Extended Differentiable Marching Cubes by Manifold-Preserving Shape Inflation

Kiichi Itoh
itok@den.t.u-tokyo.ac.jp
Tatsuya Yatagawa
tatsy@den.t.u-tokyo.ac.jp
Yutaka Ohtake
ohtake@den.t.u-tokyo.ac.jp
Hiromasa Suzuki
suzuki@den.t.u-tokyo.ac.jp

School of Engineering,
The University of Tokyo,
Tokyo, Japan

Abstract

This paper introduces an extended differentiable marching cubes (DMC) method for end-to-end learning of precise 3D surface geometries using a neural network. The original DMC method extracts the isosurface using a fixed-size voxel grid, similar to the traditional marching cubes. Therefore, the original method involves a trade-off between output resolution and memory consumption. In contrast, there remains room to increase the output resolution without increasing the number of voxels because an output surface often exists over a limited number of voxels. According to this observation, our method deforms an input point cloud to occupy the voxel grid as widely as possible, thereby refining small parts of the target shape. To obtain such deformation, we apply normalizing flow (NF), typically used to transform probability density functions. Its invertibility allows us to reproduce a mesh for the input point cloud by cancelling the deformation of the mesh obtained for the deformed point cloud using DMC. To obtain appropriate deformation, NF is conditioned by a global shape feature and is trained by several loss functions to inflate the input shape while preserving its manifold structure. We tested the proposed method with two shape datasets and showed that our extended DMC achieves higher performance than the original DMC, even used with a simple deformation.

1 Introduction

Three-dimensional (3D) shapes can be discretely represented using various formats, including voxels, point clouds, meshes, and implicit fields, for different computational purposes. These formats are often converted to meshes for graphics applications, such as rendering and modelling, because of the good balance between data size and the capability of shape representation. Therefore, surface reconstruction from the other formats has been a traditional problem in computer vision and graphics [1]. Early approaches have focused on converting the input point set to an implicit field [7, 12, 17, 29] and have extracted the output surface using the traditional marching cubes (MC) algorithm [26]. In contrast, recent data-driven

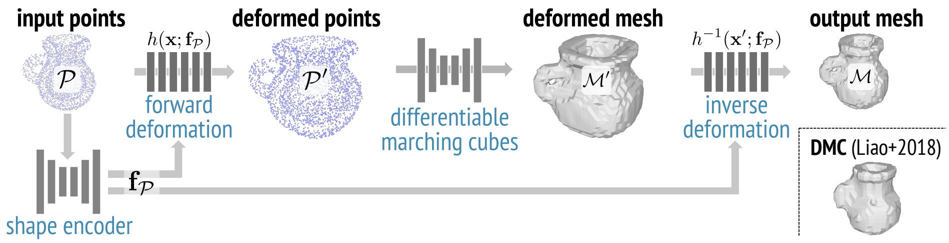

Figure 1: Our method deforms input points $\mathcal{P}$ by a deformation h defined by NF. The deformed points $\mathcal{P}'$ is supplied to the DMC layer to obtain the deformed mesh $\mathcal{M}'$. We leverage the invertibility of the NF to obtain the output mesh $\mathcal{M}$ by cancelling the deformation on $\mathcal{M}'$.

approaches train a deep neural network with large-scale datasets to achieve surface reconstruction from a point cloud. A deep neural network can define an implicit field [3, 27, 30], while an isosurface must be extracted using the traditional MC algorithm, which prohibits end-to-end learning of surface geometries. A more recent study, MeshSDF [32], made isosurface extraction differentiable with respect to the 3D position on the isosurface. MeshSDF requires pretraining the neural network, which represents a signed distance field (SDF), using the ground truth SDF, although obtaining such ground truth SDF is often challenging due to the presence of non-manifold surfaces whose inside and outside are not well defined. Other approaches use local patches [10, 45] and multi-resolution triangles [40, 41] to represent the surfaces, although they do not guarantee manifoldness of the output surfaces.

For end-to-end learning of surface geometries, Liao *et al.* [23] proposed a differentiable MC (DMC) algorithm, whose differentiability enables back-propagation of a loss defined for the output mesh to the network representing the implicit field. In this way, they achieved higher performance than the previous methods regressing the occupancy of voxels [3, 27] or SDF [30]. However, DMC is performed on a 3D grid of voxels and performs 3D convolutions on the grid, which requires a significant amount of memory on a finer voxel grid to improve the resolution of output meshes.

To address this problem, we deform a given point cloud before extracting an isosurface using DMC, aiming to increase the area of the isosurface and the resolution of the output triangle mesh. As shown in Fig. 1, DMC extracts an isosurface $\mathcal{M}'$ for a deformed point cloud $\mathcal{P}'$. Deformation of the extracted isosurface must be cancelled to obtain the output mesh $\mathcal{M}$ corresponding to the input points $\mathcal{P}$, meaning that the deformation h must be invertible. We define deformation h using the normalizing flow (NF), which is typically used to model probability density functions without closed-form representations. The NF deforms the known probability density function using a series of learnable deformations, and the learned deformation can be invertible. The flexibility and invertibility of the NF must be advantageous for our purpose of deforming point clouds. However, simple application of the NF to point and mesh deformation can complicate the surface extraction process by DMC. Therefore, we introduce several techniques to achieve preferable deformation for DMC.

