— Supplementary Material —

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

Sk Aziz Ali[1,2] Kerem Kahraman[1] Gerd Reis[2] Didier Stricker[1,2]
[1]TU Kaiserslautern [2]German Research Center for Artificial Intelligence (DFKI GmbH), Kaiserslautern

This supplementary document summarizes several relevant details for deeper understanding about our RPSRNet in three main Secs. 1, 2 and 3. The Sec. 1 lists down the parameter settings and training protocols of all benchmark methods alongside our proposed RPSRNet. We also show the effects of different batch size and scaled loss learning on the prediction accuracy of the network. The next Sec. 2 gives insights to our BH 2^D-tree convolution operation using linearized node-indexing. In the end, Sec. 3 provides more insightful results and evaluation of our method on ModelNet40 [12] and KITTI [5] datasets.

1. Evaluation Method and Training Details.

We train the registration networks of DCP [10], PRNet [11], and PointNetLK [1] methods all from scratch (w/o using their pre-trained models) following standard training protocols from the respective studies. Both DCP and its extended version PRNet, and PointNetLK take 250 epochs to train the network with a learning rate of 10^{-3} and using ADAM optimizer. During experiments, we find that PRNet has inherent issues of ill-conditioned feature embedding matrix computations which causes random crashes during training. Hence, we do not evaluate PRNet. While a batch size 10 is set for the DCP, PointNetLK is trained with batch size 32 and internal alignment iterations 10. The optimal hyper-parameters for our RPSRNet are – *required number of epochs*: 250, *batch size*: 16 (see the reason in Sec. 1.1), *learning rate:* 0.0001, *loss dissipation factor* (β): 0.2, *rotational/translational scale* ($\sigma_{\mathbf{R}}/\sigma_{\mathbf{t}}$): -5.0 / -2.5 for KITTI dataset and -9.0 / -10.0 for ModelNet40 dataset, and a *dropout*: 0.2. On the other hand, the maximum number of iterations for all the unsupervised iterative alignment methods ICP [3], CPD [7], FilterReg [4], FGR [13], and GA [6], is set to 200. The mean $\mu = 0.0$ and Gaussian spread $\sigma = 0.05$ are set for FilterReg and CPD (data-based automatic initialization of σ is also possible). FGR method uses Fast Point Feature Histogram (FPFH) [9, 8] descriptor of the input source and target point clouds and use them in their global optimization step to estimate rigid transformation. Hence, it takes a search radius to localize descriptor space for every point. We set the *search-radius* parameter to 0.15. To evaluate the GA, we set its astrophysical parameter settings as – *Gravitational constant* (G): $6.67 \cdot 10^{-3}$, *force softening length* (ϵ): 0.05, *time step* (Δt): 0.05, and *energy dissipation rate*: 0.3.

1.1. Importance of Batch-Size and Scaled Loss

On KITTI [5] and ModelNet40 [12] Datasets					
Batch Sizes	SLL $\mathbb{1}$	(K2-w) φ_{rmse}, $\mathbf{\Delta t}_{\text{rmse}}$	(K1-w/o) φ_{rmse}, $\mathbf{\Delta t}_{\text{rmse}}$	(M1-*seen*) φ_{rmse}, $\mathbf{\Delta t}_{\text{rmse}}$	(M2-*unseen*) φ_{rmse}, $\mathbf{\Delta t}_{\text{rmse}}$
8	✗	1.4394, 2.19	1.481, 1.53	5.82 , 0.04452	5.878, 0.0468
	✓	0.9747, 0.87	0.9234, 0.84	4.3111, 0.0221	4.667, 0.0281
16	✗	1.5823 , 2.24	1.4645, 1.39	6.1468 , 0.0472	6.2004, 0.0423
	✓	0.9889, **0.81**	0.9651, **0.7**	4.588, 0.02377	5.221, 0.0252
32	✗	1.5452 , 2.25	1.3855, 1.44	7.006 , 0.0536	8.613, 0.0542
	✓	**0.9407**, 1.03	**0.9025, 0.7**	**4.1318, 0.0189**	**4.203, 0.0195**

Table 1. Importance of different batch sizes and scaled loss learning (SLL) in RPSRNet: Our method using scaled loss learning (SLL) with 32 input samples in a batch reports minimum rotational and translation RMSE on both the KITTI and ModelNet40 datasets There is a notable exception on $\mathbf{\Delta t}_{\text{rmse}}$ while using the batch size 16 for KITTI (K2-w) setup.

