This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Neural Compression for 3D Geometry Sets

Siyu Ren[1] Junhui Hou[1*] Weiyao Lin[2] Wenping Wang[3]
[1]City University of Hong Kong [2]Shanghai Jiao Tong University [3]Texas A&M University
siyuren2-c@my.cityu.edu.hk, jh.hou@cityu.edu.hk, wylin@sjtu.edu.cn, wenping@cs.hku.hk

Abstract

We present NeCGS, the first neural compression paradigm, which can compress a geometry set encompassing thousands of detailed and diverse 3D mesh models by up to 900 times with high accuracy and preservation of detailed geometric structures. Specifically, we first propose TSDF-Def, a new implicit representation that is capable of ***accurately*** *representing irregular 3D mesh models with various structures into regular 4D tensors of* ***uniform*** *and* ***compact*** *size, where 3D surfaces can be extracted through the deformable marching cubes. Then we construct a quantization-aware auto-decoder network architecture to regress these 4D tensors to explore the local geometric similarity within each shape and across different shapes for redundancy removal, resulting in more compact representations, including an embedded feature of a smaller size associated with each 3D model and a network parameter shared by all models. We finally encode the resulting features and network parameters into bitstreams through entropy coding. Besides, our NeCGS can handle the dynamic scenario well, where new 3D models are constantly added to a compressed set. Extensive experiments and ablation studies demonstrate the significant advantages of our NeCGS over state-of-the-art methods both quantitatively and qualitatively. The source code is publicly available at https://github.com/rsy6318/NeCGS.*

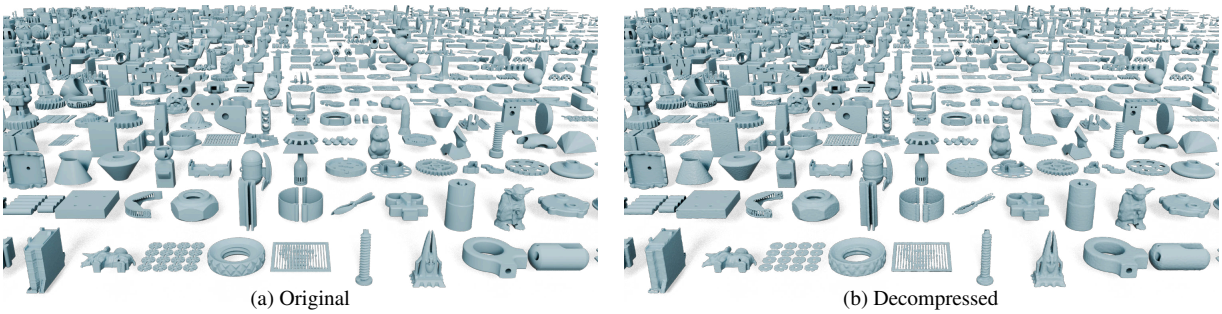
1. Introduction

3D mesh models find widespread applications across computer graphics, virtual reality, robotics, and autonomous driving. As geometric data becomes increasingly complex and voluminous, effective compression techniques have become critical for efficient storage and transmission.

In contrast to 2D images and videos typically structured as *regular* 2D or 3D tensors, 3D mesh models are commonly represented as *irregular* polygons, posing challenges for compression. Thus, a natural idea is to re-structure mesh models and then leverage image or video compression techniques to compress them. A common practice involves converting mesh models into voxelized point clouds, with the ability to reconstruct the original mesh models using surface reconstruction methods [21, 24]. Based on this, MPEG has recently developed two types of 3D point cloud compression (PCC) standards [28, 47]: geometry-based PCC (GPCC) for static models and video-based PCC (VPCC) for sequential models. Additionally, the field has seen the emergence of various learning-based PCC techniques [14, 19, 41, 54, 55] driven by advances in deep learning, improving overall compression efficiency. However, voxelized point clouds demand high resolutions (typically 2^{10} or more) to accurately represent geometric shapes, introducing redundancy and limiting compression efficiency. Moreover, these methods primarily focus on individual 3D models or temporally linked 3D model sequences, and they face challenges when dealing with diverse, unrelated 3D shape datasets.

An alternative regular representation method involves employing implicit fields of 3D mesh models, such as signed distance fields (SDFs) and truncated SDFs (TSDFs). This kind of approach entails computing the implicit field value at each uniformly distributed grid point, thereby forming a regular tensor. Mesh models can be reconstructed from these tensors using techniques like Marching Cubes [32] or its variants [15, 45]. In comparison to point clouds, implicit tensorss offer a way to represent mesh models at a relatively lower resolution. However, to capture intricate and detailed geometric structures, larger SDF/TSDF tensors are often necessary when dealing with more complex 3D models. Consequently, when working with geometry sets that encompass a variety of categories, the need for SDF/TSDF tensors of varying sizes arises, sometimes requiring very large tensors to ensure accuracy. This diversity poses significant challenges during the subsequent compression phase. Recently, neural implicit representation methods, such as DeepSDF [36], utilize multilayer perceptrons (MLPs) to regress the SDFs of any given query points. While this representation achieves high accuracy for single or similar models (e.g., chairs, tables), the limited receptive field of MLPs makes it challenging to represent large numbers of models in different

*Corresponding author. This work was supported in part by NSFC under Grant 62422118 and in part by the Hong Kong Research Grants Council under Grants 11219324 and 11219422.



(a) Original (b) Decompressed

Figure 1. (**a**) A geometry set of 1000 3D shapes with various structures from the Thingi10K dataset [61], and the data size is **335.92** MB. (**b**) The decompressed 3D shapes after being compressed into **2.064** MB by our NeCGS. Zoom in for details.

categories, which is a more common scenario in practice.

We propose NeCGS, a novel framework for compressing large sets of 3D mesh models with diverse categories. Our NeCGS framework consists of two stages: regular geometry representation and compact neural compression. In the first stage, all 3D mesh models are precisely represented into 4D tensors of uniform and compact size through the proposed efficient and effective implicit representation, called TSDF-Def, from which the 3D models can be recovered using the associated deformable marching cubes. In the second stage, we develop a quantization-aware auto-decoder to regress these 4D tensors to explore the local geometric similarity within each shape and across different shapes. The embedded features and decoder parameters of smaller data sizes represent these models, and compressing these components allows us to compress the entire geometry set. We conducted extensive experiments on various datasets, demonstrating that our NeCGS framework achieves higher compression efficiency compared to existing geometry compression methods when handling large numbers of models. Our NeCGS can achieve a compression ratio of nearly 900 on some datasets, compressing hundreds or even thousands of different models into 1$\sim$2 MB while preserving detailed structures. In addition, our NeCGS can adapt to the dynamic scenario well, where new 3D models are constantly added to the compressed set.

