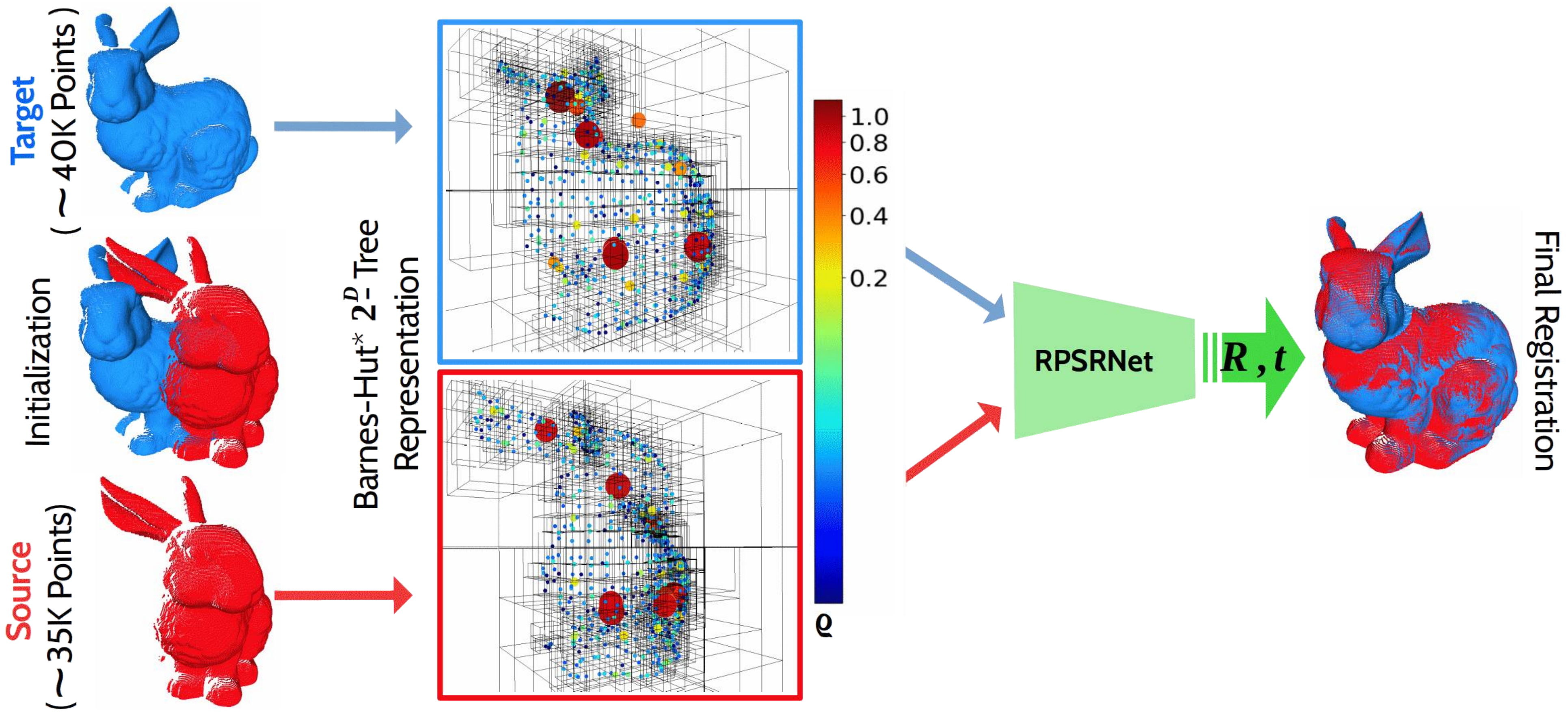
RPSRNet: End-to-End Trainable Rigid Point Set Registration Network using Barnes-Hut 2^D-Tree Representation

Sk Aziz Ali , Kerem Kahraman, Gerd Reis, Didier Stricker
(TU Kaiserslautern, German Research Center for Artificial Intelligence)

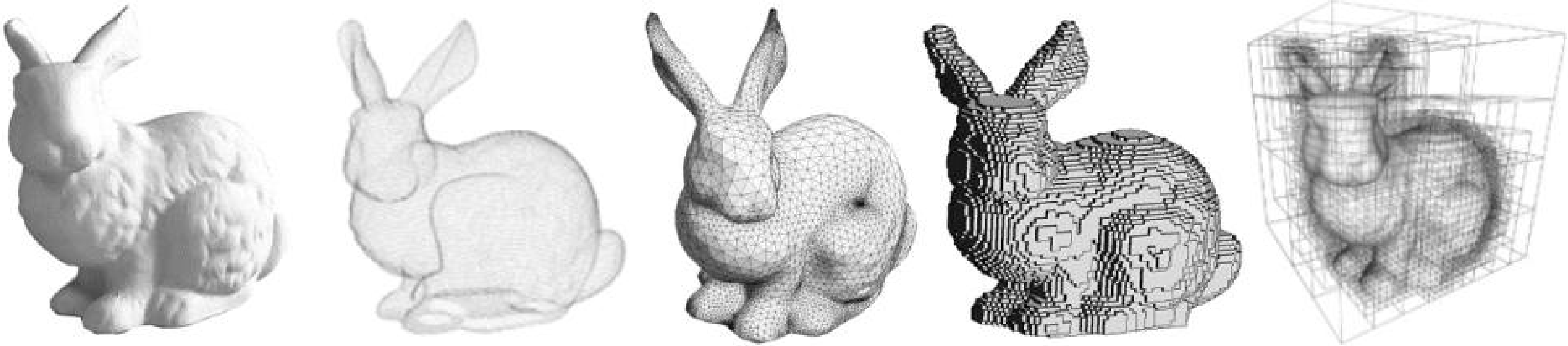
CVPR VIRTUAL JUNE 19-25 2021

A supervised deep learning framework (end-to-end trainable network)
For Rigid Point Set Registration (RPSR)



Motivation

Input Representations



CAD Model   Dense Point Cloud   Mesh   Voxels   Octree

Deep-Learning-Based 3D Point Cloud Processing Tasks

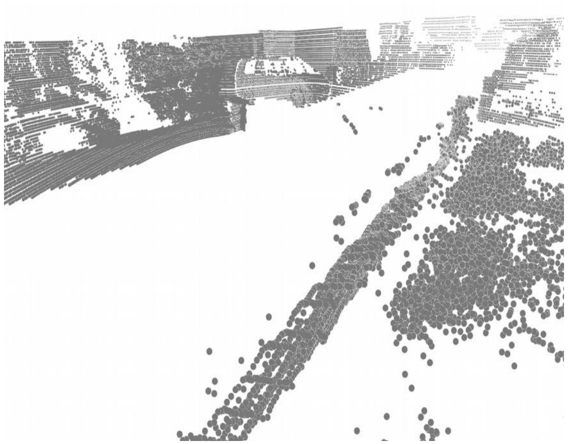
Classification, Segmentation, Matching/registration, Shape Completion, MODT

Data Dimension, Memory Efficiency, Shape Details, Computational Efficiency !

Input Representations

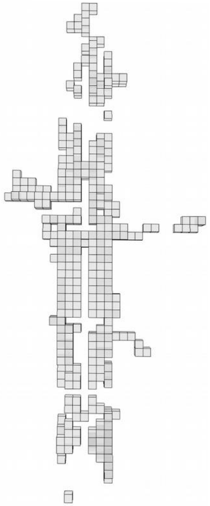
X

CAD Model

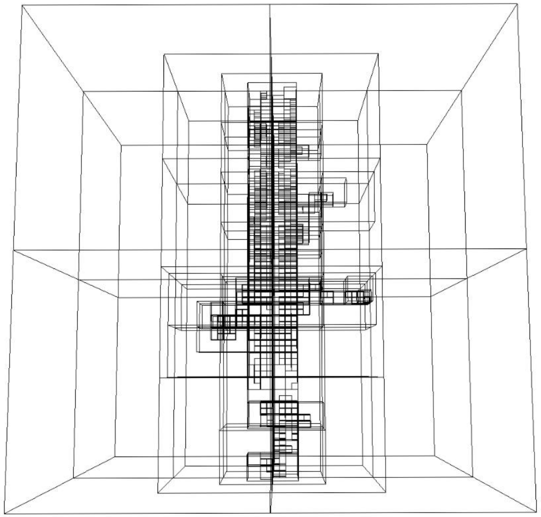


Dense Point Cloud

BEV (Birds Eye View)



