TSRNet: Scalable 3D Surface Reconstruction Network for Point Clouds using Tangent Convolution

Zhenxing Mi [*]   Yiming Luo [*]   Wenbing Tao [†]

National Key Laboratory of Science and Technology on Multi-spectral Information Processing
School of Artifical Intelligence and Automation, Huazhong University of Science and Technology, China

{m201772503, yiming_luo, wenbingtao}@hust.edu.cn

Abstract

Existing learning-based surface reconstruction methods from point clouds are still facing challenges in terms of scalability and preservation of details on point clouds of large scales. In this paper, we propose the TSRNet, a novel scalable learning-based method for surface reconstruction. It first takes a point cloud and its related octree vertices as input and learns to classify whether the octree vertices are in front or at back of the implicit surface. Then the Marching Cubes (MC) is applied to extract a surface from the binary labeled octree. In our method, we design a scalable learning-based pipeline for surface reconstruction. It does not consider the whole input data at once. It allows to divide the point cloud and octree vertices and to process different parts in parallel. Our network captures local geometry details by constructing local geometry-aware features for octree vertices. The local geometry-aware features enhance the predication accuracy greatly for the relative position among the vertices and the implicit surface. They also boost the generalization capability of our network. Our method is able to reconstruct local geometry details from point clouds of different scales, especially for point clouds with millions of points. More importantly, the time consumption on such point clouds is acceptable and competitive. Experiments show that our method achieves a significant breakthrough in scalability and quality compared with state-of-the-art learning-based methods.

1. Introduction

Point cloud is an important and widely used representation for 3D data. Surface reconstruction from point clouds has been well studied in computer graphics. A lot of geometric reconstruction methods have been proposed. A com-
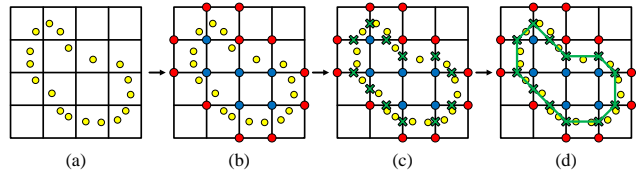
---

[*]Equal contribution
[†]Corresponding author

Figure 1. 2D example of surface reconstruction methods using implicit function. Yellow dots represent points in point cloud. Red dots represent vertices outside the implicit surface. Blue dots represent vertices inside the implicit surface. According to the implicit function value on vertices, the Marching Cubes method finds the intersections (green ×) of the grid lines and the implicit surface and then connects these intersections to extract a surface (solid green line).

monly used pipeline in geometric reconstruction methods first computes an implicit function on a 3D grid [5, 12]. Then the Marching Cubes (MC) method [15] is applied to extract an isosurface from the 3D grid through implicit function values. The grid is usually an adaptive octree. The intersection of two grid lines in an octree is named as octree vertex. Figure 1 shows a 2D example. Specifically, the implicit function can be an indicator function that indicates whether the vertices are inside or outside the implicit surface. That is, a surface reconstruction problem can be treated as a binary classification problem for octree vertices.

In real-world reconstruction tasks, the point clouds are mainly acquired by scanners or Multi-View Stereo (MVS) methods [24, 29]. These point clouds are usually dense, with complex topologies and millions of points, which bring challenges to surface reconstruction methods.

With the development of deep learning, a variety of learning-based surface reconstruction methods have been proposed [6, 22, 14, 1, 8, 17, 18]. However, there are still problems for learning-based methods to reconstruct real-world point clouds for four reasons illustrated as follows.

1) The network architectures and output representations of these learning-based methods usually force the learning-
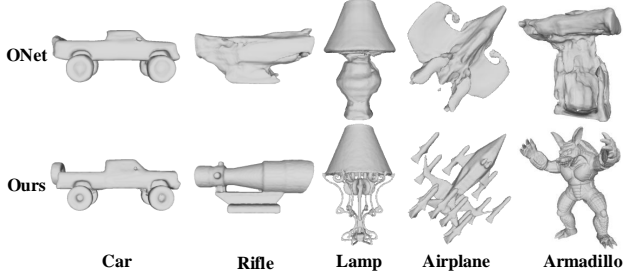
Figure 2. Examples of reconstructed surfaces of ONet [17] and Ours on ShapeNet [3] and Stanford 3D [25] (Armadillo).

based approaches to consider all the points at once. Therefore, they can not allow dividing input data and processing different parts separately. When point clouds scale to millions of points, devices may not be able to handle large amounts of memory usage, which will make the model less scalable. For example, the Deep Marching Cubes [14] transform the entire point cloud into voxel grids at once and directly output a irregular mesh, which is less scalable and is only applicable to point clouds with thousands of points.

2) Although some methods can be applied to large-scale point clouds, in reality they greatly downsample or aggregate the points into a relative small scale. For example, the Occupancy Networks (ONet) [17] and DeepSDF [18] encode the entire points into a fixed-size latent vector of 512 elements. The representing power is limited. Consequently, they come at the expense of surface quality reconstructed from large-scale point clouds.