We investigate thoroughly how different batch sizes, scaled version of our loss function (Sec.**3.3** of main matter) and pre-processing step of removing ground points impact on the final prediction accuracies. The evaluation Table 1 shows that with scaling, there are notable improvements on translation error for KITTI [5] dataset. The similar improvement on rotational error is clear for ModelNet40 [12] dataset. For the experiments in the main matter, we select batch size 16.

2. BH 2^{D}-tree Convolution and Node Indexing

This section describes the compact storage mechanism and input signal processing of our BH 2^{D}-tree representation of point cloud. To this end, we have already saved several lists of attributes of the tree nodes at every depth d:

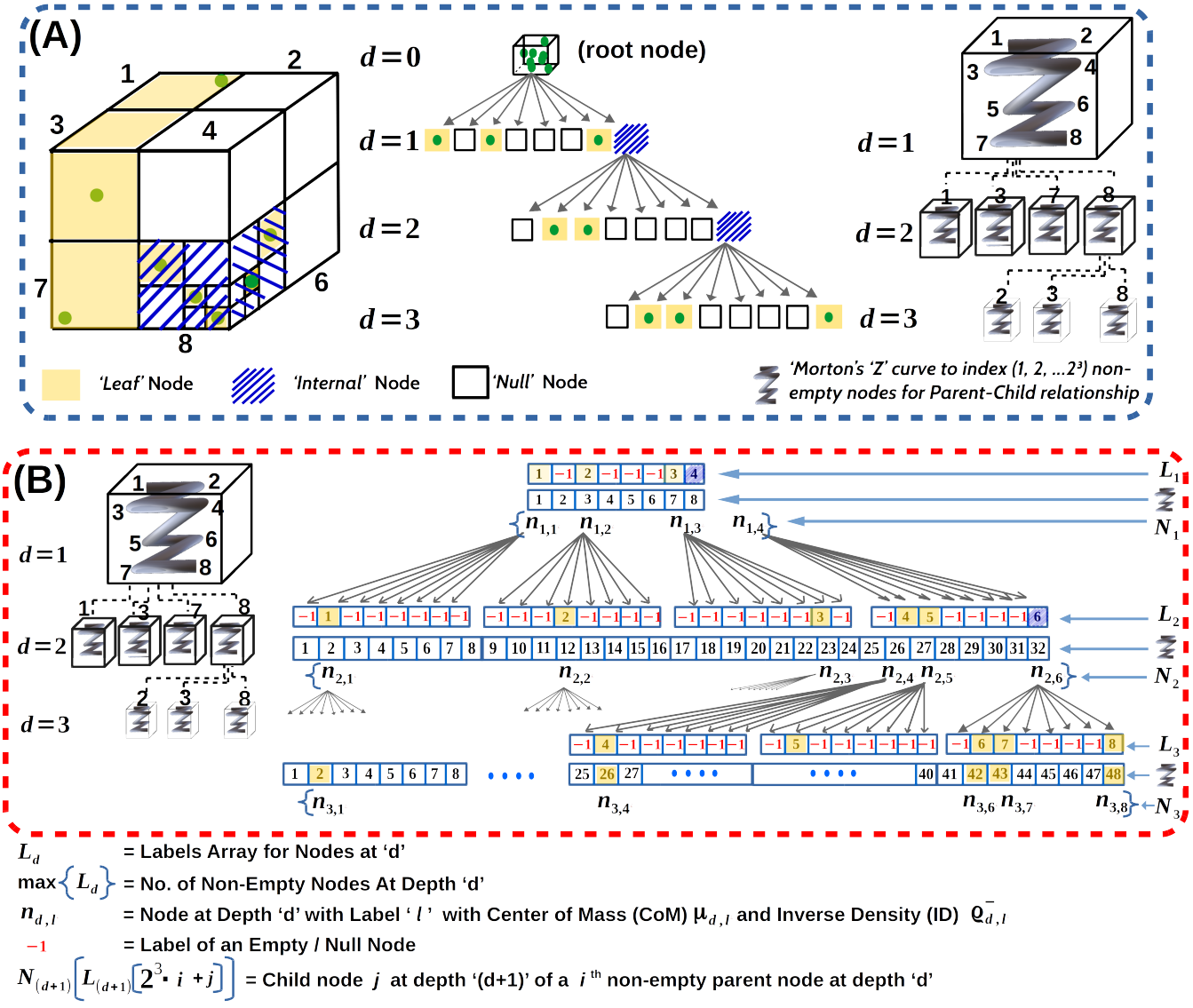

Figure 1. **An exemplary Barnes-Hut 2^D-Tree:** (A) Different types of BH-tree nodes and space-filling Morton's 'Z'-curve for tree traversal. (B) Overview of the labeling, indexing, neighborhood system through parent-child relationships.

(i) $\mathbf{M}_d$, $\mathbf{N}_d$, and ρ_d^- – the center of masses (CoMs) $\mathbf{M}_d$ and the inverse densities (IDs) ρ_d^- of the *non-empty nodes* $\mathbf{N}_d$ linearly store the respective values

(ii) $\mathbf{L}_d$ – *a label array* $\mathbf{L}_d$ stores the sorted indices of non-empty nodes in an increasing order. This array holds **-1** value to those positions where the nodes are empty. The Fig. 1 gives the complete overview of depth-wise labeling and indexing of the nodes.