Voxels



Octree

Deep-Learning-Based 3D Point Cloud Processing Tasks

Classification, Segmentation, Matching/registration, Shape Completion, MODT

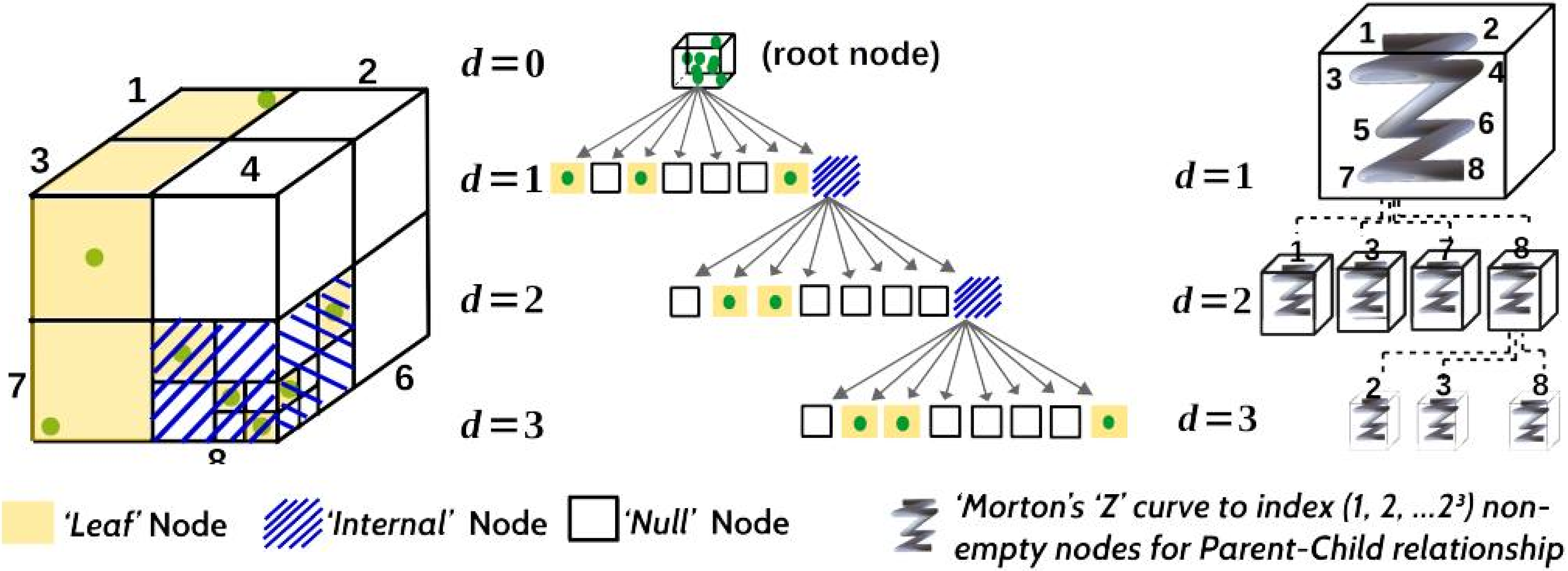
Data Dimension, Memory Efficiency, Shape Details, Computational Efficiency !

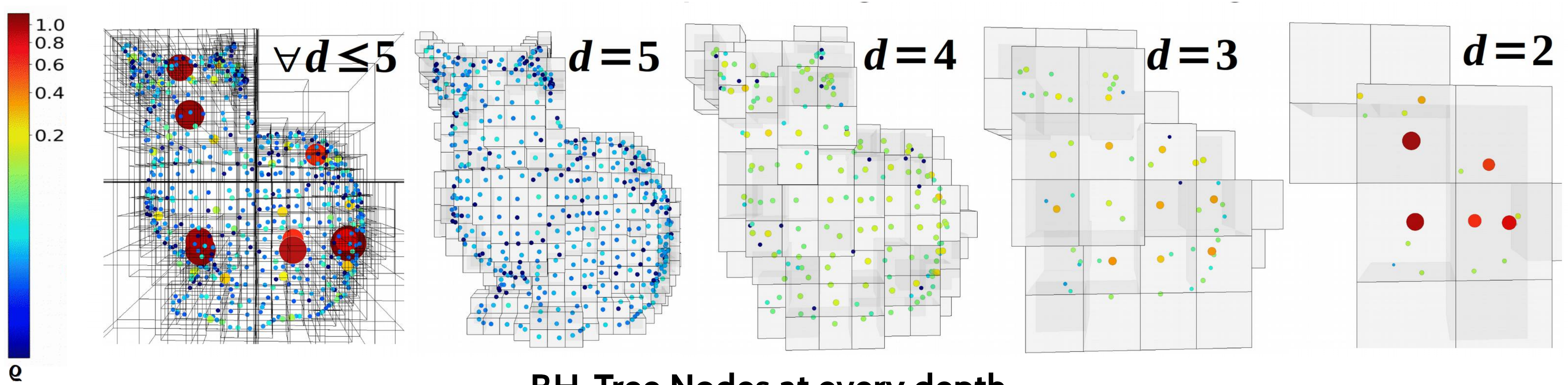
Memory Efficiency	Robust to Noise	Indoor+ Outdoor	Real-time inference	Quick-Training Time	Accept Original Point Size	Partial Data Registration
[1]DCP	[1]DCP		[1]DCP	[1]DCP		[1]DCP
		[2]PointNetLK				
		[3]DGR			[3]DGR	[3]DGR
	[4]RPMNet					[4]RPMNet
[5]PPFFoldNet+ RelativeNet			[5]PPFFoldNet+ RelativeNet		[5]PPFFoldNet+ RelativeNet	[5]PPFFoldNet+ RelativeNet
[6]DeepGMR	[6]DeepGMR		[6]DeepGMR	[6]DeepGMR		
[7]ICP		[7]ICP	[7]ICP	[7]ICP		
[8]CPD	[8]CPD					
[9]FilterReg						[9]FilterReg
	[10]GA					
[11]FGR				[11]FGR	[11]FGR	[11]FGR
RPSRNet	RPSRNet	RPSRNet	RPSRNet	RPSRNet	RPSRNet	RPSRNet

[1] Y. Wang et. al, **ICCV'19;** **[2]** Y. Aoki et. al, **CVPR'19;** **[3]** C. Choy et. al, **CVPR'20;**
[4] Z. J. Yew et. al, **CVPR'20;** **[5]** H. Deng et. al, **CVPR'19;** **[6]** W. Yuan et. al, **ECCV'20;**
[7] P. J. Besl, **TPAMI'92;** **[8]** A. Myronenko, **TPAMI'10;** **[9]** W. Gao et. al, **CVPR'19;**
[10] V. Golyanik et. al, **CVPR'16;** **[11]** Q. Y. Zhou et. al, **ECCV'16**

Input Representation & RPSRNet Framework

BH-Tree Node Partitioning and Tree Traversal Indexing





BH-Tree Nodes at every depth

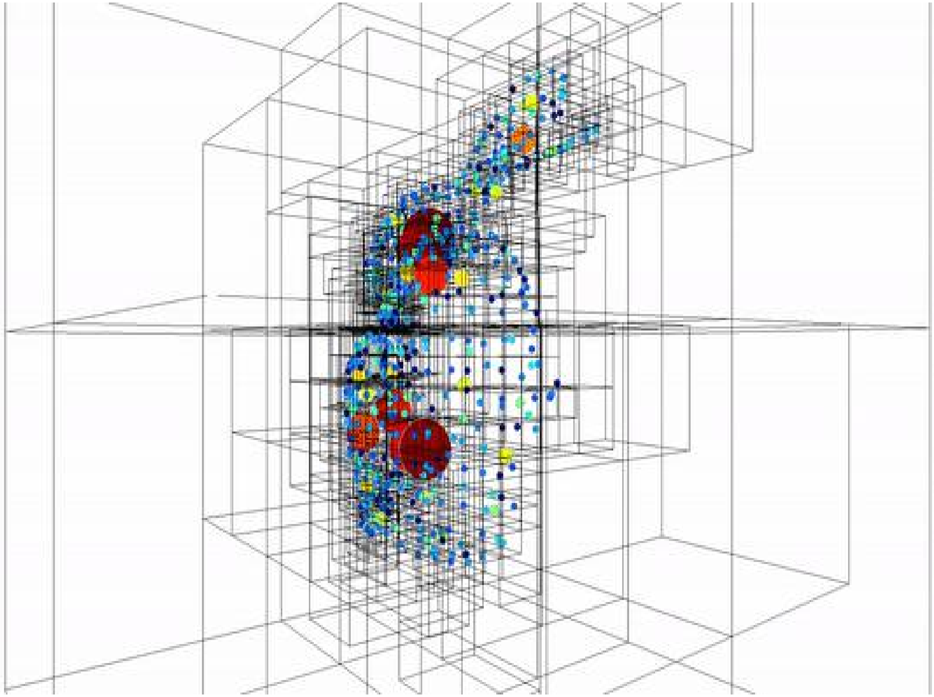
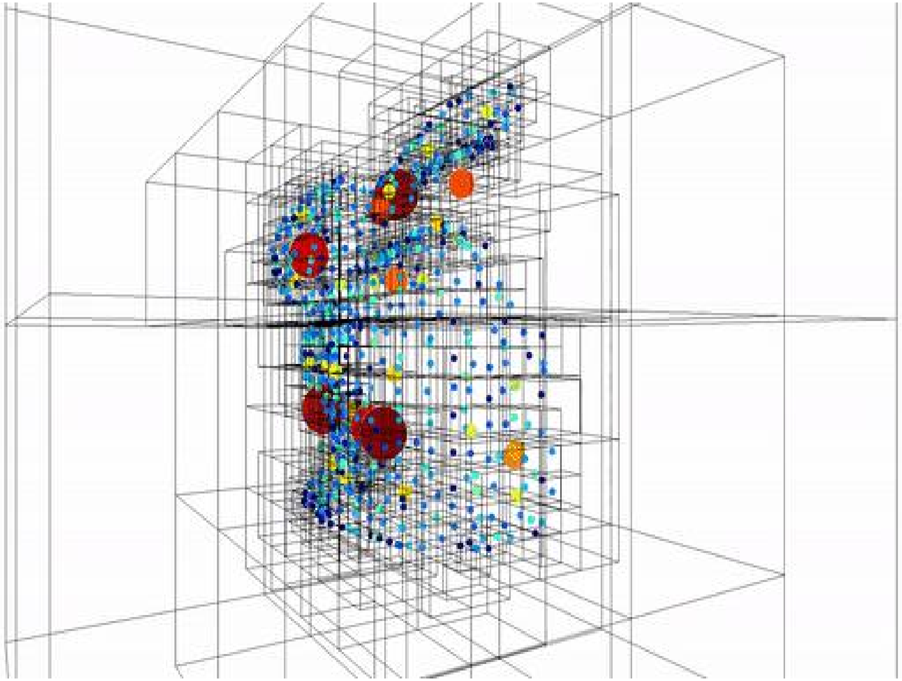
The RPSR problem is often formulated as optimization of cost function $\mathbf{U}(\mathbf{R}, \mathbf{t})$ in the form of globally **multiply-linked** correspondence distance errors between $\mathbf{X}$ and $\mathbf{Y}$:

$$\mathbf{U}(\mathbf{R}, \mathbf{t}, \mathbf{X}, \mathbf{Y}) = \sum_{i,j} \omega_{ij} \|(\mathbf{R}\mathbf{y}_i + \mathbf{t}) - \mathbf{x}_j\|_2^2,$$

$$\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \in \mathbb{R}^{N\times 3} \rightarrow \tau^{\mathbf{X}}, \quad \mathbf{M}_d^{\mathbf{Y}} = \{\mu_{d,l}^{y}\}, \rho_d^{\mathbf{X}-} = \{\varrho_{d,l}^{x-}\}, \quad \mathbf{N}_d^{\mathbf{X}} = \{\mathbf{n}_{d,l}^{x}\}$$

$$\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_M\} \in \mathbb{R}^{M\times 3} \rightarrow \tau^{\mathbf{Y}}, \quad \underbrace{\mathbf{M}_d^{\mathbf{X}} = \{\mu_{d,l}^{x}\}}_{\text{CoMs}}, \underbrace{\rho_d^{\mathbf{Y}-} = \{\varrho_{d,l}^{y-}\}}_{\text{IDs}}, \quad \underbrace{\mathbf{N}_d^{\mathbf{Y}} = \{\mathbf{n}_{d,l}^{y}\}}_{\text{Node Idx.}}$$

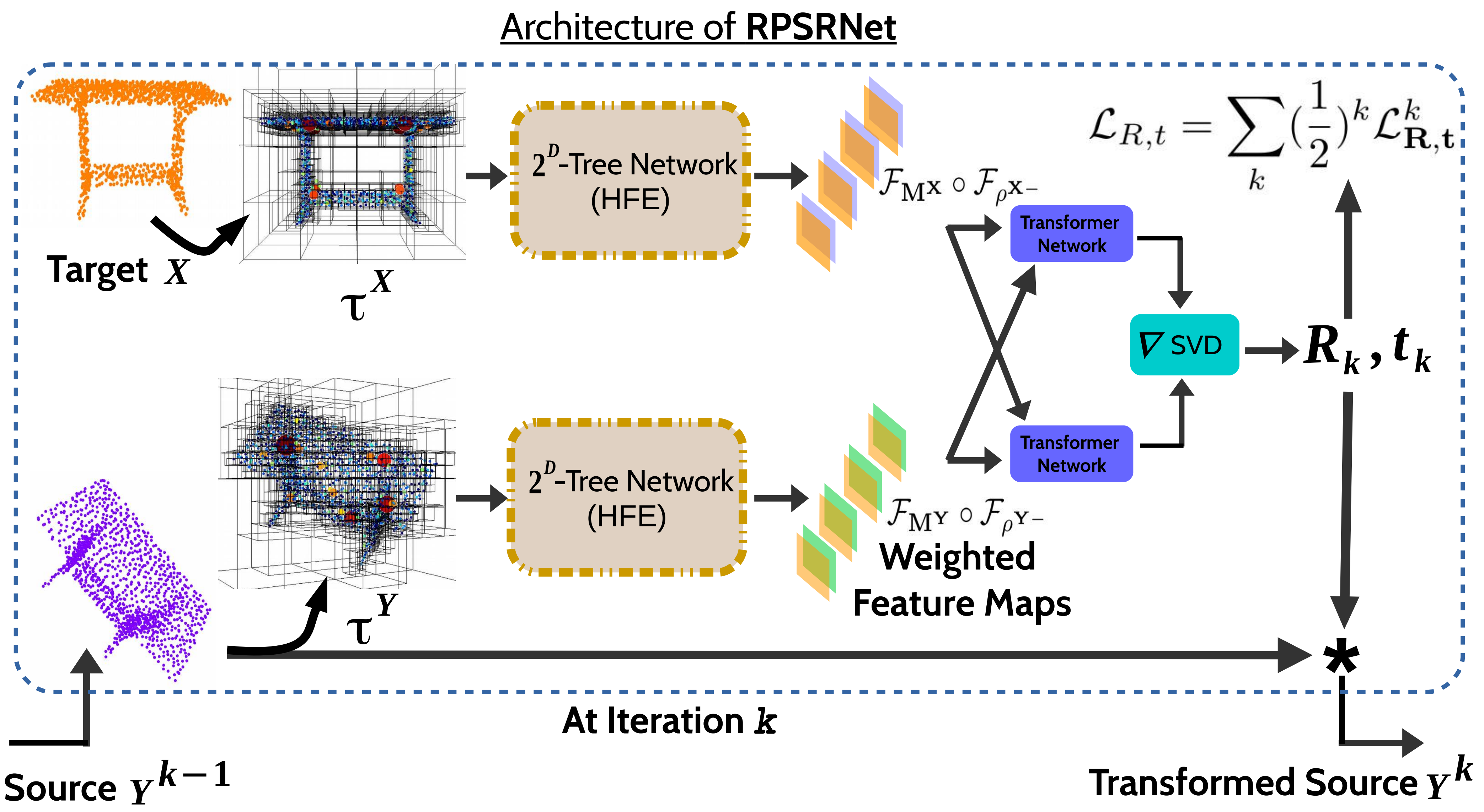
multiply-linked correspondence distance errors are now applicable on the CoMs of the non-empty nodes at every depth ($\sum_{i,j} \rightarrow \sum_{d} \sum_{l,\hat{l}}$)

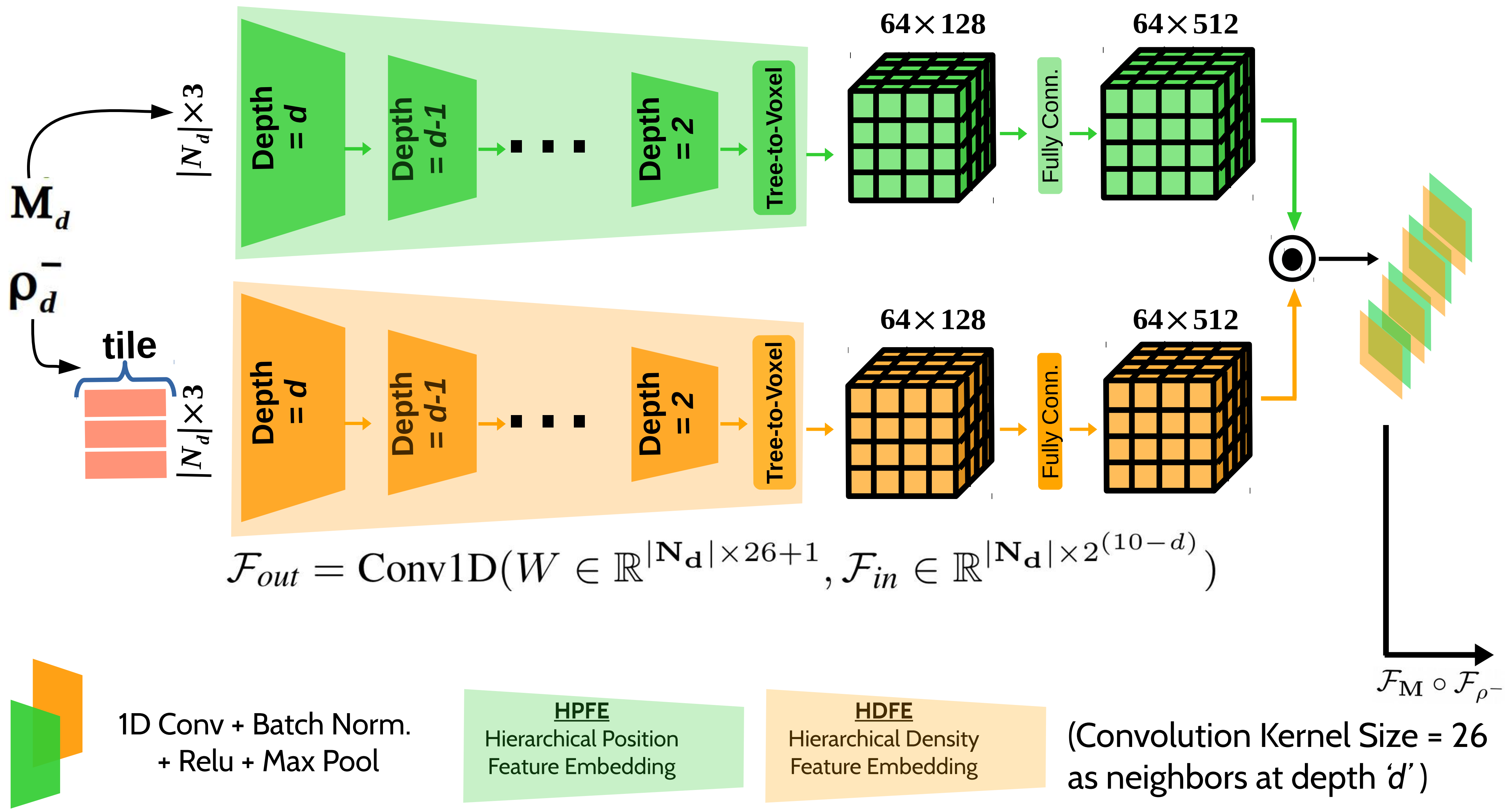
multi-scale sum of mean-squared distance errors between the CoMs

$$\mathbf{U}(\mathbf{R},\mathbf{t},\tau^{\mathbf{X}},\tau^{\mathbf{Y}}) = \sum_{d} \sum_{l,\hat{l}} \varrho^{y-}_{d,l}\,\varrho^{x-}_{d,\hat{l}}\,\|\left(\mathbf{R}\mu^{y}_{d,l}+t\right)-\mu^{x}_{d,\hat{l}}\|_2^2$$

