# Efficient Learning on Point Clouds with Basis Point Sets

Sergey Prokudin *
Max Planck Institute for Intelligent Systems
Tübingen, Germany
sergey.prokudin@tuebingen.mpg.de

Christoph Lassner †
Amazon
Tübingen, Germany
classner@amazon.com

Javier Romero †
Amazon
Barcelona, Spain
javier@amazon.com

## Abstract

*With the increased availability of 3D scanning technology, point clouds are moving into the focus of computer vision as a rich representation of everyday scenes. However, they are hard to handle for machine learning algorithms due to their unordered structure. One common approach is to apply occupancy grid mapping, which dramatically increases the amount of data stored and at the same time loses details through discretization. Recently, deep learning models were proposed to handle point clouds directly and achieve input permutation invariance. However, these architectures often use an increased number of parameters and are computationally inefficient. In this work we propose basis point sets (BPS) as a highly efficient and fully general way to process point clouds with machine learning algorithms. The basis point set representation is a residual representation that can be computed efficiently and can be used with standard neural network architectures and other machine learning algorithms. Using the proposed representation as the input to a simple fully connected network allows us to match the performance of PointNet on a shape classification task, while using three orders of magnitude less floating point operations. In a second experiment, we show how the proposed representation can be used for registering high resolution meshes to noisy 3D scans. Here, we present the first method for single-pass high-resolution mesh registration, avoiding time-consuming per-scan optimization and allowing real-time execution.*

## 1. Introduction

Point cloud data is becoming more ubiquitous than ever: anyone can create a point cloud from a set of photos with easy to use photogrammetry software or capture a point cloud directly with one of many consumer-grade depth sensors available worldwide. These sensors will soon be used in most aspects of our daily lives, with autonomous cars recording streets and city environments and VR and AR devices recording our home environment on a regular basis. The resulting data represents a great opportunity for computer vision research: it complements image data with depth information and opens up new fields of research.

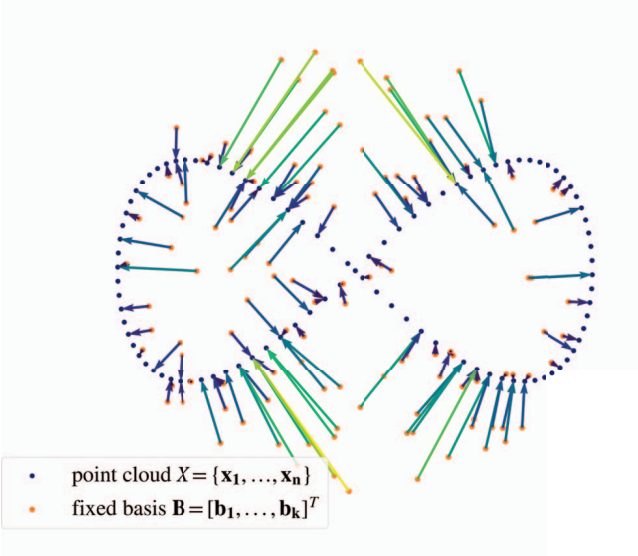


Figure 1. *Basis point set encoding for point clouds*. The encoding of a point cloud $X = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ is a fixed-length feature vector, computed as the minimal distances to a fixed set of points $\mathbf{B} = [\mathbf{b_1}, ..., \mathbf{b_k}]^T$. This representation can be used as input to arbitrary machine learning methods, in particular it can be used as input for off-the-shelf neural networks. This leads to substantial performance gains as compared to occupancy grid encoding or specialized neural network architectures without sacrificing the accuracy of predictions.

However, point cloud data itself is unstructured. This leads to a variety of problems: (a) point clouds have no fixed cardinality, varying their size depending on the recorded scene. They are also not 'registered' in the sense that it is not trivial to find correspondences between points across recordings of the same or of a similar scene. (b) Point

*work was done during internship at Amazon
†the last two authors were equally involved

