# DPDist : Comparing Point Clouds Using Deep Point Cloud Distance

Dahlia Urbach[1], Yizhak Ben-Shabat[2], and Michael Lindenbaum[1]

[1] Techion IIT, Israel

[2] Australian National University, Australian Centre for Robotic Vision, Australia

dahliau@technion.ac.il, yizhak.benshabat@anu.edu.au,
mic@cs.technion.ac.il

https://github.com/dahliau/DPDist

**Abstract.** We introduce a new deep learning method for point cloud comparison. Our approach, named Deep Point Cloud Distance (DPDist), measures the distance between the points in one cloud and the estimated surface from which the other point cloud is sampled. The surface is estimated locally using the 3D modified Fisher vector representation. The local representation reduces the complexity of the surface, enabling effective learning, which generalizes well between object categories. We test the proposed distance in challenging tasks, such as similar object comparison and registration, and show that it provides significant improvements over commonly used distances such as Chamfer distance, Earth mover's distance, and others.

**Keywords:** 3D Point Clouds, 3D Computer Vision, 3D Deep Learning, Distance, Registration

## 1  Introduction

Recent advancements in 3D sensor technology have led to the integration of 3D sensors into many application domains, such as virtual and augmented reality, robotic vision, and autonomous systems. These sensors supply a set of 3D points, sampled on surfaces in the scene, known as a point cloud. As raw, memory-efficient outputs of 3D sensors, point clouds are a common 3D data representation. However, in contrast to more traditional data (e.g., images), point clouds are unstructured, unordered, and may have a varying number of points. Therefore, unlike traditional signals, they may not be represented as values on some regular grid, and are difficult to process using common signal processing tools.

In particular, many applications, such as registration, retrieval, autoencoding etc., require comparisons between two or more point clouds. Comparing point clouds is difficult for two main reasons. First, because they are not a function on a grid, point clouds cannot be compared using a common metric (such as Euclidean metric). Second, when comparing point clouds sampled from a 3D surface, we usually want to compare their underlying surfaces and not the given
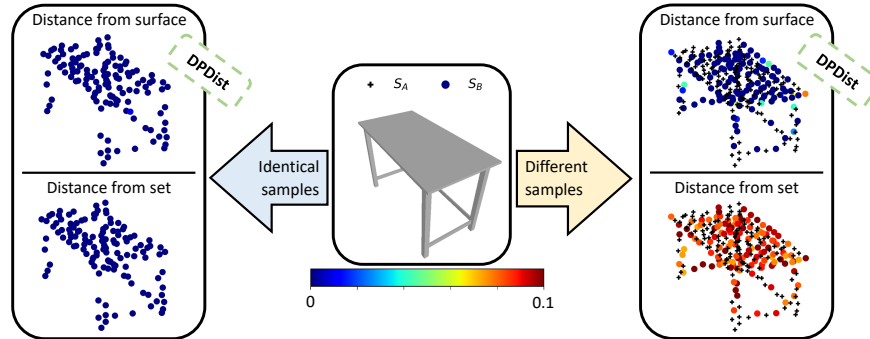
Fig. 1: Distance to surface vs. distance to sampled point cloud. Every figure contains two samples of the same object (a table). The samples can be identical (left) or different (right). The distance of every point from one point cloud (disks) to the surface estimated from the other point cloud (+) is given as a colored code in the top two figure. One can see that even for the different sampling most of the distances are still small. The distance from every point in one point cloud (disks) to the nearest point in the other point cloud (+) is presented as a color code in the bottom figure. We can see that different sampling produce significantly larger distances.

sample points. Common methods for point cloud comparisons, such as Chamfer distance and Earth mover's distance (see more in Section 2) compare the given clouds directly, rely on the unstable correspondence process, and are sensitive to sampling.

We propose a method for comparing point clouds that measures the distance between the surfaces that they were sampled on (see Fig. 1). We use a deep learning network to estimate the distance function from a point to an underlying continuous surface. By using a vector representation of point clouds, the 3D modified Fisher vector (3DmFV) [3], the network can work directly with point clouds, despite their irregular structure. An additional advantage of the particular 3DmFV representation is its grid structure, which enables the network to process a local surface representation rather than calculating a representation at every location. Working on local representation reduces the complexity of the surface, enabling efficient and effective learning, which generalizes well between object categories.

We test the robustness of the proposed approach to sampling, on challenging tasks such as detecting small transformation and comparing similar objects. We incorporate it into the learning process of a registration network. We compare it to several, more traditional, distance measures and show its advantage, which is significant, especially when the task is harder and the point cloud is sparser.

The main contributions of this paper are as follow:

- DPDist: A new distance measure between point clouds, that operates directly on raw unstructured points, but measures distance to the underlying continuous surface.
- An algorithm that implements the DPDist using local implicit representation, in an effective and efficient way.
- An improved variant of the PCRNet registration algorithm [20] that applies the proposed DPDist in a training loss function.

## 2 Related Work

### 2.1 Point Cloud Distance

Point clouds may represent raw data, but in the context of 3D sensors, they represent the surfaces on which they are sampled. As such, the distance between the point clouds, should preferably refers to the distance between the sampled continuous surfaces and be robust to sampling and noise.