In summary, the main contributions of this paper are:

- an efficient, accurate, and compact implicit representation for 3D mesh models; and
- a novel neural compression framework for 3D geometry sets; and
- a state-of-the-art benchmark for 3D geometry set compression.

2. Related Work

2.1. Geometry Representation

Explicit Geometry Representation. Among the explicit representations, voxelization [7] is the most intuitive. In this method, geometry models are represented by regularly distributed grids, effectively converting them into 3D 'images'. While this approach simplifies the processing of geometry models using image processing techniques, it requires a high resolution to accurately represent the models, which demands substantial memory and limits its application. Another widely used geometry representation method is the point cloud, which consists of discrete points sampled from the surfaces of models. This method has become a predominant approach for surface representation [2, 39, 40]. However, the discrete nature of the points imposes constraints on its use in downstream tasks such as rendering and editing. Triangle meshes offer a more precise and efficient geometry representation. By approximating surfaces with numerous triangles, they achieve higher accuracy and efficiency for certain downstream tasks.

Implicit Geometry Representation. Implicit representations use the isosurface of a function or field to represent surfaces. The most widely used implicit representations include Binary Occupancy Field (BOF) [21, 35], Signed Distance Field (SDF) [29, 36], and Truncated Signed Distance Field (TSDF) [11], from which the model's surface can be easily extracted. However, these methods are limited to representing watertight models. The Unsigned Distance Field (UDF) [8], which is the absolute value of the SDF, can represent more general models, not just watertight ones. Despite this advantage, extracting surfaces from UDF is challenging, which limits its application.

Conversion between Geometry Representations. Geometry representations can be converted between explicit and implicit forms. Various methods [6, 21, 22, 24, 29, 35, 45] are available for calculating the implicit field from given models. Conversely, when converting from implicit to explicit forms, Marching Cubes [32] and its derivatives [15, 45, 48, 49] can reconstruct continuous surfaces from various implicit fields.

2.2. 3D Geometry Data Compression

Single 3D Geometric Model Compression. In recent decades, compression techniques for images and videos have rapidly advanced [4, 5, 34, 51, 59]. However, the irregular nature of geometry data makes it more challenging to compress compared to images and video, which are

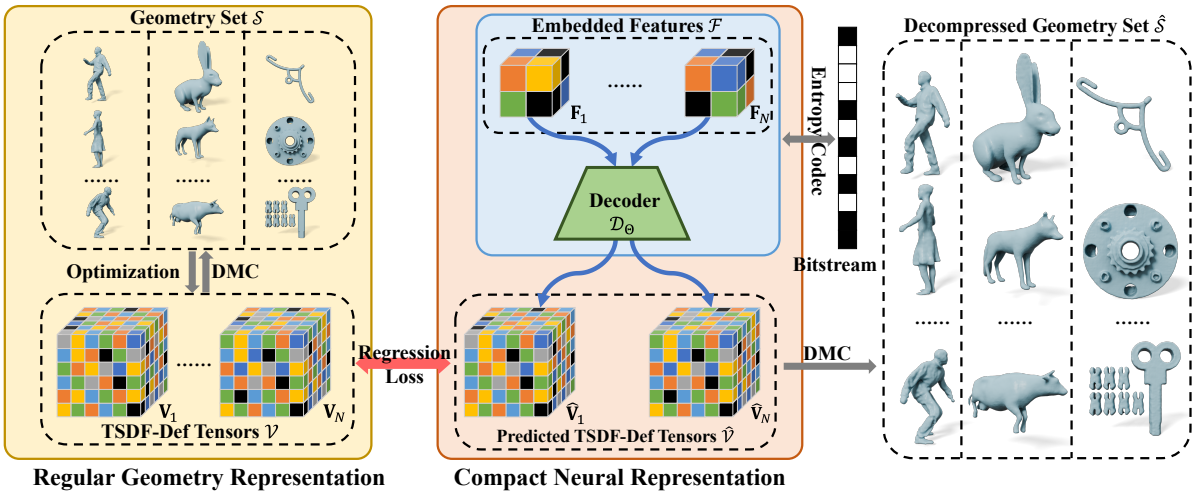


Figure 2. The pipeline of NeCGS for compressing a geometry set. It first represents irregular 3D mesh models into regular 4D tensors of uniform and compact size through the proposed TSDF-Def, a new implicit representation, and an auto-decoder network is utilized to regress these tensors. Then the embedded features and decoder parameters are compressed into bitstreams through entropy coding. When decompressing the models, the decompressed embedded features are fed into the decoder with the decompressed parameters from the bitstreams, reconstructing the TSDF-Def tensors, and the models can be extracted from them.

represented as volumetric data. A natural approach is to convert geometry data into voxelized point clouds, treating them as 3D 'images', and then applying image and video compression techniques to them. Following this intuition, MPEG developed the GPCC standards [13, 28, 46], where triangle meshes or triangle soup approximates the surfaces of 3D models, enabling the compression of models with more complex structures. Subsequently, several improved methods [37, 53, 60, 62] and learning-based methods [3, 9, 10, 18, 42, 43, 54] have been proposed to further enhance compression performance. However, these methods rely on voxelized point clouds to represent geometry models, which is inefficient and memory-intensive, limiting their compression efficiency. In contrast to the previously mentioned methods, Draco [12] uses a kd-tree-based coding method to compress vertices and employs the EdgeBreaker algorithm to encode the topological relationships of the geometry data. Draco utilizes uniform quantization to control the compression ratio, but its performance decreases at higher compression ratios.

Multiple Model Compression. Compared to compressing single 3D geometric models, compressing multiple objects is significantly more challenging. SLRMA [17] addresses this by using a low-rank matrix to approximate vertex matrices, thus compressing sequential models. Mekuria et al. [33] proposed the first codec for compressing sequential point clouds, where each frame is coded using Octree subdivision through an 8-bit occupancy code. Building on this concept, MPEG developed the VPCC standards [13, 28, 46], which utilize 3D-to-2D projection and encode time-varying projected planes, depth maps, and other data using video codecs. Several improved methods [1, 26, 44, 57] have been proposed to enhance the compression of sequential models. Recently, shape priors like SMPL [31] and SMAL [63] have been introduced, allowing the pose and shape of a template frame to be altered using only a few parameters. Pose-driven geometry compression methods [16, 56, 58] leverage this approach to achieve high compression efficiency. However, these methods are limited to sequences of corresponding geometry data and cannot handle sets of unrelated geometry data, which is more common in practice.