3) Even on small-scale point clouds, learning-based methods is good at reconstructing shapes with quite simple topology rather than those complex shapes. In fact, the result surfaces of simple shapes are usually over smoothed. The Figure 2 shows reconstruction examples of a state-of-the-art learning-based method (ONet) [17] and ours. For the simple shapes, ONet gets a smooth surface. However, for the more complex topology shapes, it can not perfectly reconstruct the relevant details.

4) Some learning-based methods need a large portion of the dataset for training. For example, the ONet [17] uses 4/5 of its dataset (about $40K$ 3D shapes) as training set. The trained network is inevitably more prone to over fitting the dataset. What's more, it can not generalize well among different datasets.

Recently the tangent convolution network has been proposed for semantic segmentation of point clouds. It constructs a set of 2D tangent images for points by projecting local neighbor points onto tangent planes. Then 2D convolutions are applied on these images to learn point features. The tangent images stores signed projection distance and other input features. The network operations of tangent convolution network are all performed in a fixed-size local spa-

tial region. They focus more on local neighbors instead of the global shape. This allows dividing the input point cloud through overlapping bounding boxes and processing them parallelly, which is different from the well-known classification network, PointNet [19].

In this paper, we take the advantages of the commonly used implicit function method discussed above for surface reconstruction. Specifically, the implicit function values are 0-1 labels of octree vertices. They indicate whether the octree vertices are in front or at back of the implicit surface. The strategy simplifies the surface reconstruction to binary classification problem, which is simpler and more effective. The ONet [17] also trains its network in a binary classification manner. However, this strategy does not guarantee quality results. There are three main limitations in dealing with large-scale data that need to be broken.

1) **The scalability of the network.** A strong scalability of the classification network is important for tackling large-scale data. However, the commonly used PointNet [19] in reconstruction methods consider entire points at once and are thus less scalable. Our network does not need the entire input data at once and wins strong scalability. This is mainly because the tangent convolution operations involved in our network are all performed in a fixed-size local spatial region, and the 1×1 convolution operations are also independent. Moreover, these network operations are independent from the octree structures. Therefore, our network allows partitioning the points and octree vertices through bounding boxes and processing different parts separately so that it wins strong scalability.

2) **The ability to reconstruct geometry details.** In order to achieve a high classification accuracy for octree vertices, the ability to capture local geometry details is required. Some methods like ONet[17] and DeepSDF [18] aggregate the features of the entire points into a fixed-size global latent vector (512 elements), which could lose a lot of geometry details. This will lead to a decrease on reconstruction quality.

In our network, we focus on learning local geometry information of the implicit surface. Since the implicit surface are determined by octree vertex labels, in order to make accurate classification for octree vertices, the most important task is to construct vertex features properly. The octree vertex features in our network are constructed from local neighbor points and take the advantage of local projection information in the tangent image. The most important information is signed projection distances between octree vertices and its neighbor points. The signed distances are commonly used in geometric methods such as the TSDF [5] method and MLS methods [13, 9]. They directly provides local geometry information for accurate front-back classification. This enables our method to capture local geometry details of large-scale point clouds. Therefore, our network

benefits from geometry-aware features to reconstruct accurate surfaces.

Some other methods such as the ONet[17] and DeepSDF [18] construct the features of vertices by directly concatenating 3D coordinates of vertices with the global latent vector of the point cloud. This feature construction method cannot provide explicit information about relative position among vertices and the implicit surface. Therefore, they cannot capture local geometry details well to reconstruct quality surfaces.

3) **The generalization ability of the network.** Giving too much attention to global feature rather than just considering local information would limit the generalization performance among different type of shapes. The octree vertex features in our network are local geometry-aware. It doesn't rely on global shape information excessively. Therefore, it has good generalization capability and does not need too much training data, which avoids overfitting on dataset.

Overall, our contribution can be summarized as the following two points. 1) We design a scalable pipeline for reconstructing surface from real-world point clouds. 2) We construct local geometry-aware octree vertex features, which leads to accurate classification for octree vertices and good generalization capability among different datasets. Experiments have shown that our method achieves a significant improvement in scalability and quality compared with state-of-the-art learning-based methods. It also achieves comparable results with state-of-the-art geometric methods. The proposed TSRNet is a practical learning-based method for real-world surface reconstruction tasks.

2. Related Work

In this section, we first review some important geometric reconstruction methods to introduce basic concepts. Then we focus on existing learning-based surface reconstruction methods from point clouds. We mainly analyze whether they are able to scale to large-scale datasets and to capture geometry details in terms of network architectures and output representations.

2.1. Geometric reconstruction methods

Geometric reconstruction methods can be broadly categorized into global methods and local methods. The global methods consider all the data at once, such as the radial basis functions (RBFs) methods [2, 28], and the (Screened) Poisson Surface Reconstruction method (PSR). [11, 12].

The local fitting methods usually define a truncate signed distance function (TSDF) [5] on a volumetric grid. The various moving least squares methods (MLS) fit a local distance field or vector fields by spatially varying low-degree polynomials and blend several nearby points together [13, 9]. The local projection and local least squares

fitting used by MLS are similar to the tangent convolution used in our network. Local methods can be well scaled.

