OMNet: Learning Overlapping Mask for Partial-to-Partial Point Cloud Registration Supplementary Material

Hao Xu[1,2] Shuaicheng Liu[1,2*] Guangfu Wang[2] Guanghui Liu[1*] Bing Zeng[1]

[1]University of Electronic Science and Technology of China
[2]Megvii Technology

1. Overview

This supplementary material provides more details on experiments in the main paper and includes more experiments to validate and analyze our proposed method.

In Sec. 2, we describe details in two data generation manners for point cloud registration, which are proposed by PRNet [7] and RPMNet [10] separately. In Sec. 3, we show more experimental results including the performance on the validation and test sets, which are generated by the above two pre-processing manners.

2. Details in Experiments

In this section, we describe two data preparation manners for the partial-to-partial point cloud registration. One of the manners proposed by PRNet [7] is detailed in Sec. 2.1, while the other used in RPMNet [10] is described in Sec. 2.2. Fig. 1 illustrates some examples of the partial-to-partial data, which shows the difference between these two pre-processing manners.

2.1. Data Generation Manner of PRNet

We use this manner to generate data for the first three experiments on the ModelNet40 dataset in our main paper. Two point clouds are randomly chosen from 40 sampled point clouds as the source point cloud $\mathbf{X}$ and reference point cloud $\mathbf{Y}$ respectively, each of which contains 2,048 points. Along each axis, we randomly draw a rigid transformation: the rotation along each axis is sampled in $[0°, 45°]$ and the translation is in $[-0.5, 0.5]$. The rigid transformation is applied to $\mathbf{Y}$, leading to $\mathbf{X}$. After that, we simultaneously partial $\mathbf{X}$ and $\mathbf{Y}$ by randomly placing a point in space and computing its 768 nearest neighbors in $\mathbf{X}$ and $\mathbf{Y}$ respectively. The left column in Fig. 1 shows some examples. However, the point clouds $\mathbf{X}$ and $\mathbf{Y}$ are similar in most cases, which means that the overlapping ratio is large.

*Corresponding author

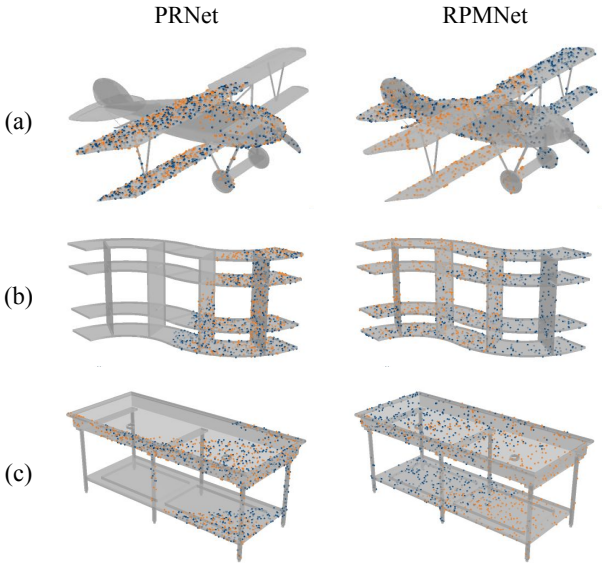


Figure 1. Some examples of the partial-to-partial data. The left and right columns denote the point clouds that are processed in the manner of PRNet and RPMNet respectively. All of them are registered by the ground truth transformations. Blue denotes the source point cloud $\mathbf{X}$ and red denotes the reference point cloud $\mathbf{Y}$. The pairs of RPMNet are more decentralized than those of PRNet.