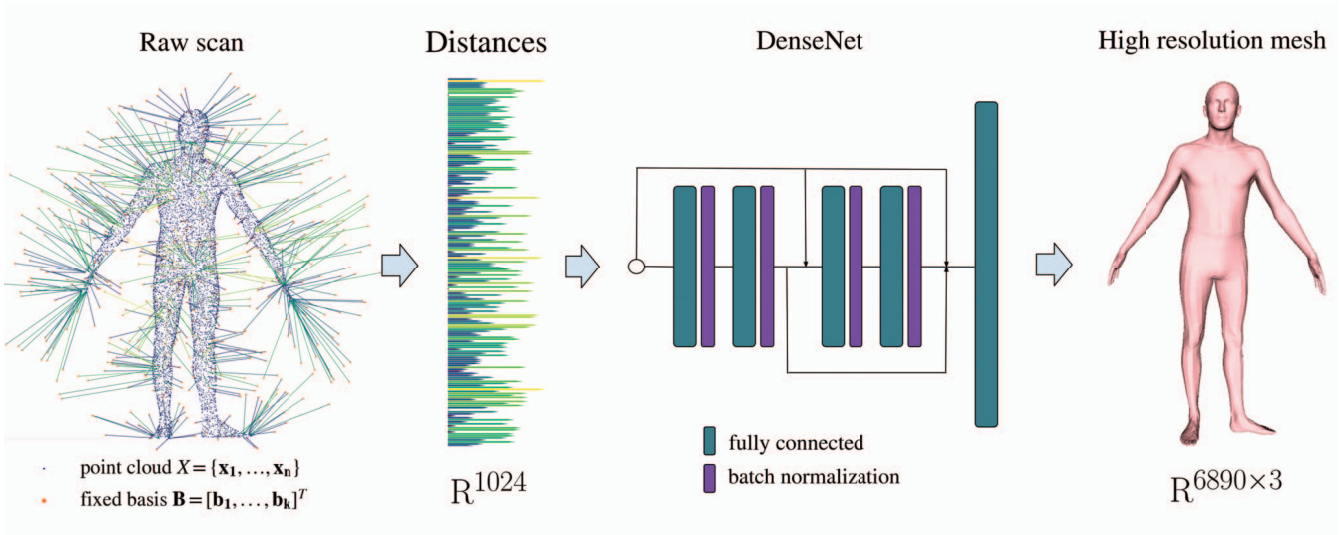


Figure 2. *Overview of our proposed model for the task of mesh registration to a noisy scan.* The computed minimal distances to the selected basis point set are provided as input to a simple dense network with two blocks of two fully connected layers. The model directly predicts mesh vertex positions, with a forward pass taking less than 1ms. We also propose a model for shape classification; see Sec. 5 for details.

clouds have no notion of neighborhood. This means that it is not clear how convolutions, one of the critical operations in deep learning, should be performed.

In this paper, we present a novel solution to the aforementioned problems, in particular the varying cloud cardinality. For an illustration, see Fig. 1. We propose to encode point clouds as minimal distances to a fixed set of points, which we refer to as *basis point set*. This representation is vastly more efficient than classic extensive occupancy grids: it reduces every point cloud to a relatively small fixed-length vector. The vector length can be adjusted to meet computational constraints for specific applications and represents a trade-off between fidelity of the encoding and computational efficiency. Compared to other encodings of point clouds, the proposed representation also has an advantage in being more efficient with the number of values needed to preserve high frequency information of surfaces.

Given its fixed length, the presented encoding can be used with most of the standard machine learning techniques. In this paper we apply mostly artificial neural networks to build models with it, due to their popularity and accuracy. In particular, we analyze the performance of the encoding in two applications: point cloud classification and mesh registration over noisy 3D scans (*c.f.*, Fig. 2).

For point cloud classification, we achieve the same accuracy on the ModelNet40 [46] shape classification benchmark as PointNet [34], while using an order of magnitude less parameters and three orders of magnitudes less floating point operations.To demonstrate the versatility of the encoding, we show how it can be used for the task of mesh registration. We use the encoded vectors as input to a neural network that directly predicts mesh vertex positions. While showing competitive performance to the state-of-the-art methods on the FAUST dataset [2], the main advantage of our method is the ability to produce an aligned high resolution mesh from a noisy scan in a single feed-forward pass. This can be executed in real time even on a non-GPU laptop computer, requiring no additional post-processing steps. We make our code for both presented tasks available, as well as a library for usage in other projects[1].

## 2. Related Work

In this section, we describe existing 3D data representations and models and put them in relation to the presented method. We focus on representations that are compatible with deep learning models, due to their high performance on a variety of 3D shape analysis tasks.

**Point clouds.** Numerous methods [34, 36, 41, 47, 25] were proposed that process 3D point clouds directly, amongst which the PointNet family of models gained the most popularity. This approach processes each point separately with a small neural network followed by an aggregation step with a pooling operation to reason about the whole point cloud. Similar pooling-based approaches for achieving feature invariance on general unordered sets were proposed in other works as well [47]. Other methods working directly on point clouds organize the data in kd-trees and other graphs [23, 12, 24]. These structures define a neighborhood and thus convolution operations can be applied. Vice versa, specific convolutional filters can be designed for sparse 3d data [44, 41].

[1] https://github.com/sergeyprokudin/bps

We borrow several ideas from these works, such as using kNN-methods for searching efficiently through local neighborhoods or achieving order invariance through the use of pooling operations over computed distances to basis points. However, we believe that the proposed encoding and model architectures offer two main advantages over existing point cloud networks: (a) higher computational efficiency and (b) conceptually simpler, easy-to-implement algorithms that do not rely on a specific network architecture or require custom neural network layers.

**Occupancy grids.** Similar to pixels for 2D images, occupancy grid is a natural way of encoding 3D information. Numerous deep models were proposed that work with occupancy grids as inputs [29, 35, 30]. However, the main disadvantage of this encoding is their cubic complexity. This results in a high amount of data needed to accurately represent the surface. Even relatively large grids by our current memory standards ($128^3, 256^3$) are not sufficient for an accurate representation of high frequency surfaces like human bodies. At the same time, this type of voxelization results in very sparse volumes when used to represent 3D surfaces: most of the volume measurements are zeros. This makes this representation an inefficient surface descriptor in multiple ways. A number of methods was proposed to overcome this problem [45, 37]. However, the problem of representing high frequency details remains, together with a large memory footprint and low computational efficiency for running convolutions.

**Signed distance fields.** Truncated signed distance fields (TSDFs) [8, 31, 38, 42, 48, 9, 33] can be viewed as a natural extension of occupancy grids: they store distance-to-surface information in grid cells instead of a simple occupancy flag. While this partially resolves the problem of representing surface information, the cubic requirement for memory and the low computational efficiency for convolutions remains. In comparison, our method can be viewed as one that uses an arbitrary subset of points from the distance field. The crucial difference is that the distance field we sample from is unsigned and non-truncated, and the number of samples is proportional to the number of points in the original cloud. We further investigate the connection between occupancy grids, TSDFs and BPS in Sec. 4.1.

