# SplineNet: B-spline neural network for efficient classification of 3D data

by

Sai Sagar Jinka, Avinash Sharma

in

*11th Indian Conference on Computer Vision, Graphics and Image Processing*
(*ICVGIP-2018*)

Hyderabad, India

Report No: IIIT/TR/2018/-1

Centre for Visual Information Technology
International Institute of Information Technology
Hyderabad - 500 032, INDIA
December 2018

# SplineNet: B-spline neural network for efficient classification of 3D data

Sai Sagar Jinka
Center for Visual Information Technology (CVIT),
Kohli Center for Intelligent Systems (KCIS),
IIIT Hyderabad
India
jinka.sagar@research.iiit.ac.in

Avinash Sharma
Center for Visual Information Technology (CVIT),
Kohli Center for Intelligent Systems (KCIS),
IIIT Hyderabad
India
asharma@iiit.ac.in

## ABSTRACT

Advancement in the field of 3D capture, owing to use of consumer depth sensors, has reinvigorated the research interest for scalable shape classification and recognition algorithms. Majority of recent deep learning pipelines for 3D shapes uses volumetric representation, extending the concept of 2D convolution to 3D domain. Nevertheless, the volumetric representation poses a serious computational disadvantage as most of the voxel grids are empty and results in redundant computation. Moreover, a 3D shape is determined by its surface and hence performing convolutions on the voxels inside the shape is sheer wastage of computation.

In this paper, we focus on constructing a novel, fast and robust characterization of 3D shapes that accounts for local geometric variations as well as global structure. We built up on the learning scheme of [19] by introducing sets of B-spline surfaces instead of point filters, in order to sense complex geometrical structures (large curvature variations). The locations of these surfaces are initialized over the voxel space and are learned during training phase. We propose SplineNet, a deep network consisting of B-spline surfaces for classification of input 3D data represented in volumetric grid. We derive analytical solutions for updates of B-spline surfaces during back propagation. We show results on publicly available dataset and achieve superior performance as compared to state-of-the-art method.

## CCS CONCEPTS

• **Computing methodologies → Computer vision**.

**ACM Reference Format:**
Sai Sagar Jinka and Avinash Sharma. 2018. SplineNet: B-spline neural network for efficient classification of 3D data. In *Proceedings of 11th Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP'18),* Anoop M. Namboodiri, Vineeth Balasubramanian, Amit Roy-Chowdhury, and Guido Gerig (Eds.). ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.475/978_1


*ICVGIP'18, Dec. 2018, Hyderabad, India*

ACM ISBN 978-1-4503-6615-1.
https://doi.org/10.475/978_1

## 1 INTRODUCTION

3D shape acquisition and analysis is an active research area in both computer vision and graphics. Advancement in the field of 3D capture, owing to use of consumer depth sensors, has reinvigorated the research interest for scalable shape classification and recognition algorithms. Recently, deep neural networks have emerged as key learning framework for various computer vision tasks [18]. Majority of existing works on deep learning on 3D data are proposed in volumetric representation where shapes are represented as occupancy grid which is analogous to pixels in the image, thereby directly extending concept of 2D convolution to 3D domain [2, 5, 24, 36].

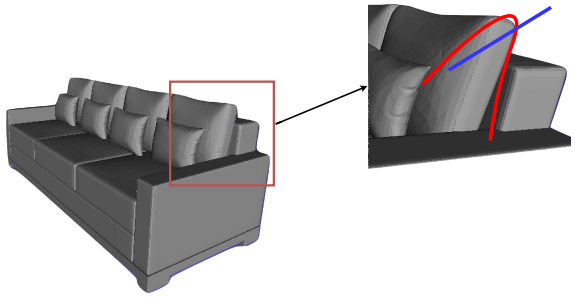

**Figure 1: In case of planar surfaces, sampling anywhere on the surface results in the same vector field. To capture non-planar surfaces, estimating field value along the red curve captures the local variations. Points on the blue line will not capture the local topology**

Nevertheless, the volumetric representation poses a serious computational disadvantage as most of the voxel grids are empty and results in redundant computation. Moreover, a 3D shape is determined by its surface and hence performing convolutions on the voxels inside the shape is sheer wastage of computation. This issue has been recently addressed in [19] by introducing field probing filters which effectively sense informative locations in the 3D space. This enables intelligent and sparse sampling in the grid space. However, the filters proposed in [19] are point-based, which evaluate functional value at a given point without accounting for geometrical information over the neighborhood. Hence this approach captures only global representation of voxelized 3D data for shape classification. As shown in Figure 1, the object has regions with flat as well as significantly varying curvature. In case of flat structures, sampling anywhere for estimating functional value (e.g., distance

transform) is acceptable. However, for regions with complex geometrical variations, point based sampling may not be sufficient. Hence, we introduce higher order *B*-spline surfaces to capture complex geometrical variations in the data.