2.2. Learning-based reconstruction methods

Network architecture. One straightforward network architecture for point clouds is to convert the point clouds into regular 3D voxel grids or adaptive octree grids and apply 3D convolutions [16, 21]. The voxel-based network is used in 3D-EPN method [6] and the Deep Marching Cubes (DMC) method [14]. The OctNetFusion [22] and 3D-CFCN [1] reconstruct implicit surface from multiple depth images based on OctNet [21]. The voxel-based network faces cubic growth of computational and memory requirements. The network operations of octree-based networks are complicated and are highly related to the octree structure. They also face computational issues. Therefore, networks operating on grids still face difficulties to reconstruct datasets with large scales.

Another type of network architecture learns point features directly. The commonly used point cloud network is the PointNet [19]. It encodes global shape features into a latent vector of fixed size. Some reconstruction methods also extract a latent vector from the point cloud, such as the the Occupancy Networks [17] and DeepSDF [18]. The networks are able to encode features of a large-scale point cloud into a latent vector. However, the latent vector of a small size limits its representation power for complex point clouds. Our TSRNet also directly learns point and vertex features but the features are learned from local neighbors. Compared with networks using latent vector, our TSRNet is able to capture geometry details of large-scale point clouds with complex topology.

Output representation. The scalability of learning-based reconstruction methods are also greatly influenced by the output representation. The reconstruction methods based on voxel or octree grids usually use the occupancy or TSDF on grids as the output representation. This output representation, together with their network operations, is highly related to the grid structure. Their scalability is limited. The Deep Marching Cubes method [14] and the AtlasNet [8] directly produce a triangular surface. The predictions of different parts of the surface are interdependent so that they have to consider the whole input point cloud at once.

The Occupancy Networks [17] and the DeepSDF [18] method learn shape-conditioned classifiers and the decision boundary is the surface. Their representations are similar to the front-back representation of our TSRNet. Although the Occupancy Networks and the DeepSDF support to make predictions over 3D locations parallelly, they need the whole point cloud to get the global latent vector. They can only reconstruct watertight surfaces so that they do not allow partitioning input points.
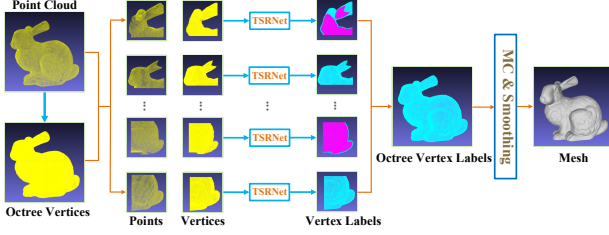
Figure 3. The pipeline of our method. Different parts are processed by our TSRNet parallelly.

2.3. Tangent Convolution

The tangent convolution [26] is a new convolutional network for semantic segmentation of 3D point clouds. The tangent convolution method defines three new network operations: tangent convolution, pooling and unpooling. The tangent convolution is defined as 2D convolution on tangent images. The tangent images are construct by local projection of neighbor points. The indices of the projected points used for tangent image are precomputed. It makes the tangent convolution much more efficient. The signals used in tangent images are signed projection distance, normals, etc that represent local surface geometry.

The neighbor points used for tangent convolution are collected by ball query that finds all points within a radius to the query point. The radius is defined as the receptive field size of the tangent convolution. The pooling and unpooling are implemented via hashing onto a regular 3D grid. The radius search and grid sampling guarantee that the tangent convolution, pooling and unpooling are performed in a fixed-size spatial region. Another representative network learning local features, the PointNet++ [20], uses iterative farthest point sampling (FPS) to choose a subset of points. Its grouping operation cannot be guaranteed to be in a fixed-size spatial region. Therefore it does not allow dividing input points through bounding boxes.

3. Method

In this section, we introduce the pipeline of our method in Section 3.1. In Section 3.2, the construction method for geometry-aware vertex feature is illustrated. In Section 3.3, we describe our network architecture and some implementation details. The surface extraction and smoothing methods are briefly introduced in Section 3.4. And in Section 3.5, we explain how to prepare the training data.

3.1. Pipeline

We design a scalable learning-based surface reconstruction method, named as TSRNet, which allows dividing the input point clouds and octree vertices. Figure 3 shows the pipeline of our method. Let $\mathcal{P} = \{\mathbf{p}_i | i = 1, ..., N\}$ be the input point cloud of N points, where $\mathbf{p}_i$ is point coordinate with normal $\mathbf{n}_i$. We construct an octree from this point cloud and extract the vertices $\mathcal{V} = \{\mathbf{v}_i | i = 1, ..., M\}$ of the finest level of this octree, where each vertex $\mathbf{v}_i$ is the coordinate. To prevent the whole input data exceeding the GPU memory, we use bounding boxes to partition the point cloud and the vertices into $\{\mathcal{P}_j | j = 1, ..., K\}$ and $\{\mathcal{V}_j | j = 1, ..., K\}$, where K is the number of boxes. The bounding boxes for vertices are not overlapping. The corresponding bounding boxes of points are larger than those of octree vertices to ensure that all the points in the largest receptive field of the vertices are included. This enables our network to make accurate prediction for vertices in the boundary of the bounding box. For the part j, the vertices $\mathcal{V}_j$ and the corresponding points $\mathcal{P}_j$ are fed into TSRNet. The TSRNet classifies each vertex $\mathbf{v}_{ji}$ in $\mathcal{V}_j$ as in front or at back of the implicit surface represented by $\mathcal{P}_j$. Let the function represented by the network be f_θ, where θ represents the parameters of the network. Then the key of our reconstruction method is to classify vertices $\mathbf{v}_{ji}$ in $\mathcal{V}_j$. It can be formulated as:

$$f_\theta(\mathcal{P}_j, \mathcal{V}_j, \mathbf{v}_{ji}) \rightarrow \{0, 1\} \tag{1}$$

The front or back of an octree vertex is defined by the normal direction of its nearest surface. To classify a vertex as front or back, this vertex needs to be near the implicit surface. The finest-level vertices of an adaptive octree are near the surface so that front-back classification on them is feasible. It is worth noting that the network operations in TSRNet are not related to the structure of the octree.

After all the vertices are labeled, we extract surface using Marching Cubes (MC) method and post-process the surface with a simple Laplacian-based mesh smoothing method.

3.2. Geometry-aware Vertex Feature

In order to get accurate classification for octree vertices and good generalization capability, we construct geometry-aware features for octree vertices from point clouds. As discussed in the Introduction, the most important features for octree vertex classification are signed projection distances among octree vertices and its neighbor points. Since the tangent image in the tangent convolution [26] is constructed by local projection of neighbor points, the tangent convolution has the potential to construct geometry-aware features for octree vertices. However, there are problems in applying tangent convolution for octree vertices directly. For a 3D location $\mathbf{p}$, the normal $\mathbf{n_p}$ of its tangent image is estimated through local covariance analysis [23]. Let $\mathbf{k}$ be the eigenvector related to the smallest eigenvalue of the covariance matrix of $\mathbf{p}$. The normal of tangent image is defined as $\mathbf{k}$. Due to the direction of eigenvector $\mathbf{k}$ is ambiguous, it may not be consistent with the real orientations of local

implicit surfaces. The signs of projection distances hence do not represent front-back information accurately.

In order to solve this problem, we modify the definition of $\mathbf{n_p}$. Since the front-back classification is related to neighbor points, we use the input normals of neighbor points as additional constraints to define $\mathbf{n_p}$. Let $\mathbf{n}_a$ be the average input normal of the neighbor points of $\mathbf{p}$. In our definition, if the angle between eigenvector $\mathbf{k}$ and $\mathbf{n}_a$ is more than $90°$, we invert the direction of $\mathbf{k}$. Then we use it as the tangent image's normal $\mathbf{n_p}$. That is, our definition of the tangent image ensures that $\mathbf{n_p}^\top \mathbf{n}_a > 0$.

The features of octree vertices constructed by modified tangent convolution directly encode front-back information. They are local and not related to the types of shapes, so our network is scalable and can generalize well among different datasets. It's worth noting that we use neighbor points in the point cloud rather than neighbor vertices to compute the tangent images. It is because our network classifies octree vertices with respect to the surface represented by points from $\mathcal{P}$, rather than represented by neighbor vertices.

3.3. Network

Network architecture. The network architecture of our TSRNet is illustrated in Figure 4. It contains two parts. The left part is for point feature extraction. The right is for vertex feature construction and label prediction. They are connected by tangent convolution layers for vertices.

The left part of our network encodes the features of the input N points from a point cloud. It is a fully-convolutional U-shaped network with skip connections. It is also built by our modified tangent convolution layers. Two pooling operations are used to increase the size of the receptive field.

The right part is the core of our method. In the right part, the geometry-aware features of octree vertices are firstly constructed by modified tangent convolution, from point features in corresponding scale levels. Then the 1×1 convolutions and unpooling operations are applied to encode more abstract features and to predict labels for each vertex. The index matrices used by unpooling layers in this part are precomputed when downsampling vertices via grid hashing.

The input signals used by our method are all local signals. We use the signed distances from neighbor points to tangent plane (D), and normals relative to the normal of the tangent plane (N) as input. Since the network operations in TSRNet are all performed in a fixed-size local region, i.e. the modified tangent convolution, the 1×1 convolution, the pooling and unpooling of points and vertices,

our network allows dividing the points and vertices with bounding-boxes and processing each part separately. It is worth noting that the bounding-box used for points are larger than octree vertices in order to make accurate prediction for vertices within the boundary.

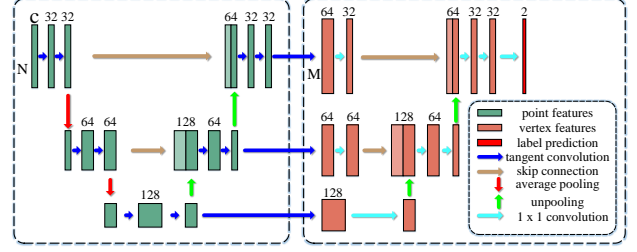**Implementation details.** In our implementation, similar to

Figure 4. Our network architecture. Arrows of different colors represent different network operations. The inputs of this network include N points and M octree vertices. The initial features of the points have C channels.

