Neural Points: Point Cloud Representation with Neural Fields

Wanquan Feng¹ Jin Li² Hongrui Cai¹ Xiaonan Luo² Juyong Zhang¹*
¹University of Science and Technology of China ²Guilin University of Electronic Technology
{lcfwq@mail.ustc, 20032201014@mails.guet, hrcai@mail.ustc, luoxn@guet, juyong@ustc}.edu.cn

Abstract

In this paper, we propose Neural Points, a novel point cloud representation. Different from traditional point cloud representation where each point only represents a position or a local plane in the 3D space, each point in Neural Points represents a local continuous geometric shape via neural fields. Therefore, Neural Points can express more complex geometry shapes and thus have a stronger representation ability. Neural Points is trained with surface containing rich geometric details, such that the trained model has enough expression ability for various shapes. Specifically, we extract deep local features on the points and construct neural fields through the local isomorphism between the 2D parametric domain and the 3D local patch. In the final, local neural fields are integrated together to form the global surface. Experimental results show that Neural Points has powerful representation ability and demonstrate excellent robustness and generalization ability. With Neural Points, we can resample point cloud with arbitrary resolutions, and it outperforms state-of-the-art point cloud upsampling methods by a large margin.

1. Introduction

Point cloud, which is the most fundamental and popular representation of 3D scenes, has been widely used in many applications like 3D reconstruction [10, 13, 22, 30], virtual/augmented reality [9, 31] and autonomous driving [15, 33]. In the traditional representation of point cloud, each point only represents a position in the 3D space, and it can be further extended to represent a local plane if its normal vector is assigned. Therefore, the representation ability of point cloud is still limited by its resolution. Although point cloud upsampling methods [16, 17, 35, 40] have been proposed to improve its representation ability, their strategy is still a “discrete-to-discrete” manner and can not overcome the limitation of current point cloud representation.

In this work, we propose Neural Points, a novel point cloud representation, which significantly improves the representation ability. Different from traditional representation, each point of Neural Points encodes a local surface patch represented via neural fields. Specifically, each point-wise surface patch is represented as a local isomorphism between the 2D parametric domain and the 3D local surface patch, and the isomorphism is implicitly represented via neural fields. Thanks to its continuous nature, free of the limitation of finite resolution, powerful representation ability to complex geometric shapes, neural fields enable Neural Points several advantages over traditional discrete point cloud representation. Meanwhile, the trained model of neural fields is shared for all local patches, and thus the storage overhead of Neural Points is quite small.

Unlike some existing methods which represent the whole surface model via one neural field [24, 38], both the neural fields and the point features passed into Neural Points are all local. With this design, our Neural Points representation shows several advantages including excellent ability to express details, strong generalization ability, and low need for training data. Specifically, we employ local neural fields to construct the continuous bijective mapping between the 2D parametric domain and the 3D local surface patch, based on the fact that 3D local patch on the 2D manifold is isomorphic to a 2D simply connected disk. With the neural field, each 3D local patch can be viewed as a parametric surface defined in a 2D parametric domain. To this end, we employ an encoder to extract point-wise local features and take the local feature as a part of the input passed into the neural field network. With the local shape information, the neural field can represent the surface patch well, and the trained network model is shared by all local patches of all 3D models. In the final, we design an integration strategy to integrate all the local neural fields together to form the final global shape. The model of Neural Points is trained with geometric surface in high resolution with rich geometric details, such that the trained Neural Points model can represent not only low-frequency geometric structure but also high-frequency geometry details.

As a natural application of Neural Points, we conduct the point cloud upsampling task based on the proposed
Neural Points representation. Given a low-resolution point cloud, the target of point cloud upsampling is to predict the upsampled point cloud with higher resolution such that it can better capture the geometric features of the underlying continuous surface. Existing point upsampling methods [16, 17, 35, 40] usually employ a “discrete-to-discrete” manner, and each trained model only works for one upsampling factor. Dissimilar to these methods, we convert the input discrete point cloud representation to continuous Neural Points representation, and then arbitrary resolution points can be sampled accordingly.

The experimental results show that our proposed Neural Points can produce high-quality continuous surfaces and overcome the limitation of point resolution well. For the point cloud upsampling task, our Neural Points representation achieves much better performance than traditional point cloud representation. In summary, the contributions of this work include:

- We propose Neural Points, a novel representation for point cloud, whose representation ability greatly outperforms the traditional point cloud representation, with low storage overhead.
- We employ the neural implicit functions along with the deep local features of surface patches to represent the local neural fields, and design an integration strategy to form the final global shape.
- Evaluation results demonstrate that the Neural Points representation has excellent robustness and generalization ability for various inputs. Application of Neural Points to point cloud upsampling task achieves much better performance over existing methods.

2. Related Works

Neural Implicit Function. Neural implicit representation has recently shown promising advantages in many applications over discrete scene representations, such as point clouds, meshes and voxel grids, owing to its continuous nature and free of the limitation of finite resolution. It has also achieved great performance in many applications including 3D shape representation [4, 20, 24], novel view synthesis [21, 32] and multi-view reconstruction [23, 38].