Most distances between points sets are based on an underlying metric, assigning a distance value $\|a - b\|$ to every two points $a, b$. Like the majority of previous work, we use a Euclidean metric.

Consider two point clouds $S_A, S_B \subseteq \mathbb{R}^3$ , with $N_A, N_B$ points in each cloud, respectively. An early method for comparing point sets is the Hausdorff distance $(\mathcal{D}_H())$, which builds on the minimum distance from a point to a set

$$d(x, y) = \|x - y\|_2 \tag{1}$$

$$D(x, S) = \min_{y \in S} d(x, y) \tag{2}$$

and calculates a symmetric max min distance [8].

$$\mathcal{D}_H(S_A, S_B) = \max\{\max_{a \in S_A} D(a, S_B), \max_{b \in S_B} D(b, S_A)\} \tag{3}$$

The Chamfer distance $(\mathcal{D}_{CD}())$ (Eq. 4), and partial Hausdorff $(\mathcal{D}_{PH(f)}())$ (Eq. 5) distances are two of its variants, based on averaging (instead of taking the maximum) and on robustly ignoring a fraction $(1 - f)$ of the points that are far from the other object, respectively

$$\mathcal{D}_{CD}(S_A, S_B) = \frac{1}{N_A} \sum_{a \in S_A} \min_{y \in S_B} d(a, y)^2 + \frac{1}{N_B} \sum_{b \in S_B} \min_{y \in S_A} d(b, y)^2 \tag{4}$$

$$\mathcal{D}_{PH(f)}(S_A, S_B) = \max\{T | \frac{|\{a | D(a, S_B) \le T\}|}{|S_A|} < f, \frac{|\{b | D(b, S_A) \le T\}|}{|S_B|} < f\} \tag{5}$$

where $a \in S_A, b \in S_B$.

Unlike the Hausdorff distance and its variant, which rely on finding the nearest neighbor to every point, the Earth mover's distance $(\mathcal{D}_{EMD})$ (Eq. 6), also

known as the Wasserstein distance, is based on finding the 1-1 correspondence (or bijection, $\xi$) between the two point sets, so that the sum of distances between corresponding points is minimal:

$$\mathcal{D}_{EMD}(S_A, S_B) = \min_{\xi: S_A \to S_B} \sum_{a \in S_A} \|a - \xi(a)\|_2 \qquad (6)$$

Clearly, this distance measure is limited to point clouds with the same point count. Other measures includes Geodesic distances, which provide deformation insensitive measures [4], and measures relying on the distance between point cloud vector descriptors, such as PointNet [16], VoxNet [13], or 3DmFV [3].

Here we focus on the direct distances, and specifically on CD and EMD like measures. These are the leading approaches for assessing subtle differences in point clouds, and are used as evaluation distances and as loss functions for the training of neural networks [20,6,1,24,7,11,25]. They highly depend on point correspondence, which makes them sensitive to sampling and noise. In this work, we propose a novel method for comparing 3D point clouds where the measured distance is between the points from the first point cloud to the surface represented by the second (without constructing a mesh) and the other way round. As a distance, it can replace the aforementioned distances as a loss function for various neural networks that require point cloud comparison.

### 2.2    Deep Learning on 3D Point Clouds

Processing a point cloud with deep learning is challenging because point cloud representation is unordered, unstructured, and has unknown number of points, rendering it an unnatural input for deep networks. Several methods were proposed to overcome these challenges. One approach quantizes the points into a voxels grid. This approach allows to directly use 3D convolutional neural networks (CNNs) [13] but induces quantization. Another approach encodes the point cloud with a kd-tree and uses it to learn shared weights for nodes in the tree [10]. A recent popular network for processing point clouds is PointNet [16]. It computes features separately for each input point, and then extracts a global feature using a permutation-independent (symmetric) function (e.g., max/avg). An effective variant, PointNet++ [17], applies the PointNet encoding locally and hierarchically.

The recent 3D modified Fisher Vector (3DmFV) approach [3] builds on the well known Fisher Vectors [19] which, in turn, are based on the Fisher Kernel (FK) principles [9]. 3DmFV represents each point's deviations from a mixture of Gaussians, lying on a regular grid, and then applies symmetric functions to get a global generalized Fisher Vector representation [19] that integrates into a 3D CNN. The proposed method uses the regular grid structure of the 3DmFV representation, to extract multiple local patches without any recalculation.

### 2.3    Deep Implicit Function

Two recent works that are the closest to the proposed method rely on continuous implicit surface representations. Occupancy Networks [14] learn an indica-

tor function between the inner part and the outer part of a model, where the boundary between these parts is an implicit function representing the continuous boundary. DeepSDF [15] learns a signed distance function to the surface, and estimates the boundary as the zero surface. Both methods rely on a global presentation of the point cloud and the estimated surface. Occupancy Networks [14] rely on PointNet, and DeepSDF specifies each model's surface by an decoder training process, which provides both the weights and the latent layer values (without using an encoder).