(iii) κ_d – Finally, we have a set of 26^{i} neighbors $\kappa_d(n_{d,l})$ for every non-empty node $n_{d,l} \in \mathbf{N}_d$ at depth $d \in \{1, 2, \ldots\}$ and label $l \in \{1, 2, \ldots, (2^3)^d\}$. The Fig. 2 describes the adjacency system of a given node.

[i] or 27 neighbors including itself

2^{D}-tree Convolution. The first hierarchical embedding block (HFE) for positional features takes the CoMs array $\mathbf{M}_d$ which has x, y, and, z coordinates as 3 input channels. The other dimension is the number of non-empty nodes, *i.e.*, $|\mathbf{N}_d|$. Another separate HFE for density features takes the IDs as 1D scalar array. For this we first tile the input to match the same dimension of position channel, *i.e.*, $|\mathbf{N}_d| \times 3$. The integer array $\mathbf{L}_d$ has the storage size of $2^3 \cdot |\mathbf{N}_d|$. For 1D convolution on all $n_{d,l} \in \mathbf{N}_d$ the with kernel size 26 and stride 26, we first perform a *neighbor-fill* operation. To keep our input tensor of fixed size equal to $26 \cdot |\mathbf{N}_d|$, we fill the positions of the array with zero which are empty.

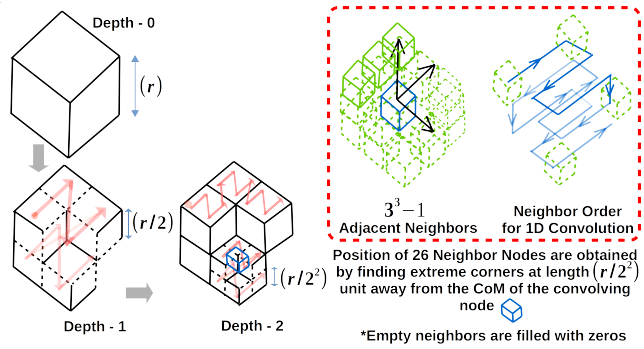

Figure 2. **Neighbors of a non-empty node at depth d.** 26 adjacent nodes $\kappa_d(n_{d,l})$ of $n_{d,l}$ (*always at the current depth of convolving node*) are the ones which are at the extreme end by $\frac{r}{2^d}$ length from $n_{d,l}$. The *zero-fill* operation sets zero values to the neighbors which are found empty. A linear array stores the neighbors contagiously following the order shown at *right*. Therefore, our 1D convolution kernel size is 26.

