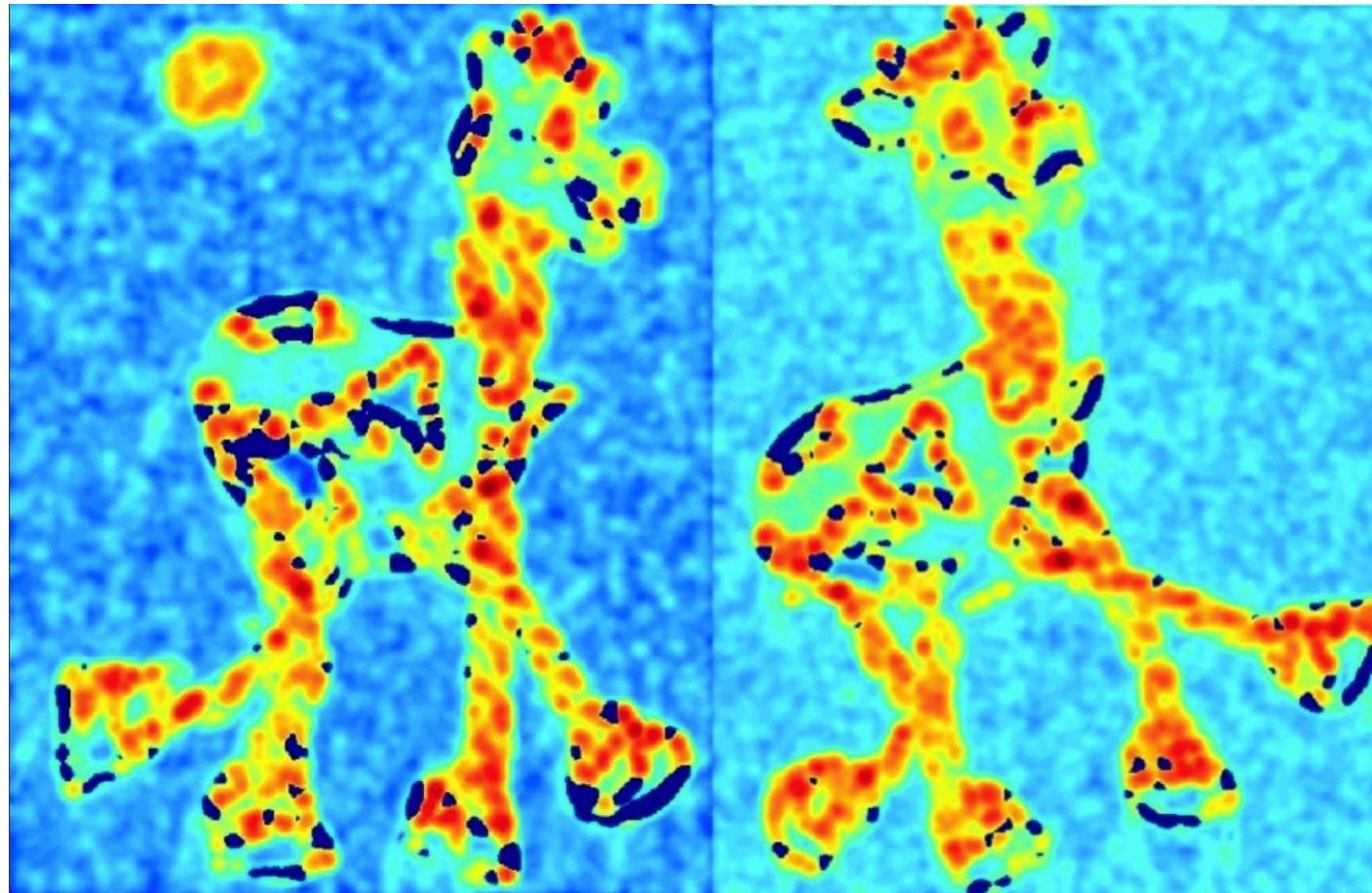


Source: 16-385 Computer Vision, CMU, Kris Kitani



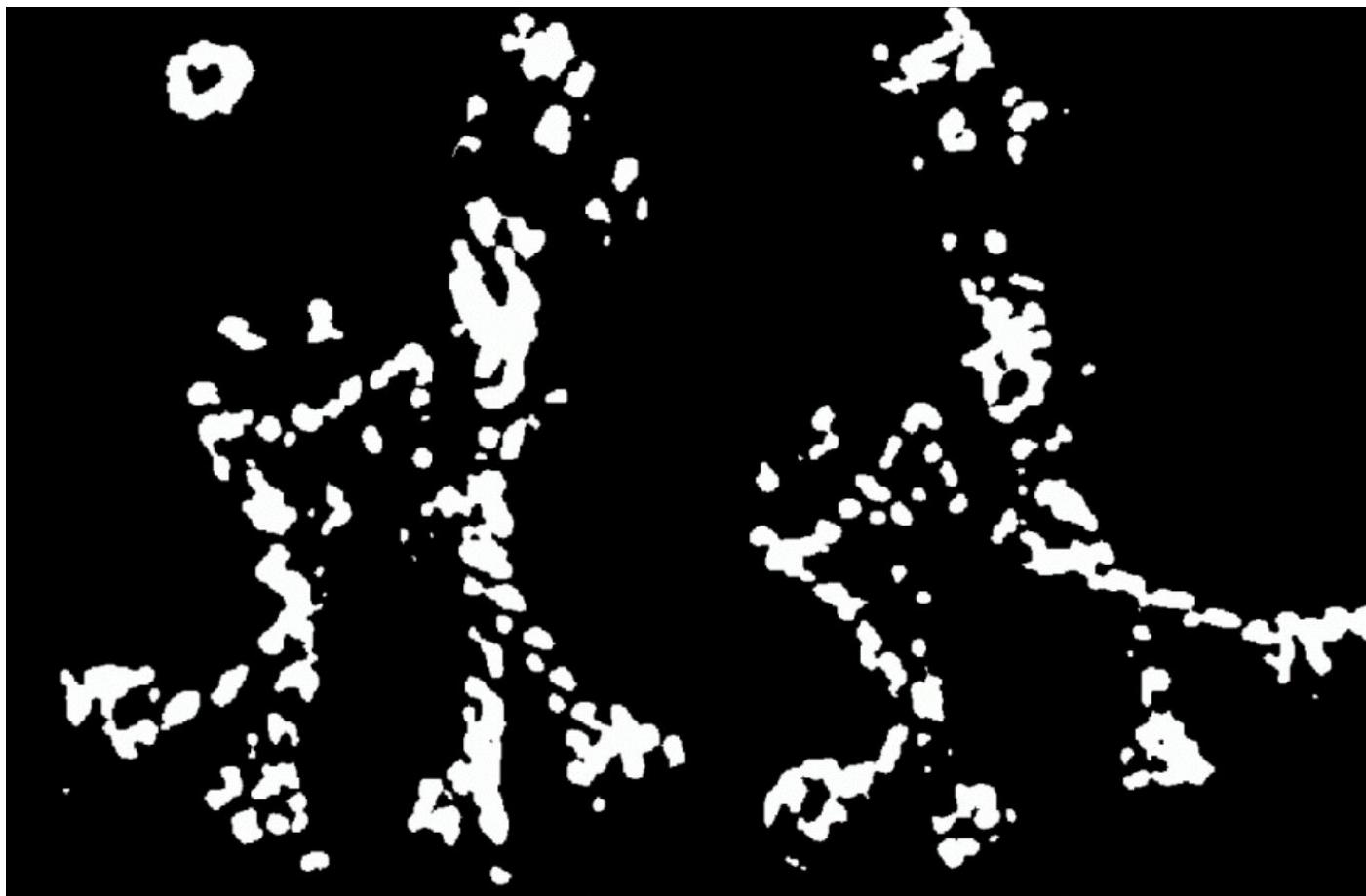
Harris Corner Detector – Corner Response

颜色越红代表这个点的 response 越大





Harris Corner Detector – Corner Response Thresholded



Source: 16-385 Computer Vision, CMU, Kris Kitani



Harris Corner Detector – NMS



Source: 16-385 Computer Vision, CMU, Kris Kitani



Harris Corner Detector – Detection Result



Source: 16-385 Computer Vision, CMU, Kris Kitani



- Harris Corner is based on **intensity change of a patch**
 - Move the patch, how much the intensity changes?

$$E(u, v) = [u \quad v] M \begin{bmatrix} u \\ v \end{bmatrix}, \quad M = \sum_{x,y \in \Omega} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Covariance of image gradient

- Assume there is intensity in the point cloud, how can we formulate **a cost function**?
 - **Points are discrete.** 点云里每个点是离散的, 不像图像里每个点组成方格
 - What is a “patch” in point cloud? **U V W**
 - How to “move a patch”?



Harris 3D with Intensity

- For a local (small) region Ω over a point
- Assume intensity is a continuous function

$$I(x, y, z): \mathbb{R}^3 \rightarrow \mathbb{R}, [x, y, z] \in \Omega$$

- Assume the small “move” is $[u, v, w]$
- The intensity change is

$$E(u, v, w) = \sum_{x, y, z \in \Omega} [I(x + u, y + v, z + w) - I(x, y, z)]^2$$



- Cost function

$$E(u, v, w) = \sum_{x,y,z \in \Omega} [I(x + u, y + v, z + w) - I(x, y, z)]^2$$

- Apply first-order Taylor Series approximation, we have

$$E(u, v, w) = [u \ v \ w] M \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \quad M = \sum_{x,y,z \in \Omega} \begin{bmatrix} I_x^2 & I_x I_y & I_x I_z \\ I_x I_y & I_y^2 & I_y I_z \\ I_x I_z & I_y I_z & I_z^2 \end{bmatrix}$$

M is the covariance matrix of intensity over the surface.

- How to compute I_x, I_y, I_z ?



- Denote the point we are analyzing is $p = [p_x, p_y, p_z]^T$
- In the neighborhood Ω , there are points $\{\mathbf{x}_i = [x_i, y_i, z_i]\}$
- The intensity gradient around p is a vector $\mathbf{e} = [e_x, e_y, e_z]^T \in \mathbb{R}^3$
 - Direction of e is the direction of greatest intensity increase
 - Magnitude $\|\mathbf{e}\|$ is the rate of intensity change
- Ideally, there will be:

$$(x_1 - p_x)e_x + (y_1 - p_y)e_y + (z_1 - p_z)e_z = I(x_1, y_1, z_1) - I(p_x, p_y, p_z)$$

$\dots = \dots$

$$(x_i - p_x)e_x + (y_i - p_y)e_y + (z_i - p_z)e_z = I(x_i, y_i, z_i) - I(p_x, p_y, p_z)$$

$\dots = \dots$



- Scalar form

$$(x_i - p_x)e_x + (y_i - p_y)e_y + (z_i - p_z)e_z = I(x_i, y_i, z_i) - I(p_x, p_y, p_z)$$

- Vector form

$$\mathbf{x}'_i^T \mathbf{e} = \Delta I_i,$$

$$\mathbf{x}'_i = [x'_i, y'_i, z'_i]^T = [x_i - p_x, y_i - p_y, z_i - p_z]^T$$

- Matrix form

$$A\mathbf{e} = \mathbf{b}, \quad A = \begin{bmatrix} x'_1 & y'_1 & z'_1 \\ \vdots & \vdots & \vdots \\ x'_i & y'_i & z'_i \\ \vdots & \vdots & \vdots \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \Delta I_1 \\ \vdots \\ \Delta I_i \\ \vdots \end{bmatrix}$$



- We want to find the intensity gradient \mathbf{e} that satisfies

$$A\mathbf{e} = \mathbf{b}, \quad A = \begin{bmatrix} x'_1 & y'_1 & z'_1 \\ \vdots & \vdots & \vdots \\ x'_i & y'_i & z'_i \\ \vdots & \vdots & \vdots \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \Delta I_1 \\ \vdots \\ \Delta I_i \\ \vdots \end{bmatrix}$$

- Which is impossible in when number of point > 3 , so actually,

$$\min_{\mathbf{e}} \|A\mathbf{e} - \mathbf{b}\|_2^2$$

- Solution given by $\boxed{\mathbf{e} = (A^T A)^{-1} A^T \mathbf{b}}$



- Come back to the Harris 3D with intensity

$$E(u, v, w) = [u \quad v \quad w] M \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \quad M = \sum_{x,y,z \in \Omega} \begin{bmatrix} I_x^2 & I_x I_y & I_x I_z \\ I_x I_y & I_y^2 & I_y I_z \\ I_x I_z & I_y I_z & I_z^2 \end{bmatrix}$$

- Now we know $\mathbf{e} = [I_x, I_y, I_z]$
- One more thing – optionally we can project \mathbf{e} onto local surface.



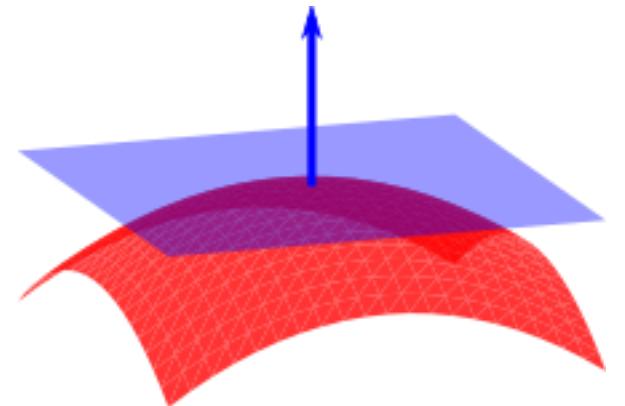
- In many cases, point cloud is from scanning, i.e., points are on the surface of the object/environment
- For every point p , we can fit a surface over the local (small) neighborhood.

$$f(x, y, z) = 0$$

- Using first-order approximation, the surface becomes a plane

$$ax + by + cz + d = 0$$

Surface normal of p : $\mathbf{n} = [n_x, n_y, n_z]^T = \frac{[a, b, c]^T}{\|[a, b, c]^T\|_2}$





Intensity Gradient of Point Cloud

local surface

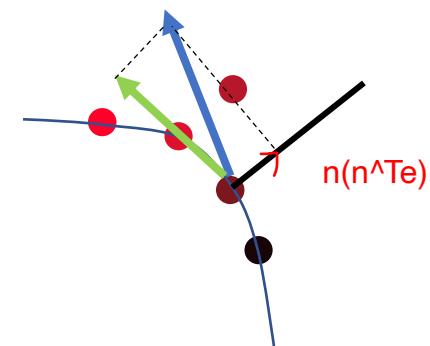
- Projecting intensity gradient e onto the surface may reduce the effect of noise

去掉噪声

- Look at the 2D example
- Color represents intensity, red – large, black – small
- Blue line - e
- Green line - e projected onto the surface (this case it is a curve) $\rightarrow e'$
- Black line – surface normal n
- How to project e onto surface/curve?
 - $e' = e - n(n^T e) = e - n(e^T n)$