In this paper, we focus on constructing a novel, fast and robust characterization of 3D shapes that accounts for local information as well as global geometry. We built up on the learning scheme of [19] by introducing sets of B-spline surfaces instead of point filters, in order to sense complex geometrical structures (large curvature variations). The locations of these surfaces are initialized randomly over the voxel space and are learned over training phase. We modify the dot product layer of [19] to aggregate local sensing and provide the global characterization of the input data. Our key contribution are listed as follows.

- We propose SplineNet, a deep network consist of B-spline surfaces for classification of input 3D data represented in volumetric fields. To the best of our knowledge, parametric curves and surfaces are not proposed in a learning setup in deep neural network for classification applications.
- We derive analytical solutions for updates of B-spline surfaces during back propagation.
- Our algorithm generates local-geometry aware global characterization of 3D shape using neural network.
- We show results on ModelNet[37] dataset and achieve superior performance to state-of-the-art method.

The remainder of the paper is organized as follows. In section 2, we present brief survey of the most relevant works on rigid 3D shape analysis, in particular, classification. Section 3 provide background details of B-spline curves and surfaces. Subsequently, in section 4 we outline our proposed B-spline neural network followed by details of experiments and results in Section 4.3.

## 2 RELATED WORK

Here we present solutions for 3D shape analysis using traditional hand-crafted features and recent learning representations from data via deep neural networks.

### 2.1 Shape descriptors

3D feature description using global and local analysis has drawn its inspiration from 2D images algorithms where features are represented using the sparse or dense set of local feature , e.g. SIFT [21]. The existing local feature descriptors are broadly categorized into extrinsic and intrinsic based on how they evaluate local geometry around a feature point. Extrinsic descriptors capture the local Euclidean geometry. Surface normals is one such descriptor used in many applications including 3D shape reconstruction, plane extraction, and point set registration [8, 12]. *Point descriptors* [9, 38] encode local features on the surface mesh by defining relative local surface normal at a sample point with respect to a superimposed plane or line segment at the sample points. Local surface normal vectors are computed at discrete points on the surface mesh to capture the local surface features in [11]. Other popular extrinsic descriptors are [13, 16]. Intrinsic descriptors capture pose invariant intrinsic geometry of the underlying manifold. However, these descriptors are confined to articulated 3D shapes. Another class of local shape descriptors are *ring-based* [25, 26] which are based on local sampling of a predefined metric over the discrete 3D surface mesh.

Global shape descriptors are quite popular for shape retrieval tasks where a single representation is used for shape retrieval. The Laplacian-Beltrami operator [29] is proposed to compute the diffusion-based shape descriptors. Heat Kernel Signature(HKS) uses eigen spectrum of the Laplacian operator to extract intrinsic properties by evaluating heat distribution on vertices of a mesh. The Wave Kernel Signature (WKS) [3] is another popular category of global descriptors that employ principles of quantum mechanics instead of heat diffusion on eigen spectrum to characterize the shape. Similarly, [32] proposed to characterize global representation of a shape which is robust against isometric deformations. This is achieved by computing the geodesic distances between sample points on the 3D surface mesh.

### 2.2 Deep learning on 3D data

Majority of deep learning works on 3D data are based on the idea of partitioning the 3D space into regular voxels and extending 2D CNNs to voxels. A deep belief network is trained for classification of ModelNet dataset in [37]. Voxel based variational auto-encoder is trained in [6] for shape modeling and object classification tasks. 3D object is recognized[30] by predicting the pose of the object in addition to the class label as a parallel task. However, these methods cannot be scaled to high resolutions due to inherent increase in computational complexity. The issue has been addressed by [19] defining field probing scheme. Nevertheless, only global representation of the object is learned.

Extensive literature on 2D CNNs has prompted many works to render images of a 3D object and use these images for feature description through CNNs. Each 3D shape is converted into a panoramic view and recognized in [31]. Information from multiple views of the object [33] is combined through novel view-pooling. [14] is also proposed on similar lines where they treat viewpoints as latent variables. 3D object is generated from a single 2D image [35] by generating images of surface normals, depth from various camera view points.

Other section of works operate directly on point cloud. PointNet [27] is a pioneering work in this direction that is proposed for raw point cloud as input and generate a permutation-invariant representation of the object. PointNet++ [28] proposed to use hierarchical neural network where PointNet is applied recursively on a nested partitioning of input point set. This approach addresses the drawback of local structure sensing of PointNet. A new architecture is proposed in [15] performs multiplicative transformations and shares parameters of these transformations according to the subdivisions of the point clouds imposed onto them by kdtrees. Similarly, OCNN [34] is proposed which is built upon the octree representation of 3D shapes. Challenges in unsupervised learning on point clouds is addressed by [39] by training an auto-encoder where decoder deforms a canonical 2D grid onto the underlying 3D object surface of a point cloud.