Hierarchical Pooling. Pooling is a converse operation of convolution where we fetch the information from child nodes to the parent nodes or in other words from nodes at the higher depth of our BH-tree to the lower depth, *e.g.*, from $d = 3$ 2. After a convolution, the parent-child information needs to be reconstructed for Max-Pooling. Lets observe the parent-child relationship using $\mathbf{L}_d$. The j^{th} child $C_j(P_i)$ of a non-empty parent node $P_i = n_{d,i} = \mathbf{N}_d[i]$ can be retrieved from the array $\mathbf{N}_{(d+1)}$, using the formula:

$$C_j(P_i) = N_{d+1}[L_{d+1}[i \cdot 8 + j]] \tag{1}$$

Since we store the labels at every depth and do not recompute the neighbors at runtime, we first perform a *child-fill* operation which is to fill attributes' values of the empty neighbor nodes by zero at the current depth. Therefore our input-tensor will have the size of $2^3 \cdot |\mathbf{N}_d|$ as of the label array L_d. For the *neighbor-fill* and *child-fill* operations, we write custom functions in CUDA/C++ for Pytorch-binding.

3. Insightful Results and Evaluations.

We present more insightful registration results for KITTI [5] LiDAR odometry and ModelNet40 [12] datasets (see Table. 2, Fig. 3, and Fig. 4 respectively). The quantitative results on KITTI dataset give important few remarks about other methods:

1. Only on seq-00 and seq-05, FGR and ICP wins the contest of lowest translation error. Although, their difference from RPSRNet is significantly small.
2. In the **(K2)** setup, all methods record higher transformation errors, especially on translational part, on most of the sequences.
3. On seq-01, GA [6], FGR [13], ICP [3] FilterReg [4], PointNetLK [1], and our RPSRNet[1] (single internal iteration) report $39\%, 119\%, 6\%, 171\%, 86\%$ and 6.90% increase in the translational error when evaluated on **(K2)** setup compared to the same on **(K1)** setup.

On the other hand, Fig. 3 and 4 show registration outcomes from competing methods on some randomly selected samples. The final alignments applying different methods show that most methods struggle to find globally optimal transformation when input data is largely incomplete.

References

[1] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. Pointnetlk: Robust and efficient point cloud registration using pointnet. In *CVPR*, 2019. 1, 3, 4

[2] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *International Conf. on Computer Vision (ICCV)*, 2019. 5

[3] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 14(2):239–256, 1992. 1, 3, 4

[4] Wei Gao and Russ Tedrake. Filterreg: Robust and efficient probabilistic point-set registration using gaussian filter and twist parameterization. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 3, 4

[5] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 1, 3, 4, 5

[6] Vladislav Golyanik, Sk Aziz Ali, and Didier Stricker. Gravitational approach for point set registration. *Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 3, 4

[7] A. Myronenko and X. Song. Point set registration: Coherent point drift. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(12):2262–2275, 2010. 1, 4

[8] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *International Conference on Robotics and Automation (ICRA)*, 2009. 1

[9] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *International Conference on Intelligent Robots and Systems (IROS)*, 2008. 1

[10] Yue Wang and Justin Solomon. Deep closest point: Learning representations for point cloud registration. In *International Conference on Computer Vision (ICCV)*, pages 3523–3532, 2019. 1, 4

[11] Yue Wang and Justin Solomon. Prnet: Self-supervised learning for partial-to-partial registration. In *Advances in Neural Information Processing Systems (NIPS)*, pages 8812–8824, 2019. 1

[12] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 3

[13] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *European Conference on Computer Vision (ECCV)*, 2016. 1, 3, 4

