NeuralSlice: Learning to Reconstruct 4D Objects

Chenbo Jiang
Nanjing University of Science and Technology ZiJin College
cbjiang@foxmail.com

Abstract

We propose a novel method NeuralSlice for reconstructing 4D objects from 3D point clouds, which has different shapes and topologies from different slices. Different from the previous Spatio-temporal 4D reconstruction, our method can not only represent the change of shape but also represent the topological change to some certain extent. Our approach can create complex geometries for some 4D games. Our template-based representation can automatically adapt to the topology for 3D reconstruction tasks without subsequent modification. NeuralSlice outperformed many template-based 3D reconstruction methods and it has great potential for 4D reconstruction tasks from a 3D point cloud. We demonstrated our 4D reconstruction results, with changes in shape and topology from different slices and NeuralSlice outperformed baselines.

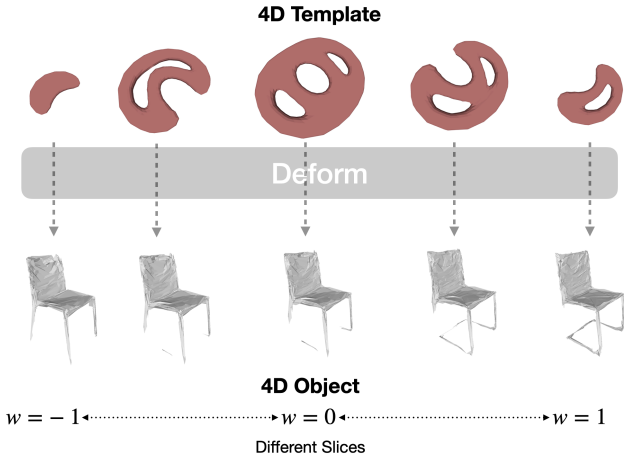

Figure 1. Our method deforms a 4D template into a 4D object. When the fourth dimension w changes, different slices of our 4D object can represent different 3D meshes. Among these 3D meshes, they have different shapes and different topologies.

1. Introduction

Recently, 4-Dimensional games like 4D toys [29] and miegakure [30] already have some dynamics [31]. But for this kind of game, it is very difficult to model 4-Dimensional shapes, so the models in these games are very simple now, and our method can naturally reconstruct the four-dimensional shapes. One of the slices is the result of 3D reconstruction. Prior works like [21] only construct 4D shapes from existing meshes movements. Our method can greatly increase the complexity of models in this type of game. However, other 3D reconstruction methods cannot do this.

Many 3D reconstruction methods need to deform a template [10, 11, 32]. If no subsequent topological modification is performed, the final generated mesh is always homomorphism to the template, which causes very large geometric limitations. Although some methods can modify the topology [22], but there are no direct methods to adapt topology automatically. Our method can automatically adapt to the topology while deforming the template, without subsequent modification.

In recent years, there has been an increasing interest in implicit surface representation for 3D Deep Learning [4, 16, 19, 23, 28]. For this kind of method, it can represents smooth surfaces, but it needs marching cubes [17] algorithm, which consumes a lot of time. At the same time, this type of method cannot reconstruct 4D objects.

We propose a novel method to reconstruct 3D objects and 4D objects at the same time. Specifically, our method deforms a 4D object and takes one of the slices as the result of 3D reconstruction, so that our method can generate 3D reconstruction and 4D reconstruction results at the same time.

In this paper, when we talk about 4D shapes, we are not only talking about space-time models we are talking about a 3-manifold $S \in \mathbb{R}^4$. If our models have time variables, it is a 5-dimensional space-time model.

Our method has three core blocks: slice block, deform block, and locate block. As illustrated in Figure 4. For slice block, it slices a 4D object and generates a 3D mesh, similar to [6, 31]. It is difficult for humans to imagine this process, but computers can easily compute it. The 3D object uses triangles to construct the basic unit. Similarly, a 4D object

uses a tetrahedron to construct the basic unit. In this case, calculating the 3D slice of the 4D object is to calculate the cross-section of a bunch of tetrahedrons and the hyperplane. Details in Section 3.2.

For deform block and locate block, unlike [10, 11, 32] extracting global feature, our method extracts local feature [16, 28] from a point cloud, similar to ConvOnet [28]. The difference is that ConvOnet is the implicit method, the local feature naturally represents the information of each location, and our method is explicit, the input is Template, and the point on the Template cannot directly obtain the local feature. To solve this problem, we built the located block to extract the local feature of each point on the template. Details in Section 3.3.

In the experiment, we propose reasonable evaluation criteria for evaluating 4D objects. This evaluation criterion describes the differences of a 4D shape on different slices. If a 4D object is no changes in different slices, then the fourth dimension is meaningless. A "good" 4D object we want to reconstruct is that the Chamfer Distance between a certain slice to a 3D object is very low, and it varies greatly in different slices. For this reason, we constructed this criterion. Our method outperformed baselines in evaluation. We have also shown that for a 4D object, when w changes, different slices of our 4D object can represent different 3D objects.

Contributions

- We propose a novel representation for 4D object reconstruction with deep learning. This representation improves the geometric complexity of many 4D Games.
- We demonstrated that our 4D objects can represent different 3D meshes with different shapes and topologies when the fourth dimension changes.
- Our representation can automatically adapt to the topology without subsequent modifications.
- We propose reasonable evaluation criteria for 4D objects. Our representation outperformed baselines.

2. Related Works

2.1. 4D Objects

Visualizing a 4D object is a very interesting and challenging problem with a long history [1,6,8,12]. Many methods have been proposed to manipulate 4D shapes [13, 36]. Recently, rigid body dynamics of 4D objects have been proposed [31]. There are also some 4D modeling methods [2, 21]. Some previous Spatio-temporal 4D reconstruction methods [14, 18, 20] only reconstruct from movements but are unable to adapt topology.