Projection onto the surface normal

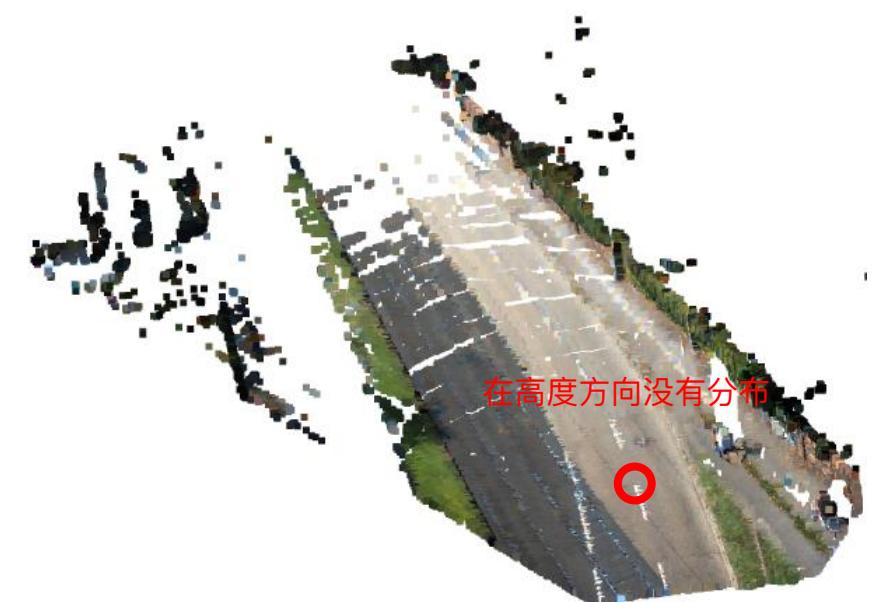




- The intensity change function

$$E(u, v, w) = [u \ v \ w] M \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \quad M = \sum_{x,y,z \in \Omega} \begin{bmatrix} I_x^2 & I_x I_y & I_x I_z \\ I_x I_y & I_y^2 & I_y I_z \\ I_x I_z & I_y I_z & I_z^2 \end{bmatrix}$$

- Now, how to **compute corner response?**
- Kanade & Tomasi (1994): $R = \lambda_3$
 - **assume eigenvalues are sorted from large to small**
- However, $\underline{\underline{R = \lambda_2}}$ is also valid.
 - **Intensity corner on a surface is still a corner.**





- Harris on image

$$E(u, v) = [u \ v] M \begin{bmatrix} u \\ v \end{bmatrix}, \quad M = \sum_{x,y \in \Omega} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Covariance of
image gradient

- Harris on point cloud with intensity

$$E(u, v, w) = [u \ v \ w] M \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \quad M = \sum_{x,y,z \in \Omega} \begin{bmatrix} I_x^2 & I_x I_y & I_x I_z \\ I_x I_y & I_y^2 & I_y I_z \\ I_x I_z & I_y I_z & I_z^2 \end{bmatrix}$$

Covariance of
intensity gradient

- How about point cloud without intensity? Still Covariance!



- A local surface around p is

$$f(x, y, z) = 0$$

- Similarly, we construct a cost function

$$E(u, v, w) = \sum_{x, y, z \in \Omega} [f(x + u, y + v, z + w) - f(x, y, z)]^2$$

- But, what does it mean?

- $f(x, y, z) = 0 \rightarrow$ point $[x, y, z]$ on the surface
- Move it by $[u, v, w]$, is it still on the surface?
 - Likely no.
 - Then, how far is it from the surface $f(x, y, z) = 0$?



- Again, **first-order approximation**. A local surface around p becomes a plane.

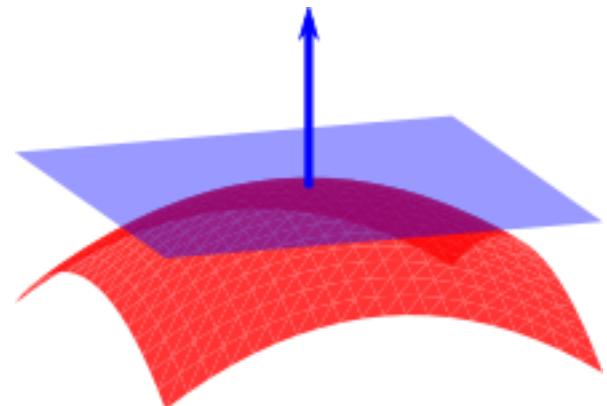
- The surface normal of p is $n = [n_x, n_y, n_z]^T$
- $ax + by + cz + d = 0$, where $a = n_x, b = n_y, c = n_z$

- What is

$$ax' + by' + cz' + d,$$
$$x' = x + u, y' = y + v, z' = z + w$$

- Distance of $[x', y', z']$ to the plane!
- Why?

- $$dist(point, plane) = \frac{ax' + by' + cz' + d}{\sqrt{a^2 + b^2 + c^2}} = \frac{ax' + by' + cz' + d}{1}$$





- Cost function

$$E(u, v, w) = \sum_{x, y, z \in \Omega} [f(x + u, y + v, z + w) - f(x, y, z)]^2$$



- Intuitively, what is that?

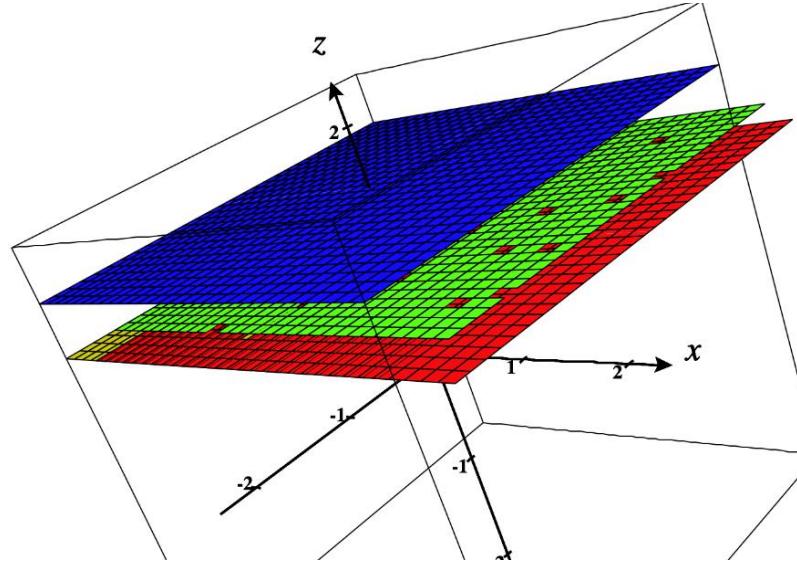
- Move a point by $[u, v, w]$, how far is the new point to the surface?



通过协方差矩阵可以知道是哪种平面，光滑的还是奇奇怪怪的

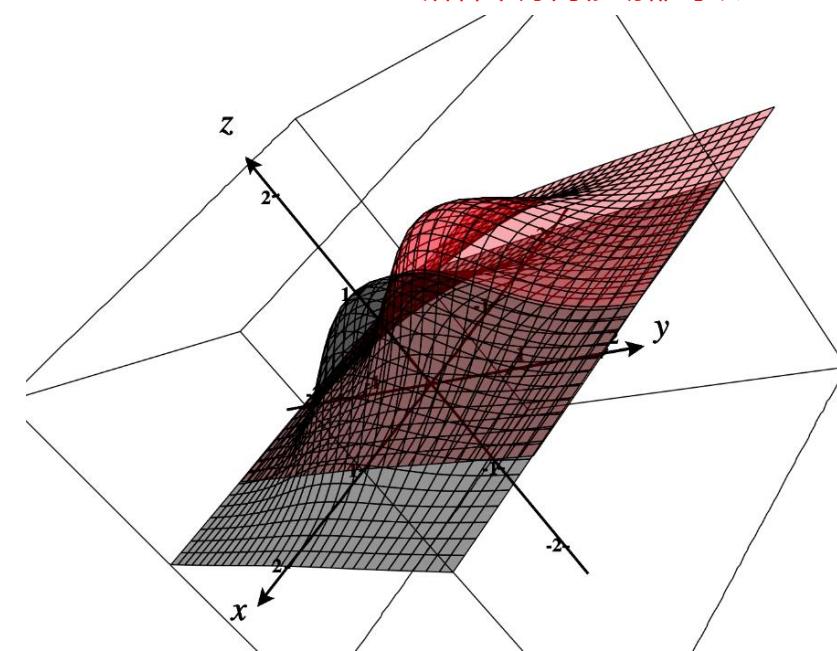
- Example 1: points distributed over a x-y plane
 - Points in Ω change cost E only with w (along z direction)

只有沿着Z方向移动才能产生一个距离



- Example 2: points distributed over some shape
 - Points in Ω change cost E in various directions.

沿各个方向移动都可以





- Cost function

$$E(u, v, w) = \sum_{x, y, z \in \Omega} [f(x + u, y + v, z + w) - f(x, y, z)]^2$$
$$f(x, y, z) = ax + by + cz + d = n_x x + n_y y + n_z z + d$$

- Apply first-order Taylor Series approximation, we have

$$E(u, v, w) = [u \quad v \quad w] M \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \quad M = \sum_{x, y, z \in \Omega} \begin{bmatrix} n_x^2 & n_x n_y & n_x n_z \\ n_x n_y & n_y^2 & n_y n_z \\ n_x n_z & n_y n_z & n_z^2 \end{bmatrix}$$

- M is the covariance matrix of surface normal over the surface.



- With intensity. Tomasi Response $R = \lambda_2$

$$E(u, v, w) = [u \ v \ w] M \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \quad M = \sum_{x,y,z \in \Omega} \begin{bmatrix} I_x^2 & I_x I_y & I_x I_z \\ I_x I_y & I_y^2 & I_y I_z \\ I_x I_z & I_y I_z & I_z^2 \end{bmatrix}$$

Covariance of intensity gradient

- Without intensity. Tomasi Response $R = \lambda_3$

$$E(u, v, w) = [u \ v \ w] M \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \quad M = \sum_{x,y,z \in \Omega} \begin{bmatrix} n_x^2 & n_x n_y & n_x n_z \\ n_x n_y & n_y^2 & n_y n_z \\ n_x n_z & n_y n_z & n_z^2 \end{bmatrix}$$

Covariance of surface normal



- Covariance matrix of $[I_x, I_y, I_z, n_x, n_y, n_z]$

$$M = \sum_{x,y,z \in \Omega} \begin{bmatrix} I_x^2 & I_x I_y & I_x I_z & I_x n_x & I_x n_y & I_x n_z \\ I_x I_y & I_y^2 & I_y I_z & I_y n_x & I_y n_y & I_y n_z \\ I_x I_z & I_y I_z & I_z^2 & I_z n_x & I_z n_y & I_z n_z \\ n_x I_x & n_x I_y & n_x I_z & n_x^2 & n_x n_y & n_x n_z \\ n_y I_x & n_y I_y & n_y I_z & n_x n_y & n_y^2 & n_y n_z \\ n_z I_x & n_z I_y & n_z I_z & n_x n_z & n_y n_z & n_z^2 \end{bmatrix}$$

R=\lambda_3

- Tomasi Response $R = \lambda_4$

- $R = \lambda_3$ is too loose
 - E.g., Harris 6D is a superset of Harris 3D without intensity when $R = \lambda_3$
- $R = \lambda_5$ is too strict
 - E.g., A corner has to be a geometric corner AND intensity corner at the same time.



Harris Family Summary

	Input	Covariance Matrix	Criteria	Intuition
Harris Image	Image	Intensity gradient	λ_2 is small	Intensity corners
Harris 3D	PC with Intensity	Intensity gradient	λ_2 is small	Intensity corners in local surface
Harris 3D	PC	Surface normals	λ_3 is small	Corners in 3D space
Harris 6D	PC with Intensity	Intensity gradient + surface normals	λ_4 is small	Corners in either 3D space / local surface intensity



- Harris Corners 3D/6D are extensions of image based Harris corners.
- How about native methods for point cloud?
- Intrinsic Shape Signatures (ISS): A Shape Descriptor for 3D Object Recognition
 - Keypoints are those have large 3D point variations in their neighborhood
 - Simple, Principle Component Analysis (PCA)
 - The smallest eigenvalue of the covariance matrix should be large.



- Given a point $p_i \in \mathbb{R}^3$, compute its **weighted covariance matrix** over a radius r
 - Points at sparse regions contribute more than those at dense regions
- 1. The weight of any point p_j is
 - $w_j = \frac{1}{|\{p_k: \|p_k - p_j\|_2 < r\}|}$
 - Inversely related to number of points in its neighborhood within r
- 2. **Weighted** covariance matrix:

$$Cov(p_i) = \frac{\sum_{\|p_j - p_i\|_2 < r} w_j (p_j - p_i)(p_j - p_i)^T}{\sum_{\|p_j - p_i\|_2 < r} w_j}$$



4. Compute eigenvalues of $Cov(p_i)$ as $\lambda_i^1, \lambda_i^2, \lambda_i^3$, in the order decreasing magnitude
排除一些没有用的特征点
5. p_i is a keypoint if

$$\frac{\lambda_i^2}{\lambda_i^1} < \gamma_{21} \text{ and } \frac{\lambda_i^3}{\lambda_i^2} < \gamma_{32}$$

- A flat surface can be $\lambda_i^1 = \lambda_i^2 > \lambda_i^3$
- A line can be $\lambda_i^1 > \lambda_i^2 = \lambda_i^3$
- So we have to ensure $\lambda_i^1 > \lambda_i^2 > \lambda_i^3$

6. Non-Maximum Suppression (NMS) with λ_i^3



Classical Methods Summary

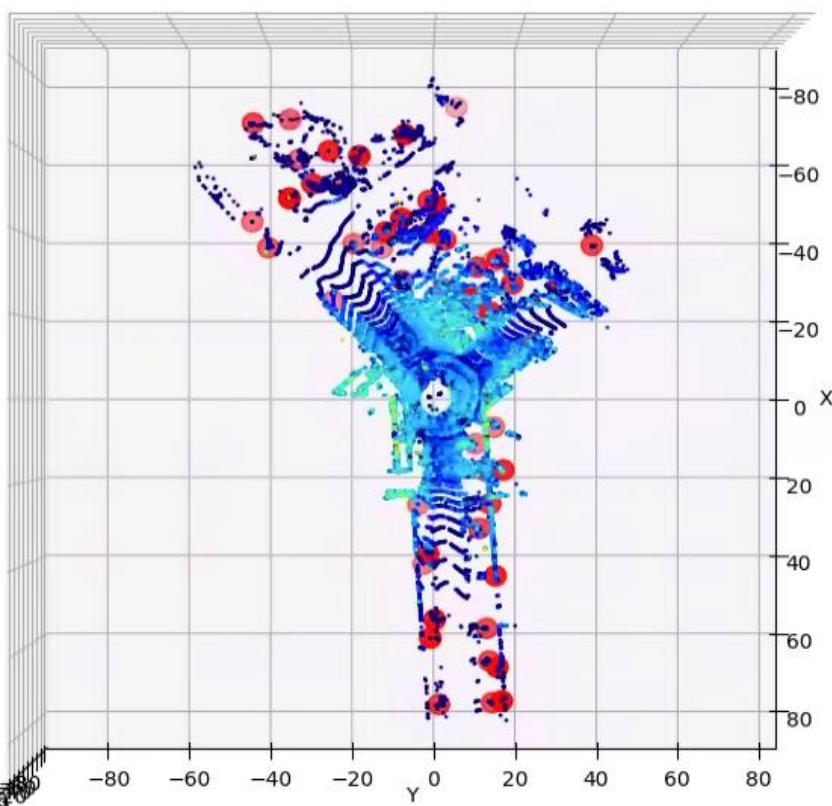
	Input	Covariance Matrix	Criteria	Intuition
Harris Image	Image	Intensity gradient	λ_2 is small	Intensity corners
Harris 3D	PC with Intensity	Intensity gradient	λ_2 is small	Intensity corners in local surface
Harris 3D	PC	Surface normals	λ_3 is small	Corners in 3D space
Harris 6D	PC with Intensity	Intensity gradient + surface normals	λ_4 is small	Corners in either 3D space / local surface intensity
ISS	PC	Weight point coordinates	$\frac{\lambda_i^2}{\lambda_i^1} < \gamma_{21}, \frac{\lambda_i^3}{\lambda_i^2} < \gamma_{32}$	Point distribution is different in 3 dimensions