Unlike DeepSDF, our proposed algorithm is able to transform a point cloud into underlying surface representation quickly and can therefore use the distance-to-surface principle to calculate the distance between point clouds. Unlike Occupancy Networks, our proposed method calculates the distance to the surface and can therefore estimate whether two point clouds, sampled from surfaces, are indeed sampled from the same surface. In addition, unlike the two aforementioned methods, our approach can extract and use effective local representation, which makes both the learning and the estimate, accurate and effective.

### 2.4   Registration

Registration between point clouds is a fundamental task. A popular, classic (non-learnable) algorithm is Iterative Closest Point (ICP) [18], which relies on finding closest point pairs. To overcome ICP's convergence to local minima, recent ICP variants [23] start by finding a good initialization at some computational expense.

Recently, deep-learning-based registration methods have been introduced. Some methods [21,2] learn to regress the transformation parameters. The PCR-Net [20], however, learns by comparing point clouds. The method aims to find the transformation that converts the measured point cloud (source) to the model (template). At every iteration, it maintains a temporary transformed source, calculated with a currently available transformation. A pose estimation network gets PointNet representations of this temporary cloud and of the template cloud, and provides a new, improved, transformation (represented as translation parameters and quaternion rotation parameters). The network is trained by a loss based on the Chamfer distance or the EMD between the temporary cloud and the template cloud. In this paper, we test the proposed new distance DPDist, as a loss function for training the PCRNet.

## 3   The Deep Distance

We now present the main contribution of this work: a new method for fine comparison of point clouds that we call the Deep Point Cloud Distance (DPDist).

The DPDist method is based on estimating the distances of points from one cloud to the underlying continuous surface corresponding to the other point cloud.

Let $A, B \in \mathbb{R}^3$ be two continuous surfaces, and $S_A = \{a_i\}_{i=1}^{N_A}, S_B = \{b_j\}_{j=1}^{N_B}$ two sets of points sampled from them. Our aim is to find the distance from each
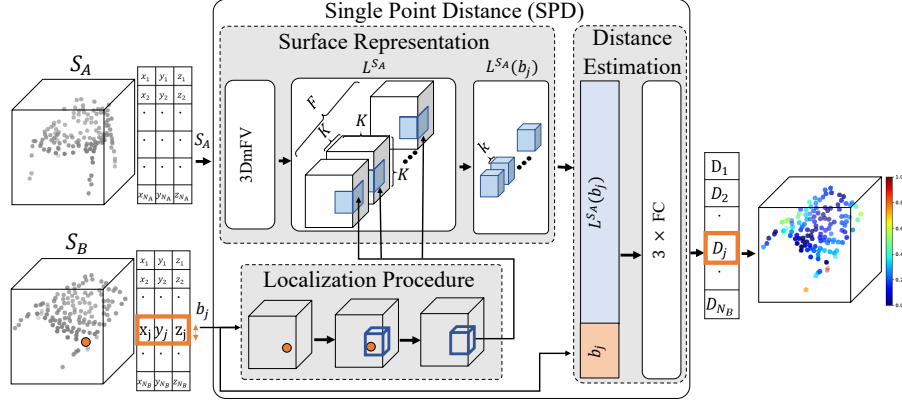
Fig. 2: SPD illustration. Two point clouds $(S_A, S_B)$ are input to a network that estimates the distance between each query point from $S_B$ to the surface represented by $S_A$. SPD uses 3DmFV to extract surface representation $L^{S_A}$ (4D tensor) from $S_A$. It then applies a localization procedure for each query point $b_j$ to find its closest grid point (the grid is distributed uniformly: $K^3$). Given the grid point, it extracts the corresponding 4D sub tensor $L^{S_A}(b_j)$. The SPD processes each query point $b_j$ and its matched local representation to produce its distance from the surface $D_j = D_{SPD}(b_j, S_A)$.

point $b_j \in S_B$ to the closest point in $A$ (and vice versa). Formally, this distance is

$$D(b, A) = \min_{y \in A} d(b, y), \tag{7}$$

where $D()$ is the distance from a point to a surface and $d()$ is any distance between points in $\mathbb{R}^3$ (here we shall use the Euclidean distance). The surface $A$ is not available, and hence we propose to use the approximation

$$\hat{D}(b, A) = \phi(b, S_A), \tag{8}$$

where $\phi$ is a learned distance function depending on the samples of $A$.

In principle, we can design a network that accepts the coordinates of the point $b$, and some representation of the cloud of points $S_A$ (e.g., PointNet [16] or 3DmFV [3]). This network should learn to provide the distance. We found, however that due to the large variation of objects, learning the distance function is hard. In particular, even with a complex network and a lot of examples, the results we obtained were not very good. See [15,14] where a similar task of estimating an implicit function by using at least 7 fully connected layers to express the complicated geometric features.

### 3.1   Local Representation

Because of the difficulty of learning the distance function, we prefer to model the surface $A$ by parts, so that every part is less geometrically complex. In principle,

we could derive such a piecewise model by training a different distance estimator for every spatial region. Instead, we choose the more elegant approach of using the 3D modified Fisher Vector (3DmFV) representation.