2.2. Data Generation Manner of RPMNet

We use this manner to generate data for the last experiment on the ModelNet40 dataset in our main paper. We use the same approach as which in RPMNet to obtain the source point cloud $\mathbf{X}$ and the reference point cloud $\mathbf{Y}$. For each point cloud, we sample a half-space with a random direction and shift it such that approximately 70% of the points are retained. Then, the point clouds are downsampled to 717 points to maintain a similar point density as the previous experiments. We sample rotations by sampling three Euler angle rotations in the range $[0°, 45°]$ and translations in the range $[-0.5, 0.5]$ on each axis. The rigid transformation is applied to $\mathbf{X}$, leading to $\mathbf{Y}$, which is opposite to

Method	RMSE(**R**)		MAE(**R**)		Error(**R**)	
	OS	*TS*	*OS*	*TS*	*OS*	*TS*
RPMNet [10]	0.312	6.531	0.200	2.972	0.432	8.454
PRNet [7]	5.979	13.773	3.779	9.670	7.714	20.692
IDAM [4]	1.605	7.725	0.905	4.364	1.850	10.940
Ours	0.766	6.258	0.347	2.877	0.873	8.606

Table 1. Results on 8 axisymmetrical categories in ModelNet40. *OS* and *TS* denotes the results on the once-sampled and our twice-sampled data separately. The performances of all methods are decreasing when changing the data sampling manner from *OS* to *TS*.

PRNet. The right column in Fig. 1 shows some examples. The points in $\mathbf{X}$ and $\mathbf{Y}$ are more decentralized than those generated in the manner of PRNet, which means that the overlapping ratio is small in some cases. As a result, it is more difficult to register with this data.

3. More Experiments on ModelNet40

In this section, we provide more experimental results on ModelNet40 [8] dataset to validate the robustness and effectiveness of our method. First, we show the decrement of performances between evaluating on the once-sampled (*OS*) and the twice-sampled (*TS*) data in Sec. 3.1, which demonstrates the over-fitting issue. Then, we show the results on the test set that generated in the manner of PRNet in Sec. 3.2, and the results on the dataset that generated in the manner of RPMNet are shown in Sec. 3.3. Besides, the comparison of speed shows the computational efficiency of our method in Sec. 3.4. Finally, Sec. 3.5 explores that how many iterations are needed.

3.1. Over-fitting Issue

In this experiment, we demonstrate that deep learning (DL) based methods can easily over-fit the original data distribution even with added noises, as shown in Table 1. Note that we train and evaluate on 8 axisymmetrical categories, where point clouds are only rotated at the z-axis. We can find that all partial-to-partial methods can achieve good performances on the *OS* data, however, performances are obviously decreasing when only changing the data sampling manner from *OS* to *TS*. As a result, all the methods can over-fit the distribution of *OS* data, while failing to register the axisymmetrical *TS* data.

3.2. Results on PRNet dataset

In our main paper, we only show the results on the validation set with Gaussian noise generated in the partial manner of PRNet [7]. To further demonstrate the robustness of our method, we show results on the unseen categories with Gaussian noise. We add noises sampled from $\mathcal{N}(0, 0.01^2)$ and clipped to $[-0.05, 0.05]$ on each axis, then repeat the second experiment (unseen categories) in our main paper.

# points	ICP [2]	FGR [12]	PointNetLK [1]	DCP [6]	PRNet [7]	FMR [3]	RPMNet [10]	IDAM [4]	DeepGMR [11]	Ours
512	33	37	73	15	79	138	58	27	9	24
1024	56	92	77	17	84	158	115	28	9	25
2048	107	237	83	26	114	295	271	33	9	27
4096	271	673	89	88	-	764	726	62	10	32

Table 2. Speed comparison for registering a point cloud pair of various sizes (in milliseconds). The missing result in the table is due to the limitation in GPU memory.

Table 3 shows the performances of various methods. Our method achieves accurate registration and ranks first.

3.3. Results on RPMNet dataset

In this subsection, we show 4 experimental results on the ModelNet40 dataset with or without Gaussian noise that generated in the data partial manner of RPMNet [10].

Unseen Shapes. In this experiment, we train the models on the training set of the first 14 categories and evaluate the registration performances on the validation set of the same categories without noise. Table 4(a) shows the results.