**2D projections.** Another common strategy is to project 3D shapes to 2D surfaces and then apply standard frameworks for 2D input processing. This includes depth maps [46], height maps [40], as well as a variety of multi-view models [43, 22, 11]. Closely related are approaches that project 3D shapes into spheres and apply spherical convolutions to achieve rotational invariance [10, 6]. While projection-based approaches show high accuracy in discriminative tasks (classification, shape retrieval), they are fundamentally limited in representing shapes that have multiple 'folds', invisible from external views. In comparison, our encoding scheme can accurately preserve surface information of objects with arbitrary topology as we show in our experiments in Sec. 4.

We now describe the algorithm for constructing the proposed basis point representation from a given point cloud.

## 3. Method

**Normalization.** The presented encoding algorithm takes a set of point clouds as input $\mathbf{X} = \{X_i, i = 1, \ldots, p\}$. Every point cloud can have a different number of points $n_i$:

$$X_i = \{\mathbf{x}_{i1}, \ldots, \mathbf{x}_{in_i}\}, \mathbf{x}_{ij} \in \mathbb{R}^d, \tag{1}$$

where $d = 3$ for the case of 3D point clouds. In first step, we normalize all point clouds to a fit a unit ball:

$$\mathbf{x}_{ij} = \frac{\mathbf{x}_{ij} - \mathbb{E}_{\mathbf{x}_{ij} \sim X_i} \mathbf{x}_{ij}}{\max_{\mathbf{x}_{ij} \in X_i} \|\mathbf{x}_{ij} - \mathbb{E}_{\mathbf{x}_{ij} \sim X_i} \mathbf{x}_{ij}\|}, \forall i, j. \tag{2}$$

***BPS* construction.** Next, we form a *basis point set*. For this task, we sample $k$ random points from a ball of a given radius $r$:

$$\mathbf{B} = [\mathbf{b}_1, ..., \mathbf{b}_k]^T, \mathbf{b}_{j \in} \mathbb{R}^d, \|\mathbf{b}_j\| <= r. \tag{3}$$

It is important to mention that this set is arbitrary but fixed for all point clouds in the dataset. $r$ and $k$ are hyper-parameters of the method, and $k$ can be used to determine the trade-off between computational complexity and the fidelity of the representation.

**Feature calculation.** Next, we form a feature vector for every point cloud in a dataset by computing the minimal distance from every basis point to the nearest point in the point cloud under consideration:

$$\mathbf{x}_i^{\mathbf{B}} = [\min_{\mathbf{x}_{ij} \in X_i} \mathrm{d}\left(\mathbf{b}_1, \mathbf{x}_{ij}\right), \ldots, \min_{\mathbf{x}_{ij} \in X_i} \mathrm{d}\left(\mathbf{b}_k, \mathbf{x}_{ij}\right)]^T, \quad \mathbf{x}_i^{\mathbf{B}} \in \mathbb{R}^k. \tag{4}$$

Alternatively, it is possible to store the full directional information in the form of delta vectors from each basis point to the nearest point in the original point cloud:

$$\mathbf{X}_i^{\mathbf{B}} = \left\{ \left( \operatorname*{argmin}_{\mathbf{x}_{ij} \in X_i} \mathrm{d}\left(\mathbf{b}_q, \mathbf{x}_{ij}\right) - \mathbf{b}_q \right) \right\} \in \mathbb{R}^{k \times d}, \tag{5}$$

Other information about nearest points (*e.g.*, RGB values, surface normals) can be saved as part of this fixed representation. The feature computation is illustrated in Fig. 1. The formulas (4) and (5) give us fixed-length representations of the point clouds that can be readily used as input for learning algorithms.

**BPS selection strategies.** We investigate a number of basis point selection strategies and provide details of these experiments in Sec. 4.2. Overall, random sampling from a uniform distribution in the unit ball provides a good trade-off between efficiency, universality of the generation process and surface reconstruction results, and we apply it throughout the experiments in this paper. Alternatively, an extensive 3D grid of basis points could be used in tandem with any existing 3D convolutional neural network in order to achieve maximum performance at the cost of increased computational complexity.

**Complexity.** In this work, we use Euclidean distances between points for creating our encoding, but other metrics could be used in principle. Since we are working with 3D point clouds (which corresponds to having a small value for $d$), the nearest neighbor search can be made efficient by using data structures like ball trees [32]. Asymptotically, $O(n \log n)$ operations are needed for constructing a ball tree from the point cloud $X_i$ and $O(k \log n)$ operations are needed to run nearest neighbor queries for $k$ basis points. This leads to an overall encoding complexity of $O(n \log n + k \log n)$ per point cloud. The kNN search step can be also efficiently implemented as part of an end-to-end deep learning pipeline [21]. Practically, we benchmark our encoding scheme for different values of $n$ and $k$ and show real-time encoding performance for values interesting for current real world applications. Please refer to the supplementary materials for further details.