For 3D shape representation, DeepSDF [24] and ONet [20] proposed to use neural implicit function to predict the signed distance and occupancy probability, respectively. However, these conventional frameworks are weak in representing complex shapes. To alleviate this issue, some recent works focus on geometric detail preservation and local shape reduction. IF-Nets [5] extracted learnable multi-scale features, which encodes local and global properties of 3D shape. LGCL [37] represented 3D shape as zero-surfaces with local latent codes, leading each local SDF responsible for a part of the shape. Due to local feature learning, these methods represent higher-quality details than common methods which use a single vector to encode a shape. In our method, we also devise a novel and effective strategy to extract deep local features from point cloud.

Neural implicit functions have also been used to represent 2D images. LIIF [3] took a pixel coordinate and its surrounding feature as input to predict the corresponding RGB value. It demonstrates excellent super-resolution effect, building a bridge between discrete and continuous representation in the 2D domain. Different from regular 2D image, irregular point cloud is a discrete representation of 3D surface embedded in 2D manifold. To handle the irregular representation, we construct the mapping between the irregular local surface patch and the regular 2D domain.

Optimization Based Point Upsampling. In optimization based point upsampling methods, shape priors like global structures and local smoothness are formulated as objective energy to constrain the upsampling results. Alexa et al. [2] employed Voronoi diagram and interpolated points at vertices of the Voronoi diagram in local tangent space as upsampled points. Then Lipman et al. [19] proposed a parameter-free method based on the locally optimal projection operator (LOP) for point sampling and surface reconstruction. Later, weighted LOP [11] conducted an iterative normal estimation process to consolidate the upsampled point. EAR [12] sampled points away from edges and progressively approaching edges and corners. Then a point-set consolidation method [36] proposed to fill large holes and complete missing regions.

Learning Based Point Upsampling. Considering that the point upsampling task is an ill-posed problem, learning priors from dataset is a natural way to tackle this challenging inverse task. In recent years, point-based network structures [7, 14, 18, 26, 27, 34] have been successfully employed to solve point cloud related tasks and achieved state-of-the-art performance. PU-Net [40] was the first point upsampling network, which can capture both local and global information by multi-level feature aggregation. EC-Net [39] minimized the point-to-edge distances to keep the edge structure of the point clouds. MPU [35] was a progressive upsampling method that achieved great performance for large upsampling factors. PU-GAN [16] was the first work to apply the GAN [8] structure to synthesize uniformly distributed points. While the above mentioned works are mainly focused on the network design, PUGeo-Net [29] utilized local differential geometry constraints to improve the upsampling results. Recently, PU-GCN [28] improved the performance of point cloud upsampling by proposing a novel NodeShuffle module. Dis-PU [17] employed a global refinement module at the end of the pipeline instead of using a single upsampling module. However, all the above methods adopt the strategy to directly predict a denser point
cloud patch from the input sparse point cloud patch. Unlike all these methods, we utilize neural fields to represent the high-resolution surface, which has a more powerful representation ability and can then be sampled in arbitrary resolutions.

3. Neural Points

3.1. Overview: Point based Representation

Point cloud $\mathcal{X} = \{x_i \in \mathbb{R}^3\}_{i=1}^I$ is the discrete representation of its underlying continuous surface $\mathcal{S}$. For the traditional point cloud representation, where each point only represents a 3D position, its representation ability totally depends on its resolution. Therefore, one direct strategy to improve its representation ability is to do point cloud upsampling:

$$\mathcal{S} \xrightarrow{\text{Discretize}} \{x_i\}_{i=1}^I \xrightarrow{\text{Upsample}} \{x_i\}_{i,r=1}^I \subset \mathcal{S},$$

which was studied in the point cloud upsampling works [17, 40]. However, the upsampling manner in Eq. (1) is discrete-to-discrete, where the upsampled result is still discrete and limited by the resolution.

In this work, we propose Neural Points, a novel point cloud representation, which has better representation ability than the traditional point cloud. Neural Points representation employs the discrete-to-continuous strategy, which is totally different from the form of Eq. (1). Given the input point cloud $\mathcal{X} = \{x_i\}_{i=1}^I$, we describe the underlying continuous surface

$$\mathcal{S} \xrightarrow{\text{Discretize}} \{x_i\}_{i=1}^I \xrightarrow{\text{Neural Points}} \mathcal{S'} \approx \mathcal{S},$$

where $\mathcal{S'}$ is a continuous surface represented by Neural Points. With the form in Eq. (2), our proposed framework can overcome the limitation of point cloud resolution and achieve arbitrary-factored point cloud sampling on $\mathcal{S'}$. In the following, we will describe the algorithm details of our Neural Points representation.

3.2. Pipeline

The whole pipeline is given in Fig. 1. (1) Given the input point cloud, we first construct local neural fields for each local neural patch, which is based on local parameterization; (2) The local neural fields are integrated together to form the global shape; (3) With the constructed continuous neural representation, we can resample an arbitrary number of points. In the following, we introduce the details of each component.