We can find that all traditional methods and most DL based methods perform poorly because of the large difference in initial positions and partiality. Note that the normals are calculated after the data pre-processing, so that the normals of points in $\mathbf{X}$ can be different from their correspondences in $\mathbf{Y}$. Although Go-ICP [9] attempts to improve the performance of the original ICP [2] by adopting a brute-force branch-and-bound strategy, it may not suitable for this scene and brings negative implications. Our method outperforms all the traditional and DL based methods except 3 metrics on the *OS* data compared with RPMNet.

Unseen Categories. We evaluate the performance on unseen categories without noise in this experiment. The models are trained on the first 14 categories and tested on the other 18 categories. The results are summarized in Table 4(b). We can find that the performances of all DL based methods become worse without training on the same categories. Nevertheless, the traditional methods are not affected so much due to the handcrafted features. Our method outperforms all the traditional and DL based methods.

Gaussian Noise. To evaluate the capability of robustness to noise, we add noises sampled from $\mathcal{N}(0, 0.01^2)$ on each axis and clipped to $[-0.05, 0.05]$, then repeat the first two experiments (unseen shapes and unseen categories). Table 4(c)(d) show the performances of different algorithms. FGR [12] is sensitive to noise, so that it performs much worse than the noise-free case. Almost all the DL based methods become worse with noise injected. Our method achieves the best performance compared to all competitors.

3.4. Efficiency

We profile the inference times in Table 2. We test DL based models on a NVIDIA RTX 2080Ti GPU and two 2.10

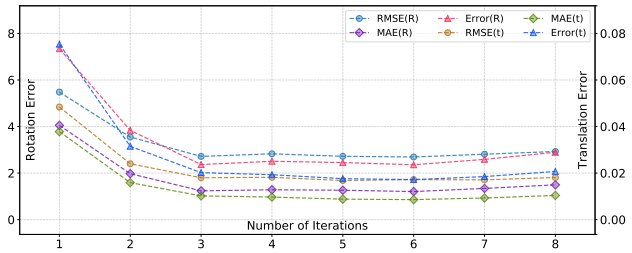


Figure 2. Anisotropic and isotropic errors of our method over registration iterations.

GHz Intel Xeon Gold 6130 CPUs for the other methods. For our approach, we provide the time of $N = 4$ iterations. The computational time is averaged over the entire test set. The speeds of traditional methods are variant under different settings. Note that ICP is accelerated using the k-D tree. We do not compare with Go-ICP because its obviously slow speed. Our method is faster especially with large inputs but is slower than the non-iterative DCP and DeepGMR.

3.5. Number of Iterations

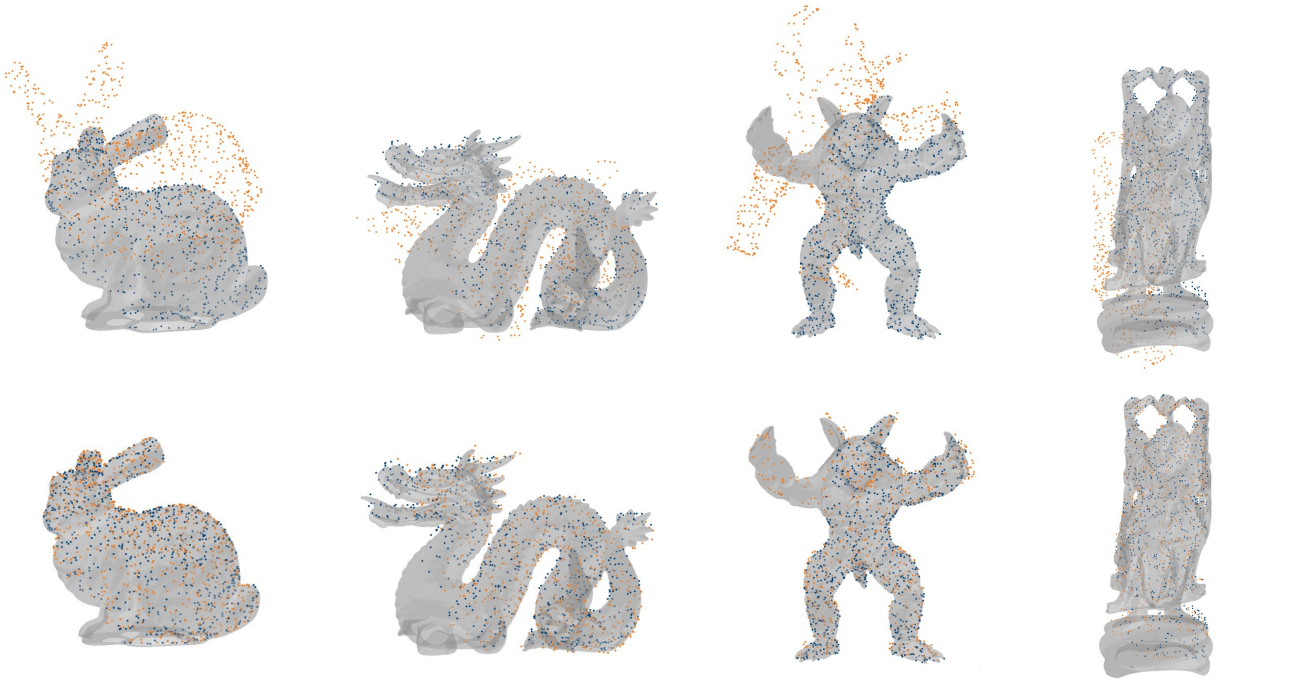
The model is trained with different iterations to show that how many iterations are needed. The anisotropic and isotropic errors are calculated after each iteration, as illustrated in Fig. 2. We can find that most performance gains are in the first two iterations, and we choose $N = 4$ for the trade-off between the speed and the accuracy in all experiments.