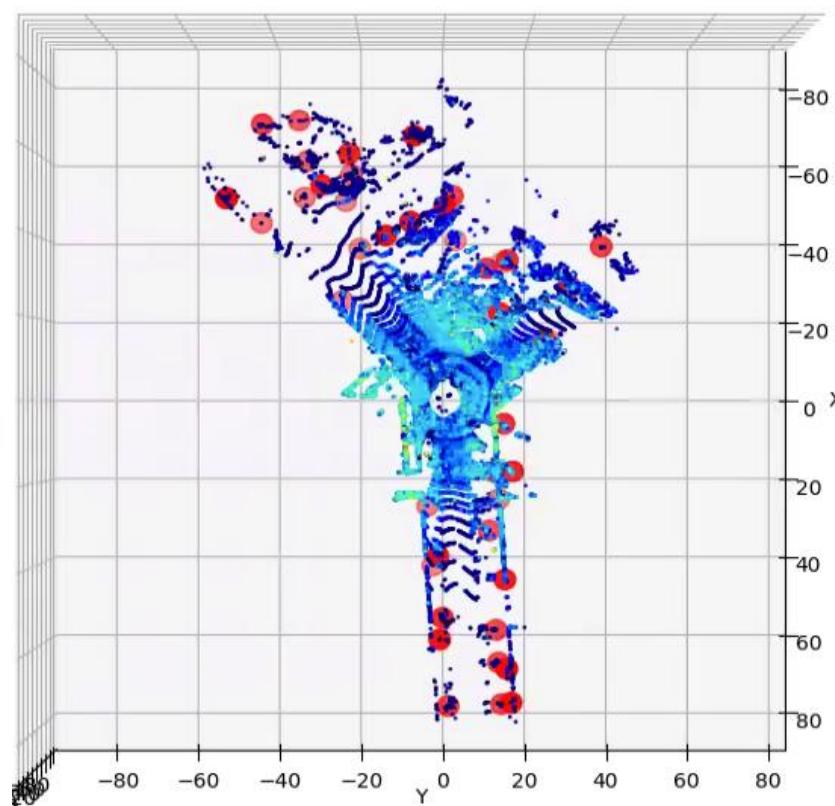
Examples - KITTI

传统方法对噪声很敏感，远处的噪声带来的协方差矩阵很高

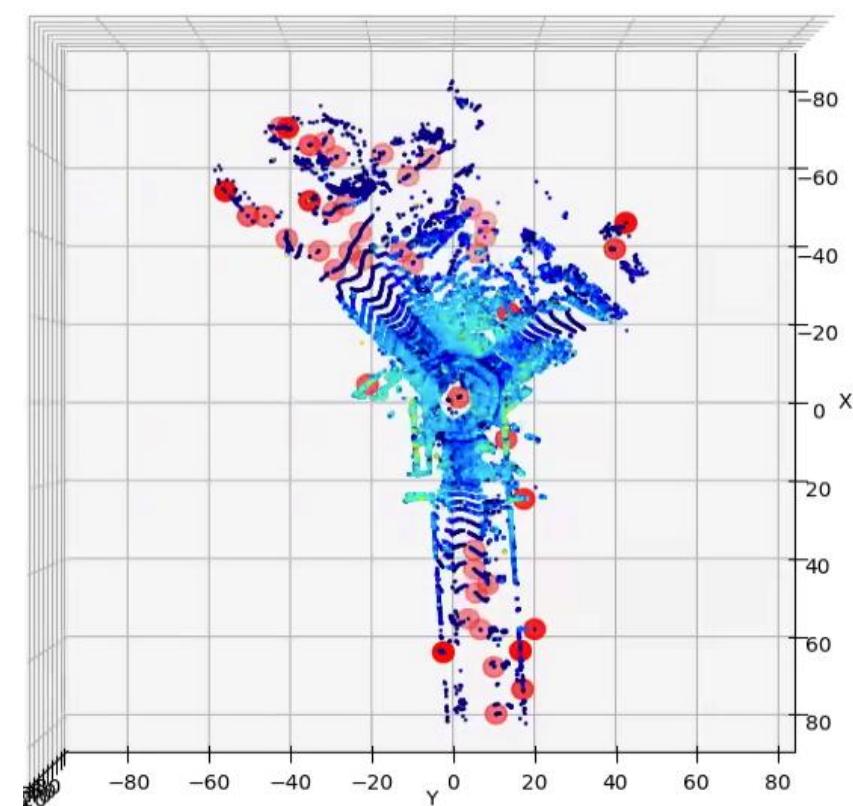
Harris 3D
without intensity



Harris 6D
with intensity



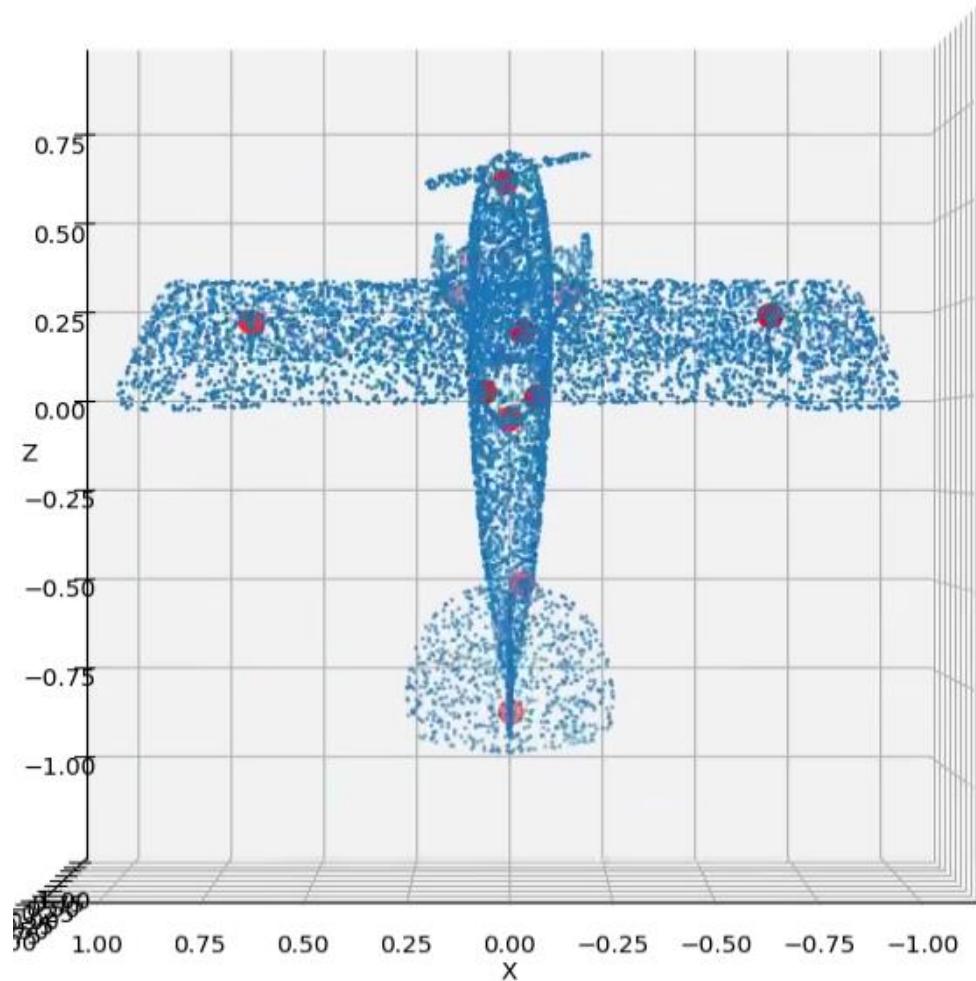
ISS



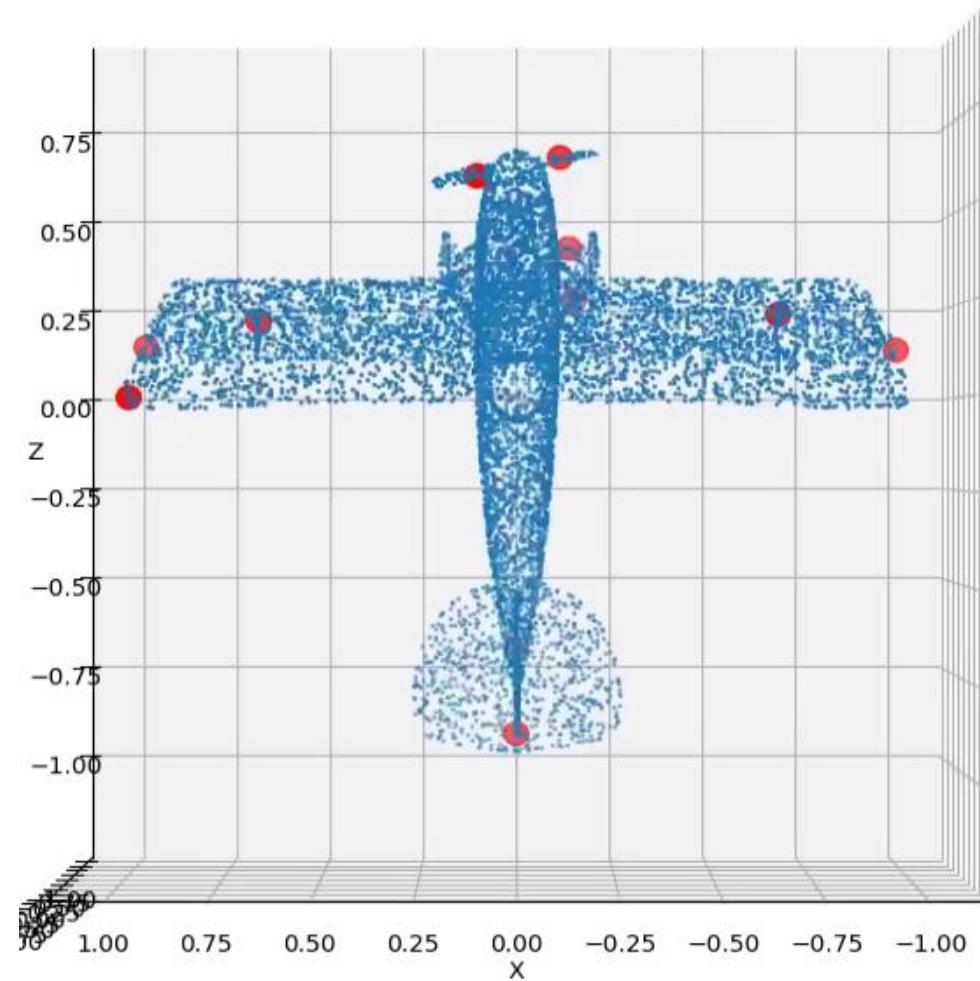


Examples – ModelNet

Harris 3D
without intensity



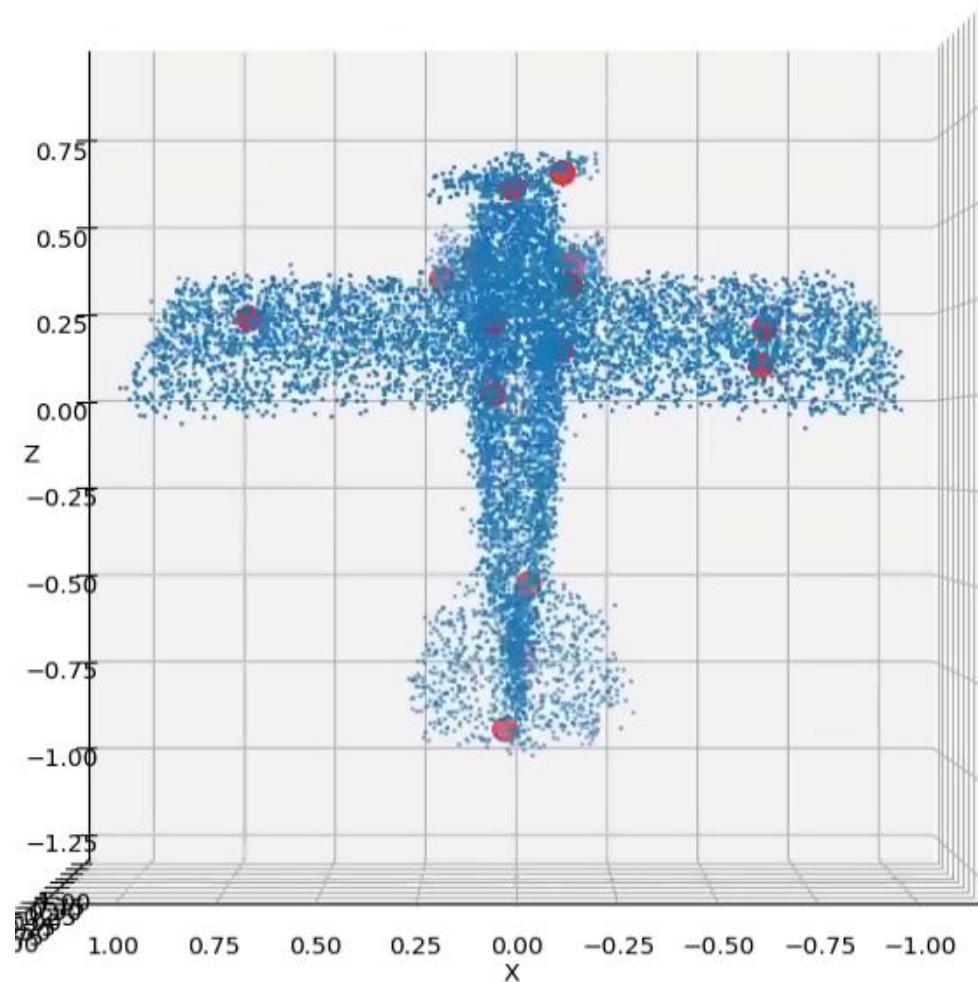
ISS





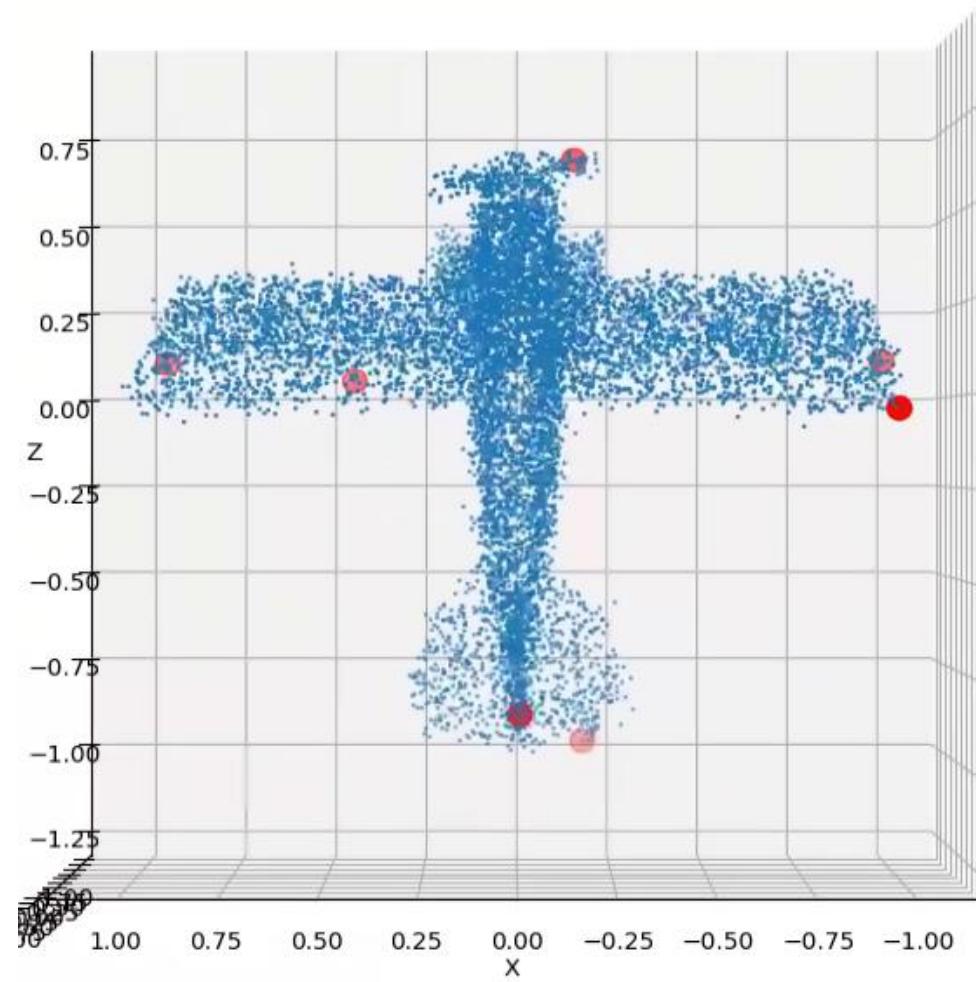
Examples – ModelNet with Noise

Harris 3D
without intensity



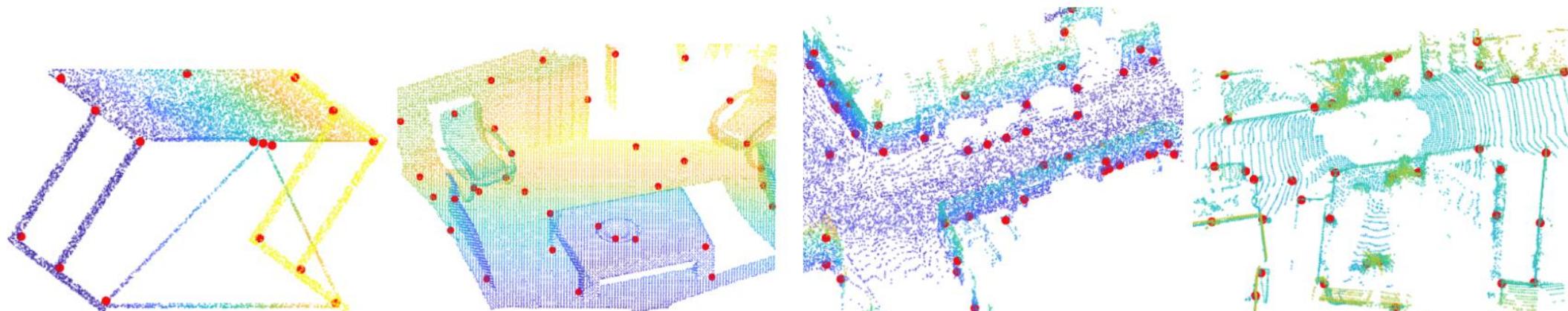
噪声干扰比较大

ISS





- Very few deep learning method for feature detection in point cloud
 - Definition of “feature” is unclear.
 - No annotation/dataset available.
 - Other difficulties of point cloud processing
 - Rotation equivariance.
 - Sparsity
近处观察和远处观察时特征的定义不清晰
 -
- Probably there are only 2 methods before 2020.
 - USIP: Unsupervised Stable Interest Point Detection from 3D Point Clouds
 - 3DFeat-Net: Weakly Supervised Local 3D Features for Point Cloud Registration