Local Neural Fields. We employ local neural fields to describe the underlying continuous surface. The input point cloud is still denoted as $\mathcal{X} = \{x_i\}_{i=1}^I$. Like in [29], we take $\{x_i\}_{i=1}^I$ as the center points to divide the surface into overlapping local patches $\{\mathcal{P}_i\}_{i=1}^I \subset \mathbb{R}^3$. At each point $x_i$, the 3D local patch $\mathcal{P}_i$ is isomorphic to the 2D parametric domain $\mathcal{D} \subset \mathbb{R}^2$ (we use $\mathcal{D} = [-1, 1]^2$ in our work), which means that we can construct a bijective mapping between them:

$$\phi_i : \mathcal{D} \to \mathcal{P}_i, \quad \psi_i : \mathbb{R}^3 \to \mathcal{D},$$

where $\psi_i |_{\mathcal{P}_i} = \phi_i^{-1}$. The illustration of the neural field is shown in Fig. 2. Given any 2D sampling point $(u_r, v_r) \in \mathcal{D}$, we can compute $x_i^r = \phi_i(u_r, v_r) \in \mathcal{P}_i$ as its corresponding 3D sampling point. Similarly, for any 3D point $x_i^r \in \mathcal{P}_i$, we can compute its corresponding 2D coordinate $(u_r, v_r) = \psi_i(x_i^r) \in \mathcal{D}$. Furthermore, we can compute the point normal $n_{x_i^r}$ at point $x_i^r$ via:

$$n_{x_i^r} = \nabla_u \phi_i \times \nabla_v \phi_i,$$

and then normalize it to unit length.

Neural Fields Integration. Although the local neural fields $\{\phi_i\}_{i=1}^I$ can construct correspondence between 2D parametric domain and 3D local patches, different local neural
fields are defined in different local coordinate systems. As a result, we need to integrate them together to obtain a globally continuous neural field based surface.

For \( \forall x \in \mathbb{R}^3 \), we aim to pull \( x \) onto the neural fields based surface. We find the nearest center points in \( \{x_i\}_{i=1}^I \) to \( x \) and denote the set of indexes of the neighbour points as \( N(x) \). For \( \forall k \in N(x) \), a combination weight is computed according to the distance from \( x \) to \( x_k \):

\[
w_k = e^{-\alpha_1 \|x - x_k\|^2}, \forall k \in N(x).
\]  

(3)

With \( w_k \), the point on the neural fields based surface can be computed by:

\[
\rho(x) = \left( \sum_{k \in N(x)} w_k \cdot \phi_k \circ \psi_k(x) \right) / \left( \sum_{k \in N(x)} w_k \right), \forall x \in \mathbb{R}^3.
\]  

(4)

Specifically, the input point \( x \) is mapped to 2D parametric domain through the neighbour neural fields as \( \psi_k(x) \in \mathcal{D} \), and then map the 2D point back to 3D coordinate as \( \phi_k \circ \psi_k(x) \in \mathcal{P}_k \). In this way, a neural fields based continuous surface is constructed.

The normal at point \( \rho(x) \) is computed as:

\[
n_{\rho(x)} = \left( \sum_{k \in N(x)} w_k \cdot n_{\phi_k \circ \psi_k(x)} \right) / \left( \sum_{k \in N(x)} w_k \right), \forall x \in \mathbb{R}^3,
\]  

(5)

and the combination vector is normalized to unit length. Note that the orientations of \( \{n_{\phi_k \circ \psi_k(x)}\}_{k \in N(x)} \) are adjusted to be along the same direction before computing the weighted sum.

**Point Cloud Sampling.** We can also resample the point cloud from the neural fields represented continuous surface. Considering that the point cloud sampling process is essentially applied on the 2D manifold, we start the sampling operation in the 2D parametric domain \( \mathcal{D} \).

We sample points uniformly in the 2D parametric domain \( \mathcal{D} \), and the 2D sampled points are mapped onto the 3D local patch. Specifically, for center point \( x_i \), we uniformly sample \( R \) points \( \{(u_r, v_r) \in \mathcal{D} \}_{r=1}^R \) and then map them to 3D as \( \{x_i^r \in \mathcal{P}_i\}_{r=1}^R \). For the whole input point cloud, we obtain the union of sampled points from all patches:

\[
\mathcal{X}_R = \bigcup_{i=1}^I \{x_i^r \in \mathcal{P}_i\}_{r=1}^R = \{x_i^* \in \mathcal{P}_i\}_{i=1}^I.
\]

Then we can uniformly sample \( J \) points from \( \mathcal{X}_R \):

\[
\mathcal{Y}^* = \{y_j^* \in \bigcup_{i=1}^I \mathcal{P}_i\}_{j=1}^J
\]

where the value of \( J \) is arbitrary in the training and inferring stages. Then we pull \( y_j^* \) onto the Neural Points surface with Eq. (4) as:

\[
\mathcal{Y} = \{y_j = \rho(y_j^*)\}_{j=1}^J.
\]

### 3.3. Network Structure

In this part, we introduce the network structures, including the local feature extraction and the local neural fields.