Method	RMSE(**R**)		MAE(**R**)		RMSE(**t**)		MAE(**t**)		Error(**R**)		Error(**t**)	
	OS	*TS*	*OS*	*TS*	*OS*	*TS*	*OS*	*TS*	*OS*	*TS*	*OS*	*TS*
ICP [2]	17.439	18.588	8.954	9.628	0.0848	0.0920	0.0460	0.0521	17.435	18.720	0.0905	0.1026
Go-ICP [9]	13.081	15.214	3.617	4.650	0.0455	0.0566	0.0169	0.0223	7.184	9.002	0.0334	0.0445
Symmetric ICP [5]	6.447	7.096	5.790	6.280	0.0615	0.0688	0.0552	0.0617	11.340	12.531	0.1065	0.1191
FGR [12]	19.027	33.723	8.383	19.268	0.1041	0.1593	0.0498	0.0914	15.902	35.971	0.0981	0.1828
PointNetLK [1]	27.589	29.747	16.047	18.550	0.1516	0.1841	0.0955	0.1081	30.406	32.760	0.1907	0.1959
DCP [6]	7.353	7.300	4.923	4.378	0.0657	0.0389	0.0451	0.0272	9.624	8.853	0.0902	0.0539
PRNet [7]	3.241	5.883	1.632	3.037	0.0181	0.0380	0.0127	0.0237	3.095	5.974	0.0254	0.0472
FMR [3]	4.819	5.304	2.488	2.779	0.0345	0.0323	0.0158	0.0172	4.824	5.392	0.0313	0.0342
IDAM [4]	5.188	8.008	3.114	4.559	0.0377	0.0484	0.0208	0.0291	5.836	8.774	0.0418	0.0578
Ours	2.203	2.563	0.910	1.215	0.0155	0.0183	0.0078	0.0098	1.809	2.360	0.0156	0.0196