## 4. Analysis

### 4.1. Comparison to occupancy grids, TSDFs and plain point clouds

**Informal intuition.** Compared to occupancy grids and TSDFs, the efficiency and superiority of the proposed BPS encoding is based on two key observations. First, it is beneficial for both surface reconstruction and learning to *store some continuous global information* (*e.g.*, Euclidean distance to the nearest point) in every cell of the grid instead of simple binary flags or local distances. In the latter case, most of the voxels remain empty and, moreover, the feature vector will change dramatically when slight translations or rotations are applied to an object. In comparison, every BPS cell always stores some information about the encoded object and the feature vector changes smoothly with respect to affine transformations. From this also stems the second important observation: when every cell stores some global information, we can use a much smaller number of them in order to represent the shape accurately, thus *avoiding the cubical complexity of the extensive grid representation*. This can be seen in Fig. 1 and bottom right Fig. 3, where $k \approx n$ basis points are able to capture the outline of the original cloud.

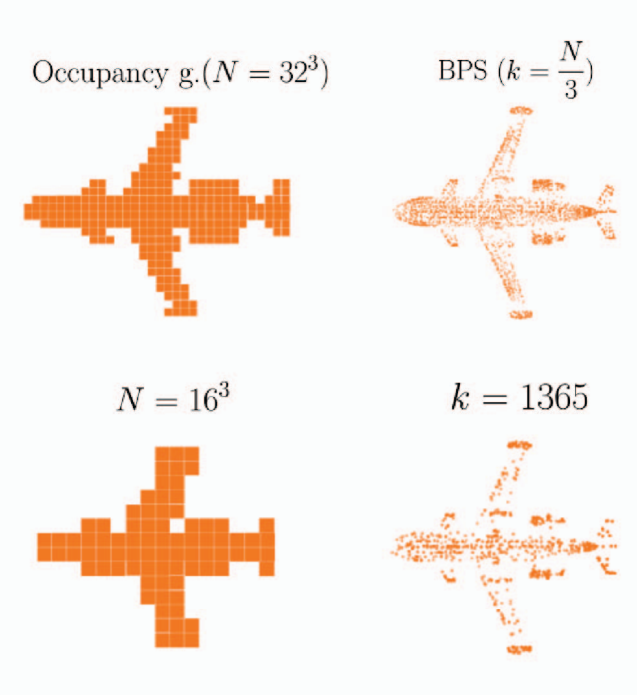


Figure 3. *Surface encoding with occupancy grids (left) and basis point sets (right).* With the same length of encoding $N$ our method can capture surface details more accurately. Even when using only $k \approx 10^3$ basis points, our method can capture details of a surface (bottom right).

We will now validate this intuition by comparing the aforementioned representations in terms of surface reconstruction and actual learning capabilities.

**Surface reconstruction experiments.** Independent of a certain point cloud at hand, how well does the encoding capture the details from the object? To answer this question, we take $10^3$ random CAD models from the ModelNet40 [46] dataset and construct synthetic point clouds by sampling $10^4$ points from each surface. We compare three approaches of encoding the resulting point clouds: storing them as is (raw point cloud), occupancy grid and the proposed encoding via basis point sets as suggested in Eq. 5.

For all methods we define a fixed allowed description length $N$ (as $N$ floating point values) and compare the normalized bidirectional Chamfer distance between the original point cloud $X$ and the reconstructed point cloud $X^r$ for the different encodings:

$$\mathrm{d}_{CD}(X, X^r) = \frac{1}{|X|} \sum_{\mathbf{x}_i \in X} \min_{\mathbf{x^r}_i \in X^r} ||\mathbf{x}_i - \mathbf{x^r}_i||^2 + \frac{1}{|X^r|} \sum_{\mathbf{x^r}_i \in X^r} \min_{\mathbf{x}_i \in X} ||\mathbf{x}_i - \mathbf{x^r}_i||^2. \quad (6)$$

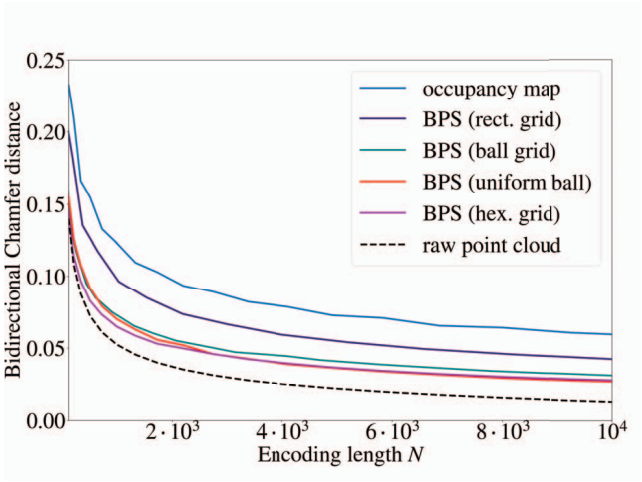


Figure 4. *Surface reconstruction quality vs. encoding length for different 3D data encoding methods.* We measured the Chamfer distance on $10^3$ encoded and reconstructed random shapes from the ModelNe40 dataset. The suggested representation is more accurate in representing surface details than standard occupancy grid. The performance of our best basis selection methods is close to encoding the surface with subsampled unordered point clouds while being a fixed-length representation that can be directly used with a wide range of machine learning algorithms. See Sec. 4 for further details.

With the same length of the description $N$ we can either store $N/3$ points from the original point cloud, $\sqrt[3]{N} \times \sqrt[3]{N} \times \sqrt[3]{N}$ binary occupancy flags or $N/3$ basis points with the matrix $\mathbf{X}_i^{\mathbf{B}}$ defined in Eq. 5. From this matrix, a subset of original points can be reconstructed by simply adding corresponding basis point coordinates to every delta vector. For the occupancy grid encoding, we use the centers of occupied grid cells; please note that though a full floating point representation is not necessary to store the binary flag, in reality the majority of machine learning methods will work with floating point encoded occupancy grids and we assume this representation.

