Supplementary Material

Details of the Model

Both DCP-v1 and DCP-v2 use DGCNN to embed point clouds. In DGCNN, we use five EdgeConv layers. The numbers of filters per layer are $[64, 64, 128, 256, 512]$. In each EdgeConv layer, BatchNorm is used [2] with 0.1 momentum, followed by ReLU [4]. Following [9], we concatenate the outputs from the first four layers and feed them into the last one. Our local aggregation function is $\max$, and no global aggregation function is needed.

For DCP-v2, in the Transformer layer, the architecture is the same as the one proposed in [8]. The only difference is that we do not add positional encoding, because the position of each point in $\mathbb{R}^3$ is not correlated with its index in the array.

We use one encoder and one decoder. In both the encoder and decoder, we use multi-head attention with 4 heads and 512 embedding dimensions, followed by a MLP with 1024 hidden dimensions. ReLU [4] is also used in the MLP. Inside the Transformer, LayerNorm [1] is used after MLP and multi-head attention and before the residual connection. Unlike [8], we do not use Dropout [7].

We use PyTorch's [5] built-in SVD layer. Other numerical solvers that support gradient backpropagation could also be used to recover the rigid transformation.

Adam [3] with initial learning rate 0.001 is used for training. The coefficients used for computing running averages of the gradient and its square are 0.9 and 0.999, resp.

We use weight decay 10^{-4} to regularize the model. We train the model a total of 250 epochs, and at epochs 75, 150 and 200, the learning rate is divided by 10.

The MLP we use in ablation study has 3 fully connected layers and the number of filters are $[256, 128, 64]$ respectively. BatchNorm [3] and ReLU [4] are used after each fully connected layer. Finally, another two fully connected layers are used to project the embeddings to quaternion and translation vector separately.

The architecture of PointNet in ablation study is the same as the basic version in [6]. The number of filters in each layer are $[64, 64, 64, 128, 512]$.

Additional Figures

We provide additional figures of results with DCP-v2 tested on different objects in Figure 1. Figure 2 shows results in which we test with rotations in all of $\mathrm{SE}(3)$, meaning along each axis, we randomly sample rotations in $[0, 360^\circ]$. The model used here is still trained with rotations in $[0, 45^\circ]$. As shown in Figure 2, our model generalizes reasonably well to large motions. Figure 3 shows the results of DCP-v2 tested on noisy point clouds. The noise (sampled from $\mathcal{N}(0, 0.01)$) is added independently to each point of two input point clouds.

Model	MSE($\boldsymbol{R}$)	RMSE($\boldsymbol{R}$)	MAE($\boldsymbol{R}$)	MSE($\boldsymbol{t}$)	RMSE($\boldsymbol{t}$)	MAE($\boldsymbol{t}$)
ICP	2845.295	53.341	45.415	0.084	0.289	0.248
Go-ICP	3778.314	61.468	35.032	0.004	0.066	0.033
FGR	560.506	23.675	8.118	0.001	0.029	0.007
PointNetLK	4826.940	69.476	44.307	0.006	0.080	0.047
DCP-v2 (ours)	**225.736**	**15.025**	**2.260**	**0.000002**	**0.001**	**0.001**

Table 1. ModelNet40: Test on unseen point clouds ($[0, 90^\circ]$)

Model	MSE($\boldsymbol{R}$)	RMSE($\boldsymbol{R}$)	MAE($\boldsymbol{R}$)	MSE($\boldsymbol{t}$)	RMSE($\boldsymbol{t}$)	MAE($\boldsymbol{t}$)
ICP	11175.784	105.716	90.713	0.085	0.292	0.250
Go-ICP	13169.908	114.760	92.941	0.013	0.115	0.080
FGR	13808.102	117.508	87.678	0.008	0.090	0.047
PointNetLK	13751.237	117.266	94.539	0.011	0.104	0.067
DCP-v2 (ours)	**13152.160**	**114.683**	**78.333**	**0.000005**	**0.002**	**0.001**

Table 2. ModelNet40: Test on unseen point clouds ($[0, 180^\circ]$)

