NUS National University of Singapore | School of Computing

CS4277 / CS5477
3D Computer Vision

Lecture 12:
3D Point Cloud Processing

Assoc. Prof. Lee Gim Hee

AY 2022/23

Semester 2

Course Schedule

Week	Date	Topic	Assignments
1	11 Jan	2D and 1D projective geometry	**Assignment 0:** Getting started with Python (Ungraded)
2	18 Jan	3D projective geometry, Circular points and Absolute conic	
3	25 Jan	Rigid body motion and Robust homography estimation	
4	01 Feb	Camera models and calibration	**Assignment 1:** Metric rectification and robust homography (10%) **Due:** 2359hrs, 07 Feb
5	08 Feb	Single view metrology	**Assignment 2**: Affine 3D measurement from vanishing line and point (10%) **Due:** 2359hrs, 14 Feb
6	15 Feb	The Fundamental and Essential matrices	
-	22 Feb	Semester Break	No lecture
7	**01 Mar**	**Mid-term Quiz (20%)** Lecture: Generalized cameras	**In-person Quiz (LT 15, 1900hrs – 2000hrs)** Lecture: 2000hrs – 2130hrs
8	08 Mar	Absolute pose estimation from points or lines	
9	15 Mar	Three-view geometry from points and/or lines	
10	22 Mar	Structure-from-Motion (SfM) and bundle adjustment	**Assignment 3:** SfM and Bundle adjustment (10%) **Due:** 2359hrs, 28 Mar
11	29 Mar	Two-view and multi-view stereo	**Assignment 4:** Dense 3D model from multi-view stereo (10%) **Due:** 2359hrs, 04 Apr
12	05 Apr	3D Point Cloud Processing	
13	12 Apr	Neural Field Representations	

Final Exam: 03 MAY 2023

Learning Outcomes

- Students should be able to:

1. Define implicit and explicit surfaces.
2. Explain the pro and cons of point clouds representation.
3. Describe how deep learning can be used on point cloud processing.
4. Define the tasks on 3D point cloud processing, i.e., place-recognition; keypoint detector and descriptor; 3D object detection; 3D semantic segmentation; point cloud registration; image-to-point cloud registration.

Acknowledgements

- A lot of slides and content of this lecture are adopted from:

1. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, CVPR 2017
2. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space, NeurIPS 2017
3. PointNetVLAD: Deep Point Cloud Based Retrieval for Large-Scale Place Recognition, CVPR 2018
4. USIP: Unsupervised Stable Interest Point Detection from 3D Point Clouds, ICCV 2019
5. 3DFeat-Net: Weakly Supervised Local 3D Features for Point Cloud Registration, ECCV 2018
6. Deep Hough Voting for 3D Object Detection in Point Clouds, ICCV 2019
7. Weakly Supervised Semantic Point Cloud Segmentation: Towards 10× Fewer Labels", CVPR 2020
8. RPM-Net: Robust Point Matching using Learned Features, CVPR2020.
9. DeepI2P: Image-to-Point Cloud Registration via Deep Classification, CVPR 2021

3D Surface Representations

- There are two major classes of surface representations: parametric (explicit) and implicit representations.

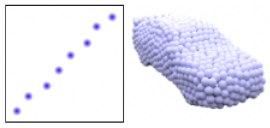
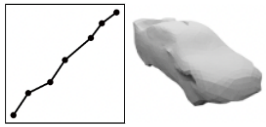
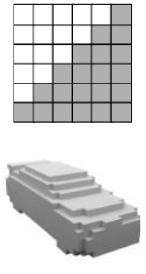
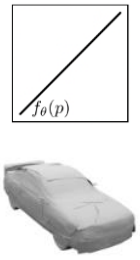
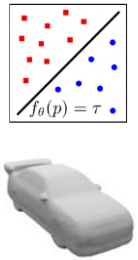
	Explicit	Implicit
Discrete	Point Cloud  Mesh 	Voxel 
Continuous	Neural Atlas 	Neural Field 

Image adapted from: L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy Networks: Learning 3D Reconstruction in Function Space," CVPR 2019

3D Surface Representations

- Parametric surfaces are defined by a vector valued parametrization function:

$$f: \Omega \mapsto S.$$

- $\Omega \in \mathbb{R}^2$ is the 2D parameter domain and $S \in \mathbb{R}^3$ is the 3D surface.

- **Intuition:** The $\mathbb{R}^2 \mapsto \mathbb{R}^3$ mapping is possible since all 3D surfaces are 2-manifolds.

- **Examples:** Point clouds, meshes, spline surfaces, atlas maps (next lecture).

3D Surface Representations

- Implicit or volumetric surface representation is defined to be the zero set of scalar-valued function: $F\colon \mathbb{R}^3 \mapsto \mathbb{R}$, i.e.:

$$S = \{\, \mathbf{x} \in \mathbb{R}^3 \mid F(\mathbf{x}) = 0 \,\}.$$

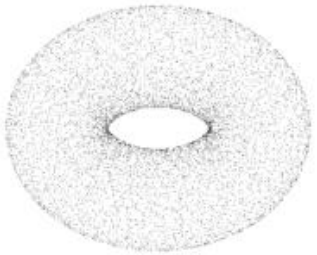
- $\mathbf{x} \in \mathbb{R}^3$ denotes a point in the 3D space, and $F(\mathbf{x}) = 0$ is the zero set.

- **Examples:** occupancy grids/voxels, signed distance fields (SDF), radiance fields, adaptive data structures (e.g. octree).

What is a 3D Point Cloud?

- Data structure for 3D representation: $N \times 3$ matrix or $N \times (3 + D)$, i.e.

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_N & y_N & z_N \end{bmatrix} \quad \begin{bmatrix} x_1 & y_1 & z_1 & d_{11} & \cdots & d_{1D} \\ x_2 & y_2 & z_2 & d_{21} & \cdots & d_{2D} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N & y_N & z_N & d_{N1} & \cdots & d_{ND} \end{bmatrix}$$

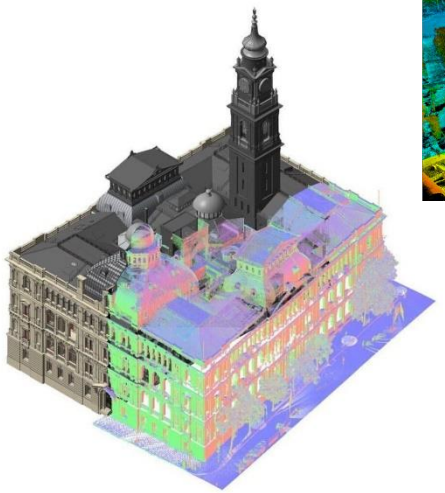


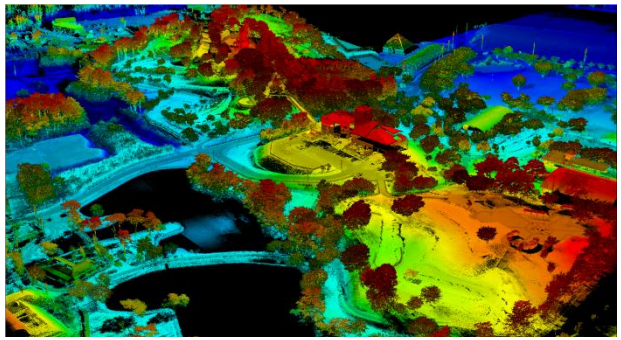
- D is addition information such as color, i.e., rgb.
- Data sources: LiDAR, RGB-D, CAD models, Structure-from-Motion (SfM), depth from images, etc.

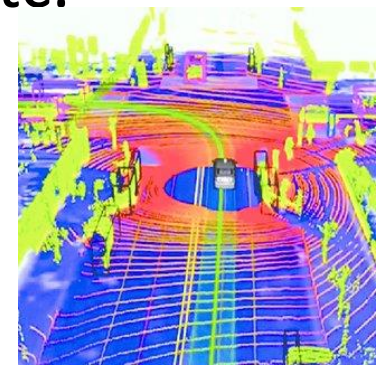
Image source: https://en.wikipedia.org/wiki/Point_cloud

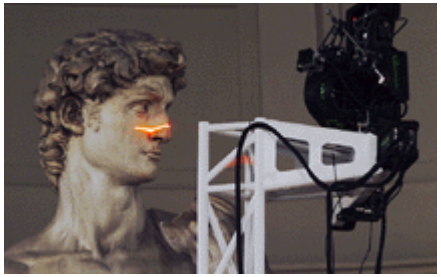
Applications of 3D Point Clouds

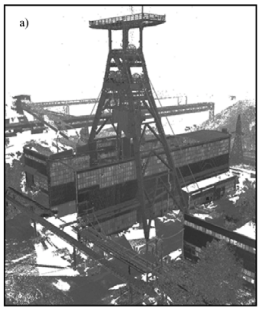
- Some important applications of 3D point clouds include: land surveying, urban planning, metrology, architecture, archeology , robotics, etc.











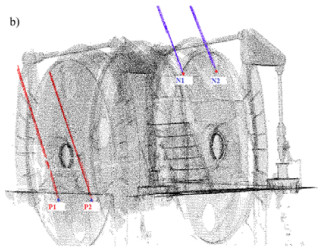


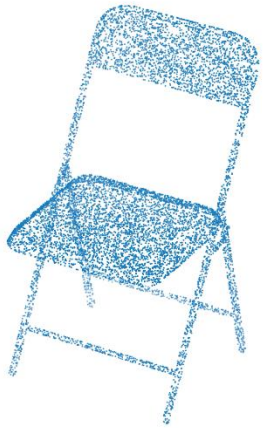
Image sources: https://measuredsurvey365.co.uk/services/point-cloud-surveys/
https://www.prodrone.com/usecases/drone-surveying-and-mapping/
https://www.researchgate.net/figure/A-point-cloud-image-of-a-vehicle-approaching-an-intersection-illustrates-the-complexity_fig1_305677214
https://www.researchgate.net/figure/Point-cloud-representing-the-a-headframe-and-b-hoisting-machine-P1-P2-N1_fig3_323965168
https://www.bricsys.com/blog/whats-new-for-point-clouds-in-bricscad-v22

Pros and Cons of 3D Point Clouds

- Strengths:
  1. 3D information with no depth ambiguity (*c.f.* images)
  2. Mathematically simple and concise

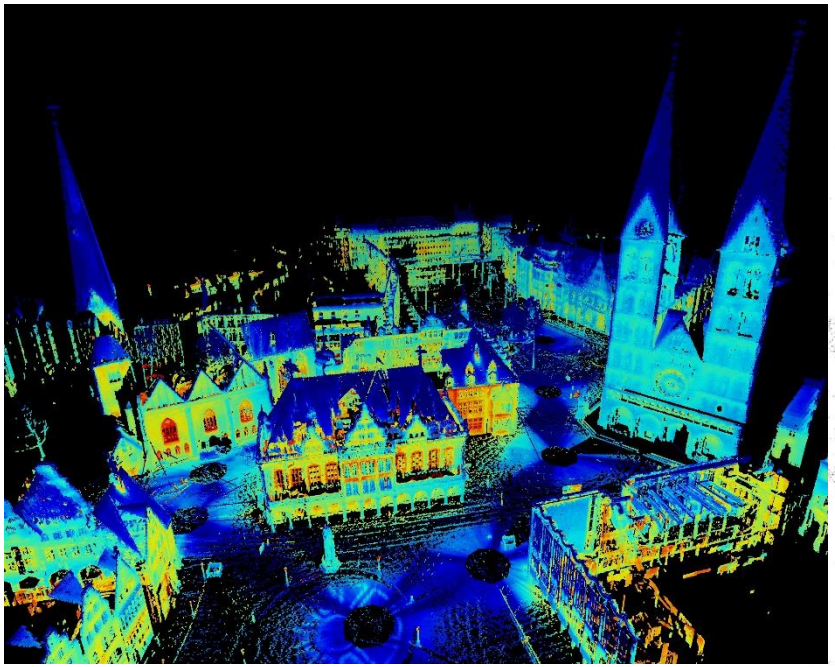
$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_N & y_N & z_N \end{bmatrix}$$

- Weaknesses:
  1. Sparsity
  2. Irregular – difficulty in neighbor search
  3. Lack of texture information
  4. Unordered – difficulty in deep learning
  5. Not rotation equivariant / invariant



Pros and Cons of 3D Point Clouds

- Lidar directly gives depth, which is more accurate than image-based reconstruction.



Lidar-based reconstruction

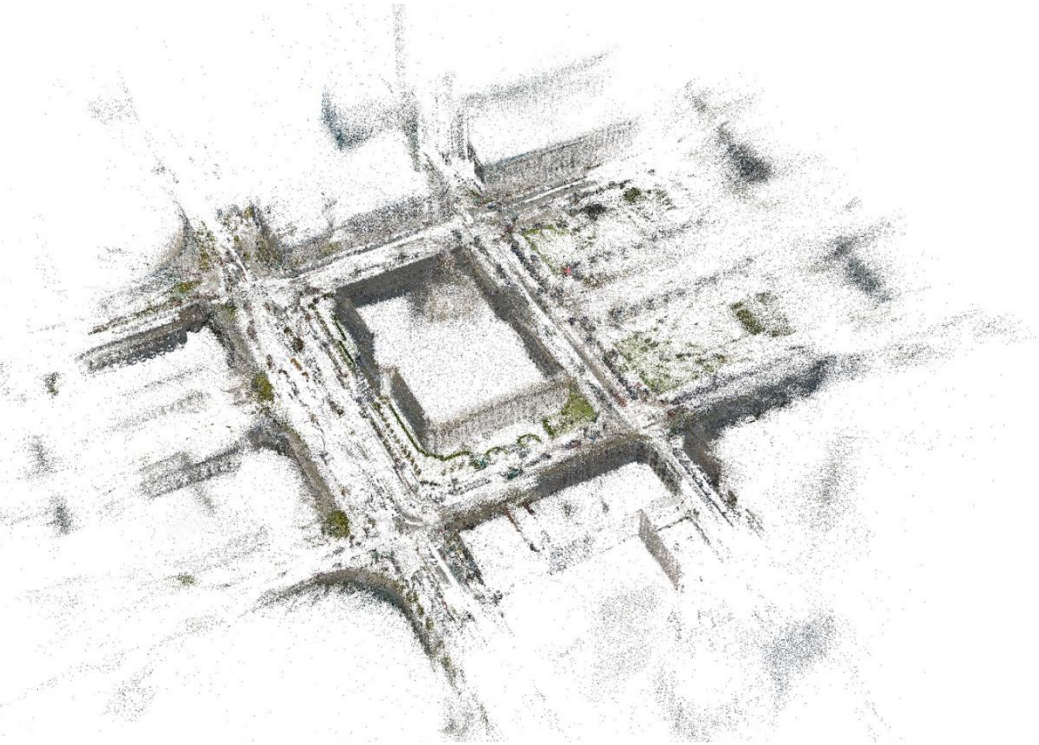


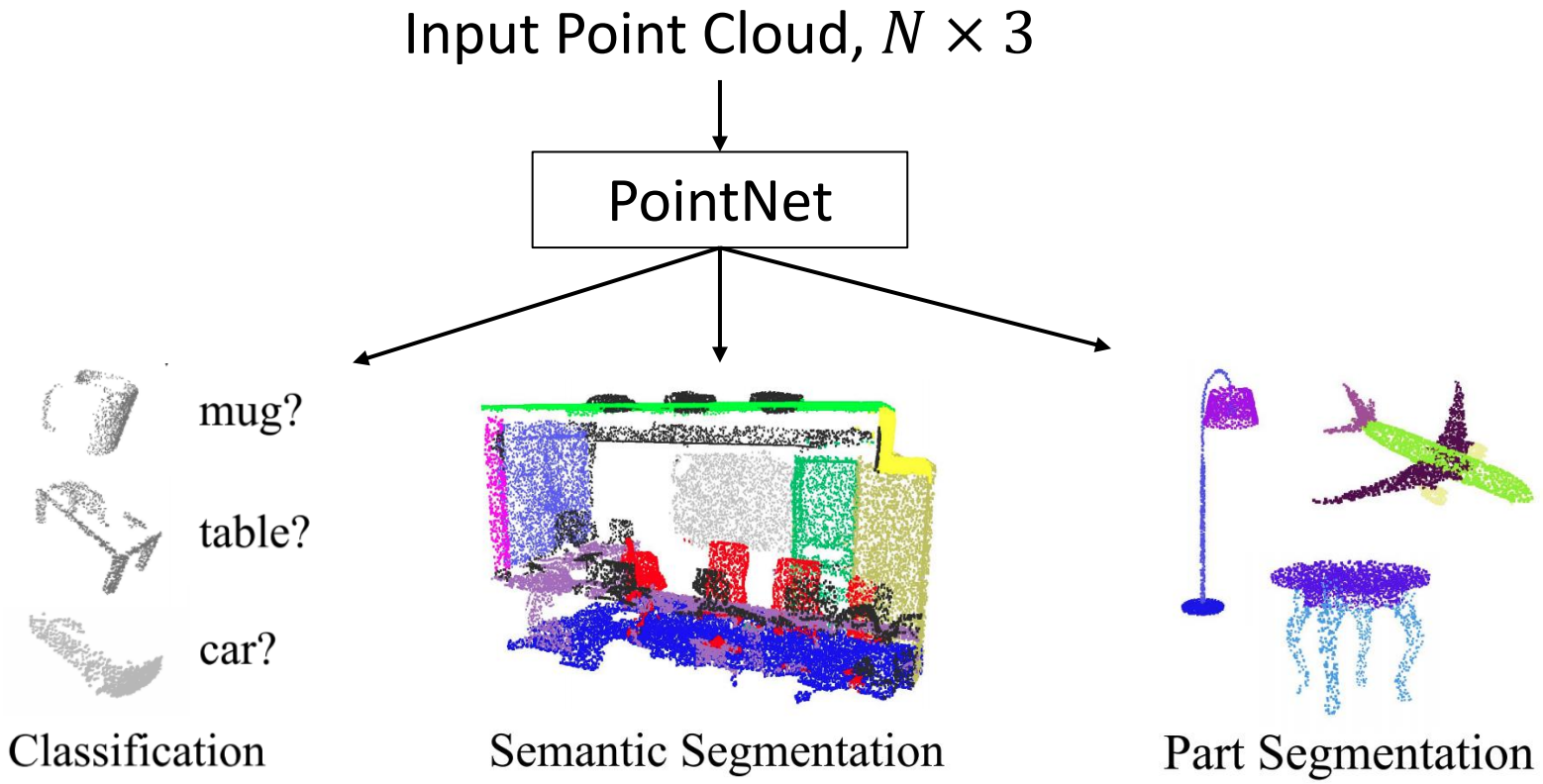
Image-based reconstruction

Image source: http://kos.informatik.uni-osnabrueck.de/3Dscans/

O. Saurer, et al, Sparse to Dense 3D Reconstruction from Rolling Shutter Images, CVPR 2016.

Deep Learning for 3D Point Clouds

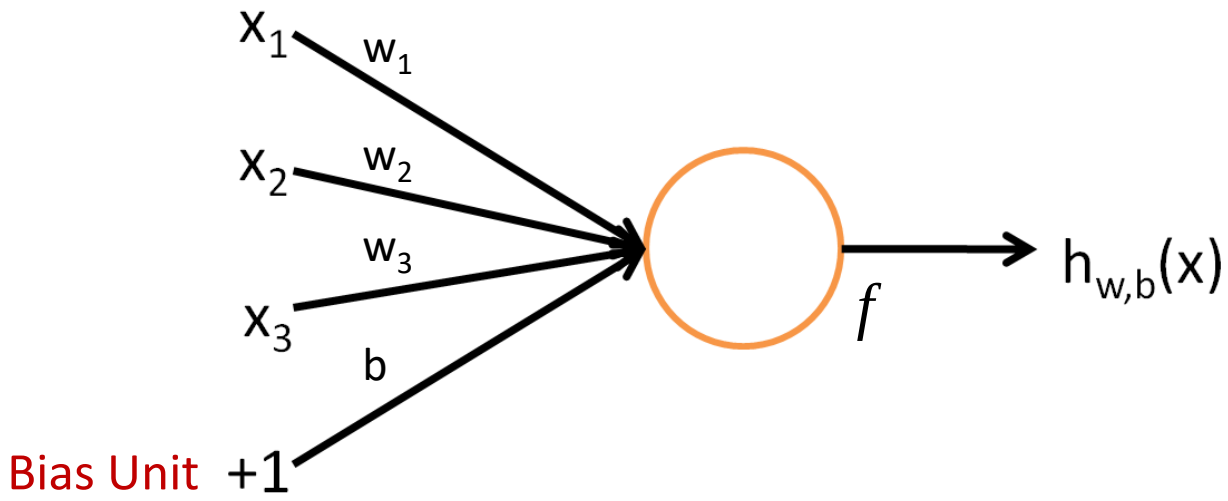
- The pioneering work: PointNet [Qi et. al, CVPR 2017]



PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, CVPR 2017

Problem with Neural Networks

Not Permutation Invariant!



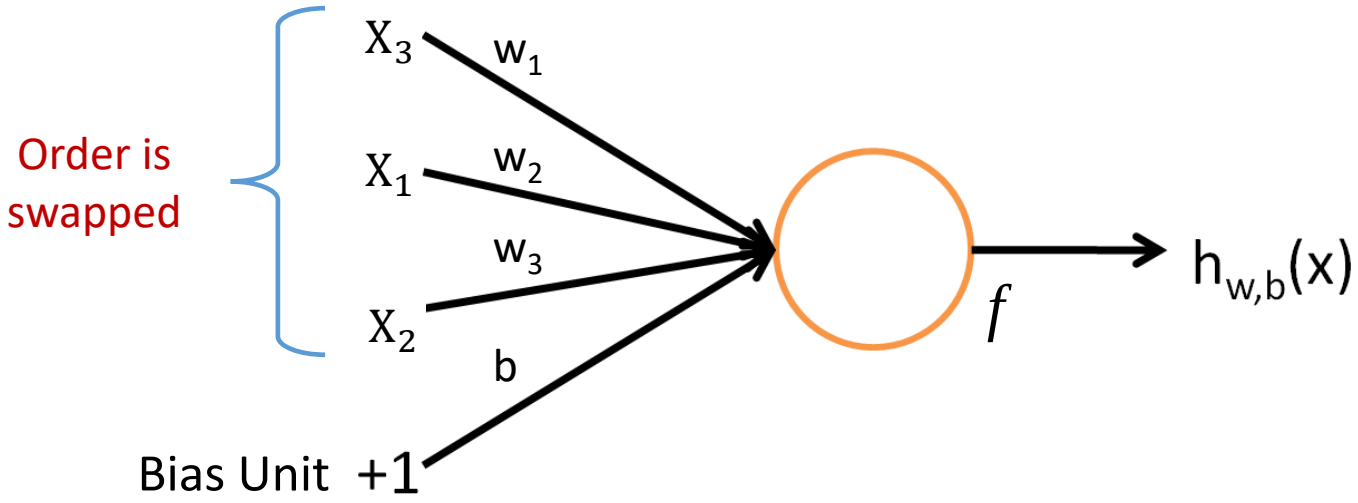
Activation function

$$h_{W,b}(x) = f(W^{\top}x) = f(\sum_{i=1}^{3} W_i x_i + b), \quad f: \mathbb{R} \mapsto \mathbb{R}$$

$$= w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

Problem with Neural Networks

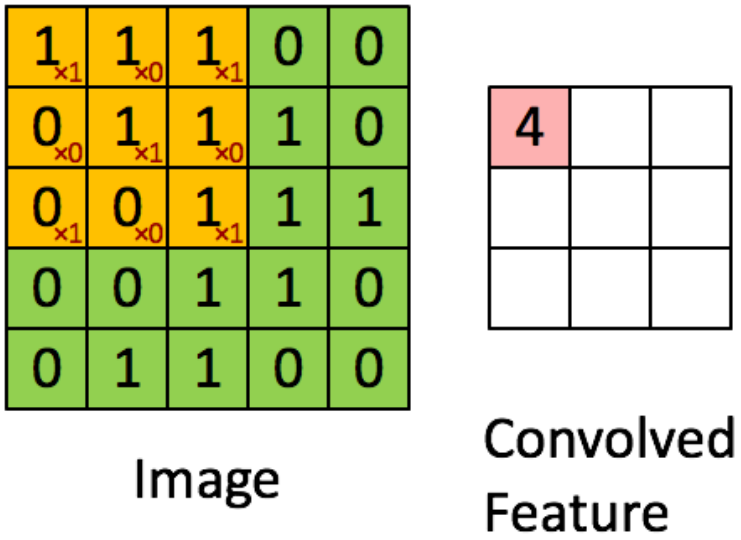
Not Permutation Invariant!



$$w_1 x_3 + w_2 x_1 + w_3 x_2 + b \neq h_{W,b}(x)$$

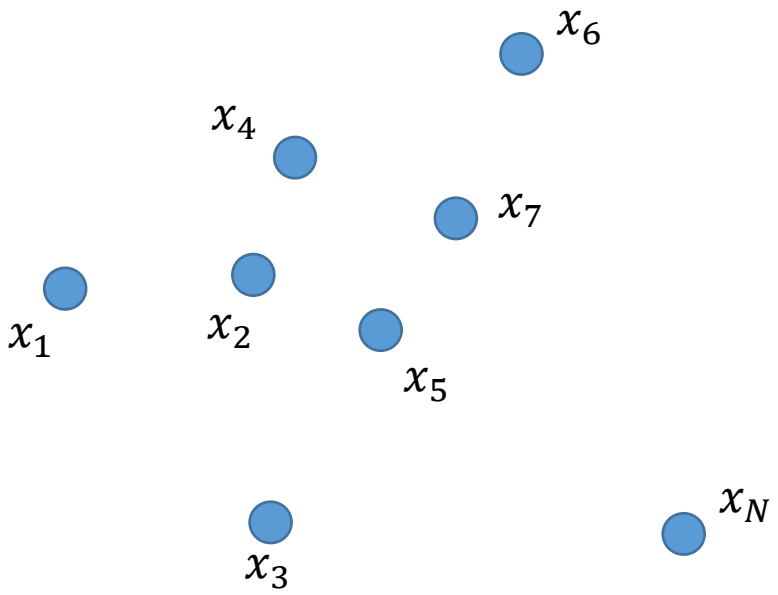
Regular Order of Image Pixels

- Non-permutation invariance is NOT a problem for images.
- Image pixels are arranged in regular order.