However, these methods are directly to reconstruct 4D objects, and because people's understanding of 4-Dimensional space is not in place, these methods are difficult to reconstruct complex geometry from 3D point clouds. And our learning-based method can naturally reconstruct 4D objects from 3D point clouds.

2.2. Learning-based 3D Representations

Learning-based 3D representations can generally be classified as explicit or implicit.

Voxels representation is one of the earliest learning-based representations of 3D reconstruction [5, 34, 35]. But voxel-based techniques are limited in terms of memory and computation. Point cloud [9, 25, 26] is a very efficient representation for 3D Deep Learning, furthermore, they cannot represent topological relations. Mesh is a very popular representation for 3D Deep Learning [10, 11, 32, 33], but many methods require to deform of the template mesh of fixed topology. Our method has the ability to generate arbitrary topology.

Recently, there has been increasing research popularity in implicit surface representation for 3D Deep Learning [4, 16, 19, 23, 28]. Implicit SDF is a good 3D representation for Deep Learning. This method can generate smooth, continuous, and arbitrary topology models. But it also has some problems, it cannot directly predict the model, it needs marching cubes [17] algorithm for iso-surface extraction. In addition, this method is unable to reconstruct 4D objects.

In contrast to all the above methods, we develop a method that can reconstruct 4D objects which can represents different topological structures, and the computational efficiency is higher than implicit fields.

3. Method

3.1. Overview

In this paper, we introduced two tasks that use our representation, 3D reconstruction, and 4D reconstruction. These two tasks are very similar, they both need to use slice block, so we will introduce slice block first. In these two tasks, a 4D object is generated. In the 3d reconstruction task, the 4D object is only meaningful for a specific slice, while in the 4D reconstruction, different slices represent different 3D meshes, and they have different shapes and topologies.

3.2. Slice a 4 Dimensional Shape

Previous works like Atlas-O [10], Pix2mesh [32], NFM [11], are all deforming a sphere $S^2 = \{p = (x, y, z) \mid x^2+y^2+z^2 = 1\}$ and the Neural Nets can be written: $F_{\Theta|z} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$. While Θ represent the network parameters and z represent a latent code of a shape. But it results in the homeomorphism with the template and the final shape.

In this paper, we deform a 3-manifold $\mathcal{S}^3 \in \mathbb{R}^4$ and then slice the 3-manifold by a hyperplane $w = \alpha$. Triangle is simplex of 2-manifold, we use triangles to construct meshes. A tetrahedron is a simplex of a 3-manifold, so a tetrahedron is a unit for us to construct 4D shapes. As illustrated in Figure 2.

A 4D shape is set of tetrahedrons including points $\mathcal{P}$ and tetrahedrons $\mathcal{T}$. A 4D shape is $\mathcal{S}^3 = \{\mathcal{P}, \mathcal{T}\}$. Every point in points set $p_i \in \mathcal{P}$ has 4 dimensions $p_i = (x_i, y_i, z_i, w_i)$, and Every tetrahedron in tetrahedrons set $t_i \in \mathcal{T}$ has 4 points $t_i = (p_i^1, p_i^2, p_i^3, p_i^3)$.

$$\mathcal{P} = \{p \mid p_i = (x_i, y_i, z_i, w_i)\}$$

$$\mathcal{T} = \{t \mid t_i = (p_i^1, p_i^2, p_i^3, p_i^3)\}$$

Our goal is to generate a 3D mesh $\mathcal{M} = \{\mathcal{E}, \mathcal{V}\}$ with given 4D object $\mathcal{S} = \{\mathcal{P}, \mathcal{T}\}$ and the fourth dimension w. $\mathcal{E}$ means edges of a 3D mesh, $\mathcal{V}$ means vertices of a 3D mesh.

$$\mathcal{M} = Slice(\mathcal{S}, w) \tag{1}$$

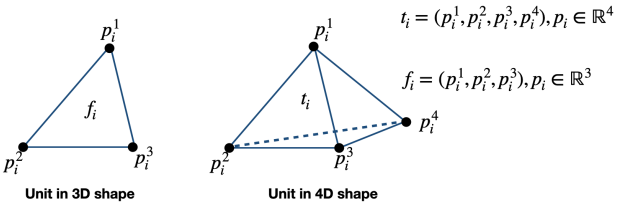

Figure 2. Unit in 3D shape and 4D shape.

A tetrahedron in a 4D shape is a subset of a hyperplane $P_0 : a_0x + b_0y + c_0z + d_0w + e_0 = 0$. This hyperplane may cross-section with another hyperplane $P_1 : a_1x + b_1y + c_1z + d_1w + e_1 = 0$. There are only two possible situations where we need to compute points in P_1. Two points of the tetrahedron are behind the hyperplane and two points are front. Another situation is one point of the tetrahedron is behind and three points are front. As illustrated in Figure 3.

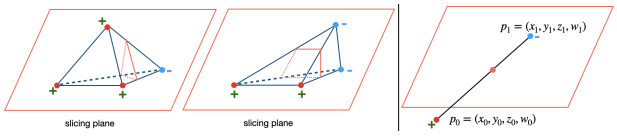

Figure 3. Left: Two possible cross-sections when slicing a tetrahedron with a plane: a triangle or a quadrilateral; note that vertices behind the slicing planes are colored in blue. Right: Compute point. The positive part is above the plane, the negative part is below the plane.

All points we need to compute are in a line that depends on two points in a tetrahedron and when we plug points into the hyperplane, we got a positive number and a negative number. As illustrated in Figure 3.