The 3DmFV representation is preferable for two reasons. First, it provides better performance than some other representations (and in particular, it is better than the PointNet; see [3]). In addition, it uses a grid structure therefore allowing us to specify a local, partial representation by choosing a subgrid.

First, we calculate the global representation from the cloud $S_A$. To that end, we decide on the representation grid size $K$, and specify the 3D grid of size $K \times K \times K$. This grid specifies a mixture of Gaussians, one for each grid point. The representation itself results from calculating the derivative of the Gaussian with respect to its parameters, and taking maximum, minimum, and average statistics over these derivatives; see [19,3]. Overall, we calculate $F$ statistics for each Gaussian and concatenate them to a global representation $L^{S_A} \in \mathbb{R}^{K \times K \times K \times F}$. This representation is calculated once for every point cloud and the partial representations are simply cut from it.

We extract the local representation from this global representation as follows: for each query point $b_j$ from the point cloud $S_B$, find its nearest grid point and specify a subgrid of size $k$, centered at this grid point ($k$ is always odd). Then, the entries of $L^{S_A}$ corresponding to the subgrid are extracted and concatenated to give the local representation, denoted $L^{S_A}(b_j) \in \mathbb{R}^{k \times k \times k \times F}$. Note that the local representation is a representation of the point cloud $S_A$, in a region determined by $b_j$.

### 3.2   Point-to-Implicit-Surface Distance Estimation

To estimate the distance from a point to the underlying surface associated with a point cloud, we use a learned function. We chose a multilayer, fully connected neural network (FC). The input to this network is the coordinate vector specifying the point $b_j$, concatenated to the local representation $L^{S_A}(b_j)$. The neural network, denoted $\psi$, learns to estimate the distance $\psi(b_j, L^{S_A}(b_j))$ between the point and the corresponding local representation. The full process, which gets the point $b_j$ and the point cloud $S_A$ (as input), is carried out as follows. The process first extracts the local representation corresponding to $b_j$ from the 3DmFV vector; concatenates it to the point coordinates; and processes them using the network $\psi$. We denote it: SPD (Single point distance). That is

$$\hat{D}(b_j, A) = D_{SPD}(b_j, S_A) = \psi(b_j, L^{S_A}(b_j)) \tag{9}$$

The detailed description of the SPD neural network is presented in Fig. 2. We found indeed that learning the distance function of a local representation is faster, more accurate, and can generalize better for unseen objects. Furthermore, representing a surface $A$ directly from a set of points $S_A$ using the proposed approach yields sampling invariance (sparse or non-uniform point clouds may represent the same surface). Fig. 3, illustrates the distance maps of SPD and compares it to the distance to the cloud, for a dense set of query points that lie
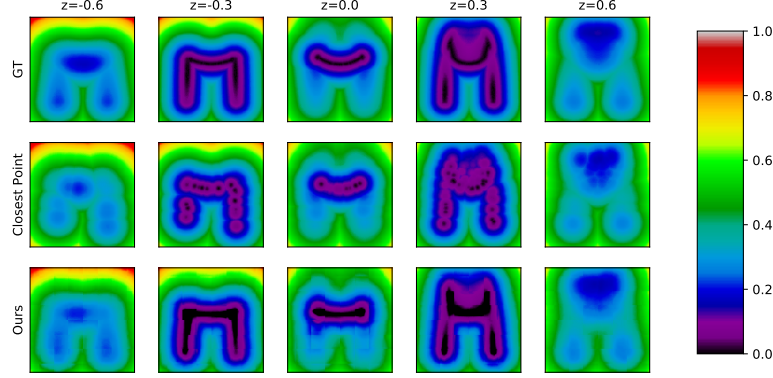
Fig. 3: Three distance map to the "Chair" CAD model: The ground truth distance from the CAD model (upper row), the distance to the 'closest point' in a 128 point cloud, sampled from the CAD model (middle row), and the proposed SPD network distance from the surface specified by the same point cloud (bottom row). For each 3D distance map, several XY slices are shown. Clearly, the SPD is smoother than the distance to the point cloud, and approximates the ground truth better.

on a regular XY grid in different spatial depths ($z = \{-0.6, -0.3, 0, 0.3, 0.6\}$). The advantage of the SPD for this low sampling data ($N = 128$) is clear.

### 3.3  Estimating the Distance Between Point Clouds

The point to surface distance estimates, developed above may be easily applied to measuring the distance between two point clouds. The average distance between all points in one cloud, $S_B$, to the underlying surface corresponding to the other point cloud is a straightforward choice. Here we often use a symmetrized version to yield the DPDist distance between two point clouds.

$$\mathcal{D}_{DPDist}(A, B) = \frac{1}{N_A} \sum_{i=1}^{N_A} D_{SPD}(a_i, S_B) + \frac{1}{N_B} \sum_{j=1}^{N_B} D_{SPD}(b_j, S_A) \qquad (10)$$

### 3.4  DPDist as Loss Function for Training Neural Networks

After DPDist is fully trained, it can be used as a distance estimator building block that can be connected as a loss function to various tasks. Generally, its weights can either be frozen or adjusted to the desired task, but as a loss function, the weights remain constant. It takes two point clouds $S_A$,$S_B$ as input and outputs the estimated distance between the underlying represented surfaces.