Point Cloud is Orderless!

$$S = \{\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_N\},\ \mathbf{x}_n \in \mathbb{R}^2 \text{ or } \mathbf{x}_n \in \mathbb{R}^3$$



$$\mathbf{X}_1 = [\mathbf{x}_1^\top, \mathbf{x}_2^\top, \dots, \mathbf{x}_N^\top]^\top$$

$$\mathbf{X}_2 = [\mathbf{x}_2^\top, \mathbf{x}_1^\top, \dots, \mathbf{x}_N^\top]^\top$$

$$\vdots$$

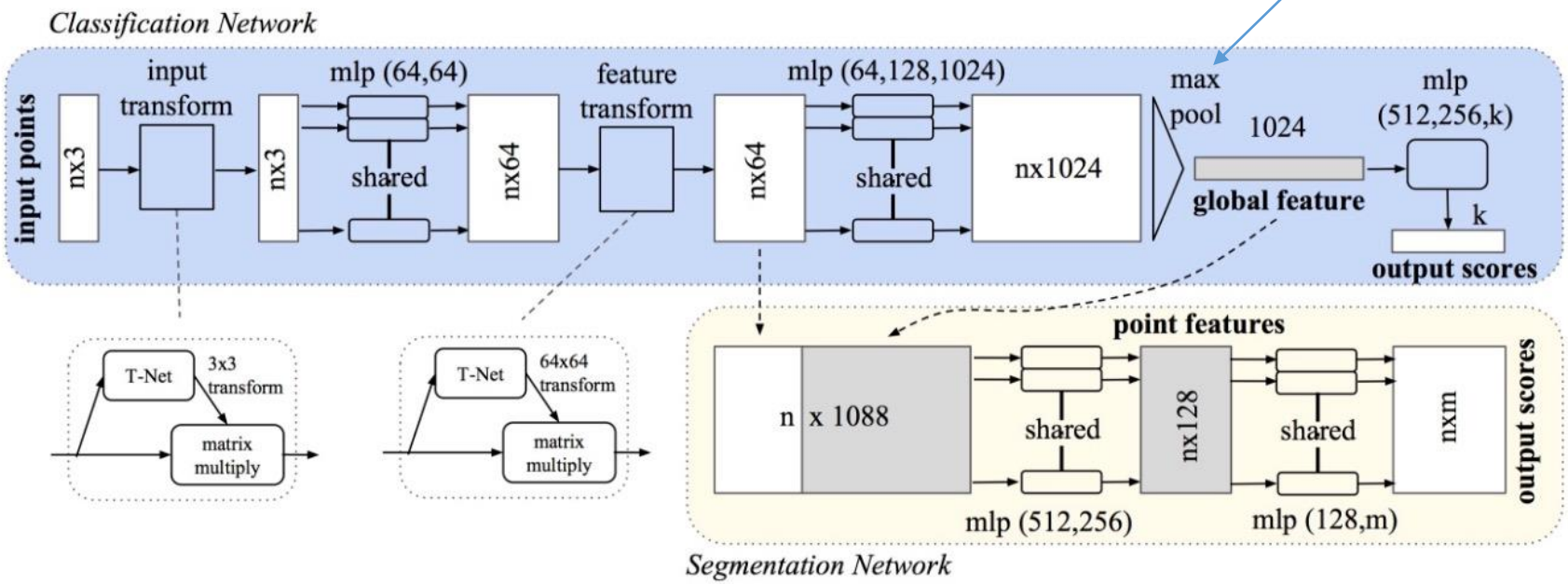
$$\mathbf{X}_{N!} = [\mathbf{x}_i^\top, \mathbf{x}_j^\top, \dots, \mathbf{x}_n^\top]^\top$$

$N!$ possibilities

PointNet: Pioneer Work

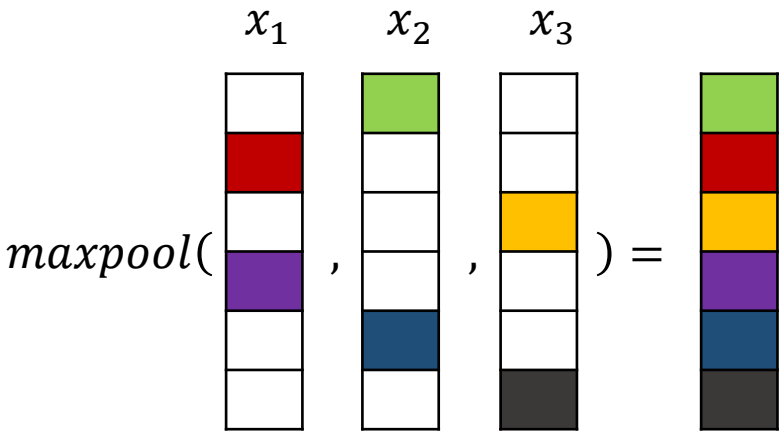
Points are treated independently

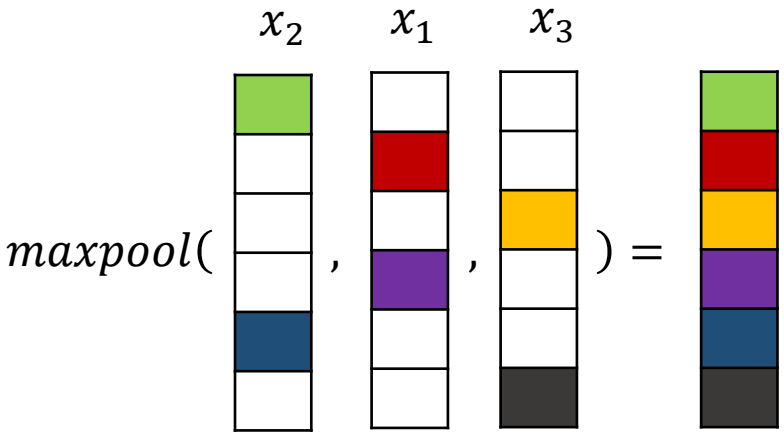
Maxpool operation is permutation invariant



Charles Qi et. al., CVPR 2017

Maxpool Operation: Permutation Invariance



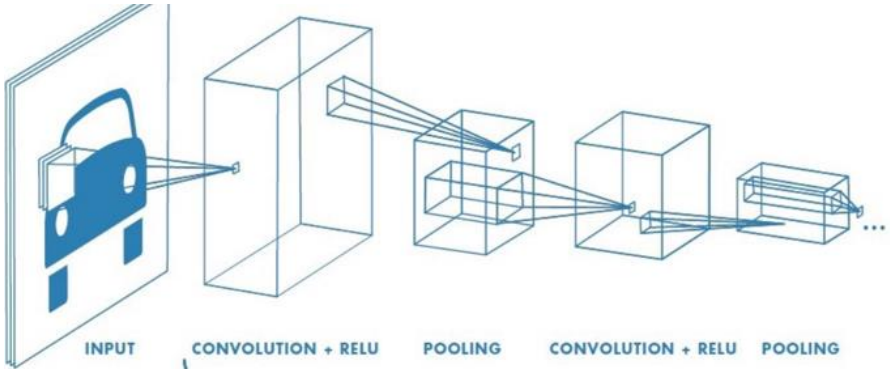


Same results regardless of input order!

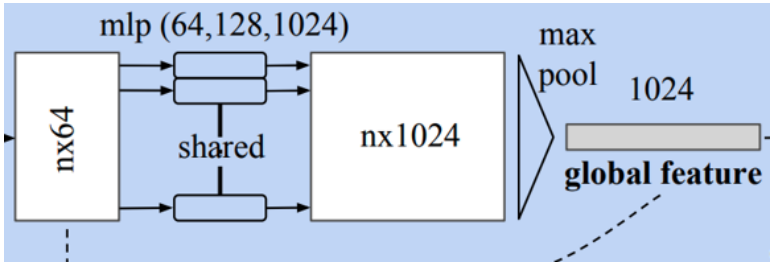
Limitations of PointNet

- Lack of hierarchical feature aggregation:
  - CNN has multiple, increasing receptive field
  - PointNet has one receptive field – all points

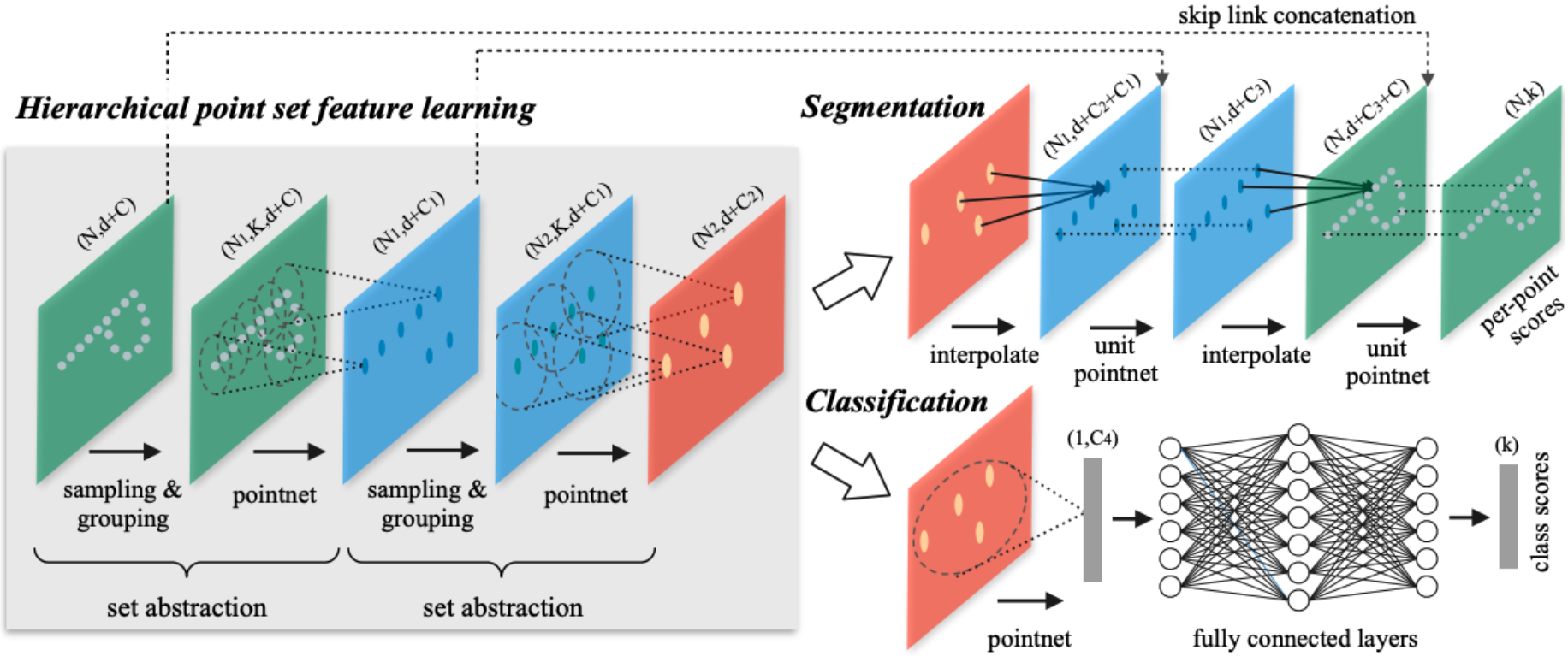
CNN



PointNet



PointNet++: Hierarchical Features



Qi el at, PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space, NeurIPS 2017

PointNet++: Hierarchical Features

Furthest point sampling (FPS):

Given input points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, we choose a subset of points $\{\mathbf{x}_{i1}, \mathbf{x}_{i2}, \ldots, \mathbf{x}_{im}\}$, such that $\mathbf{x}_{ij}$ is the most distant point from the set $\{\mathbf{x}_{i1}, \mathbf{x}_{i2}, \ldots, \mathbf{x}_{im}\} \backslash \mathbf{x}_{ij}$.

Grouping Layer:

Each group corresponds to a local region of K points in the neighborhood of the FPS points as centroids.

PointNet++: Hierarchical Features

PointNet Layer:

Translate points in local regions into a local frame relative to the centroid point: $\mathbf{x}_i = \mathbf{x}_i - \hat{\mathbf{x}}$ for $i = 1, 2, \ldots, K$, where $\hat{\mathbf{x}}$ is the coordinate of the centroid.

Multi-resolution grouping (MRG):

Concatenate the multi-scale feature vectors.

Concatenate

Simple PN++

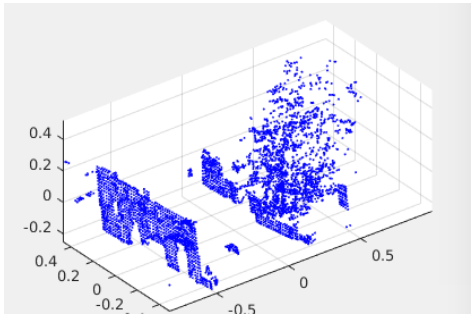
MRG PN++

Various Point Cloud-Based Tasks

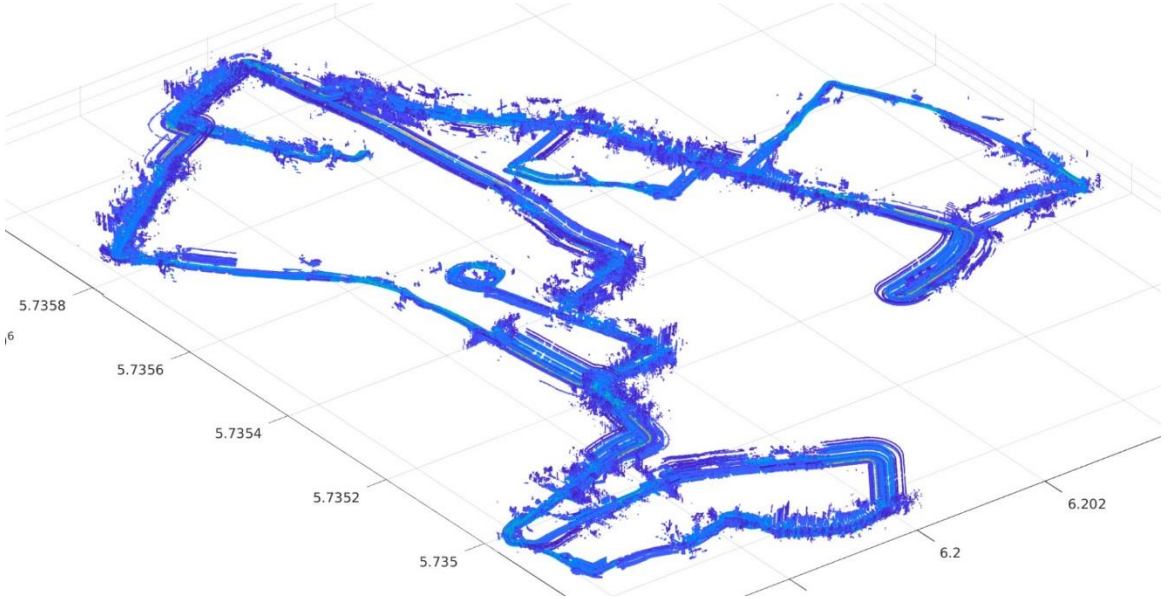
1. Point cloud-based retrieval (place-recognition)
2. Keypoint Detector and Descriptor
3. 3D Object Detection
4. 3D Semantic Segmentation
5. Point cloud registration
6. Image-to-point cloud registration.

Point Cloud-Based Place Recognition

- Where am I in the point cloud-based map?



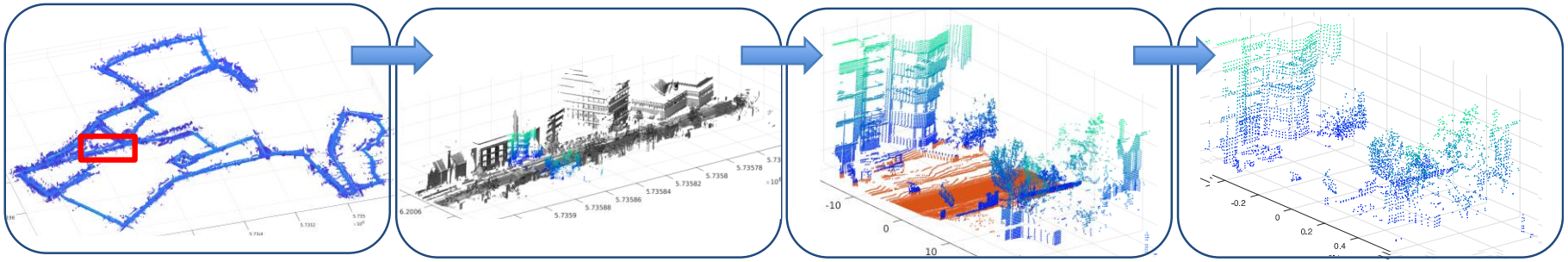
Query Point Cloud



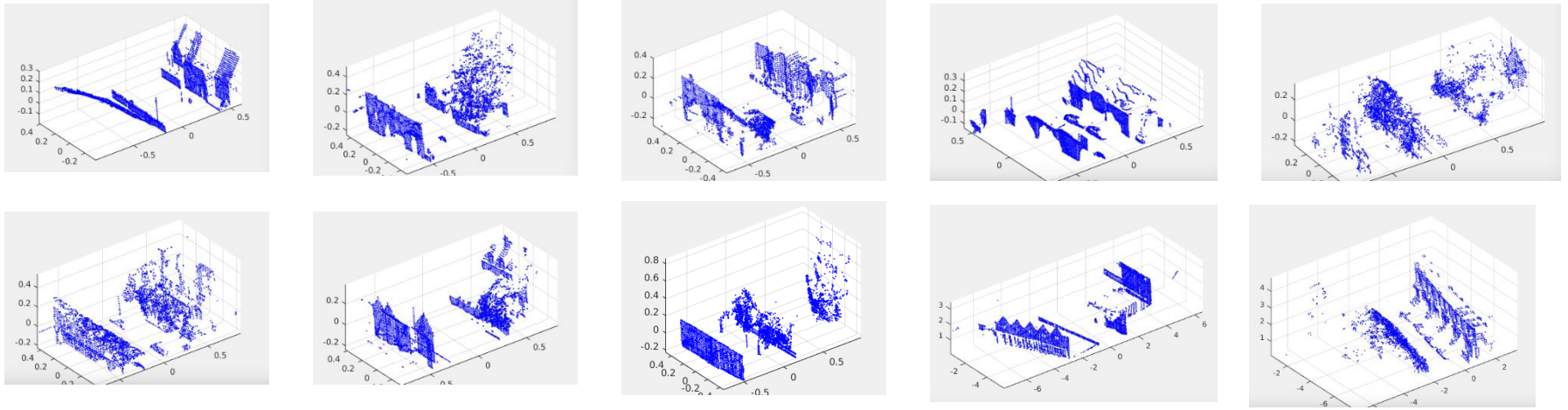
3D Point Cloud Map

Point Cloud-Based Place Recognition

- Create a database of submaps that are geo-tagged:

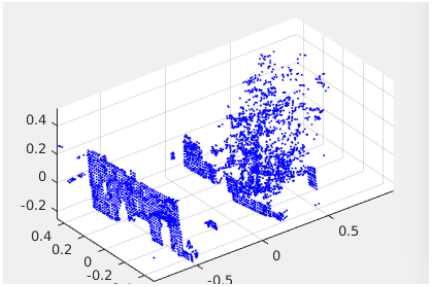


(a) Full route (b) Zoomed-in region (c) Submap (d) Downsampled

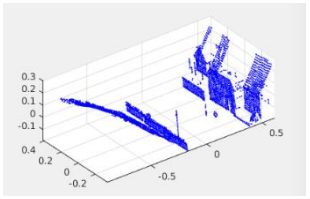
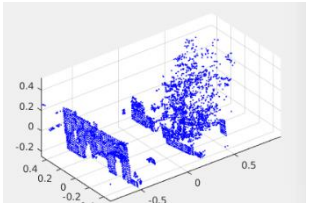
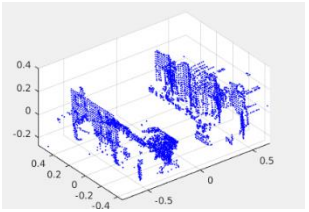
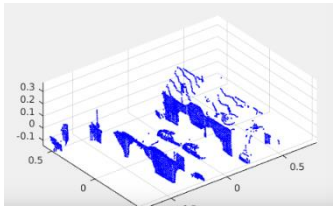
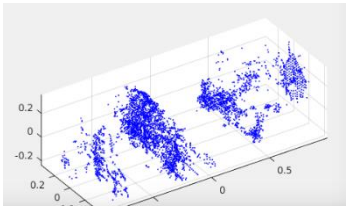
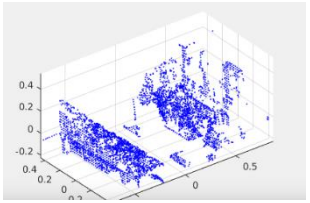
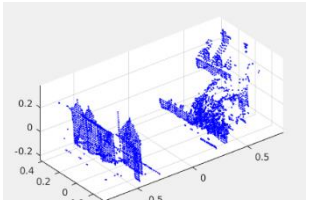
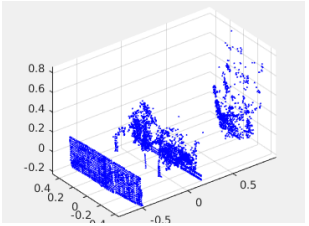
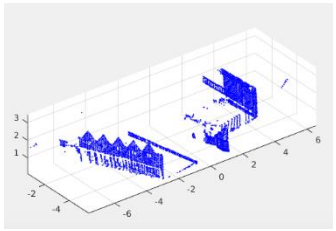
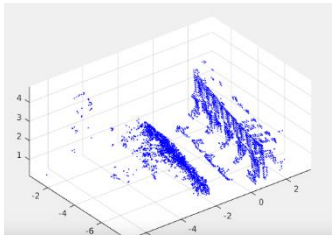


Point Cloud-Based Place Recognition

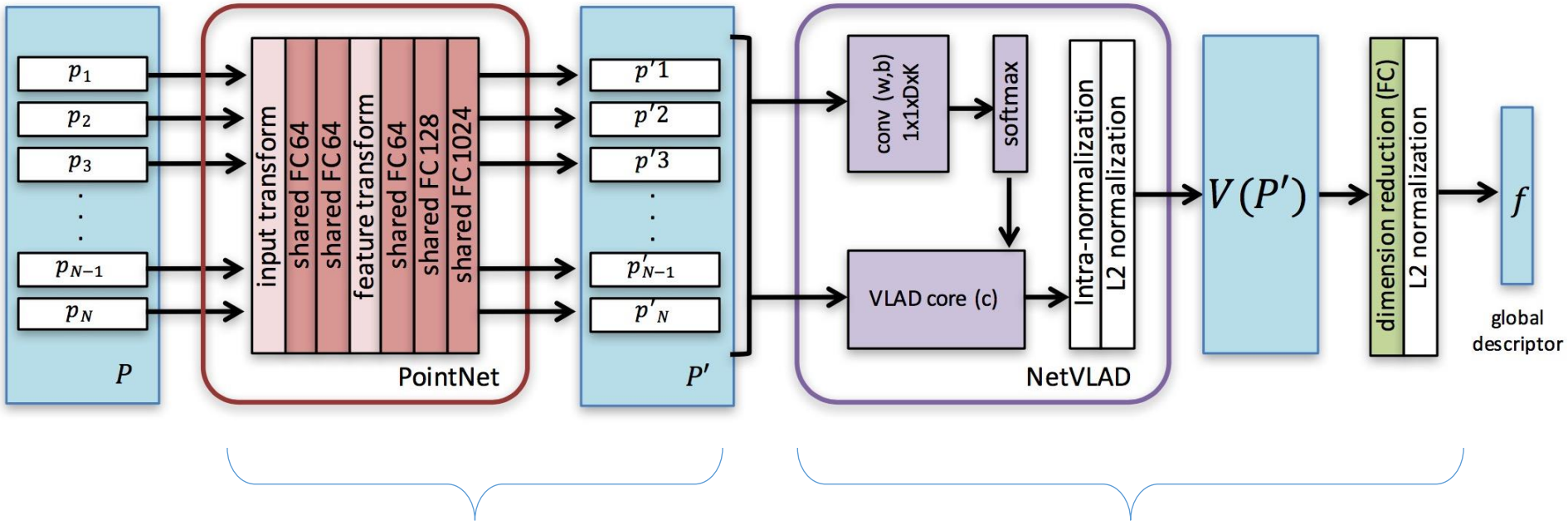
Given: Query Submap



Find: The most similar submap in the database

PointNetVLAD



Extract local features

Aggregate local features into a global descriptor

PointNetVLAD: Deep Point Cloud Based Retrieval for Large-Scale Place Recognition
Mikaela Angelina Uy, Gim Hee Lee, CVPR 2018

Proof of Permutation Invariance

Thereom: *NetVLAD is a symmetric function*

Proof: Given feature representation of input point cloud P as $\{p'_1, p'_2, \ldots, p'_N\}$, we have output vector $V = [V_1, V_2, \ldots, V_K]$ of NetVLAD such that $\forall k$,

$$V_k = h_k(p'_1) + h_k(p'_2) + \ldots + h_k(p'_K) = \sum_{t=1}^{K} h_k(p'_t),$$

where

$$h_k(p') = \frac{e^{w_k^T p' + b_k}}{\sum_{k'} e^{w_{k'}^T p' + b_{k'}}}(p' - c_k).$$

Proof of Permutation Invariance

- Suppose we have another point cloud similar to P except for reordered points p_i and p_j, i.e. feature representation is:

$$\{p'_1, \ldots, p'_{i-1}, p'_j, p'_{i+1}, \ldots, p'_{j-1}, p'_i, p'_{j+1}, \ldots, p'_N\}.$$

- Hence $\forall k$, we have

$$\begin{aligned}\tilde{V}_k =& h_k(p'_1) + \ldots + h_k(p'_{i-1}) + \\ & h_k(p'_j) + h_k(p'_{i+1}) + \ldots + h_k(p'_{j-1}) + \\ & h_k(p'_i) + h_k(p'_{j+1}) + \ldots + h_k(p'_N) \\ =& \sum_{t=1}^{K} h_k(p'_t) = V_k.\end{aligned}$$

- Thus, $f(P) = f(\tilde{P})$ and completes our proof for symmetry. □

Training: Metric Learning

- Our PointNetVLAD is trained end-to-end to learn $f(.)$ that maps P to a discriminative compact global descriptor vector $f(P) \in \mathbb{R}^O$, where $\| f(P) \|^2 = 1$.

- Training data are a set of tuples:

$$\mathcal{T} = (P_a, P_{pos}, \{P_{neg}\})$$

P_a : an anchor point cloud

P_{pos} : a structurally similar ("positive") point cloud to P_a

P_{neg} : a set of structurally dissimilar ("negative") point clouds to P_a

Training: Metric Learning

- Lazy Triplet Loss:

$$\mathcal{L}_{lazyTrip}(\mathcal{T}) = \max_j([\alpha + \delta_{pos} - \delta_{neg_j}]_+)$$

- Lazy Quadruplet Loss:

$$\mathcal{L}_{lazyQuad}(\mathcal{T}, P_{neg^*}) = \max_j([\alpha + \delta_{pos} - \delta_{neg_j}]_+) + \max_k([\beta + \delta_{pos} - \delta_{neg_k^*}]_+)$$

where

$$\delta_{pos} = d(f(P_a), f(P_{pos}))$$

$$\delta_{neg_j} = d(f(P_a), f(P_{neg_j}))$$

PointNetVLAD: Results

	Training^{+}		Test$^{\times}$	
	Baseline	Refine	Baseline	Refine
Oxford	21711	21711	3030	3030
U.S.	-	8442	400*	80*
R.A.	-		320* 4542	75* 1766
B.D.	-	-	200*	200*

Table 1. Number of training and testing submaps for our baseline and refined networks. *approximate number of submaps/run is given because the number of submaps differ slightly between each run; $^{+}$overlapping and $^{\times}$disjoint submaps.