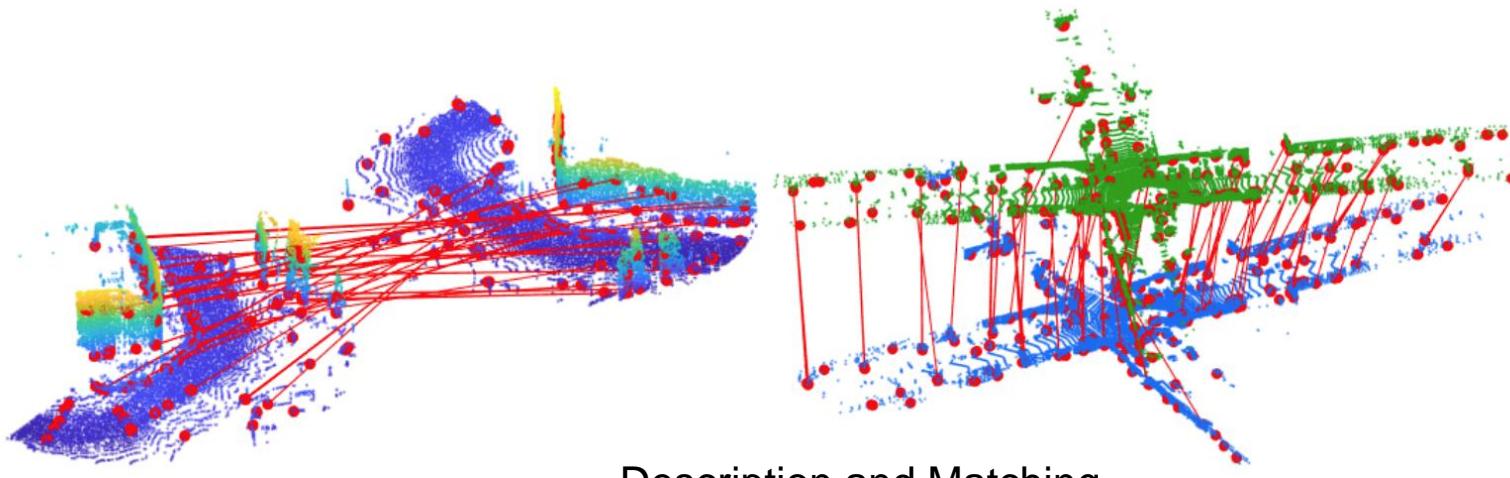


CAD Model

RGBD

Oxford Robotcar

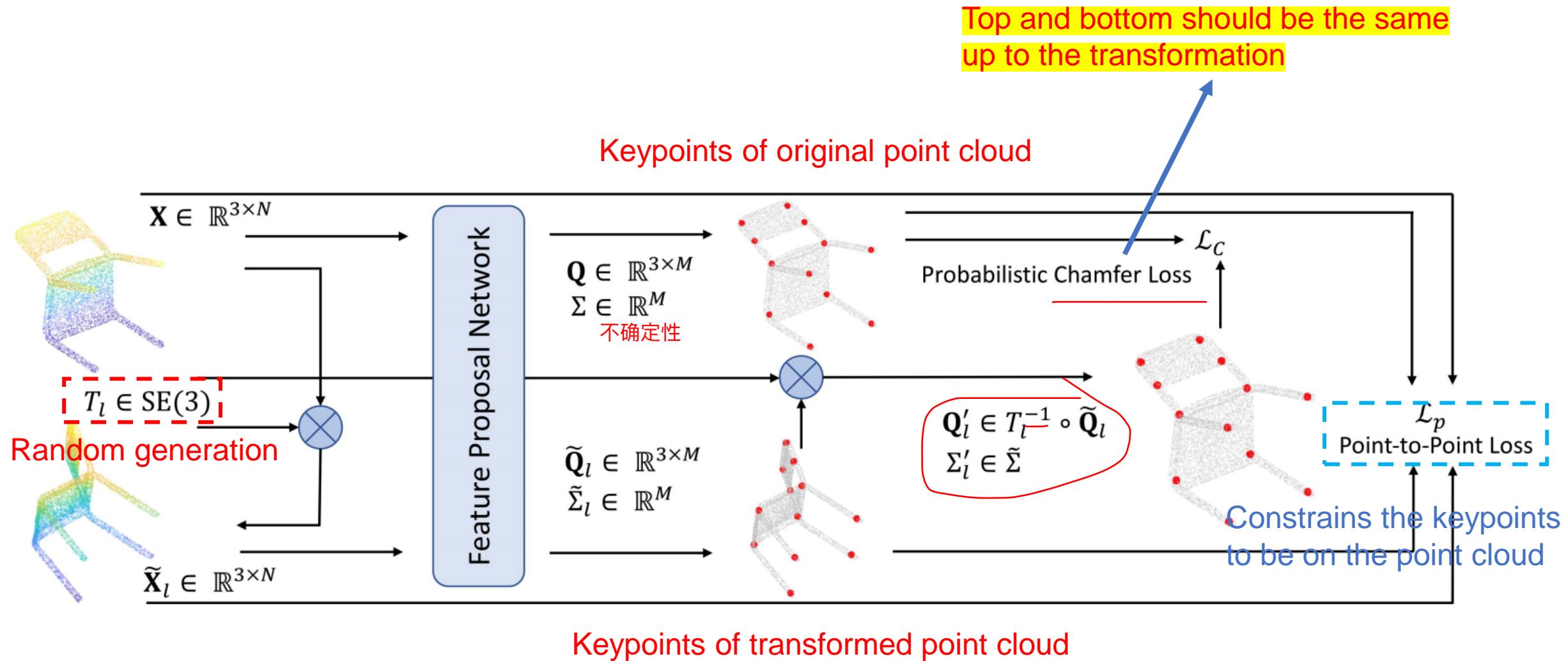
KITTI



Description and Matching



- Unsupervised – Let the network decide what is a keypoint
- Intuition
 - A keypoint is still a keypoint no matter how we transform the object.
 - The concept “keypoint” depends on scale / level-of-detail
不同尺度下
 - The center of an object is a keypoint, if we look at a large scale
 - The textures of the tire are keypoints if we look at the surface of the tire, not keypoints if we are at the scale of the vehicle





- Notations

- Point cloud $X = [X_1, \dots, X_N] \in \mathbb{R}^{3 \times N}$
- Transformation matrix $T \in SE(3)$, i.e., $T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4 \times 4}$

- Input

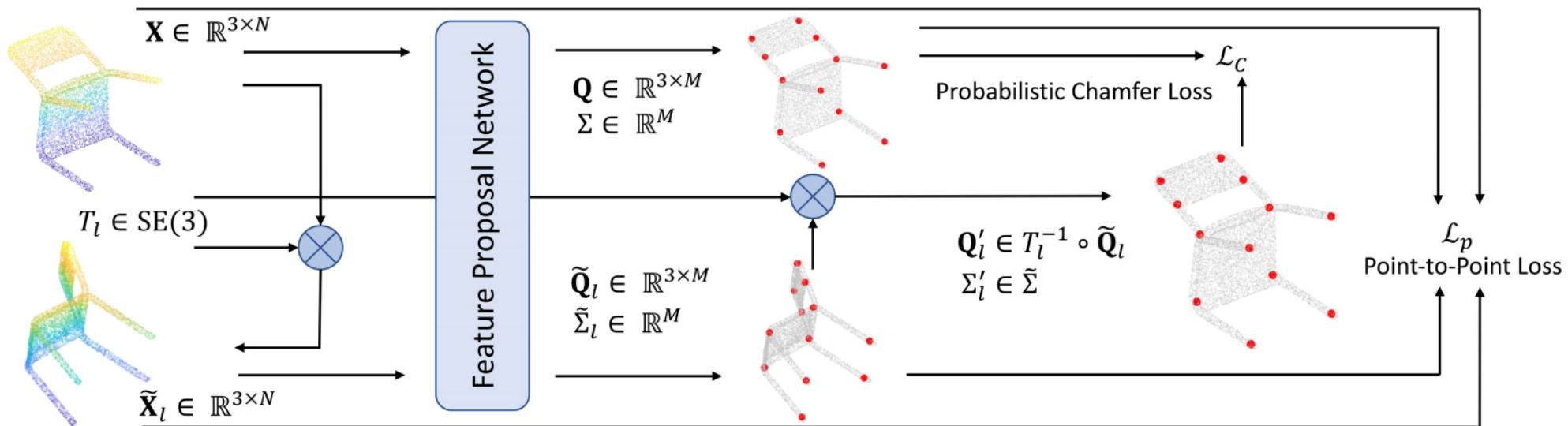
- A point cloud: X

- Output

- M Keypoints $Q = \{Q_1, \dots, Q_M\}, Q_i \in \mathbb{R}^{3 \times M}$
 - The uncertainties $\Sigma = \{\sigma_1, \dots, \sigma_M\}, \sigma_i \in \mathbb{R}^+$