Graph-based approaches characterizes point clouds as graphs. A 3D point cloud can be represented as a polygon mesh or connectivity graph which is converted to the spectral representation

and apply convolution in spectral domain employing analogy between the classical Fourier transforms and projections onto the eigen basis of the graph Laplacian operator [7]. Recurrent Chebyshev polynomials to circumvent the problem of computation of the Laplacian eigenvectors is proposed in [10]. The first work in this approach is Geodesic CNN [22] where local patches represented in geodesic polar coordinates were applied with filters. Anisotropic heat kernels were used as an alternative way of extracting intrinsic patches on manifolds [4]. [23] provides a good overview of recent advancement in the field of graph deep learning for non-Euclidean data.

## 3 BACKGROUND

In this section, we discuss various properties of B-spline surfaces that are exploited for efficient feature representation using neural network. Parametric curves and surfaces are most commonly used in computer graphics for generation of 3D objects. A parametric surface in 3D is defined by three bivariate functions as

$$\alpha(u,v) = (\alpha_x(u,v), \alpha_y(u,v), \alpha_z(u,v)) \tag{1}$$

B-spline surfaces share similar properties to that of B-spline curves. Since curves are easier to visualize, we discuss about B-spline curve, a parametric polynomial curve which is defined as

$$\alpha(u) = \sum_{i=0}^{n} N_{i,k}(u) x_i \tag{2}$$

$$0 \leq u \leq n - k + 2$$

where $k$ is the order of the curve and we have $n+1$ control points and $N_{i,k}$ are termed as *SplineBasis* (refer to A.1.1 for calculating spline basis). The curve/surface is obtained by the blending of its control points and the blending functions are provided by spline basis. The most important properties of B-spline curves and surfaces include:

- The curve can be defined using arbitrarily large number of points without increasing the degree of the curve.
- The curve is a piecewise curve with each component a curve of degree $k-1$.
- A B-spline curve is enclosed in the convex hull of its control polyline. Specifically, if $u$ is in knot span $[u_i, u_{i+1})$, then $\alpha(u)$ is in the convex hull of control points $x_{i-k}, x_{i-k+1}, ..., x_i$.
- The continuity of the curve/surface is $k-2$ in each parametric dimension and hence is differentiable and derivatives can be computed analytically.
- Local Modification: By changing the position of control point $x_i$, only affects the curve $\alpha(u)$ on interval $[u_i, u_{i+k+1})$ as shown in Figure 2. This property is primarily exploited in our method for the calculation of local surface information.

## 4 OUR METHOD

We design a novel learning scheme for classification of rigid 3D objects which is learned using deep neural network. Figure 3 outlines the architecture of the proposed SplineNet. The input to our network is a 3D distance field or any differentiable vector field. Subsequently, we process this vector field input with novel SplinePatch layer for capturing local geometric variations. Later on, we pass the output of this layer to either Gaussian layer or directly to LocalAggregration layer. As explained in [19], Gaussian layer ensures that function values around object surface are retained. LocalAggregation layer accumulates sensing done by spline surfaces to recover the global characterization of object structure. Finally, the output is fed to a FC (Fully Connected) layer to be able to learn and predict final classifications labels.

### 4.1 SplinePatch Layer

This layer is accountable to sense the local information and pass it to LocalAggregation layer. It consists of $N$ sets of surfaces. Each set is initialized with a line in the 3D grid space as described in Section 5.1. On each line, we randomly sample $P$ points. At every sampled point, a B-spline patch is initialized with $m+1(n+1)$ control points along $U(V)$ directions. Essentially, this layers contains $N \times P$ patches. Each control point is three dimensional and the total number of parameters in this layer are $N \times P \times ((m+1) \times (n+1)) \times 3$.

Each point on the surface is expressed as

$$\alpha(u,v)(x) = \sum_{i=0}^{m+1} \sum_{j=0}^{n+1} N_{i,k}(u) * N_{j,l}(v) * x_{i,j} \tag{3}$$

$$0 \leq u \leq m - k + 2$$

$$0 \leq v \leq n - l + 2$$

The parametric space of $U, V$ is divided into $D = M \times N$ divisions i.e. if 3 and 2 divisions, the parametric space is $(0 - 0.33; 0.33 - 0.66; 0.66 - 1.0)$ and $(0 - 0.5; 0.5 - 1)$ along U and V directions respectively. It is illustrated on a spline curve for better visualization in Appendix A.2. In each division, we randomly sample $s$ sampling points. We evaluate the differentiable functional value at these sampling points $f(\alpha(u,v))$, for instance, distance transform. Notice that the functional value at sampled points depend on the control points $x_{i,j}$. Hence, during back-propagation, the functional value affects the location of control points. The gradient allows these locations to drift for effective sensing. The analytical solution for updates is discussed in back propagation section.

These sampling points provide local sensing. We introduce min-pooling in each division for distance transform function which is forwarded to LocalAggregation layer. Since the sensing is done in all divisions of a patch, the network tries to approximate the surface over the region.

$$f(\alpha(u,v))_{d_i,p,n} = min(f(\alpha(u_s, v_s))) \forall s \in d_i \tag{4}$$

where $d_i$ is $i^{th}$ division of $p^{th}$ patch in $n^{th} set$.