[26], we downsample the input points and vertices using grid hashing for pooling and unpooling. Given an octree of depth D, The length the finest-level edges is $1/2^D$. We set the initial receptive field size in our network as $4/2^D$. The grid hashing size of input points at three scales are set as $1/2^{(D+2)}, 1/2^{(D+1)}, 1/2^D$. We don't apply gird hashing for vertices in the first scale because the vertices have an initial grid size of $1/2^D$ originally. The last two grid hashing sizes of vertices are set as $1/2^{(D-1)}$ and $1/2^{(D-2)}$. In order to retain more surface details for the feature construction of vertices, we set smaller grid hashing sizes for points.

3.4. Surface Extraction and Smoothing

We use the Marching Cubes (MC) method [15] to extract surface from the labeled octree vertices. The MC needs to find the intersections between octree edges and the implicit surface using the labels of vertices. Since we directly use the mid-points of edges as the intersections, the resulting mesh has discretization artifacts inevitably. In order to refine the output surfaces, we use a simple Laplacian-based mesh smoothing method [27] as a post-processing step.

3.5. Data Preparation

To prepare training data, we first normalize the coordinates of the input point cloud and surfaces. Then an octree is built from the point cloud. The octree building method is adapted from the open-source code of Poisson Surface Reconstruction [11, 12]. It ensures that the finest level of the result octree contains not only cubes with points in them, but also their $3 \times 3 \times 3$ neighbor cubes. Therefore, the octree is dense enough to completely reconstruct the surface. We then use the ground truth surface with normals to label these vertices. For datasets without ground truth surfaces, we generate ground truth surfaces through PSR [12]. More detailed introduction of our data preparing method can be found in our supplementary file.

4. Experiments

In this section we perform a series of experiments on datasets of different scales to evaluate our method. The datasets are all widely used in 3D-related tasks. We mainly examine the ability to capture geometry details, the scalability and generalization capability of our method on both small-scale and large-scale datasets. Due to the lack of triangular meshes, the ground-truth meshes used for training are all produced by PSR [12].

4.1. Datasets and Evaluation Metrics

Datasets. In order to compare with several state-of-the-art learning-based methods, we do experiments on the same subset of ShapeNet [3] as ONet [17], in which the point clouds are relatively small with tens of thousands of points in each. Furthermore, in order to better evaluate the scalability and generalization capability to handle large-scale point clouds, we choose the scans of DTU dataset [10] and Stanford 3D Scanning Repository [25]. Most point clouds in the two datasets contain millions of points.

Evaluation Metrics. The direct metric of our network is the classification accuracy of octree vertices, which matters a lot to the quality of the final mesh. For mesh comparison on ShapeNet, we use the same evaluation metrics with ONet [17], including the volumetric IoU (higher is better), the Chamfer-L_1 distance (lower is better) and the normal consistency score (higher is better). On the DTU dataset, we use DTU evaluation method [10], which mainly evaluates DTU Accuracy and DTU Completeness (both lower is better). And for Stanford datasets, Chamfer distance (CD) (lower is better) is adopted in order to evaluate the points of the output meshes, and the CD is also taken into consideration in the evaluation on DTU dataset.

4.2. Classification Accuracy on Different Datasets

We train our network on the ShapeNet dataset and DTU dataset respectively. The testing results on the three datasets mentioned above are satisfactory. Detailed training settings and train/test split are described in Section 4.3 and 4.4. The classification accuracy of octree vertices are shown in Table 1. Note that there are two testing results on Stanford dataset in the table. They are obtained by the model trained on ShapeNet and DTU respectively.

The classification accuracy of octree vertices on ShapeNet dataset shows strong robustness of our network. Since the original models of ShapeNet is synthetic, the point clouds are clean and ideal. We apply Gaussian noise with zero mean and standard deviation 0.05 to the point clouds in ShapeNet as the ONet does [17]. We can see that our network is still very robust and can achieve high classification accuracy of 97.6% on noisy point clouds.

For 3D scans of large scales, noise in the point clouds is inevitable and we don't need to add any noise to them. On

Table 1. Classification Accuracy of TSRNet on Datasets of different scales. ShapeNet-Model and DTU-Model represent the models we trained on ShapeNet [3] and DTU [10] respectively.

Dataset	ShapeNet [3]	DTU [10]	Stanford [25]	
			ShapeNet-Model	DTU-Model
Accuracy (%)	97.6	95.7	98.1	98.2

Table 2. Results of learning-based methods on ShapeNet [3]. NC = Normal Consistency. The volumetric IoU (higher is better), the Chamfer-L_1 distance (lower is better) and NC (higher is better) are reported.