Fig. 4 shows the encoding length and the reconstruction quality measured as Chamfer distance (*c.f.*, Eq. 6). The proposed encoding produces less than half of the encoding error compared to occupancy grids for point clouds up to roughly $10^4$ points (see Fig. 3 for a qualitative comparison). This is an indicator for its superiority for preserving shape information. The error curve for the basis point sets is close to the one of the subsampled point cloud representation. The basis point set representation is less accurate than the raw point cloud since the resulting extracted points are not necessarily unique. However, the basis point set is an ordered, fixed-length vector encoding well-suited to apply machine learning methods.

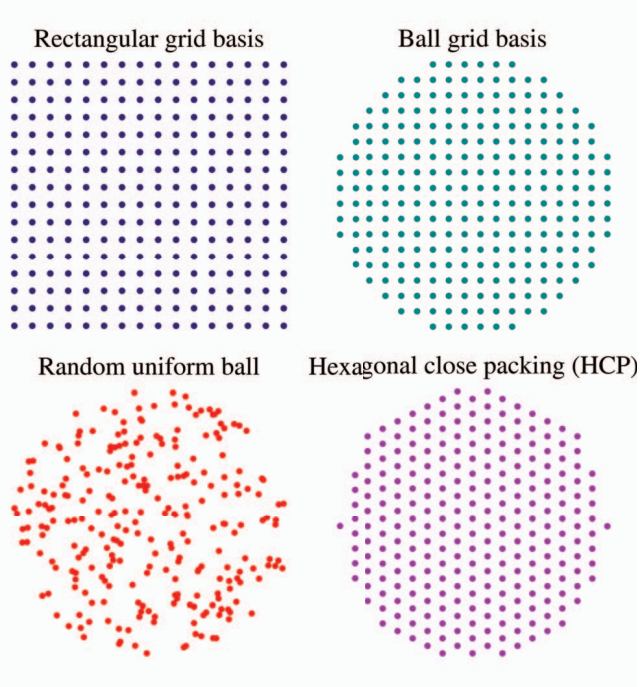


Figure 5. *Different basis point selection strategies.* See Sec. 4.2 for details. In this work, we mainly use random uniform ball sampling for its simplicity and efficiency, as well as rectangular grid basis that allows us to apply 3D convolutions in a straightforward manner. Different BPS arrangements allow the usage of different types of convolutions.

### 4.2. Basis point selection strategies

We investigate four different variants of selecting basis points visualized in Fig. 5.

**Rectangular grid basis.** A basic approach to basis set construction is to simply arrange points on a rectangular $[-1, 1]^3$ grid. In that case, the basis point set representation resembles the truncated signed distance field [8] representation. However, one important difference is that we do not truncate the distances for far-away basis points, allowing every point in the set to store some information about the object surface. We will show in Sec. 5.1 that this small conceptual difference has an important effect on performance. We are also allowing the full directional information to be stored in the cell as defined in Eq. 5. Finally, BPS does not require the point clouds to be converted into watertight surfaces since *unsigned* distances are used.

**Ball grid basis.** Since all point clouds are normalized to fit in the unit ball by the transformation defined in Eq. 2, the basis points at the corners of the rectangular grid are located far away from the point cloud. These corner points in fact constitute $47.6\%$ of all the samples (this can be derived by comparing the volume ratio of a unit ball to a unit cube).

| id | Method | acc. | FLOPs | params |
|---|---|---|---|---|
| 1 | VoxNet [29] | 83.0% | $> 10^8$ | $9.0 \times 10^5$ |
| 2 | Occ-MLP ($32^3$ grid) | $79.9\% \pm 0.3$ | $3.4 \times 10^7$ | $1.7 \times 10^7$ |
| 3 | Occ-MLP ($8^3$ grid) | $74.5\% \pm 0.2$ | $1.1 \times 10^6$ | $5.5 \times 10^5$ |
| 4 | TDF-MLP ($32^3$ grid) | $80.0\% \pm 0.3$ | $3.4 \times 10^7$ | $1.7 \times 10^7$ |
| 5 | TDF-MLP ($8^3$ grid) | $75.9\% \pm 0.3$ | $1.1 \times 10^6$ | $5.5 \times 10^5$ |
| 6 | BPS-MLP ($32^3$ grid) | $88.3\% \pm 0.2$ | $3.4 \times 10^7$ | $1.7 \times 10^7$ |
| 7 | BPS-MLP ($8^3$ grid) | $87.6\% \pm 0.3$ | $1.1 \times 10^6$ | $5.5 \times 10^5$ |
| 8 | BPS-MLP ($8^3$ ball) | $87.7\% \pm 0.3$ | $1.1 \times 10^6$ | $5.5 \times 10^5$ |
| 9 | BPS-MLP ($8^3$ rand) | $88.0\% \pm 0.3$ | $1.1 \times 10^6$ | $5.5 \times 10^5$ |
| 10 | BPS-MLP ($8^3$ HCP) | $88.1\% \pm 0.3$ | $1.1 \times 10^6$ | $5.5 \times 10^5$ |
| 11 | BPS-Conv3D ($32^3$ grid) | $89.8\% \pm 0.2$ | $3.5 \times 10^8$ | $1.7 \times 10^7$ |
| 12 | $9 \rightarrow$ direct. vect. | $86.2\% \pm 0.3$ | $2.2 \times 10^6$ | $1.1 \times 10^6$ |
| 13 | $11 \rightarrow$ direct. vect. | $90.8\% \pm 0.3$ | $3.8 \times 10^8$ | $1.7 \times 10^7$ |
| 14 | BPS-ERT [13] ($16^3$ g.) | $85.4\% \pm 0.2$ | N/A | N/A |
| 15 | BPS-XGBoost ($32^3$ g.) | $86.1\% \pm 0.1$ | N/A | N/A |

Table 1. *Comparison between occupancy grids, truncated distance fields (TDF) and BPS as input features for 3D shape classification on the ModelNet40 [46] challenge.* We keep the model architecture fixed across experiments. Global BPS encoding significantly outperforms its local counterparts. See Sec. 5.1 for further details.