3. Proposed Method

Overview. Given a set of N 3D *mesh* models containing diverse categories and structures, denoted as $\mathcal{S} = \{\mathbf{S}_i\}_{i=1}^N$, we aim to compress them into a bitstream while maintaining the quality of the decompressed models as much as possible. To this end, we propose a neural compression paradigm called NeCGS. As shown in Fig. 2, NeCGS consists of two main modules, i.e., Regular Geometry Representation (RGR) and Compact Neural Representation (CNR). Specifically, RGR first represents each *irregular* mesh model within $\mathcal{S}$ into a *regular* 4D tensor, namely TSDF-Def that *mplicitly* describes the geometry structure of the model, via a rendering-based optimization, thus leading to a set of 4D tensors $\mathcal{V} := \{\mathbf{V}_i\}_{i=1}^N$ with $\mathbf{V}_i$ corresponding to $\mathbf{S}_i$. Then CNR further obtains a more compact neural representation of $\mathcal{V}$, where a *quantization-aware* auto-decoder-based network regresses these 4D tensors to explore the local geometric similarity both within each shape and across different shapes for redundancy removal, producing embedded features for them. Finally, the embedded features, along with the network parameters, are encoded into a bitstream through a typical entropy coding method to achieve compression. Note that

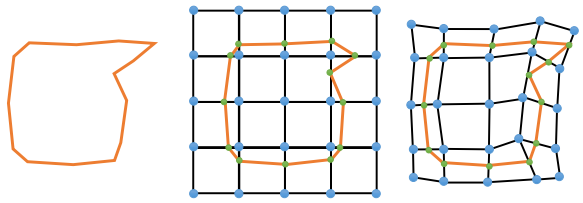


Figure 3. **Left:** The original surface. **Middle:** Surface recovered from traditional TSDF through MC. **Right:** Surface recovered from our TSDF-Def tensor through DMC. The blue points refer to the deformable grid points, the green points refer to the vertices of the extracted surfaces, and the orange lines refer to the faces of the extracted surfaces.

our NeCGS can adapt to the dynamic scenario well, where new 3D models are constantly added to the compressed set.

3.1. Regular Geometry Representation

Unlike 2D images and videos, where pixels are uniformly distributed on 2D *regular* girds, the *irregular* characteristic of 3D mesh models makes it challenging to compress them efficiently and effectively. We propose to convert each 3D mesh model to a 4D regular TSDF-Def tensor, which implicitly represents the geometry structure of the model. Such a regular representation can describe the model precisely, and its regular nature proves beneficial for compression in the subsequent stage.

Definition of TSDF-Def. Inspired by recent shape extracting methods [48, 49], we propose TSDF-Def, which extends the traditional TSDF by introducing an additional deformation for each grid point to adjust the detailed structure during the extraction of models, as shown in Fig. 3. Specifically, each irregular 3D mesh model $\mathbf{S}$ is represented as a regular 4D tensor, denoted as $\mathbf{V} \in \mathbb{R}^{K\times K\times K\times 4}$, where K is the resolution. The value of the grid point located at (u, v, w) is corresponding to $\mathbf{V}(u, v, w) := [\texttt{TSDF}(u, v, w), \Delta u, \Delta v, \Delta w]$, where $(\Delta u, \Delta v, \Delta w)$ are the deformation for the grid point and $1 \leq u, v, w \leq K$. Accordingly, we design differentiable *Deformable Marching Cubes* (DMC), the variant of the Marching Cubes method [32], for surface extraction from a TSDF-Def tensor.

Remark. Traditional SDFs/TSDFs need to increase the tensor size/resolution to capture intricate and detailed geometric structures of 3D models. Consequently, when working with a geometry set encompassing a variety of categories, the need for SDF/TSDF tensors of varying sizes arises, sometimes requiring *very large* tensors to ensure accuracy. This diversity is clearly unfavorable for the subsequent compression process. In contrast, our proposed TSDF-Def tensor introduces deformation for each grid, allowing it to represent detailed structures of models at a much **lower** and **uniform** resolution, which benefits the following compression process. See the experimental comparisons in Sec. 4.3.

Optimization of TSDF-Def Tensors. In the context of representing a 3D mesh model as a 4D tensor through TSDF-Def, the primary goal is to minimize geometry distortion in the reconstructed surface from it. Usually, the majority of an object's regions could be precisely captured using TSDF alone, with only a few areas featuring intricate geometric details that necessitate deformation. Hence, superfluous deformations throughout the optimization process should be reduced. Consequently, we formulate the optimization of the TSDF-Def tensor as

$$\min_{\mathbf{V}} \mathcal{E}_{\text{Rec}}(\texttt{DMC}(\mathbf{V}), \mathbf{S}) + \lambda_{\text{Reg}} \|\mathbf{V}[..., 1:3]\|_1, \quad (1)$$

where $\texttt{DMC}(\cdot)$ refers to the differentiable DMC process for extracting surfaces from TSDF-Def tensors, $\mathcal{E}_{\text{Reg}}(\cdot, \cdot)$ measures the differences between the rendered depth and silhouette images of two mesh models through the differentiable rasterization [25], $\|\cdot\|_1$ is the ℓ_1 norm of a tensor, and $\lambda_{\text{Reg}} > 0$ is a hyper-parameter to balance the two terms.

Algorithm 1 summarizes the whole optimization process. More details can be found in Sec. 1.2 of the *Supplemental Material*.

Algorithm 1: Optimization of TSDF-Def Tensor

Input: 3D mesh model $\mathbf{S}$; the maximum number of iterations `maxIter`.
Output: The optimal TSDF-Def tensor $\mathbf{V} \in \mathbb{R}^{K\times K\times K\times 4}$.

Place uniformly distributed grids in the cube of $\mathbf{S}$, denoted as $\mathbf{G} \in \mathbb{R}^{K\times K\times K\times 3}$;
Initialize $\mathbf{V}[..., 0]$ as the ground truth TSDF of $\mathbf{S}$ at the location of $\mathbf{G}$, the deformation $\mathbf{V}[..., 1:3]$=0, and the current iteration $\texttt{Iter} = 0$;
while $\texttt{Iter} < \texttt{maxIter}$ **do**
  Recover shape from $\mathbf{V}$ according to DMC, $\texttt{DMC}(\mathbf{V})$;
  Calculate the reconstruction error, $\mathcal{E}_{\text{Rec}}(\texttt{DMC}(\mathbf{V}), \mathbf{S})$;
  Optimize $\mathbf{V}$ using ADAM optimizer based on the reconstruction error;
  `Iter`:=`Iter`+1;
end
return $\mathbf{V}$;

3.2. Compact Neural Representation

Observing the similarity of local geometric structures within a typical 3D shape and across different shapes, i.e., redundancy, we further propose a *quantization-aware* neural representation process to summarize the similarity within $\mathcal{V}$, leading to more compact representations with redundancy removed.