	IoU	Chamfer-L_1	NC
3D-R2N2 [4]	0.565	0.169	0.719
PSGN [7]	–	0.202	–
DMC [14]	0.647	0.117	0.848
ONet [17]	0.778	0.079	0.895
Ours	**0.957**	**0.024**	**0.967**

DTU dataset, which only includes open scenes, our network gets an average classification accuracy of 95.7% . In order to test the generalization ability of our method, we test on the Stanford dataset with models trained on ShapeNet and DTU respectively. We still achieve high average classification accuracy of 98.1% and 98.2% with the two models. It proves strong generalization ability of our method. Frankly, the powerful generalization of our network makes it unnecessary to retrain the model on different datasets. More generalization capability will be illustrated in Section 4.5.

4.3. Results on Shapenet

In our first experiment, we evaluate the capacity of our network to capture shape details on ShapeNet. We use the same test split as ONet [17] for fair comparisons. The testing set consists of about $10K$ shapes. For training our network, we only randomly select a subset from the training set of ONet. The training set of our method ($4K$ shapes) is 1/10 of the training set of ONet ($40K$ shapes). We do not use a large amount of training data as ONet did mainly for two reasons. On the one hand, it is necessary to eliminate the possibility of overfitting a dataset due to too much training data. On the other hand, our network learns local geometry-aware features so that it can capture more geometry details with a relatively small-scale training data. Therefore, it is unnecessary for us to use too many training samples.

In our experiments, each point cloud in training and testing set contains $100K$ points. We add Gaussian noise with zero mean and standard deviation 0.05 to the point clouds as ONet did. Our network is superiorly efficiency to handle point clouds with large number of points so that our method do not downsample the point clouds. The original synthetic models of ShapeNet don't have consistent normals so that we reconstruct surfaces from the ground truth point clouds using PSR on octrees at depth 9 to generate the traning data.
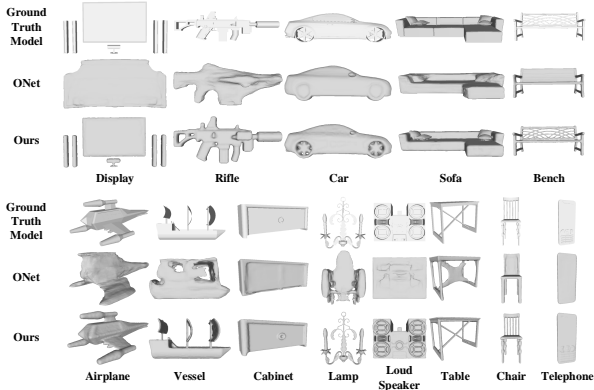
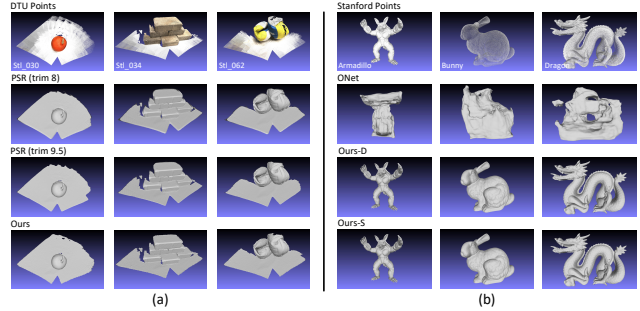Figure 5. Examples of reconstructed surfaces on testing set of ShapeNet [3] by ONet [17] and our method.

Figure 6. Examples of surfaces of our method and PSR [12] on DTU [10] (a). Generalization testing results of our method and ONet [17] on Stanford 3D [25] (b). Ours-S and Ours-D represent the our models trained on ShapeNet [3] and DTU [10] respectively. Background colors are for better visualization for point clouds.

Table 3. Surface quality on DTU [10] testing scenes. Surfaces used for evaluation criterias are all reconstructed at octree depth 9. DA=DTU Accuracy, DC=DTU Completeness, CD=Chamfer distance (all lower is better).

Method	DA		DC		CD	
	Mean	Var.	Mean	Var.	Mean	RMS
PSR(trim 8) [12]	0.473	1.33	0.327	0.220	3.16	12.5
PSR(trim 9.5) [12]	0.330	0.441	0.345	0.438	**1.17**	4.49
Ours	**0.321**	**0.285**	**0.304**	**0.0888**	1.46	**4.42**

We use octrees of depth 8 in our method for training and testing. We apply the post-processing step of our method to the result meshes of the testing set.

For mesh comparison, we use the evaluation metrics on ShapeNet mentioned in Section 4.1, including the volumetric IoU, the Chamfer-L_1 distance and the normal consistency score. We evaluate all shapes in the testing set on these metrics. The results of existing well-known learning based 3D reconstruction approaches, i.e. the 3D-R2N2 [4], PSGN [7], Deep Marching Cube (DMC) [14] and ONet [17], are obtained from the paper of ONet. As mentioned in ONet [17], it is not possible to evaluate the IoU for PSGN for the reason that it doesn't yield watertight meshes. Quantitative results are shown in Table 2.

The Table 2 shows that our method gets great improvements on these metrics. We achieve the highest IoU, lower Chamfer-L_1 distance and higher normal consistency compared with the other learning-based baselines. More specifically, the IoU of our results is about 18% higher than ONet. It is worth noting that our network has only 0.49 million parameters, while ONet has about 13.4 million parameters.