Now we have two points $p_0 = (x_0, y_0, z_0, w_0)$, $p_1 = (x_1, y_1, z_1, w_1)$ and a hyperplane $ax+by+cz+dw+e = 0$. $ax_0 + by_0 + cz_0 + dw_0 + e > 0$ and $ax_1 + by_1 + cz_1 + dw_1 + e < 0$.

The line located in p_0 and p_1 is:

$$\begin{cases} x = x_0 + mt, \\ y = y_0 + nt, \\ z = z_0 + pt, \\ w = w_0 + qt, \end{cases} \quad \begin{cases} m = x_0 - x_1, \\ n = y_0 - y_1, \\ p = z_0 - z_1, \\ q = w_0 - w_1, \end{cases} \tag{2}$$

Simultaneous equations:

$$\begin{cases} \dfrac{x - x_0}{m} = \dfrac{y - y_0}{n} = \dfrac{z - z_0}{p} = \dfrac{w - w_0}{q} \\ \\ ax + by + cz + dw + e = 0, \end{cases} \tag{3}$$

Let $w = \alpha$, the solution is:

$$\begin{cases} x = \dfrac{(\alpha - w_0)m}{q} + x_0, \\ y = \dfrac{(\alpha - w_0)n}{q} + y_0, \\ z = \dfrac{(\alpha - w_0)p}{q} + z_0, \\ w = \alpha, \end{cases} \tag{4}$$

Then we have computed points, and the face is easy to establish.

3.3. 3D Reconstruction Networks

3.3.1 Architecture

In Atlas-O [10] and NFM [11]. They all use an encoder to extract the global feature of the point cloud, which leads to a problem: the deform network cannot work for the local feature, which limits the expressive ability of the deform network. These networks can all be described as:

$$p_O^i = F_{\Theta|z}\left(p_I^i, z\right) \tag{5}$$

In the equation, p_I^i represents points in the inputs template. p_O^i represents outputs points of every point in the inputs template after deform. $F_{\Theta|z}$ means a deform network with parameters Θ and latent code z from encoder networks with inputs of point clouds or images.

For an input latent code, all points on the template need to use this latent code to deform the template, which is very unreasonable. Inspired by [28], we proposed a method of using a local feature deform template. Specifically:

$$p_O^i = F_{\Theta|z}\left(p_I^i, \Psi(p_I^i, z)\right) \tag{6}$$

Among them, Ψ represents, for a volume feature z, what is the local feature corresponding to point p_I^i on the input

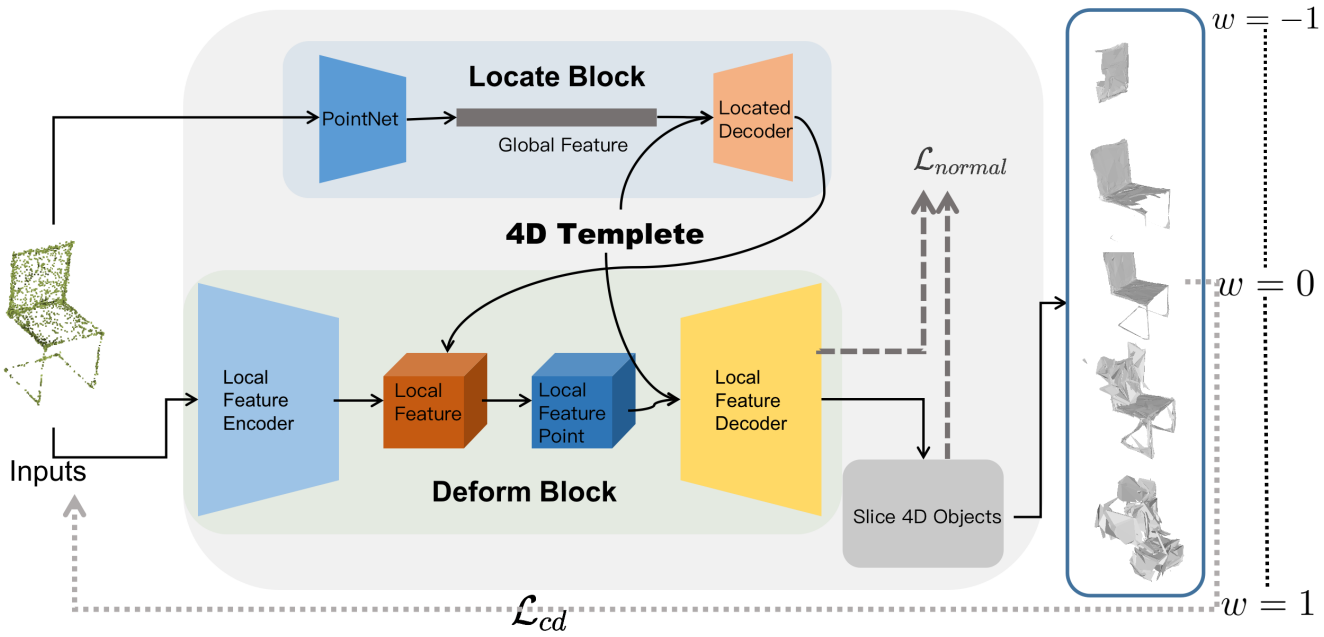

Figure 4. The pipeline of 3D reconstruction. In the 3D reconstruction task, we have three modules, deform block, locate block, and slice block. First, we use the encoders of the deform block and the locate block to extract features, and then the decoder of the locate block outputs location information of the template. We use this location to extract the information of each point on the template, and then put this feature into the decoder of the deform block to deform the template. Finally, we use the slice block to obtain the 3D mesh from the 4D object and calculate the loss.