BH-trees **Depth** **Node Labels** **Inverse Node Densities** **Center of Masses of Nodes**



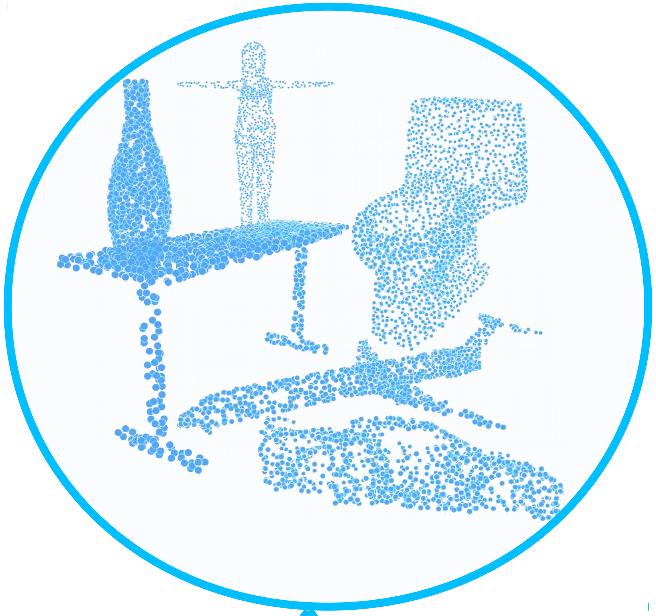
Architecture of RPSRNet
Target X
τ^X
2^D-Tree Network (HFE)
$\mathcal{F}_{M^X} \circ \mathcal{F}_{\rho^{X-}}$
$\mathcal{L}_{R,t} = \sum_k (\frac{1}{2})^k \mathcal{L}^k_{\mathbf{R},\mathbf{t}}$
Transformer Network
∇ SVD
R_k, t_k
Transformer Network
τ^Y
2^D-Tree Network (HFE)
$\mathcal{F}_{M^Y} \circ \mathcal{F}_{\rho^{Y-}}$
Weighted Feature Maps
At Iteration k
Source Y^{k-1}
Transformed Source Y^k



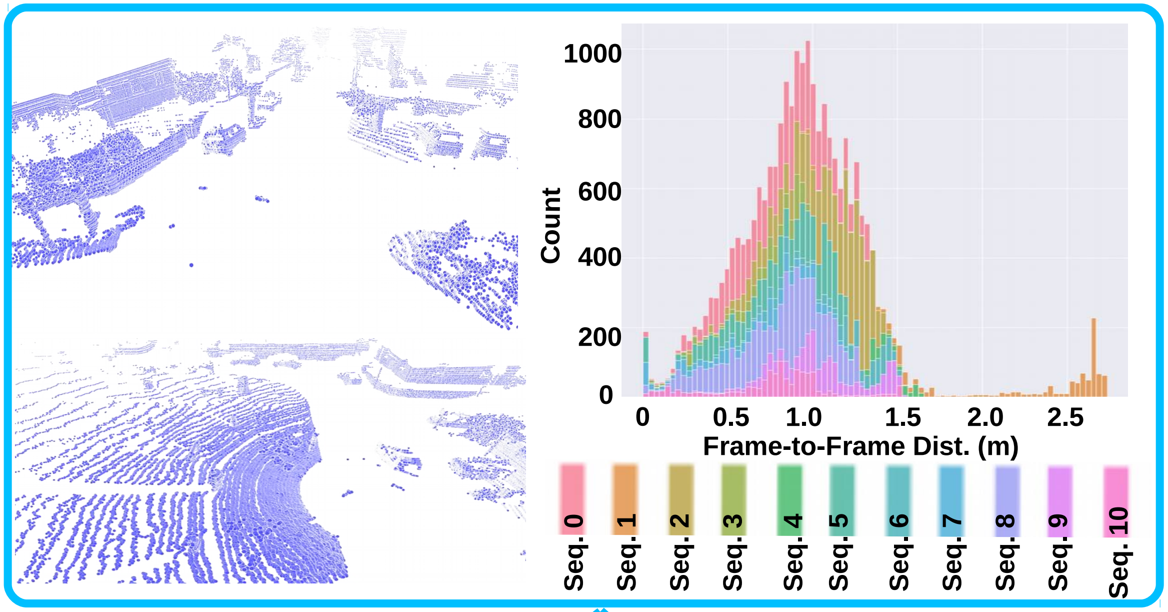
$\mathbf{M}_d$
$|N_d| \times 3$
ρ_d^-
tile
$|N_d| \times 3$
Depth = d
Depth = d-1
Depth = 2
Tree-to-Voxel
64×128
Fully Conn.
64×512
$\mathcal{F}_{out} = \mathrm{Conv1D}(W \in \mathbb{R}^{|\mathbf{N_d}| \times 26+1}, \mathcal{F}_{in} \in \mathbb{R}^{|\mathbf{N_d}| \times 2^{(10-d)}})$
$\mathcal{F}_{\mathbf{M}} \circ \mathcal{F}_{\rho^-}$
1D Conv + Batch Norm. + Relu + Max Pool
HPFE
Hierarchical Position Feature Embedding
HDFE
Hierarchical Density Feature Embedding
(Convolution Kernel Size = 26 as neighbors at depth 'd')

Experiments & Results

ModelNet40 Dataset



KITTI LiDAR Odometry Dataset



M1-seen **M2-unseen**

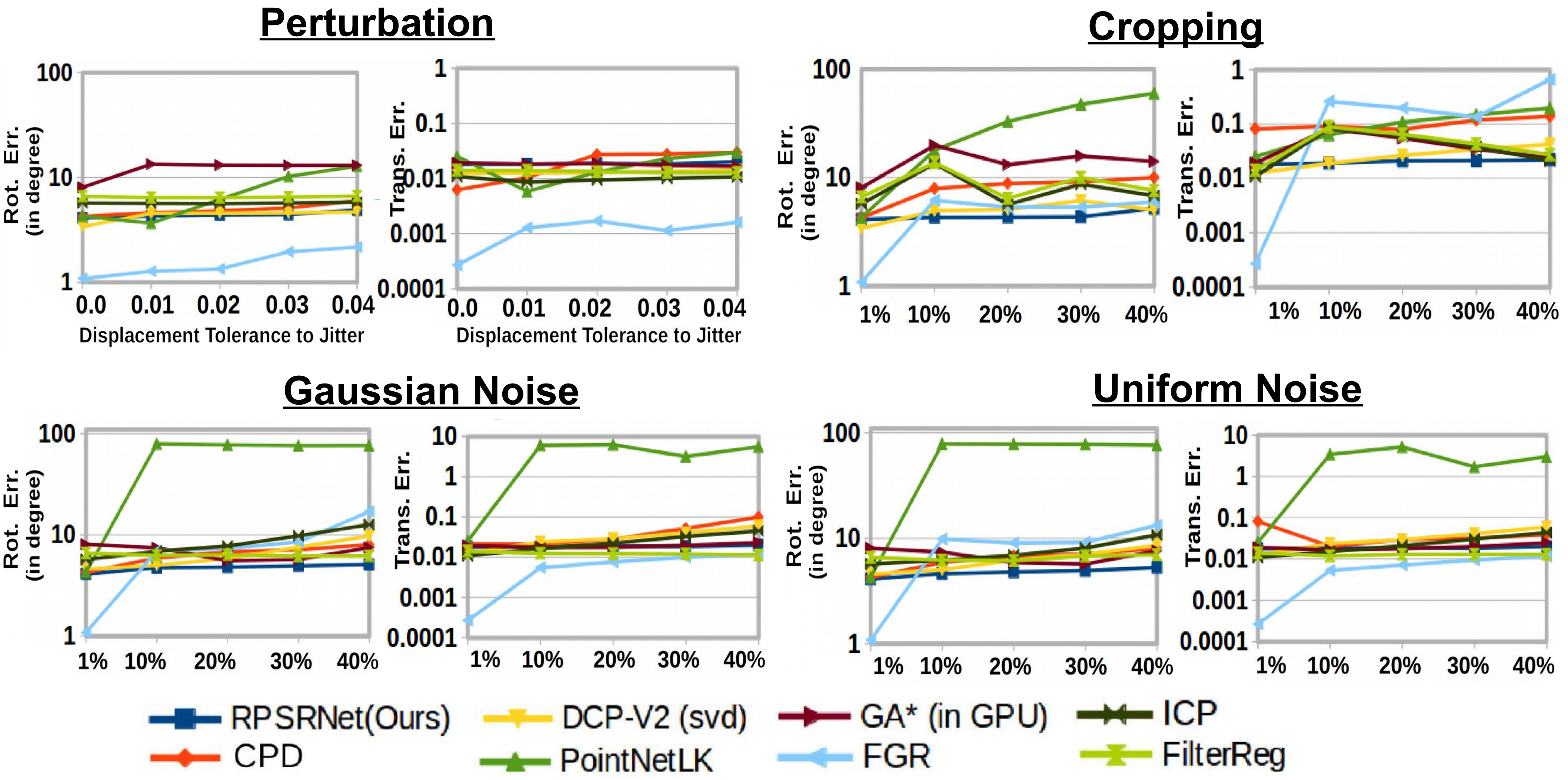
(**Data Disturbances:** Clean, Jitters, Copping, Gaussian Noise, Uniformly Dist. Noise)
*Each with 5 increasing levels