Seq.	CPD [7] φ_{rmse}, $\Delta\mathbf{t}_{\text{rmse}}$	GA* [6] φ_{rmse}, $\Delta\mathbf{t}_{\text{rmse}}$	FGR [13] φ_{rmse}, $\Delta\mathbf{t}_{\text{rmse}}$	ICP [3] φ_{rmse}, $\Delta\mathbf{t}_{\text{rmse}}$	FilterReg [4] φ_{rmse}, $\Delta\mathbf{t}_{\text{rmse}}$	DCP-v2 [10] φ_{rmse}, $\Delta\mathbf{t}_{\text{rmse}}$	PointNetLK [1] φ_{rmse}, $\Delta\mathbf{t}_{\text{rmse}}$	RPSRNet[1] (ours) φ_{rmse}, $\Delta\mathbf{t}_{\text{rmse}}$	RPSRNet[3] (ours) φ_{rmse}, $\Delta\mathbf{t}_{\text{rmse}}$
	On KITTI [5] Dataset								
00	4.99, 1.12	4.82, 1.09	4.77, 0.82	4.72, **0.80**	4.77, 0.80	4.87, 1.07	5.44, 1.27	4.48, 1.01	**3.30**, 0.81
	4.91, 1.39	*4.78, 0.93*	*4.78, 0.83*	*4.70, 0.84*	*4.92, 0.88*	*4.78, 0.80*	*6.11, 1.27*	*4.56, **0.73***	***3.31**, 1.0*
01	3.18, 1.54	2.99, 1.74	3.06, 1.10	3.06, 2.27	3.02, 0.89	3.0, 0.95	4.50, 1.33	2.79, 1.28	**2.13, 0.48**
	3.15, 1.88	*3.06, 2.42*	*2.88, 2.42*	*2.83, 2.41*	*2.77, 2.41*	***1.80**, **0.68***	*5.22, 1.18*	*3.04, 1.38*	*2.0, 1.03*
02	3.87, 1.28	3.67, 0.98	3.71, 1.02	3.63, 1.0	3.42, 0.69	3.52, 0.76	4.21, 1.0	3.69, 0.73	**2.66, 0.6**
	3.0, 1.11	*3.71, 1.09*	*3.66, 0.9*	*3.66, 0.91*	*3.81, 1.1*	*3.52, **0.53***	*5.77, 0.96*	*3.73, 1.10*	***2.88**, 1.11*
03	0.38, 0.88	0.34, 0.72	0.31, 0.59	0.14, 0.56	0.14, 0.49	0.24, 0.66	0.90, 0.81	0.14, 0.72	**0.10, 0.41**
	0.23, 0.58	*0.18, 0.43*	*0.20, 0.74*	*0.13, 0.91*	*0.35, 0.78*	*0.18, **0.3395***	*2.1, 1.01*	*0.16, 0.85*	***0.08**, 0.68*
04	2.58, 1.09	2.64, 1.12	2.65, 1.06	2.64, 1.07	1.97, 0.89	2.20, 1.37	3.88, 1.31	2.74, 0.55	**1.11, 0.50**
	2.77, 0.91	*2.64, 0.85*	*2.64, 1.09*	*2.65, 1.11*	*2.11, 1.18*	*2.07, 1.01*	*4.86, 1.27*	*2.28, 0.38*	***1.19**, **0.21***
05	3.81, 0.79	3.41, 0.7135	3.29, **0.4142**	2.95, 0.75	3.16, 0.62	2.01, 0.42	4.09, 0.79	3.09, 0.87	**1.91, 0.71**
	3.0, 0.69	*3.30, 0.64*	*3.27, 0.80*	*2.8, 1.12*	*3.89, 0.99*	*3.15, **0.63***	*4.2, 1.12*	*3.35, 1.06*	***1.11**, 0.91*
06	4.67, 0.96	4.13, 0.85	4.04, 0.87	3.64, 1.20	4.04, 0.88	3.02, 0.69	3.08, 0.99	4.01, 0.64	**3.0, 0.50**
	2.85, 1.28	***1.55**, 1.02*	*4.13, 1.21*	*3.26, 1.32*	*4.03, 1.29*	*3.74, **0.48***	*4.87, 0.97*	*2.64, 0.48*	*3.94, 0.69*
07	4.89, 1.0	4.39, 0.78	4.46, 0.91	4.41, 1.03	4.11, 0.99	4.48, 1.22	6.04, 1.44	4.06, 0.81	**3.58, 0.61**
	*4.31, **0.71***	*4.34, 0.77*	*4.46, 0.88*	*4.37, 0.95*	*4.22, 0.99*	*4.45, 1.58*	*8.21, 1.81*	*4.45, 1.08*	***3.11**, 1.07*

Table 2. Evaluation on KITTI [5] LiDAR sequences 00 to 07. The first row under each sequence reports the RMSE on angular and translational deviations from ground-truth measured on its validation samples in **(K1-w/o)** setup: without 'ground points'. The *next* row under the same sequence reports the same error (with gray-colored and *italic* font-style for better distinction) on the same validation set in **(K2-w)** setup: with 'ground points'. The lowest transformation errors achieved by a method is highlighted in **bold** or underlined ***bold*** font.