Model	MSE($\boldsymbol{R}$)	RMSE($\boldsymbol{R}$)	MAE($\boldsymbol{R}$)	MSE($\boldsymbol{t}$)	RMSE($\boldsymbol{t}$)	MAE($\boldsymbol{t}$)
ICP	1125.095	33.542	25.032	0.088	0.296	0.251
Go-ICP	216.790	14.724	3.511	**0.001**	**0.032**	0.012
FGR	130.621	11.429	2.960	**0.001**	**0.032**	**0.008**
PointNetLK	374.948	19.364	7.389	0.003	0.053	0.029
DCP-v2 (ours)	**23.471**	**4.845**	**2.486**	**0.001**	0.033	0.024

Table 3. ModelNet40: Test on unseen *partial* point clouds ($[0, 45^\circ]$)

# Iters	MSE($\boldsymbol{R}$)	RMSE($\boldsymbol{R}$)	MAE($\boldsymbol{R}$)	MSE($\boldsymbol{t}$)	RMSE($\boldsymbol{t}$)	MAE($\boldsymbol{t}$)
1	1.307329	1.143385	0.770573	0.000003	0.001786	0.001195
3	0.568834	0.754211	0.223045	0.000001	0.000872	0.000519
5	0.267259	0.516971	0.168858	0.000001	0.000870	0.000518
7	0.163877	0.404817	0.145387	0.000001	0.000870	0.000517

Table 4. ModelNet40: Test on unseen point clouds (iterative inference on testing, $[0, 45^\circ]$)

Additional Ablation Studies

We carried out more ablation studies dealing with large motions, iterative inference during testing, and dealing with partial point clouds (R2). Tables 1 and 2 show the results when the testing rotation is sampled in $[0, 90^\circ]$ and $[0, 180^\circ]$, respectively. The results suggest that in more challenging scenarios, DCP still outperforms other methods. Table 4 shows that when testing with iterative inference, DCP consistently increases performance. For partial matching, we simulate partial point clouds of $\mathcal{X}$ and $\mathcal{Y}$ by randomly placing a point in space and computing its 768 nearest neighbors in $\mathcal{X}$ and $\mathcal{Y}$ respectively; Table 3 shows the results.

References

[1] Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016. 1

[2] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on International Conference on Machine Learning*, ICML'15, pages 448–456. JMLR.org, 2015. 1