### 4.2 LocalAggregation Layer

The LocalAggregation layer attempts to perform a two-level aggregation. Firstly, this layer aggregates the functional values sensed from each of the divisions of a patch (output of SplinePatch layer). This operation helps the network to analyze the local geometry. The input to this layer is the output of SplinePatch/Gaussian layer which is of dimension $N \times P \times D \times C$ where $C$ is the number of channels in the function and concatenates the local information of all divisions around each patch. Essentially, the feature of a patch is D dimensional. We have also used various other operations such as average of the functional values of all the divisions. However, concatenation results in better performance.

Second level aggregation attempts to generate global characterization which is obtained by performing dot product operation

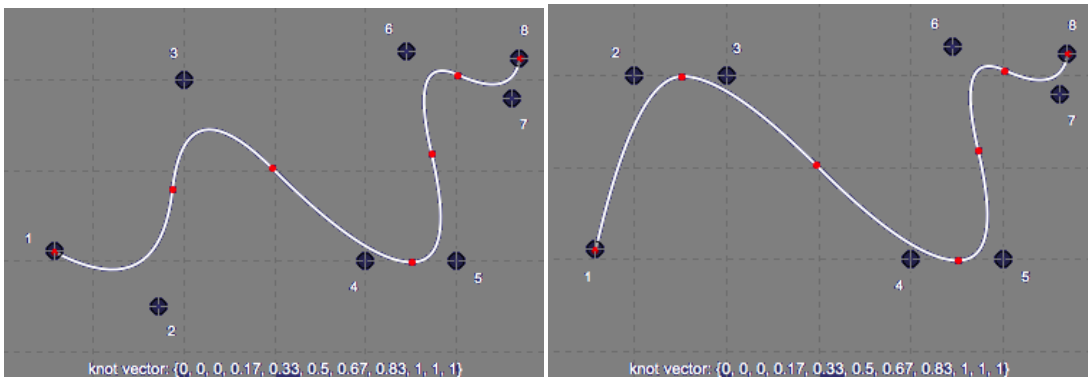


**Figure 2: The curve at left is deformed at position 2. The resulting curve at right is deformed around 2nd position**

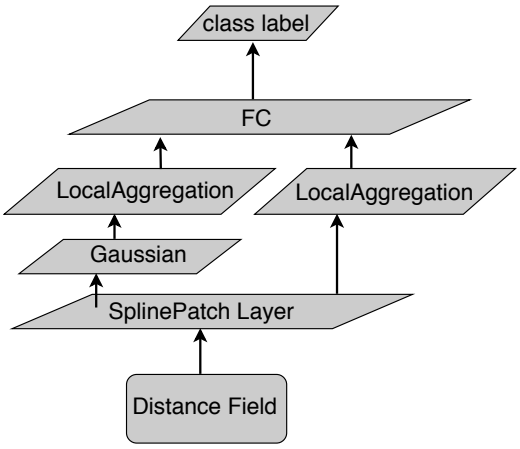


**Figure 3: Overview of our SplineNet architecture. Input shapes represented in volumetric fields are fed to SplinePatch layer for effective local sensing which is then optionally passed to Gaussian layer to retain values near surface boundaries. LocalAggregation layer accumulates local sensing to give local geometry aware global characterization of input shapes. Resulting characterization is fed to Fully Connected(FC) layers from which class label is predicted.**

across the set.

$$f_{net,n} = \sum_{p=0}^{P} \sum_{d_i=0}^{D} \sum_{c=0}^{C} f(\alpha(u,v))_{d_i,p,c} \times \beta_{d_i,p,c} \tag{5}$$

where $f_{net}$ is the net global and local contribution of the $n^{th}$ set and $\beta_{d_i,p,c}$ is the weight of $d_i^{th}$ division of $p^{th}$ patch and $c^{th}$ channel. The output of this layer is of dimension $N$ and is connected to Fully connected (FC) layers. The final FC layer is connected loss layer to predict the label of the input shape.

### 4.3 Back-Propagation

For training SplineNet trainable, it is necessary to compute gradients with respect to update the location of control points and the weights to perform dot product operation in the LocalAggregation layer. Let $E$ be the error function. Analytical solutions can be derived as

$$\begin{aligned} \frac{\partial E}{\partial f_{net,n}} &= \begin{bmatrix} \frac{\partial E}{\partial f(\alpha(u,v))_{d_i,p,c}} \\ \frac{\partial E}{\partial \beta_{d_i,p,c}} \end{bmatrix} \\ &= \begin{bmatrix} \beta_{d_i,p,c} \\ f(\alpha(u,v))_{d_i,p,c} \end{bmatrix} \end{aligned} \tag{6}$$