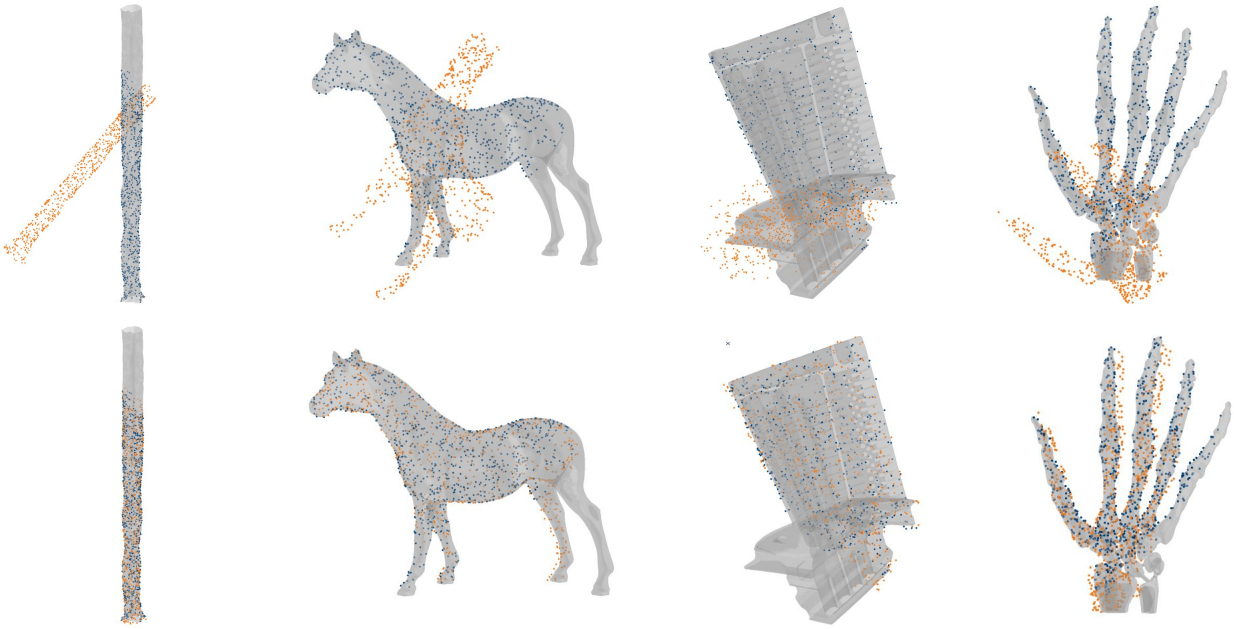
Table 3. Results on point clouds of unseen categories with Gaussian noise in ModelNet40, which are generated in the manner of PRNet. Red indicates the best performance and blue indicates the second-best result.