Network Architecture. We construct an auto-decoder network architecture to regress these 4D TSDF-Def tensors. Specifically, it comprises a head layer, which increases the channel of its input, and L cascaded upsampling modules, which progressively upscale the feature tensor. We also utilize the PixelShuffle technique [50] between the convolution and activation layers to achieve upscaling. We refer

reviewers to Sec. 2.1 of *Supplemental Material* for more details. For TSDF-Def tensor $\mathbf{V}_i$, the corresponding input to the auto-decoder is the embedded feature, denoted as $\mathbf{F}_i \in \mathbb{R}^{K' \times K' \times K' \times C}$, where K' is the resolution satisfying $K' \ll K$ and C is the number of channels. Moreover, we integrate differentiable quantization to the embedded features and network parameters in the process, which can efficiently reduce the quantization error. In all, the compact neural representation process can be written as

$$\widehat{\mathbf{V}}_i = \mathcal{D}_{\mathcal{Q}(\mathbf{\Theta})}(\mathcal{Q}(\mathbf{F}_i)), \tag{2}$$

where $\mathcal{Q}(\cdot)$ stands for the differentiable quantization operator, and $\widehat{\mathbf{V}}_i$ is the regressed TSDF-Def tensor.

Loss Function. We employ a joint loss function comprising Mean Absolute Error (MAE) and Structural Similarity Index (SSIM) to simultaneously optimize the embedded features $\{\mathbf{F}_i\}$ and the network parameters $\mathbf{\Theta}$. In computing the MAE between the predicted and ground truth TSDF-Def tensors, we concentrate more on the grids near the surface. These surface grids crucially determine the surfaces through their TSDFs and deformations; hence we assign them higher weights during optimization than the grids farther away from the surface. The overall loss function for the i-th model is written as

$$\begin{aligned}\mathcal{L}(\widehat{\mathbf{V}}_i, \mathbf{V}_i) =& \|\widehat{\mathbf{V}}_i - \mathbf{V}_i\|_1 + \lambda_1 \|\mathbf{M}_i \odot (\widehat{\mathbf{V}}_i - \mathbf{V}_i)\|_1 \\ &+ \lambda_2 (1 - \texttt{SSIM}(\widehat{\mathbf{V}}_i, \mathbf{V}_i)),\end{aligned} \tag{3}$$

where $\mathbf{M}_i = \mathbb{1}(|\mathbf{V}_i[..., 0]|) < \tau)$ is the mask, indicating whether a grid is near the surface, i.e., its TSDF is less than the threshold τ, while λ_1 and λ_2 are the weights to balance each term of the loss function.

Entropy Coding. After obtaining the quantized features $\{\widetilde{\mathbf{F}}_i = \mathcal{Q}(\mathbf{F}_i)\}$ and quantized network parameters $\widetilde{\mathbf{\Theta}} = \mathcal{Q}(\mathbf{\Theta})$, we adopt the Huffman Codec [20] to further compress them into a bitstream. More advanced entropy coding methods can be employed to further improve compression performance.

3.3. Decompression

To obtain the 3D mesh models from the bitstream, we first decompress the bitstream to derive the embedded features, $\{\widetilde{\mathbf{F}}_i\}$ and the decoder parameter, $\widetilde{\mathbf{\Theta}}$. Then, for each $\widetilde{\mathbf{F}}_i$, we feed it to the decoder $\mathcal{D}_{\widetilde{\mathbf{\Theta}}}(\cdot)$ to generate its corresponding TSDF-Def tensor, i.e., $\widehat{\mathbf{V}}_i = \mathcal{D}_{\widetilde{\mathbf{\Theta}}}(\widetilde{\mathbf{F}}_i)$. Finally, we utilize DMC to recover each shape from $\widehat{\mathbf{V}}_i$, $\widehat{\mathbf{S}}_i = \texttt{DMC}(\widehat{\mathbf{V}}_i)$, forming the set of decompressed geometry data, $\widehat{\mathcal{S}} = \{\widehat{\mathbf{S}}_i\}_{i=1}^N$.

4. Experiment

4.1. Experimental Setting

Implementation Details. In the process of optimizing TSDF-Def tensors, we employed the ADAM optimizer [23] for 500 iterations per shape, using a learning rate of 0.01. The resolution of TSDF-Def tensors was $K = 128$, and the weight for L1 regularization of deformation was $\lambda_{\text{Reg}} = 10$. The resolution and the number of channels of the embedded features were $K' = 4$ and $C = 16$, respectively. And the decoder is composed of $L = 5$ upsampling modules with an up-scaling factor of 2. During the optimization, we set $\lambda_1 = 5$ and $\lambda_2 = 10$, and the embedded features and decoder parameters were optimized by the ADAM optimizer for 400 epochs, with a learning rate of 1e-3. We achieved different compression efficiencies by adjusting decoder sizes. We conducted all experiments on an NVIDIA RTX 3090 GPU with Intel(R) Xeon(R) CPU.

Datasets. We evaluated our NeCGS on a range of datasets, including models of humans, animals, and CAD designs. For human models, we randomly selected 500 shapes from the AMA dataset [52] (378.41 MB[1]). For animal models, we selected 500 shapes from the DT4D dataset [27] (683.80 MB). For CAD models, we chose 1000 shapes from the Thingi10K dataset [61] (335.92 MB). Additionally, we created a more challenging dataset, labeled as Mixed, by randomly selecting 200 models from each dataset, totaling 496.16 MB. In all experiments, we scaled all models in a cube with a range of $[-1, 1]^3$ to ensure they are in the same scale.

Methods under Comparison. In terms of traditional geometry codecs, we chose the three most impactful geometry coding standards with released codes, G-PCC[2] and V-PCC[3] from MPEG (see more details about them in [13, 28, 46]), and Draco [4] from Google as the baseline methods. Additionally, we compared our approach with state-of-the-art deep learning-based compression methods, specifically PCGCv2 [54]. Furthermore, we adapted DeepSDF [36] with quantization to serve as another baseline method, denoted as QuantDeepSDF. More details can be found in Sec. 3.2 of *Supplemental Material*. It is worth noting that while some of the chosen baseline methods were originally designed for point cloud compression, we utilized voxel sampling and SPSR [21] to convert them between the forms of point cloud and surface.

Evaluation Metrics. Following previous reconstruction methods [35, 38], we utilize Chamfer Distance (CD), Normal Consistency (NC), F-Score with the thresholds of 0.005 and 0.01 (F1-0.005 and F1-0.01) as the evaluation metrics. Furthermore, to comprehensively compare the compression efficiency of different methods, we use Rate-Distortion (RD) curves. These curves illustrate the distortions at various

[1] The original geometry data is kept as triangle meshes, so the storage size is much less than the voxelized point clouds. All data is stored in OBJ format.