2 Related Work

Marching cubes on various grid primitives. While the original MC algorithm [26] is used for a rectilinear grid of voxels, numerous studies have been conducted to define the MC algorithm on other types of grids, as summarised in a survey [28]. For example, the marching tetrahedra method [38] divides a cubic cell into four tetrahedra and calculates the triangles for each tetrahedron in the same manner as the original MC. Furthermore, the MC algorithm is extended for a grid of octahedra and hexahedra [2] and scientific data defined on spherical and polar coordinate systems [9]. Hierarchical data structures such as an octree are also popular choices to concentrate computational resources to the detailed part of shapes [35, 37]. In contrast, several approaches preserve sharp edges by optimising vertex positions inside each cell of the grid [15, 20], while a recent study proposed a learning-based optimisation of vertex placement inside the grid cells [4].

In contrast, we deform a given point cloud and apply DMC to extract a surface. In this case, we can interpret the regular grid for the deformed point cloud as an irregular and non-rectilinear grid for the input point cloud. To the best of our knowledge, our method is the first extension of the MC algorithm, performed on such a nonlinearly deformed grid, which is a technically interesting aspect of our study.

3D shape deformation. A goal of non-rigid shape deformation is to align the same or similar objects with different poses. Early studies have targeted the extraction of good shape descriptors for aligning shapes, often based on local geometric properties [39]. In contrast, recent approaches acquire such descriptors by learning a large number of shape pairs with ground truth correspondences [6, 24, 25, 42]. Such deformation has been applied to reconstruct the surface geometry by fitting a template mesh to a point cloud [22, 36] or a deficient mesh [21]. Recent approaches [43, 44] have been employed to deform template meshes using deep neural networks to predict the 3D geometry associated with a single image.

However, all these approaches deform a given shape to align with another target shape. The aim of our deformation is ambiguous because the target shape for extracting a high-resolution mesh with DMC cannot be defined clearly. Therefore, we leveraged the flexibility of learning-based deformation, trained to increase the surface area of the output mesh extracted by DMC on a grid with a limited number of voxels.

3 Extended Differentiable Marching Cubes

This section describes the algorithm of DMC [23] briefly, which corresponds to "no deform" in Fig. 2(a). Then, we introduce a simple linear stretch to increase the area of the isosurface (see Fig. 2(b)). Although we apply NF to deform the point cloud to further increase the isosurface area, its direct application does not aid in isosurface extraction by DMC (see Fig. 2(c)). We introduce a new loss function to preserve an underlying manifold structure of the input point cloud, using vacant spaces in the grid to increase the resolution of extracted meshes (see Fig. 2(d)).

3.1 Differentiable Marching Cubes

DMC extracts a triangular mesh corresponding to an isosurface at a specific level set of the implicit field represented by a neural network. Similar to the traditional MC algorithm, DMC uses a set of cubes to define the triangles on an isosurface. In this study, we refer to each cube as a *cell* and assume that each corner of the cell corresponds to a *voxel*. Furthermore, we refer to a 3D collection of cubes or voxels as *grid*.

Let $\mathbf{D} \in \mathbb{R}^{N \times N \times N}$ be a volume data or a discretized SDF, where N is the number of voxels along each dimension, and $\mathbf{n} = (i, j, k) \in \mathbb{N}^3$ be a multi-index for each voxel. When

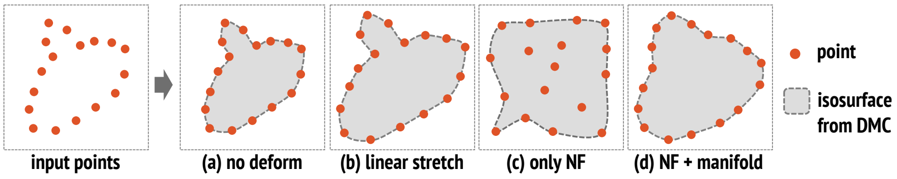

Figure 2: Illustrations for point clouds obtained by different deformation methods and corresponding isosurfaces extracted by DMC.

a voxel value $d_{\mathbf{n}}$ on a voxel $\mathbf{n}$ is larger than a predefined threshold λ, we assume that the voxel is occupied by an object. We further denote the occupancy data on the volume by $\mathbf{O} \in \{0,1\}^{N\times N\times N}$ and the occupancy of each voxel as $o_{\mathbf{n}} \in \{0,1\}$. For an edge $(\mathbf{n},\mathbf{n}')$ of the cell between two voxels, a vertex is placed when $o_{\mathbf{n}} \neq o_{\mathbf{n}'}$. The location of the vertex $\mathbf{v}$ is determined by the voxel values of two endpoints $\mathbf{v}_{\mathbf{n}}$ and $\mathbf{v}_{\mathbf{n}'}$, *i.e.* $\mathbf{v} = (1-u)\mathbf{v}_{\mathbf{n}} + u\mathbf{v}_{\mathbf{n}'}$ using $u = (d_{\mathbf{n}} - \lambda)/(d_{\mathbf{n}} - d'_{\mathbf{n}})$. We refer to this interpolation ratio u as a displacement. Although there are $3N^2(N-1)$ edges on a voxel grid with N voxels along each dimension, we denote displacement data simply as $\mathbf{U} \in \mathbb{R}^{N\times N\times N\times 3}$ by introducing redundant voxels. Then, the MC algorithm processes each cubic cell defined by $2\times 2\times 2$ neighbouring voxels. The binary patterns of these eight corners define a topology $\mathbf{T}$, and the topology $\mathbf{T}$ and the displacement for the cell define a set of triangles assigned.