DPDist uses a neural network, and is therefore differentiable and can be easily integrated as a loss function for training in different tasks.

## 4  Experiments

We start by conducting a thorough analysis of the method's robustness to sampling. We would like to emphasize that for tasks involving coarse differences between point clouds, such as categorization and coarse registration, distances that are based directly on the point clouds (e.g., Chamfer distance) or even distances based on vector representations (e.g., distance between 3DmFVs) should suffice. The distance suggested here can detect small, subtle differences. Therefore, we chose to experiment with the following challenging tasks: object instance identification, detecting small translations, and detecting small rotations. We compare our method with the following methods: Hausdorff (H), Chamfer distance (CD), partial Hausdorff (PHx), Earth mover's distance (EMD), and 3DmFV representation (we have shortened PH(f) to PHx, where f= 0.x). We then evaluate our method's effectiveness as a loss function in training a registration network. We provide additional point cloud generation experiments in the supplemental material. They show that DPDist's main advantage of sampling invariance can introduces significant challenges for this task. Additionally, we report in the supplemental results on real-world data that align with our findings on CAD models.

### 4.1  Setup Details

**Dataset** The experiments were conducted on the ModelNet40 dataset [22]. It contains 12311 CAD models from 40 object categories. We normalize each shape to 80% of the unit sphere. We split the data into train/test as in [16].

**Training data** For training the SPD network, we sample points from the CAD models, keeping their distance from the surface. At each training step, we sample two point clouds. The first, representing the surface, contains $N$ points sampled from the surface. The second set is the query set. It contains $0.5N$ points sampled from the surface, $0.25N$ points sampled uniformly in the region closer than 0.1 to the surface, and $0.25N$ points uniformly sampled from the unit cube. Adding the second type of query points increases the density of the query points near the surface and enhances the learning of small geometric details; see [15]).

**Training details** Fig. 2 describes the overall architecture of SPD: We set the 3DmFV layer parameters to $K = 8$, $k = 5$, and the Gaussian's sigma to 0.125. Additional ablation study of the influence of 3DmFV parameters is provided in the supplemental material. To evaluate the parameters we used marching cubes [12] for surface reconstruction as done in [15,14]. The neural network is composed of three fully connected layers with a size of 1024, each processed with
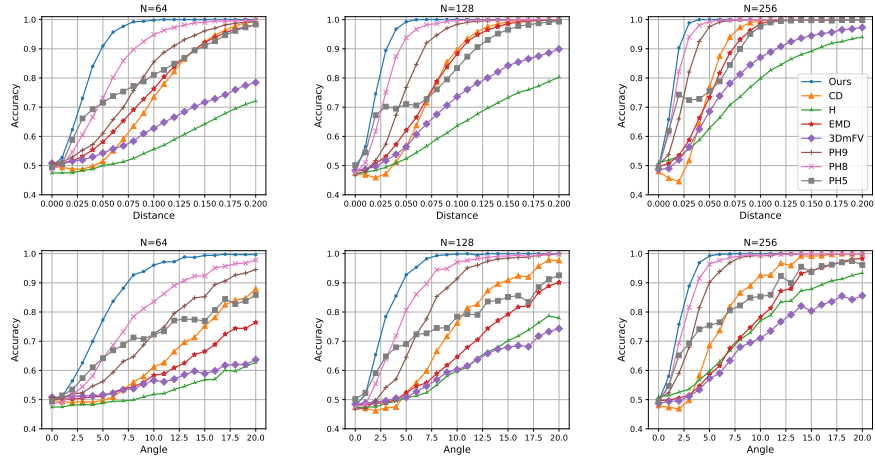
Fig. 4: Detecting translation (top) and rotations (bottom) using DPDist and other point cloud distances, for different point cloud densities $N = 64, 128, 256$.

RELU activation. During training, we minimized the mean $L_1$ distance between the network output (our prediction) and the real distance (GT) associated with each query point, i.e. $Loss = \frac{1}{|S_B|} \sum_{x_i \in S_B} ||D_{SPD}(x_i, S_A) - GT(x_i)||_1$ .

We train separate networks for each point cloud input size $N = 32, 64, 128, 256, 512, 1024$. Our early experiments showed that we could use a network trained for an input size of $N = 512$ for all different size inputs; however, training each network for a specific input size yields higher accuracy.

For the following experiments, we trained the SPD using the train-set of the "Chair" category, which contains 889 CAD models.For specific training parameters and further details, please refer to the supplemental material.

**Evaluation data sampling** To obtain the data for evaluation, we first sample $N_C$ points uniformly from each CAD model using Farthest Point Sampling (FPS) [5]. We then sample two disjoint sets $S_A, S_B$ with $N_A, N_B$. In most experiments, $N_A = N_B = N_C/2 = N$.