**Local Feature Extraction.** As discussed above, for each center point \( x_i \), we extract a local feature as a part of the input into the neural field \( \phi_i \), which we denote as \( c_i \). For each \( x_k \), we first extract its neighbour points \( \{x_k\}_{k \in N(x_i)} \) and decentralize to \( \{x_k - x_i\}_{k \in N(x_i)} \). Then we apply the DGCNN [34] backbone to extract feature on the point set with \( |N(x_i)| \) points. Specifically, we employ several EdgeConv layers with dynamic graph update. The features of each layer are concatenated and then passed into another EdgeConv layer and max-pooling layer to get \( c_i^* \). For each \( i \), we concatenate \( c_i^* \) with the local pooling of its neighbors to get the final local feature:

\[
c_i = c_i^* \oplus \text{MaxPool}\{c_k^*\}_{k \in N(x_i)},
\]

where \( \oplus \) denotes the concatenation operation.

**Structure of Neural Fields.** We employ the MLP-based network equipped with the ReLU activation layers to represent \( \phi_i \). The implicit functions are shared by all patches of all 3D models, and they are denoted as \( \Psi \). As we mentioned above, the input of \( \Psi \) should contain the local shape information involved in \( c_i \) and the 2D parametric coordinates as the querying point.

For the 2D querying coordinates passed into \( \phi_i \), we apply position encoding as the design in [21] and denote the position encoding function as \( \gamma \). The position code is concatenated with the local feature as the input of the neural implicit function. Specifically, we can formulate \( \Psi \) as:

\[
\Phi((u_r, v_r), c_i, x_i) = x_i + \theta_{\Phi}(\gamma(u_r, v_r) \oplus c_i),
\]

where \( \theta_{\Phi} \) is the MLP in \( \Phi \). Naturally, for each \( i \), we can formulate \( \phi_i(\cdot) = \Phi(\cdot, c_i, x_i) \).

Another key function is \( \psi_i \), who is the extension of \( \phi_i^{-1} \). In our implementation, \( \psi_i \) is defined as:

\[
\psi_i(x) = \phi_i^{-1}(\text{Proj}(x, \phi_i(D))), \forall x \in \mathbb{R}^3,
\]  

(6)

where we use \( \text{Proj}(x, \phi_i(D)) \) to denote the nearest point in \( \phi_i(D) \) to \( x \). The advantage of this definition is that we can formulate \( \phi_i \circ \psi_i \) as:

\[
\phi_i \circ \psi_i(x) = \text{Proj}(x, \phi_i(D)), \forall x \in \mathbb{R}^3,
\]  

(7)

which is used in Eq. (4).

At last, we explain our projection process in Eq. (6) and Eq. (7). In our implementation, we use the approximation for \( \phi_i(D) \) as:

\[
\phi_i(D) \approx \{x_i^r\}_{r=1}^R.
\]

Then, the projection operation should be formulated from a 3D point and a point set. For the convenience of description,
we consider a 3D point denoted as \( \mathbf{p} \) and a point set denoted as \( \mathcal{Q} = \{ \mathbf{q}_i \}_{i=1}^T \). We find the index set of neighbor points of \( \mathbf{p} \) in \( \mathcal{Q} \) and denote it as \( \mathcal{N}(\mathbf{p} ; \mathcal{Q}) \). The projection from \( \mathbf{q} \) to \( \mathcal{Q} \) is formulated as:

\[
\text{Proj}(\mathbf{p}, \mathcal{Q}) = \left( \sum_{k \in \mathcal{N}(\mathbf{p} ; \mathcal{Q})} w_k \cdot \mathbf{q}_k \right) / \left( \sum_{k \in \mathcal{N}(\mathbf{p} ; \mathcal{Q})} w_k \right),
\]

where \( w_k \) is computed as:

\[
w_k = e^{-\alpha_2 \| \mathbf{p} - \mathbf{q}_k \|^2_2}, \quad \forall k \in \mathcal{N}(\mathbf{p} ; \mathcal{Q}).
\]

Similar to Eq. (5), its normal \( \mathbf{n}_{\text{Proj}(\mathbf{p}, \mathcal{Q})} \) can be obtained.

### 3.4. Loss Function

In this part, we introduce the loss terms, including the constraints on surface shape, point normal, and the integration quality. The total loss is:

\[
\mathcal{L} = \mathcal{L}_{\text{shape}} + \omega_1 \cdot \mathcal{L}_{\text{nor}} + \omega_2 \cdot \mathcal{L}_{\text{int}}.
\]

Point clouds with higher resolution are used as ground truth in our current implementation, and other shape representations can also be used accordingly. We denote the ground truth point cloud as \( \mathcal{X}_R = \{ \mathbf{x}_i \}_{i=1}^L \) and denote the normal of \( \mathbf{z}_i \) as \( \mathbf{n}_{\mathbf{z}_i} \). As described in Sec. 3.2, the output of the framework can be summarized as \( \mathcal{X}_R^i = \{ \mathbf{x}_i^r; \mathbf{n}_{\mathbf{x}_i^r} \}_{i,r=1}^{L,R} \) and \( \mathcal{Y} = \{ \mathbf{y}_j; \mathbf{n}_{\mathbf{y}_j} \}_{j=1}^T \).