DMC [23] obtains the occupancy $\mathbf{O}$ and displacement $\mathbf{U}$ from an input point cloud using a deep neural network, whereas the traditional MC evaluates an implicit field obtained, *e.g.* by solving a Poisson equation [16, 17]. DMC first encodes the input point cloud as a set of feature vectors $\mathbf{F} \in \mathbb{R}^{N\times N\times N\times C}$, where C is the number of feature dimensions. Then, $\mathbf{F}$ is regularized by an hourglass network to obtain $\mathbf{O}$ and $\mathbf{U}$. Different from traditional MC, the occupancy of voxels is defined stochastically as $\mathbf{O} \in [0,1]^{N\times N\times N}$, which defines the topology for each cell stochastically as well. The probability density for a topology $\mathbf{T}$ induced by the stochastic occupancy is obtained by a Bernoulli distribution:

$$p_{\mathbf{n}}(\mathbf{T}) = \prod_{\mathbf{m}\in\{0,1\}^3} (o_{\mathbf{n}+\mathbf{m}})^{t_{\mathbf{m}}} (1 - o_{\mathbf{n}+\mathbf{m}})^{1-t_{\mathbf{m}}},$$

where $t_{\mathbf{m}} \in \{0,1\}$ is a random variable which represents whether a cell $\mathbf{m}$ is occupied $(t=1)$ or unoccupied $(t=0)$. The distance between the input point cloud and a stochastic surface representation induced by $\mathbf{T}$ is integrated over all the voxels to define the point-to-mesh loss:

$$\mathcal{L}_{\text{mesh}} = \sum_{\mathbf{n}} \mathbb{E}_{p_{\mathbf{n}}(\mathbf{T})} \left[\sum_{\mathbf{x}\in\mathcal{P}_{\mathbf{n}}} \Delta(\mathcal{M}_{\mathbf{n}}(\mathbf{T},\mathbf{U}),\mathbf{x}) \right], \tag{1}$$

where $\mathcal{P}_{\mathbf{n}}$ is a part of the target point cloud in cell $\mathbf{n}$, $\mathcal{M}_{\mathbf{n}}(\mathbf{T},\mathbf{U})$ is the mesh induced by topology $\mathbf{T}$ and displacement $\mathbf{U}$, and Δ is a point-to-triangle distance. The original DMC is trained using three more losses: the occupancy loss $\mathcal{L}_{\text{occ}}$, occupancy smoothness loss $\mathcal{L}_{\text{smooth}}$, and curvature loss $\mathcal{L}_{\text{curve}}$. The overall training loss $\mathcal{L}_{dmc}$ is defined as a weighted sum of these losses:

$$\mathcal{L}_{\text{dmc}} = w_{\text{mesh}}\mathcal{L}_{\text{meth}} + w_{\text{occ}}\mathcal{L}_{\text{occ}} + w_{\text{smooth}}\mathcal{L}_{\text{smooth}} + w_{\text{curve}}\mathcal{L}_{\text{curve}}. \tag{2}$$

3.2 Manifold-preserving shape inflation

The purpose of deformation for a given point cloud is to improve the resolution of the output mesh without increasing the number of voxels in a grid. The most straightforward approach is to linearly stretch the input point cloud along each dimension. Let $x_\bullet, y_\bullet, z_\bullet$ ($\bullet \in \{\min, \max\}$) be the minimum and maximum $\{x,y,z\}$-coordinates of a given point cloud. Then, we can define the linear stretch with margin δ at the boundary of the grid, as follows:

$$x = \rho \frac{x - x_{\min}}{x_{\max} - x_{\min}} + \delta, y = \rho \frac{y - y_{\min}}{y_{\max} - y_{\min}} + \delta, z = \rho \frac{z - z_{\min}}{z_{\max} - z_{\min}} + \delta, \tag{3}$$

where $\rho = N - 1 - 2\delta$ is the size of the available space after the margin δ at both sides is excluded. However, the linear stretch is often suboptimal when an object diagonally lies toward the voxel grid and has some protruding parts, as shown in Fig. 2(b). Thus, we deform the input point cloud using an approach based on a neural network and extract a surface mesh for the deformed point cloud.

Shape deformation by Real NVP. As previously explained with Fig. 1, the deformation defined by the neural network must be invertible because we need to cancel the deformation of the deformed mesh $\mathcal{M}'$ to obtain the output mesh $\mathcal{M}$ corresponding to the input points $\mathcal{P}$. Hence, we use the NF to ensure the invertibility of deformation. Specifically, we employ real-valued non-volume preserving transformations (Real NVP) [8], which is a simple yet powerful approach to transform a shape of function using a series of learnable affine transformations. Meanwhile, most discussions below can also be applied to other NFs [11, 19].

The network of Real NVP comprises layers called an affine coupling layer. In this layer, an input vector is separated into two parts along its dimension, and one is used to define the parameters to deform the other. We separate the three coordinates $\{x,y,z\}$ of a point into two, *e.g.* $\{x\}$ and $\{y,z\}$. Then, affine parameters to transform $\{x\}$, *i.e.* scalar scaling s and translation t, are calculated using $\{y,z\}$:

$$x' = sx + t, \qquad s = \sigma(y,z), \quad t = \tau(y,z), \tag{4}$$

where σ and τ are defined by neural networks. Let f_i be the i-th transformation applied to either of the three coordinates. Then, we can define the overall deformation as $\mathbf{x}' = (f_{n-1} \circ \cdots \circ f_0)(\mathbf{x}) \equiv h(\mathbf{x})$ by n consecutive transformations, which we refer to as a *shape deformation layer*. In Eq. (4), only x is transformed, and other coordinates, *i.e.* y and z, remain unchanged. Therefore, s and t in Eq. (4) remain the same when they are calculated for $\{x', y, z\}$, which allows us to define the inverse deformation simply by $x = (x' - t)/s$.