Hence we can improve our sampling efficiency by simply trimming the corners of the grid and using more sampling locations within the unit ball.

**Random uniform ball sampling.** One generic simple strategy to select points lying inside a $d-$dimensional ball is uniform sampling. This can be done by either rejection sampling from a $d-$dimensional cube or other efficient methods that are summarized in [17].

**Hexagonal close packing (HCP).** We also experiment with *hexagonal close packing* [7] of basis points. Informal intuition behind this point selection strategy is that it will optimally cover the unit ball with equally sized balls centered at the basis points [16].

We show a comparison of reconstruction errors of $10^3$ ModelNet objects using the different sampling strategies in Fig. 4. Overall, the random uniform and HCP selection strategies provide the best reconstruction results. Using regular grids opens up possibilities for applying convolution operations and adds the possibility to learn translation and rotation invariant features.

We now evaluate the different encodings and basis point selection strategies with respect to their applicability with machine learning algorithms.

## 5. Learning with Basis Point Sets

### 5.1. 3D Shape Classification

One of the classic tasks to perform on point clouds is classification. We present results for this task on the *ModelNet40* [46] dataset. We benchmark several deep learning architectures that use the proposed point cloud representation and compare them to existing methods that use alternative encodings. The dataset consists of $12 \cdot 10^3$ CAD models from 40 different categories, of which $9.8 \cdot 10^3$ are used for training. We use the same procedure for obtaining point clouds from CAD models as in [34], *i.e.*, we sample $n = 2048$ points from mesh faces, followed by the normalization process defined in Eq. (2).

**Comparison to occupancy grids and VoxNet.** To show the superiority of BPS features and to disambiguate contributions (*i.e.*, the BPS encoding itself and the proposed network architectures), we fix a simple generic MLP architecture with 2 blocks of [fully-connected, relu, batchnorm, dropout] layers and perform training with $32^3$ rectangular grids of occupancy maps, truncated distance fields (TDFs) and BPS as inputs.

Results are summarized in Tab. 1, rows 1-7. Using global distances as features instead of occupancy flags with the same network clearly improves accuracy, outperforming an architecture that was specifically designed for processing this type of input: VoxNet [29] (row 1). TDFs store only local distances within the grid cell and suffer from the same locality problem as voxels (r. 4). It is also important to note that reducing the grid size affects these methods dramatically (rows 3 and 5, 5% drop in accuracy), while the effect on the BPS is marginal (r. 6, $-0.7\%$).

We also compare different BPS selection strategies in the rows 7-10 of Tab. 1. In the absence of network operators exploiting the point ordering (e.g. 3D convolutions), random and HCP strategies give a slight boost in performance. When the point order in a rectangular BPS grid is exploited with 3D convolutional deep learning models like VoxNet, performance improves at the cost of increased computational complexity (approximately two orders of magnitude more flops, Tab. 1, r. 11).

Substituting Euclidean distances with full directional information defined by Eq. 5 negatively affects the performance of a plain fully-connected network (Tab. 1, r.12) whereas it improves the performance of a 3D convolutional model (Tab. 1, r. 13).

To show the versatility of the proposed representation, we also use the same BPS features as input to an ensemble of extremely randomized trees (ERT [13]) and XGBoost [5] frameworks.

**Comparison to other methods.** Finally, we combine these findings with other enhancements (*e.g.*, augmenting the data with few fixed rotations, improving learning schedule and regularization - please refer to the supplementary material and corresponding repository for further details) and compare our two best-performing models to other methods in Tab. 2.

| Method | acc. | FLOPs | params |
|---|---|---|---|
| RotationNet 20x [22] | **97.37%** | $>10^9$ | $5.8 \times 10^7$ |
| MVCNN 80x [43] | 90.1% | $6.2 \times 10^{10}$ | $9.9 \times 10^7$ |
| VoxNet [29] | 83.0% | $>10^8$ | $9.0 \times 10^5$ |
| Spherical CNNs [10] | 88.9% | $2.9 \times 10^7$ | $5.0 \times 10^5$ |
| *point cloud based methods:* | | | |
| KD-networks [23] | **91.8%** | $>10^9$ | $>10^7$ |
| KCNet [41] | 91.0% | $>10^8$ | $9.0 \times 10^5$ |
| SO-Net [25] | 90.9% | $>10^8$ | $>10^6$ |
| DeepSets [47] | 90.0% | $1.5 \times 10^9$ | $2.1 \times 10^5$ |
| PointNet++ [36] | 90.7% | $1.6 \times 10^9$ | $1.7 \times 10^6$ |
| PointNet [34] | 89.3% | $4.4 \times 10^8$ | $3.5 \times 10^6$ |
| PointNet(vanilla) [34] | 87.2% | $1.4 \times 10^8$ | $8.0 \times 10^5$ |
| DeepSets (micro) [47] | 82.0% | $3.8 \times 10^7$ | $2.1 \times 10^5$ |
| Ours (BPS-MLP) | 89.0% | $\mathbf{7.6 \times 10^5}$ | $3.8 \times 10^5$ |
| Ours (BPS-Conv3D) | 90.8% | $3.5 \times 10^8$ | $4.4 \times 10^6$ |
| Ours (BPS-Conv3D, 10x) | 91.6% | $3.5 \times 10^9$ | $4.4 \times 10^7$ |

Table 2. *Results on the ModelNet40 [46] 3D shape classification challenge.* Simple fully connected network can be trained on BPS features in several minutes on a single GPU to reach the performance of PointNet.

