Isomorphic Mesh Generation From Point Clouds With Multilayer Perceptrons

Shoko Miyauchi, Member, IEEE, Ken’ichi Morooka, Member, IEEE, and Ryo Kurazume, Senior Member, IEEE

Abstract—A novel neural network called the isomorphic mesh generator (iMG) is proposed to generate isomorphic meshes from point clouds containing noise and missing parts. Isomorphic meshes of arbitrary objects exhibit a unified mesh structure, despite objects belonging to different classes. This unified representation enables various modern deep neural networks (DNNs) to easily handle surface models without requiring additional pre-processing. Additionally, the unified mesh structure of isomorphic meshes enables the application of the same process to all isomorphic meshes, unlike general mesh models, where processes need to be tailored depending on their mesh structures. Therefore, the use of isomorphic meshes can ensure efficient memory usage and reduce calculation time. Apart from the point cloud of the target object used as input for the iMG, point clouds and mesh models need not be prepared in advance as training data because the iMG is a data-free method. Furthermore, the iMG outputs an isomorphic mesh obtained by mapping a reference mesh to a given input point cloud. To stably estimate the mapping function, a step-by-step mapping strategy is introduced. This strategy enables flexible deformation while simultaneously maintaining the structure of the reference mesh. Simulations and experiments conducted using a mobile phone have confirmed that the iMG reliably generates isomorphic meshes of given objects, even when the input point cloud includes noise and missing parts.

Index Terms—Data-free, isomorphic mesh generation, point clouds with noise and missing parts.

I. INTRODUCTION

M

ODERN mobile devices such as iPhones are equipped with depth sensors for capturing three-dimensional (3D) point clouds of objects and natural scenes. Such mobile devices with depth sensors can handle 3D object models generated from a 3D point cloud. Furthermore, 3D object models facilitate the application of deep neural networks (DNNs) to classify, segment, and reconstruct 3D shapes and surfaces [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25].

The input data of a DNN is generally represented by a fixed-size vector, array, or graph, for fixing the architecture of the input layer of the DNN. Therefore, when applying DNNs to 3D object models, a vector, array, or graph of a specific dimension should be used to describe the models. Furthermore, the vector, array, or graph elements of the 3D object model should have a well-defined permutation for comparison with other 3D object models.

Notably, the number of points in each model varies considerably, even among object models of the same class. PointNet-based methods [26], [27], [28], [29], [30], capable of handling such point clouds directly, ensure permutation invariance. A point cloud is a set of points sampled from the surface of an object but without information regarding the connectivity between those points. If the relationships between points, such as the geodesic distance, are utilized during DNN training, shape features of 3D object models can be extracted, considering their surface shapes. These features can enhance the accuracy of DNN-based applications using 3D objects. A method to enable DNNs to handle 3D object models while considering such relationships involves converting these point clouds into mesh models in advance. However, when the structures of surface models differ, DNNs cannot directly process them as inputs.

A solution for handling 3D object models using DNNs while considering the connecting relationship is to convert the point cloud into a surface mesh with a unified structure called an “isomorphic mesh” (Fig. 1). This phenomenon indicates that the surface meshes of all objects have the same number of vertices and mesh structures. When using isomorphic meshes, all meshes can be represented by a specific-dimensional vector. This unified representation enables modern DNNs with various structures to handle surface models. Moreover, in the case of general
mesh models, to apply one process, we need to prepare multiple systems depending on their corresponding mesh structures. Conversely, the unified mesh structure of isomorphic meshes enables one system to be applied to all isomorphic meshes. Therefore, compared with the use of general mesh models, the use of isomorphic meshes increases memory usage efficiency and reduces calculation time.

In this paper, a semantic correspondence means that common parts of different isomorphic meshes in the same class are represented by the same vertex ID numbers while satisfying a one-to-one vertex correspondence among the meshes. When isomorphic meshes satisfying the semantic correspondence are given, it is possible to perform fast and reliable analyses which need to consider the commonality and variety of shapes among target objects; examples include real-time animations [31], deformation transfers [32], and the easy generation of statistical shape models [33], [34].

Various methods have been developed to generate surface meshes from point clouds [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46]. However, few mesh generation methods have considered the generation of an isomorphic mesh with a fixed number of vertices. For example, Fig. 2 reveals two mesh models constructed by applying Poisson surface reconstruction to point clouds, each of which consists of 2,500 points. As shown in Fig. 2, even if two objects belonging to the same class have the same number of input points, the generated meshes consist of different numbers of vertices (sofa 1: 3,983 vertices, sofa 2: 4,423 vertices). Moreover, point clouds obtained by sensors can contain noise and missing data. Therefore, the method for generating isomorphic meshes should recover the object shape while filling in missing parts and removing noise.