PointNetVLAD: Results

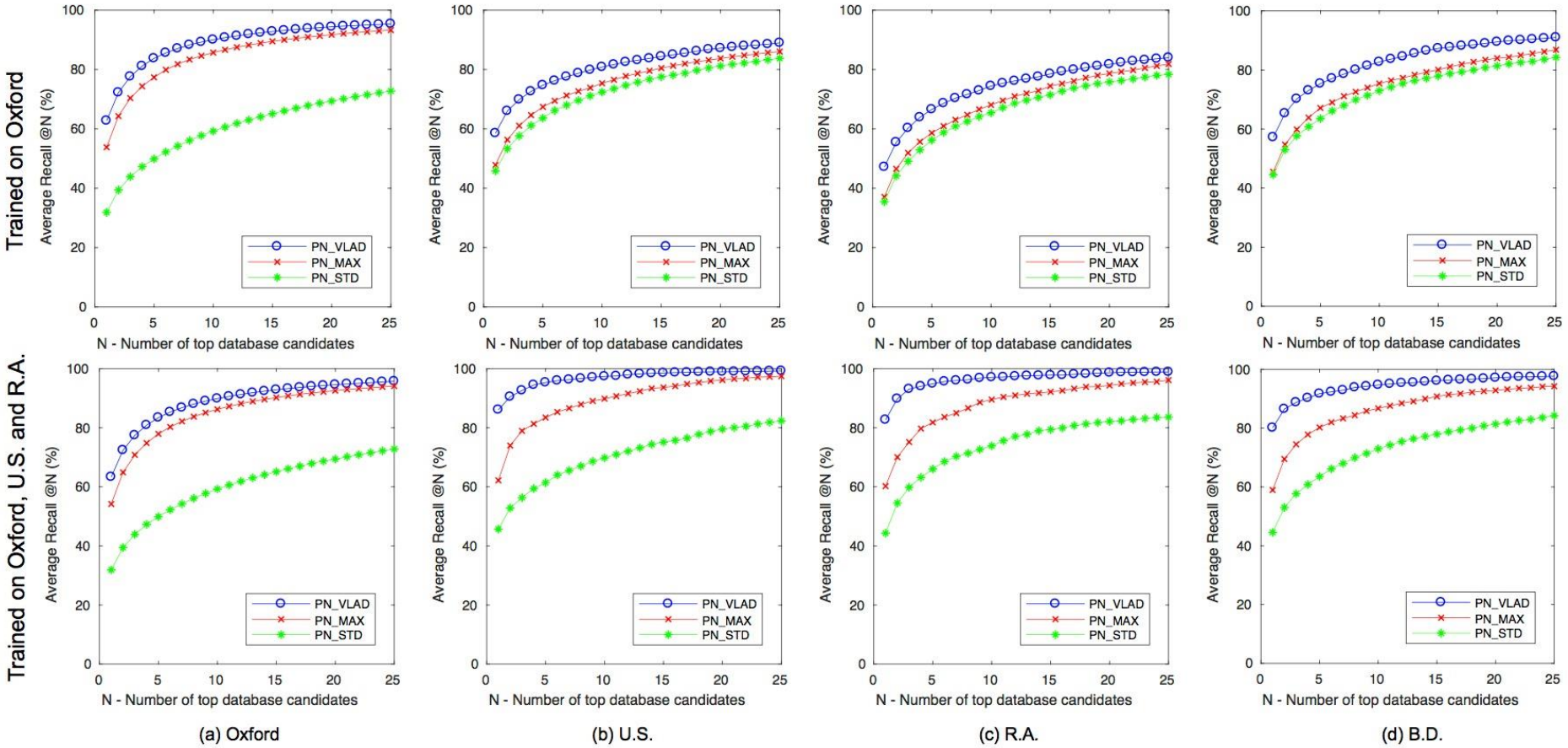
Baseline results showing the average recall (%) at top 1% for each of the models.

	PN_VLAD	PN_MAX	PN_STD
Oxford	**80.31**	73.44	46.52
U.S.	**72.63**	64.64	61.12
R.A.	**60.27**	51.92	49.07
B.D.	**65.30**	54.74	53.02

Final results showing the average recall (%) at top 1% (@1%) and at top 1 (@1) after training on Oxford, U.S. and R.A.

	Ave recall @1%			Ave recall@1		
	PN_ VLAD	PN_ MAX	PN_ STD	PN_ VLAD	PN_ MAX	PN_ STD
Ox.	80.09	73.87	46.52	63.33	54.16	31.87
U.S.	90.10	79.31	56.95	86.07	62.16	45.67
R.A.	93.07	75.14	59.81	82.66	60.21	44.29
B.D.	**86.49**	69.49	53.02	**80.11**	58.95	44.54

PointNetVLAD: Results



Average recall of the networks. Top row shows the average recall when PN_VLAD and PN_MAX were only trained on Oxford. Bottom row shows the average recall when PN_VLAD and PN_MAX were trained on Oxford, U.S. and R.A.

PointNetVLAD: Results

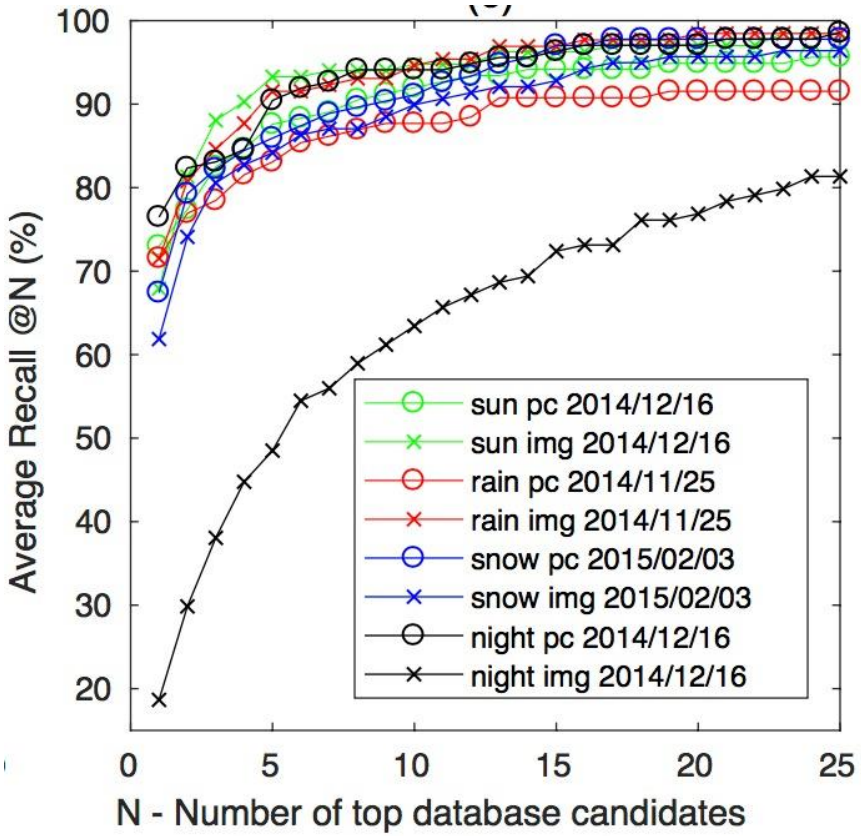
Average recall (%) at top 1% on the different datasets for output dimensionality analysis of PN_VLAD and PN_MAX.

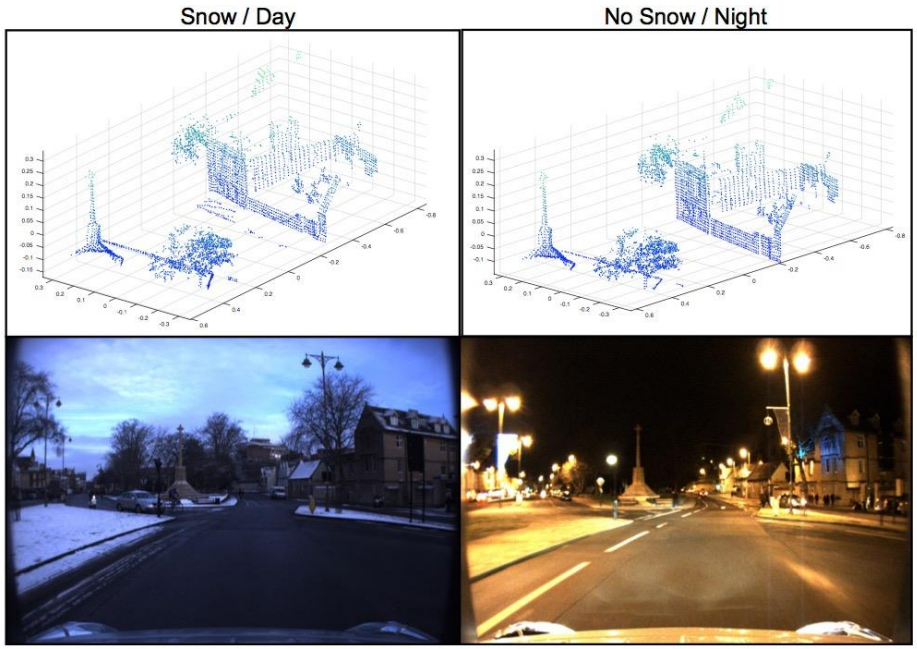
	PN_VLAD			PN_MAX		
	D-128	D-256	D-512	D-128	D-256	D-512
Ox.	**74.60**	80.31	80.33	71.93	73.44	**74.79**
U.S.	**66.03**	72.63	76.24	61.15	64.64	**65.79**
R.A.	**53.86**	60.27	63.31	49.25	51.92	**52.32**
B.D.	**59.84**	65.30	66.75	53.25	54.74	**56.63**

Results representing the average recall (%) at top1% of PN_VLAD tested and trained using different losses on Oxford.

	Average recall
Triplet Loss	71.20
Quadruplet Loss	74.13
Lazy Triplet Loss	**78.99**
Lazy Quadruplet Loss	**80.31**

PointNetVLAD: Results

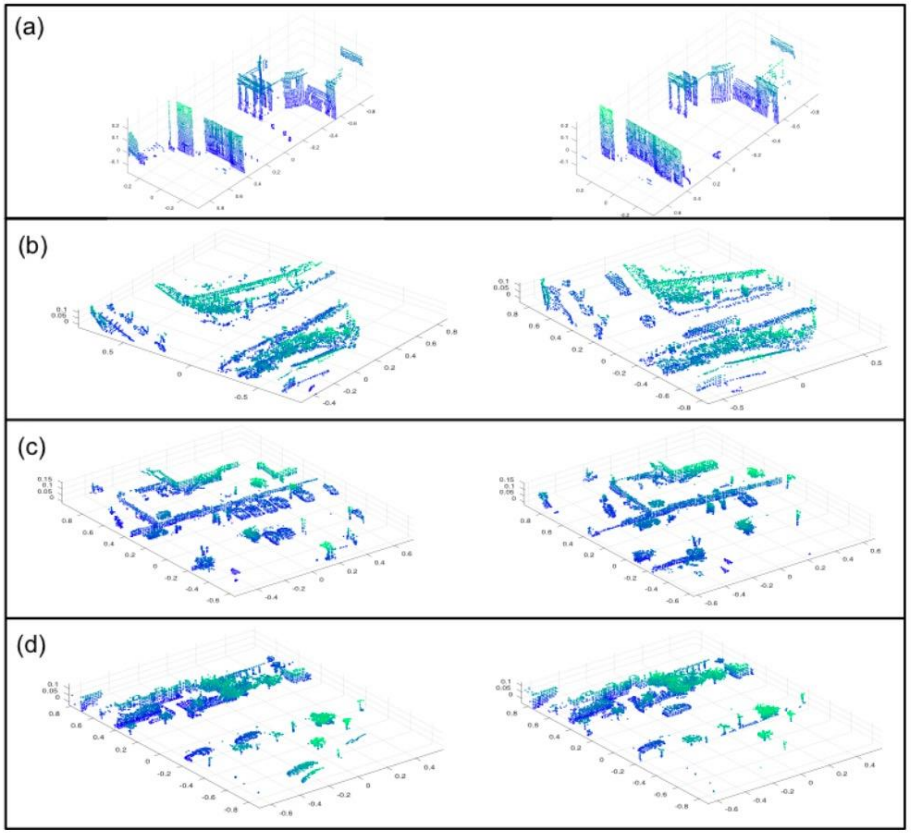




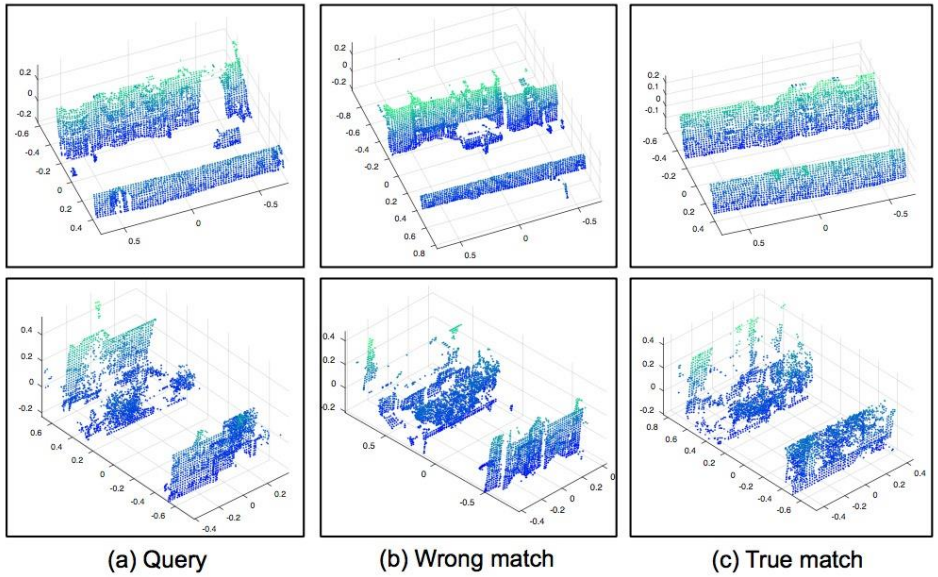
Average recall @N with point clouds (pc) and images (img) as queries under various scene conditions, and retrieving from an overcast database in Oxford dataset.

PointNetVLAD: Results

Examples of successful retrievals



Examples of wrong retrievals

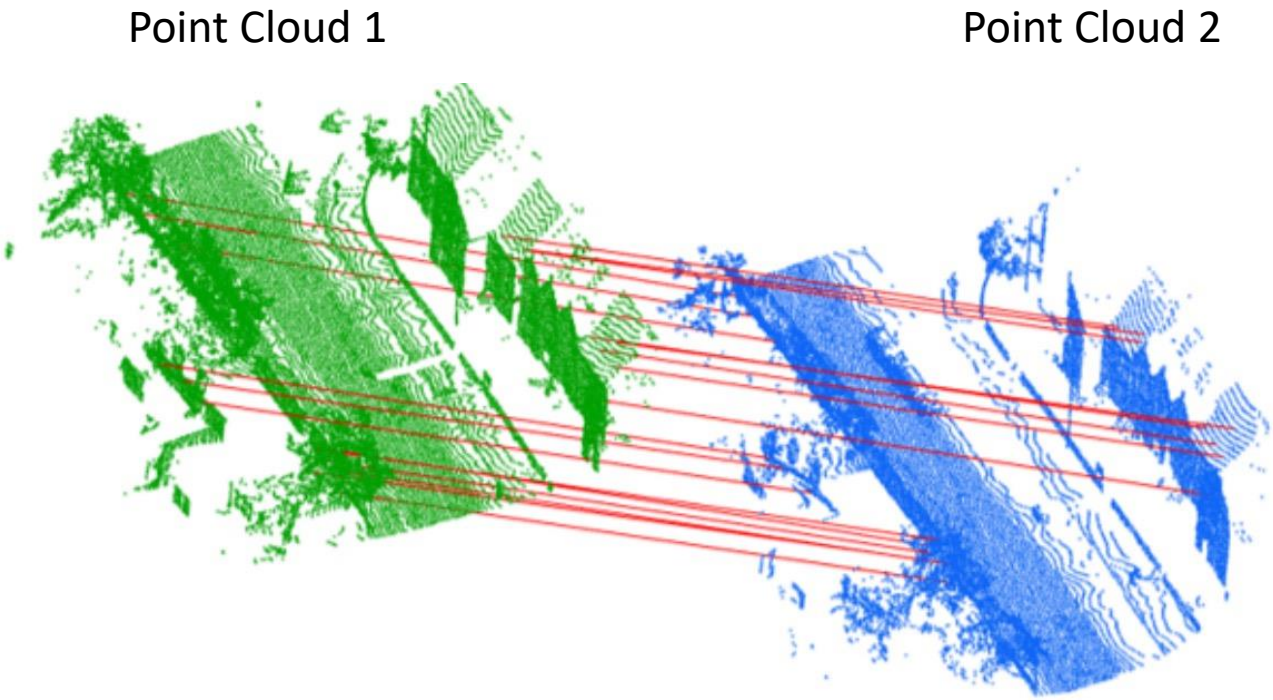


Keypoint Detector and Descriptor

Image 1 Image 2

- Image-based keypoint detectors and descriptors, e.g. SIFT, give us good correspondences.

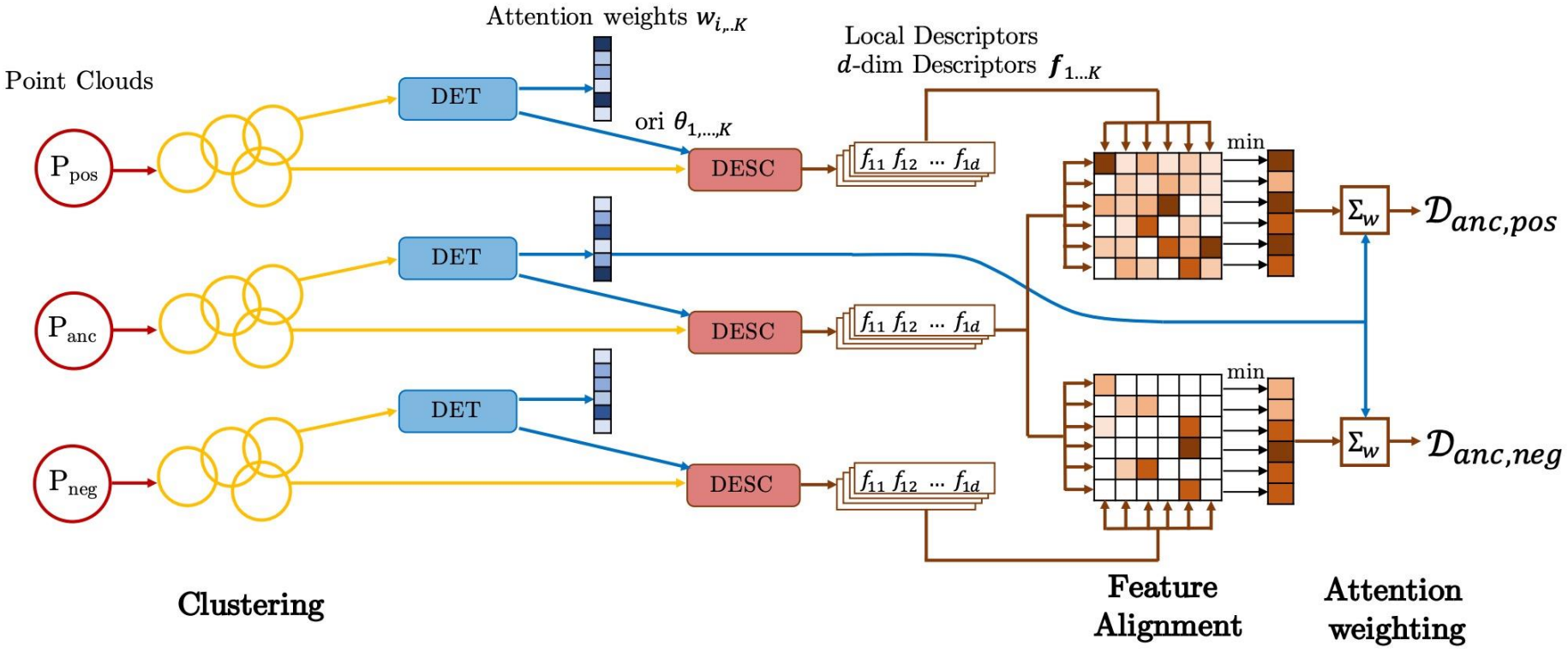
Keypoint Detector and Descriptor



- Can we also design a keypoint detector and descriptor to find correspondences in point clouds?

3DFeat-Net: Network Architecture

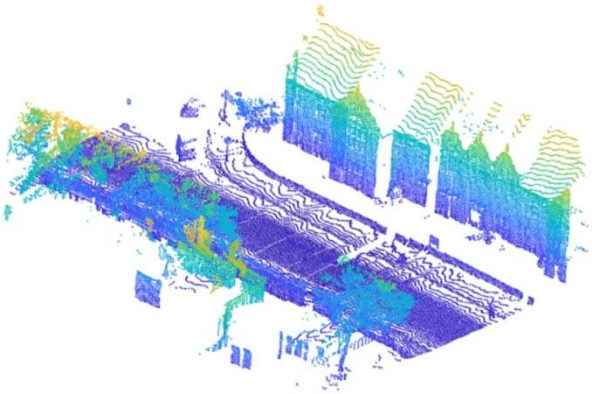
- Weakly supervised Triplet Network



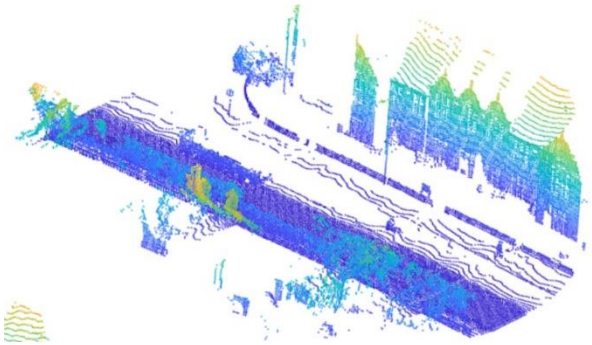
3DFeat-Net: Weakly Supervised Local 3D Features for Point Cloud Registration
Zi Jian Yew, Gim Hee Lee, ECCV 2018

Weakly supervised Triplet Network

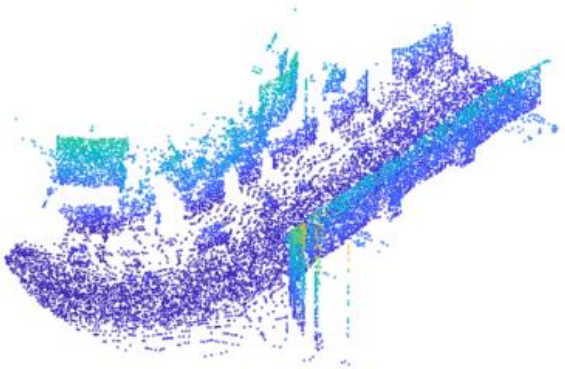
- We just need to know the approximate GPS/INS location of each point cloud P_i.
- Form training tuple: $\{P_{pos}, P_{anc}, P_{neg}\}$.



P_{anc}: anchor point cloud
Taken at time X



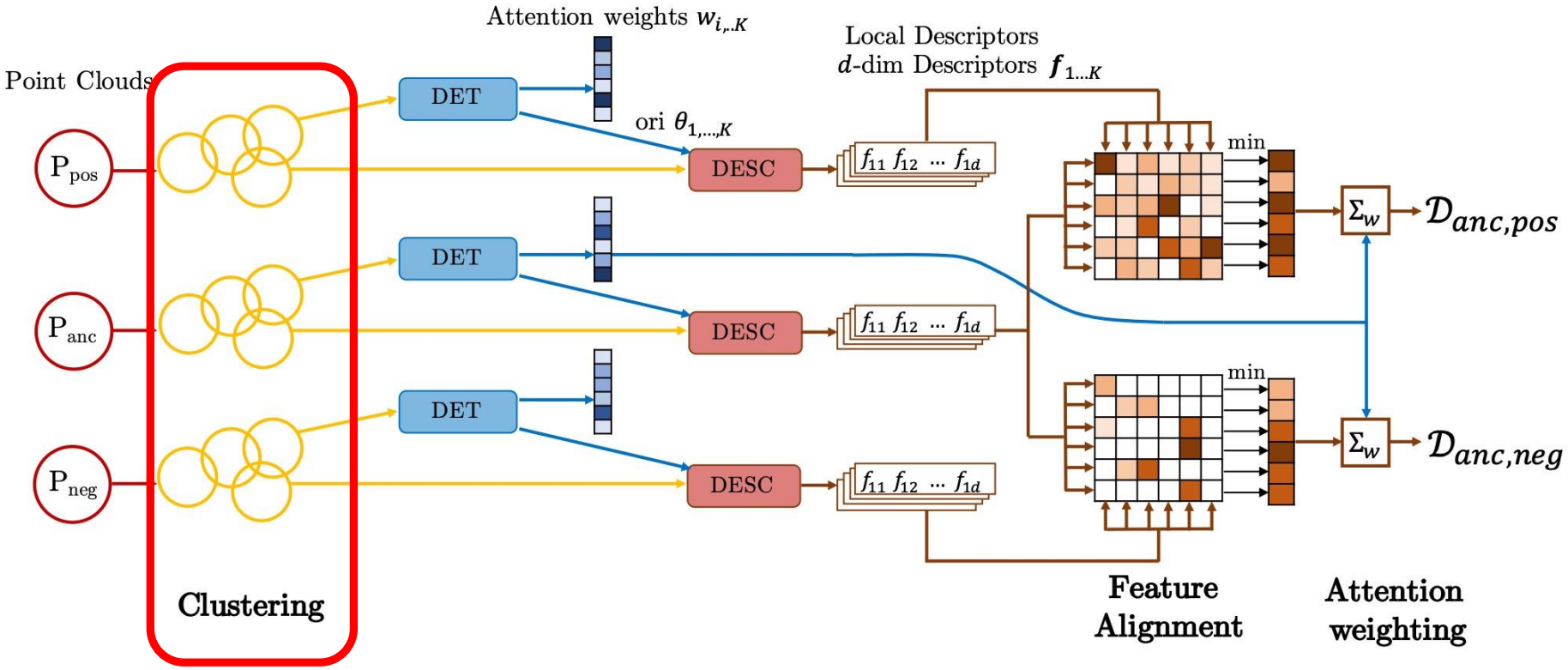
P_{pos}: positive point cloud
Taken at time Y
$\sim$ same location as P_{anc}



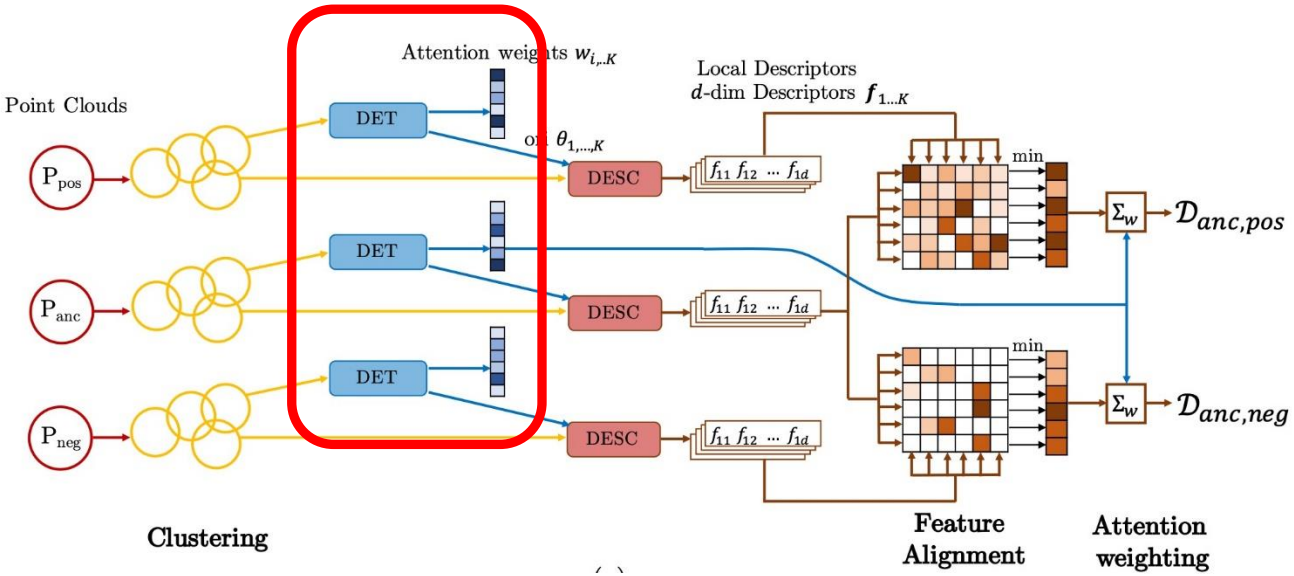
P_{neg}: negative point cloud
Taken at time Y
Different location as P_{anc}

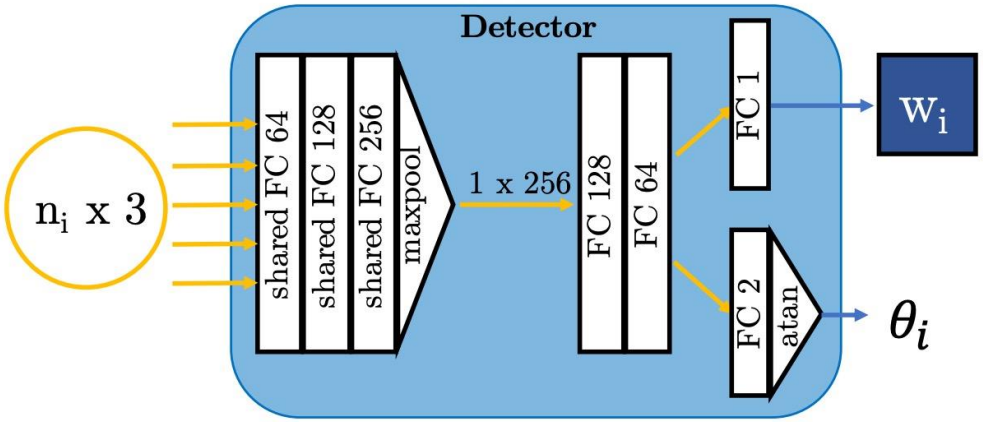
3DFeat-Net: Clustering

- Sample centroids of clusters with iterative farthest point.
- Neighboring points within a predefined radius to each centroid are used to form a cluster.