The supervision is discrete, but \( \mathcal{X}_R \) and \( \mathcal{Y} \) should be viewed as arbitrary sampling from continuous surface. To supervise a continuous surface with the discrete supervision, we do not employ the Chamfer loss which is based on closest point searching, though it was widely used in the previous point upsampling works [17, 28]. Instead, we employ the projection strategy described in Eq. (8). Generally, for two point clouds \( \mathcal{P} = \{ \mathbf{p}_s \}_{s=1}^S \) and \( \mathcal{Q} = \{ \mathbf{q}_t \}_{t=1}^T \), we define their distance as:

\[
d(\mathcal{P}, \mathcal{Q}) = \frac{1}{S} \sum_{s=1}^S \| \mathbf{p}_s - \text{Proj}(\mathbf{p}_s, \mathcal{Q}) \|^2_2.
\]

We also define their difference of point-wise normal as:

\[
d_n(\mathcal{P}, \mathcal{Q}) = \frac{1}{S} \sum_{s=1}^S \| \mathbf{n}_{\mathbf{p}_s} - \mathbf{n}_{\text{Proj}(\mathbf{p}_s, \mathcal{Q})} \|^2_2.
\]

To constrain \( \mathcal{X}_R \) and \( \mathcal{Y} \) to be close to \( \mathcal{Z} \), the loss term is:

\[
\mathcal{L}_{\text{shape}} = d(\mathcal{X}_R, \mathcal{Z}) + d(\mathcal{Z}, \mathcal{X}_R) + d(\mathcal{Y}, \mathcal{Z}) + d(\mathcal{Z}, \mathcal{Y}).
\]

To supervise the point-wise normal, the loss term is:

\[
\mathcal{L}_{\text{nor}} = d_n(\mathcal{X}_R, \mathcal{Z}) + d_n(\mathcal{Z}, \mathcal{X}_R) + d_n(\mathcal{Y}, \mathcal{Z}) + d_n(\mathcal{Z}, \mathcal{Y}).
\]

Furthermore, we also employ a loss term for the integration quality. The overlap shape from neighbour neural fields should be identical, and the constructed global neural points surface should cover the input surface. Based on these requirements, we design the loss term as:

\[
\mathcal{L}_{\text{int}} = \sum_{j=1}^J \sum_{k \in \mathcal{N}(\mathbf{y}_j; \mathcal{X})} \| \mathbf{y}_j - \text{Proj}(\mathbf{y}_j; \{ \mathbf{x}_k^r \}_{r=1}^R) \|^2_2.
\]

### 4. Experiments

In this section, we give the implementation details, ablation studies, results, comparisons and the test on generalization and robustness.

#### 4.1. Implementation Details

**Dataset.** We train and test our model on Sketchfab [1] dataset collected by PU-Geo-Net [29], which contains 90 training and 13 testing models with rich geometry features. We train all the comparison methods with the same dataset for fair comparison. Similar to other point cloud upsampling methods, we employ Poisson disk sampling [6] algorithm to extract points on the models to obtain the input and ground truth of the upsampling algorithm. Specifically, we extract 10,000 points as the whole input point cloud and extract 40,000 and 160,000 points as the whole ground truth point cloud for 4× and 16× experiments. We also follow the previous point upsampling works to extract some anchor points and the neighboring local parts (subsets of the whole point cloud) as the input of the network instead of passing the whole model into the network. For all, we choose 1,000 anchors on each training model and 114 anchors on each testing model. During training, the point number of the input point clouds is set as 256.

To test the effectiveness of our trained model, we not only test on the testing set of Sketchfab, but also test on more unseen datasets without retraining our network. We test all the methods on dataset collected by PU-GAN [16], where the number of testing models is 27. The shape of the testing models in this dataset is relatively simple, and thus we extract 2,000 points on each model as the whole input point cloud. In addition to the synthetic data mentioned above, we further test on real captured data. We also evaluate our method on point clouds captured by depth sensor on iPhone X.

**Experimental Setting.** All the inputs passed into the network are normalized into the 3D unit ball. In all our experiments, we set \( R = [4 \cdot J / I] \). In the local feature extracting network, the number of neighboring point set is 10. For all the \( \text{Proj}(., .) \) operations, we set the number of neighboring points as 4. For the loss terms, \( \omega_1 \) and \( \omega_2 \) are set as 0.01 and 0.3, respectively. \( \alpha_1 \) in Eq. (3) and \( \alpha_2 \) in Eq. (9) are set as \( 10^2 \) and \( 10^3 \) respectively to compute the exponential weights. We employ 5 convolution layers in the DGCNN backbone and 3 linear layers in the MLP of \( \Phi \). We set the
batch size as 8 and train the network with 25,000 iterations for all. The learning rate starts at 0.01, and is multiplied by 0.5 every 1,250 iterations. For all the comparison methods, we train them with the same settings as our own method. The model is trained with PyTorch [25]. All the training and testing is conducted on a workstation with four 32G V100 GPUs, 32 Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz, and 128GB of RAM. Our trained network model can be applied to all point clouds. The storage overhead of Neural Points is to store the pre-trained model, whose total size is only 2.53MB (1.35MB for the local feature extracting network and 1.18MB for the Neural field MLP network.).