[2] https://github.com/MPEGGroup/mpeg-pcc-tmc13

[3] https://github.com/MPEGGroup/mpeg-pcc-tmc2

[4] https://github.com/google/draco

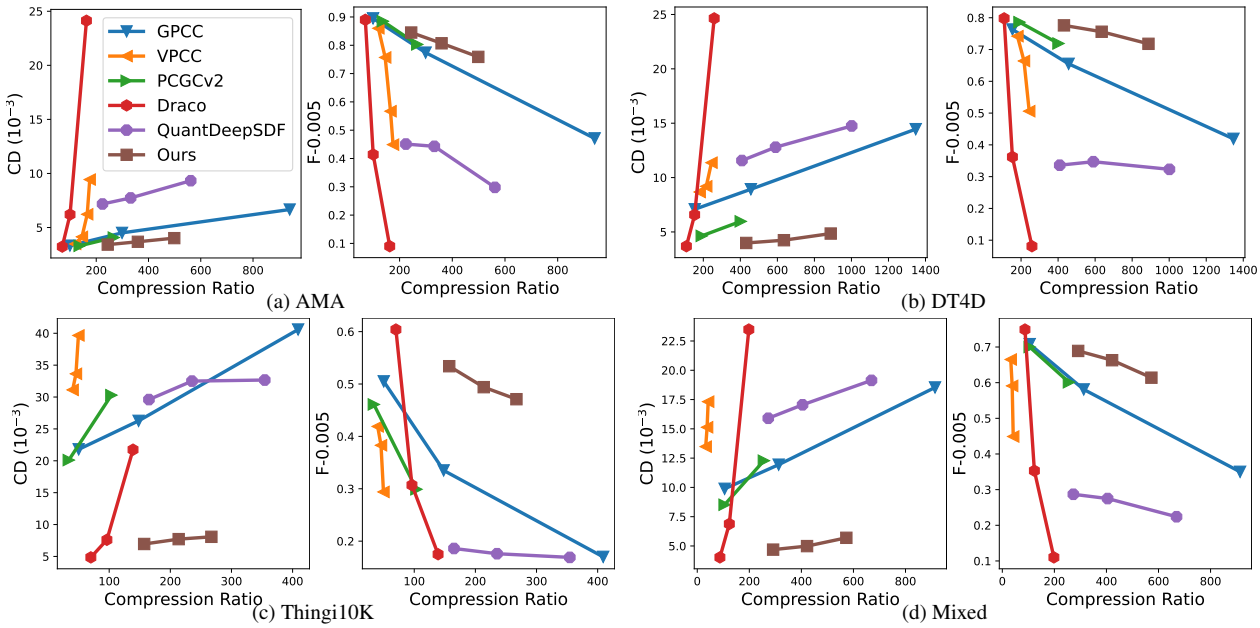


Figure 4. Quantitative comparisons of different methods on four 3D geometry sets. All figures use a common legend.

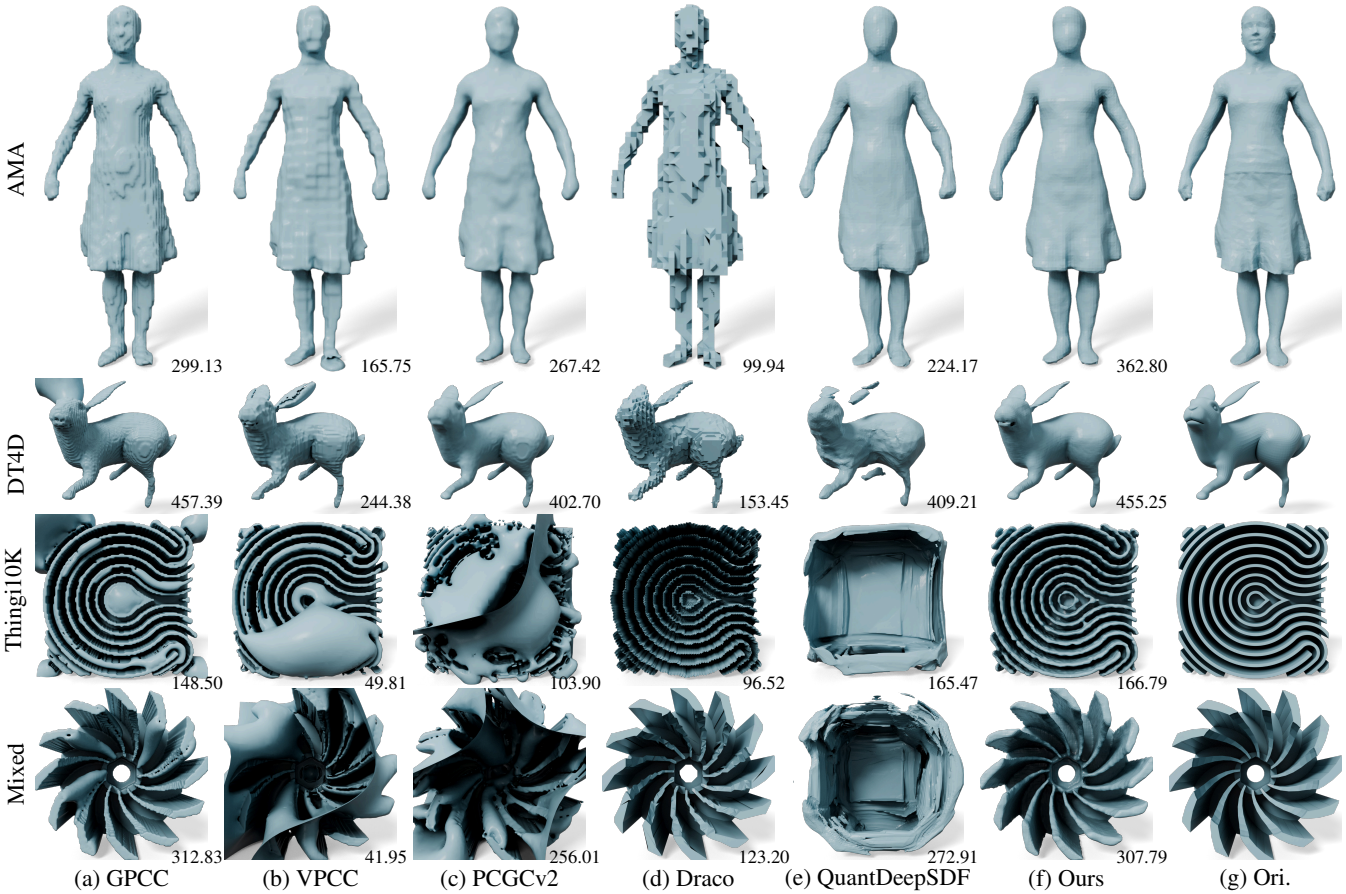


Figure 5. Visual comparisons of different compression methods. All numbers in corners represent the compression ratio. Zoom in for details.

compression ratios, with CD and F1-0.005 specifically describing the distortion of the decompressed models. Our goal is to minimize distortion, indicated by a low CD and a high F1-Score, while maximizing the compression ratio. Therefore, for the RD curve representing CD, optimal compression performance is achieved when the curve is closest to the lower right corner. Similarly, for the RD curve representing the F1-Score, the ideal compression performance is when the curve is nearest to the upper right corner. Their detailed definition can be found in Sec. 3.1 of *Supplemental Material*.