As shown in Figure 5, the surface quality of ours is generally better than those of the ONet. Our network is good at capturing local geometry details of shapes of different classes, even with quite different and complex topology. The ONet can reconstruct smooth surfaces for shapes with simple topology. However, since the global latent vector encoder in ONet loses shape details, it tends to generate an over-smoothed shape of a complex topology. For more visual results, please see our supplementary materials.

4.4. Results on 3D Scans of Larger Scales

Evaluation on DTU dataset. We train and test our network on DTU [10] at octree depth 9 [1]. Since the ground truth surfaces of DTU are not available, we reconstruct surfaces using PSR on octrees of depth 10 to generate the traning data. We trim PSR surfaces using SurfaceTrimmer software provided in PSR with trimming value 8. We randomly extract batches of points from each training scene to train our network. Even though we use only 6 scenes in training set, we achieve a high accuracy and good generalization capability. Table 3 gives the quantitative results on testing scenes in DTU dataset. Qualitative results are shown in Figure 6 (a).

The point clouds in DTU are all open scenes. Although ONet can finish reconstruction tasks of watertight surfaces, it cannot reconstruct surface from point clouds of open scenes. Here we set results of PSR as an evaluation reference. Surfaces reconstructed by PSR for evaluation are also reconstructed at octree depth 9. The results of PSR are always closed surfaces even reconstructed from open scenes. Therefore, we trim them with trimming value 8 and 9.5.

As illustrated in Table 1, our network gets a high average classification accuracy of 95.7%. Compared with PSR with

---

[1] We use scenes {1, 2, 3, 4, 5, 6} as training set, scenes {7, 8, 9} as validation set and scenes {10, 11, 12, 13, 14, 15, 16, 18, 19, 21, 24, 29, 30, 34, 35, 38, 41, 42, 45, 46, 48, 51, 55, 59, 60, 61, 62, 63, 65, 69, 84, 93, 94, 95, 97, 106, 110, 114, 122, 126, 127, 128} as testing set.

trimming value 8, our surfaces outperform them on DTU Accuracy, DTU Completeness and CD. As for PSR with trimming value 9.5, we get similar performance on DTU Accuracy and perform better on DTU Completeness. It is because larger trimming value on PSR surfaces decrease the completeness. The PSR with trimming value 9.5 is more accurate on CD while it is at cost of completeness. As shown in Figure 6 (a), our method can reconstruct more complete boundaries for open scenes. In conclusion, the quality of surfaces of our method is comparable with PSR on DTU datasets. We perform better with respect to the completeness on open scenes. Detailed figures and videos also provided in our supplementary materials.

4.5. Generalization Capability

In this section we evaluate the generalization capability of our method. It is worth noting that three datasets (ShapeNet, DTU, Stanford 3D) are quite different in terms of data scale and data category. Data in the ShapeNet are all synthetic. The DTU only contains real-world scan data of open scenes while the Stanford 3D only has closed scenes. Therefore, tests among these three datasets can well evaluate the generalization performance of the model.

We train our network on the same dataset (ShapeNet) with ONet for fair comparisons. We do not use a large amount of training data ($4K$ shapes) as ONet did ($40K$ shapes) since our network doesn't rely on the training data excessively. Since ONet cannot reconstruct surface through point clouds of open scenes, here we test the two trained models on Stanford 3D. Table 4 shows that the our method gets high classification accuracy. It generalizes better than ONet on Stanford 3D in metric of Chamfer Distance. Figure 6 (b) shows that our results achieve outstanding visual effects and reconstruct geometry details of the shapes well.

We also train our network on DTU and test on Stanford 3D. We only train our network on 6 scenes in DTU dataset and get good generalization capability. Table 4 shows that our network trained on open scenes can also achieve higher classification accuracy on closed scenes. Besides, classification accuracy of our models trained on ShapeNet and DTU are almost exactly the same. We can also find that our models trained on ShapeNet and DTU have obtained roughly similar evaluation results on CD. In general, the results prove good generalization capability of our network. One strong benefit is that we don't need to retrain our network on different datasets to complete the reconstruction work.

4.6. Efficiency

We test our network on 4 GeForce GTX 1080 Ti GPUs in a Intel Xeon(R) CPU system with 32×2.10 GHz cores. We implement data preprocessing method with CUDA. In our experiments, we set the maximum number of point and ver-

Table 4. Generalization on Stanford 3D [25].The Chamfer Distances are in units of 10^{-6}. Surfaces used for distance criterias are all reconstructed at octree depth 9. Ours-S and Ours-D represent the models we trained on ShapeNet [3] and DTU [10] respectively.

Data	Accuracy(%)		CD Mean			CD RMS		
	Ours S	Ours D	ONet [17]	Ours S	Ours D	ONet [17]	Ours S	Ours D
Armadillo	98.2	97.8	93.46	0.028	**0.023**	168.59	0.131	**0.056**
Bunny	98.2	98.7	94.88	0.064	**0.057**	165.44	0.134	**0.087**
Dragon	98.0	98.0	40.69	0.053	**0.047**	74.99	0.208	**0.150**

Table 5. The time performance of our method. We report the first scale point number, first scale octree vertex number and triangle number in our results using million (M) as unit. The preprocessing time (Prep. time) includes octree construction time, downsampling time, tangent images precomputing time and batches computing time. The prediction time (Pred. time) is the time for loading partitioned points and vertices and predicting labels.