Evaluation Metric. Similar to recent point cloud upsampling works [17, 28], we employ Chamfer Distance (CD), Hausdorff Distance (HD), and Point-to-Surface (P2F) as the metrics. For all the metrics, the smaller the metric, the better the quality of the results.

4.2. Ablation Study

We conduct ablation studies to show how each component influences the final result. Specifically, we use the Sketchfab dataset as our benchmark. We mainly design the ablation study for the local feature extractor structure, the loss terms and the integration process. The quantitative results of the ablation study experiment are shown in Tab. 1. In the experiment of the 1-st row, we apply a DGCNN backbone for the whole input point cloud instead of the local KNN point set. We can see the quantitative result is quite poor, which verifies that it is necessary to utilize a shared backbone for KNN point sets and limit the size of the receptive field in the feature extractor. The 2-3-th rows show different loss term settings, where we remove the $L_{nor}$ and $L_{int}$ terms, respectively. We can see that removing any of the two terms would make the result to be not as good as the full model. Among them, removing the $L_{int}$ loss term will greatly reduce the algorithm performance. At last, we remove the integration process and all related algorithm settings, and show the results in the 4-th row. We can see that the result without the integration process is much worse than the full model.

| Ablation Settings | CD (×10⁻³) | HD (×10⁻³) | P2F (×10⁻³) |
|-------------------|-------------|-------------|-------------|
| w/o input local KNN | 2.49        | 9.54        | 3.03        |
| w/o $L_{nor}$     | 0.70        | 3.46        | 0.75        |
| w/o $L_{int}$     | 0.77        | 4.09        | 0.81        |
| w/o integration   | 0.83        | 5.64        | 0.88        |
| full model        | **0.66**    | **3.32**    | **0.69**    |

Table 1. Results of the ablation study, with metrics CD(×10⁻³), HD(×10⁻³), and P2F(×10⁻³).

4.3. Results and Comparisons

We compare our method with five point upsampling methods: PU-Net [40], PU-GAN [16], PU-GCN [28], Dis-PU [17] and PUGeo-Net [29]. Among them, PU-Geo-net, PU-GCN and Dis-PU are recently proposed methods that achieve the state-of-the-art performance. All the comparison methods are in a discrete-to-discrete manner, and are trained with the same training set with our method. For all methods, we train two versions: 4x and 16x, respectively. In each version, we train all the methods with ground truth data in the corresponding resolution. For fair comparison, we share the same experiment settings (including batch size, iteration numbers, optimizer, learning rate, etc) among all the methods.

The statistical results and comparisons of 4x and 16x upsampling, with metrics CD(×10⁻³), HD(×10⁻³), and P2F(×10⁻³).

| Factor | Method   | Sketchfab | PU-GAN |
|--------|----------|-----------|--------|
|        |          | CD | HD | P2F | CD | HD | P2F |
| 4x     | PU-Net   | 5.93 | 4.98 | 4.51 | 23.61 | 13.91 | 10.02 |
|        | PU-GAN   | 3.30 | 3.45 | 3.61 | 16.79 | 9.36  | 7.04  |
|        | PU-GCN   | 2.85 | 3.21 | 2.79 | 14.74 | 11.97 | 6.36  |
|        | Dis-PU   | 2.61 | 3.25 | 2.67 | 13.79 | 11.83 | 7.14  |
|        | PUGeo-Net| 2.28 | **2.10** | 1.04 | 11.26 | 3.54  | 2.14  |
|        | Ours     | **2.17** | **2.46** | **0.93** | **8.17** | **3.08** | **1.59** |
| 16x    | PU-Net   | 5.25 | 5.82 | 5.99 | 20.70 | 15.49 | 12.16 |
|        | PU-GAN   | 2.97 | 3.99 | 3.76 | 11.89 | 10.81 | 7.48  |
|        | PU-GCN   | 2.35 | 3.84 | 3.02 | 11.37 | 12.69 | 6.95  |
|        | Dis-PU   | 2.36 | 3.79 | 3.31 | 12.75 | 13.65 | 8.09  |
|        | PUGeo-Net| 0.83 | 3.50 | 0.97 | 3.58  | 7.14  | 1.94  |
|        | Ours     | **0.66** | **3.32** | **0.69** | **3.35** | **6.52** | **1.49** |

Table 2. Results and comparisons for 4x and 16x upsampling, with metrics CD(×10⁻³), HD(×10⁻³), and P2F(×10⁻³).