(a) Ori. (b) 455.25 (c) 651.85 (d) 899.73

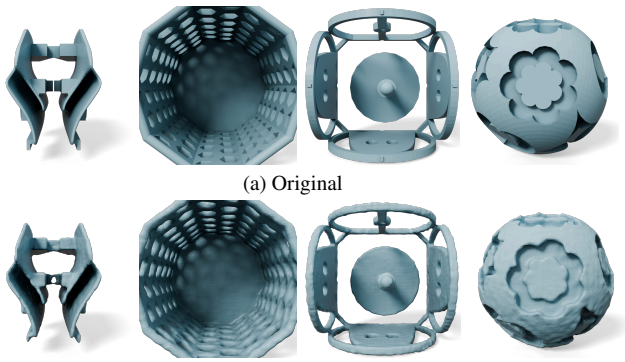
Figure 6. Decompressed models under different compression ratios.

4.2. Results and Comparisons

The RD curves of different compression methods under different datasets are shown in Fig. 4. As the compression ratio increases, the distortion also becomes larger. It is obvious that our NeCGS can achieve **much better** compression performance than the baseline methods when the compression ratio is high, even in the challenging Mixed dataset. In particular, our NeCGS achieves a minimum compression ratio of 300, and on the DT4D dataset, the compression ratio even reaches nearly 900, with minimal distortion. Due to the larger model differences within the Thingi10K and Mixed datasets compared to the other two datasets, the compression performance on these two datasets is inferior.

The visual results of different compression methods are shown in Fig. 5. Compared to other methods, models compressed using our approach occupy a larger compression ratio and retain more details after decompression. Fig. 6 illustrates the decompressed models under different compression ratios. Even when the compression ratio reaches nearly 900, our method can still retain the details of the models. Fig. 7 shows the visual results on complex models.

Efficiency Comparison. Table 1 lists the compression time of various compression methods for a set and the decompression time for a 3D model. Compared to the methods for single shapes, i.e., GPCC, PCGCv2, and Draco, our NeCGS consumes more time during compression. While compared with VPCC, which can compress the entire dataset at once, our NecGS is much faster. As for decompression, our NeCGS takes the *least* time. *It is worth noting* that our NeCGS is designed for the offline compression of geometry sets to save storage space, where decompression time assumes a crucial role as it directly impacts the efficiency of subsequent applications on the set, with compression time playing a secondary role.



(a) Original

(b) Decompressed

Figure 7. Visualization of complex shapes from the Thingi10K dataset. Here the compression ratio is 162.

Table 1. Efficiency analysis of different methods.

Method	GPCC	VPCC	PCGCV2	Draco	Ours
Comp. Time (h)	0.62	39.34	1.76	0.06	10.01
Decomp. Time (ms)	562.56	762.87	100.32	365.18	98.95

4.3. Ablation Study

In order to illustrate the efficiency of each design of our NeCGS, we conducted extensive ablation study about them on the Mixed dataset.

Figure 8. Models recovered from different regular geometry representations under various tensor resolutions. From **Left** to **Right**: Original, TSDF with $K = 64$, TSDF with $K = 128$, TSDF-Def with $K = 64$, and TSDF-Def with $K = 128$.

Comparison between TSDF and TSDF-Def. As shown in Fig. 8, compared with 3D models recovered from TSDF tensors through MC, the models recovered from TSDF-Def tensors through DMC preserve more details of the thin structures, especially when the tensor resolutions are relatively small. Quantitatively, as listed in Table 2, our TSDF-Def improves the representation accuracy obviously over TSDF. In addition, the necessity and advantage of the L1 regularization in Eq. (1) is verified by comparing the results of TSDF-Def w/o L1 Reg. and TSDF-Def in Table 2.

Table 2. Quantitative comparisons between TSDF and TSDF-Def on 100 3D shapes with thin and fine structures from the Mixed dataset.

Representation	CD ($\times 10^{-3}$) ↓	NC ↑	F1-0.005 ↑	F1-0.01 ↑
TSDF	6.630	0.903	0.621	0.883
TSDF-Def w/o L1 Reg.	6.205	0.910	0.649	0.907
TSDF-Def	5.396	0.917	0.677	0.920

Neural Representation Structure. To illustrate the superiority of the auto-decoder framework, we utilized an auto-encoder to regress the TSDF-Def tensor. Technically, we

used a ConvNeXt block [30] as the encoder by replacing 2D convolutions with 3D convolutions. Under the auto-encoder framework, we optimize the parameters of the encoder to change the embedded features. The RD curves about these two structures are shown in Fig. 9a, demonstrating the rationality of our decoder structure.

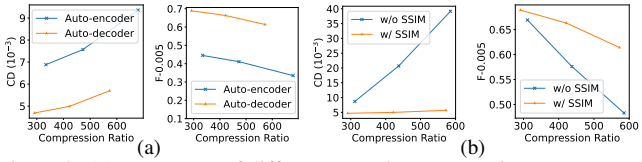
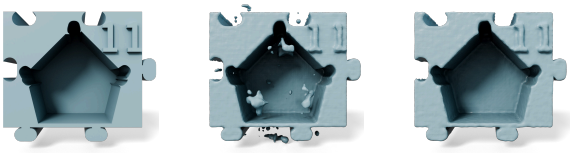


Figure 9. (a) RD curves of different neural representation structures. (b) RD curves of different regression losses.

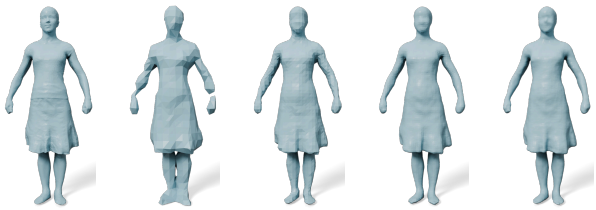


(a) Original (b) w/o SSIM (c) w/ SSIM

Figure 10. Visual comparison of regression loss w/ and w/o SSIM item.

SSIM Loss. Compared to MAE, which focuses on one-to-one errors between predicted and ground truth TSDF-Def tensors, the SSIM item in Eq. 3 emphasizes more on the local similarity between tensors, increasing the regression accuracy. To verify this, we removed the SSIM item and kept the others unchanged. Their RD curves are shown in Fig. 9b, and it is obvious that the SSIM item in the regression loss increases the compression performance. The visual comparison is shown in Fig. 10, and without SSIM, there are floating parts around the decompressed models.