Number	Armadillo	Bunny	Dragon	stl_030	stl_034	stl_062
Point /M	2.16	0.361	1.71	2.43	2.01	2.19
Vertex /M	3.29	3.62	3.07	1.07	0.766	0.922
Triangle /M	1.18	1.52	1.16	0.42	0.31	0.36
Batch	109	73	77	59	88	86
Time /s	Armadillo	Bunny	Dragon	stl_030	stl_034	stl_062
Prep	19.4	8.86	16.6	10.9	9.34	9.99
Pred (1 GPU)	133	82.2	110.5	63.5	84.5	87.9
Pred (4 GPUs)	**50.5**	**27.3**	**38.2**	30.4	31.1	34.3
Total (1 GPU)	153	92.0	128	75.4	94.5	98.6
Total (4 GPUs)	**70.6**	**37.1**	**55.7**	42.3	41.1	45.0

tex in one batch as 300,000. As is illustrated in Table 5, our method can tackle millions of points and vertices in reasonable time [2]. As a learning-based method, the time consumption is generally acceptable. Since our network is able to perform predictions parallelly using multi-GPUs, our prediction procedure can be accelerated by more GPUs. As shown in the table, the predictions with 4 GPUs are about 2.6 times faster than 1 GPU. This reveals the efficiency potential of our method scaling to large datasets. When conditions permitting (more GPUs), it can be accelerated a lot.

5. Discussion and Conclusion

Our TSRNet has some advantages. One of the advantages is its strong scalability that allows dividing the input data and processing different parts parallelly. It has successfully reconstruct quality surfaces from point clouds with millions of points in reasonable time.

The other important advantage of our method is the construction of local geometry-aware features. The geometry-aware features are only related to local neighbor information and doesn't rely on any global shape information. Giving too much attention to global shape feature rather than just considering local geometry information would limit the

---

[2]DTU [10] contains a series of point clouds, here we just randomly list time consumption of stl_030, stl_034, stl_062. We do not consider the time of reading and writing files.

generalization capability among different shape types. Thus our method has good generalization capability and it does not need too much training data, which avoids overfitting.

In conclusion, we have successfully designed a scalable network for quality surface reconstruction from point clouds and make a significant breakthrough compared with existing state-of-the-art learning-based methods. We believe that it can inspire more researches in related directions, such as more efficient network architectures and more accurate large-scale 3D surface datasets, etc.

References

[1] Yan-Pei Cao, Zheng-Ning Liu, Zheng-Fei Kuang, Leif Kobbelt, and Shi-Min Hu. Learning to reconstruct high-quality 3d shapes with cascaded fully convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 616–633, 2018. 1, 3

[2] Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76. ACM, 2001. 3

[3] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2, 6, 7, 8

[4] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision (ECCV)*, pages 628–644. Springer, 2016. 6, 7

[5] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. 1996. 1, 2, 3

[6] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5868–5877, 2017. 1, 3

[7] Haoqiang Fan, Hao Su, and Leonidas J Guibas. 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 (CVPR)*, pages 605–613, 2017. 6, 7

[8] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 216–224, 2018. 1, 3

[9] Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. In *ACM Transactions on Graphics (TOG)*, volume 26, page 23. ACM, 2007. 2, 3

[10] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis eval-

uation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413. IEEE, 2014. 6, 7, 8

[11] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006. 3, 5

[12] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):29, 2013. 1, 3, 5, 6, 7

[13] David Levin. Mesh-independent surface interpolation. In *Geometric modeling for scientific visualization*, pages 37–49. Springer, 2004. 2, 3

[14] Yiyi Liao, Simon Donné, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2916–2925, 2018. 1, 2, 3, 6, 7

[15] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987. 1, 5

[16] 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

[17] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 1, 2, 3, 6, 7, 8

[18] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 1, 2, 3

[19] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017. 2, 3

[20] 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. 4

[21] 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 (CVPR)*, pages 3577–3586, 2017. 3

[22] 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. 1, 3

[23] Samuele Salti, Federico Tombari, and Luigi Di Stefano. Shot: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125:251–264, 2014. 4

[24] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for

unstructured multi-view stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 501–518. Springer, 2016. 1

[25] Stanford 3D. The stanford 3d scanning repository. http://graphics.stanford.edu/data/3Dscanrep, 2013. 2, 6, 7, 8

[26] 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 (CVPR)*, pages 3887–3896, 2018. 4, 5

[27] Gabriel Taubin. A signal processing approach to fair surface design. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 351–358. ACM, 1995. 5

[28] Greg Turk and James F O'Brien. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics (TOG)*, 21(4):855–873, 2002. 3

[29] Qingshan Xu and Wenbing Tao. Multi-scale geometric consistency guided multi-view stereo. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1