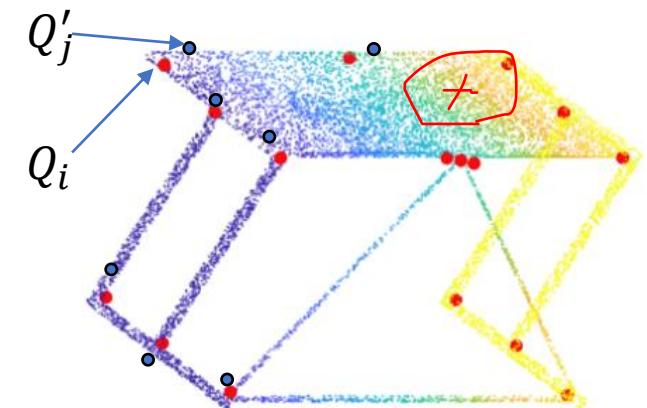
1. Feed X into FPN (Feature Proposal Network), get Q, Σ
2. Randomly generate $T_l \in SE(3)$
3. Transform X into $\tilde{X}_l = T_l \circ X$, where $\tilde{X}_l = T_l \circ X = RX + t$
4. Feed \tilde{X}_l into FPN, get $\tilde{Q}_l, \tilde{\Sigma}_l$
5. Transform $\tilde{Q}_l, \tilde{\Sigma}_l$ back into $Q'_l = T_l^{-1} \circ \tilde{Q}_l, \Sigma'_l = \tilde{\Sigma}_l$
6. Loss function: make $\{Q, \Sigma\} = \{Q'_l, \Sigma'_l\}$





- Standard Chamfer Loss

$$\sum_{i=1}^M \min_{Q'_j \in \mathbf{Q}'} \|Q_i - Q'_j\|_2 + \sum_{j=1}^M \min_{Q_i \in \mathbf{Q}} \|Q_i - Q'_j\|_2$$

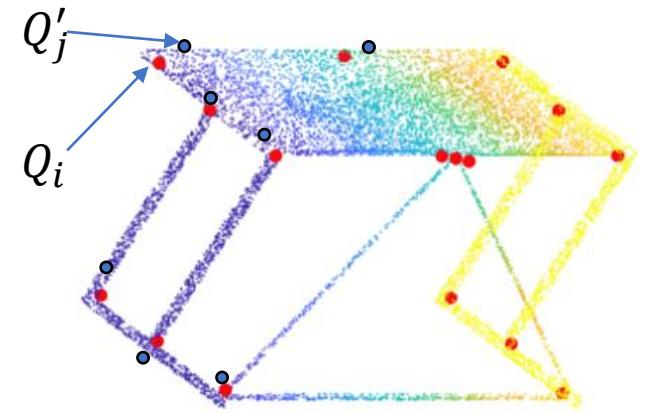


- Problems:
 - Not all keypoints are equal.
 - The receptive field (explained later) of the FPN is limited 每个特征点的感知域有限
 - Some keypoint Q_i locates on smooth surface
 - These highly uncertain keypoints hurts the loss function
- Solution → Probabilistic Chamfer Loss
 - Add weighting according the the predicted uncertainty Σ



Probabilistic Chamfer Loss

- How to convert the uncertainty $\sigma_i \in \mathbb{R}^+$ into probability?
 - Consider a pair of keypoints $\{Q_i, \sigma_i\}$ and $\{Q'_j, \sigma'_j\}$
 - Distance $d_{ij} = \|Q_i - Q'_j\|_2$
- Model the probability of Q_i, Q'_j being the same keypoint (they are matched) as,
$$p(d_{ij} | \sigma_{ij}) = \frac{1}{\sigma_{ij}} \exp\left(-\frac{d_{ij}}{\sigma_{ij}}\right), \quad \sigma_{ij} = \frac{\sigma_i + \sigma'_j}{2} > 0, \quad d_{ij} = \min_{Q'_j \in \mathbf{Q}'} \|Q_i - Q'_j\|_2 \geq 0.$$
- This is an exponential distribution
 - d_{ij} is the random variable
 - σ_{ij} is the parameter of the distribution, which is given by the FPN





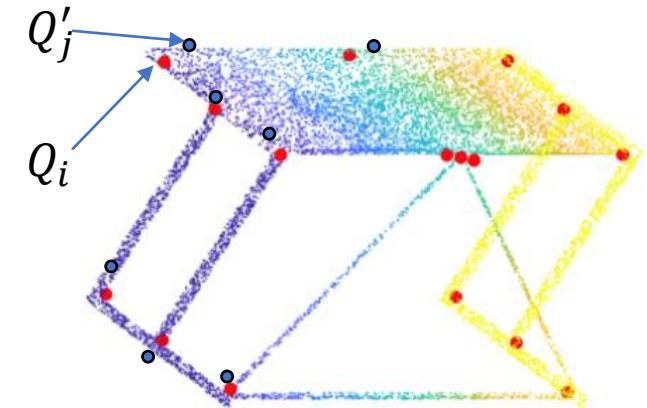
- For Q_i , how do I know which $\{Q'_j, \sigma'_j\}$ is the matched one?
 - Same as standard Chamfer loss
 - For each Q_i , find the nearest keypoint in Q' as the matched
 - For each Q'_j , find the nearest keypoint in Q as the matched
- Now, for the blue / green part, we have

$$p(D_{ij} \mid \Sigma_{ij}) = \prod_{i=1}^M p(d_{ij} \mid \sigma_{ij})$$

$$\sigma_{ij} = \frac{\sigma_i + \sigma'_j}{2} > 0, \quad d_{ij} = \min_{Q'_j \in Q'} \|Q_i - Q'_j\|_2 \geq 0$$

$$p(D_{ji} \mid \Sigma_{ji}) = \prod_{j=1}^M p(d_{ji} \mid \sigma_{ji})$$

$$\sigma_{ji} = \frac{\sigma'_j + \sigma_i}{2} > 0, \quad d_{ji} = \min_{Q_i \in Q} \|Q_i - Q'_j\|_2 \geq 0$$

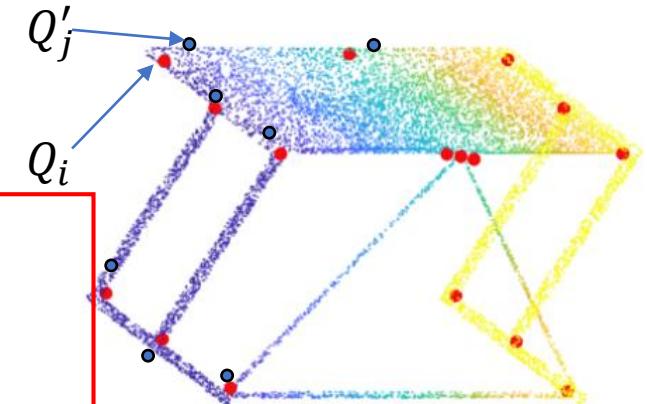




- Final loss, maximize the joint likelihood \rightarrow minimize the ^{negative} log likelihood

$$\mathcal{L}_c = -\ln p(D_{ji} | \Sigma_{ji}) - \ln p(D_{ij} | \Sigma_{ji})$$

$$\begin{aligned}\mathcal{L}_c &= \sum_{i=1}^M -\ln p(d_{ij} | \sigma_{ij}) + \sum_{j=1}^M -\ln p(d_{ji} | \sigma_{ji}) \\ &= \sum_{i=1}^M \left(\ln \sigma_{ij} + \frac{d_{ij}}{\sigma_{ij}} \right) + \sum_{j=1}^M \left(\ln \sigma_{ji} + \frac{d_{ji}}{\sigma_{ji}} \right)\end{aligned}$$



- In fact, according to our Probabilistic Chamfer Loss, uncertainty σ has physical meanings!



Probabilistic Chamfer Loss

- Look at the basic element of the loss function

$$p(d_{ij} | \sigma_{ij}) = \frac{1}{\sigma_{ij}} \exp\left(-\frac{d_{ij}}{\sigma_{ij}}\right)$$

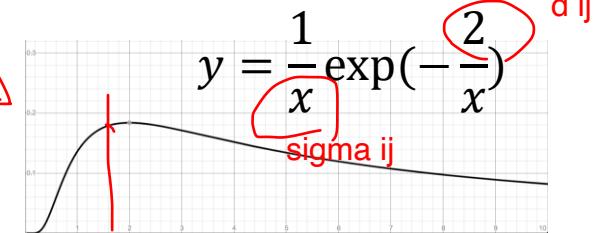
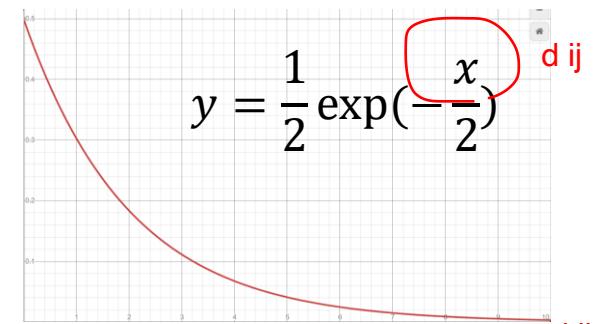
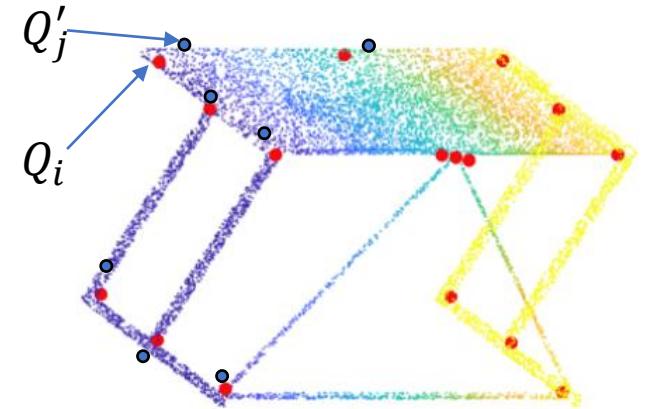
- Compute its first derivative over σ_{ij}

$$\frac{\partial p(d_{ij} | \sigma_{ij})}{\partial \sigma_{ij}} = \frac{d_{ij} \exp(-d_{ij}/\sigma_{ij})}{\sigma_{ij}^3} - \frac{\exp(-d_{ij}/\sigma_{ij})}{\sigma_{ij}^2}$$

- Solve for the stationary point:

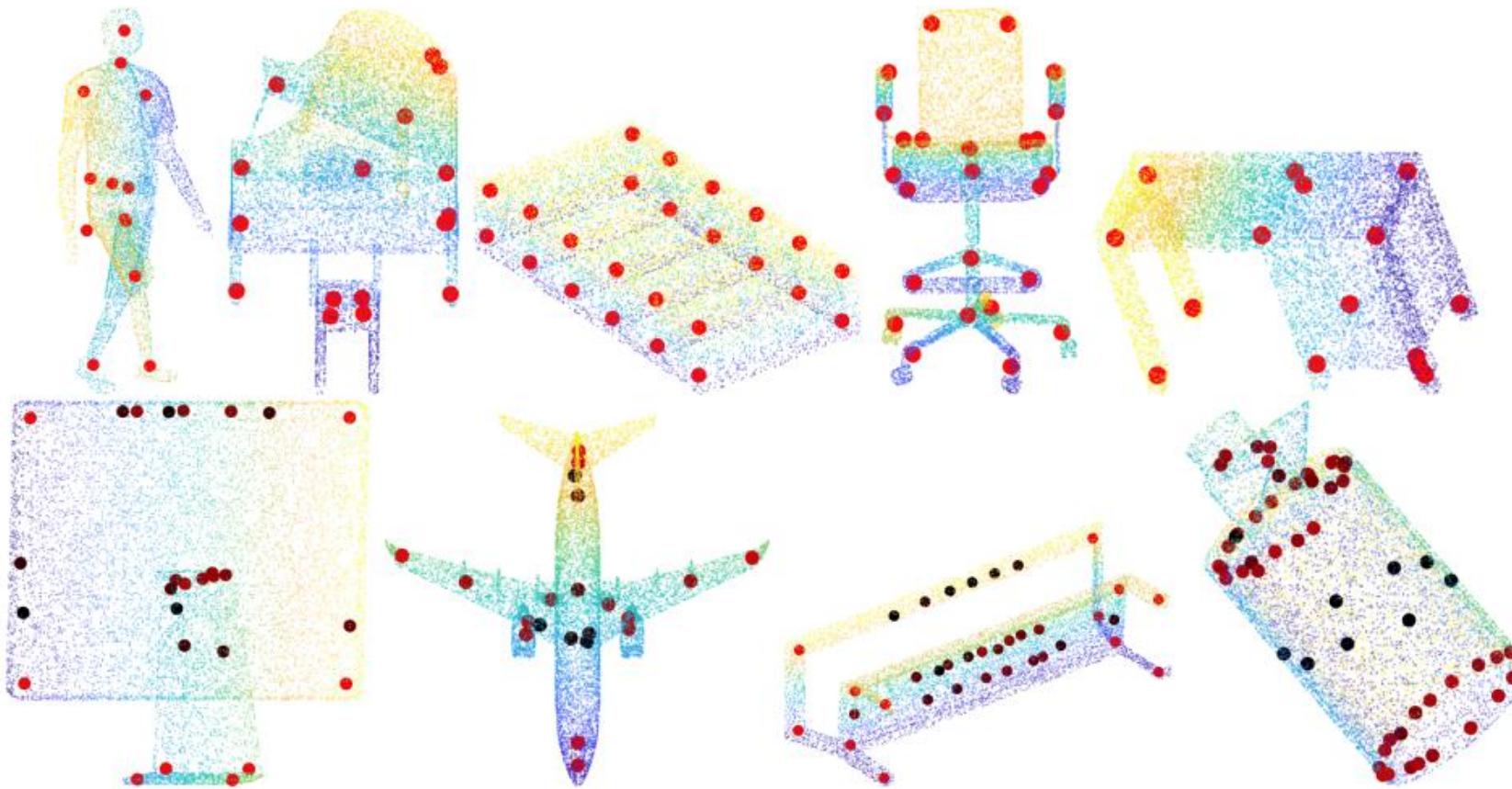
$$\frac{\partial p(d_{ij} | \sigma_{ij})}{\partial \sigma_{ij}} = 0 \Rightarrow \sigma_{ij} = d_{ij}$$

- It means the best σ_{ij} equals to d_{ij} . Network is predicting the matching distance!



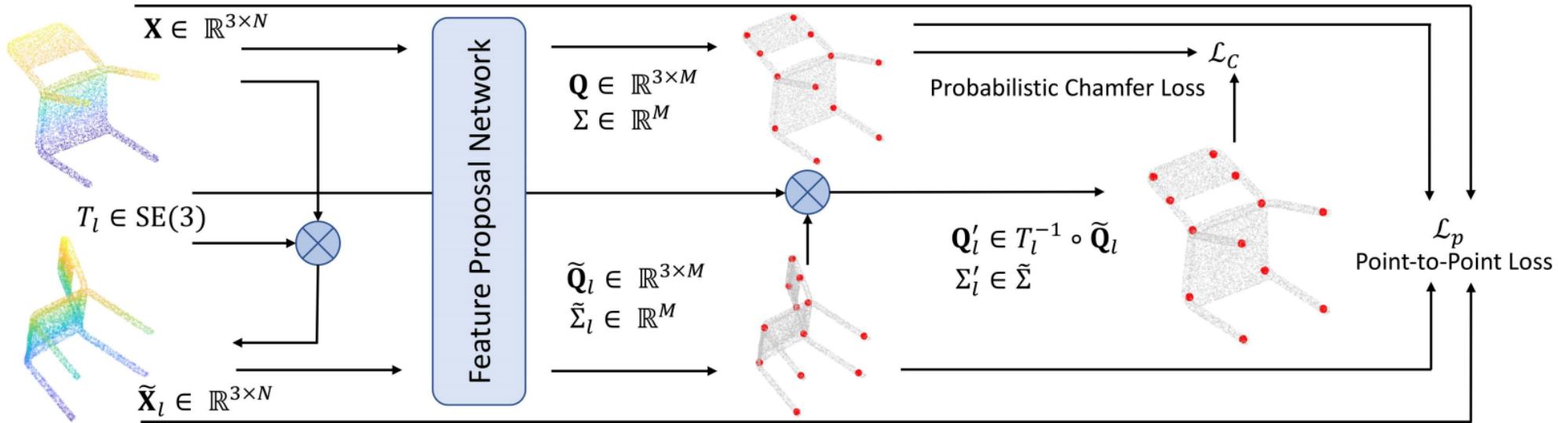


Probabilistic Chamfer Loss



USIP keypoints on ModelNet40.