In summary, simple fully connected network, trained on BPS features in several minutes on a single GPU, is reaching the performance of PointNet [34], one of the most widely used networks for point cloud analysis. 3D-convolutional model trained on BPS rectangular grid is matching the performance of the PointNet++[36], while still being computationally more efficient. Finally, crude ensembling of 10 such models allows us to match state-of-the-art performance [23] among methods working only on point clouds as inputs (*e.g.*, without using surface normals that are available in CAD models but rarely in real-world scenarios).

### 5.2. Single-Pass Mesh Registration from 3D Scans

We showcase a second experiment with a different, generative task to demonstrate the versatility and performance of the encoding. For this, we pick the challenging problem of human point cloud registration. In this problem, correspondences are found between an observed, unstructured point cloud and a deformable body template. Traditionally, human point cloud registration has been approached with iterative methods [18, 49]. However, they are typically computationally expensive and require the use of a deformable model at application time. Machine learning based methods [15] remove this dependency by replacing them with a sufficiently large training corpus. However, current solutions like [15] rely on multistage models with complex internal representations, which makes them slow to train and test. We encourage the reader interested in human mesh registration to review the excellent summary of previous work provided in [15].

| Method | Intra (mms) | Inter (mms) |
|---|---|---|
| Stitched puppets [49] | 1.568 | 3.126 |
| 3D-CODED [15] | 1.985 | 2.878 |
| Ours | 2.327 | 4.529 |
| Deep functional maps [26] | 2.436 | 4.826 |
| FARM [28] | 2.81 | 4.123 |
| Convex-Opt [4] | 4.86 | 8.304 |

Table 3. *Results for all published methods in the intra and inter challenge for the FAUST dataset, sorted by error in the intra challenge.* Our BPS-based network has a performance comparable to other methods while allowing single pass, real-time mesh registration, with no per-scan optimizations.

We use a simple DenseNet-like [20] architecture with two blocks (see Figure 2), where the input is a BPS encoding of a point cloud and the output is the *location of each vertex* in the common template. Note that there is no deformable model in our system and that we do not estimate deformable model parameters or displacements; the networks learns to reproduce coherent bodies just based on its training data.

To generate this training data, we use the SMPL body model [27]. SMPL is a reshapeable, reposable model that takes as input pose parameters related to posture, and shape parameters related to the intrinsic characteristics of the underlying body (*e.g.*, height, weights, arm length). We sample shape parameters from the CAESAR [39] dataset, which contains a wide variety of ages, body constitution and ethnicities. For sampling poses we use two sources: the CMU dataset [1] and a small set of poses inferred from a 3D scanner . Since the CMU dataset is heavily populated with walking and running sequences, we perform weighted sampling of poses with the inverse Mahalanobis distance from the sample to the CMU distribution as weight. We roughly align the CMU poses to be frontal. To increase the variation of the training data, we introduce noise sampled from the covariance of all the considered poses to half of the data points. From these meshes, a set of $10^4$ points is sampled uniformly from the surface of the posed and shaped SMPL template. These point clouds are then used to compute the BPS encoding. We train the alignment network for 1000 epochs in only 4 hours and its inference time is less than 1ms on a non-GPU laptop.

To evaluate our method, we process the test set from the FAUST [2] dataset. It is used to compare mesh correspondence algorithms by using a list of scan points in correspondence. To find correspondences between two point clouds, we process each of them with our network, obtaining as a result two registered mesh templates. The templates then define the dense correspondences between the point clouds.

Figure 6. *Point clouds of the FAUST dataset and the predicted meshes.* **Blue:** point cloud from a 3D scanner. **Skin color:** predicted mesh by our model through processing of its BPS representation. Note that the network produces the position of each output vertex; their coherent structure is learned solely from the training data.

We obtain an average performance of 2.327mm in the intra-subject challenge and 4.529mm in the inter-subject challenge (see Tab. 3). These numbers are comparable, but higher than state-of-the-art methods like [15] or [49]. However, we note that the two methods outperforming BPS in the FAUST intra challenge are orders of magnitude slower than our system. The two-stage procedure in [15] takes multiple minutes and the particle optimization in [49] takes hours, while our system produces alignments in 1ms (for qualitative results, see Fig. 6). This enables real-time processing of 3D scans, which was previously impossible, or can be used as a first step for faster multistage systems that refine the accuracy of this single stage method. We also provide a qualitative evaluation on the Dynamic FAUST[3] dataset in the supplementary video [2].