K1-w/o ground **K2-w ground**

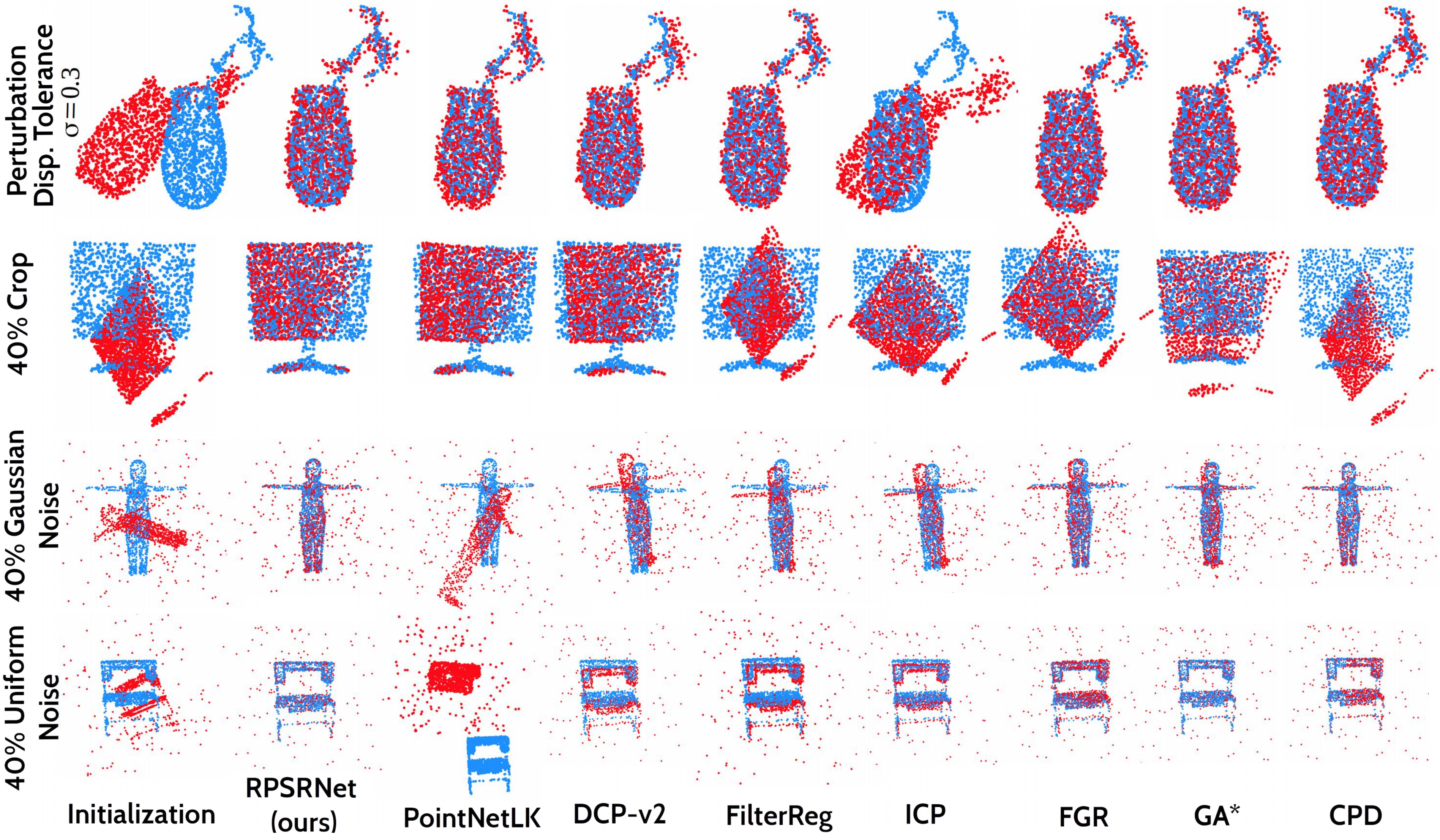
(**Train**: 70%; **Test**: 20%; **Validation**: 10%;)
Of random samples from every Seq.

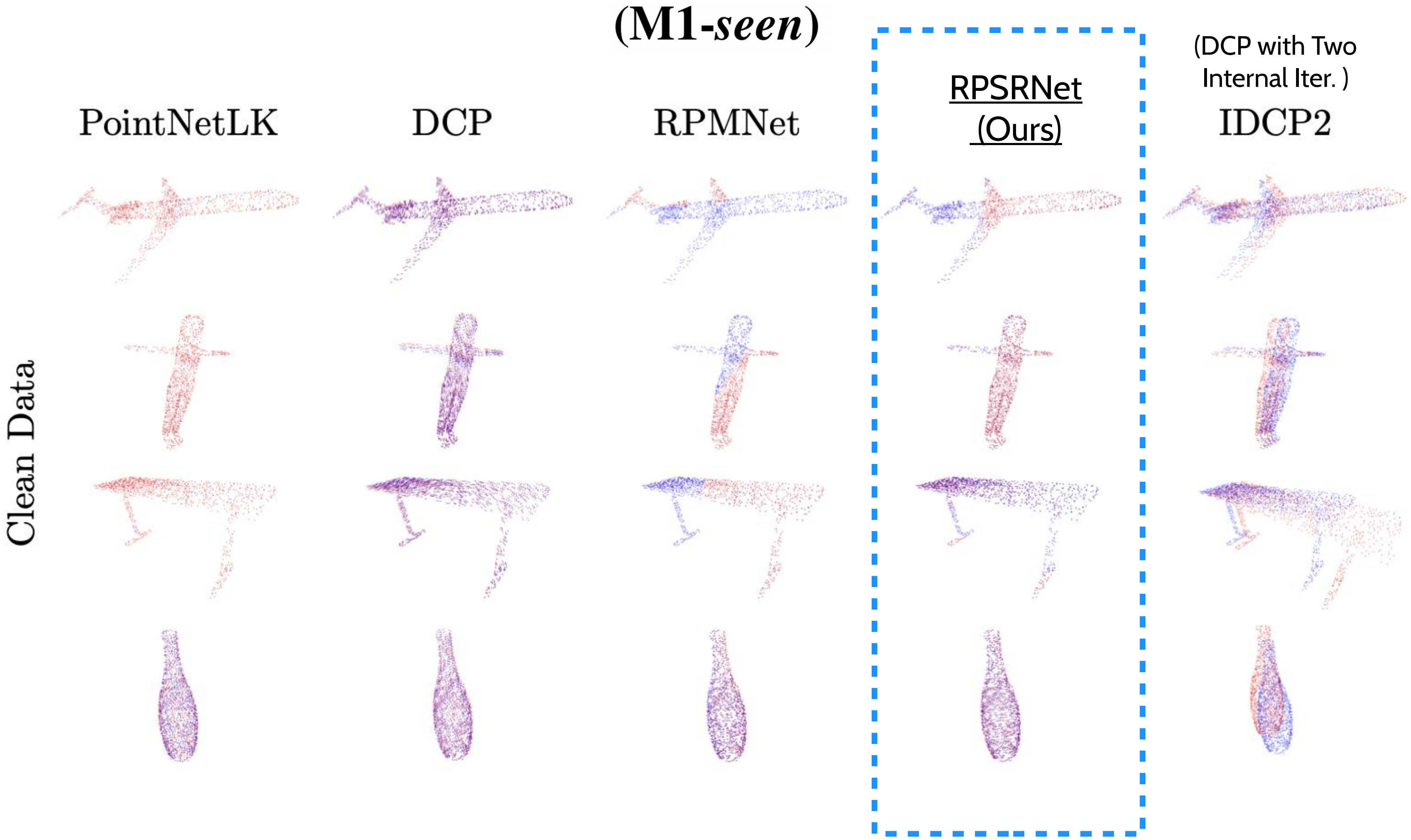
$$\varphi = \cos^{-1}\left(0.5(\mathrm{tr}\left(\mathbf{R}_{gt}^{T}\mathbf{R}\right) - 1)\right), \quad \mathbf{\Delta t} = \|\mathbf{t}_{gt} - \mathbf{t}\|.$$