Red - σ is small, low uncertainty. Black - σ is large, high uncertainty



- What if the network predicts the **object center** as the keypoint?

- $Q_1 = Q_2 = \dots = Q_M = \text{object center} = \tilde{Q}_i = Q'_i$
- $L_c = 0$, perfect!
- But this is not what we want!

Lemma 1. $f(\mathbf{Y}') \equiv Rf(\mathbf{Y}) \oplus t$ when $f(\cdot)$ outputs the **centroid** of the input point cloud, i.e., $f(\mathbf{Y}) = \frac{1}{N} \sum_n Y_n$ and $f(\mathbf{Y}') = \frac{1}{N} \sum_n Y'_n$.



- Further more, the loss L_c will be small if
 - Network outputs points on the principle axis
- Intuition:
 - principle axis doesn't change no matter what is the transform T_l
 - Similar to object center
- Mathematically, it is Lemma 2
- How to proof?
 - Similar to the proof of PCA
 - Here we just need to understand the intuition

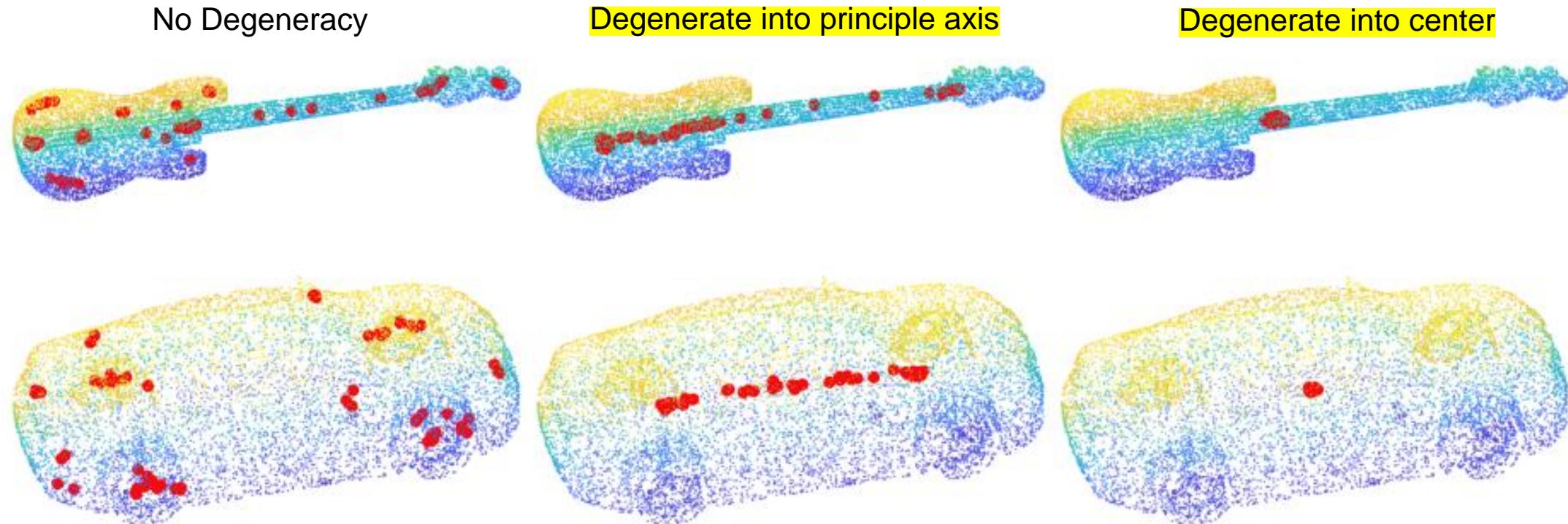
Lemma 2. $f(\mathbf{Y}') \equiv Rf(\mathbf{Y}) \oplus t$ when $f(\cdot)$ is translational equivariant, i.e., $f(\cdot) \oplus t = f(\cdot \oplus t)$, and outputs points that are in the linear subspace of any **principal axis** from the input point cloud denoted as $\mathbf{U} = [U_1, U_2, U_3] \in \mathbb{R}^{3 \times 3}$, i.e., $f(\mathbf{Y}) = [c_1 U_i^T, \dots, c_M U_i^T]^T$ and

$$\begin{aligned} f(\mathbf{Y}') &= f(R\mathbf{Y} \oplus t) \\ &= f(R\mathbf{Y}) \oplus t \quad (\text{translation equivariance}) \quad (10) \\ &= [c_1 U'_i^T, \dots, c_M U'_i^T]^T \oplus t, \end{aligned}$$

where U_i can be any principal axis in \mathbf{U} and c_1, \dots, c_M are scalar coefficients in \mathbb{R} .



Degeneracy and Receptive Field



- When does these two “degeneracy” happen?
 - The **receptive field is large enough**
 - The network can predict **where is the center / principle**

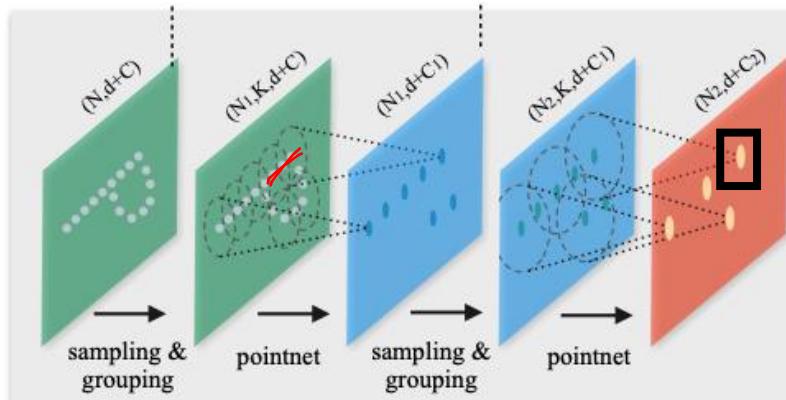


当神经网络看到整个物体时，才能预测中心或者主轴，因此只要限制感知域就可以了

- There are two **degeneracy cases**:
 - **Object center**
 - **Principle axis** 主向量
- How to prevent them?
 - Design the receptive field of the FPN carefully.
 - A keypoint's **Receptive Field**
 - What are the points that are visible to the FPN when predicting this keypoint?
 - If the FPN only sees a small part of the object, it can't predict where is the center / principle axis



- FPN can be any network, with one requirement
 - The receptive field of each keypoint is controllable.
- For example, we can use PointNet++ as the FPN.



Each point has limited receptive field.

- In fact, USIP uses SO-Net as FPN
 - SO-Net: Self-Organizing Network for Point Cloud Analysis
 - Better performance, adaptive to various point density

对点云的密度不太敏感



- The Feature Proposal Network (FPN) doesn't ensure keypoints on the object
 - Simply add a constraint by standard Chamfer Loss

$$\mathcal{L}_{point} = \sum_{i=1}^M \min_{X_j \in \mathbf{X}} \|Q_i - X_j\|_2^2 + \sum_{i=1}^M \min_{\tilde{X}_j \in \tilde{\mathbf{X}}} \|\tilde{Q}_i - \tilde{X}_j\|_2^2$$

- Final Loss function

$$L = L_c + \lambda L_{point}$$

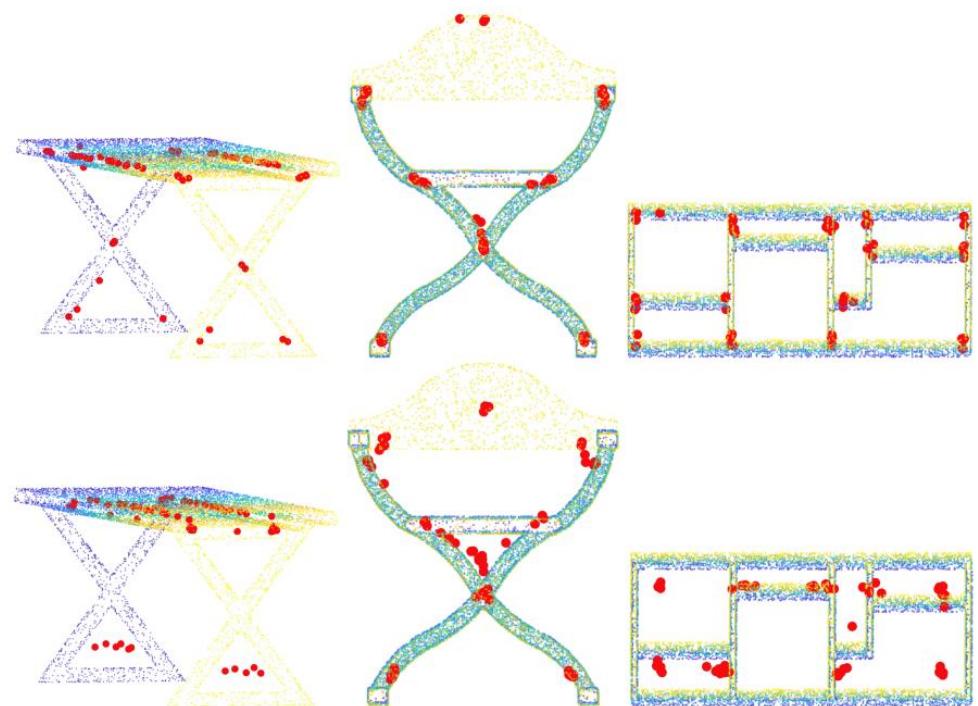
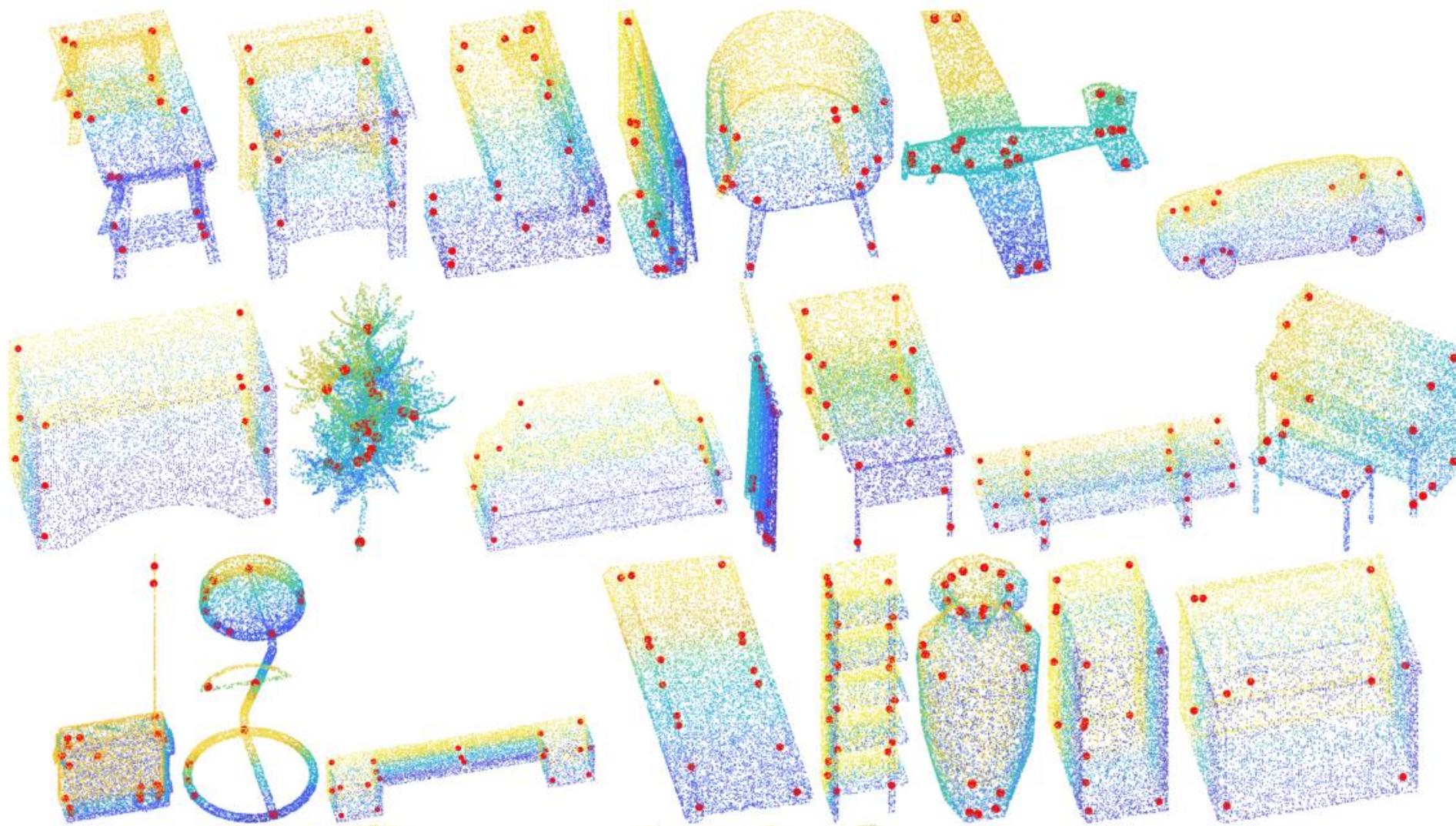


Figure 9. Visualization of USIP keypoints with different λ in Point-to-Point loss. First row $\lambda = 6$, second row $\lambda = 0$.