Resolution of TSDF-Def Tensors. We tested the compression performance at different resolutions of TSDF-Def tensors on the AMA dataset, where the decoder layers are adjusted accordingly. Specifically, we removed the last two layers for a resolution of 32, removed the last layer for a resolution of 64 and added an extra layer for a resolution of 256. The quantitative and numerical comparisons are shown in Table 3 and Fig. 11, respectively. Obviously, increasing the resolution can enhance the compression effectiveness,



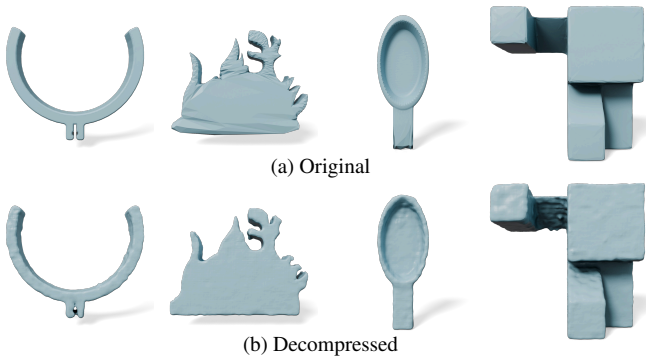
(a) Ori. (b) 32 (c) 64 (d) 128 (e) 256

Figure 11. Visual comparison under different resolutions of TSDF-Def tensors.

resulting in more detailed structures preserved after decompression. However, the inference time also increases accordingly due to more layers involved, which directly affects the decompression efficiency.

Table 3. Quantitative comparisons of different resolutions of TSDF-Def tensors.

Res.	Size (MB)	Com. Ratio	CD ($\times 10^{-3}$) ↓	NC ↑	F1-0.005 ↑	F1-0.01 ↑	Infer. Time (ms)
32	1.037	364.89	16.982	0.872	0.215	0.542	15.30
64	1.408	268.75	4.271	0.927	0.721	0.966	38.97
128	1.493	253.45	3.436	0.952	0.842	0.991	98.95
256	1.627	232.58	3.234	0.962	0.870	0.995	421.94



(a) Original

(b) Decompressed

Figure 12. Visual demonstration of the effectiveness of our NeCGS under the dynamic scenario, where new models were added to the Thingi10K dataset.

Effectiveness under Dynamic Scenarios. Our NeCGS can effectively handle a dynamic scenario, where new 3D models are added to the compressed set, i.e., the trained auto-decoder with existing 3D models contained in the set can be generalized to new 3D models. Specifically, given a new 3D model, we first represent it as a TSDF-Def tensor using Algorithm 1. Then, following the optimization process in Sec. 3.2, we optimize solely the corresponding embedded feature while keeping the network parameters of the trained auto-decoder unchanged. The resulting features of the newly added 3D models are entropy-coded into the bitstream. As illustrated in Fig. 12, obviously, the newly-added 3D models can be well processed, demonstrating our assertion

5. Conclusion

We have presented NeCGS, a highly effective neural compression scheme for 3D geometry sets. NeCGS has achieved remarkable compression performance on various datasets with diverse and detailed shapes, outperforming state-of-the-art compression methods to a large extent. These advantages are attributed to our regular geometry representation and the compression accomplished by a convolution-based auto-decoder. We believe our NeCGS framework will inspire further advancements in the field of geometry compression.

References

[1] Ashek Ahmmed, Manoranjan Paul, Manzur Murshed, and David Taubman. Dynamic point cloud geometry compression using cuboid based commonality modeling framework. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2159–2163. IEEE, 2021. 3

[2] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, pages 586–606. Spie, 1992. 2

[3] Sourav Biswas, Jerry Liu, Kelvin Wong, Shenlong Wang, and Raquel Urtasun. Muscle: Multi sweep compression of lidar using deep entropy models. *Advances in Neural Information Processing Systems*, 33:22170–22181, 2020. 3

[4] Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser Nam Lim, and Abhinav Shrivastava. Nerv: Neural representations for videos. *Advances in Neural Information Processing Systems*, 34:21557–21568, 2021. 2

[5] Hao Chen, Matthew Gwilliam, Ser-Nam Lim, and Abhinav Shrivastava. Hnerv: A hybrid neural representation for videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10270–10279, 2023. 2

[6] Z-Q Cheng, Y-Z Wang, B Li, K Xu, G Dang, and S-Y Jin. A survey of methods for moving least squares surfaces. In *Proceedings of the Fifth Eurographics/IEEE VGTC conference on Point-Based Graphics*, pages 9–23, 2008. 2

[7] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6970–6981, 2020. 2

[8] Julian Chibane, Gerard Pons-Moll, et al. Neural unsigned distance fields for implicit function learning. *Advances in Neural Information Processing Systems*, 33:21638–21652, 2020. 2

[9] Tingyu Fan, Linyao Gao, Yiling Xu, Dong Wang, and Zhu Li. Multiscale latent-guided entropy model for lidar point cloud compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 33(12):7857–7869, 2023. 3

[10] Chunyang Fu, Ge Li, Rui Song, Wei Gao, and Shan Liu. Octattention: Octree-based large-scale contexts model for point cloud compression. In *Proceedings of the AAAI conference on artificial intelligence*, pages 625–633, 2022. 3

[11] Peng Gao, Zhengkai Jiang, Haoxuan You, Pan Lu, Steven CH Hoi, Xiaogang Wang, and Hongsheng Li. Dynamic fusion with intra-and inter-modality attention flow for visual question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6639–6648, 2019. 2

[12] Google. Point cloud compression reference software. Website. https://github. com/google/draco. 3

[13] Danillo Graziosi, Ohji Nakagami, Satoru Kuma, Alexandre Zaghetto, Teruhiko Suzuki, and Ali Tabatabai. An overview of ongoing point cloud compression standardization activities: Video-based (v-pcc) and geometry-based (g-pcc). *APSIPA Transactions on Signal and Information Processing*, 9:e13, 2020. 3, 5

[14] André FR Guarda, Nuno MM Rodrigues, and Fernando Pereira. Point cloud coding: Adopting a deep learning-based approach. In *2019 Picture Coding Symposium (PCS)*, pages 1–5. IEEE, 2019. 1

[15] Benoit Guillard, Federico Stella, and Pascal Fua. Meshudf: Fast and differentiable meshing of unsigned distance field networks. In *European Conference on Computer Vision*, pages 576–592, 2022. 1, 2

[16] Junhui Hou, Lap-Pui Chau, Nadia Magnenat-Thalmann, and Ying He. Compressing 3-d human motions via keyframe-based geometry videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(1):51–62, 2014. 3

[17] Junhui Hou, Lap-Pui Chau, Nadia Magnenat-Thalmann, and Ying He. Sparse low-rank matrix approximation for data compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(5):1043–1054, 2015. 3

[18] Lila Huang, Shenlong Wang, Kelvin Wong, Jerry Liu, and Raquel Urtasun. Octsqueeze: Octree-structured entropy model for lidar compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1313–1323, 2020. 3