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 1

[4] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress. 1

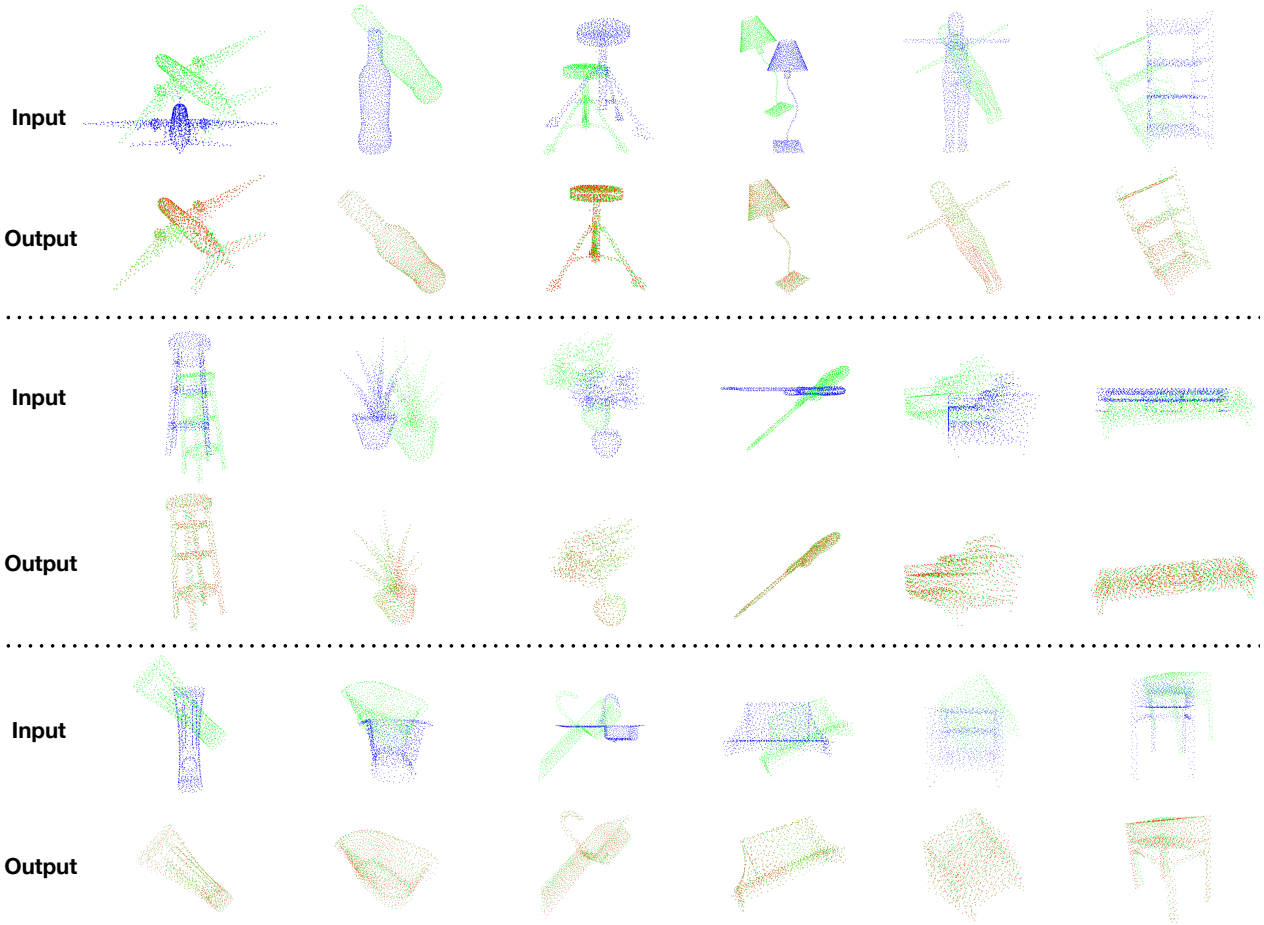

Figure 1. Results of DCP-v2. **Top**: inputs. **Bottom**: outputs of DCP-v2.

[5] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. 2017. 1

[6] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85. IEEE Computer Society, 2017. 1

[7] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, Jan. 2014. 1

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017. 1

[9] Yue Wang, Yongbin Sun, Sanjay E. Sarma Ziwei Liu, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics, to appear*, 2019. 1

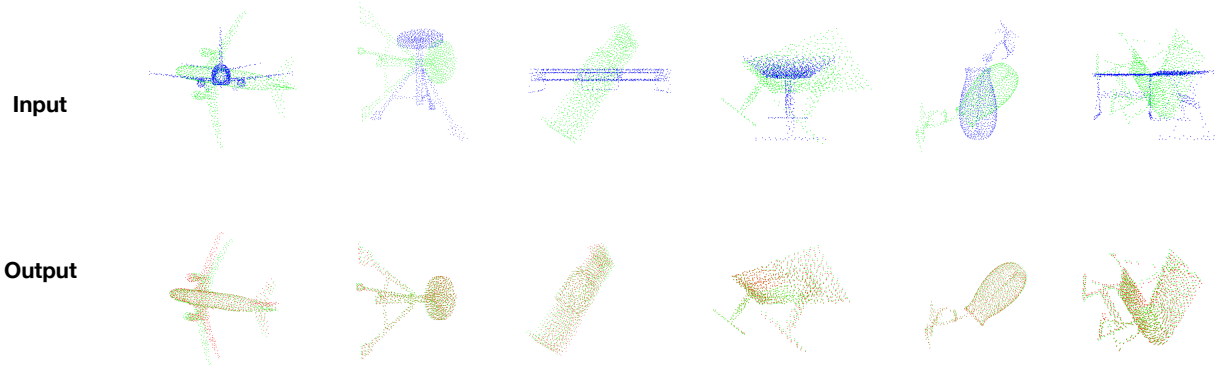

Figure 2. Results of DCP-v2 tested with large motion. **Top**: inputs. **Down**: outputs of DCP-v2