特征需要落在原始点云上去

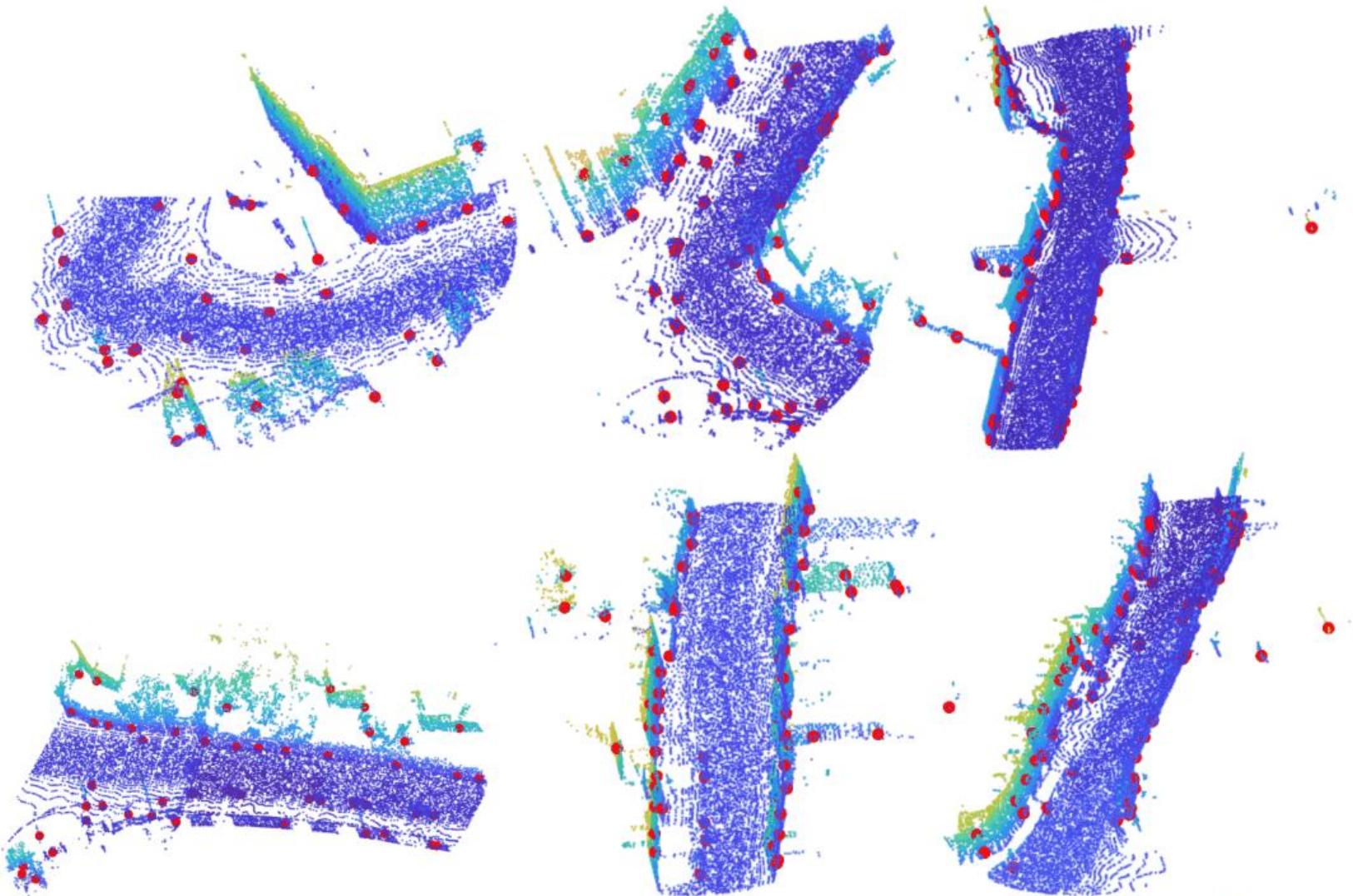


USIP Results – ModelNet40



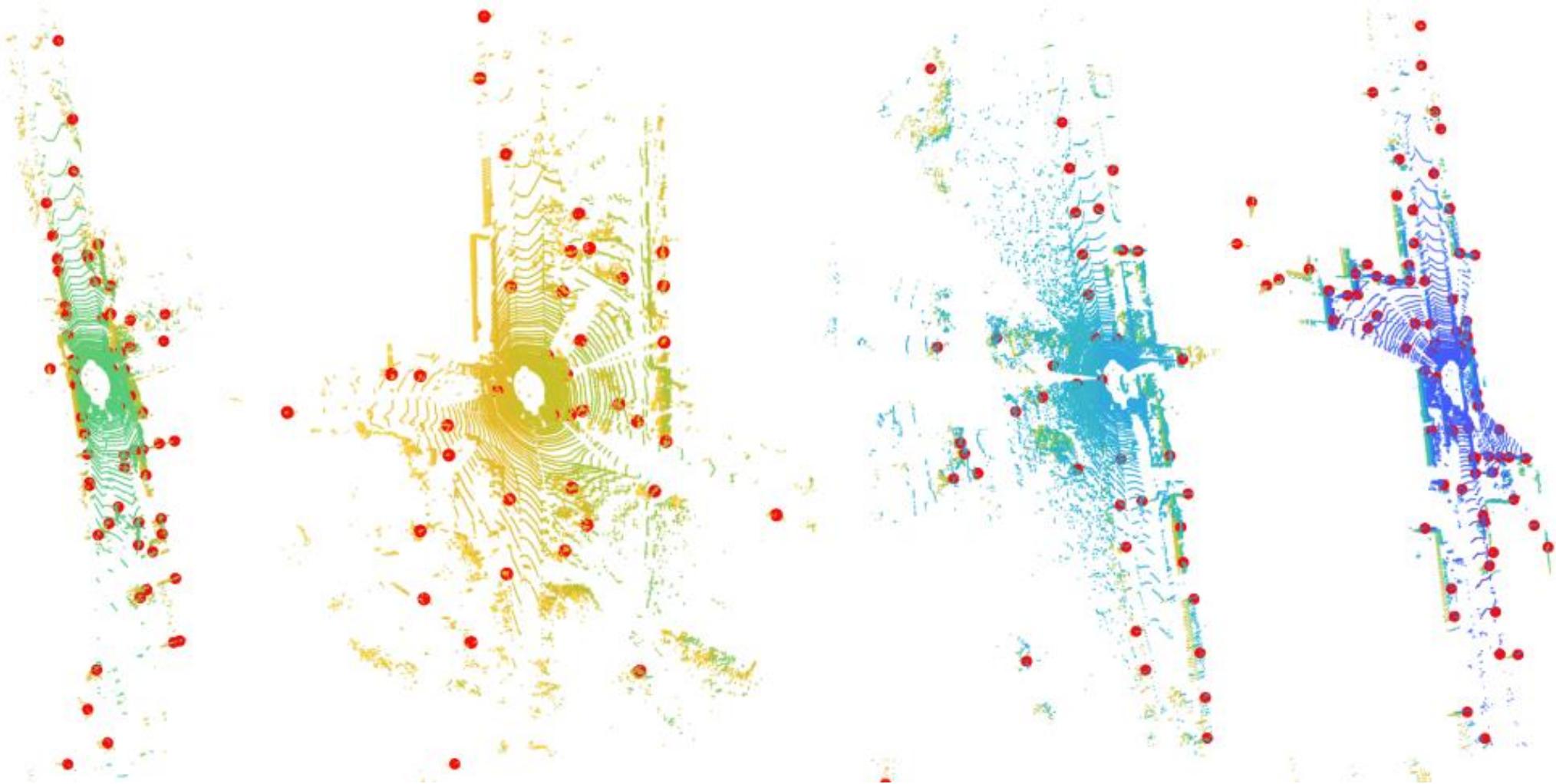


USIP Results – Oxford RobotCar





USIP Results – KITTI





- Ideas
 - Predicts keypoint locations and uncertainties
 - Unsupervised training by Probability Chamfer Loss
 - Controllable receptive field to prevent degeneracy
- Advantages
 - Data driven
 - Robust to noise / sparsity
 - Very stable & repeatable



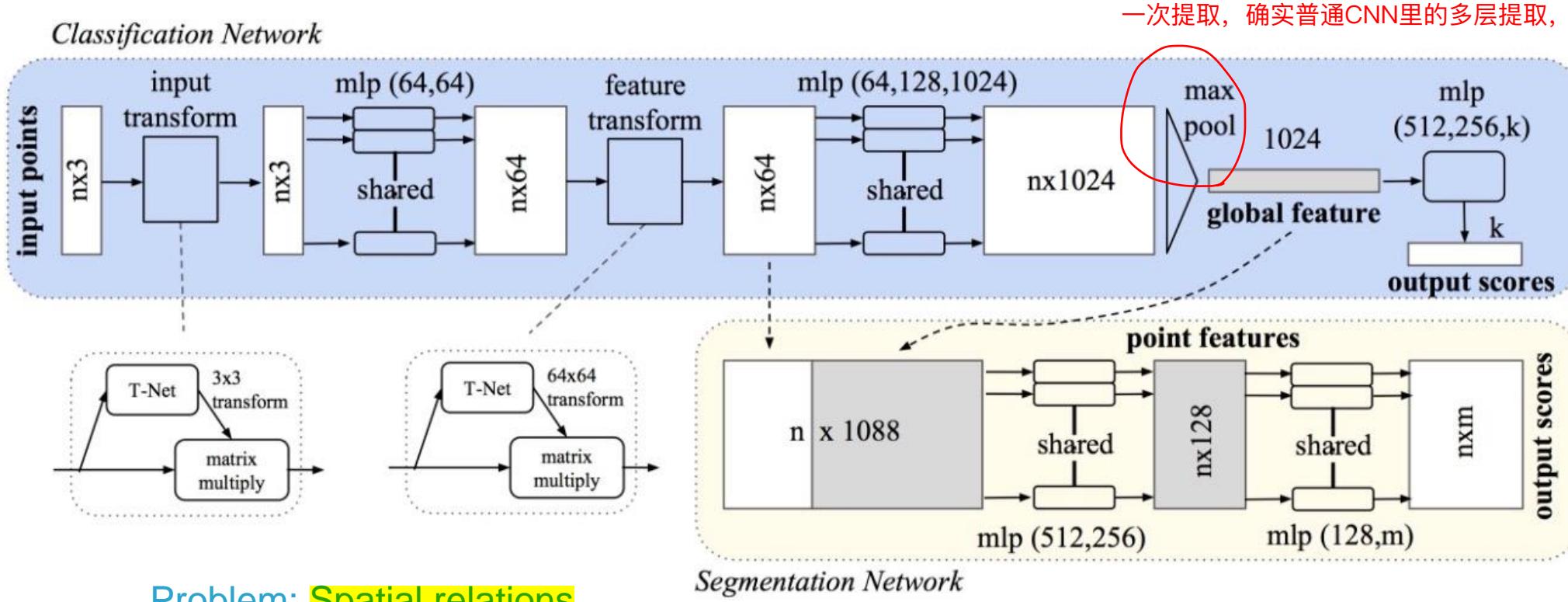
- Handcrafted – Not stable, sensitive to noise. But doesn't need training.
 - Harris family
 - Harris 3D with/without intensity
 - Harris 6D with intensity
 - ISS
- Deep learning – Stable and robust. But needs training (unsupervised).
 - USIP



- Implement your own ISS keypoint detection.
 - Apply your own ISS on ModelNet40 (choose 3 objects from different categories)
 - Visualize the object and the keypoints together.
 - Submit your code and the visualize screenshots
- Implement feature descriptors.
 - Homework of Lecture 8.
- Register two point clouds by feature detection and description
 - Homework of Lecture 9.



SO-Net – Prior Works

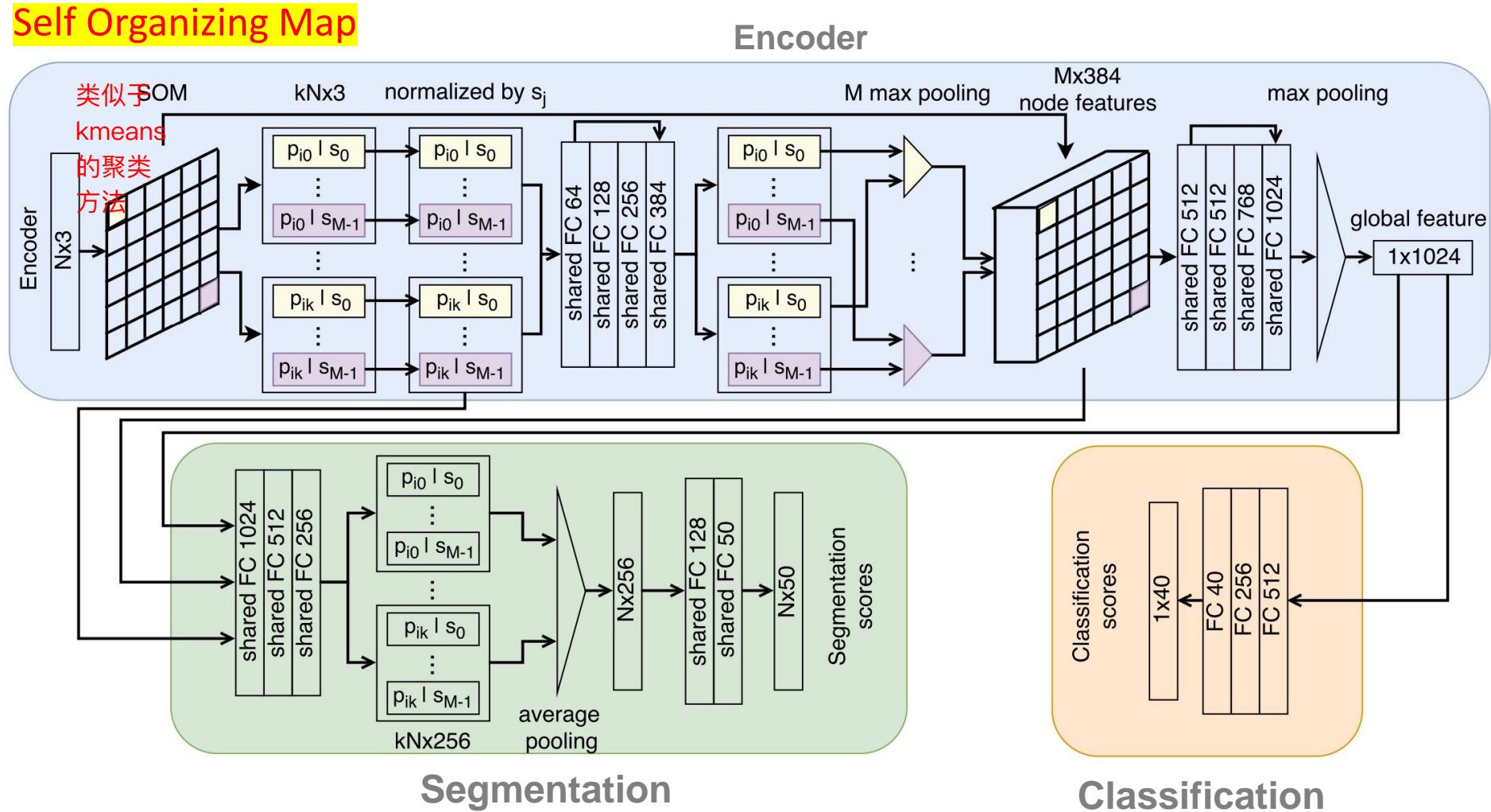


Problem: Spatial relations
between points are ignored!