template. This is what ConvOnet [28] does. ConvOnet is the implicit method. Its input point is a 3D grid, which naturally carries location information. But in our input template, the location information cannot be well included in the input. To solve this problem, we use another network to roughly locate the position in the volume feature corresponding to the point on the template and then use this coordinate to extract the local feature. In equation 7, ψ is the locate block in Figure 4, and $F_{\Theta|z}$ is the deform block in that Figure.

$$p_O^i = F_{\Theta|z}\left(p_I^i, \Psi(\psi(p_I^i, z_g), z)\right) \tag{7}$$

Specifically, the encoder of locate block is a PointNet [26] with latent code 256, the decoder of a located block is an MLP with inputs latent code z_g and template $p_I^i \in \mathbb{R}^4$, the output is $loc \in \mathbb{R}^3$ to locate local feature.

$$loc = \psi(p_I^i, z_g) \tag{8}$$

In deform block, our encoder is similar to ConvOnet, with point clouds inputs and outputs a volume feature. Then we input this volume into a U-Net [7] to enhance representation ability. After that use the loc from locate block to obtain local feature. The function Ψ is a trilinear interpolation to the feature volume z with local position loc.

3.3.2 Loss

In this network, the loss function has two parts, $\mathcal{L}_{cd}$ and $\mathcal{L}_{normal}$. $\mathcal{L}_{cd}$ is used to fit the input point cloud, and $\mathcal{L}_{normal}$ is used to improve the quality of the output mesh and tetrahedron.

Shape loss is the main loss in our loss functions. It defines the gap between two point clouds, and the small $\mathcal{L}_{cd}$ represents two similar point clouds. Among them, S_I represents the input point cloud, S_O represents the output point cloud.

$$\mathcal{L}_{cd} = \sum_{x \in S_I} \min_{y \in S_O} \|x - y\|_2^2 + \sum_{y \in S_O} \min_{x \in S_I} \|x - y\|_2^2 \tag{9}$$

$\mathcal{L}_{normal}$ includes two parts, $\mathcal{L}_{normal3}$ represents mesh normalization and $\mathcal{L}_{normal4}$ represents tetrahedron normalization. In $\mathcal{L}_{normal3}$, $\mathcal{E}_3$ represents the predicted edge length of meshes, ϵ_3 represents the target edge length of meshes. In $\mathcal{L}_{normal4}$, $\mathcal{E}_4$ represents the predicted edge length of tetrahedrons, ϵ_4 represents the target edge length of tetrahedrons.

$$\mathcal{L}_{normal3} = \|\mathcal{E}_3 - \epsilon_3\|_2^2 \tag{10}$$

$$\mathcal{L}_{normal4} = \|\mathcal{E}_4 - \epsilon_4\|_2^2 \tag{11}$$

The overall loss function can be formulated as the weighted sum of these functions.

$$\mathcal{L} = \lambda_1 \mathcal{L}_{cd} + \lambda_2 \mathcal{L}_{normal3} + \lambda_3 \mathcal{L}_{normal4} \quad (12)$$

3.3.3 Training Details

We implement all models in PyTorch [24] and use the Adam [15] optimizer with a learning rate of 10^{-3}. We use 32 batch size and learning rate decay when epoch equals 20, 50, 100, 125, 145, 150, 170, 200, 250, 290. We train this network for 300 epochs, with a RTX 3090 GPU for about 2-3 days. In loss function $\lambda_1 = 1, \lambda_2 = \lambda_3 = 0.005$.

3.4. 4D Reconstruction

In the 3D reconstruction task, our method can generate a 4D object, one of which is the 3D mesh we want to slice by $w = \alpha$. In this method, other slices of the 4D object are useless. The 4D object we reconstructed in Section 3.3 caused a certain degree of waste in the fourth dimension. Therefore, we want to achieve different 3D meshes at different slices. These 3D meshes have different shapes and different topological structures.

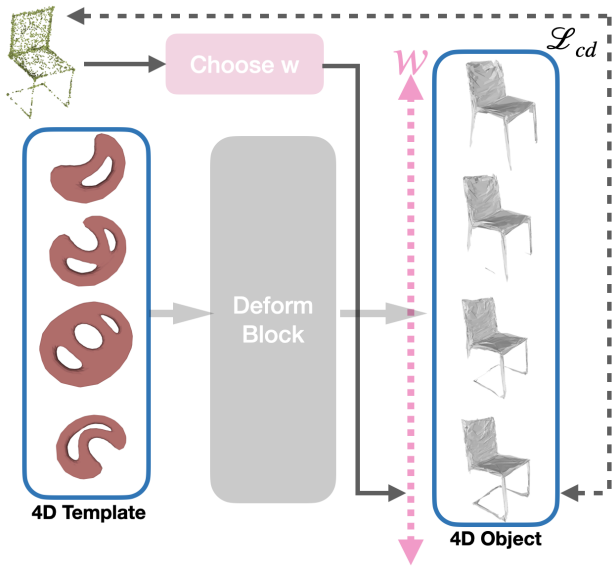

Figure 5. The pipeline of 4D reconstruction. There is one more slice selection module than 3D reconstruction, which allows the network to find suitable slices by itself.

To achieve this goal, we built another block to choose which slices of the 4D object should be used for a specific point cloud inputs. Specifically, $w \in \mathbb{R}$, $\{p^i\}_{i=1}^{\mathcal{N}}$ is input point cloud, the network is: $P : \mathbb{R}^{3\times\mathcal{N}} \rightarrow \mathbb{R}$. $w = P(\{p^i\}_{i=1}^{\mathcal{N}})$. $\mathcal{M}$ represents sliced meshes. $\mathcal{T}$ represents output 4D object.

$$\mathcal{M} = Slice(\mathcal{T}, P(\{p^i\}_{i=1}^{\mathcal{N}})) \quad (13)$$

The encoder of this network P is a PointNet [26] with latent code 256, the decoder of this block is an MLP with inputs latent code and output a number w, which represents the fourth dimension of a 4D object be used to slice a 3D mesh. The final output of this part is a 3D mesh $\mathcal{M}$, which is sliced from the deformed 4D template by the fourth dimension w.