When training Real NVP for probability density estimation, we often minimise the negative log determinant of its Jacobian. Because a Jacobian represents the change in a differential volume, the minimisation of the negative log determinant seems to be appropriate to achieve our target shape inflation. However, our preliminary experiment showed that the simple application of Real NVP could disorder the original shape of an object because Real NVP does not consider the adjacency of each point to its neighbours. Eventually, only using the NF disorders the arrangement of points and prohibits DMC from extracting an isosurface corresponding to the input point cloud (see Fig. 2(c)).

Manifold-preserving loss. To maintain the adjacency of points, we introduce a loss function inspired by locally linear embedding (LLE) [33], which is a popular approach of manifold learning. Let $K(i)$ be the index set of the k-nearest neighbours of $\mathbf{x}_i$. Then, we can obtain affine combination weights $\mathbf{w}_i = \{w_{ij}\}_{j \in K(i)}$ for each point $\mathbf{x}_i$ and its neighbours $\{\mathbf{x}_j\}_{j \in K(i)}$ as follows:

$$\mathbf{w}_i = \operatorname*{argmin}_{\mathbf{w}_i} \left\| \mathbf{x}_i - \sum_{j \in K(i)} w_{ij} \mathbf{x}_j \right\|^2 \quad \text{such that} \quad \sum_{j \in K(i)} w_{ij} = 1.$$

The solution for this constrained least-squares problem can be obtained by solving a small linear system. We can preserve the manifold structure even after deformation by constraining the deformed point cloud to have the local linearity defined by $\mathbf{w}_i$. A loss function to preserve the manifold structure is defined as follows:

$$\mathcal{L}_{\text{mp}} = \sum_{i=0}^{M-1} \left\| \mathbf{x}'_i - \sum_{j \in K(i)} w_{ij} \mathbf{x}'_j \right\|,$$

where M is the number of points in a given point cloud, and $\mathbf{x}'_i$ is the position of the i-th point after deformation. The loss prevents the NF from disordering the point arrangement and helps DMC in extracting the isosurface from the inflated point cloud (see Fig. 2(d)).

Point-to-mesh loss for deformed shapes. In the original DMC method, a point-to-mesh loss in Eq. (1) is defined as a weighted sum of the distances between a triangle induced by topology $\mathbf{T}$ and each target point $\mathbf{x}$. However, the deformation defined using Real NVP is nonlinear, and triangles obtained by DMC become curvy after inverse deformation. Thus, calculating the distance between a triangle and a point is not straightforward. Instead, we calculate the point-to-mesh loss in Eq. (1) for the deformed points and the deformed mesh rather than non-deformed ones. When the point-to-mesh loss is calculated for the deformed shapes, an input point cloud can be deformed to shrink to a single point because the most dominant $\mathcal{L}_{\text{mesh}}$ is minimised to be zero in that case. Therefore, we avoid such trivial solutions by penalising the shrinkage of the differential volume measured by the Jacobian of the deformation. We redefine the point-to-mesh loss in Eq. (1) by weighting each term for $\mathbf{x}'$ by the inverse of the Jacobian:

$$\mathcal{L}^{*}_{\text{mesh}} = \sum_{\mathbf{n}} \mathbb{E}_{p_{\mathbf{n}}(\mathbf{T})} \left[\sum_{\mathbf{x}' \in \mathcal{P}'} \left(\det \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}} \right)^{-1} \Delta(\mathcal{M}'_{\mathbf{n}}(\mathbf{T}, \mathbf{U}), \mathbf{x}') \right],$$

where $\mathcal{P}'$ and $\mathcal{M}'$ are the deformed point cloud and mesh, respectively, and the superscript $*$ denotes that the loss is defined for deformed shapes and uses the inverse Jacobian weight. We can also avoid over-inflation of shapes using the inverse Jacobian weights because it increases the loss where the Jacobian, *i.e.* the change in differential volume, is large.

Shape-aware deformation. To further enhance deformation quality, we extend the affine coupling layers of Real NVP by passing a feature vector representing the global shape of an input point cloud. We denote the NF conditioned by the feature vector as *C-NF* (conditional NF). We calculate the global shape feature $\mathbf{f}$ of a given point cloud using PointNet++ [31] and update σ and τ in Eq. (4) to take the feature vector, *i.e.* $s = \sigma(y, z, \mathbf{f})$ and $t = \tau(y, z, \mathbf{f})$. Even when the feature vector is passed to σ and τ, deformation is still easily invertible. The supplementary document provides further details of the conditional affine coupling.

We can make the feature vector more informative by introducing a classifier network to map feature vectors to the class labels. However, the training data for this extension needs to have class labels representing the shape category. Thus, we use the classifier only as an option, whereas we will later confirm its effect on the output mesh by an experiment.

Point repulsion loss. We regularize the point arrangement after deformation using a point repulsion loss [47]:

$$\mathcal{L}_{\text{rep}} = \sum_{i=0}^{N-1} \sum_{j \in K(i)} \eta(\|\mathbf{x}'_i - \mathbf{x}'_j\|) w(\|\mathbf{x}'_i - \mathbf{x}'_j\|)$$

$$\eta(r) = \exp(-r^2/\nu^2), \quad w(r) = -r,$$

where $\|\cdot\|$ is an ℓ_2 norm, and ν controls the distance between two points. For ν, we used a repulsion radiance of 10% of the size of an entire voxel grid. The index set $K(i)$ is the k-nearest neighbour of the input points. This loss penalizes any two points that are too close or too distant. Hence, as the inverse Jacobian weights do, it also avoids over-inflation.