4.4. Generalization and Robustness

Generalization to Unseen Data. We test on another dataset
which was commonly used in previous dataset, the dataset
collected by PU-GAN [16]. Without retraining, we apply
the trained models on the PU-GAN dataset again and show
the visual and quantitative results. The results and compar-
isons are in Tab. 2 and Fig. 4. We can see that our model
still achieves the best performance on both of quantitative
results and visualization results, proving the good general-
ization ability of the Neural Points representation.

**Robustness to Large Upsampling Factor.** We explore the
upsampling effect of different methods with very large up-
sampling factors. Notice that all the testing methods in-
cluding our own method are trained with a $16 \times$ supervision
signal. We adjust the \( N \) in our method as \( 256 \cdot I \) to obtain the \( 256 \times \) result. For other methods, we apply the trained \( 16 \times \) model for two times to obtain the \( 256 \times \) results. The results are shown in Fig. 5. The visualization results of \( 16 \times \) upsampling by different methods do not have too much difference, while the results of \( 256 \times \) upsampling are quite different. After applying the model for two times, the results of PU-GAN, PU-GCN, Dis-PU and PUGeo-Net contain some flaws that are easy to observe. At the same time, our result still keeps in a good quality. Since Neural Points represents the surface in a continuous and resolution-free way, our result will not be affected by large sampling factors.

### Robustness to Real Captured Data

We further compare our method with two most representative methods (Dis-PU [17] and PUGeo-Net [29]) on real captured data. Specifically, we capture depth images of human face with the depth sensor equipped on iPhone X. Taking the point cloud converted from the depth image as input, we apply upsampling to the point clouds with different methods. As shown in Fig. 6, our method performs quite well, even the scanned point clouds contain lots of noise and bumpy local regions. In contrast, the results generated by Dis-PU [17] fail to preserve the geometry features and contain some artifacts. The results of PUGeo-Net [29] preserve some geometric features of the input point cloud, but we can still observe some artifacts in their results. This test verifies the robustness and effectiveness of Neural Points to real captured data.

### 5. Conclusion&Limitation

We proposed Neural Points, where each point represents a local continuous geometric shape via neural fields instead of only a position or a local plane in the 3D space. Neural Points can express much more complex details and thus have a stronger representation ability than the traditional point cloud. We trained Neural Points with surface containing rich geometric details, such that the trained model has enough expression ability for various shapes. Specifically, we extracted deep local features on the points and constructed neural fields through the local isomorphism between the 2D parametric domain and the 3D local patch. The final global continuous surface is obtained by integrating the neural fields. The powerful representation ability, robustness and generalization ability of Neural Points have been verified by extensive experiments. The great performance by applying it to point cloud upsampling task further verifies its nice properties.

Currently, we only utilize the local geometry shape to train Neural Points, which might limit its applications to more broad areas. In the future, we plan to utilize other modalities like corresponding color images and textures, or global semantic structure to further improve its representation ability.
References

[1] Sketchfab. https://sketchfab.com. 5, 7

[2] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Cláudio T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 9(1):3–15, 2003. 2

[3] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8628–8638, 2021. 2

[4] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5939–5948, 2019. 2

[5] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6968–6979, 2020. 2

[6] Massimiliano Corsini, Paolo Cignoni, and Roberto Scopigno. Efficient and flexible sampling with blue noise properties of triangular meshes. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 18(6):914–924, 2012. 5

[7] Wanquan Feng, Juyong Zhang, Hongrui Cai, Haofei Xu, Junhui Hou, and Hujun Bao. Recurrent multi-view alignment network for unsupervised surface registration. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10297–10307, 2021. 2

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014. 2

[9] Robert Held, Ankit Gupta, Brian Curless, and Maneesh Agrawala. 3d puppetry: a kinect-based interface for 3d animation. In *ACM Symposium on User Interface Software and Technology (UIST)*, pages 423–434, 2012. 1

[10] Hugues Hoppe, Tony DeRose, Tom Duchamp, John Alan McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 71–78, 1992. 1

[11] Hui Huang, Dan Li, Hao Zhang, Uri M. Ascher, and Daniel Cohen-Or. Consolidation of unorganized point clouds for surface reconstruction. *ACM Transactions on Graphics (TOG)*, 28(5):176, 2009. 2

[12] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri M. Ascher, and Hao (Richard) Zhang. Edge-aware point set resampling. *ACM Transactions on Graphics (TOG)*, 32(1):9:1–9:12, 2013. 2

[13] Michael M. Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(3):29:1–29:13, 2013. 1

[14] Artem Komarichev, Zichun Zhong, and Jing Hua. A-CNN: annularly convolutional neural networks on point clouds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7421–7430, 2019. 2

[15] Alex H Lang, Souabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12697–12705, 2019. 1

[16] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Peng-Ann Heng. PU-GAN: A point cloud upsampling adversarial network. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7202–7211, 2019. 1, 2, 5, 6, 7, 8

[17] Ruihui Li, Xianzhi Li, Peng-Ann Heng, and Chi-Wing Fu. Point cloud upsampling via disentangled refinement. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 344–353, 2021. 1, 2, 3, 5, 6, 8