3DFeat-Net: Keypoint Detector



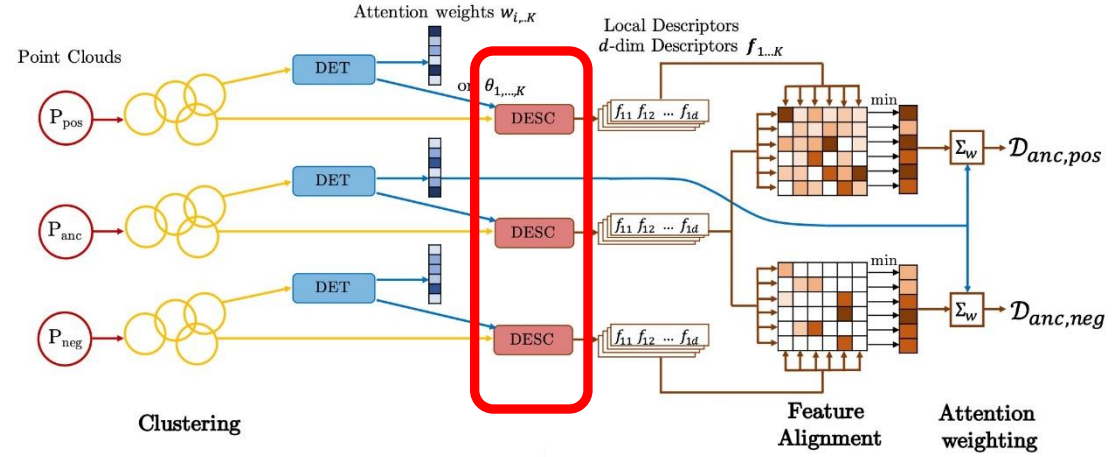


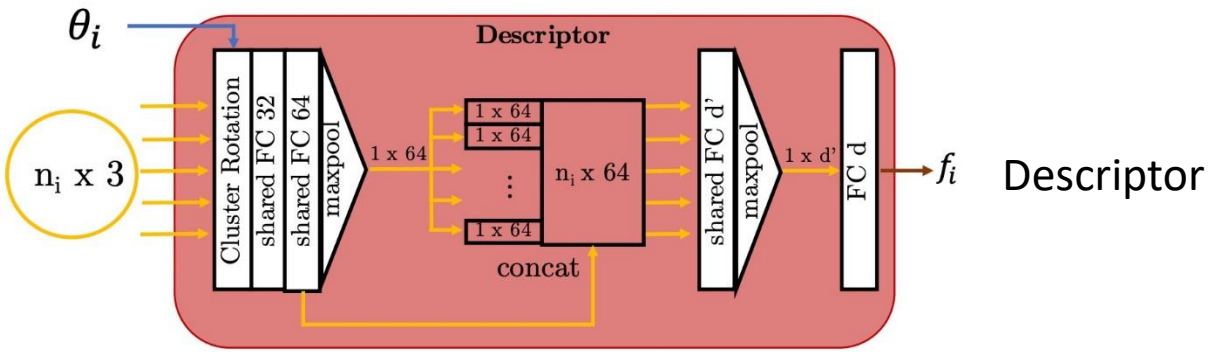
Attention Weight

Orientation

3DFeat-Net: Keypoint Descriptor

- Orientation to rotate clusters into a canonical orientation.
- Concatenate cluster feature vector with individual point features to incorporate context.

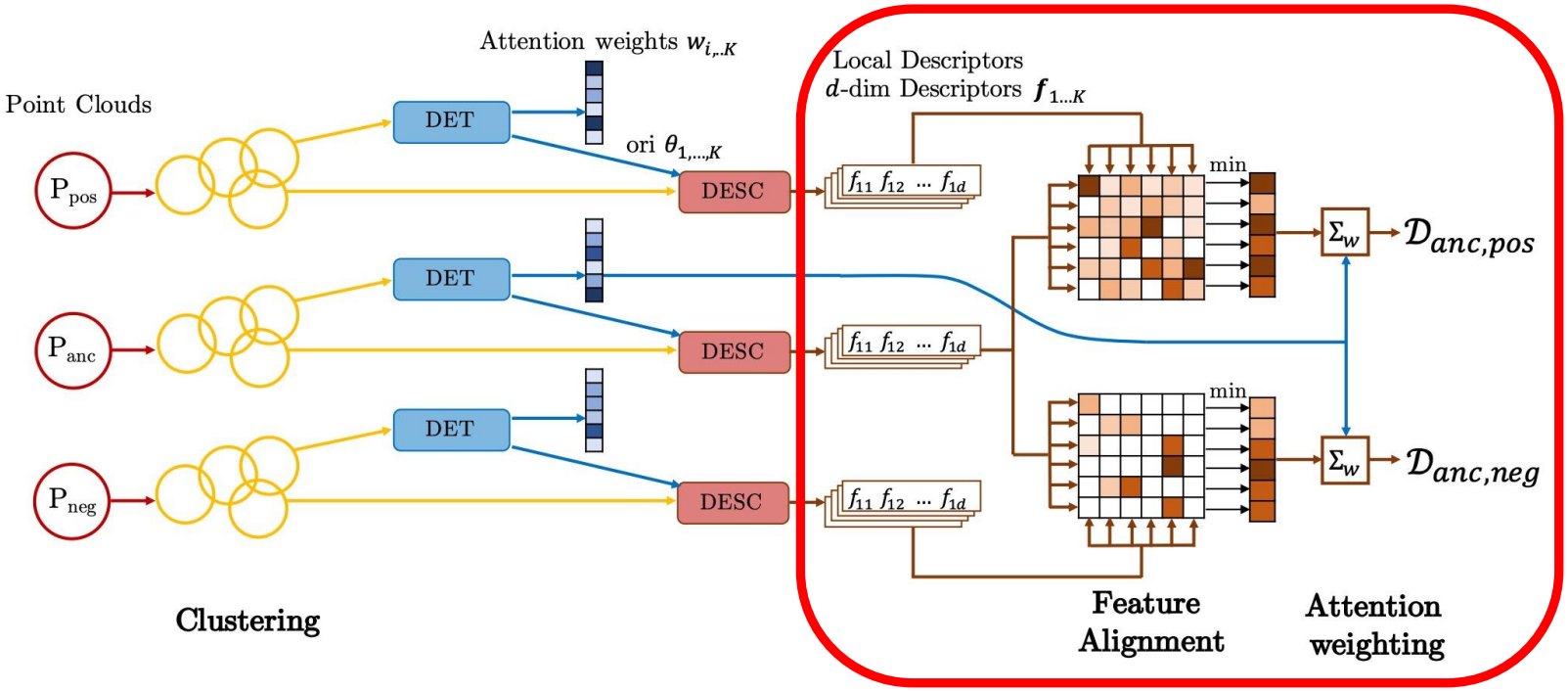




3DFeat-Net: Alignment Triplet Loss

$$\mathcal{L}_{triplet} = [\mathcal{D}_{anc,pos} - \mathcal{D}_{anc,neg} + \gamma]_+,$$

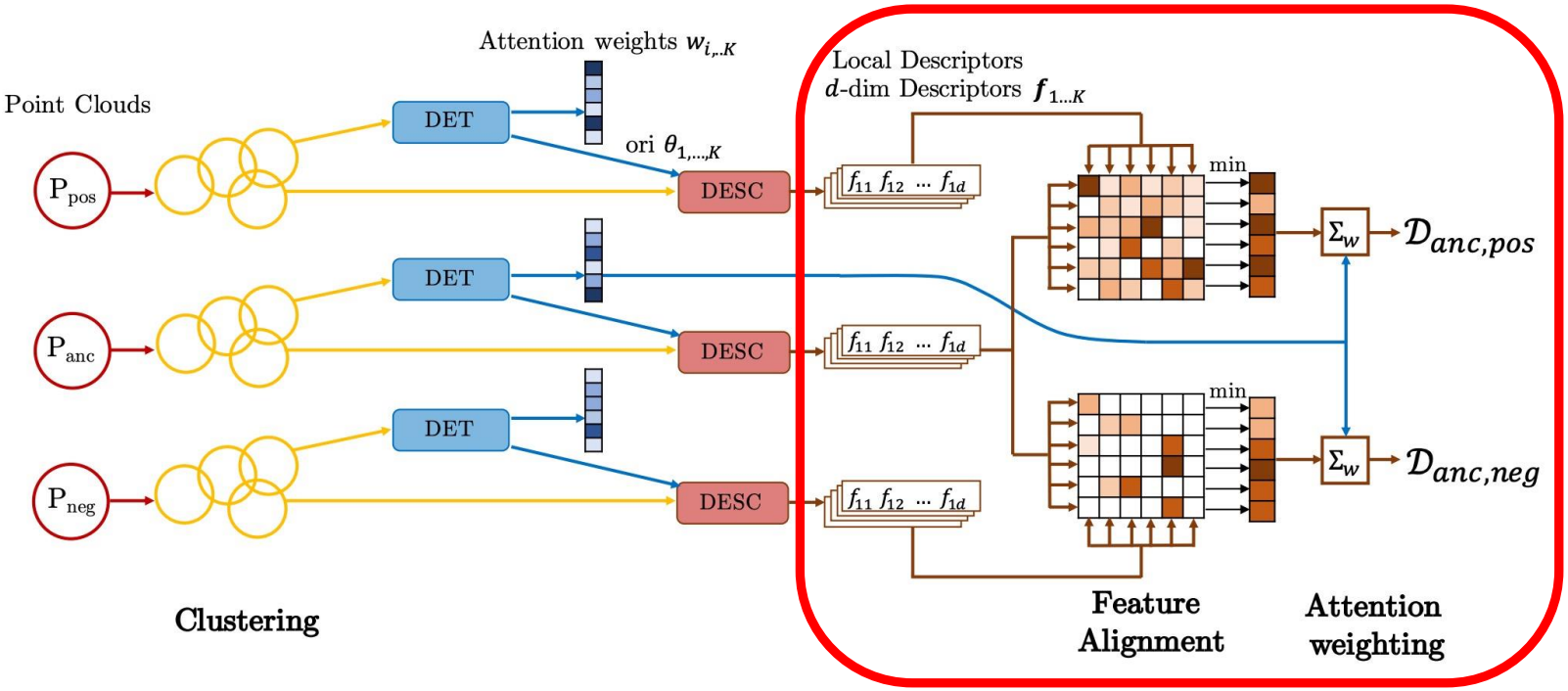
where $[z]_+ = max(z, 0)$ (hinge loss)



3DFeat-Net: Alignment Triplet Loss

- Align every descriptor from the first point cloud its closest descriptor in the second point cloud.

$$\mathcal{D}_{m,n} = \sum_{C_i \in \mathbf{C}^{(m)}} \left(w'_i \cdot \min_{C_j \in \mathbf{C}^{(n)}} \|f_i - f_j\|_2 \right), \quad w'_i = \frac{w_i}{\sum_{j \in P^{(m)}} w_j},$$



3DFeat-Net: Results

Table 1. Breakdown of Oxford RobotCar Data for Training and Testing

	# Traversals	# Point clouds	# Matched pairs
Train	35	21875	-
Test	5	828	-
Test (after registration)	5	794	3426

- We also test the network on the KITTI and ETH datasets.

3DFeat-Net: Results

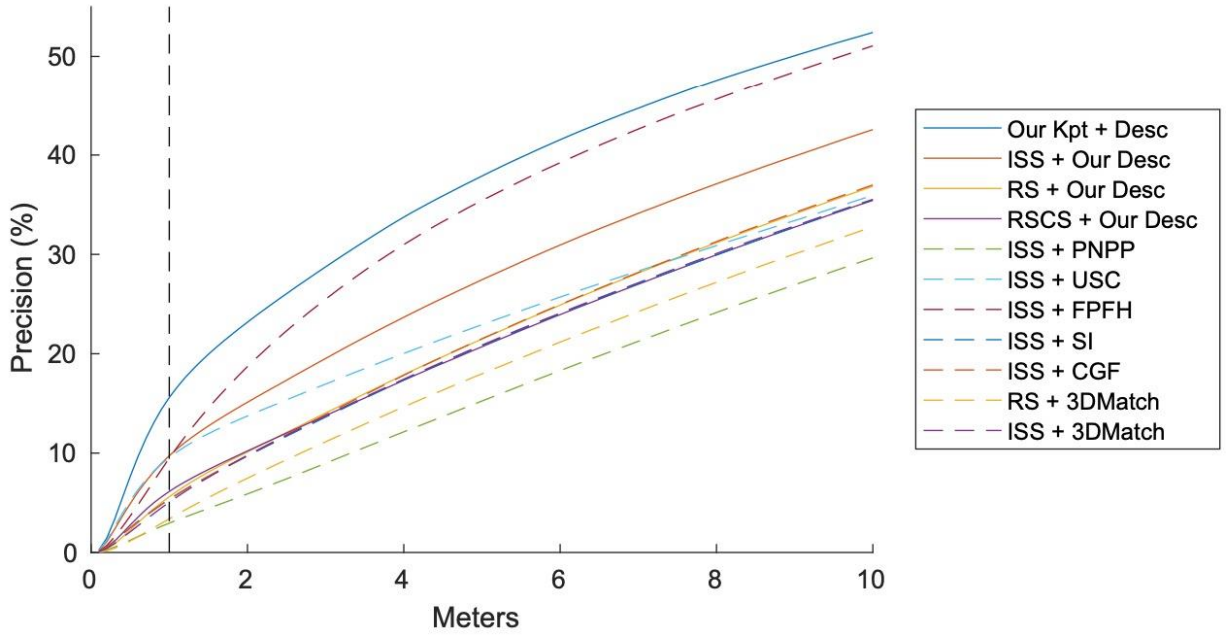


Fig. 4. Precision on the Oxford Robotcar dataset using different keypoint and descriptor combination. (Kpt = keypoints, Desc = descriptors)

3DFeat-Net: Results

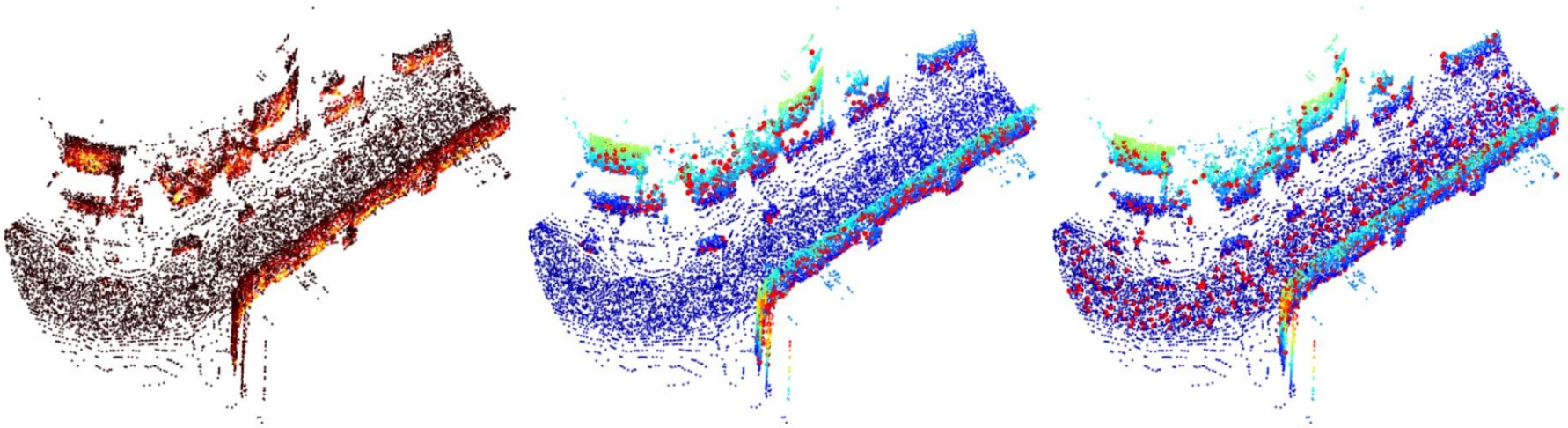


Fig. 5. Left: Attention using proposed method (brighter colors indicate higher attention). Middle: Our Keypoints (red dots). Right: ISS keypoints (red dots). Colors in middle and right images indicate different heights above ground.

3DFeat-Net: Results

Table 3. Performance on the Oxford RobotCar dataset.

Method	RTE (m)	RRE (°)	Success Rate	Avg # iter
ISS [35] + FPFH [25]	0.396 ± 0.290	1.60 ± 1.02	92.32%	7171
ISS [35] + SI [13]	0.415 ± 0.309	1.61 ± 1.12	87.45 %	9888
ISS [35] + USC [30]	0.324 ± 0.270	1.22 ± 0.95	94.02%	7084
ISS [35] + CGF [15]	0.431 ± 0.320	1.62 ± 1.10	87.36%	9628
RS + 3DMatch [34]	0.616 ± 0.407	2.02 ± 1.17	54.64%	9848
ISS [35] + 3DMatch [34]	0.494 ± 0.366	1.78 ± 1.21	69.06%	9131
ISS [35] + PN++ [24]	0.511 ± 0.391	1.88 ± 1.20	48.86%	9904
RS + Our Desc	0.435 ± 0.305	1.64 ± 1.04	90.28%	9941
RSCS [8] + Our Desc	0.386 ± 0.292	1.46 ± 1.01	92.64%	9913
ISS [35] + Our Desc	0.314 ± 0.262	1.08 ± 0.83	97.66%	7127
Our Kpt + Desc	$\mathbf{0.300 \pm 0.255}$	$\mathbf{1.07 \pm 0.85}$	**98.10%**	**2940**

3DFeat-Net: Results

Table 4. Performance on the KITTI odometry dataset.

Method	RTE (m)	RRE (°)	Success Rate	Avg # iter
ISS [35] + FPFH [25]	0.325 ± 0.270	1.08 ± 0.82	58.95%	7462
ISS [35] + SI [13]	0.358 ± 0.304	1.17 ± 0.94	55.92%	9219
ISS [35] + USC [30]	0.262 ± 0.275	0.83 ± 0.75	78.24%	7873
ISS [35] + CGF [15]	$\mathbf{0.233 \pm 0.266}$	0.69 ± 0.60	87.81%	7442
RS + 3DMatch [34]	0.377 ± 0.298	1.21 ± 0.84	83.96%	8674
ISS [35] + 3DMatch [34]	0.283 ± 0.272	0.79 ± 0.65	89.12%	7292
Our Kpt + Desc	0.259 ± 0.262	$\mathbf{0.57 \pm 0.46}$	**95.97%**	**3798**

3DFeat-Net: Results

Table 5. Performance on the ETH dataset, the results of FPFH and LORAX are taken from [8]. The last column indicates the success rate over the entire dataset.

Method	RTE (m)	RRE (°)	Success Rate	(All)
FPFH [25]	0.44 ± 0.2	12.2 ± 4.8	67%	-
LORAX [8]	0.42 ± 0.27	2.5 ± 1.2	83%	-
ISS [35] + SI [13]	0.176 ± 0.083	1.97 ± 0.74	**100%**	93.7%
ISS [35] + USC [30]	$\mathbf{0.130 \pm 0.056}$	1.52 ± 0.30	**100%**	**100%**
ISS [35] + CGF [15]	0.157 ± 0.066	$\mathbf{1.47 \pm 0.60}$	**100%**	92.1%
RS + 3DMatch [34]	0.292 ± 0.199	4.71 ± 3.16	91.7%	33.3%
ISS [35] + 3DMatch [34]	0.401 ± 0.222	5.32 ± 3.25	**100%**	33.3%
Our Kpt + Desc	0.156 ± 0.112	1.56 ± 0.66	**100%**	95.2%

3DFeat-Net: Results

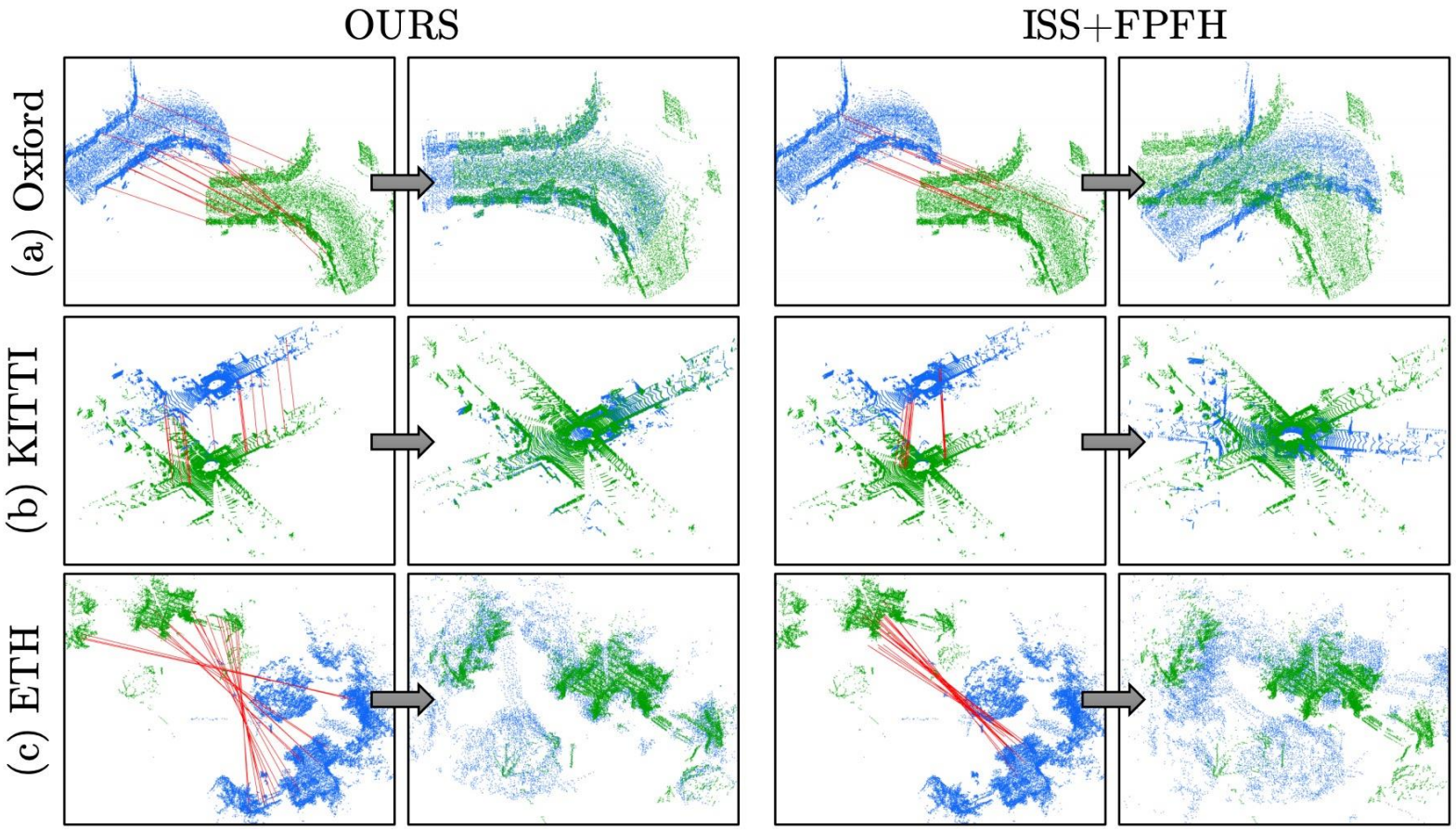
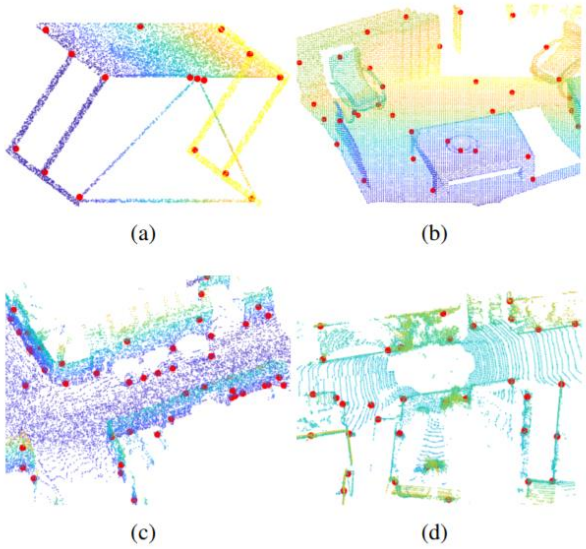


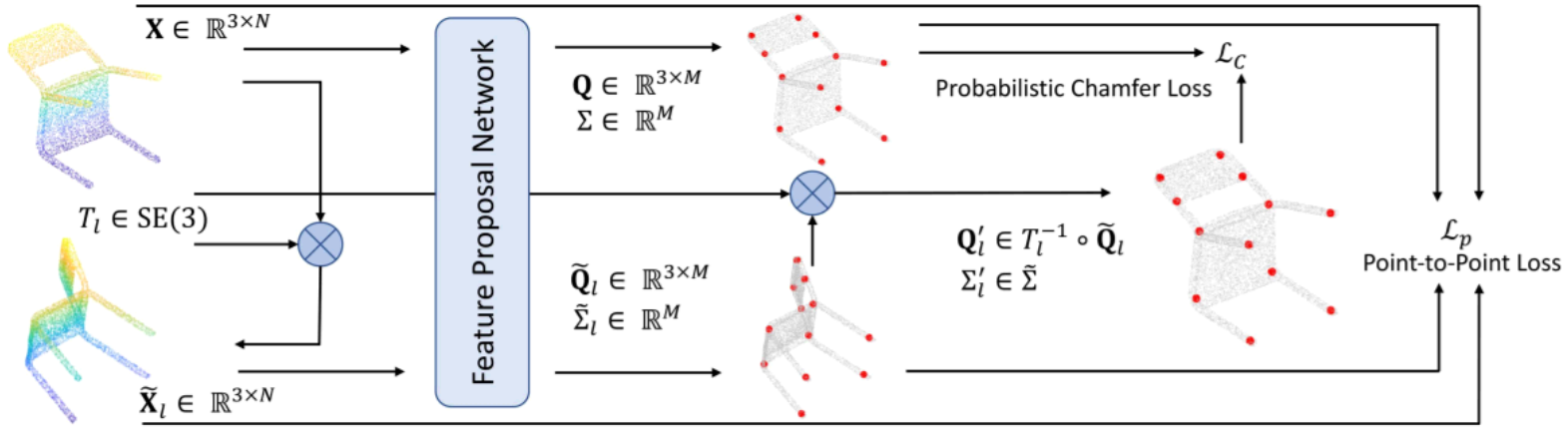
Fig. 6. Qualitative registration results, using our approach (left) and ISS + FPFH (right). We only show a random subset of matches retained after RANSAC, and exclude the ground in (c) for clarity. We also show the results using ISS + FPFH.