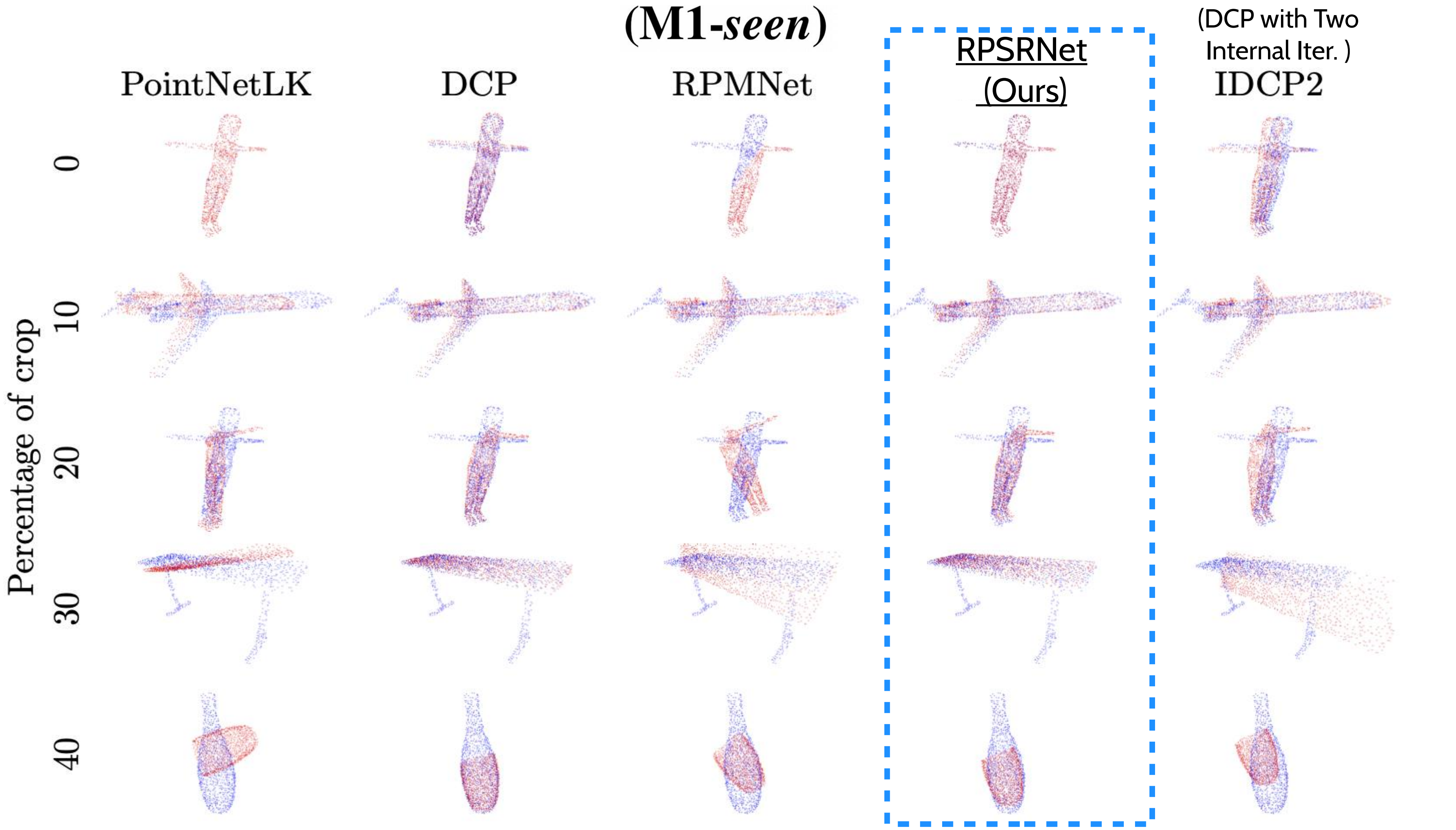
(M1-*seen*)

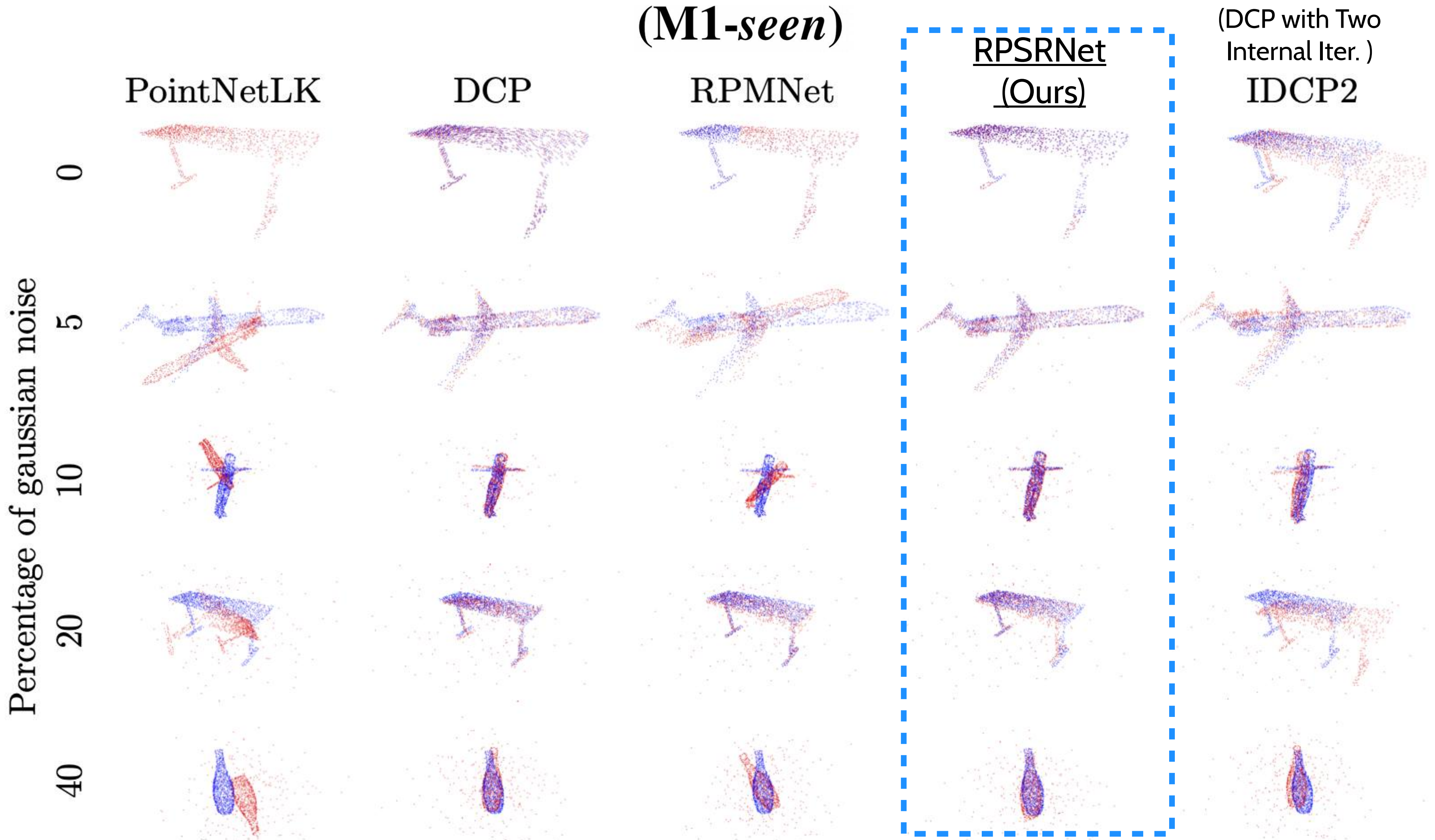

Perturbation
Cropping
Gaussian Noise
Uniform Noise
Rot. Err. (in degree)
Trans. Err.
Displacement Tolerance to Jitter
0.0
0.01
0.02
0.03
0.04
1%
10%
20%
30%
40%
100
10
1
0.1
0.01
0.001
0.0001
RPSRNet(Ours)
DCP-V2 (svd)
GA* (in GPU)
ICP
CPD
PointNetLK
FGR
FilterReg

(M1-*seen*)