### 4.2   Robustness to Sampling

Discriminating between very different objects from dense samples of their surfaces is easy, and performance is highly independent of the actual sampling. Harder tasks and sparser sampling are more challenging. We consider here three discrimination tasks associated with similar objects, perform them using different distances between their point cloud samples, and examine their robustness to sampling.

We conducted the experiments on the "Chair" category test-set from Modelnet40 with a total of 100 chairs.
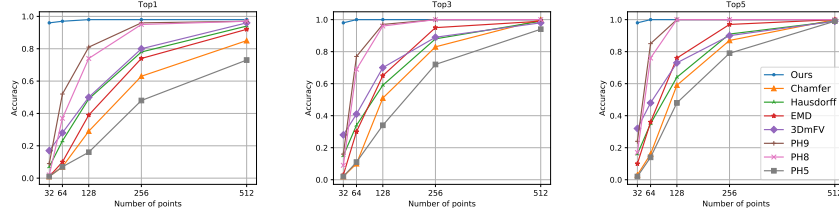
Fig. 5: Instance identification test - comparing top 1, 3, and 5 accuracy in nearest neighbor identification robustness test. The DPDist is the clear winner.

**Translation detection test:** The task here is to discriminate between a stationary object and a moving object. Thus, in every test we consider three point clouds: $S_A$ - a sampling of the object in its original position, $S_B$ - a sampling of the object after it was translated (see below), and $S_C$ - another sampling of the same object without movement. We can discriminate between the stationary object and the moving object, when the following distance inequality $\mathcal{D}(S_C, S_A) < \mathcal{D}(S_B, S_A)$ holds true. We consider translation along the 26 directions $(\theta_1, \theta_2, \theta_3), \theta_{1,2,3} \in \{0, 1, -1\}$, with magnitude uniformly sampled in $[0, 0.2]$. The accuracy is defined separately for each translation magnitude, as a fraction of successful tests. The results are shown in Fig. 4 (top). As expected, for large translations and high sampling density, almost all methods succeed. For small densities and translations, some distances succeed better and the proposed method is best by a large margin. Remarkably, the partial Hausdorff variants succeed better than CD and EMD.

**Rotation detection test:** This time, the task is to discriminate between a stationary object and a rotated object. The rotation angle magnitude is uniformly sampled in $[0, 20]$ (deg) around the 26 directions specified in the last section. Fig. 4 (bottom) shows that DPDist performs significantly better than the other distances in the presence of small rotations and densities.

**Identification test:** We evaluate the sampling robustness of different point cloud distances by testing their ability to discriminate between objects from the same category. Given two samples of the same CAD object $S_A$ and $S'_A$, we test if the two samples of the same object are closer to each other than to samples of any other objects from the same category. For each point cloud $S_A$, we sort its distances from $S'_A$ and from the point cloud of all other objects. The identification is considered "Top $m$" successful if $S'_A$ is in the "$m$" closest point clouds to $S_A$ (out of 100). The success rate is the fraction of objects for which the identification is "Top $m$" successful. Fig. 5 shows that the proposed DPDist method is robust and works much better than other methods. Remarkably, the partial Hausdorff distance is the next best method, and it easily overcomes the more standard CD and EMD.

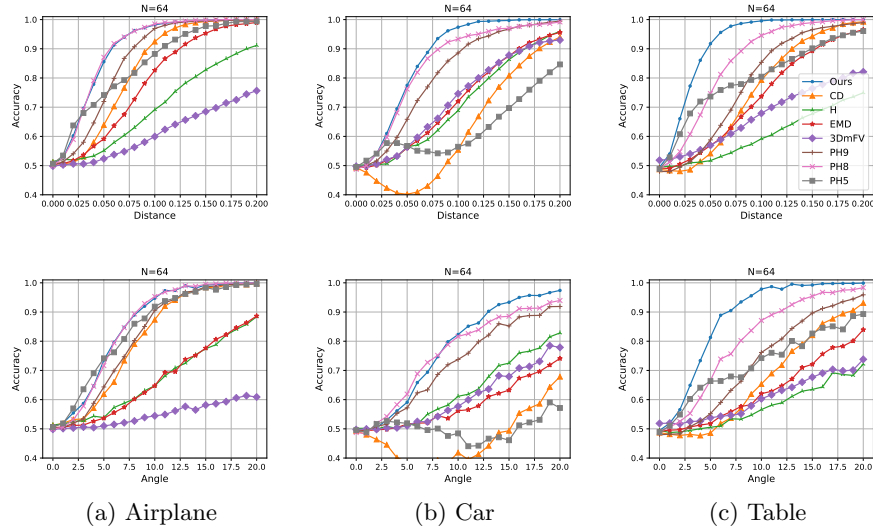(a) Airplane               (b) Car                  (c) Table

Fig. 6: Translation (top) and rotation (bottom) detection experiments evaluated
on (a) "Airplane" (b) "Car", and (c) "Table" as before. Remarkably, DPDist
is still the best method, although the SPD was trained over another category
("Chair") and not over these categories. The generalization ability is a result of
using local surface modelling.

### 4.3   Local Representation Learning Generalization

A major advantage of the proposed method is its ability to learn local representa-
tions, which makes the implicit surface estimation effective and computationally
efficient.