Unsupervised Learning: Keypoint Detection

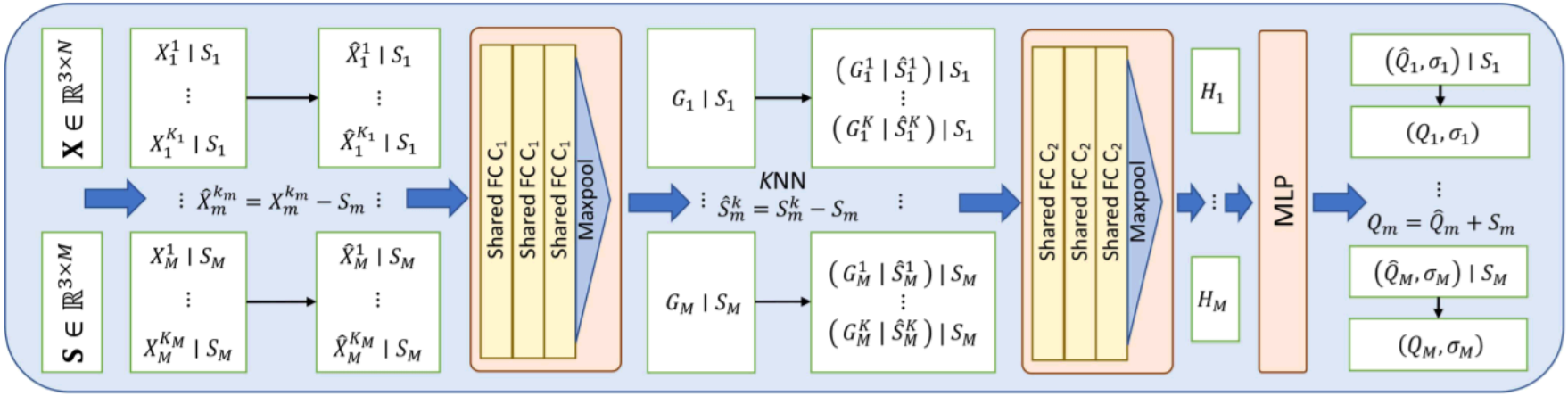
- Can we do better than weak supervision?
- **Goal:**
  Detect highly repeatable and accurately localized keypoints from 3D point clouds under arbitrary transformations **without** ground truth training data.



USIP: Unsupervised Stable Interest Point Detector



(a)



(b)

Jiaxin Li, Gim Hee Lee, **USIP: Unsupervised Stable Interest Point Detection from 3D Point Clouds,** ICCV 2019.

Loss Functions: Probabilistic Chamfer Loss

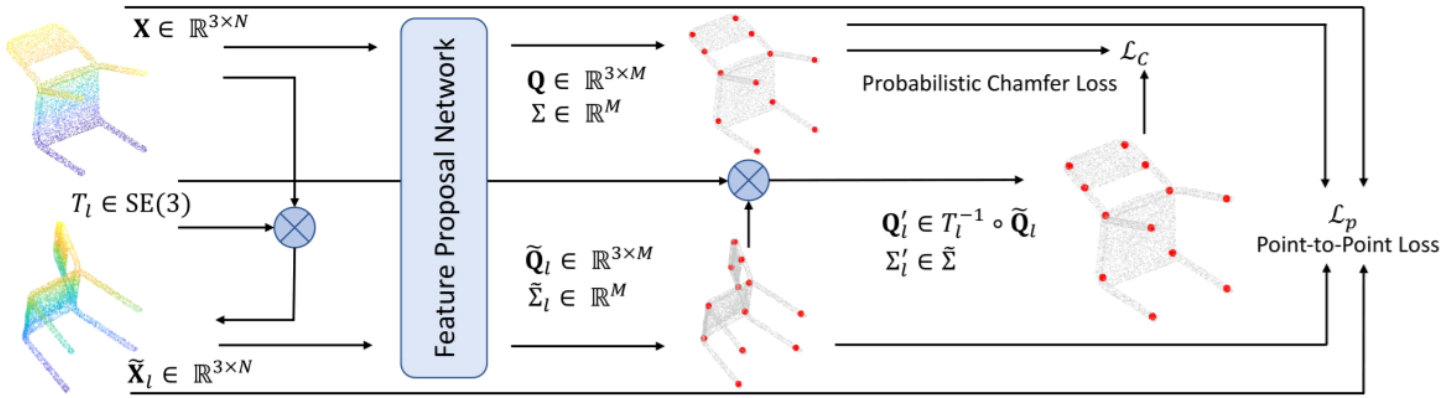
- A straightforward way to minimize the difference between $\mathbf{Q}$ and $\mathbf{Q}'$ is to use the chamfer loss:

$$\sum_{i=1}^{M} \min_{Q'_j \in \mathbf{Q}'} \|Q_i - Q'_j\|_2^2 + \sum_{j=1}^{M} \min_{Q_i \in \mathbf{Q}} \|Q_i - Q'_j\|_2^2,$$

- However, the M proposals are not equally salient.

- The receptive field of a point Q_i can be a featureless surface since the receptive field is limited to a small volume.

Loss Functions: Probabilistic Chamfer Loss



Probabilistic chamfer loss between $\boldsymbol{Q}'$ and $\boldsymbol{Q}$ is given by:

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

Exponential distribution between Q_i and Q'_j:

$$p(d_{ij} \mid \sigma_{ij}) = \frac{1}{\sigma_{ij}} \exp\left(-\frac{d_{ij}}{\sigma_{ij}} \right), \quad \text{where}$$

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

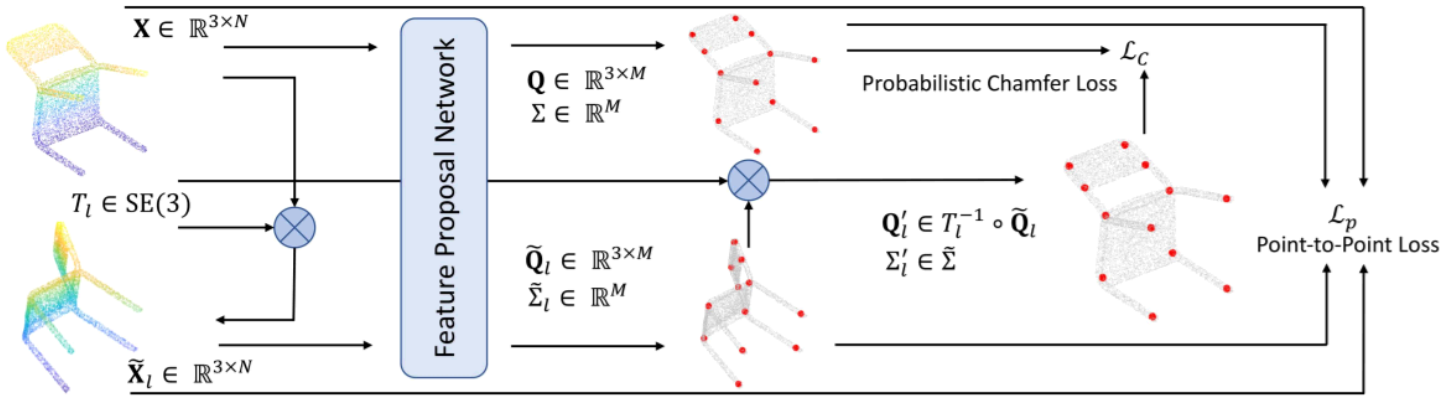
Joint distribution between Q_i and Q'_j:

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

Joint distribution between Q'_j and Q_i:

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

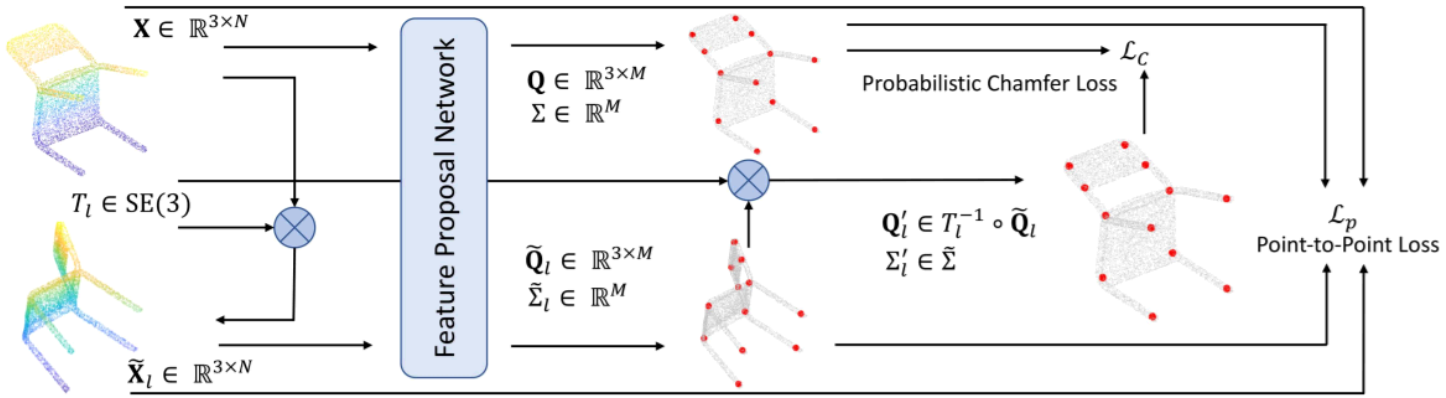
Loss Functions: Point-to-Point Loss



- To avoid quantization error, we design FPN such that it is not necessary that the proposal keypoints Q to be any points in X.
- However, this can cause the FPN to give erroneous proposal keypoints Q that are far away from the point cloud X.
- **Point-to-Point Loss**: Penalizes $Q_i \in Q$ for being too far from X

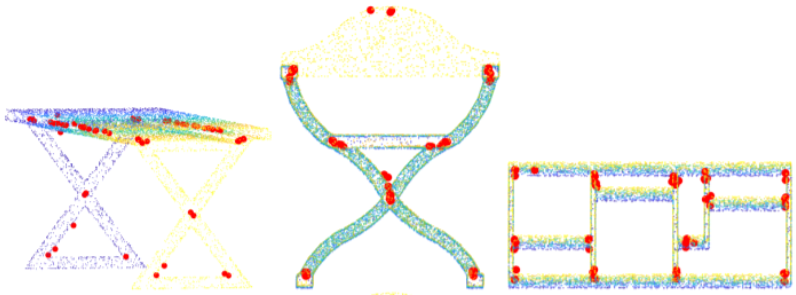
$$\mathcal{L}_{\text{point}} = \sum_{i=1}^{M} \min_{X_j \in \mathbf{X}} \|Q_i - X_j\|_2^2 + \sum_{i=1}^{M} \min_{\tilde{X}_j \in \tilde{\mathbf{X}}} \|\tilde{Q}_i - \tilde{X}_j\|_2^2$$

Loss Functions: Point-to-Point Loss

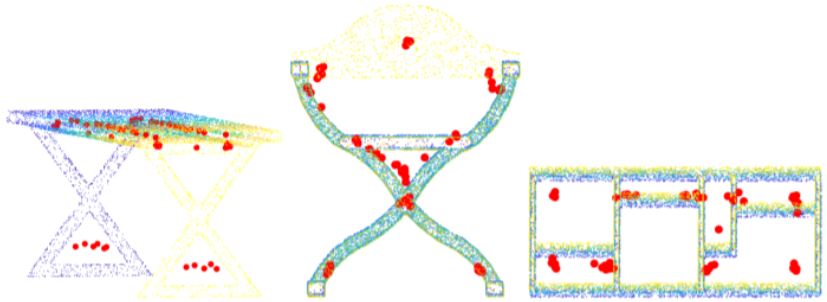


$$\mathcal{L}_{\text{point}} = \sum_{i=1}^{M} \min_{X_j \in \mathbf{X}} \|Q_i - X_j\|_2^2 + \sum_{i=1}^{M} \min_{\tilde{X}_j \in \tilde{\mathbf{X}}} \|\tilde{Q}_i - \tilde{X}_j\|_2^2$$

With point-to-point loss:



Without point-to-point loss:



Experimental Results

	KITTI	Oxford	Redwood	ModelNet40
Type	Velodyne lidar	SICK lidar	RGB-D	CAD Model
Scale	200m	60m	10m	2
# point	16,384	16,384	10,240	5,000
ϵ in Eq. 14	0.5m	0.5m	0.1m	0.03
Rotation	2D	2D	3D	3D
Noise	Sensor	Sensor	Gaussian	Gaussian
Occlusion	Yes	Yes	Yes	No
Density Variation	Yes	No	No	No
Missing Parts	Yes	Yes	Yes	No

Table 1. Datasets used in evaluating keypoint repeatability.

Two evaluation criteria:

1. **Repeatibility**: ability of a detector to detect keypoints in the same locations under various disturbances such as view-point variations, noise, missing parts, etc.
2. **Distinctiveness:** is a measure of the performance of keypoint detectors and descriptors for finding correspondences in point cloud registration.

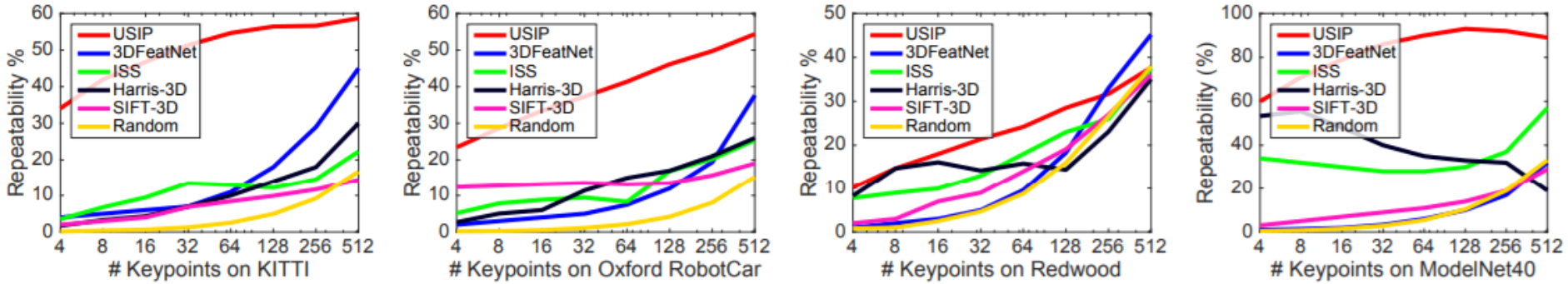


Figure 4. Relative repeatability when different number of keypoints are detected. Left to right: KITTI, Oxford, Redwood, ModelNet40.

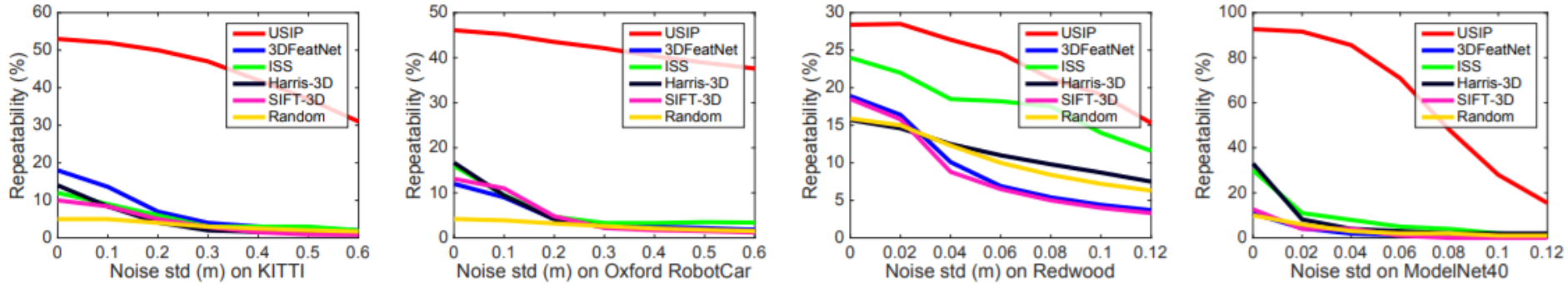


Figure 5. Relative repeatability when Gaussian noise $\mathcal{N}(0, \sigma_{noise})$ is added to the input point clouds. Keypoint number is fixed to 128.

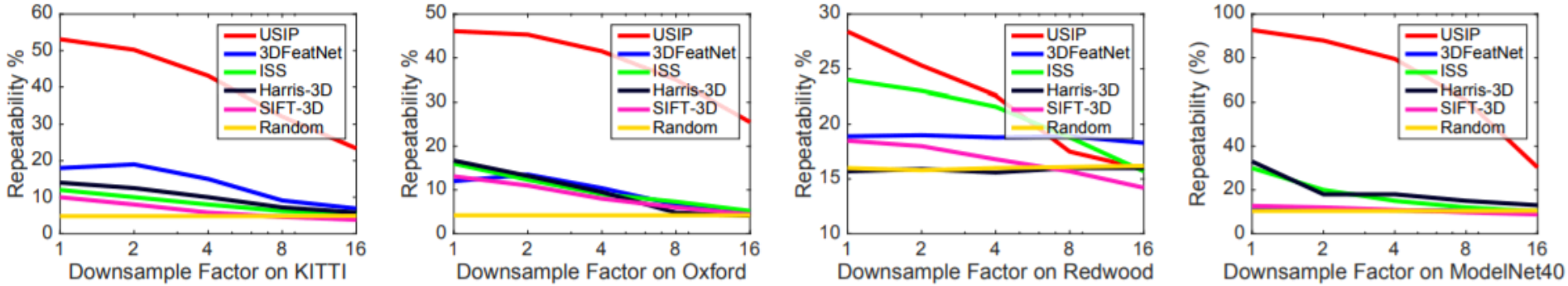


Figure 6. Relative repeatability when the input point cloud is randomly downsampled by some factors. Keypoint number is fixed to 128.

	Registration Failure Rate (%)				Inlier Ratio (%)			
	Our Desc.	3DFeatNet[36]	FPFH[27]	SHOT[31]	Our Desc.	3DFeatNet	FPFH	SHOT
Random	18.83	42.14	49.95	68.39	7.47	4.48	5.45	4.46
SIFT-3D[29, 20]	15.44	42.63	79.72	84.49	7.36	5.47	4.24	4.11
ISS[29, 39]	5.97	25.96	37.09	69.83	8.52	4.71	4.44	3.45
Harris-3D[29, 12]	3.81	13.56	49.49	51.29	10.57	6.58	4.78	5.00
3DFeatNet[36]	2.61	2.26	12.15	11.76	15.66	10.76	9.55	8.46
USIP	**1.41**	**1.55**	**8.37**	**5.40**	**32.20**	**22.48**	**18.77**	**18.21**

Table 2. Point cloud registration results on KITTI. The number of keypoints is fixed to 256.

Detector	Descriptor	Fail(%)	Inlier(%)	RTE(m)	RRE (°)
3DFeat-Net	3DFeat-Net	0.57	12.9	0.26 ± 0.26	0.56 ± 0.46
USIP	Our Desc.	**0.24**	**28.0**	$\mathbf{0.21 \pm 0.24}$	$\mathbf{0.42 \pm 0.32}$

Table 3. Point cloud registration on KITTI from the optimal configurations of 3DFeat-Net and our USIP.

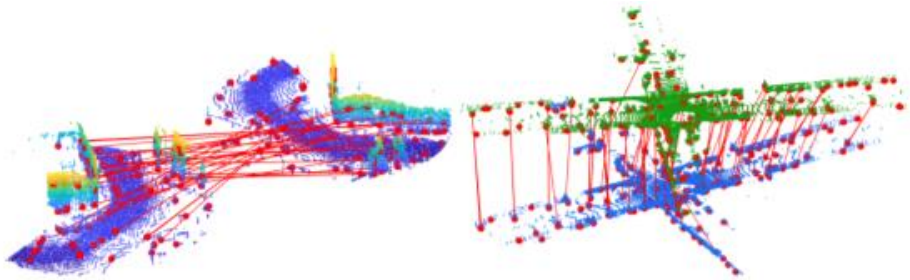
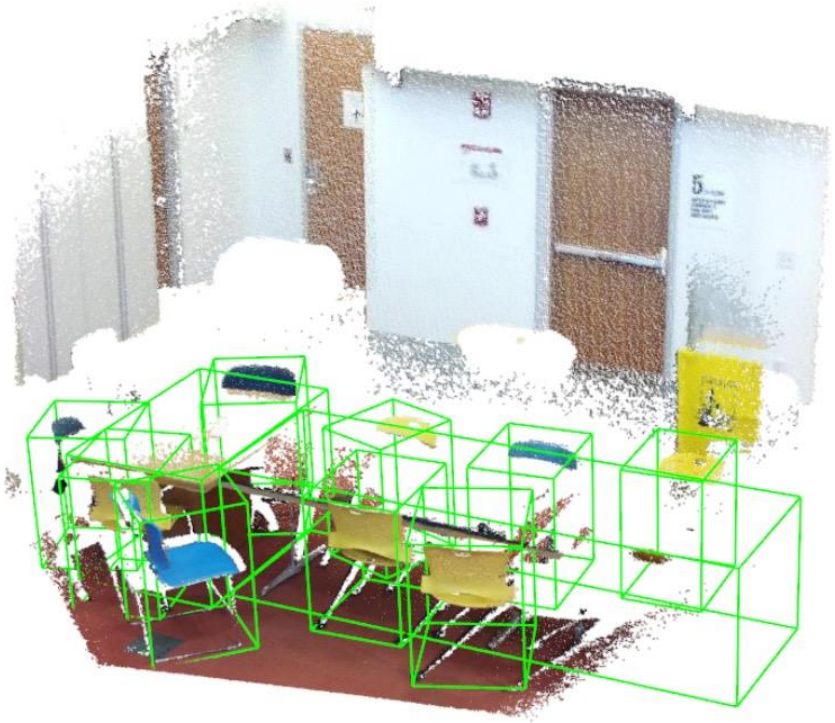


Figure 8. Keypoints and matches from our USIP detector and "Our Desc.". Best view with color and zoom-in.

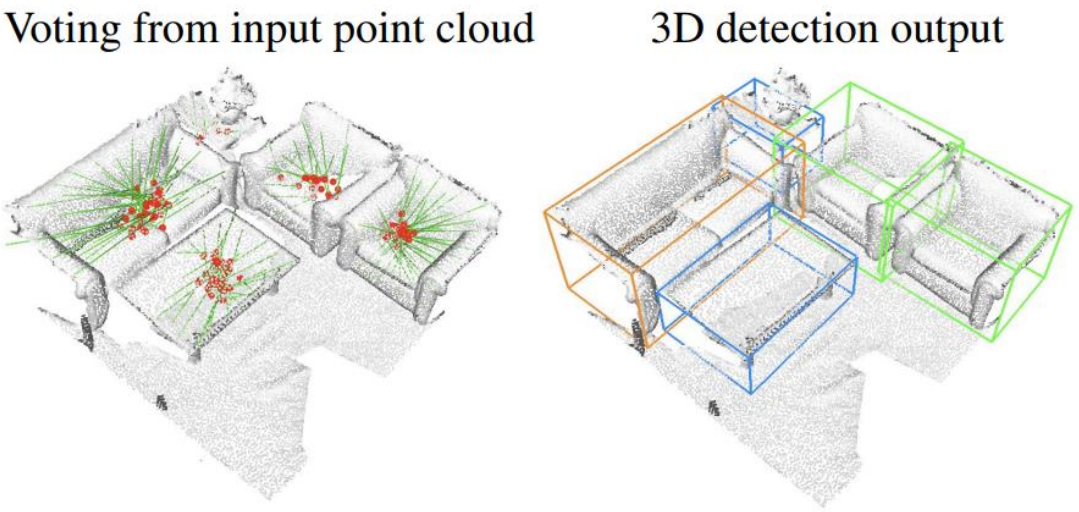
3D Object Detection

Given the input 3D point cloud, the goal is to find the tightest bounding box parameterized by $(x, y, z, h, w, l, \theta)$ around each object and its semantic class.



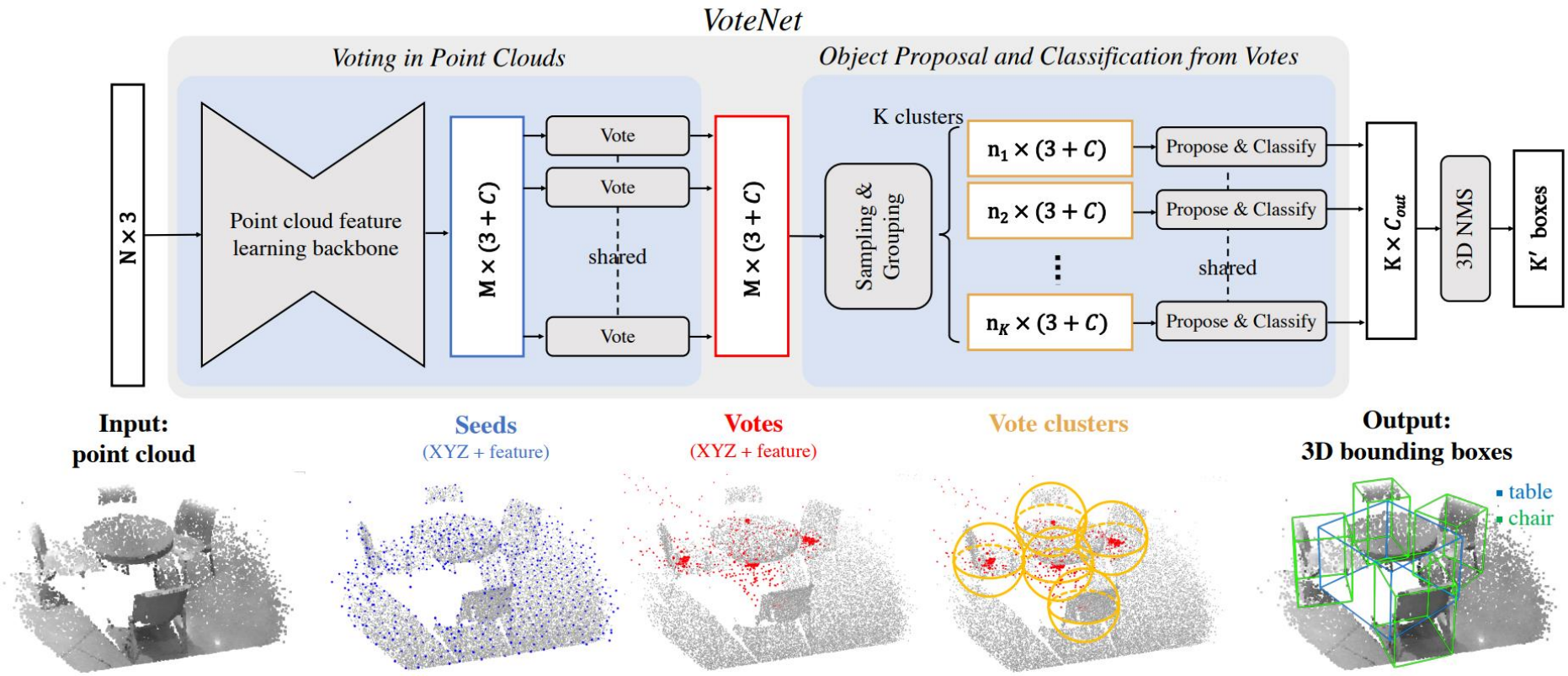
Deep Hough Voting for 3D Object Detection in Point Clouds

- Given a point cloud of a 3D scene, VoteNet votes to object centers, and then groups and aggregates the votes to predict 3D bounding boxes and semantic classes of objects.



Qi et al, Deep Hough Voting for 3D Object Detection in Point Clouds, ICCV 2019

Deep Hough Voting for 3D Object Detection in Point Clouds



Qi et al, Deep Hough Voting for 3D Object Detection in Point Clouds, ICCV 2019

Deep Hough Voting for 3D Object Detection in Point Clouds

Point cloud feature learning:

From an input point cloud of size $N \times 3$, use PointNet++ to generate M seed points of dimension $(3 + C)$. Each seed point generates one vote.

Deep Hough Voting for 3D Object Detection in Point Clouds

Hough voting with deep networks:

- Given a set of seed points $\{s_i\}_i^M = 1$ where $s_i = [x_i\ ;\ f_i]$ with $x_i \in \mathbb{R}^3$ and $f_i \in \mathbb{R}^C$, a shared voting module generates votes from each seed independently.