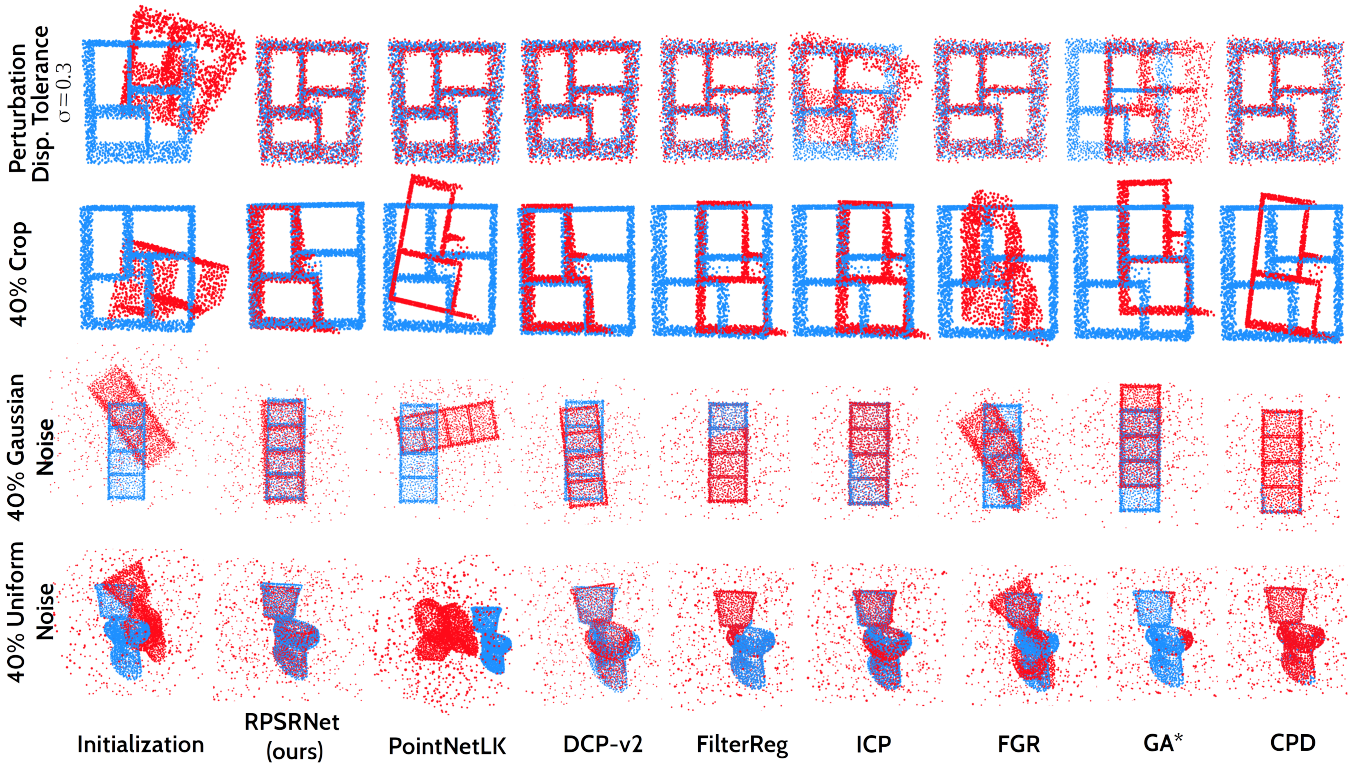

Figure 3. Registration outcomes on few samples of ModelNet40 dataset. The results show robustness of RPSRNet compared to state-of-the-art methods.