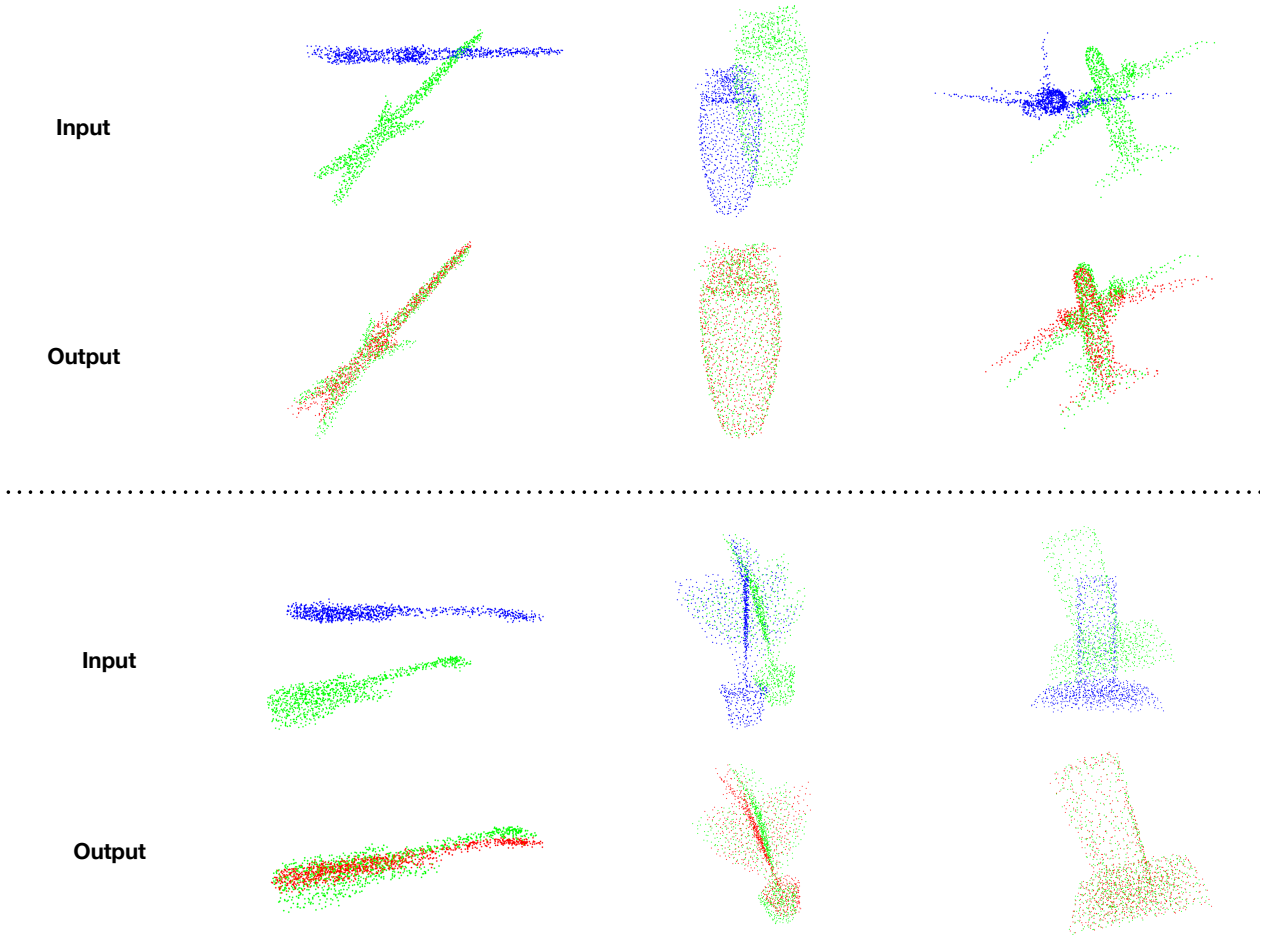

Figure 3. Results of DCP-v2 tested on noisy point clouds. **Top**: inputs. **Down**: outputs of DCP-v2