Loss function. The total loss function for the DMC and shape deformation layers is:

$$\mathcal{L} = w_{\text{dmc}} \mathcal{L}^*_{\text{dmc}} + w_{\text{mp}} \mathcal{L}_{\text{mp}} + w_{\text{rep}} \mathcal{L}_{\text{rep}},$$

where $w_{\text{dmc}} = 1.0$ and $w_{\text{mp}} = w_{\text{rep}} = 0.1$ in the following experiments. A modified loss $\mathcal{L}^*_{\text{dmc}}$ for the DMC layer is defined by replacing $\mathcal{L}_{\text{mesh}}$ with $\mathcal{L}^*_{\text{mesh}}$ in Eq. (2). We exclude the curvature loss from Eq. (2) because we obtain a slight performance loss. The parameters of both the DMC and shape deformation layers are optimized using Adam [18] with a learning rate of $\gamma = 10^{-4}$ and $(\beta_1, \beta_2) = (0.9, 0.999)$.

4 Experiments

The dataset used in the original DMC paper only includes three shape categories, which is extremely few for verifying the robustness of the proposed method. Instead, we prepared another dataset by pre-processing the ModelNet40 dataset [46] with 40 shape categories. Because the original shape data in the ModelNet40 can be non-manifold, we first convert the meshes to be manifold using a technique to obtain watertight meshes [13]. Then, we sample random points on the surface using ordinary Poisson disk sampling [6]. The number of points may vary depending on the input meshes; we further sample 3,000 points randomly from them. In this way, we obtain 9,842 training samples[1] and 2,468 evaluation samples. We trained the network for 32^3 voxels over 10 epochs using mini-batches of size 8. The whole training required approximately five hours using a single NVIDIA Quadro A6000 GPU. The memory consumption of our method is less than 4 GB for the above experimental setup.

First, we confirm a key assumption of this study, i.e., whether learning-based point cloud deformation improves the performance of DMC. To this end, we compare our extended DMC (E-DMC) with the original DMC. We also test their variants where an input point cloud is linearly stretched along each dimension using Eq. (3) with $\delta = 1$. We denote these variants as *DMC+L* and *E-DMC+L*. We tested these approaches on a grid of 32^3 voxels. We compared these methods using the DMC layer with the traditional screened Poisson surface reconstruction (SPSR) [16] as a baseline. We applied SPSR for two different grid resolutions, *i.e.* 32^3 voxels (SPSR-5) and 256^3 voxels (SPSR-8). In addition, we followed the original DMC paper [23] and implemented the same baseline methods regressing either occupancy (Occ) and truncated SDF (TSDF). Furthermore, wTSDF refers to a variant of TSDF in which higher importance is given to voxels closer to the isosurface. We added

[1]We excluded a training data *door #0059*, which is a simple flat plane because we failed to process it using [13].

		SPSR-5	SPSR-8	Occ	TSDF	wTSDF	DMC [23]	DMC+L	E-DMC	E-DMC+L
ModelNet40	Acc.	1.204	0.454	0.642	0.429	0.580	1.102	0.171	0.454	**0.167**
	Comp.	0.264	**0.188**	0.525	0.803	0.628	0.253	0.200	0.307	0.190
	Overall	0.734	0.321	0.584	0.616	0.604	0.678	0.185	0.381	**0.178**
SHREC	Acc.	1.328	0.465	0.631	0.389	0.421	0.891	0.158	0.298	**0.155**
	Comp.	0.309	**0.196**	0.509	1.035	0.688	0.273	0.218	0.314	0.215
	Overall	0.819	0.330	0.570	0.712	0.554	0.582	0.188	0.306	**0.185**

Table 1: Comparison of the accuracy, completeness, and overall scores of previous methods and the proposed method (lower is better). Among the three metrics, E-DMC performs better than DMC, and E-DMC+L performs better than DMC+L. This behaviour is observed commonly when the models are tested with the ModelNet40 [46] and SHREC [34] datasets.

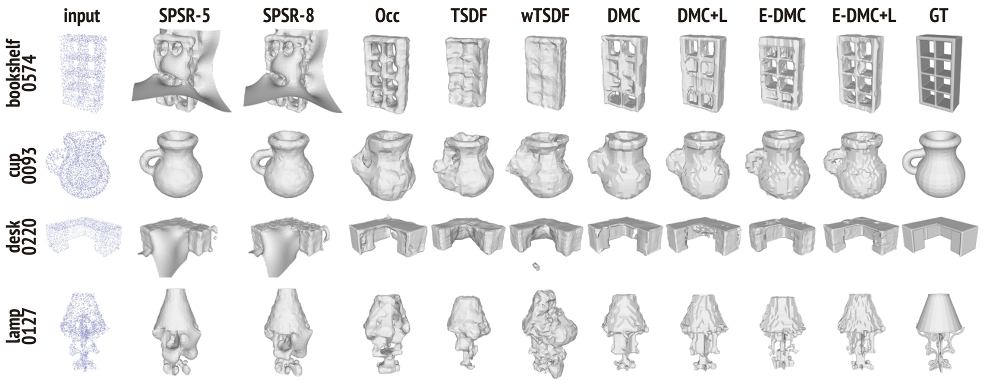

Figure 3: Visual comparison of our results with the original DMC [23] and other baseline methods. E-DMC+L performs the best among all in terms of the overall score and well reproduces shapes with non-zero genus such as a bookshelf and a cup with a handle.