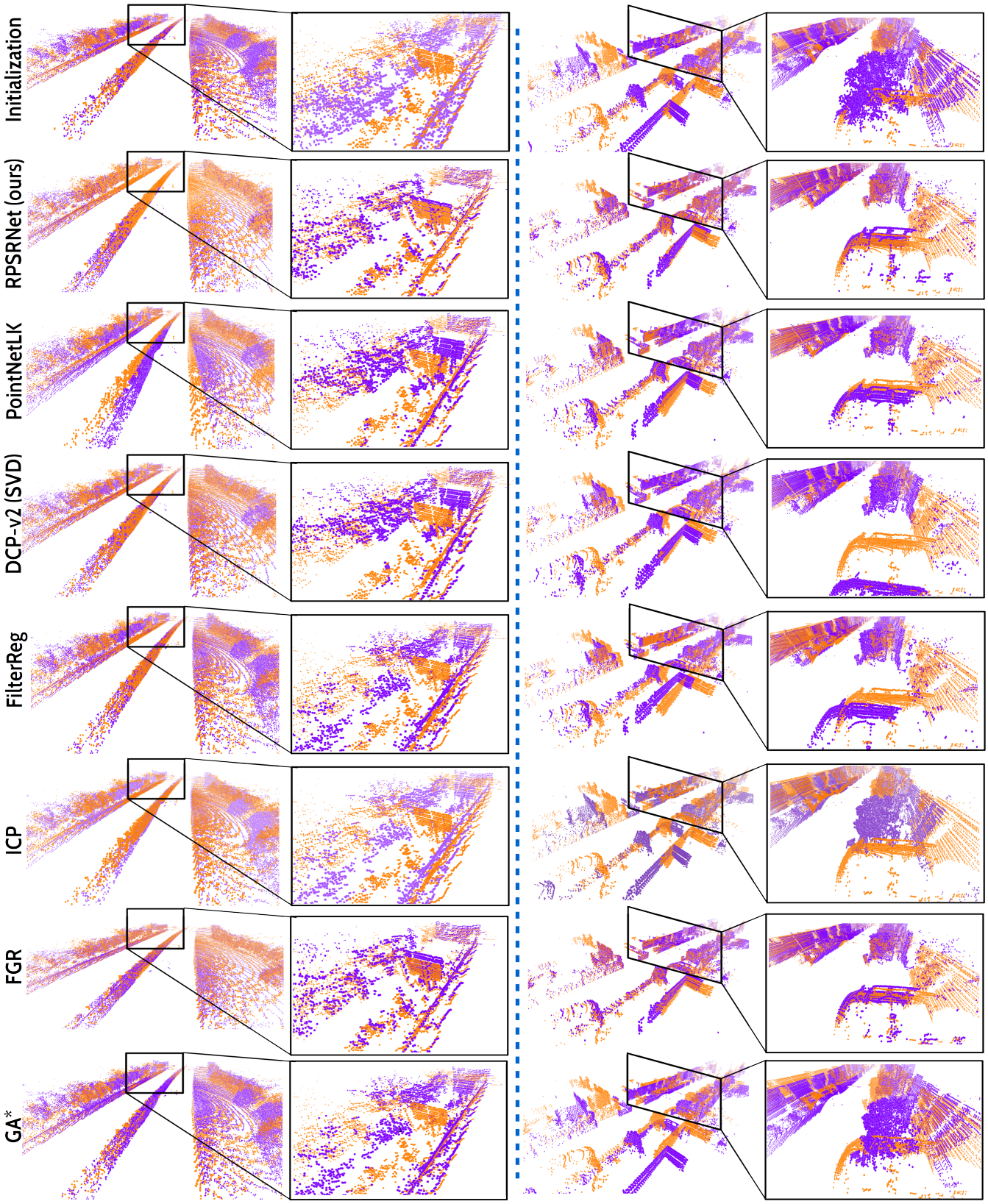

Figure 4. Results of our RPSRNet[3] on some randomly selected samples from the validation set of KITTI [5, 2] dataset (*left column*: frame 000131 (as $\mathbf{Y}$) and 000136 (as $\mathbf{X}$) of seq-01 and *right column*: frame 001721 (as $\mathbf{Y}$) and 001726 (as $\mathbf{X}$) from the seq-05). Zoomed parts of each image highlights how other competing methods perform in aligning static objects of target scene.