The update of control points can be derived as

$$\begin{aligned} \frac{\partial E}{\partial x_{i,j}} = \sum_{d_i=0}^{D} \sum_{c=0}^{C} \Bigg( & \frac{\partial E}{\partial f(\alpha(u_m,v_m))_{d_i,p,c}} * \frac{\partial f(\alpha(u_m,v_m))_{d_i,p,c}}{\partial \alpha(u_m,v_m)_{d_i,,p,c}} \\ & * \frac{\partial \alpha(u_{min},v_m)_{d_i,p,c}}{\partial x_{i,j}} \Bigg) \end{aligned}$$

$$= \sum_{d_i=0}^{D} \sum_{c=0}^{C} (\beta_{d_i,p,c} * f^{'}(\alpha(u_m,v_m))_{d_i,p,c}) * N_{i,k}(u_m) * N_{j,l}(v_m) \tag{7}$$

From Eq. 3,

$$\frac{\partial \alpha(u_m,v_m)_{d_i,p,c}}{\partial x_{i,j}} = N_{i,k}(u_m) * N_{j,l}(v_m) \tag{8}$$

$$(u_m,v_m)_{d_i,p,c} = \underset{u_s,v_s}{\text{argmin}}(f(\alpha(u_s,v_s)))_{d_i,p,c} \tag{9}$$

Similarly, $\frac{\partial E}{\partial y_{i,j}}$ and $\frac{\partial E}{\partial z_{i,j}}$ are updated. During training, while computing the functional value of a division, the indices $(u_m, v_m)$ of min pooling operation are stored. The detailed algorithm is provided below in Algorithm 1.

## 5 EXPERIMENTS AND RESULTS

We used Nvidia's GTX 1080Ti, with 11 GB of VRAM to train our network. The point clouds are converted into distance fields. We used a batch size of 32, learning rate of 0.01 with SGD solver and momentum 0.75 and weight decay of $10^{-5}$.

**Baseline:** We show our results on ModelNet40 [37] dataset which has 40 classes of rigid 3D CAD models. As mentioned in Section 2,

**Result:** Updated locations of control points of each surface.
*Initialize:* Number of sets of surfaces N, number of candidates in each set P, number of sampling points S, randomly initialize locations of control points of each surface $x_{i,j}, y_{i,j}, z_{i,j}$, Degree of the curve along both parametric directions K, number of divisions in parametric space D, number of iterations T;
**while** $iterations \leq T$ **do**
  **Forward Pass:**
  **for** $p := 0$ *to* $P$ **do**
    **for** $d := 0$ *to* $D$ **do**
      **for** $s := 0$ *to* $S$ **do**
        evaluate $\alpha(u_s, v_s)$ using Eq 3
        $f(\alpha(u_s, v_s))_{d_i,p} \leftarrow min(f(\alpha(u_s, v_s)))\forall s \in S$
        $u_m, v_m \leftarrow \text{argmin}_{u_s, v_s} f(\alpha(u_s, v_s))$
      **end**
    **end**
    $f_{net,n} = concat(f(\alpha(u_s, v_s))_{d_i,p}))\forall d_i \in D$
  **end**
  Calculate global value by Eq 5
  **Backward Pass:**
  **for** $p := 0$ *to* $P$ **do**
    **for** $d := 0$ *to* $D$ **do**
      Compute $\frac{\partial E}{\partial x_{i,j}}, \frac{\partial E}{\partial y_{i,j}}$ and $\frac{\partial E}{\partial z_{i,j}}$ by using Eq 6 and 7 with $u_m, v_m$
    **end**
  **end**
**end**

**Algorithm 1:** The learning scheme

there are several works for classification of ModelNet datasets. However, the input formats are different for each of these works. We will compare our results with volumetric input, in particular, the FPNN [19].

We train our SplineNet with varied parameters and settings and compare the results with FPNN. To argue that constructing local geometry aware global characterization greatly enhances the classification performance on ModelNet40 dataset, we perform experiments where the locations of the control points of each surface are updated and not updated. For this evaluation, we have used 1024 sets of surfaces wherein each set has 8 surfaces and each surface has 9 control points. The number of divisions in each surface are 6. In each division, the sampling points are 5. The order of the polynomial is fixed to 4 along each of the parametric direction. We show the quantitative results of both the experiments in Table 1.

It is evident that local sensing is adding more essential information and the performance is increased from 79.1% to 82.94%. FPNN constructs a robust global representation which can be improved by adding local geometry to achieve a better performance as shown in Table 1 on 64 resolution data.

## 5.1 Initialization of surfaces

A surface is defined by the locations of its control points. For initialization of these control points, we assume the volumetric grid to be unit dimensional. We initialize a line of random length $l$ and

| Method | without updating locations | Updated locations |
|---|---|---|
| FPNN [19] | 79.1 | 85.0 |
| OURS | **82.94** | **86.8** |