[18] Yangyan Li, Rui Bu, Mengchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 828–838, 2018. 2

[19] Yaron Lipman, Daniel Cohen-Or, David Levin, and Hil-lel Tal-Ezer. Parameterization-free projection for geometry reconstruction. *ACM Transactions on Graphics (TOG)*, 26(3):22, 2007. 2

[20] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4460–4470, 2019. 2

[21] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, pages 405–421, 2020. 2, 4

[22] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pusmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew W. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136, 2011. 1

[23] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3504–3515, 2020. 2

[24] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deep sdf: Learning continuous signed distance functions for shape representation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019. 1, 2

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning...
library. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, volume 32, pages 8024–8035, 2019.

[26] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017.

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

[28] Guocheng Qian, Abdulellah Abualshour, Guohao Li, Ali K. Thabet, and Bernard Ghanem. PU-GCN: point cloud upsampling using graph convolutional networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11683–11692, 2021.

[29] Yue Qian, Junhui Hou, Sam Kwong, and Ying He. Pugeonet: A geometry-centric network for 3d point cloud upsampling. In *European Conference on Computer Vision (ECCV)*, volume 12364, pages 752–769, 2020.

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

[31] José Miguel Santana, Jochen Wendel, Agustín Trujillo, José Pablo Suárez, Alexander Simons, and Andreas Koch. Multimodal location based services—semantic 3d city data as virtual and augmented reality. In *Progress in location-based services (LBS)*, pages 329–353. 2017.

[32] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 1119–1130, 2019.

[33] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark E. Campbell, and Kilian Q. Weinberger. Pseudolidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8445–8453, 2019.

[34] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38(5):146:1–146:12, 2019.

[35] Yifan Wang, Shihao Wu, Hui Huang, Daniel Cohen-Or, and Olga Sorkine-Hornung. Patch-based progressive 3d point set upsampling. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5958–5967, 2019.

[36] Shihao Wu, Hui Huang, Minglun Gong, Matthias Zwicker, and Daniel Cohen-Or. Deep points consolidation. *ACM Transactions on Graphics (TOG)*, 34(6):176:1–176:13, 2015.

[37] Shun Yao, Fei Yang, Yongmei Cheng, and Mikhail G Mozharov. 3d shapes local geometry codes learning with sdf. In *IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 2110–2117, 2021.

[38] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[39] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Ec-net: An edge-aware point set consolidation network. In *European Conference on Computer Vision (ECCV)*, volume 11211, pages 398–414, 2018.

[40] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2790–2799, 2018.
Neural Points: Point Cloud Representation with Neural Fields
Supplementary Material

Figure 7. The visualization of the local patches of our Neural Points representation. We show some examples of local point clouds in the first row, where the center point is colored red. To clearly show the underlying local shape, we also show the underlying surface from which we extract the local points. The corresponding local patches generated from Neural Fields in our algorithm are shown in the second row. Each column shows different local parts in our testing set.

This supplementary material shows more results and analysis which are not included in the paper due to limited space, including the visualization of the local Neural Fields, results with more sampling factors, and the qualitative results of ablation study.

A. Visualization of Neural Field Patches

To clearly show the Neural Fields of each point, we visualize some local patches generated from the Neural Fields in Fig. 7. We show the local point cloud where we extract local features in the first row and then show their corresponding local Neural Field patches in the second row. The center points are rendered in red. In the first row, to clearly show the local underlying shape, we show the underlying surface from which we extract the local points. In the second row, we zoom in the local parts and show the generated Neural Field patches. We can see that the generated patches can cover a local area smoothly and fit well with the shape of the local point cloud. Each column shows different local parts extracted from the models in our testing set. For different local shapes, the Neural Field can obtain satisfactory local patches, which contributes a lot to the overall expressive ability of the Neural Points representation.

B. Results with More Sampling Factors

As stated in the paper, our trained model can be applied with arbitrary sampling factors. In this section, we show results with more sampling factors, and non-integer factors are also supported in our model. In Fig. 8, we show some examples. Specifically, we upsample the input points with factors $1.7, 3.3, 8.4,$ and $15.1$. We can see that our model can generate good results for various factors within a large range. Due to the limited space, we only show these factors, but our model also supports other factors. We strongly recommend watching our video, where we show the results with continuously varying factors.

C. Visualization of Ablation Study

Except for the quantitative experiment in the paper, we also show the visualization results of the ablation study in Fig. 9. If we apply the backbone for the whole point cloud instead of the local KNN point set, the training effect would be quite poor, and the visual results look messy. If we remove the loss term for normal $\mathcal{L}_{nor}$, the training effect would
not be as good as the full model. The visual results look not smooth enough. If we remove the integration term $L_{\text{cat}}$, the results would contain some holes. Further, when we remove all the processes related to integration, the results not only have holes but also suffer from some splicing dislocations. Compared to these algorithm settings, our full model can generate satisfying results.
Figure 9. The visualization results of our ablation study experiments.