- A MLP takes f_i and outputs the Euclidean space offset $\Delta x_i \in \mathbb{R}^3$ and a feature offset $\Delta f_i \in \mathbb{R}^C$ such that the vote $v_i = [y_i;\ g_i]$, where $y_i = x_i + \Delta x_i$ and $g_i = f_i + \Delta f_i$.

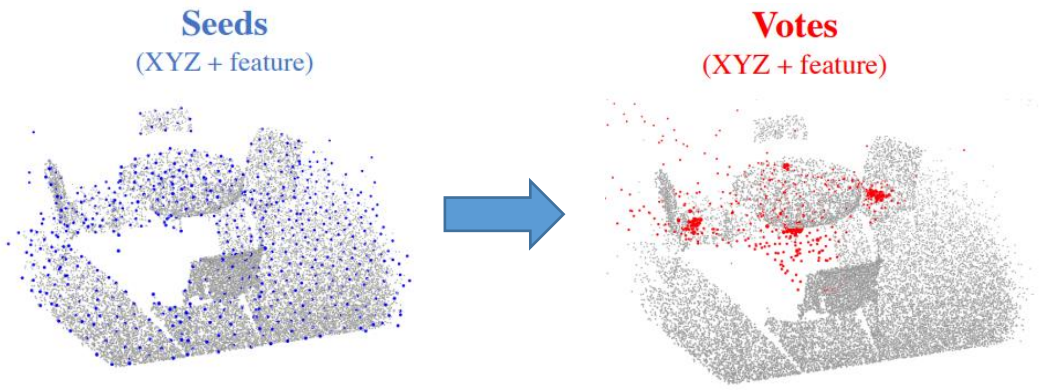
Deep Hough Voting for 3D Object Detection in Point Clouds

Hough voting with deep networks:

- The predicted 3D offset Δx_i is explicitly supervised by a regression loss:

$$L_{\text{vote-reg}} = \frac{1}{M_{\text{pos}}} \sum_i \|\Delta x_i - \Delta x_i^*\| \mathbb{1}[s_i \text{ on object}],$$

- $\mathbb{1}[s_i \text{ on object}]$ indicates whether a seed point s_i is on an object surface and M_{pos} is the count of total number of seeds on object surface.

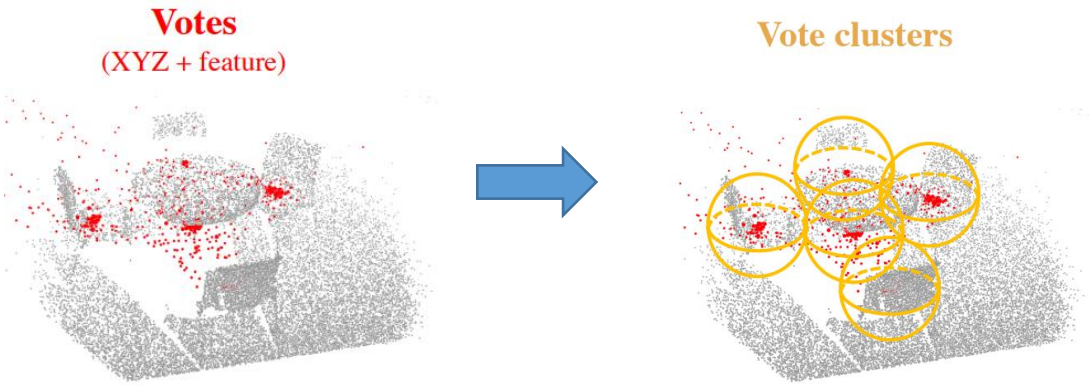


Deep Hough Voting for 3D Object Detection in Point Clouds

Vote clustering through sampling and grouping:

- From a set of votes $\{v_i = [y_i;\ g_i] \in \mathbb{R}^{3+C}\}_{i=1}^{M}$, we sample a subset of K votes using farthest point sampling based on $\{y_i\}$ in 3D Euclidean space, to get $\{v_{ik}\}$ with $k = 1, \dots, K$.

- Then, form K clusters by finding neighboring votes to each of the v_{ik}'s 3D location: $C_k = \left\{ v_i^{(k)} \mid \|v_i - v_{ik}\| \leq r \right\}$ for $k = 1, \dots, K$.



Deep Hough Voting for 3D Object Detection in Point Clouds

Proposal and classification from vote clusters:

- Given a vote cluster $C = \{w_i\}$ with $i = 1, \dots, n$ and its cluster center w_j, where $w_i = [z_i; h_i]$ with $z_i \in \mathbb{R}^3$ as the vote location and $h_i \in \mathbb{R}^C$ as the vote feature.
- To enable usage of local vote geometry, we transform vote locations to a local normalized coordinate system by $z_i' = (z_i - z_j)/r$.

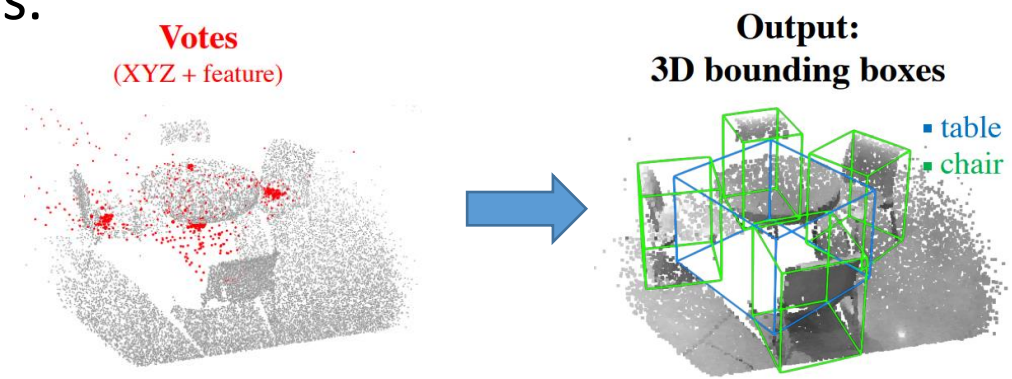
Deep Hough Voting for 3D Object Detection in Point Clouds

Proposal and classification from vote clusters:

- Then an object proposal for this cluster $p(C)$ is generated by passing the set input through a PointNet-like module:

$$p(\mathcal{C}) = \mathbf{MLP}_2 \left\{ \max_{i=1,\ldots,n} \{\mathbf{MLP}_1([z_i'; h_i])\} \right\}$$

- The proposal p is a multidimensional vector with an objectness score, bounding box parameters and semantic classification scores.



Results

3D object detection results on SUN RGB-D val set

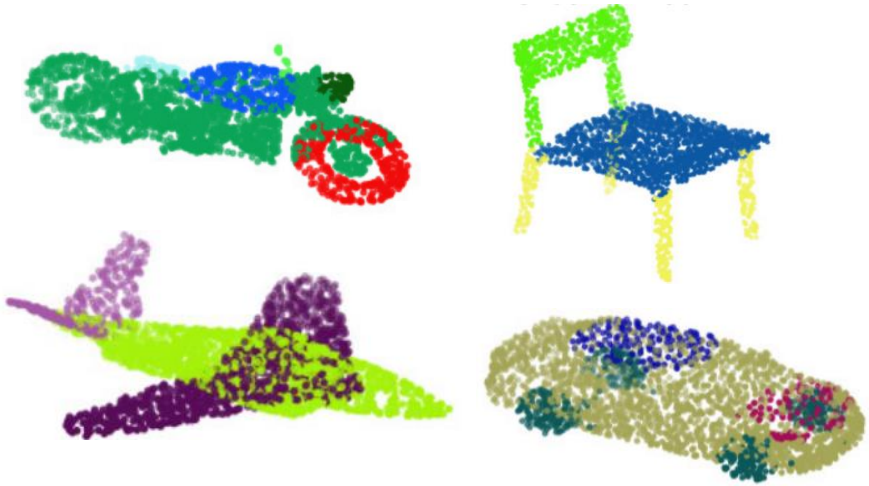
	Input	bathtub	bed	bookshelf	chair	desk	dresser	nightstand	sofa	table	toilet	mAP
DSS [42]	Geo + RGB	44.2	78.8	11.9	61.2	20.5	6.4	15.4	53.5	50.3	78.9	42.1
COG [38]	Geo + RGB	58.3	63.7	31.8	62.2	**45.2**	15.5	27.4	51.0	**51.3**	70.1	47.6
2D-driven [20]	Geo + RGB	43.5	64.5	31.4	48.3	27.9	25.9	41.9	50.4	37.0	80.4	45.1
F-PointNet [34]	Geo + RGB	43.3	81.1	**33.3**	64.2	24.7	**32.0**	58.1	61.1	51.1	**90.9**	54.0
VoteNet (ours)	**Geo only**	**74.4**	**83.0**	28.8	**75.3**	22.0	29.8	**62.2**	**64.0**	47.3	90.1	**57.7**

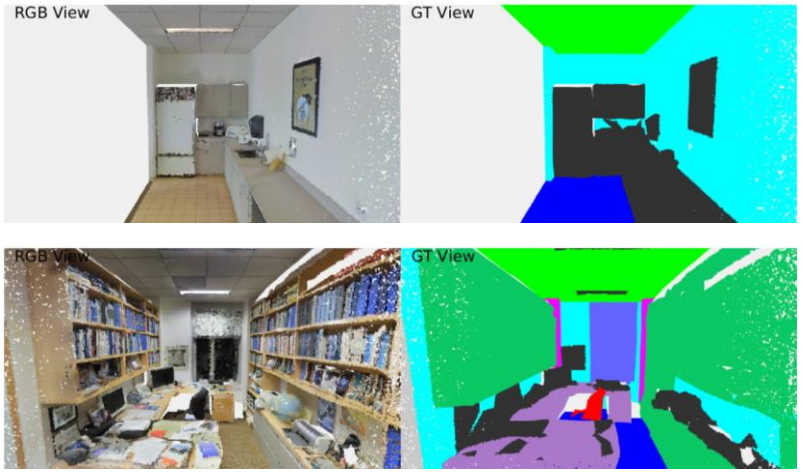
3D object detection results on ScanNetV2 val set

	Input	mAP@0.25	mAP@0.5
DSS [42, 12]	Geo + RGB	15.2	6.8
MRCNN 2D-3D [11, 12]	Geo + RGB	17.3	10.5
F-PointNet [34, 12]	Geo + RGB	19.8	10.8
GSPN [54]	Geo + RGB	30.6	17.7
3D-SIS [12]	Geo + 1 view	35.1	18.7
3D-SIS [12]	Geo + 3 views	36.6	19.0
3D-SIS [12]	Geo + 5 views	40.2	22.5
3D-SIS [12]	Geo only	25.4	14.6
VoteNet (ours)	Geo only	**58.6**	**33.5**

3D Semantic Segmentation

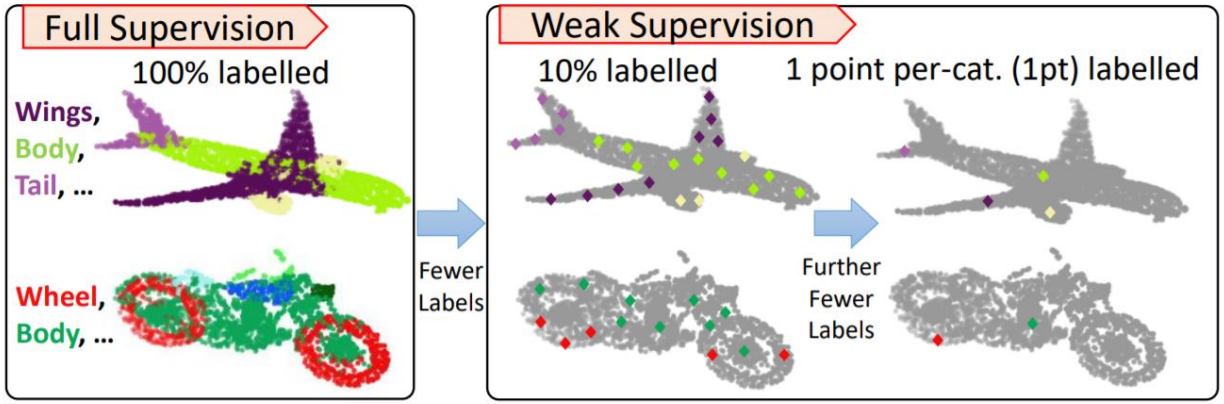
- **Problem:** 3D semantic segmentation tasks such as:
  i. 3D shape segmentation: Requires 1k to 10k points labels.
  ii. 3D indoor scene segmentation: The order of magnitude increases drastically to millions of points.



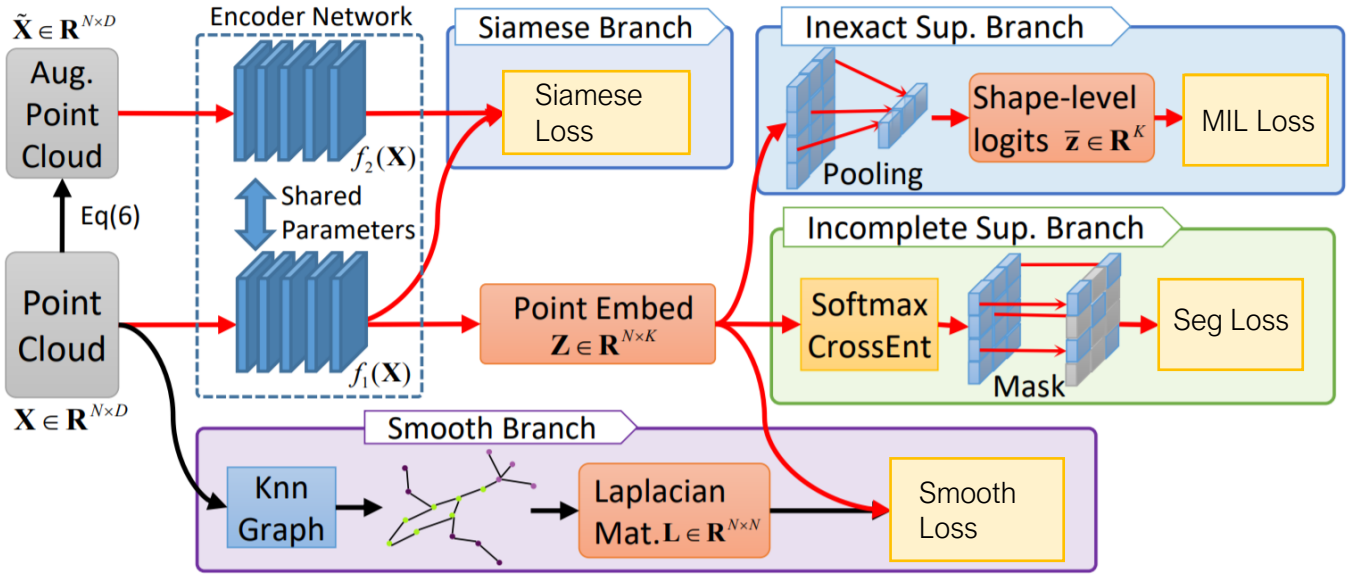


Weakly Supervised 3D Semantic Segmentation

Xun Xu, Gim Hee Lee, **"Weakly Supervised Semantic Point Cloud Segmentation: Towards 10× Fewer Labels",** CVPR 2020.

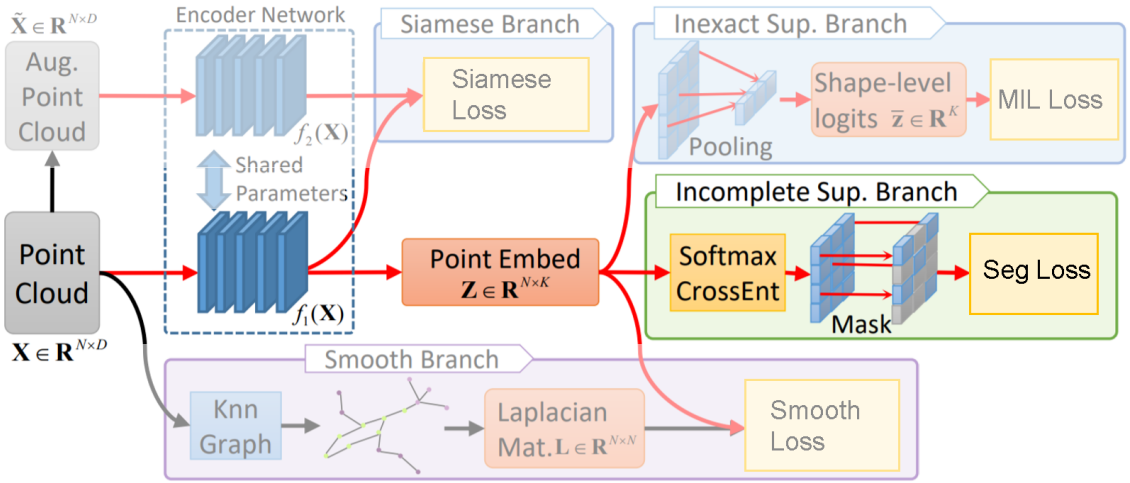


Weakly Supervised 3D Semantic Segmentation



Weakly Supervised 3D Semantic Segmentation

1. Incomplete Supervision Branch



Softmax cross-entropy loss on the labeled points:

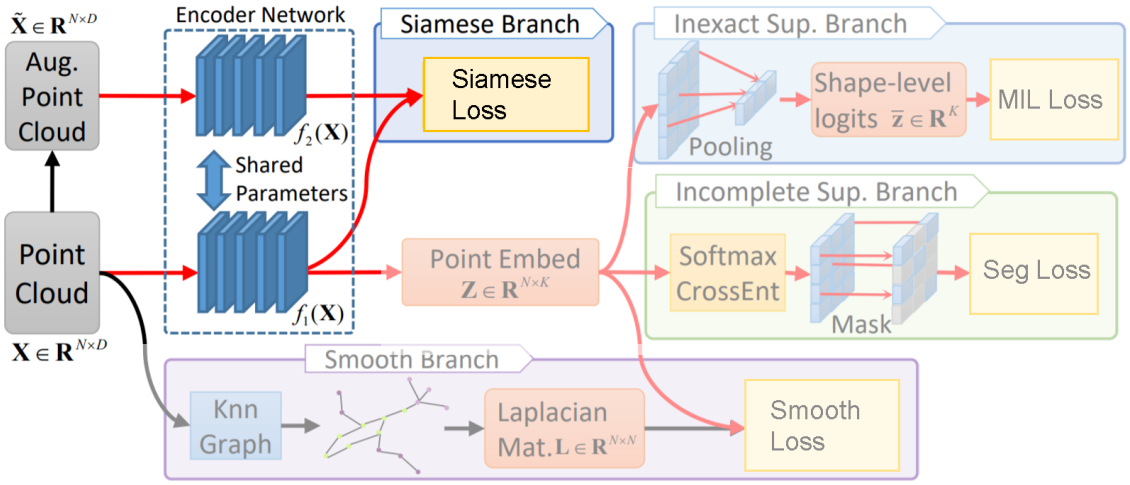
$$l_{seg} = -\frac{1}{C}\sum_b\sum_i m_{bi}\sum_k \hat{y}_{bik}\log\frac{\exp(z_{bik})}{\sum_k \exp(z_{bik})},$$

$\hat{\mathbf{Y}} \in \{0,1\}^{B\times N\times K}$: ground truth labels

$\mathbf{M} \in \{0,1\}^{B\times N}$: mask for valid labels

Weakly Supervised 3D Semantic Segmentation

2. Siamese Branch



L2 distance to measure divergence:

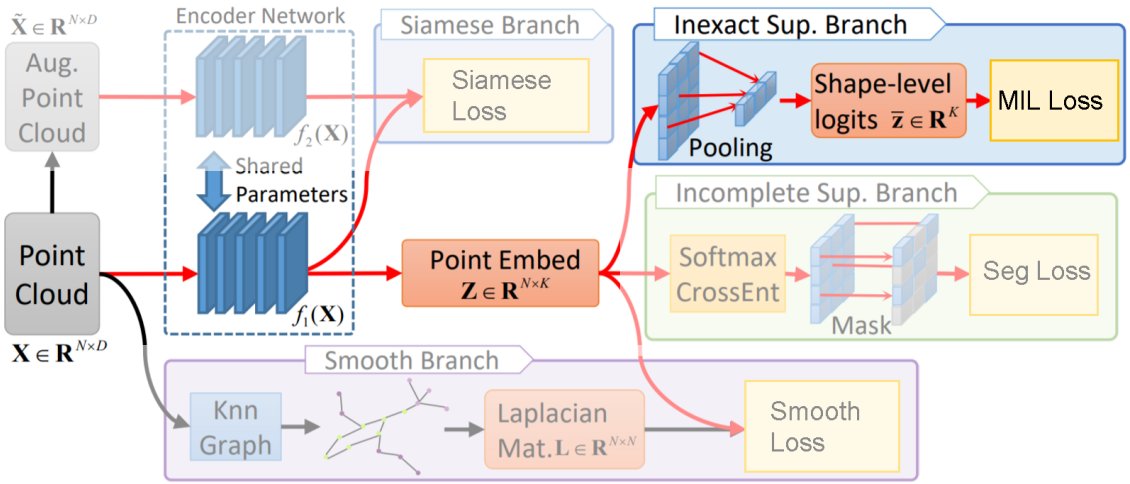
$$l_{sia} = \frac{1}{B \cdot N \cdot K} \sum_{b} ||g(f_1(\mathbf{X}_b)) - g(f_2(\tilde{\mathbf{X}}_b))||_F^2,$$

$\mathrm{X_b}$: input point cloud

$\tilde{\mathrm{X}}_b$: $\mathrm{X_b}$ mirrored along X/Y or XY in plane rotation

Weakly Supervised 3D Semantic Segmentation

3. Inexact Supervision Branch



Multi-instance labeling (MIL):

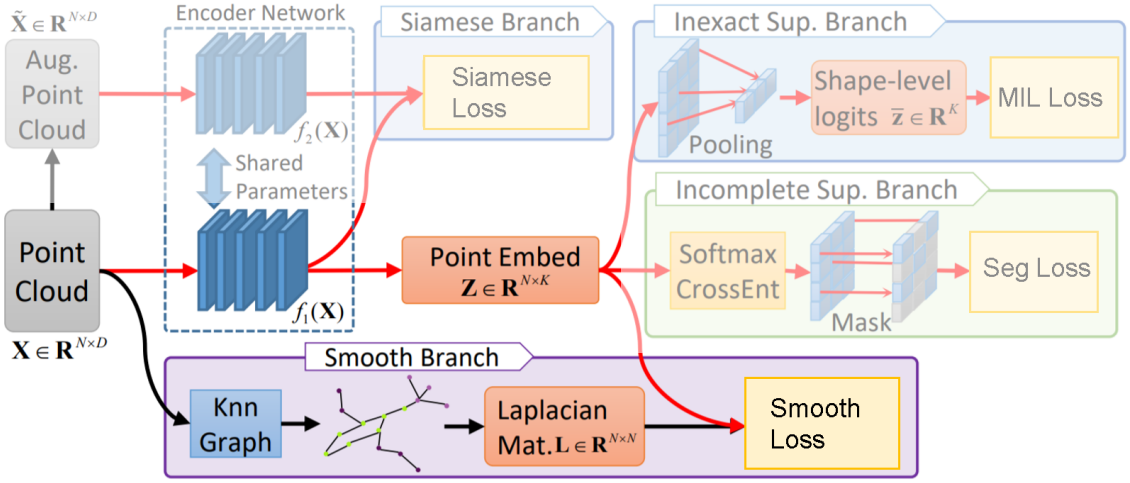
$$l_{mil} = -\frac{1}{B \cdot K} \sum_b \sum_k \bar{y}_{bk} \log \frac{1}{1+\exp(-\bar{z}_{bk})} + (1-\bar{y}_{bk})(\log(\frac{\exp(-\bar{z}_{bk})}{1+\exp(-\bar{z}_{bk})})).$$

$$\bar{\mathbf{y}}_b = \max_i \hat{\mathbf{y}}_{bi}$$

$$\bar{\mathbf{z}}_b = \max_i \mathbf{z}_{bi}.$$

Weakly Supervised 3D Semantic Segmentation

4. Smoothness Branch



Smoothness Loss (Spatial & Color Smoothness Constrain):

$$l_{smo} = \frac{1}{||\mathbf{W}||_0} \sum_i \sum_j w_{ij} ||f(\mathbf{x}_i) - f(\mathbf{x}_j)||_2^2 ,$$

$$w_{ij}^c = \begin{cases} \exp(-p_{ij}^c/\eta), & j \in NN_k(\mathbf{x}_i) \\ 0, & \text{otherwise} \end{cases} , \forall i, j \in \{1, \cdots N\}.$$

$$p_{ij}^c = ||\mathbf{x}_i^c - \mathbf{x}_j^c||_2, \ \forall i, j \in \{1, \cdots N\}$$

c: channel (xyz or rgb)

Experimental Results

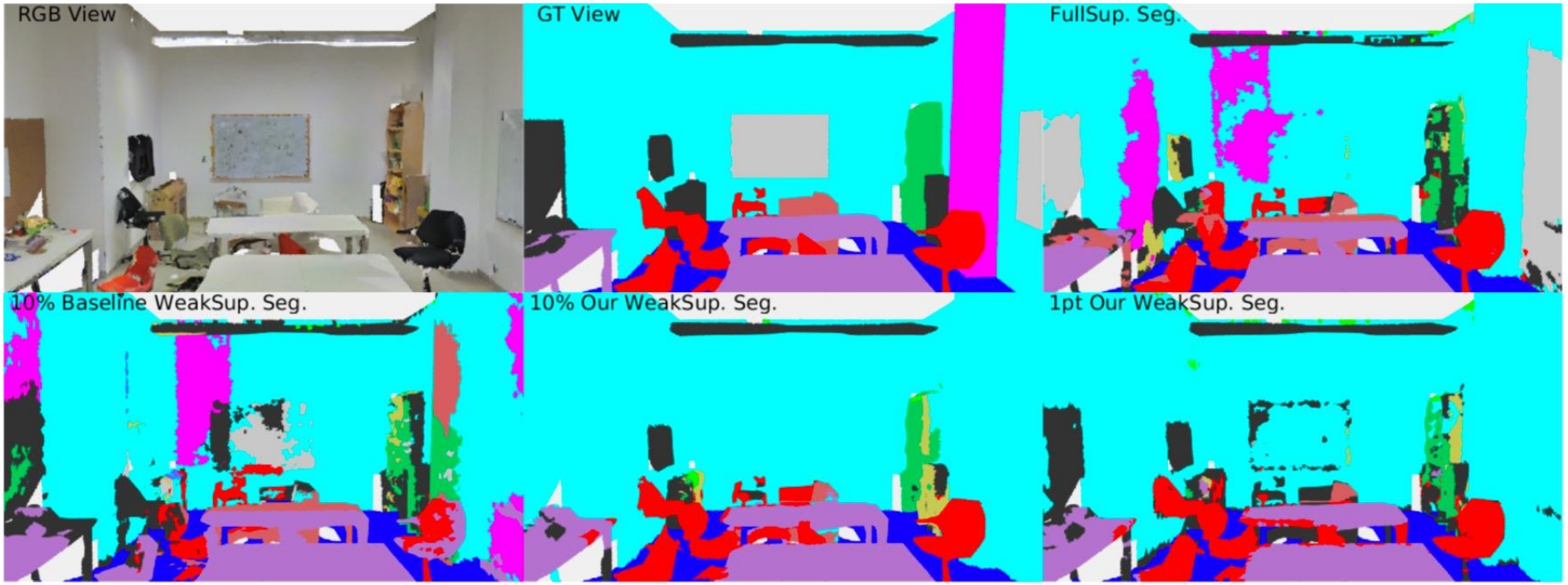
Table 5: Evaluation of alternative encoder network on ShapeNet dataset.