[2] https://youtu.be/kc9wRoI5JbY

## 6. Conclusion and Future Work

In this paper, we introduced *basis point sets* for obtaining a compact fixed-length represenation of point clouds. BPS computation can be used as a pre-processing step for a variety of machine learning models. In our experiments, we demonstrated in two applications and with different models the computational superiority of our approach with orders of magnitudes advantage in processing time compared to existing methods, remaining competitive accuracy-wise. We have shown the advantage of using rectangular BPS grid in combination with standard 3D-convolutional networks. However, in future work it would be interesting to consider other types of BPS arrangements and corresponding convolutions [19, 6, 10, 14] for improved efficiency and learning rotation-invariant representations.

## References


[1] http://mocap.cs.cmu.edu. 7

[2] Federica Bogo, Javier Romero, Matthew Loper, and Michael J Black. Faust: Dataset and evaluation for 3d mesh registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3794–3801, 2014. 2, 7

[3] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: Registering human bodies in motion. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017*, Piscataway, NJ, USA, July 2017. IEEE. 8

[4] Qifeng Chen and Vladlen Koltun. Robust nonrigid registration by convex optimization. In *ICCV*. IEEE Computer Society, 2015. 7

[5] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016. 6

[6] Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. *arXiv preprint arXiv:1801.10130*, 2018. 3, 8

[7] John Horton Conway and Neil James Alexander Sloane. *Sphere packings, lattices and groups*, volume 290. Springer Science & Business Media, 2013. 6

[8] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. 1996. 3, 5

[9] Angela Dai and Matthias Nießner. Scan2mesh: From unstructured range scans to 3d meshes. *arXiv preprint arXiv:1811.10464*, 2018. 3

[10] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning so (3) equivariant representations with spherical cnns. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–68, 2018. 3, 7, 8

[11] Yifan Feng, Zizhao Zhang, Xibin Zhao, Rongrong Ji, and Yue Gao. Gvcnn: Group-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 264–272, 2018. 3

[12] Matheus Gadelha, Subhransu Maji, and Rui Wang. Shape generation using spatially partitioned point clouds. *arXiv preprint arXiv:1707.06267*, 2017. 2

[13] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006. 6

[14] Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017. 8

[15] Thibault Groueix, Matthew Fisher, Vladimir Kim, Bryan Russell, and Mathieu Aubry. 3d-coded : 3d correspondences by deep deformation. Sept. 07 2018. 7, 8

[16] Thomas C Hales. A proof of the kepler conjecture. *Annals of mathematics*, pages 1065–1185, 2005. 6

[17] Radoslav Harman and Vladimír Lacko. On decompositional algorithms for uniform sampling from n-spheres and n-balls. *Journal of Multivariate Analysis*, 101(10):2297–2304, 2010. 6

[18] David A Hirshberg, Matthew Loper, Eric Rachlin, and Michael J Black. Coregistration: Simultaneous alignment and modeling of articulated 3d shape. In *European Conference on Computer Vision*, pages 242–255. Springer, 2012. 7

[19] Emiel Hoogeboom, Jorn WT Peters, Taco S Cohen, and Max Welling. Hexaconv. *arXiv preprint arXiv:1803.02108*, 2018. 8

[20] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 7

[21] Łukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. Learning to remember rare events. *arXiv preprint arXiv:1703.03129*, 2017. 4

[22] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5010–5019, 2018. 3, 7

[23] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 863–872, 2017. 2, 7

[24] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018. 2

[25] Jiaxin Li, Ben M Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9397–9406, 2018. 2, 7

[26] Or Litany, Tal Remez, Emanuele Rodolà, Alexander M. Bronstein, and Michael M. Bronstein. Deep functional maps: Structured prediction for dense shape correspondence. *CoRR*, abs/1704.08686, 2017. 7

[27] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):248, 2015. 7

[28] Riccardo Marin, Simone Melzi, Emanuele Rodolà, and Umberto Castellani. Farm: Functional automatic registration method for 3d human bodies. *CoRR*, abs/1807.10517, 2018. 7

[29] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015. 3, 6, 7

[30] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5079–5088, 2018. 3

[31] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136. IEEE, 2011. 3

[32] Stephen M Omohundro. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989. 4

[33] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *arXiv preprint arXiv:1901.05103*, 2019. 3

[34] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017. 2, 6, 7

[35] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016. 3

[36] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017. 2, 7

[37] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3577–3586, 2017. 3

[38] Gernot Riegler, Ali Osman Ulusoy, Horst Bischof, and Andreas Geiger. Octnetfusion: Learning depth fusion from data. In *2017 International Conference on 3D Vision (3DV)*, pages 57–66. IEEE, 2017. 3

[39] Kathleen M. Robinette, Sherri Blackwell, Hein Daanen, Mark Boehmer, Scott Fleming, Tina Brill, Dãvid Hoeferlin, and Dennis Burnsides. Civilian American and European Surface Anthropometric Resource (CAESAR) final report. Technical report, US Air Force Laboratory, 2002. 7

[40] Kripasindhu Sarkar, Basavaraj Hampiholi, Kiran Varanasi, and Didier Stricker. Learning 3d shapes as multi-layered height-maps using 2d convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 71–86, 2018. 3

[41] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4548–4557, 2018. 2, 7

[42] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1746–1754, 2017. 3

[43] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015. 3, 7

[44] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018. 2

[45] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG)*, 36(4):72, 2017. 3

[46] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 2, 3, 4, 6, 7

[47] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017. 2, 7

[48] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1802–1811, 2017. 3

[49] Silvia Zuffi and Michael J Black. The stitched puppet: A graphical model of 3d human shape and pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3537–3546, 2015. 7, 8