**Table 1: A comparison of accuracy on ModelNet40 [37] in 1FC setting on 64 resolution input.**

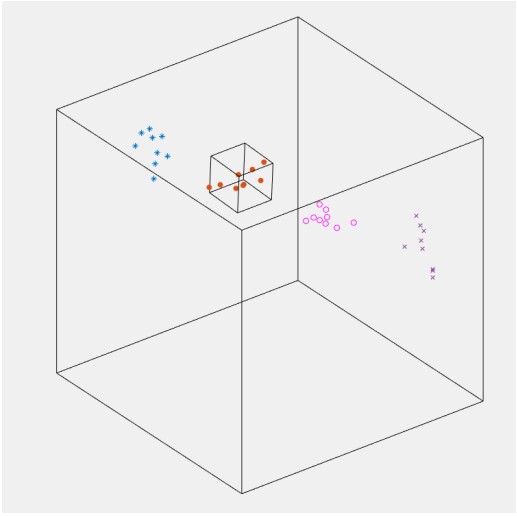

**Figure 4: Initialization of control points of each surface. A single set of four surfaces i.e. their control points are visualized. Please refer Section 5.1 for more details.**

orientation varying from $0.1 - 0.9$. On the line, we randomly chose 8 points. With the chosen points as center, we initialize cuboid which has random length, breadth, height and orientation with not exceeding $0.4 * l$.

This procedure of initialization ensures that each set of surfaces has a different span in the volumetric grid and the range of sensing is maintained. Within each cuboid we initialize uniformly sampled control points for each of the surface. We demonstrate only four candidates in Figure 4 for better visualization. However, in experiments we used 8 surfaces.

## 5.2 Hyper-parameter Estimation

The network has many hyper parameters which include number of surface sets to be initialized, order of the surfaces, number of divisions of the parametric space etc. In order to estimate these parameters, we train the SplineNet under different settings without updating the locations of the control points. We performed experiments with varying number of sets of b-spline surfaces initialized. The quantitative results are shown in Figure 5. It is to be observed that 256 sets of surfaces with 8 surfaces in each set is required to match the performance of FPNN which has 1024 filters and 8 points in a filter.

The smoothness of the surface is dependent on the order of the curve. If the order i.e. (degree+1) of the curve is 2, we get piecewise continuous planes. Increasing in the order of the surface results in a much smoother surface. In all our experiments, we use the same

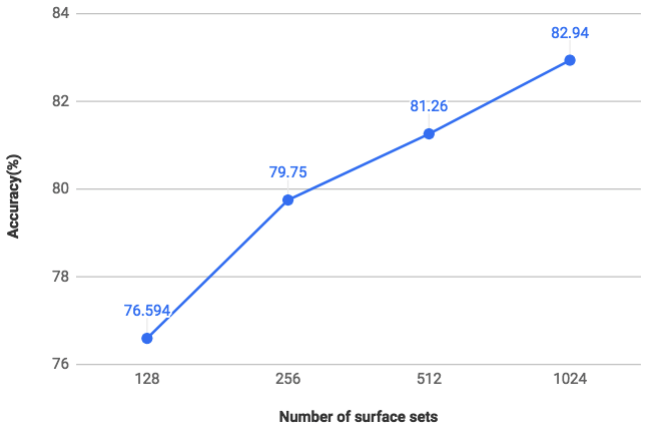


**Figure 5: Accuracy of SplineNet with varied number of surface sets**

order along U,V directions. Table 2 show the results of experiments.

| order | (3,3) | (4,4) |
|---|---|---|
| accuracy | 82.17 | 82.94 |

**Table 2: Evaluation of our approach by varying the order of the curve. We keep the order of curve same along U,V directions**

To sense the surface information locally, we divide each patch into several divisions in parametric space. From each division, we evaluate the functional value at $n = 5$ random locations. We take minimum of these values and forward to LocalAggregation layer. Increase in the number of divisions results in dense sampling on the surface. Hence, the performance is increased with the increase of number of divisions. We train our network with number of divisions set to 6 further in all experiments.

| Divisions | 4 | 6 | 9 |
|---|---|---|---|
| accuracy | 82.3 | 82.94 | 82.96 |

**Table 3: Performance of number of parametric divisions on 128 resolution data**

To study the importance of various functions on local geometry of 3D shapes, we tested with several functions in the LocalAggregation layer apart from concatenation. From each of the patch initialized, we take the mean of the functional values of randomly sampled points instead of concatenating. We consider this mean value as the contributed functional value for of the patch which is fed to further layers. We have achieved an under-par accuracy of 78.3 accuracy when the locations are not updated. Similarly, we also used standard deviation in the functional values as the net contribution and learned that concatenation greatly influences the performance.

We also observe that our method performs better when the resolution of the input is high as shown in Table 4. This is essentially because of the fact that in forward propagation, we evaluate the functional value at a point by performing tri-linear interpolation. Increasing the resolution results in effective sensing in local neighborhood which the surface attempts to exploit.

| Resolution | 32 | 64 | 128 |
|---|---|---|---|
| accuracy | 84.8 | 86.8 | 87.4 |

**Table 4: Classification accuracy of our method on varying input resolutions**

| Method | Accuracy |
|---|---|
| Aravind et al. [1] | 86.5 |
| 3D-CapsNets [17] | 82.73 |
| VSL [20] | 84.5 |
| Ours | 86.28 |

**Table 5: Classification accuracy (%) on ModelNet40 comparison of our model and other recent volumetric methods**

## 5.3 Visualization of SplineNet features

We show tSNE features of the fully connected layer of our SplineNet in Figure 6. It is easy to infer that SplineNet is able to efficiently embed similar class candidates in one neighborhood. This embedding suggests that the learned features are generalizable for retrieval tasks whereas the network is trained for classification.