Encoder Net	Setting	ShapeNet																	
		CatAvg	SampAvg	Air.	Bag	Cap	Car	Chair	Ear.	Guitar	Knife	Lamp	Lap.	Motor.	Mug	Pistol	Rocket	Skate.	Table
PointNet++	FullSup	81.87	84.89	82.74	81.19	87.84	78.11	90.71	73.40	90.93	86.04	83.36	95.07	72.38	94.96	80.03	55.13	76.05	81.98
	1pt WeakSup	80.82	84.19	80.86	76.90	86.94	75.57	90.35	74.00	90.34	86.05	83.66	95.12	66.97	93.22	79.20	57.93	74.30	81.68
	10% WeakSup	81.27	84.70	82.36	76.55	86.82	77.48	90.52	73.01	91.16	85.35	83.07	95.34	69.97	94.88	80.04	57.03	74.59	82.14

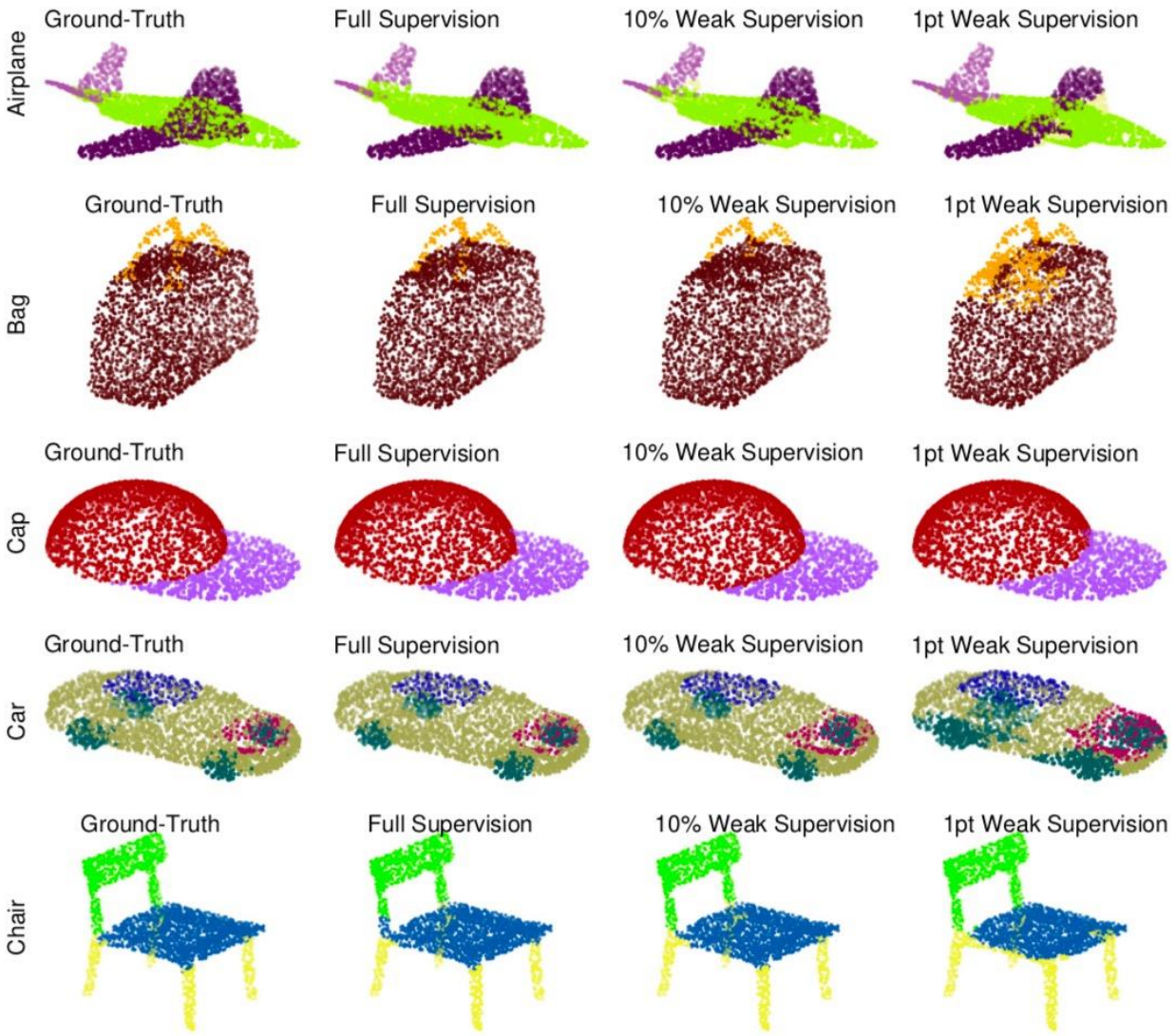
Table 6: Detailed results on PartNet dataset.

Setting		Model	CatAvg	Bag	Bed	Bott.	Bowl	Chair	Clock	Dish.	Disp.	Door	Ear.	Fauc.	Hat	Key	Knife	Lamp	Lap.	Micro.	Mug	Frid.	Scis.	Stora.	Table	Trash.	Vase
Ful.Sup.		PointNet	57.9	42.5	32.0	33.8	58.0	64.6	33.2	76.0	86.8	64.4	53.2	58.6	55.9	65.6	62.2	29.7	96.5	49.4	80.0	49.6	86.4	51.9	50.5	55.2	54.7
		PointNet++	65.5	59.7	51.8	**53.2**	**67.3**	68.0	**48.0**	**80.6**	89.7	59.3	**68.5**	**64.7**	62.4	62.2	64.9	**39.0**	**96.6**	55.7	**83.9**	51.8	87.4	58.0	**69.5**	64.3	64.4
		DGCNN	**65.6**	**53.3**	**58.6**	48.9	66.9	**69.1**	35.8	75.2	**91.2**	**68.5**	59.3	62.6	**63.7**	**69.5**	**71.8**	38.5	95.7	**57.6**	83.3	**53.7**	**89.7**	**62.6**	65.3	**67.8**	**66.8**
Weak Sup.	1pt	Baseline	50.2	24.4	30.1	20.5	38.0	65.9	35.3	64.9	**84.3**	**52.6**	36.7	47.1	47.9	52.2	55.2	34.1	92.4	49.3	59.5	49.6	80.1	44.6	49.8	40.4	49.5
		Ours	**54.6**	**28.4**	**30.8**	**26.0**	**54.3**	**66.4**	**37.7**	**66.3**	81.0	51.7	**44.4**	**51.2**	**55.2**	**56.2**	**63.1**	**37.6**	**93.5**	**49.7**	**73.5**	**50.6**	**83.6**	**46.8**	**61.1**	**44.1**	**56.8**
	10%	Baseline	63.2	54.4	56.8	44.1	**57.6**	67.2	41.3	**70.0**	**91.3**	**61.8**	**65.8**	57.2	**64.2**	64.2	66.7	**37.9**	94.9	49.1	80.2	**49.6**	84.1	59.3	69.7	66.7	63.0
		Ours	**64.5**	**47.3**	**55.5**	**64.7**	56.2	**69.1**	**44.3**	68.3	91.1	61.3	62.8	**65.2**	63.0	**64.6**	**67.9**	37.8	**95.5**	**50.1**	**82.7**	**49.6**	**85.8**	**59.5**	**71.2**	**67.7**	**65.9**

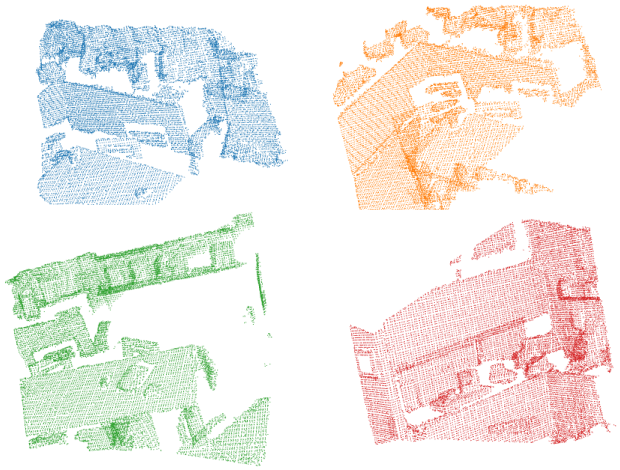
Results: S3DIS Scene Segmentation



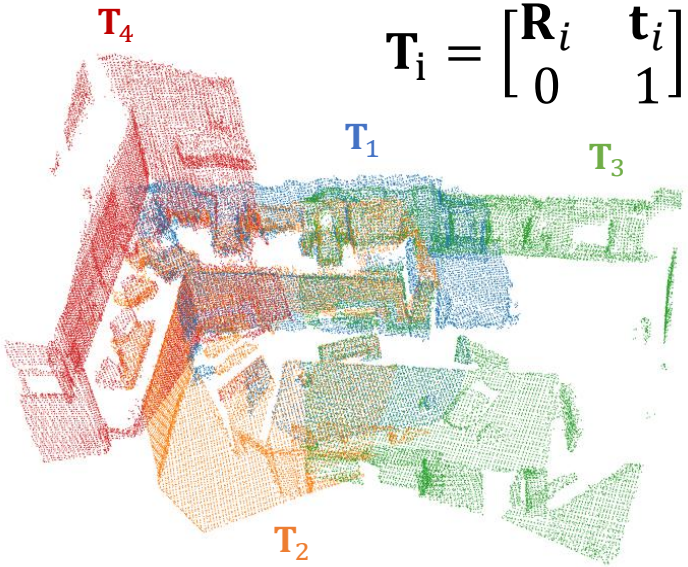
Results: ShapeNet part segmentation



Point Cloud Registration

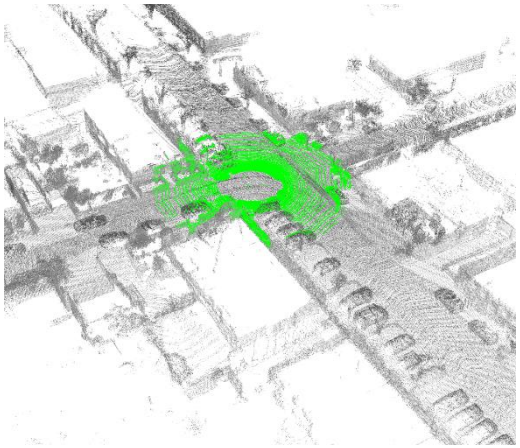


Registration

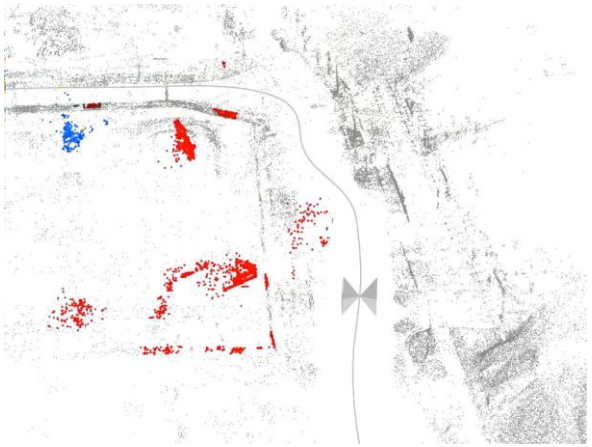


$$\mathbf{T}_i = \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ 0 & 1 \end{bmatrix}$$

Applications:



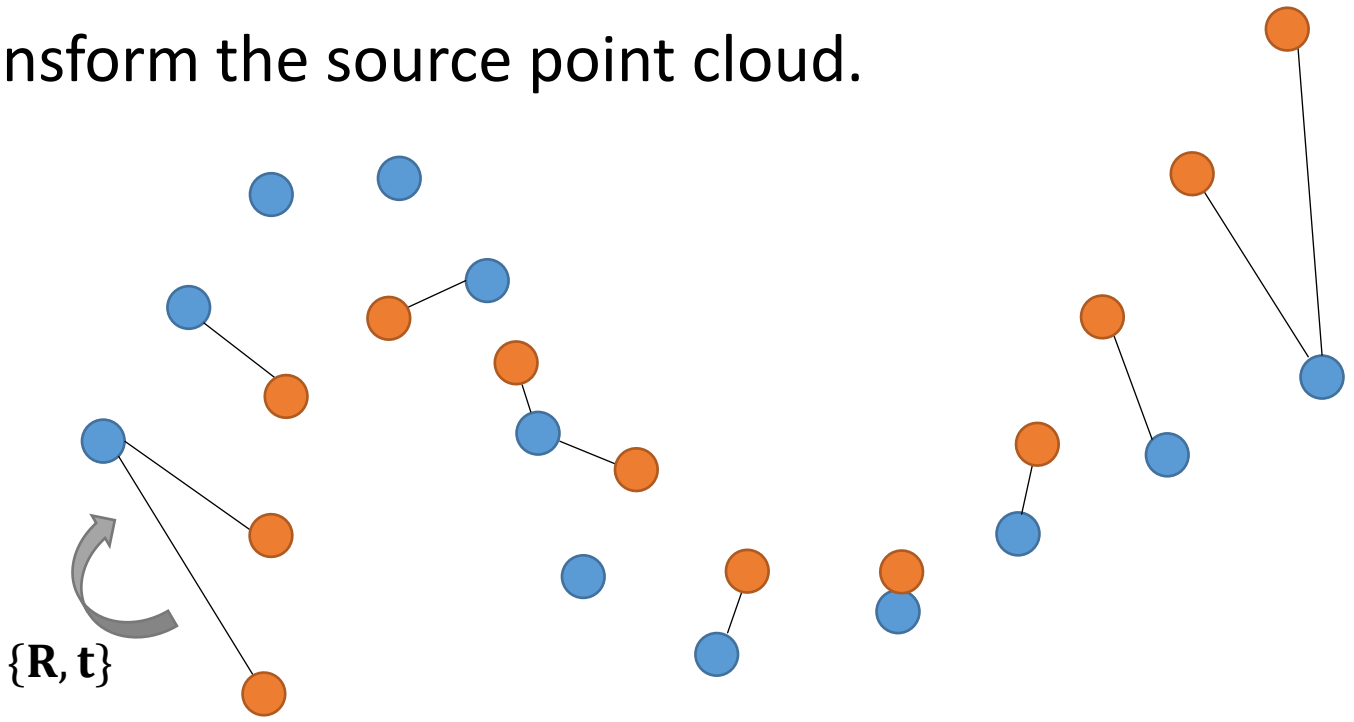
Localization of autonomous vehicles



Change Detection

Iterative Closest Point (ICP)

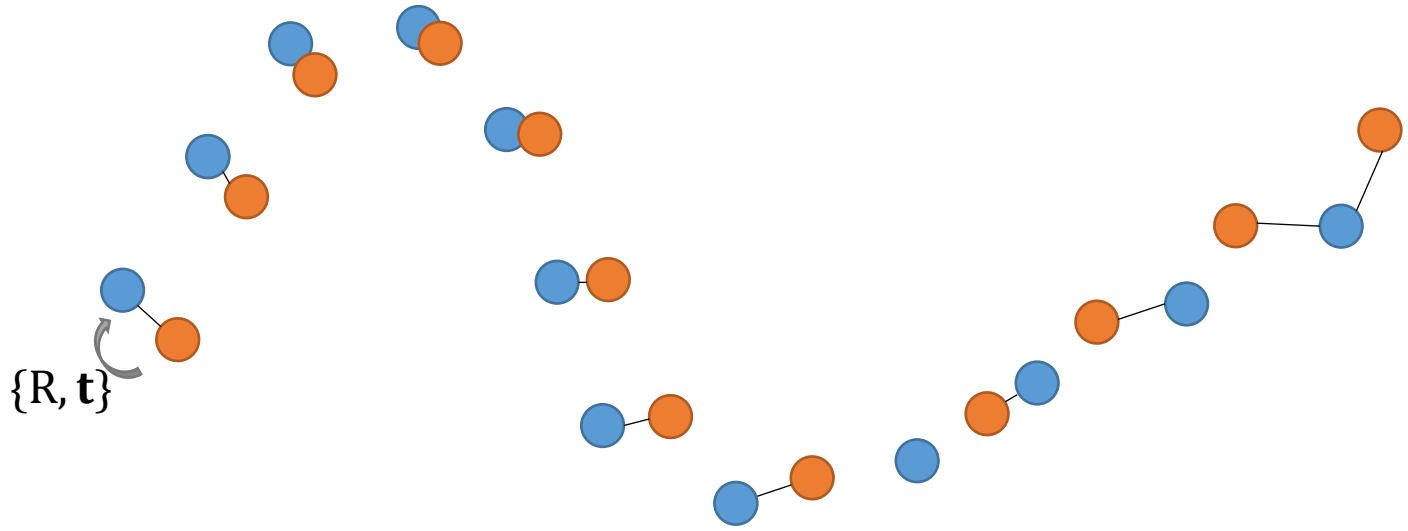
1. For each point in the source point cloud, match the closest point in the reference point cloud.
2. Estimate rigid transformation $\mathrm{R}, \mathbf{t}$ that minimizes some error metric.
3. Transform the source point cloud.



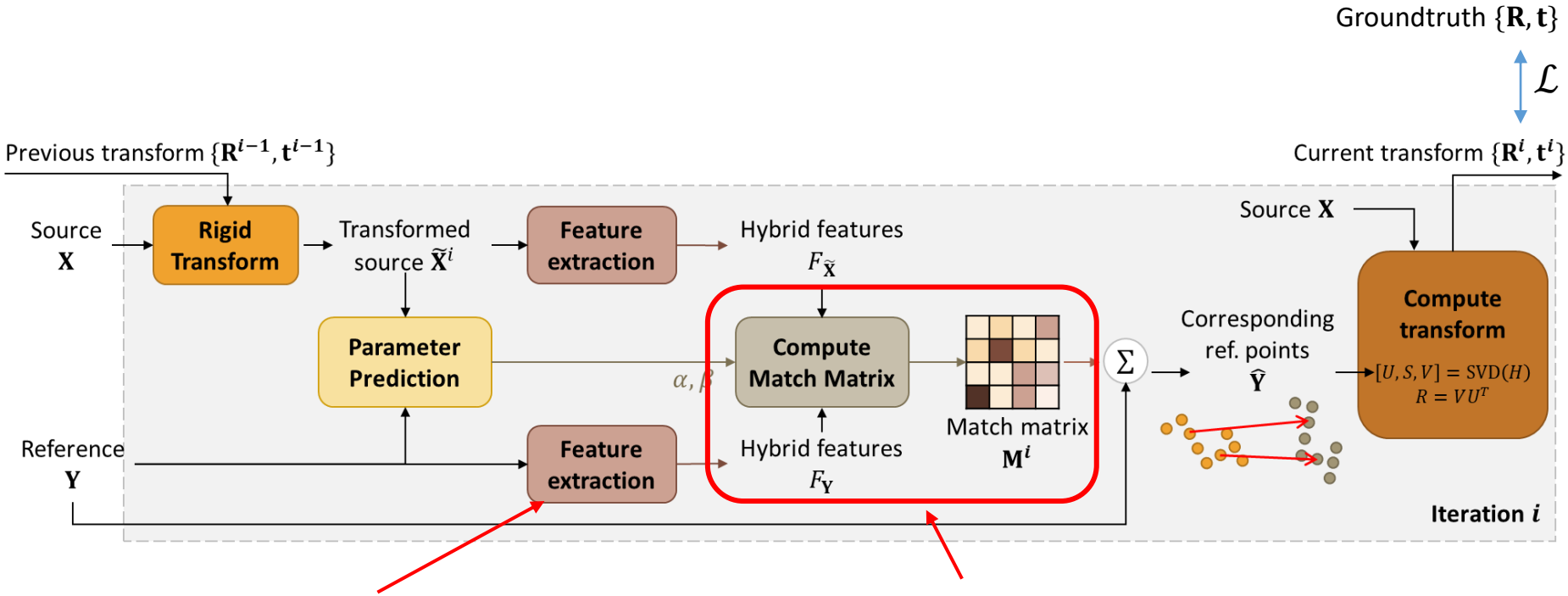
Iterative Closest Point (ICP)

4. Repeat steps 1-3 until converged

Main limitation: Susceptibility to local minima



RPM-Net: Robust Point Matching using Learned Correspondences

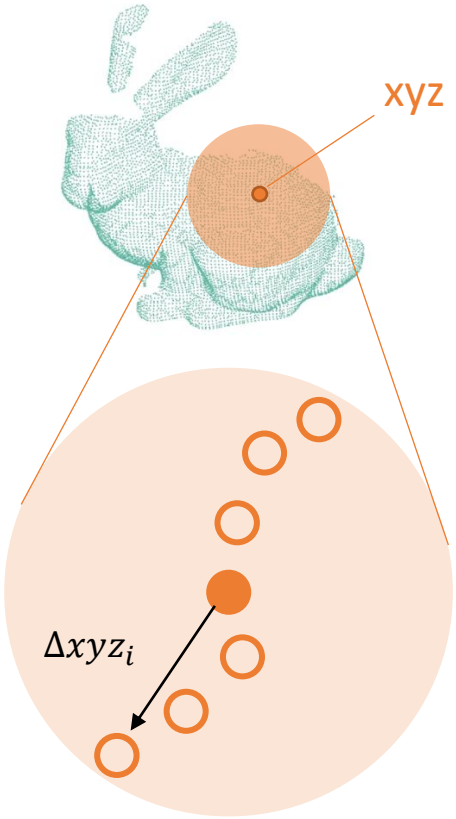


Hybrid (position + geometric) features

- Soft correspondences
- Sinkhorn normalization with slack to handle partial overlap
- Annealing

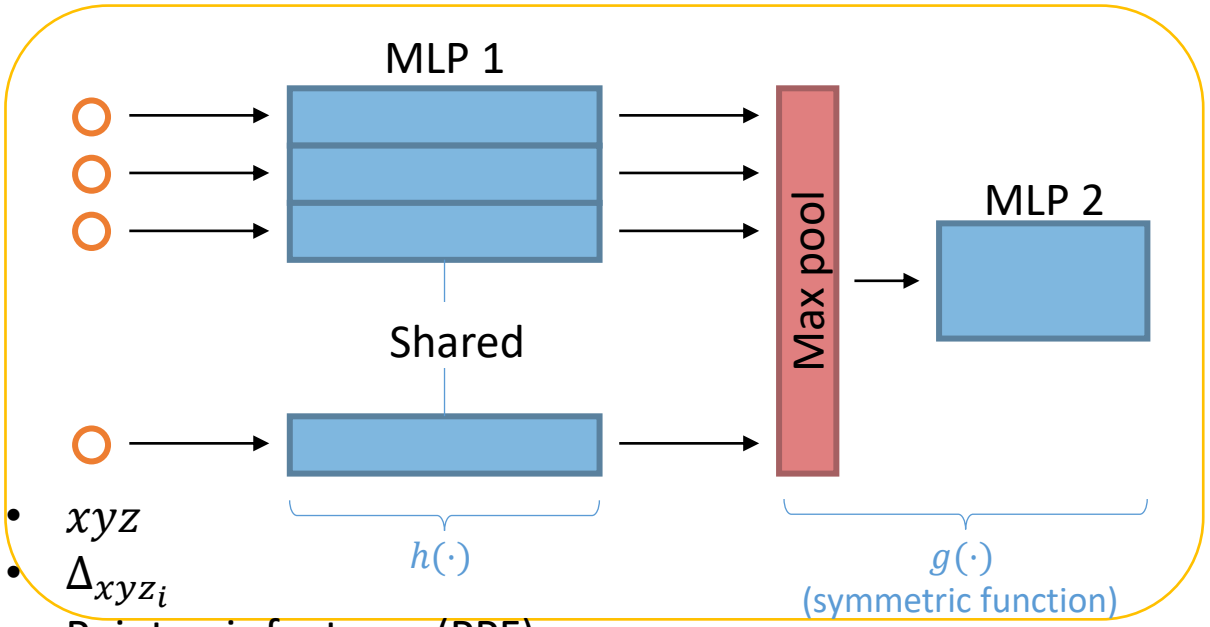
Zi Jian Yew and Gim Hee Lee, "RPM-Net: Robust Point Matching using Learned Features", CVPR2020.

RPM-Net: Feature Extraction



Feature $F \in \mathbb{R}^d$

PointNet



- xyz
- Δ_{xyz_i}
- Point pair features (PPF)

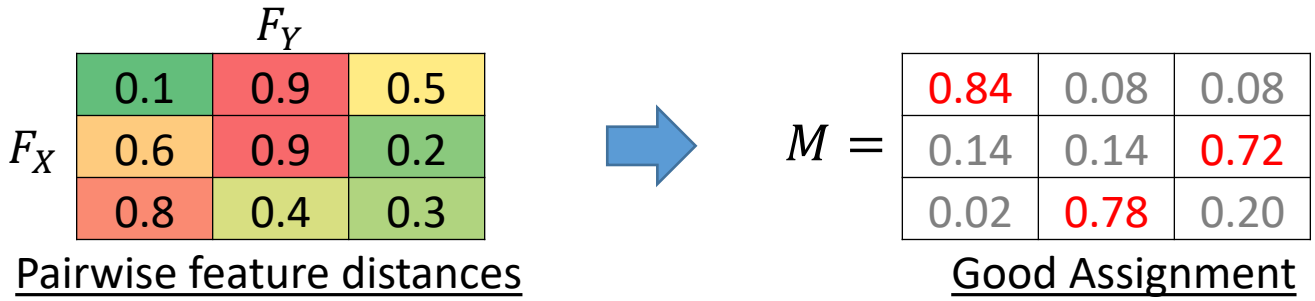
where $\mathrm{PPF}(\mathbf{x}_c) = \left(\angle(\mathbf{n}_c, \Delta\mathbf{x}_{c,i}), \angle(\mathbf{n}_i, \Delta\mathbf{x}_{c,i}), (\mathbf{n}_c, \mathbf{n}_i), \|\Delta\mathbf{x}_{c,i}\|\right)$

Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." CVPR2017

Deng, Haowen, Tolga Birdal, and Slobodan Ilic. "PPFNet: Global context aware local features for robust 3d point matching." CVPR2018.

RPM-Net: Computing the Match Matrix

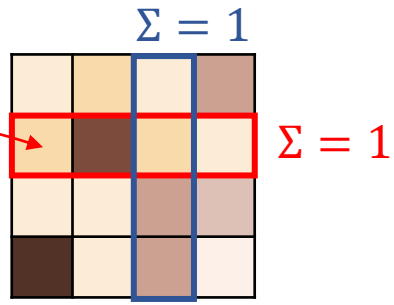
- Let us first assume that every point in X has a correspondence in Y.
- We want to find the match matrix $\mathbf{M} \in [0,1]^{J \times K}$, so that the correspondences result in a small cost.



RPM-Net: Computing the Match Matrix

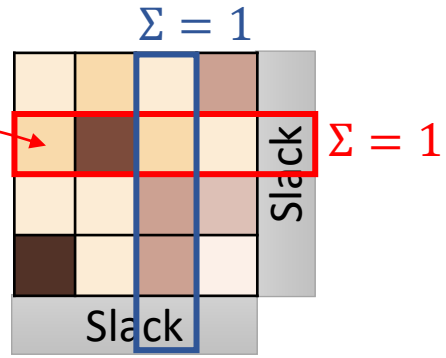
- Match matrix is estimated using Sinkhorn algorithm, which consists of repeated applications of alternating row and column normalizations.

$$m_{jk} \leftarrow e^{-\beta\left(\left\|F_X^i - F_Y^j\right\|^2\right)}$$



- When some points do not have correspondences:

$$m_{jk} \leftarrow e^{-\beta\left(\left\|F_X^i - F_Y^j\right\|^2 - \alpha\right)}$$

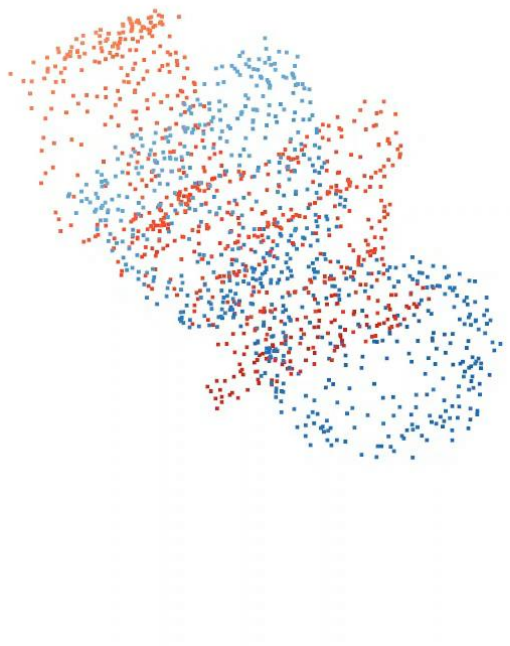


An additional slack row/column to the normalization, so that each row/column can sum to less than 1.