Gaussian noise with $\sigma = 0.15$ voxels to the input point clouds during both training and evaluation. For quantitative comparison, we used a set of distance-based metrics [14], *i.e.* accuracy, completeness, and overall score (mean of the accuracy and completeness).

Figure 3 shows a visual comparison of the results. Although SPSR fails to reconstruct surface geometries when vertex normals are not estimated appropriately, other approaches using the DMC layer robustly reproduce surfaces without using vertex normals. In the cup and lamp examples, E-DMC successfully increased the resolution of output meshes to a level higher than those of DMC. Although the original DMC paper [23] reported that DMC outperforms other baselines, *i.e.* Occ, TSDF, and wTSDF, we obtained different results where DMC is worse than the baselines. In contrast, our E-DMC achieved significantly better scores than the other baselines. Comparing DMC+L and E-DMC+L, we found that E-DMC+L better reproduces the shapes with holes with non-zero genus, observed in the bookshelf and cup examples. Other results for all the 40 shape categories of ModelNet40 are provided in the supplementary document. In addition to the better performance shown by the visual comparison, the quantitative analysis in Table 1 shows that our methods, *i.e.* E-DMC and E-DMC+L, perform better than the baseline methods, *i.e.* DMC and DMC+L, respectively. All these results demonstrate that our learnable deformation improves the performance of surface extraction by DMC.

We also tested our method on the SHREC dataset [34] with 55 categories, using the model trained on ModelNet40 without any fine-tuning. We used 10,365 evaluation models

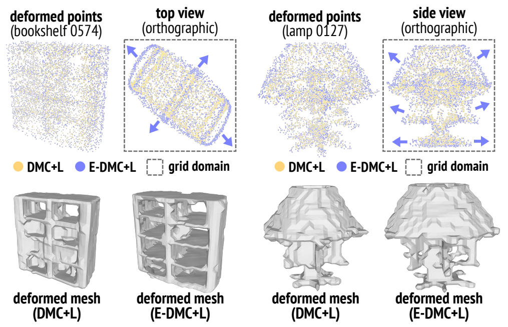

Figure 4: Deformed point clouds and the corresponding meshes extracted by the DMC layer.

	Occ (↑)	△ area (↓)	# of △ (↑)
DMC [■]	0.047	0.444	728
DMC+L	0.134	0.192	1556
E-DMC	0.113	0.227	1318
E-DMC+L	**0.179**	**0.160**	**1888**

Table 2: Comparison of the average number of occupied voxels (Occ, higher is better), the average triangle area (△ area, smaller is better), and the number of triangles of the output mesh (# of △, larger is better). In all these standards, E-DMC+L performed the best.

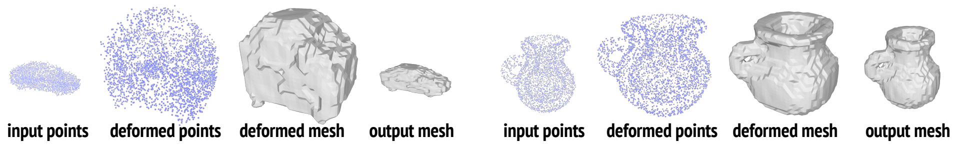

Figure 5: Visualization of intermediate and final results of our E-DMC+L. The roof part of the car is inflated more significantly than other regions in the left example (*car 0203*), and the spherical shape of the cup is deformed to be cubic so that it occupies the entire voxel grid in the right example (*cup 0093*).

of the SHREC dataset, which were processed equally as ModelNet40, and compared our methods and baseline methods using them. The bottom part of Table 1 shows the results with the SHREC dataset. As with the ModelNet40 dataset, E-DMC and E-DMC+L performed better than their baseline methods, *i.e.* DMC and DMC+L, respectively. This result implies that both the DMC and E-DMC (and their variants with linear stretching) equip sufficient generalizability to extract isosurfaces for point clouds not in the training data, and more importantly, E-DMC and E-DMC+L outperforms their baseline methods even in that case.

Next, we examine whether E-DMC+L, which performed the best in the previous experiment, deforms input shapes as intended. Figure 5 shows the deformed points and meshes for two example shapes. In the car example, the roof is inflated more significantly than the other regions and fills the upward region of a cubic domain of the voxel grid. In the cup example, the spherical shape of the cup is inflated to be cubic. Thus, the deformed shape occupies most regions of the voxel grid. These results demonstrate that our conditional Real NVP inflates the input shapes to occupy more voxels. Figure 4 shows a comparison between the deformations of DMC+L and E-DMC+L. The top images in this figure depict the deformed point clouds of DMC+L and E-DMC+L with yellow and blue dots, respectively. As indicated by blue arrows in the top and side views for each of the two shapes, the deformation by E-DMC+L moves the points to vacant regions near the grid corners. This deformation helps the DMC layer to reconstruct intricate structures like small holes better. Let us further compare the average number of occupied voxels, the average triangle area, and the number of triangles to evaluate how much the output mesh resolution increases. As shown in Table 2, deformation by the NF increases the average occupancy and the number of triangles and decreases the average triangle area when E-DMC and E-DMC+L are compared with DMC and

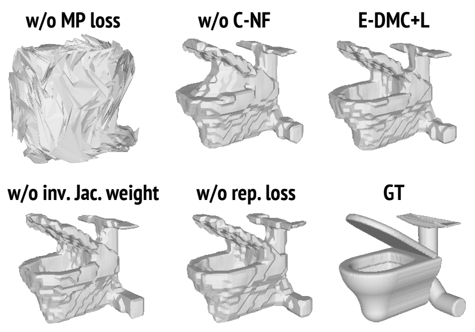

Figure 6: Visual comparison of the output meshes for the ablation study (*toilet 0375*).