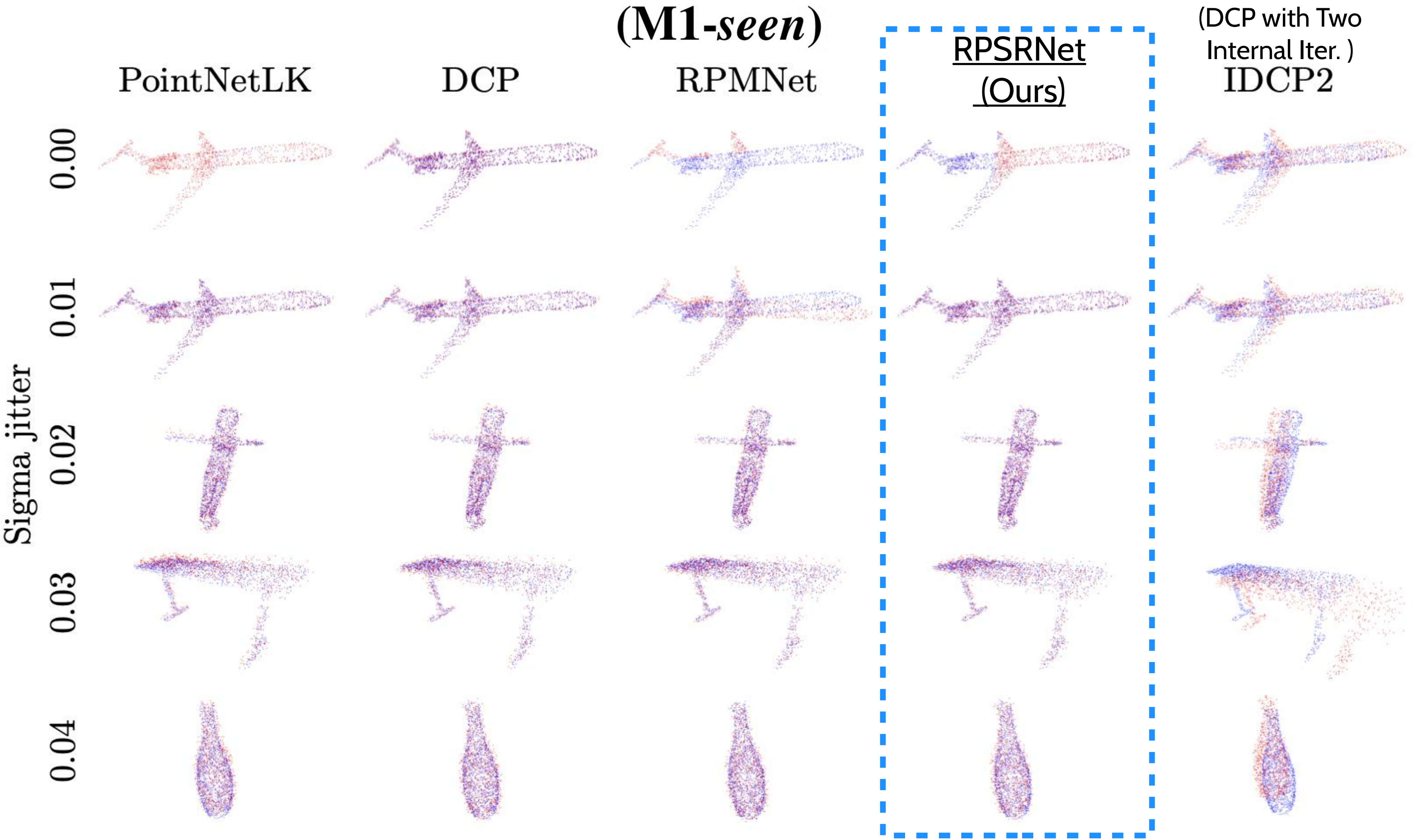

Perturbation
Disp. Tolerance
σ=0.3
40% Crop
40% Gaussian
Noise
40% Uniform
Noise
Initialization
RPSRNet
(ours)
PointNetLK
DCP-v2
FilterReg
ICP
FGR
GA*
CPD

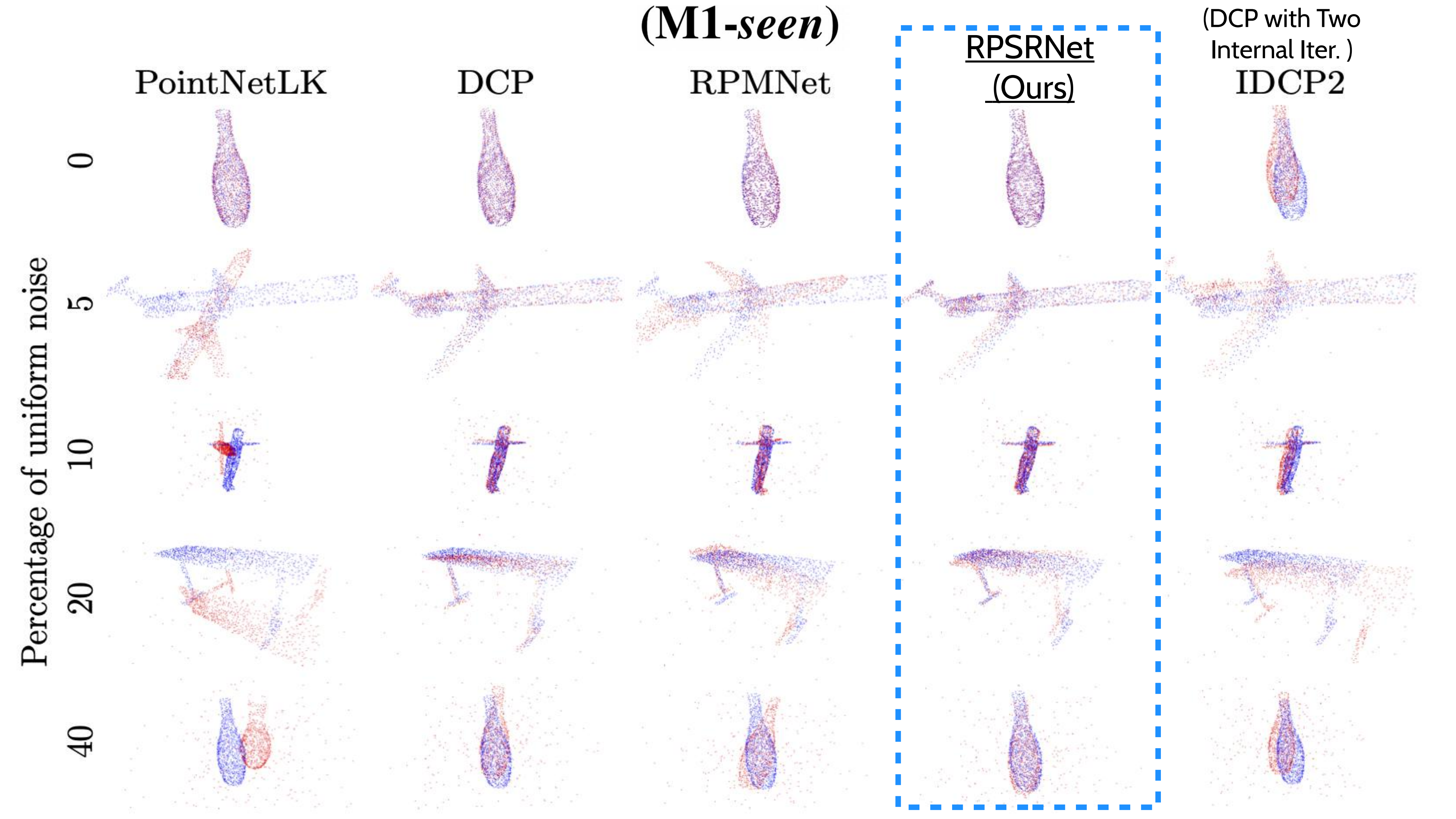

(M1-seen)
(DCP with Two Internal Iter.)
PointNetLK
DCP
RPMNet
RPSRNet (Ours)
IDCP2
Clean Data


(M1-seen)
PointNetLK
DCP
RPMNet
RPSRNet (Ours)
(DCP with Two Internal Iter.)
IDCP2
Percentage of crop
0
10
20
30
40



(M1-seen)
(DCP with Two Internal Iter.)
PointNetLK
DCP
RPMNet
RPSRNet (Ours)
IDCP2
Percentage of gaussian noise
0
5
10
20
40

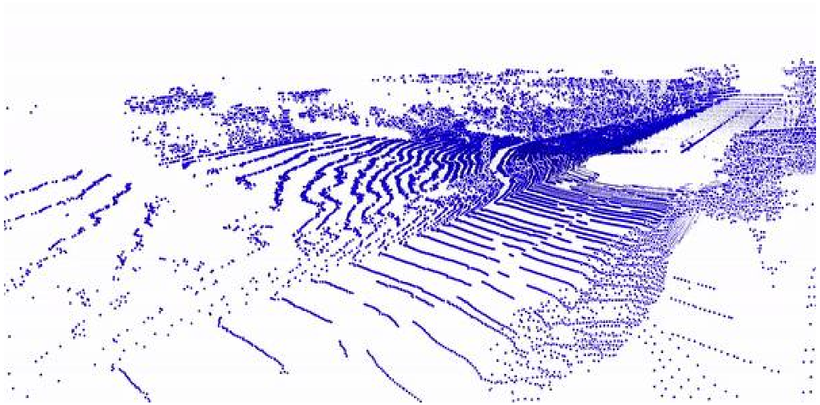

(M1-*seen*)
PointNetLK
DCP
RPMNet
RPSRNet (Ours)
(DCP with Two Internal Iter.)
IDCP2
Sigma jitter
0.00
0.01
0.02
0.03
0.04

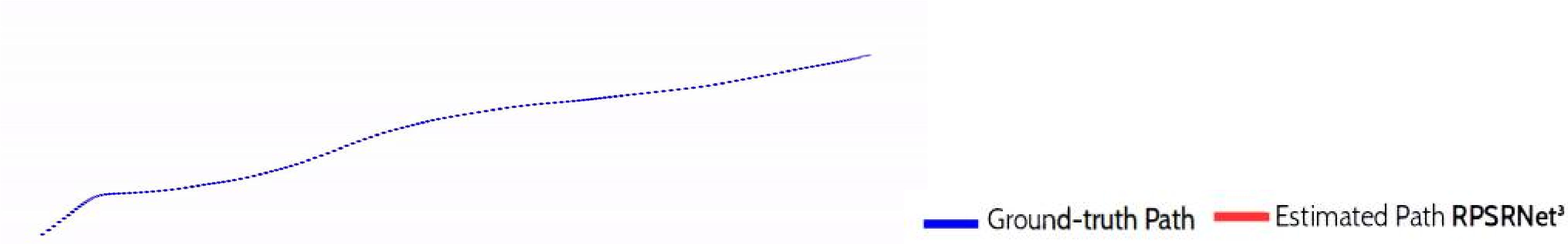


(M1-*seen*)
PointNetLK
DCP
RPMNet
RPSRNet (Ours)
(DCP with Two Internal Iter.)
IDCP2
Percentage of uniform noise
0
5
10
20
40

KITTI LiDAR Odometry (Without Pose Graph Optimization and Loop Closure)