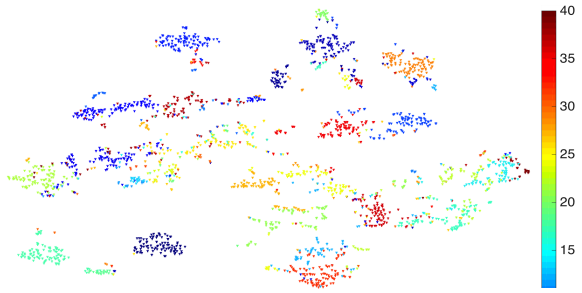


**Figure 6: tSNE feature visualization**

## 6 CONCLUSION AND FUTURE WORK

We propose SplineNet, a novel learning paradigm to address the challenging issues of efficient and effective 3D volumetric data classification. In particular, to account for local geometric variations while generating global representation of 3D data, we introduce B-spline surfaces in SplineNet. The locations of these surfaces are learned from data and analytical solutions to perform back propagation are derived. We show results on publicly available dataset and show superior over state-of-the-method. We also demonstrate the robustness of features learned by SplineNet While the spline surfaces are used for classification in this paper, reconstructing 3D

shapes by generating surfaces is a feasible future direction that has many potential applications.

# A APPENDIX

## A.1 B-spline Curves and Surfaces

*A.1.1 B-spline basis calculation.* $N_{i,k}$ in Equation 2 can be computed recursively as

$$N_{i,k}(u) = \frac{(u - t_i)N_{i,k-1}(u)}{t_{i+k-1} - t_i} + \frac{(t_{i+k} - u)N_{i+1,k+1}(u)}{t_{i+k} - t_{i+1}} \quad (10)$$

where $t_i$ are knot values. The number of knot values is equal to sum of number of points and degree of the curve. They are computed as follows

$$t_i = \begin{cases} 0 & \text{if } i < k \\ i - k + 1 & \text{if } k \leq i \leq n \\ n - k + 2 & \text{if } i > n \end{cases}$$

$$N_{i,k}(u) = \begin{cases} 1 & \text{if } t_i \leq u \leq t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

## A.2 B-spline Neural Network

The division of parametric space is illustrated in Fig 7. The curve is generated by 12 control points represented in black dots. The red dots represents knot values. B-spline curve is piece-wise continuous in every consecutive knot interval. For instance, each knot interval can be assumed as a division of parametric space. We randomly evaluate the functional values in each segment and the minimum value $f(u_i)$ is sent to further layers. We perform similar operations in surface which is a natural extension of the curve

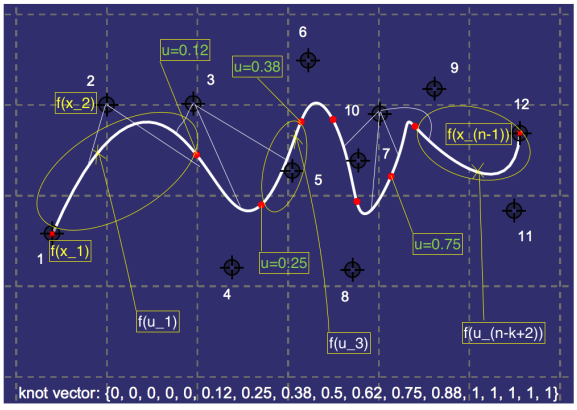


**Figure 7: Quartic Curve generated by 12 control points. Knot values are shown. Parametric divisions are visualized.**

# REFERENCES


[1] Varun Arvind, Anthony Costa, Marcus Badgeley, Samuel Cho, and Eric Oermann. 2017. Wide and deep volumetric residual networks for volumetric image classification. *arXiv preprint arXiv:1710.01217* (2017).

[2] Varun Arvind, Anthony B. Costa, Marcus A. Badgeley, Samuel Cho, and Eric K. Oermann. 2017. Wide and deep volumetric residual networks for volumetric image classification. *CoRR* abs/1710.01217 (2017). arXiv:1710.01217 http://arxiv.org/abs/1710.01217

[3] Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. 2011. The wave kernel signature: A quantum mechanical approach to shape analysis. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. IEEE, 1626–1633.

[4] Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael Bronstein. 2016. Learning shape correspondence with anisotropic convolutional neural networks. In *Advances in Neural Information Processing Systems*. 3189–3197.

[5] André Brock, Theodore Lim, James M. Ritchie, and Nick Weston. 2016. Generative and Discriminative Voxel Modeling with Convolutional Neural Networks. *CoRR* abs/1608.04236 (2016). arXiv:1608.04236 http://arxiv.org/abs/1608.04236

[6] André Brock, Theodore Lim, James M Ritchie, and Nick Weston. 2016. Generative and Discriminative Voxel Modeling with Convolutional Neural Networks. CoRR abs/1608.04236 (2016). (2016).