[19] Tianxin Huang and Yong Liu. 3d point cloud geometry compression on deep learning. In *Proceedings of the 27th ACM international conference on multimedia*, pages 890–898, 2019. 1

[20] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40 (9):1098–1101, 1952. 5

[21] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013. 1, 2, 5

[22] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 61–70, 2006. 2

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[24] Ravikrishna Kolluri. Provably good moving least squares. *ACM Transactions on Algorithms*, 4(2):1–25, 2008. 1, 2

[25] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics (ToG)*, 39(6):1–14, 2020. 4

[26] Li Li, Zhu Li, Vladyslav Zakharchenko, Jianle Chen, and Houqiang Li. Advanced 3d motion prediction for video-based dynamic point cloud compression. *IEEE Transactions on Image Processing*, 29:289–302, 2019. 3

[27] Yang Li, Hikari Takehara, Takafumi Taketomi, Bo Zheng, and Matthias Nießner. 4dcomplete: Non-rigid motion estimation beyond the observable surface. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12706–12716, 2021. 5

[28] Hao Liu, Hui Yuan, Qi Liu, Junhui Hou, and Ju Liu. A comprehensive study and comparison of core technologies for mpeg 3-d point cloud compression. *IEEE Transactions on Broadcasting*, 66(3):701–717, 2019. 1, 3, 5

[29] Shi-Lin Liu, Hao-Xiang Guo, Hao Pan, Peng-Shuai Wang, Xin Tong, and Yang Liu. Deep implicit moving least-squares functions for 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1788–1797, 2021. 2

[30] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022. 8

[31] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Smpl: A skinned multi-person linear model. *ACM Trans. Graph.*, 34(6), 2015. 3

[32] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. 1, 2, 4

[33] Rufael Mekuria, Kees Blom, and Pablo Cesar. Design, implementation, and evaluation of a point cloud codec for tele-immersive video. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(4):828–842, 2016. 3

[34] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Practical full resolution learned lossless image compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10629–10638, 2019. 2

[35] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 2, 5

[36] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 1, 2, 5

[37] Eduardo Peixoto. Intra-frame compression of point cloud geometry using dyadic decomposition. *IEEE Signal Processing Letters*, 27:246–250, 2020. 3

[38] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision*, pages 523–540. Springer, 2020. 5

[39] 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*, pages 652–660, 2017. 2

[40] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30:1–xxx, 2017. 2

[41] Maurice Quach, Giuseppe Valenzise, and Frederic Dufaux. Learning convolutional transforms for lossy point cloud geometry compression. In *2019 IEEE international conference on image processing (ICIP)*, pages 4320–4324. IEEE, 2019. 1

[42] Maurice Quach, Giuseppe Valenzise, and Frederic Dufaux. Learning convolutional transforms for lossy point cloud geometry compression. In *2019 IEEE international conference on image processing (ICIP)*, pages 4320–4324. IEEE, 2019. 3

[43] Zizheng Que, Guo Lu, and Dong Xu. Voxelcontext-net: An octree based framework for point cloud compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6042–6051, 2021. 3

[44] Evaristo Ramalho, Eduardo Peixoto, and Edil Medeiros. Silhouette 4d with context selection: Lossless geometry compression of dynamic point clouds. *IEEE Signal Processing Letters*, 28:1660–1664, 2021. 3

[45] Siyu Ren, Junhui Hou, Xiaodong Chen, Ying He, and Wenping Wang. Geoudf: Surface reconstruction from 3d point clouds via geometry-guided distance representation. In *Proceedings of the IEEE/CVF Internation Conference on Computer Vision*, pages 14214–14224, 2023. 1, 2

[46] Sebastian Schwarz, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, Pablo Cesar, Philip A Chou, Robert A Cohen, Maja Krivokuća, Sébastien Lasserre, Zhu Li, et al. Emerging mpeg standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9 (1):133–148, 2018. 3, 5

[47] Sebastian Schwarz, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, Pablo Cesar, Philip A Chou, Robert A Cohen, Maja Krivokuća, Sébastien Lasserre, Zhu Li, et al. Emerging mpeg standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9 (1):133–148, 2018. 1

[48] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems*, 34:6087–6101, 2021. 2, 4

[49] Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. Flexible isosurface extraction for gradient-based mesh optimization. *ACM Transactions on Graphics (TOG)*, 42(4):1–16, 2023. 2, 4

[50] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016. 4

[51] Yannick Strümpler, Janis Postels, Ren Yang, Luc Van Gool, and Federico Tombari. Implicit neural representations for image compression. In *European Conference on Computer Vision*, pages 74–91. Springer, 2022. 2

[52] Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics*, 27(3):1–9, 2008. 5

[53] Chaofei Wang, Wenjie Zhu, Yingzhan Xu, Yiling Xu, and Le Yang. Point-voting based point cloud geometry compression. In *2021 IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–5. IEEE, 2021. 3

[54] Jianqiang Wang, Dandan Ding, Zhu Li, and Zhan Ma. Multiscale point cloud geometry compression. In *2021 Data Compression Conference (DCC)*, pages 73–82. IEEE, 2021. 1, 3, 5

[55] Jianqiang Wang, Hao Zhu, Haojie Liu, and Zhan Ma. Lossy point cloud geometry compression via end-to-end learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(12):4909–4923, 2021. 1

[56] Xinju Wu, Pingping Zhang, Meng Wang, Peilin Chen, Shiqi Wang, and Sam Kwong. Geometric prior based deep human point cloud geometry compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 2024. 3

[57] Jian Xiong, Hao Gao, Miaohui Wang, Hongliang Li, King Ngi Ngan, and Weisi Lin. Efficient geometry surface coding in v-pcc. *IEEE Transactions on Multimedia*, 25:3329–3342, 2022. 3

[58] Ruoke Yan, Qian Yin, Xinfeng Zhang, Qi Zhang, Gai Zhang, and Siwei Ma. Pose-driven compression for dynamic 3d human via human prior models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 3

[59] Yibo Yang, Robert Bamler, and Stephan Mandt. Improving inference for neural image compression. *Advances in Neural Information Processing Systems*, 33:573–584, 2020. 2

[60] Xiang Zhang, Wen Gao, and Shan Liu. Implicit geometry partition for point cloud compression. In *2020 Data Compression Conference (DCC)*, pages 73–82. IEEE, 2020. 3

[61] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016. 2, 5

[62] Wenjie Zhu, Yiling Xu, Dandan Ding, Zhan Ma, and Mike Nilsson. Lossy point cloud geometry compression via region-wise processing. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(12):4575–4589, 2021. 3

[63] Silvia Zuffi, Angjoo Kanazawa, David Jacobs, and Michael J. Black. 3D menagerie: Modeling the 3D shape and pose of animals. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3