	Method	RMSE(**R**)		MAE(**R**)		RMSE(**t**)		MAE(**t**)		Error(**R**)		Error(**t**)	
		OS	*TS*	*OS*	*TS*	*OS*	*TS*	*OS*	*TS*	*OS*	*TS*	*OS*	*TS*
(a) Unseen Shapes	ICP [2]	20.036	22.840	10.912	12.147	0.1893	0.1931	0.1191	0.1217	22.232	24.654	0.2597	0.2612
	Go-ICP [9]	70.776	71.077	39.000	38.266	0.3111	0.3446	0.1807	0.1936	71.597	76.492	0.3996	0.4324
	Symmetric ICP [5]	10.419	11.295	8.992	9.592	0.1367	0.1394	0.1082	0.1124	17.954	19.571	0.2367	0.2414
	FGR [12]	48.533	46.766	29.661	29.635	0.2920	0.3041	0.1965	0.2078	55.855	57.685	0.4068	0.4263
	PointNetLK [1]	23.866	27.482	15.070	18.627	0.2368	0.2532	0.1623	0.1778	29.374	36.947	0.3454	0.3691
	DCP [6]	12.217	11.109	9.054	8.454	0.0695	0.0851	0.0524	0.0599	7.835	9.216	0.1049	0.1259
	RPMNet [10]	1.347	2.162	0.759	1.135	0.0228	0.0267	0.0089	0.0141	1.446	2.280	0.0193	0.0302
	FMR [3]	7.642	8.033	4.823	4.999	0.1208	0.1187	0.0723	0.0726	9.210	9.741	0.1634	0.1617
	DeepGMR [11]	72.310	70.886	49.769	47.853	0.3443	0.3703	0.2462	0.2582	82.652	86.444	0.5044	0.5354
	Ours	0.771	1.384	0.277	0.542	0.0154	0.0226	0.0056	0.0093	0.561	1.118	0.0122	0.0198
(b) Unseen Categories	ICP [2]	20.387	22.906	12.651	13.599	0.1887	0.1994	0.1241	0.1286	25.085	26.819	0.2626	0.2700
	Go-ICP [9]	69.747	64.455	39.646	34.017	0.3035	0.3196	0.1788	0.1888	68.329	68.920	0.3893	0.4091
	Symmetric ICP [5]	12.291	12.333	10.841	10.746	0.1488	0.1456	0.1212	0.1186	21.399	21.437	0.2577	0.2521
	FGR [12]	46.161	41.644	27.475	26.193	0.2763	0.2872	0.1818	0.1951	49.749	51.463	0.3745	0.4003
	PointNetLK [1]	27.903	42.777	18.661	28.969	0.2525	0.3210	0.1752	0.2258	36.741	53.307	0.3671	0.4613
	DCP [6]	13.387	12.507	9.971	9.414	0.0762	0.1020	0.0570	0.0730	11.128	12.102	0.1143	0.1493
	RPMNet [10]	3.934	7.491	1.385	2.403	0.0441	0.0575	0.0150	0.0258	2.606	4.635	0.0318	0.0556
	FMR [3]	10.365	11.548	6.465	7.109	0.1301	0.1330	0.0816	0.0837	12.159	13.827	0.1773	0.1817
	DeepGMR [11]	75.773	68.425	53.689	46.269	0.3485	0.3667	0.2481	0.2595	85.210	87.192	0.5074	0.5323
	Ours	3.719	4.014	1.314	1.619	0.0392	0.0406	0.0151	0.0179	2.657	3.206	0.0321	0.0383
(c) Unseen Shapes with Gaussian Noise	ICP [2]	20.245	23.174	11.134	12.405	0.1902	0.1932	0.1214	0.1231	22.580	25.147	0.2634	0.2639
	Go-ICP [9]	72.221	72.030	40.516	39.308	0.3162	0.3468	0.1860	0.1977	74.420	77.519	0.4089	0.4405
	Symmetric ICP [5]	11.087	11.731	9.671	10.042	0.1453	0.1436	0.1157	0.1163	19.174	20.292	0.2517	0.2486
	FGR [12]	53.186	47.816	33.189	30.572	0.3059	0.3149	0.2117	0.2185	63.019	59.759	0.4368	0.4459
	PointNetLK [1]	24.162	26.235	16.222	17.874	0.2369	0.2582	0.1684	0.1805	32.108	36.109	0.3555	0.3771
	DCP [6]	12.387	12.393	9.147	9.534	0.0656	0.1008	0.0495	0.0717	8.341	8.955	0.0989	0.1516
	RPMNet [10]	1.670	2.955	0.889	1.374	0.0310	0.0360	0.0111	0.0163	1.692	2.746	0.0242	0.0353
	FMR [3]	8.026	8.591	5.051	5.303	0.1244	0.1249	0.0755	0.0776	9.657	10.383	0.1702	0.1719
	DeepGMR [11]	74.958	70.810	52.119	47.954	0.3520	0.3689	0.2538	0.2597	86.935	87.444	0.5189	0.5360
	Ours	0.998	1.522	0.555	0.817	0.0172	0.0189	0.0078	0.0098	1.078	1.622	0.0167	0.0208
(d) Unseen Categories with Gaussian Noise	ICP [2]	20.566	21.893	12.786	13.402	0.1917	0.1963	0.1265	0.1278	25.417	26.632	0.2667	0.2679
	Go-ICP [9]	70.417	65.402	40.303	34.988	0.3072	0.3233	0.1822	0.1929	69.175	71.054	0.3962	0.4170
	Symmetric ICP [5]	12.183	12.576	10.723	10.987	0.1487	0.1478	0.1210	0.1203	21.169	21.807	0.2576	0.2560
	FGR [12]	49.133	46.213	31.347	30.116	0.3002	0.3034	0.2068	0.2141	56.652	58.968	0.4230	0.4364
	PointNetLK [1]	26.476	29.733	19.258	21.154	0.2542	0.2670	0.1853	0.1937	37.688	42.027	0.3831	0.3964
	DCP [6]	13.117	12.730	9.741	9.556	0.0779	0.1072	0.0591	0.0774	11.350	12.173	0.1187	0.1586
	RPMNet [10]	4.118	6.160	1.589	2.467	0.0467	0.0618	0.0175	0.0274	2.983	4.913	0.0378	0.0589
	FMR [3]	10.604	11.674	6.725	7.400	0.1300	0.1364	0.0827	0.0867	12.627	14.121	0.1788	0.1870
	DeepGMR [11]	75.257	68.560	53.470	46.579	0.3509	0.3735	0.2519	0.2654	84.121	87.104	0.5180	0.5455
	Ours	3.572	4.356	1.570	1.924	0.0391	0.0486	0.0172	0.0223	3.073	3.834	0.0359	0.0476