In the 4D object reconstruction task, the input of deform block is constant. It means in 4D object reconstruction, our network over-fit to several point clouds and outputs a constant 4D object with different 3D slices.

We train our network for 1500 iterations with a learning rate of 10^{-4}. The loss function in this task is equation 9. We choose one point cloud of the input point clouds as deform input and use a pre-trained network from 3D reconstruction network in Section 3.3. Train this network needs an RTX 3090 GPU for several minutes.

4. Experiments

We conduct three types of experiments to validate the proposed representation. First, we analyze the representation power of our method by evaluating how difference the network reconstructs complex 4D shapes from different slices. This gives us the possibility to reconstruct complex 4D geometry. Second, we demonstrated 4D reconstruction results including shapes and topologies. Finally, we validate 3D reconstruction on noisy point clouds and compare the performance of our method to several baselines.

Dataset: For all of our experiments we use the ShapeNet [3] provided by [5]. We also use the same train/test/validation split as [11].

4.1. Representation Power

In our first experiment, we investigate how well our representation represents 4D geometry. We have constructed a new metric, which describes the degree of change of a 4D object in different sections. If a 4D object does not change much in different slices, then adding a dimension does not make much sense. A reasonable 4D object we want is that the Chamfer Distance between a certain slice and 3D object is very low, and varies greatly in different slices. For this reason, we construct this metric.

Specifically, our 4D object is the result of 3D reconstruction, in the 3D reconstruction task when $w = 0$, which is marked as $\mathcal{M}_0$, the slice where $w = \alpha$ is marked as $\mathcal{M}_1$, and the slice where $w = -\alpha$ is marked as $\mathcal{M}_2$. Our metric is the differences between $\mathcal{M}_0$ and $\mathcal{M}_1, \mathcal{M}_2$. The larger the value, the greater the degree of change, and the better,

Metrics								
ChamferL1	ChamferL2	Diff	#V	#T	#Tem	Parameters	Topology-Agnostic	Method
0.294	1.356	0.205	33306	184320	**1**	11.7M	✗	Atlas-O 4D
0.263	0.979	0.203	39325	180000	25	177.7M	✔	Atlas-25 4D
0.239	**0.754**	**0.711**	**3739**	**13720**	**1**	**5.2M**	✔	NeuralSlice

Table 1. Details about different 4D representations. Among them, ChamferL1 and ChamferL2 represent the chamfer distance between our 4d object and the input point cloud on specific slice $w = 0$. #V represents the number of points of the 4d object, #T represents the number of tetrahedrons. #Tem represents the number of template. Parameters represent the size of the network. Diff is a new indicator we established. It is worth noting that although Atlas-25 can represent different topologies, the 3D meshes it generates are always homomorphism to 25 patches. In other words, it cannot automatically adapt to the topology, but our method can. ChamferL1 $\times 10$ and ChamferL2 $\times 10^3$.

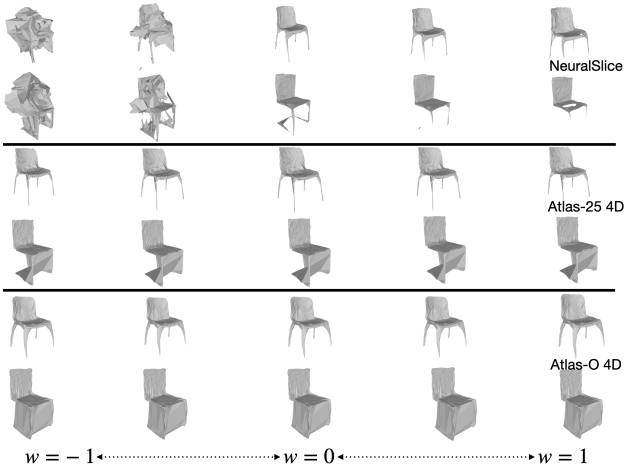

Figure 6. 4D representation power: NeuralSlice has different shapes and topologies in different slices, while other methods change very slightly in the fourth dimension. In other words, Neural Slice has great potential in 4D reconstruction tasks.

in the table is Diff.

$$\mathcal{CD}(\mathcal{M}_a, \mathcal{M}_b) = \sum_{x \in \mathcal{M}_b} \min_{y \in \mathcal{M}_a} \|x - y\|_2^2 + \sum_{y \in \mathcal{M}_a} \min_{x \in \mathcal{M}_b} \|x - y\|_2^2 \tag{14}$$

$$Diff = \frac{1}{2}(\mathcal{CD}(\mathcal{M}_0, \mathcal{M}_1) + \mathcal{CD}(\mathcal{M}_0, \mathcal{M}_2)) \tag{15}$$

Since there is no similar method to achieve similar functions, we modified AtlasNet [10] as a baseline. In AtlasNet, the decoder is: $p_O^i = F_{\Theta|z}\left(p_I^i, z\right), p_I^i \in \mathbb{R}^3, p_O^i \in \mathbb{R}^3$. p_I^i is the input template and p_O^i is the deformed template. For him to generate 4D objects, we added an input α to his decoder to indicate what the 3D meshes generated when $w = \alpha$ is. When the fourth dimension changes, we generate a series of 3D meshes. Our baseline is $p_O^i = F_{\Theta|z}\left(p_I^i, w, z\right)$. After getting this series of 3D meshes, we can reconstruct these 4D objects with this series of 3D meshes like [21] does.