RPM-Net: Results

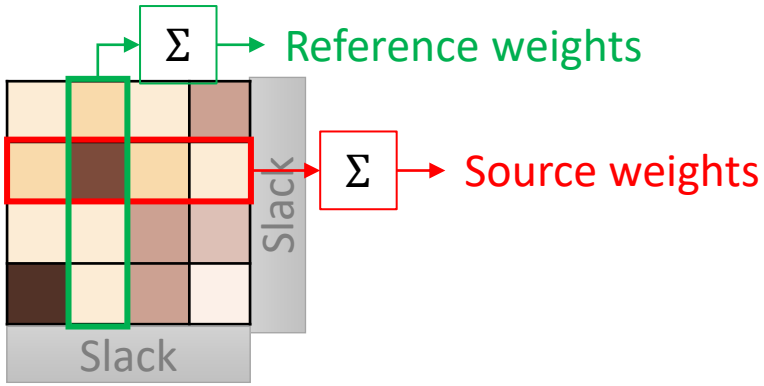
- Performance on Partially Visible Data (ModelNet40 dataset)

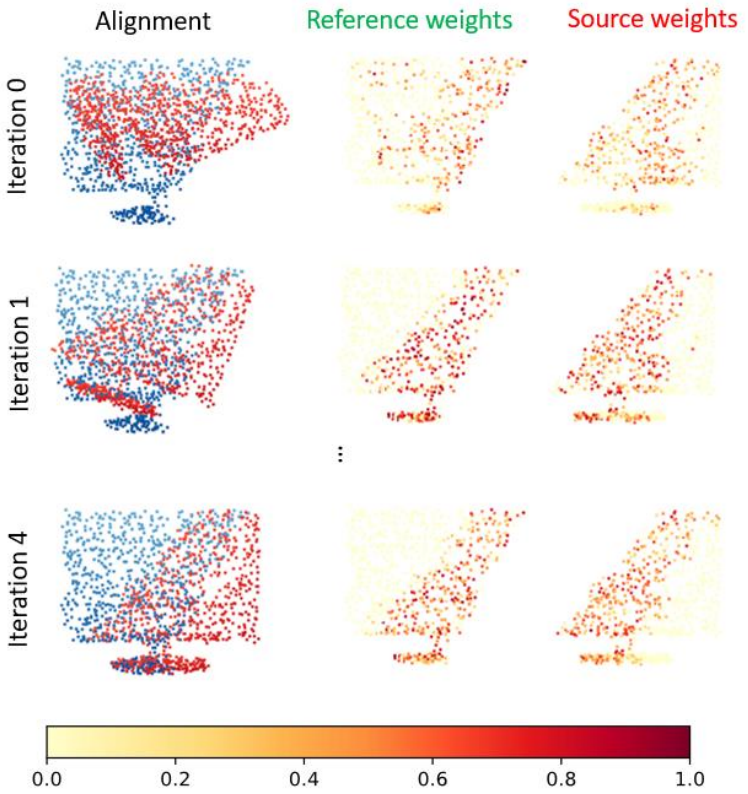
Method	Rotation error (°)	Translation error	Chamfer Distance metric
ICP	27.250	0.280	0.0153
RPM	19.551	0.212	0.0081
FGR	30.839	0.192	0.0119
PointNetLK	29.725	0.297	0.0235
DCP-v2	12.607	0.169	0.0113
Ours	**1.712**	**0.018**	**0.00085**



RPM-Net: Results

- We analyze the predicted inliers in the point clouds.
- At the first iteration, the weights are rather arbitrary.
- With further iterations, network focuses on the overlapping region.

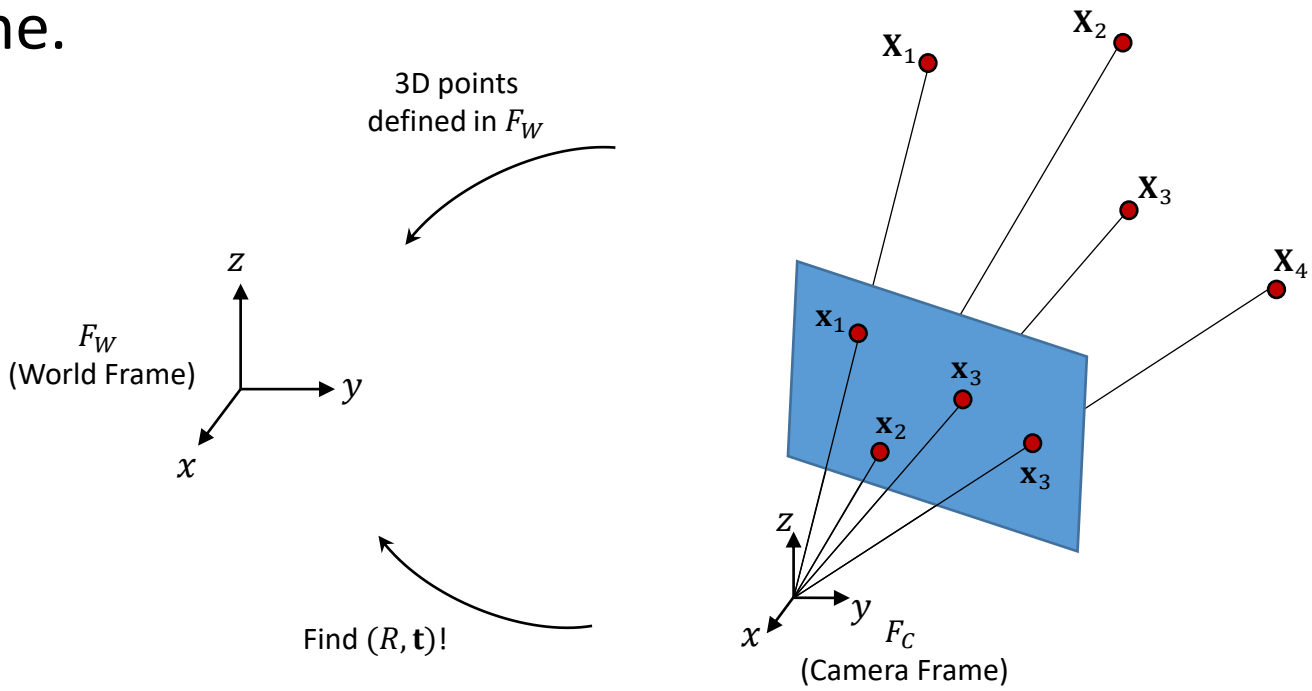




Perspective Pose Estimation Problem

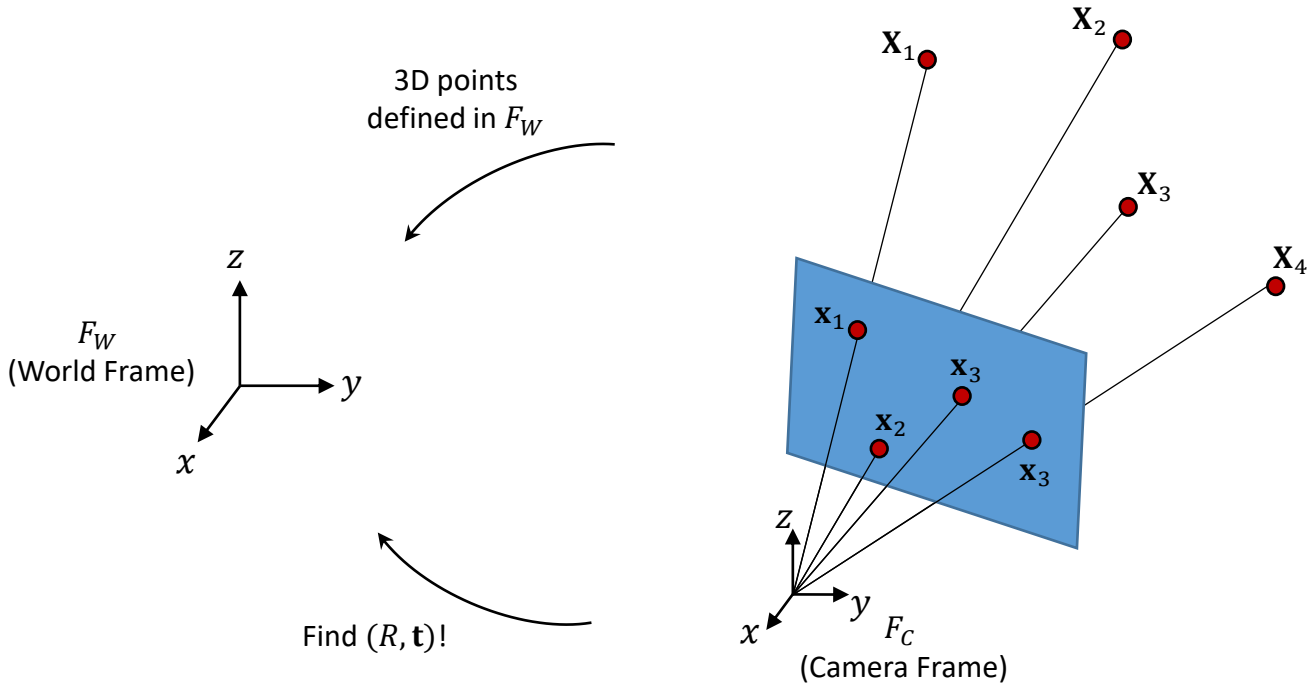
Given: a set of 3D points $\{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N\}$ defined in a world coordinate frame, and a set of 2D image points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$.

Find: the camera pose $(R, \mathbf{t})$ in the world coordinate frame.



Perspective Pose Estimation Problem

- This problem is also known as the "Perspective-n-Point" or PnP problem.



Two Important Steps

1. Establish 2D-3D correspondences, i.e. $\{\mathbf{X}_i \leftrightarrow \mathbf{x}_i\}$.

Remarks: Challenging to establish due to the image and point cloud cross-modality.

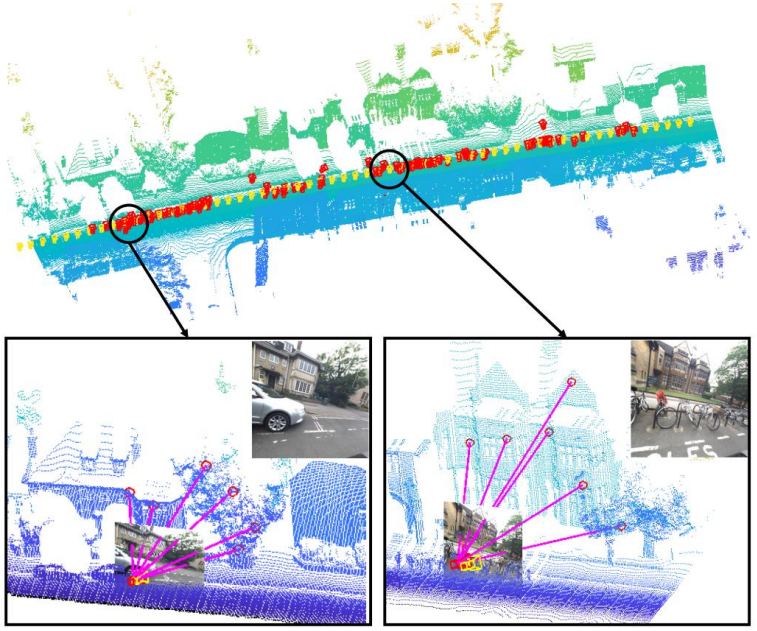
2. Solve for the camera pose $(R, \mathbf{t})$ using the 2D-3D correspondences.

Remarks: Well-established solutions exist for this geometry-based problem. But will fail badly if $\{\mathbf{X}_i \leftrightarrow \mathbf{x}_i\}$ is wrong!

Learning Cross-Modal 2D-3D Correspondences

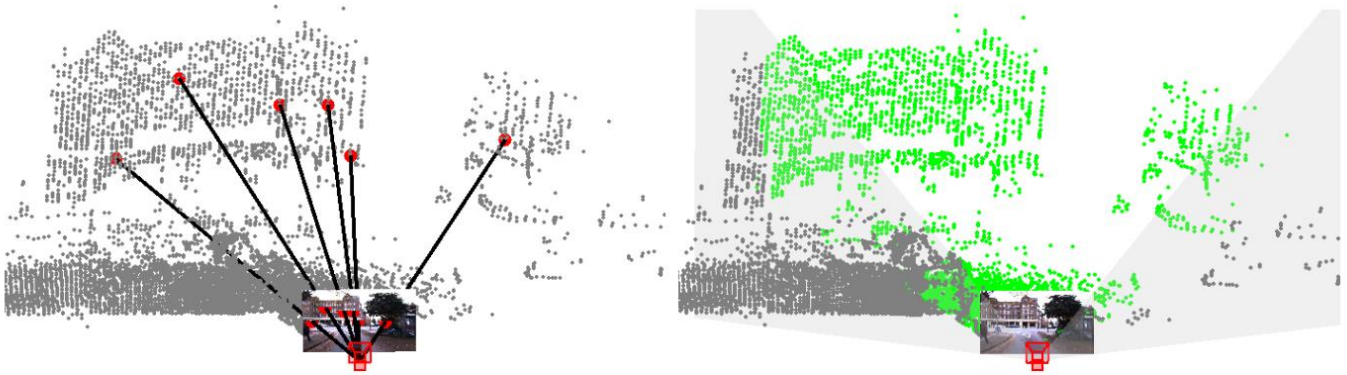
- Feng et al. proposed to directly learn the cross-modal descriptors to match 2D SIFT and 3D ISS keypoints.

Remarks:
This leads to low matching counts due to the inherent difference in modality!



Mengdan Feng, Sixing Hu, Marcelo H Ang, and Gim Hee Lee.
2d3d-matchnet: Learning to match keypoints across 2d image and 3d point cloud, In ICRA 2019.

DeepI2P: Image-to-Point Cloud Registration via Deep Classification



- Instead of detecting and matching features across modalities, we convert the registration problem into a classification problem.

Jiaxin Li, Gim Hee Lee, **DeepI2P: Image-to-Point Cloud Registration via Deep Classification**, In CVPR 2021

DeepI2P Overview

- Two-stage “Frustum classification + Pose Optimization” pipeline:

1. A classification neural network is designed to label whether the projection of each point in the point cloud is within or beyond the camera frustum.

2. These labeled points are subsequently passed into a novel inverse camera projection solver to estimate the relative pose.

Frustum Classification Network

- **Input:** a pair of image I and point cloud $P \in \{\mathbf{P}_1, \mathbf{P}_2, \cdots, \mathbf{P}_N \mid \mathbf{P}_n \in \mathbb{R}^3\}$.

- **Output:** is a per-point classification for P.

  - Assign a label to each point, $L^c = \{l_1^c, l_2^c, \cdots, l_N^c\}$, where $l_n^c \in \{0, 1\}$.

  - $l_n^c = 0$ if the point $\mathbf{P}_n$ is projected to outside the image I, and vice versa.

Frustum Classification Network

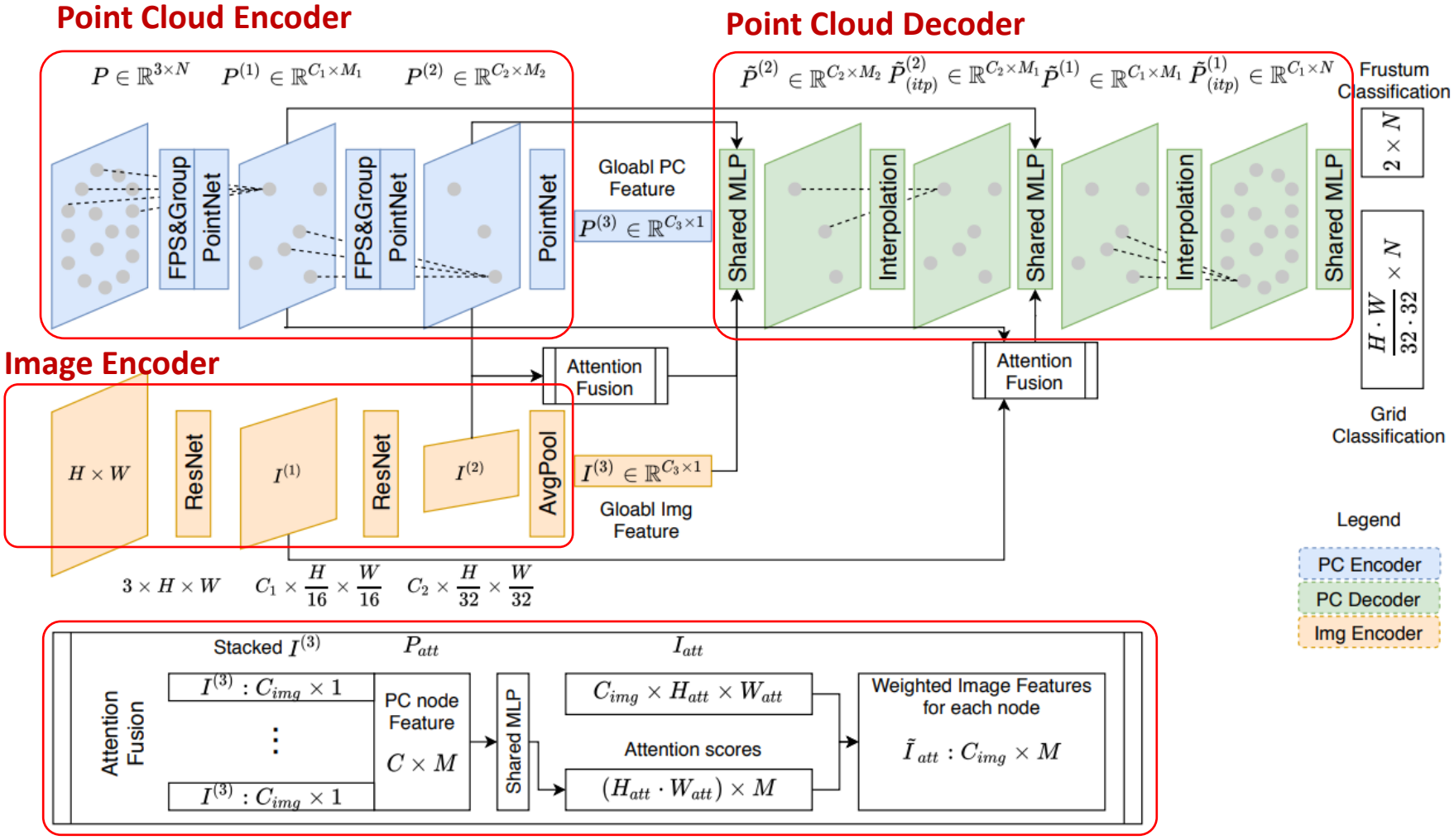


Image-Point Cloud Attention Fusion

Pose Optimization

- **Given:** the point cloud P, frustum predictions $L^c = \{l_1^c, l_2^c, \cdots, l_N^c\}$, $l_n^c \in \{0, 1\}$ and camera intrinsic matrix K.

- **Find:** pose of the camera in the point cloud reference frame, denoted as $\hat{G} \in \mathrm{SE}(3)$.

Pose Optimization

- Formally, we can find the optimal pose as:

$$\hat{G} = \underset{G \in \mathrm{SE}(3)}{\arg\max} \sum_{i=1}^{N} \big(f(\mathbf{P}_i; G, K, H, W) - 0.5\big)\big(\hat{l}_i^c - 0.5\big).$$

where

$$f(\mathbf{P}_i; G, K, H, W) =$$
$$\begin{cases} 1 & : 0 \le p'_{x_i} \le W - 1, 0 \le p'_{y_i} \le H - 1, z'_i > 0 \\ 0 & : \text{otherwise} \end{cases},$$

which assigns a label of 1 to a point $\mathbf{P}_i$ that projects within the image, and 0 otherwise.

Pose Optimization

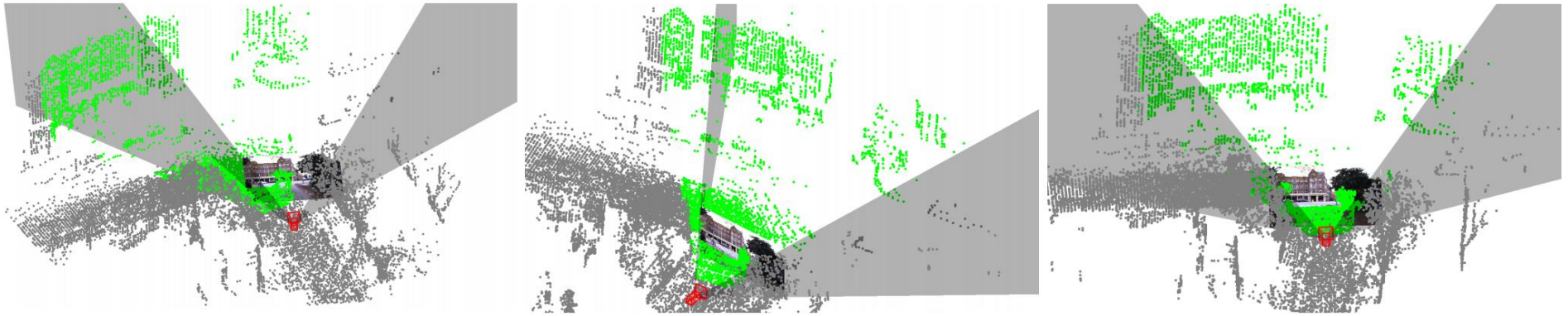
$$\hat{G} = \underset{G \in \mathrm{SE}(3)}{\arg\max} \sum_{i=1}^{N} \big(f(\mathbf{P}_i; G, K, H, W) - 0.5\big)\big(\hat{l}_i^c - 0.5\big).$$

- Intuitively, we seek to find the optimal pose $\hat{G}$ such that all 3D points with label $\hat{l}_i^c = 1$ from the network are projected into the image, and vice versa.

- We reformulate the cost function into an unconstrained optimization (see paper for detail) that can be solved using Gauss-Newton solver.

Pose Optimization

- Visualizations of the Gauss-Newton at iteration 0 / 40 / 80 from left to right. Green points are classified as inside image FoV.



Baselines

- **Grid Cls. + PnP** is the result of our “Grid classification + PnP” baseline method.
- **Direct Regression** uses a deep network to directly regress the relative poses.
- **Monodepth2+USIP** converts the cross-modality registration problem into point cloud-based registration by using Monodepth2 to estimate a depth map from a single image.

Jiaxin Li, Gim Hee Lee, **USIP: Unsupervised Stable Interest Point Detection from 3D Point Clouds**, In ICCV 2019.

Baselines

- **Monodepth2+GT-ICP** acquires a depth map with absolute scale in the same way as Monodepth2+USIP.

  Remark: it uses Iterative Closest Point (ICP) with ground truth initial pose to estimate the pose between the depth map and point cloud.

- 2D3D-MatchNet is the only prior work for cross-modal image-to-point cloud registration to our best knowledge.

  Remark: the rotation between camera and Lidar is almost zero in their experiment setting.

Results

- Our image-to-point cloud registration approach is evaluated with Oxford Robotcar and KITTI dataset.

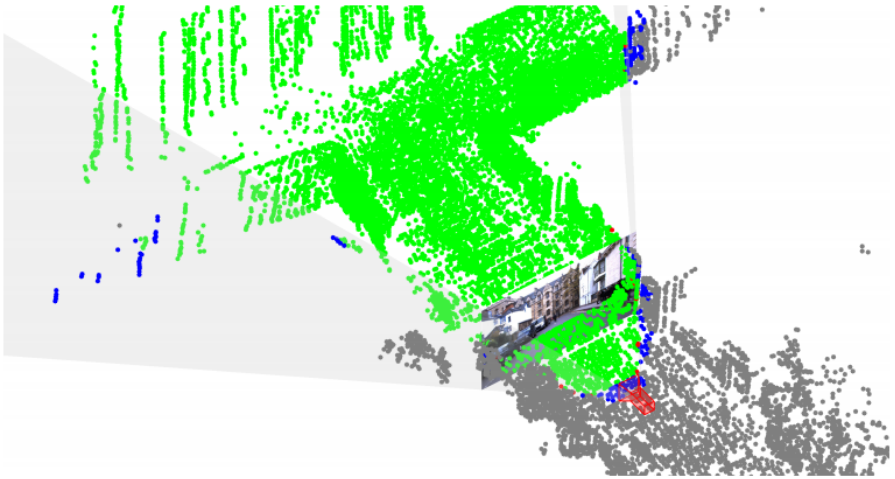
Table 1. Registration accuracy on the Oxford and KITTI datasets.

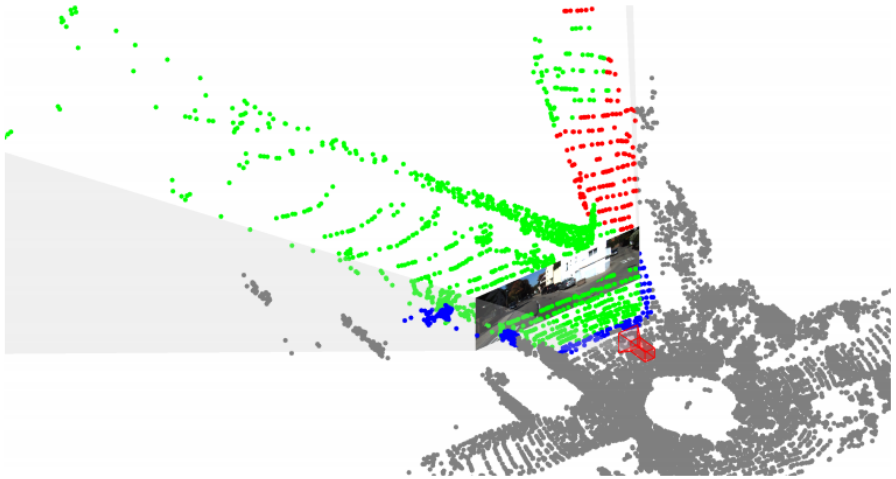
	Oxford		KITTI	
	RTE (m)	RRE (°)	RTE (m)	RRE (°)
Direct Regression	5.02 ± 2.89	10.45 ± 16.03	4.94 ± 2.87	21.98 ± 31.97
MonoDepth2 [14] + USIP [22]	33.2 ± 46.1	142.5 ± 139.5	30.4 ± 42.9	140.6 ± 157.8
MonoDepth2 [14] + GT-ICP	$\mathbf{1.3 \pm 1.5}$	6.4 ± 7.2	$\mathbf{2.9 \pm 2.5}$	12.4 ± 10.3
2D3D-MatchNet [11] (No Rot[§])	1.41	6.40	NA	NA
Grid Cls. + PnP	1.91 ± 1.56	5.94 ± 10.72	3.22 ± 3.58	10.15 ± 13.74
Frus. Cls. + Inv.Proj. 3D	2.27 ± 2.19	15.00 ± 13.64	3.17 ± 3.22	15.52 ± 12.73
Frus. Cls. + Inv.Proj. 2D	1.65 ± 1.36	$\mathbf{4.14 \pm 4.90}$	3.28 ± 3.09	$\mathbf{7.56 \pm 7.63}$

[§]Point clouds are not randomly rotated in the experiment setting of 2D3D-MatchNet [11].

Results

- 3D Visualization of the frustum classification and inverse camera projection on the Oxford (Left) and KITTI (Right).





Summary

- We have looked at how to:

1. Define implicit and explicit surfaces.
2. Explain the pro and cons of point clouds representation.
3. Describe how deep learning can be used on point cloud processing.
4. Define the tasks on 3D point cloud processing, i.e., place-recognition; keypoint detector and descriptor; 3D object detection; 3D semantic segmentation; point cloud registration.