	Acc.	Comp.	Overall
E-DMC+L	0.167	0.190	0.178
w/o $\mathcal{L}_{\mathrm{mp}}$	1.128	0.346	0.737
w/o C-NF	0.180	0.207	0.193
w/o iJW	0.172	0.195	0.183
w/o $\mathcal{L}_{\mathrm{rep}}$	0.169	0.187	0.178
w/ classifier	0.160	0.189	0.174
E-DMC	0.454	0.307	0.381
w/o $\mathcal{L}_{\mathrm{rep}}$	0.646	0.252	0.449

Table 3: Quantitative comparison for the ablation study. Although the effect of $\mathcal{L}_{\mathrm{rep}}$ is limited for E-DMC+L, it appropriately works to improve the performance of E-DMC without linear stretching.

DMC+L. These observations also demonstrate the improvement of mesh resolution by the proposed method without increasing the number of voxels.

Finally, we conducted an ablation study to validate the effect of each technique proposed in this paper. In this experiment, we ablated each component from E-DMC+L. The results of this ablation study are shown in Fig. 6 and Table 3. The output meshes in Fig. 6 show that a mesh topology obtained by DMC for the deformed point cloud is completely disordered when the manifold-preserving loss $\mathcal{L}_{\mathrm{mp}}$ is ablated. We also found that shape deformation without using a global shape feature ignores the concavity of the toilet. The inflation of the concave part of the toilet is insufficient, and its lid and body are inappropriately connected. Thus, the global shape should be considered to obtain a better deformation. This fact is supported by the result that the performance of E-DMC+L is further improved when the shape feature encoder is trained together with a shape classifier.

In contrast, the output of E-DMC+L is approximately identical to those for the cases where either of the inverse Jacobian weight and point repulsion loss $\mathcal{L}_{\mathrm{rep}}$ is ablated. The limited effect of these components is because DMC can obtain a somewhat precise surface geometry by only linearly stretching the input shape. Particularly, the effect of the point repulsion is almost negligible, as shown in Table 3, whereas the inverse Jacobian weight can improve the surface reproduction performance quantitatively. To further analyse the effect of the point repulsion, we ablated it from E-DMC without linear stretching. The results at the bottom of Table 3 have shown that the point repulsion improves the performance of E-DMC when the input shape is located locally in the voxel grid. Thus, all techniques presented in this paper help DMC to extract higher-resolution surface meshes.

5 Conclusion

This study presented an extension of DMC [23] using learning-based point cloud inflation. We employed Real NVP [8] to define the deformation and trained the whole network by constraining the deformation to preserve the underlying manifold structure of the point cloud. We evaluated our extension using the ModelNet40 and SHREC datasets with various shape categories, thus demonstrating the enhancement of surface extraction using DMC.

References

[1] Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Gaël Guennebaud, Joshua A. Levine, Andrei Sharf, and Claudio T. Silva. A survey of surface reconstruction from point clouds. *Computer Graphics Forum*, 36(1):301–329, 2017. doi:10.1111/cgf.12802.

[2] Hamish Carr, Thomas Theußl, and Torsten Möller. Isosurfaces on Optimal Regular Samples. In G.-P. Bonneau, S. Hahmann, and C. D. Hansen, editors, *Eurographics / IEEE VGTC Symposium on Visualization*, 2003. doi:10.2312/VisSym/VisSym03/039-048.

[3] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. doi:10.1109/cvpr.2019.00609.

[4] Zhiqin Chen and Hao Zhang. Neural marching cubes. *arXiv preprint arXiv:2106.11272*, 2021. URL https://arxiv.org/abs/2106.11272.

[5] Etienne Corman, Maks Ovsjanikov, and Antonin Chambolle. Supervised descriptor learning for non-rigid shape matching. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 283–298. Springer, 2014. doi:10.1007/978-3-319-16220-1_20.

[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*, 18(6):914–924, 2012. doi:10.1109/tvcg.2012.34.

[7] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, page 303–312, 1996. doi:10.1145/237170.237269.

[8] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016. URL https://arxiv.org/abs/1605.08803.

[9] Jeff Goldsmith and Allan S Jacobson. Marching cubes in cylindrical and spherical coordinates. *Journal of Graphics Tools*, 1(1):21–31, 1996.

[10] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 216–224, 2018. doi:10.1109/cvpr.2018.00030.

[11] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 2722–2730, 2019. URL http://proceedings.mlr.press/v97/ho19a.html.

[12] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, page 71–78, 1992. doi:10.1145/133994.134011.

[13] Jingwei Huang, Hao Su, and Leonidas Guibas. Robust watertight manifold surface generation method for shapenet models. *arXiv preprint arXiv:1802.01698*, 2018. URL `https://arxiv.org/abs/1802.01698`.

[14] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 406–413, 2014. doi:10.1109/CVPR.2014.59.

[15] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. In *Proceedings of ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 339–346, 2002. doi:10.1145/566570.566586.

[16] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics*, 32(3), 2013. doi:10.1145/2487228.2487237.

[17] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of Eurographics Symposium on Geometry Processing (SGP)*, volume 7, 2006. doi:10.2312/SGP/SGP06/061-070.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. URL `https://arxiv.org/abs/1412.6980`.