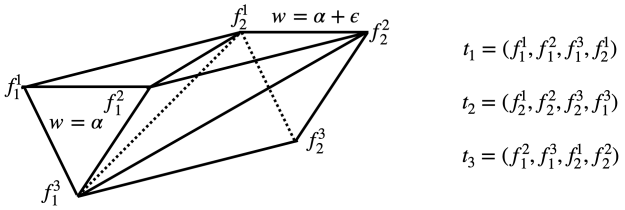

Figure 7. Two adjacent triangular faces f_1, f_2 form a triangular prism, which can be decomposed into three tetrahedrons t_1, t_2, t_3.

Specifically, for a slice $w = \alpha$, when w has a tiny change $w = \alpha + \epsilon$. Since $w = \alpha$ and $w = \alpha + \epsilon$ are homomorphism, each triangle pair can form a triangular prism. Each triangular prism can be disassembled into three tetrahedrons, these tetrahedrons are the basic units that constitute a 4D object. As illustrated in Figure 7.

At the same time, we also measured the chamfer distance between the specific slice $w = 0$ and the output point cloud. This metric represents the accuracy of the 3D reconstruction of these 4D representations.

For a 4D representation, memory efficiency is also very important. In this experiment, we measured the number of points and tetrahedrons required for these 4D representations. It also measures the size of the network that generates 4D objects. Topology-Agnostic is also an important indicator. NeuralSlice can also achieve better results with fewer templates. Our method has great advantages in all indicators. As illustrated in Table 1.

Our method can also automatically adapt to the topology. Unlike us, the slices generated by Atlas-O are always homomorphism to a sphere. Although Atlas-25 can also represent different topologies, the slices it generates are always homomorphism to 25 patches. As illustrated in Figure 6.

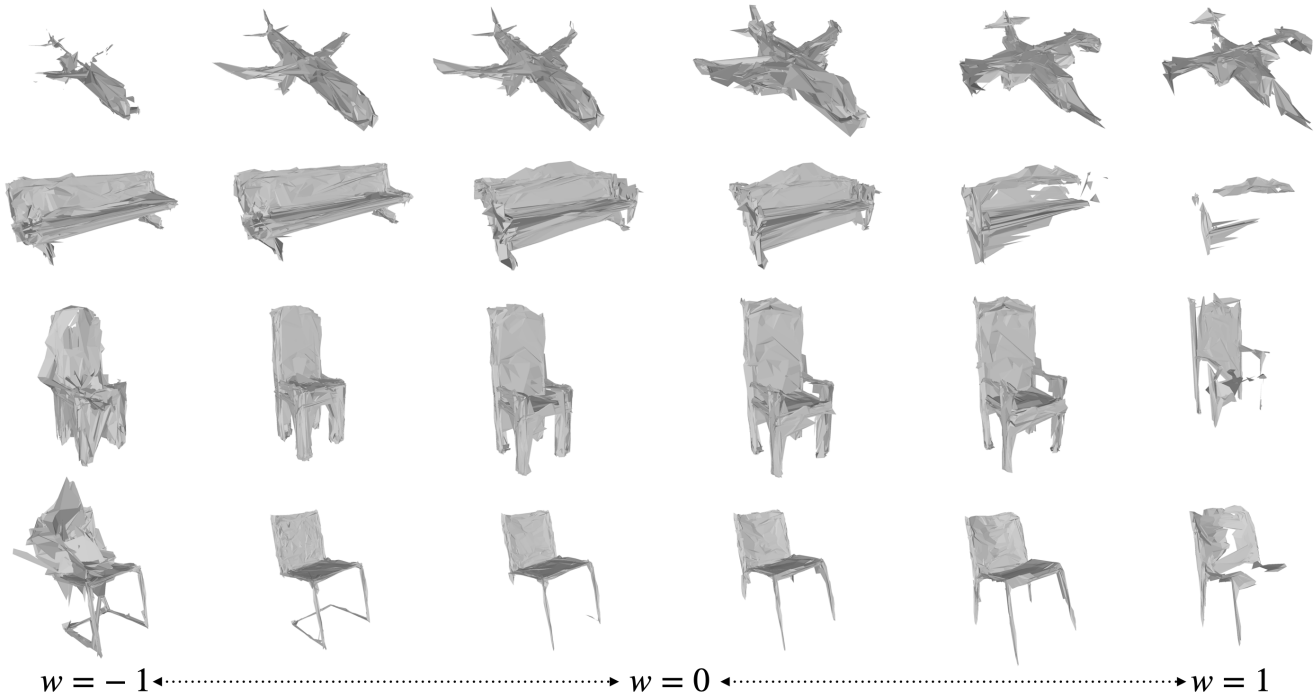

Figure 8. 4D reconstruction results. NeuralSlice can represent different shapes and topologies in different slices. With the change of slices, a 4D object can represents different shapes and topologies. This means that for a 4D object, the difference in the fourth dimension represents 3D meshes of different shapes and topologies. The meshes in each row in the figure **come from the same 4D object.**

4.2. 4D Reconstruction Results

In the previous experiment, we explained the representation power, but this only shows that our representation has the potential to represent complex geometries, rather than the ability to represent complex geometries. So in the second experiment, we will illustrate the performance of our method on 4D reconstruction.

In this experiment, our network input is constant, while outputting a 4D object. Ours choose w block P inputs different point clouds and for each point cloud output a number w, which represents which slice of this 4D object is it. The output of NeuralSlice is only one 4D object, but it can represent different shapes and topologies. As illustrated in Figure 8.