Objects are different from each other but often share local parts. That is,
many objects contain planes, corners, curved surfaces, etc. We therefore hy-
pothesize that training the SPD network over local representations of just one
reasonably rich object category (e.g., "Chair") makes the learning universal, be-
cause the category includes enough variations of small patches to generalize for
unseen categories. Therefore, although we train our SPD only on the "Chair"
category, we expect it to generalize to other categories.

We test our hypothesis by performing the translation and rotation tests over
"Airplane", "Car", and "Table" categories on a network that was trained only
on "Chair". Fig. 6 shows that the DPDist is still the most discriminative method
and more robust to sampling than the other tested methods.

### 4.4   Applying Point Cloud Distances to Registration Learning

In this section, we use an existing registration network, and show its performance
improvement when using DPDist as its loss function. We choose the recent PCR-
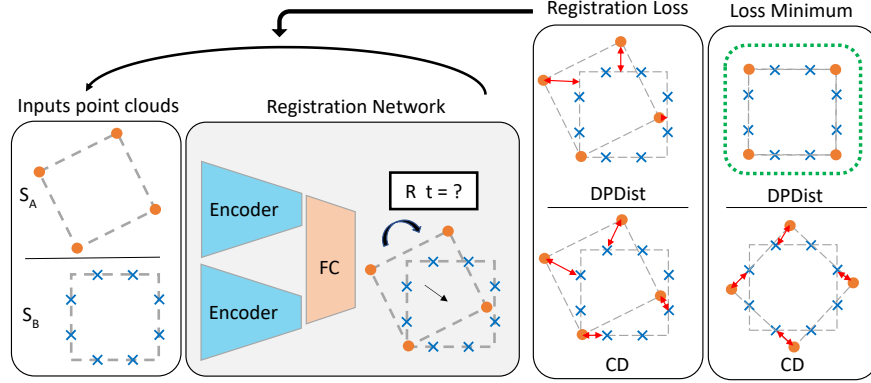Net [20] (see also Section 2.4) since it compares point clouds in its training loss

Fig. 7: Learning registration using DPDist vs. CD distances as loss function. Two point cloud $(S_A,\ S_B)$ are input to a network that regresses the transformation between them. The chosen loss highly effects the output transformation as the DPDist loss minimum is consistent with zero distance, while Chamfer loss results in nonzero distance for different sampling of the same object.

function. Fig. 7 illustrates the challenges of using a correspondence-based distance vs. the proposed distance. We compare DPDist to CD and EMD distances as the loss function for training the iterative PCRNet.

We used the PCRNet as described in [20], but found that limiting the output rotation angle to 45 degrees improves the stability when training on small point clouds. In the context of iterative registration, this modification does not limit the registration range. Using the "Chair" category and following [20], we randomly generate 5070 different transformations for training and other 5070 transformations for testing. The transformations include a rotation between $[-45, 45]$ degrees about one random direction and a translation between $[-0.1, 0.1]$ in another random direction. The evaluation metric at test time is the "success ratio" [20]: the percentage of point clouds with a transformation error under a given 'max error' threshold.

To demonstrate the DPDist robustness, we conduct the following experiment: Given a CAD model input, we generate two point clouds by sampling $2N$ points using FPS, and randomly divide it into two disjoint clouds $S'_A$ and $S_B$ of size $N$. We transform $S'_A$ to $S_A$ and store the transformation matrix as ground truth. We then train PCRNet using different distance losses. Fig. 8 shows the results for training PCRNet using CD, EMD, and the proposed DPDist for different sample sizes. The significant advantage of DPDist is clear.

The relative inaccuracy of the CD and EMD distances is due to the different samples. To demonstrate that this is indeed the reason, we added another success ratio curve with 128 points, but this time, $S'_A = S_B$. That is, the two point clouds (before transformation) are identical. We can see in Fig. 8d that this time, the performance with CD and EMD is nearly perfect.
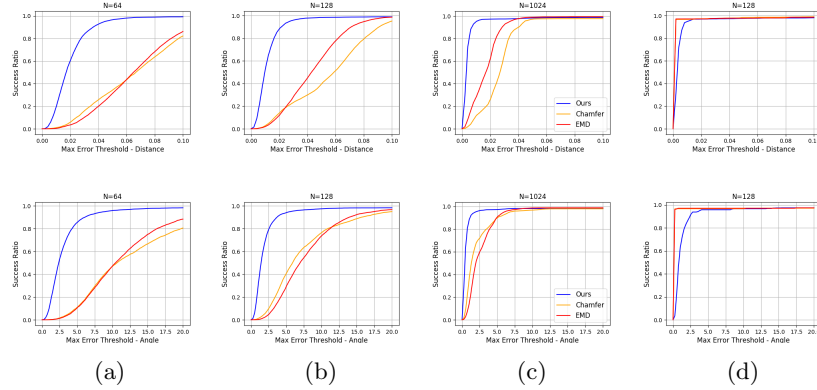
Fig. 8: Results for training PCRNet using CD, EMD, and the proposed DPDist over a single category ("Chair"). We show the success ratio vs. max error threshold for translation (top) and rotation (bottom) of the following input point cloud sizes: $N = 64, 128, 1024$ (a,b,c). DPDist consistently achieves high performance for various densities. (d) shows the case of two identical samples $S'_A = S_B$, where CD and EMD gets nearly perfect results thanks to the high correspondence between the clouds' samples.