[19] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. URL `https://proceedings.neurips.cc/paper/2018/file/d139db6a236200b21cc7f752979132d0-Paper.pdf`.

[20] Leif P. Kobbelt, Mario Botsch, Ulrich Schwanecke, and Hans-Peter Seidel. Feature sensitive surface extraction from volume data. In *Proceedings of ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, page 57–66, 2001. doi:10.1145/383259.383265.

[21] Vladislav Kraevoy and Alla Sheffer. Template-Based Mesh Completion. In Mathieu Desbrun and Helmut Pottmann, editors, *Proceedings of Eurographics Symposium on Geometry Processing (SGP)*, 2005. doi:10.2312/SGP/SGP05/013-022.

[22] Guo Li, Ligang Liu, Hanlin Zheng, and Niloy J Mitra. Analysis, reconstruction and manipulation using arterial snakes. *ACM Transactions on Graphics*, 29(6):152, 2010. doi:10.1145/1882262.1866178.

[23] Yiyi Liao, Simon Donné, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2916–2925, 2018. doi:10.1109/CVPR.2018.00308.

[24] Or Litany, Tal Remez, Emanuele Rodola, Alex Bronstein, and Michael Bronstein. Deep functional maps: Structured prediction for dense shape correspondence. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 5659–5667, 2017. doi:10.1109/iccv.2017.603.

[25] Roee Litman and Alexander M Bronstein. Learning spectral descriptors for deformable shape correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(1):171–180, 2014. doi:10.1109/TPAMI.2013.148.

[26] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. volume 21, page 163–169, 1987. doi:10.1145/37402.37422.

[27] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4460–4470, 2019. doi:10.1109/cvpr.2019.00459.

[28] Timothy S. Newman and Hong Yi. A survey of the marching cubes algorithm. *Computers & Graphics*, 30(5):854–879, 2006. doi:10.1016/j.cag.2006.07.021.

[29] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. *ACM Transactions on Graphics*, 22(3): 463–470, 2003. doi:10.1145/882262.882293.

[30] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019. doi:10.1109/cvpr.2019.00025.

[31] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. URL `https://proceedings.neurips.cc/paper/2017/file/d8bf84be3800d12f74d8b05e9b89836f-Paper.pdf`.

[32] Edoardo Remelli, Artem Lukoianov, Stephan Richter, Benoit Guillard, Timur Bagautdinov, Pierre Baque, and Pascal Fua. Meshsdf: Differentiable iso-surface extraction. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22468–22478. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/fe40fb944ee700392ed51bfe84dd4e3d-Paper.pdf`.

[33] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000. doi:10.1126/science.290.5500.2323.

[34] Manolis Savva, Fisher Yu, Hao Su, M Aono, B Chen, D Cohen-Or, W Deng, Hang Su, Song Bai, Xiang Bai, et al. Shrec16 track: largescale 3d shape retrieval from shapenet core55. In *Proceedings of the Eurographics Workshop on 3D Object Retrieval*, volume 10, 2016.

[35] Scott Schaefer and Joe Warren. Dual marching cubes: Primal contouring of dual grids. In *Proceedings of Pacific Conference on Computer Graphics and Applications*, pages 70–76, 2004. doi:10.1109/PCCGA.2004.1348336.

[36] Andrei Sharf, Thomas Lewiner, Ariel Shamir, Leif Kobbelt, and Daniel Cohen–Or. Competing fronts for coarse–to–fine surface reconstruction. *Computer Graphics Forum*, 25(3):389–398, 2006. doi:10.1111/j.1467-8659.2006.00958.x.

[37] Raj Shekhar, Elias Fayyad, Roni Yagel, and J. Fredrick Cornhill. Octree-based decimation of marching cubes surfaces. In *Proceedings of IEEE Visualization*, page 335–342, 1996. doi:10.1109/VISUAL.1996.568127.

[38] G.M. Treece, R.W. Prager, and A.H. Gee. Regularised marching tetrahedra: improved iso-surface extraction. *Computers & Graphics*, 23(4):583–598, 1999. doi:10.1016/S0097-8493(99)00076-X.

[39] Oliver van Kaick, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. A survey on shape correspondence. *Computer Graphics Forum*, 30(6):1681–1707, 2011. doi:10.1111/j.1467-8659.2011.01884.x.

[40] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis. *ACM Transactions on Graphics*, 36(4), 2017. doi:10.1145/3072959.3073608.

[41] Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong. Adaptive O-CNN: A Patch-based Deep Representation of 3D Shapes. *ACM Transactions on Graphics*, 37(6), 2018. doi:10.1145/3272127.3275050.

[42] Yiqun Wang, Jing Ren, Dong-Ming Yan, Jianwei Guo, Xiaopeng Zhang, and Peter Wonka. Mgcn: descriptor learning using multiscale gcns. *ACM Transactions on Graphics*, 39(4):122–1, 2020. doi:10.1145/3386569.3392443.

[43] Udaranga Wickramasinghe, Edoardo Remelli, Graham Knott, and Pascal Fua. Voxel2mesh: 3d mesh model generation from volumetric data. In *Proceedings of Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 299–308, 2020. doi:10.1007/978-3-030-59719-1_30.

[44] Udaranga Wickramasinghe, Pascal Fua, and Graham Knott. Deep active surface models. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11652–11661, 2021.

[45] Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo. Deep geometric prior for surface reconstruction. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10122–10131, 2019. doi:10.1109/CVPR.2019.01037.

[46] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. doi:10.1109/cvpr.2015.7298801.

[47] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. doi:10.1109/CVPR.2018.00295.