Supplementary Material

5.1 Training details

The SPD network is trained with a batch size of 16 for 1000 epochs, using the Adam optimizer, a learning rate of 0.001 with an exponential decay rate of 0.5 every 3×10^5 steps, and batch normalization. Training the SPD network takes approximately 3 hours ($N = 256$), and using DPDist to measure the distance between point clouds takes 6ms. All timing approximations were tested on NVIDIA GeForce RTX 2080 Ti GPU and an Intel Core i9 CPU at 3.60GHz.

5.2 Ablation study

We explore the influence of the 3DmFV parameters (number of local and global Gaussians) on our method's performance. Given an input point cloud of 512 points, we reconstruct a mesh from the learned implicit function representation using Marching cubes [12]. We then sample 10k points from the reconstructed mesh and compare them to 10k points sampled on the original CAD model using Chamfer L1 and normal consistency as specified in [14]. For each point from one set, we find it's nearest neighbor in the second set. Then we compute the euclidean distances and the normal consistency between them and average over all points in the set. We then alternate sets and average between the two results. We evaluate our results on ModelNet40 datase's "chair" category.

In the first experiment we explore the influence of the global Gaussian grid size. We use a local grid size of 3^3 and a global size of $4^3, 8^3$, and 16^3. The results in Table 1 are consistent with the thorough hyper parameter study conducted in [3] and show that 8^3 Gaussian grid is adequate, balancing the computation-accuracy trade-off.

In the second experiment we explore the influence of the local Gaussian grid size. We use a global grid size of 8^3 and a local size of $1^3, 3^3$, and 5^3. The results in Table 2 show that the results between 3^3 and 5^3 are comparable with a slight advantage to 3^3. However, this is most likely attributed to the small size of the dataset and we chose to use 5^3 in our experiments, maintaining a higher network capacity.

Number of Gaussian	$4 \times 4 \times 4$	$8 \times 8 \times 8$	$16 \times 16 \times 16$
Chamfer L1 $\downarrow$	0.130189	0.071805	0.058943
Normal Consistency $\uparrow$	0.718058	0.794560	0.809462

Table 1: Comparing global Gaussian grid sizes using Chamfer L1 and Normal consistency evaluation metrics. This experiment was done with a local patch size of 3^3.

Local Patch Size	$1 \times 1 \times 1$	$3 \times 3 \times 3$	$5 \times 5 \times 5$
Chamfer L1 $\downarrow$	0.103590	0.071805	0.072714
Normal Consistency $\uparrow$	0.693455	0.794560	0.739863

Table 2: Comparing local patch Gaussian grid sizes using Chamfer L1 and Normal consistency evaluation metrics. This experiment was done with a global Gaussian grid size of 8^3.

5.3 Training Auto-Encoders

In this experiment we compare DPDist to CD as loss function for training a simple auto-encoder. We use a PointNet encoder [16] and three fully connected layers with sizes of $1024, 1024, N * 3$ as the decoder (N is the number of output points). The loss for the auto-encoder is defined as the similarity between the output point cloud to the input point cloud. Previous works [6,1,24,7,11,25] use the CD or EMD loss between the point clouds.

Fig. 9 shows that when using DPDist, the generated point clouds suffer from high non-uniformity. Essentially, multiple points are able to coincide and satisfy the objective function. This flaw is a direct consequence of the main strength of the proposed method: the sampling invariance property. This property makes it robust to changes in sampling and replaces the comparison between samples to comparison between underlying surfaces. Our method is robust for both non-uniform and sparse sampling, and this is the reason it is effective in the registration task. In essence, there is a trade-off between sampling invariance and generation coverage. The focus of this paper is comparing between point clouds, therefore exploring modifications required for point generation is left for future work.

5.4 Real-world data

In this experiment, we use the Sydney Urban dataset, which contains LiDAR scans of outdoor objects. Because this dataset does not provide a ground truth surface, we use an equivalent class in the ModelNet dataset for training. We conduct the Translation detection test (Sec 4.2) on the car class and compare DPDist performance to the other measures. Remarkably, although the training was done on synthetic data, our method outperforms CD, EMD, and Hausdorff and is comparable to partial Hausdorff. Table 3 reports the transformation distance where the method reached a minimum (i.e. lower is better). These results align with our CAD experiments.

Note that we train our method on synthetic data without data corruptions such as noise and occlusions. Further improvement may be achieved by adding more realistic scenarios into the training data or by training the proposed method directly on data collected by real-world sensors.

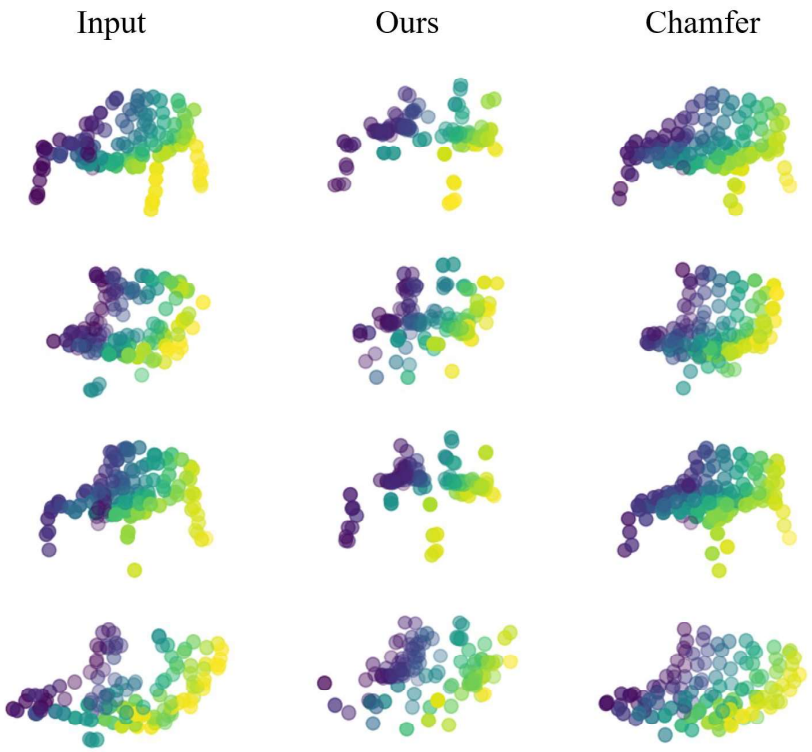

Fig. 9: Point cloud auto-encoding results. We can see the auto-encoder output results when training with DPDist (middle), and Chamfer (right) distance loss for $N = 128$. While Chamfer distance provides better coverage, our method gives a sparser output due to its sampling invariance property.

Method	Ours	CD	EMD	Hausdorff	PH9	PH8	PH5
Mean	0.01385	0.02879	0.02381	0.03201	0.01091	0.00863	0.02807
Std.	0.01061	0.01119	0.01335	0.01391	0.00740	0.00752	0.01491

Table 3: Detecting translation - given a set of translations, we report the transformation distance where the method reached a minimum (i.e., lower is better). Our method outperforms the commonly used CD and EMD, and is comparable with the partial Hausdorff variants.

.