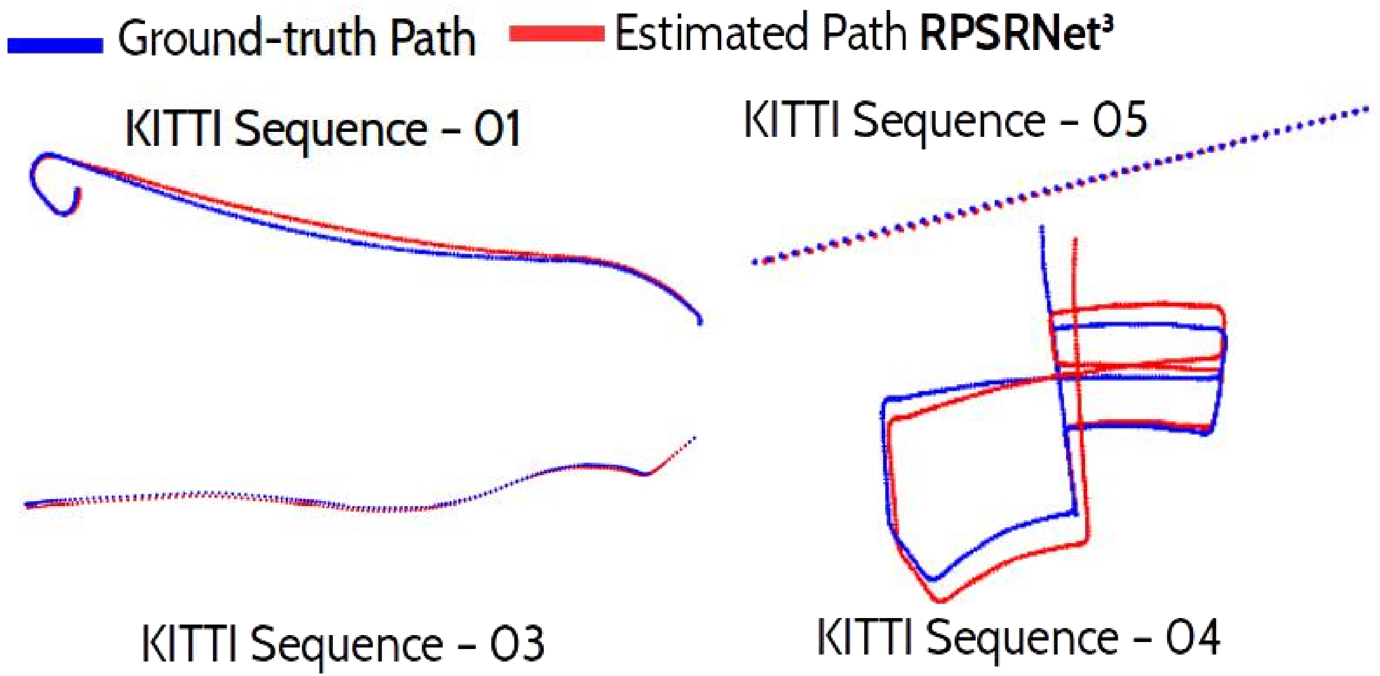
Seq.	CPD [44]	GA* [25]	FGR [77]	ICP [10]	FilterReg [21]	DCP-v2 [66]	PointNetLK [5]	Single Iteration RPSRNet[1] (ours)	Three Iterations RPSRNet[3] (ours)
	$\varphi_{\text{rmse}}, \Delta t_{\text{rmse}}$	$\varphi_{\text{rmse}}, \Delta t_{\text{rmse}}$	$\varphi_{\text{rmse}}, \Delta t_{\text{rmse}}$	$\varphi_{\text{rmse}}, \Delta t_{\text{rmse}}$	$\varphi_{\text{rmse}}, \Delta t_{\text{rmse}}$	$\varphi_{\text{rmse}}, \Delta t_{\text{rmse}}$	$\varphi_{\text{rmse}}, \Delta t_{\text{rmse}}$	$\varphi_{\text{rmse}}, \Delta t_{\text{rmse}}$	$\varphi_{\text{rmse}}, \Delta t_{\text{rmse}}$
mean	3.55, 1.08	3.30, 1.0	3.29, 0.85	3.15, 1.08	3.08, 0.77	2.92, 0.89	4.02, 1.12	3.13, 0.88	**2.22, 0.58**
	3.03, 1.07	*2.94, 1.02*	*3.25, 1.11*	*3.06, 1.20*	*3.26, 1.20*	*2.96, **0.76***	*5.17, 1.20*	*3.03, 1.01*	***2.18**, 0.84*



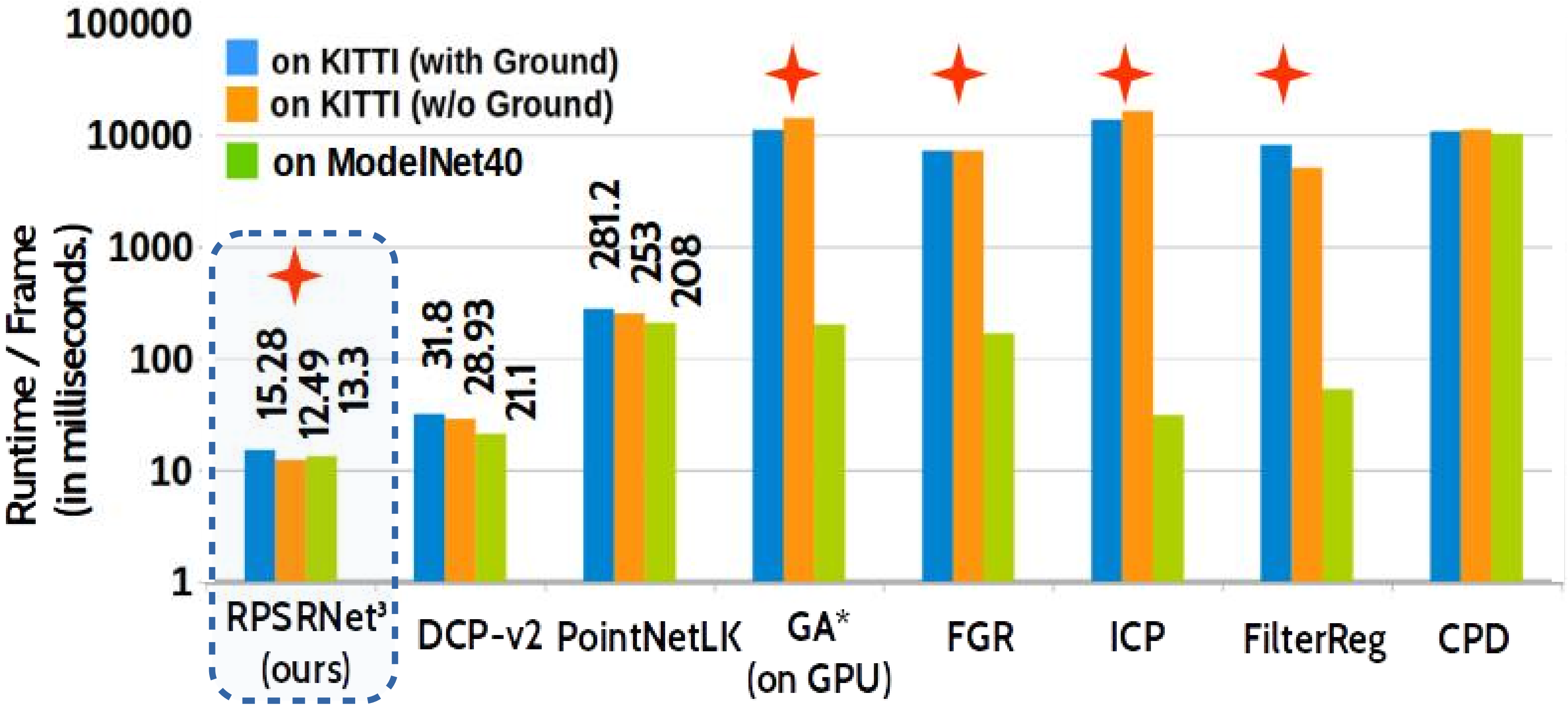


KITTI LiDAR Odometry (Without Pose Graph Optimization and Loop Closure)

								Single Iteration	Three Iterations
	CPD [44]	GA* [25]	FGR [77]	ICP [10]	FilterReg [21]	DCP-v2 [66]	PointNetLK [5]	RPSRNet[1] (ours)	RPSRNet[3] (ours)
Seq.	$\varphi_{\text{rmse}}, \Delta t_{\text{rmse}}$	$\varphi_{\text{rmse}}, \Delta t_{\text{rmse}}$	$\varphi_{\text{rmse}}, \Delta t_{\text{rmse}}$	$\varphi_{\text{rmse}}, \Delta t_{\text{rmse}}$	$\varphi_{\text{rmse}}, \Delta t_{\text{rmse}}$	$\varphi_{\text{rmse}}, \Delta t_{\text{rmse}}$	$\varphi_{\text{rmse}}, \Delta t_{\text{rmse}}$	$\varphi_{\text{rmse}}, \Delta t_{\text{rmse}}$	$\varphi_{\text{rmse}}, \Delta t_{\text{rmse}}$
mean	3.55, 1.08	3.30, 1.0	3.29, 0.85	3.15, 1.08	3.08, 0.77	2.92, 0.89	4.02, 1.12	3.13, 0.88	**2.22, 0.58**
	3.03, 1.07	*2.94, 1.02*	*3.25, 1.11*	*3.06, 1.20*	*3.26, 1.20*	*2.96, **0.76***	*5.17, 1.20*	*3.03, 1.01*	***2.18**, 0.84*



Runtime Evaluation On KITTI and ModelNet40 Datasets (*Clean*)



Using Original Point Size (i.e., without sub-sampling)

**BH-Tree Construction time ~4 ms

Thanks for Watching!