## 5    Conclusions

Comparing between point clouds is a fundamental data analysis task. For point clouds, obtained by sampling some underlying continuous object surfaces, the actual hidden goal is to compare between these objects. Most methods, however, rely on distances between the raw points and are therefore sensitive to the uncertainties involved in the sampling process.

The method proposed here, DPDist, estimates the distances of points from one cloud to the underlying continuous surface corresponding to the other point cloud and the other way around. DPDist is fast and effective. It is more accurate than the commonly used Chamfer distance and Earth mover's distance methods. Its advantage is significant especially for difficult tasks, such as discriminating between similar objects, and for sparse point clouds. For example, using a DPDist dependent loss function, we were able to train a registration network so that it provides good results with point clouds of size 64.

Unlike other methods, we provide a fast process for generating the surface representation. This representation is local but is created efficiently as part of a global representation. Being local enables us to represent fine surface details with a moderately deep network, and makes the learned network universal: learning the local descriptions associated with one category is general enough to represent local descriptions of objects from other categories.

# References

1. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning representations and generative models for 3d point clouds. arXiv preprint arXiv:1707.02392 (2017)
2. Aoki, Y., Goforth, H., Srivatsan, R.A., Lucey, S.: Pointnetlk: Robust & efficient point cloud registration using pointnet. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7163–7172 (2019)
3. Ben-Shabat, Y., Lindenbaum, M., Fischer, A.: 3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks. IEEE Robotics and Automation Letters **3**(4), 3145–3152 (2018)
4. Bronstein, A.M., Bronstein, M.M., Guibas, L.J., Ovsjanikov, M.: Shape google: Geometric words and expressions for invariant shape retrieval. ACM Transactions on Graphics (TOG) **30**(1), 1–20 (2011)
5. Eldar, Y., Lindenbaum, M., Porat, M., Zeevi, Y.Y.: The farthest point strategy for progressive image sampling. IEEE Transactions on Image Processing **6**(9), 1305–1315 (1997)
6. Fan, H., Su, H., Guibas, L.J.: A point set generation network for 3d object reconstruction from a single image. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 605–613 (2017)
7. Groueix, T., Fisher, M., Kim, V., Russell, B., Aubry, M.: Atlasnet: A papier-mâché approach to learning 3d surface generation. In: CVPR 2018 (2018)
8. Huttenlocher, D.P., Klanderman, G.A., Rucklidge, W.J.: Comparing images using the hausdorff distance. IEEE Transactions on pattern analysis and machine intelligence **15**(9), 850–863 (1993)
9. Jaakkola, T., Haussler, D.: Exploiting generative models in discriminative classifiers. In: Advances in neural information processing systems. pp. 487–493 (1999)
10. Klokov, R., Lempitsky, V.: Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In: Computer Vision (ICCV), 2017 IEEE International Conference on. pp. 863–872. IEEE (2017)
11. Li, J., Chen, B.M., Hee Lee, G.: So-net: Self-organizing network for point cloud analysis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
12. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. ACM siggraph computer graphics **21**(4), 163–169 (1987)
13. Maturana, D., Scherer, S.: Voxnet: A 3d convolutional neural network for real-time object recognition. In: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on. pp. 922–928. IEEE (2015)
14. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4460–4470 (2019)
15. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 165–174 (2019)
16. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 652–660 (2017)
17. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems. pp. 5099–5108 (2017)

18. Rusinkiewicz, S., Levoy, M.: Efficient variants of the icp algorithm. In: Proceedings Third International Conference on 3-D Digital Imaging and Modeling. pp. 145–152. IEEE (2001)

19. Sánchez, J., Perronnin, F., Mensink, T., Verbeek, J.: Image classification with the fisher vector: Theory and practice. International Journal of Computer Vision **105**(3), 222–245 (2013)

20. Sarode, V., Li, X., Goforth, H., Aoki, Y., Srivatsan, R.A., Lucey, S., Choset, H.: Pcrnet: Point cloud registration network using pointnet encoding. arXiv preprint arXiv:1908.07906 (2019)

21. Wang, Y., Solomon, J.M.: Deep closest point: Learning representations for point cloud registration. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3523–3532 (2019)

22. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2015)

23. Yang, J., Li, H., Campbell, D., Jia, Y.: Go-icp: A globally optimal solution to 3d icp point-set registration. IEEE transactions on pattern analysis and machine intelligence **38**(11), 2241–2254 (2015)

24. Yang, Y., Feng, C., Shen, Y., Tian, D.: Foldingnet: Point cloud auto-encoder via deep grid deformation. In: 31st Meeting of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2018. pp. 206–215. IEEE Computer Society (2018)

25. Zhao, Y., Birdal, T., Deng, H., Tombari, F.: 3d point capsule networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)