[7] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral Networks and Locally Connected Networks on Graphs. *CoRR* abs/1312.6203 (2013). arXiv:1312.6203 http://arxiv.org/abs/1312.6203

[8] Yang Chen and Gérard Medioni. 1992. Object modelling by registration of multiple range images. *Image and vision computing* 10, 3 (1992), 145–155.

[9] Chin Seng Chua and Ray Jarvis. 1997. Point signatures: A new representation for 3d object recognition. *International Journal of Computer Vision* 25, 1 (1997), 63–85.

[10] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. *CoRR* abs/1606.09375 (2016). arXiv:1606.09375 http://arxiv.org/abs/1606.09375

[11] Timothy Gatzke, Cindy Grimm, Michael Garland, and Steve Zelinka. 2005. Curvature maps for local shape comparison. In *Shape Modeling and Applications, 2005 International Conference*. IEEE, 244–253.

[12] Dirk Holz and Sven Behnke. 2013. Fast range image segmentation and smoothing using approximate surface reconstruction and region growing. In *Intelligent autonomous systems 12*. Springer, 61–73.

[13] Andrew E Johnson and Martial Hebert. 1999. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 5 (1999), 433–449.

[14] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. 2016. RotationNet: Joint Object Categorization and Pose Estimation Using Multiviews from Unsupervised Viewpoints. *arXiv preprint arXiv:1603.06208* (2016).

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

[16] Marcel Körtgen, Gil-Joo Park, Marcin Novotni, and Reinhard Klein. 2003. 3D shape matching with 3D shape contexts. In *The 7th central European seminar on computer graphics*, Vol. 3. Budmerice, 5–17.

[17] Ryan Lambert. 2018. Capsule Nets for Content Based 3D Model Retrieval. (2018). https://github.com/Ryanglambert/3d_model_retriever

[18] Yann Lecun, LÃIon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*. 2278–2324.

[19] Yangyan Li, Soeren Pirk, Hao Su, Charles R Qi, and Leonidas J Guibas. 2016. Fpnn: Field probing neural networks for 3d data. In *Advances in Neural Information Processing Systems*. 307–315.

[20] Shikun Liu, Lee Giles, and Alexander Ororbia. 2018. Learning a Hierarchical Latent-Variable Model of 3D Shapes. In *2018 International Conference on 3D Vision (3DV)*. IEEE, 542–551.

[21] David G Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision* 60, 2 (2004), 91–110.

[22] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. 2015. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*. 37–45.

[23] Jonathan Masci, Emanuele Rodolà, Davide Boscaini, Michael M Bronstein, and Hao Li. 2016. Geometric deep learning. In *SIGGRAPH ASIA 2016 Courses*. ACM, 1.

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

[25] Michela Mortara, Giuseppe Patané, Michela Spagnuolo, Bianca Falcidieno, and Jarek Rossignac. 2004. Blowing bubbles for multi-scale analysis and decomposition of triangle meshes. *Algorithmica* 38, 1 (2004), 227–248.

[26] Helmut Pottmann, Johannes Wallner, Qi-Xing Huang, and Yong-Liang Yang. 2009. Integral invariants for robust geometry processing. *Computer Aided Geometric Design* 26, 1 (2009), 37–60.

[27] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2016. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *CoRR* abs/1612.00593 (2016). arXiv:1612.00593 http://arxiv.org/abs/1612.00593

[28] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *CoRR* abs/1706.02413 (2017). arXiv:1706.02413 http://arxiv.org/abs/1706.02413

[29] Raif M Rustamov. 2007. Laplace-Beltrami eigenfunctions for deformation invariant shape representation. In *Proceedings of the fifth Eurographics symposium on Geometry processing*. Eurographics Association, 225–233.

[30] Nima Sedaghat, Mohammadreza Zolfaghari, and Thomas Brox. 2016. Orientation-boosted Voxel Nets for 3D Object Recognition. *CoRR* abs/1604.03351 (2016). arXiv:1604.03351 http://arxiv.org/abs/1604.03351

[31] Baoguang Shi, Song Bai, Zhichao Zhou, and Xiang Bai. 2015. Deeppano: Deep panoramic representation for 3-d shape recognition. *IEEE Signal Processing Letters* 22, 12 (2015), 2339–2343.

[32] Dirk Smeets, Jeroen Hermans, Dirk Vandermeulen, and Paul Suetens. 2012. Isometric deformation invariant 3D shape recognition. *Pattern Recognition* 45, 7 (2012), 2817–2831.

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

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

[35] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. 2017. Marrnet: 3d shape reconstruction via 2.5 d sketches. In *Advances in neural information processing systems*. 540–550.

[36] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. 2016. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems*. 82–90.

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

[38] Sameh M Yamany and Aly A Farag. 1999. Free-form surface registration using surface signatures. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, Vol. 2. IEEE, 1098–1104.

[39] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. 2018. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Vol. 3.