Table 4. Results on point clouds of unseen shapes in ModelNet40, which are generated in the manner of RPMNet.



Error(**R**)=1.758, Error(**t**)=0.0140 | Error(**R**)=2.839, Error(**t**)=0.0169 | Error(**R**)=2.831, Error(**t**)=0.0222 | Error(**R**)=1.819, Error(**t**)=0.0166



Error(**R**)=4.941, Error(**t**)=0.0148 | Error(**R**)=2.323, Error(**t**)=0.0151 | Error(**R**)=1.425, Error(**t**)=0.0143 | Error(**R**)=4.090, Error(**t**)=0.0226

Figure 3. Results on the Stanford 3D Scan dataset. The model is trained on the training set on ModelNet40, and no fine-tuning is done on Stanford 3D Scan. In each cell separated by the horizontal line, the top row shows the initial positions of the two point clouds, and the bottom row shows the results of registration. Anisotropic and isotropic errors of each result is shown bellow the point clouds.

References

[1] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. PointNetLK: Robust & efficient point cloud registration using pointnet. In *Proc. CVPR*, pages 7163–7172, 2019. 2, 4

[2] Paul J. Besl and Neil D. McKay. A method for registration of 3d shapes. volume 14, pages 239–256, 1992. 2, 4

[3] Xiaoshui Huang, Guofeng Mei, and Jian Zhang. Feature-metric registration: A fast semi-supervised approach for robust point cloud registration without correspondences. In *Proc. CVPR*, pages 11366–11374, 2020. 2, 4

[4] Jiahao Li, Changhao Zhang, Ziyao Xu, Hangning Zhou, and Chi Zhang. Iterative distance-aware similarity matrix convolution with mutual-supervised point elimination for efficient point cloud registration. In *Proc. ECCV*, pages 378–394, 2020. 2, 4

[5] Szymon Rusinkiewicz. A symmetric objective function for icp. *ACM Trans. Graphics*, 38(4):1–7, 2019. 4

[6] Yue Wang and Justin M. Solomon. Deep closest point: Learning representations for point cloud registration. In *Proc. ICCV*, pages 3523–3532, 2019. 2, 4

[7] Yue Wang and Justin M. Solomon. Prnet: Self-supervised learning for partial-to-partial registration. In *Proc. NeurIPS*, pages 8814–8826, 2019. 1, 2, 4

[8] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d ShapeNets: A deep representation for volumetric shapes. In *Proc. CVPR*, pages 1912–1920, 2015. 2

[9] Jiaolong Yang, Hongdong Li, and Yunde Jia. Go-icp: Solving 3d registration efficiently and globally optimally. In *Proc. CVPR*, pages 1457–1464, 2013. 2, 4

[10] Zi Jian Yew and Gim Hee Lee. Rpm-net: Robust point matching using learned features. In *Proc. CVPR*, pages 11824–11833, 2020. 1, 2, 4

[11] Wentao Yuan, Benjamin Eckart, Kihwan Kim, Varun Jampani, Dieter Fox, and Jan Kautz. Deepgmr: Learning latent gaussian mixture models for registration. In *Proc. ECCV*, pages 733–750, 2020. 2, 4

[12] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *Proc. ECCV*, pages 766–782, 2016. 2, 4