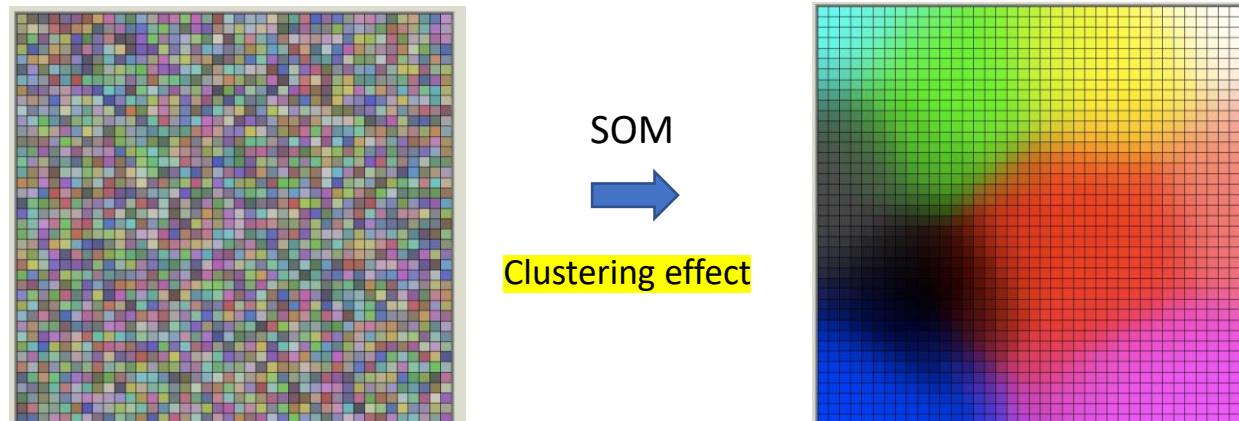
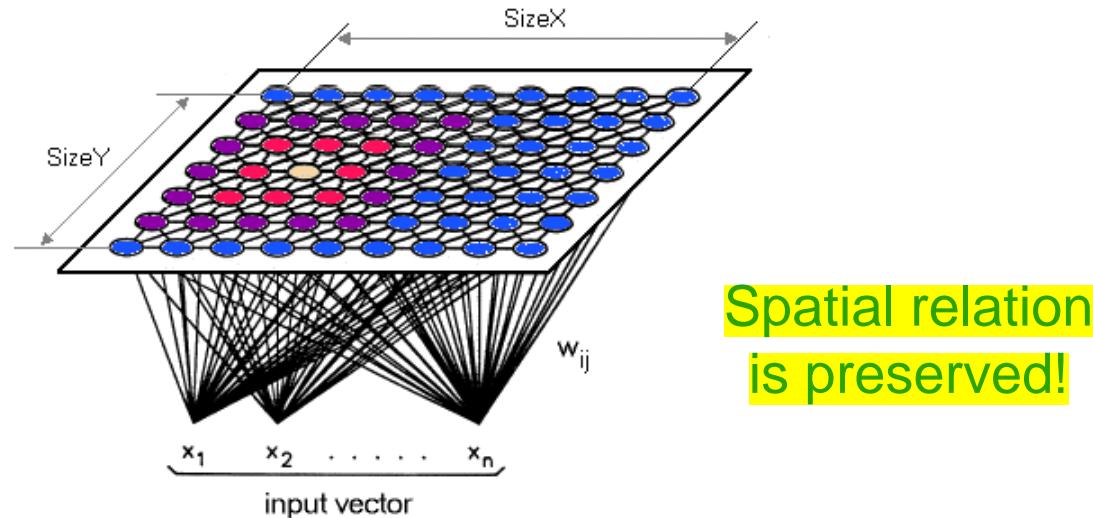
SO-Net – Model the Space with SOM

重点是怎么切割点云





Self-Organizing Map



具有拓扑结构



Self-Organizing Map

$$P = \{p_i \in \mathbb{R}^3, i = 1, \dots, N\}$$
$$S_M = \{s_j \in \mathbb{R}^3, j = 1, \dots, M\}$$

Competition

Given p_i , find the winning node s_j

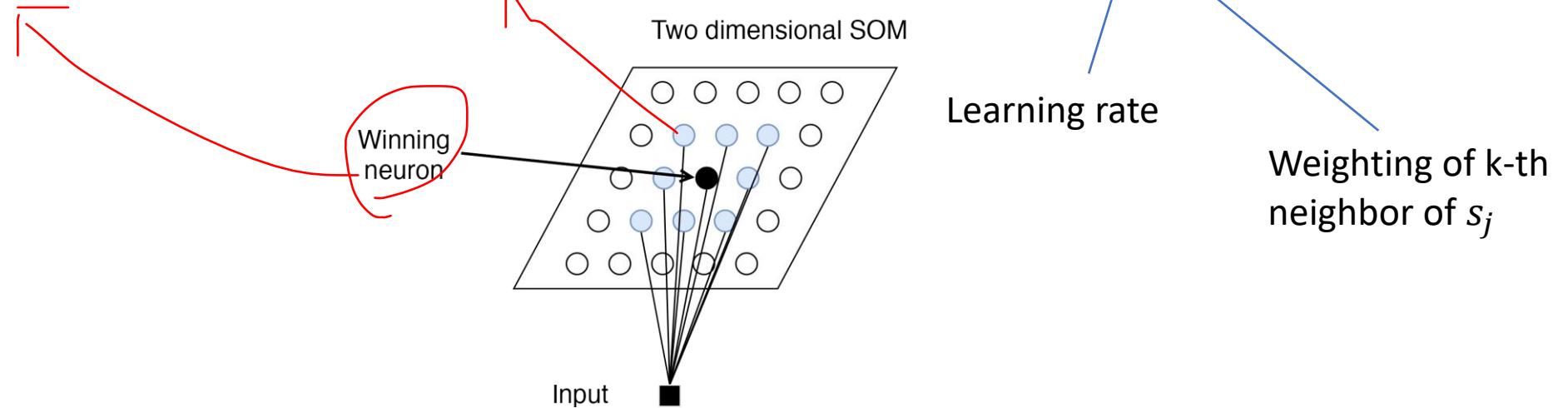
Cooperation

Find the topological neighborhood around node s_j
和 K-means 不同的地方

蓝色的邻域

Adaptation

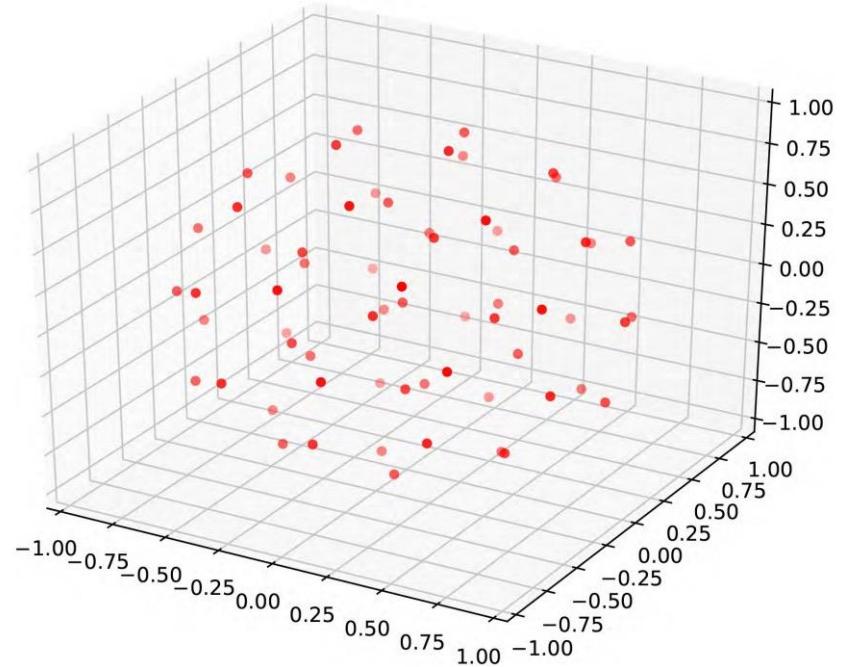
Update node s_j and its neighbors
$$s_{j_k} = s_{j_k} + \lambda w_k (\text{input} - s_{j_k})$$



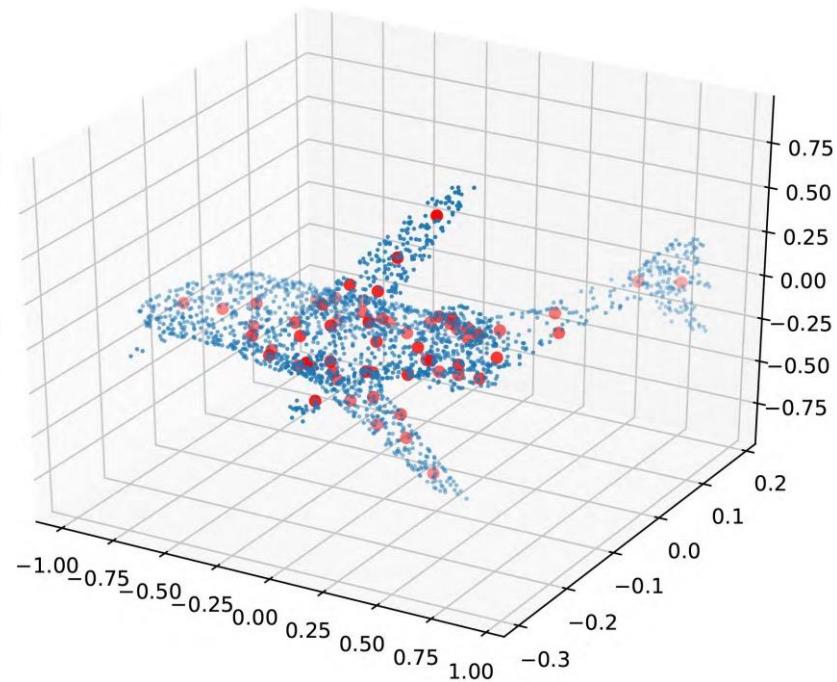


Self-Organizing Map

Initial nodes in a unit ball

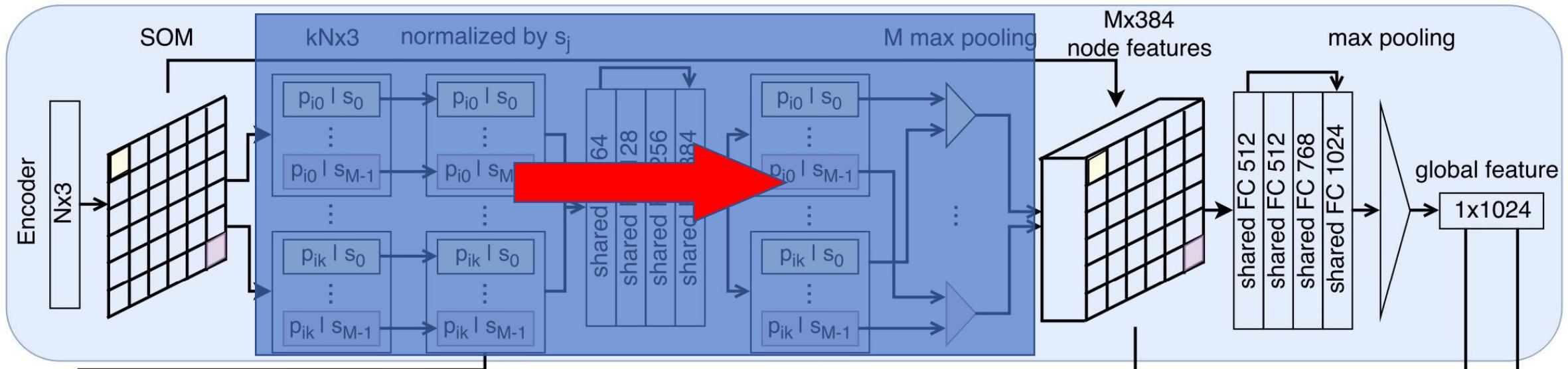


Weights after SOM





- Assume we have the SOM now
- How to convert SOM into “SOM Feature Map”



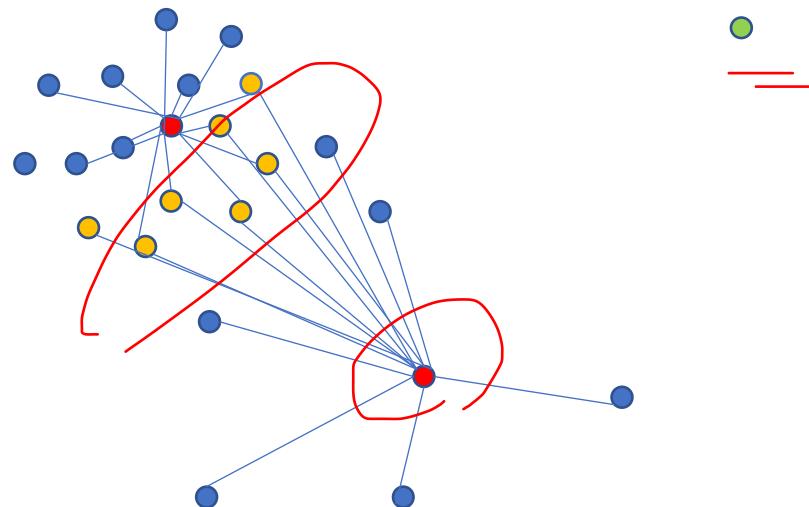


- Method 1, Node-to-Point (PointNet++ style)

- Not adaptive to scale and density
- Difficult to control receptive field
 - Some points will never be connected
 - Some points will be connected multiple times.

感知域不好控制

取决于点怎么分布



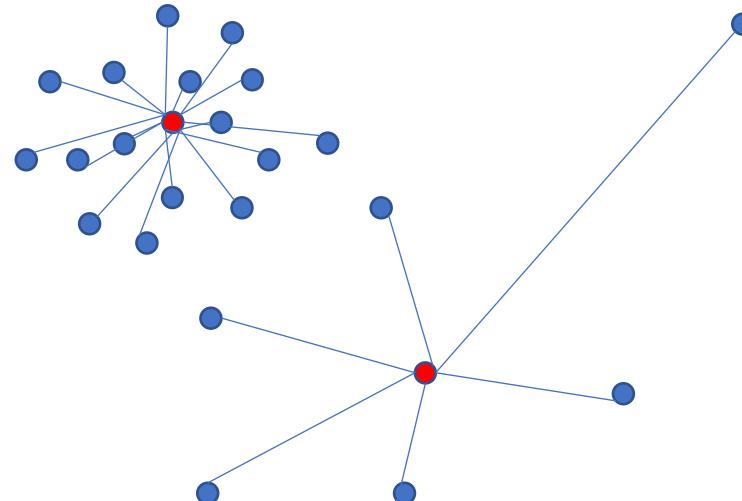
Node-to-Point kNN, k=13

- Method 1, Point-to-Node (SO-Net)

- Adaptive to scale and density
- Precise control of receptive field
 - Each point is connected k times.

每个数据点只会连到一个控制点

感知域容易控制，更准确，
outlier 也会被处理好



Point-to-Node kNN, k=1



SO-Net – Grouping & PointNet & Maxpool