To solve these problems, this study proposes a DNN-based method for generating isomorphic meshes from object point clouds. The proposed method, called isomorphic mesh generator (iMG), is based on a deep geometric prior (DGP) [47] in which the point cloud is divided into multiple local regions to output a dense and smooth point cloud for each local region. DGP is a data-free method that requires only the given input data for its construction. Moreover, it is robust to noise and missing data. Accordingly, DGP can be applied to point clouds with noise and missing parts. However, because the purpose of the DGP is to increase the density of a point cloud, DGP cannot generate isomorphic meshes directly from point clouds. In contrast to DGP, the proposed iMG generates an isomorphic mesh from the point cloud of an object. When the point cloud of an object and a reference mesh are provided, the iMG outputs a deformed reference mesh that recovers the shape of point clouds. The input point clouds of different objects can be described using the mesh structure of the reference mesh.

During model generation, iMG deforms the reference mesh to fit the point-cloud distribution of the target object. When the shape of the target object is considerably different from that of the reference mesh, the reference mesh drastically deforms to fit the shape of the target object. Such drastic deformation results in the generation of an incomplete mesh that includes many self-intersections. A solution for self-intersection during deformation is to prepare various reference meshes based on the shape of the point cloud [48]. However, this solution requires additional preprocessing to determine a reference mesh with a suitable shape for the input point cloud. Moreover, if the reference meshes have distinct structures, then these meshes should be converted into a unified mesh structure to obtain isomorphic meshes.

To solve this problem, three-step mapping that consists of global mapping, coarse local mapping, and fine local mapping is performed in iMG (see Fig. 3). First, given the point cloud of a target object, global mapping recovers the approximate shape of the target object by deforming the reference mesh. Second, after dividing the point cloud into several local regions, coarse local mapping is performed to independently recover the approximate shape of each local region. Third, fine local mapping is performed to represent the detailed shape of the cloud and improve shape-recovery accuracy. The shape-recovery accuracy indicates the degree of shape matching between the target object and reference mesh obtained by mapping. Three-step mapping achieves fine and stable shape recovery of the target object when using a common reference mesh whose shape differs from that of the target object. In this study, we aim to generate the isomorphic meshes of genus zero.

II. RELATED WORK

Studies on reconstructing surface meshes from point clouds can be classified into two categories based on the use of neural networks: classical methods that do not use neural networks and neural-network-based methods.

A. Classical Methods Without Using Neural Networks

A simple method for reconstructing surface meshes includes a Voronoi-based algorithm [49] and a ball-pivoting algorithm [50] that generates a mesh by connecting two points with an edge. However, the accuracy of the mesh generated by such simple methods is sensitive to noise and missing parts because of the direct use of the given point clouds.

Another mesh-generation method approximates the local surface region of a point cloud using parametric functions [35], [51], [52], [53], [54]. Although mesh-generation methods that use parametric functions are robust against noise and missing data, the generation of isomorphic meshes is not guaranteed each time. Therefore, an additional remeshing process should be applied to the mesh models represented by parametric functions to obtain an isomorphic mesh.

Deformable surface meshes [55], [56] are used to generate isomorphic meshes by deforming a reference mesh to fit a point cloud. However, the shape-recovery accuracy of deformable
surface meshes tends to be sensitive to the initial shape of the reference mesh as well as the noise and missing points in the target point cloud.

B. Neural-Network-Based Methods

Studies have been conducted on surface reconstruction from point clouds using neural networks. These studies are classified into two types based on the necessity for additional data, excluding the input point cloud. Data-driven methods [25], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66] require additional data, whereas data-free methods [47], [48], [65] use only input data and no additional data.

Among data-driven methods, a linear-blend skinning (LBS) autoencoder [58] generates a surface mesh by fitting a reference mesh to a target point cloud with or without missing parts. However, the LBS autoencoder assumes that the initial shape of the reference mesh is similar to that of the target point cloud. Therefore, for each object class, the LBS autoencoder requires an initial reference mesh with a representative or average class shape.

AtlasNet [59] outputs a collection of local surfaces estimated from an input point cloud. To obtain the local surfaces, the point cloud was divided into multiple local regions. For each region, a multilayer perceptron (MLP) outputs a 3D point on a local surface mapped from a point on the 2D plane. Moreover, AtlasNet generates a surface mesh by training a map from a unit spherical surface mesh to the point cloud without dividing the point cloud. This phenomenon indicates that AtlasNet generates an isomorphic mesh. However, AtlasNet does not consider the effects of noise or missing data.

Voxel2mesh [67] generates a mesh model from an input voxel model by deforming the reference mesh described as a unit spherical surface mesh. However, to improve the shape reconstruction accuracy of the output mesh, a triangular patch subdivision process was introduced in Voxel2mesh for the reference mesh. Because of the subdivision process, the mesh structure of the generated model changes according to each input data. Therefore, Voxel2mesh has no guarantee of generating isomorphic meshes represented by a common mesh structure.