We manually selected four groups of 3D point clouds with topological changes as the data for this experiment. Both NeuralSlice and baselines are overfitted in a set of 3D point clouds.

The baselines we chose are similar to Section 4.1. In order to be fair, these baselines must have the following characteristics: input a constant point cloud, have different performance when the fourth dimension w changes, and have a choose w block to locate the fourth coordinate. So we made its input constant and added a choose w block P to the baselines of the previous experiment, outputting different meshes and supervising when w changes.

	NeuralSlice	Atlas-O 4D	Atlas-25 4D
chair1	**0.247**	2.222	1.137
chair2	**0.688**	1.648	1.176
chair3	**0.685**	2.230	1.601
plane	**0.439**	1.546	1.067
mean	**0.515**	1.912	1.246

Table 2. 4D reconstruction results. $\mathcal{CD}_4$ of each method and 4D object. NeuralSlice has advantages in all objects. All the data here are $\times 10^3$

We also compared the chamfer distance between the corresponding slice and the corresponding point cloud called $\mathcal{CD}_4$ as the quantified result. As illustrated in Table 2. $Slice$ represents slice block. $\mathcal{T}$ represents the output 4D object. P represents choose w block. $\{p^i\}$ represents the input point cloud.

$$\mathcal{CD}_4 = \sum_j \mathcal{CD}(Slice(\mathcal{T}, P(\{p^i\}^j)), \{p^i\}^j) \tag{16}$$

Our method is not only superior to baselines in topology and shape diversity, but our data quantification performance

Metrics		
ChamferL1 $\times 10$	ChamferL2 $\times 10^3$	Method
0.419	2.976	NMF-3D
0.341	1.873	NMF-6D
0.290	1.312	Atlas-O
0.262	0.966	Atlas-25
0.241	0.762	NeuralSlice
0.253	0.857	NeuralSlice(with smooth)
0.239	**0.754**	NeuralSlice(with noise)

Table 3. 3D reconstruction result. The performance of our method on chamfer distance is not only better than Atlas-O, NFM, these methods deform a template like us. It is also better than Atlas-25 deformed with 25 templates. For the noisy part, we apply Gaussian noise with zero mean and standard deviation of 0.005.

is also superior to baselines. As illustrated in Table 2 and Figure 8.

4.3. 3D Reconstruction from Point Clouds

The first two experiments illustrate the performance of NeuralSlice on 4D objects. And NeuralSlice can also do 3D reconstruction tasks. In this experiment, we will explain the performance of our method in 3D reconstruction tasks.

In the 3D reconstruction task, our main advantage is that it can automatically adapt to the topology without subsequent modification, and we only have one template. The reconstruction accuracy of NeuralSlice is also very high.

We compared the following baselines: NFM-3D, NFM-6D [11], Atlas-O, Atlas-25 [10]. Among them, NFM-6D is our improvement to NFM-3D. Specifically, the core module of NFM is Neural ODE [27], and this module has some problems with homomorphism mapping. A NODE $F : \mathbb{R}^3 \to \mathbb{R}^3$ is not capable of expressing all homomorphism mappings in $\mathbb{R}^3$. But a NODE $F : \mathbb{R}^6 \to \mathbb{R}^6$ can represent all homomorphism mappings in $\mathbb{R}^3$. Proved by [37]. So we changed the core module to $F : \mathbb{R}^6 \to \mathbb{R}^6$ called NFM-6D.

In this experiment, all methods are scaled into [-1, 1] and the input is 2562 points, sample 2562 points on the output meshes, and calculate the chamfer distance. It measures the difference between two point clouds.

The performance of our method is more powerful than other template-based methods in chamfer distance. As illustrated in Table 3.

The greater advantage of our method is topology. Specifically, NFM, and Atlas-O only deform one template, which causes the generated 3D meshes to always be homomorphism to a sphere. Although Atlas-25 can represent different topologies, there is no connection between 25 patches, which makes it not a mesh, but a combination of multiple patches. And the meshes generated by Atlas-25 are always homomorphism to 25 patches. NeuralSlice can effectively use the fourth dimension and automatically adapt to changes in topology. As illustrated in Figure 9.

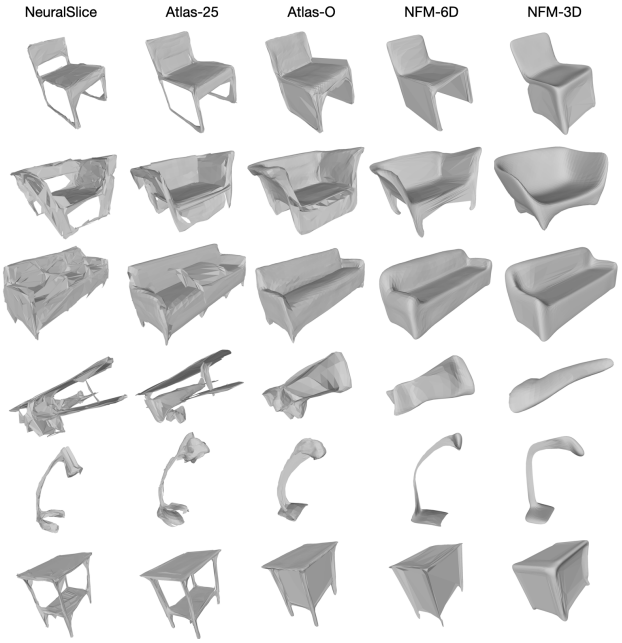

Figure 9. 3D reconstruction result. It can be seen that our method has advantages in topology. In some cases, only our method can represent the correct topology. Among them, NMF-3D and NMF-6D have over-smooth problems. NMF-3D, NMF-6D, Atlas-O cannot represent different topologies. Atlas-25 can represents the topology is also limited, and it does not generate complete meshes.

But our method also has some problems. The meshes we generate are not smooth, with many burrs and uneven surfaces. The opposite of us is NMF, the meshes they generate are somewhat over-smooth, which leads to their low accuracy.

5. Conclusion

In this paper, we introduced a new representation for 4D geometry. In contrast to existing representations, ours has different shapes and topologies from different slices and can hence be used to represent complex 4D geometries.

Our method also has some problems, such as the surface is not smooth enough, the model is not watertight, and it is difficult to calculate the surface's normal direction. This is the direction of our future work.

References

[1] E. A. Abbott and A. Square. "flatland: A romance of many dimensions". *American Journal of Hematology*, 88(1):77, 2008. 2

[2] Praveen Bhaniramka, Rephael Wenger, and Roger Crawfis. Isosurfacing in higher dimensions. *Proceedings of the IEEE Visualization Conference*, 05 2000. 2

[3] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, H. Su S. Song, J. Xiao, L. Yi, and F. Yu. Shapenet: An information-rich 3d model repository. *arXiv.org, 1512.03012*, 2015. 5

[4] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2

[5] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European Conference on Computer Vision (ECCV)*, 2016. 2, 5

[6] Alan Chu, Chi-Wing Fu, Andrew Hanson, and Pheng-Ann Heng. Gl4d: A gpu-based architecture for interactive 4d visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1587–1594, 2009. 1, 2

[7] Ö. Çiçek, A. Abdulkadir, S.S. Lienkamp, T. Brox, and O. Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation. *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2016. 4

[8] M. Emmer and T. F. Banchoff. Beyond the third dimension: Geometry, computer graphics, and higher dimensions. *Computers & Graphics*, 25(3):385, 1990. 2

[9] H. Fan, S. Hao, and L. Guibas. A point set generation network for 3d object reconstruction from a single image. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[10] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 3, 6, 8

[11] K. Gupta and M. Chandraker. Neural mesh flow: 3d manifold mesh generation via diffeomorphic flows. In *Conference on Neural Information Processing Systems (NeurIPS).*, 2020. 1, 2, 3, 5, 8

[12] D. Hilbert and S. Cohn-Vossen. Anschauliche geometrie. *Mathematical Gazette*, 36(317), 1932. 2

[13] Z. Hui and A. J. Hanson. Physically interacting with four dimensions. In *Advances in Visual Computing, Second International Symposium, ISVC 2006, Lake Tahoe, NV, USA, November 6-8, 2006 Proceedings, Part I*, 2006. 2

[14] B. Jiang, Y. Zhang, X. Wei, X. Xue, and Y. Fu. Learning compositional representation for 4d captures with neural ode. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[15] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *Proceedings of International Conference on Machine Learning (Proceedings of International Conference on Machine Learning (ICML))*, 2015. 5

[16] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, May 2021. 1, 2

[17] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, pages 163–169, 1987. 1, 2

[18] Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Mohamed Elgharib, Pascal Fua, Hans-Peter Seidel, Helge Rhodin, Gerard Pons-Moll, and Christian Theobalt. Xnect: Real-time multi-person 3d motion capture with a single rgb camera. *ACM Trans. Graph.*, 2020. 2

[19] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2

[20] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *IEEE International Conference on Computer Vision (ICCV)*, October 2019. 2

[21] Ikuru Otomo, Masahiko Onosato, and Fumiki Tanaka. Direct construction of a four-dimensional mesh model from a three-dimensional object with continuous rigid body movement. *Journal of Computational Design and Engineering*, 55, 04 2014. 1, 2, 6

[22] Junyi Pan, Xiaoguang Han, Weikai Chen, Jiapeng Tang, and Kui Jia. Deep mesh reconstruction from single rgb images via topology modification networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 1

[23] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2

[24] A. Paszke, S. Gross, F. Massa, A. Lerer, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Conference on Neural Information Processing Systems (NeurIPS).*, 2019. 5

[25] Sergey Prokudin, Christoph Lassner, and Javier Romero. Efficient learning on point clouds with basis point sets. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[26] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 4, 5

[27] T. Q. Chen Ricky, Rubanova Yulia, Bettencourt Jesse, and Duvenaud. David. Neural ordinary differential equations. In *Conference on Neural Information Processing Systems (NeurIPS).*, 2018. 8

[28] Peng Songyou, Niemeyer Michael, Mescheder Lars, Pollefeys Marc, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 2, 3, 4

[29] Marc ten Bosch. 4d toys. https://4dtoys.com. 1

[30] Marc ten Bosch. Miegakure. https://miegakure.com. 1

[31] Marc ten Bosch. N-dimensional rigid body dynamics. *ACM Trans. Graph.*, 39(4):55:1–55:6, jul 2020. 1, 2

[32] N. Wang, Y. Zhang, Z. Li, Y. Fu, and Y. G. Jiang. Pixel2mesh: 3d mesh model generation via image guided deformation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (IEEE TAPMI)*, PP(99):1–1, 2020. 1, 2

[33] Chao Wen, Yinda Zhang, Zhuwen Li, and Yanwei Fu. Pixel2mesh++: Multi-view 3d mesh generation via deformation. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 2

[34] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Conference on Neural Information Processing Systems (NeurIPS).*, 2016. 2

[35] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[36] X. Yan, C. W. Fu, and A. J. Hanson. Multitouching the fourth dimension. *Computer*, 45(9):80–88, 2012. 2

[37] Han Zhang, Xi Gao, Jacob Unterman, and Tom Arodz. Approximation capabilities of neural ordinary differential equations. *arXiv preprint arXiv:1907.12998*, 2(4):3–1, 2019. 8