Multiple data-driven methods represent the shape of a target object with implicit functions using neural networks, including IM-NET [25], occupancy networks [60], [68], convolutional occupancy networks [61], Neural-Pull [63], POCO [66], and DeepSDF [62], [64], [69], [70]. In the first four networks, the points are sampled from the 3D bounding volume covering the input point cloud. Using the sampled points, the network generates implicit functions that binary classify each sampled point as inside or outside the target object. By contrast, the DeepSDF estimates the implicit function using signed distance fields, which represent the signed distance from the surface of the target object. However, because these networks use implicit functions to output a voxel model, additional remeshing processes are required to convert the voxel model into an isomorphic mesh. There are many other methods [43], [44], [46], [65], [71] for generating mesh models using neural networks. However, these methods do not allow to explicitly generate a mesh model with a specific mesh structure, making it difficult to generate isomorphic meshes directly.

In contrast to data-driven methods, data-free methods, including DGP [47] and Point2mesh [48], enable the recovery of various shapes with no additional data except input data. The DGP generates a dense and smooth point cloud from the input point cloud with or without noise. However, post-processing is required for generating an isomorphic mesh from the point cloud obtained by DGP.

Similar to the deformable model, point2mesh generates a surface mesh from the target point cloud by deforming a reference mesh for the target point cloud. For this deformation, point2mesh learns typical shapes frequently shown in the target point cloud. The learning process renders point2mesh robust against random noise and defects. To improve the shape-recovery accuracy of the output mesh, a triangular patch subdivision process is introduced in point2mesh. However, the subdivision process results in a difference in the structure of the generated mesh according to each input data. This phenomenon indicates that point2mesh does not guarantee the generation of isomorphic meshes represented by a common mesh structure.

In addition, Shape As Points [65] achieves a fast mesh generation from a point cloud of an arbitrary topology in both data-free and data-driven cases. However, since the obtained mesh structure is changed depending on the object shape, Shape As Points has no guarantee the generation of isomorphic meshes.

III. iMG

In iMG, the target object has a genus-zero closed surface. The point cloud of the target object, which may contain noise and
missing parts, is the input to iMG. In iMG, a spherical surface mesh generated by subdividing an icosahedron is used as the reference mesh. An isomorphic mesh of the target object is generated by deforming the reference mesh to the shape of the target object.

The iMG architecture is based on a DGP composed of MLPs. The MLPs are optimised to map the 3D points on the reference mesh to the 3D points on the surface of the target object. The parameters of the optimised MLPs are regarded as those of the mapping function. Generally, a complex mapping function with many parameters exhibits a high degree of shape representation. Therefore, the reference mesh deformed by the complex mapping function tends to fit the input point cloud of the target object with complex shapes. To achieve this, MLPs have a complex architecture. However, training such complex MLPs sometimes fails because determining all optimal parameters in the mapping function is difficult.

Considering these factors, in DGP, the input point cloud is divided uniformly into multiple local regions. Then, the object shape is recovered using multiple simple mapping functions with fewer parameters for the local regions. This approach is also used in iMG. The reference mesh is divided into local meshes. Each local mesh is deformed to reconstruct the shape of its corresponding local point cloud. When using the aforementioned approach to obtain an isomorphic mesh with acceptable accuracy, a suitable correspondence should be determined between local meshes and local clouds. However, because the initial shape of the reference mesh differs from that of the point cloud, determining a suitable correspondence before deformation is difficult. Therefore, iMG introduces a three-step strategy for deforming the reference mesh.

First, an approximate global deformation of the reference mesh is performed to fit the input point cloud. Second, the deformed reference mesh and input point cloud are divided into a fixed number of local meshes and local clouds, respectively. Each local mesh is deformed closer to the corresponding local cloud. Moreover, the reference model is deformed to improve the shape-recovery accuracy in the third step, rather than in the second step. Practically, the deformed reference mesh in the second step and the input point cloud are divided again into local meshes and local clouds according to the local shape of the input point cloud. Each local mesh is deformed to fit its corresponding local cloud. By gradually deforming the reference mesh while changing its division method, we generate an isomorphic mesh with high shape-recovery accuracy while preventing a large self-intersection of the model.

Given initial reference mesh \( \mathbf{R}^{(0)} \) and input point cloud \( \Pi \), the algorithm of our iMG is described as follows (Fig. 3):

Step 1 \textit{Global mapping}: \( \mathbf{R}^{(0)} \) is deformed using an MLP to approximately fit \( \Pi \); the deformation result is called the first reference mesh \( \mathbf{R}^{(1)} \).

Step 2 \textit{Coarse local mapping}: \( \mathbf{R}^{(1)} \) and \( \Pi \) are divided into 32 local meshes and their corresponding 32 local clouds. For each local mesh, an MLP is trained to fit the local mesh to its corresponding local cloud. After deforming all local meshes, the second reference mesh \( \mathbf{R}^{(2)} \) is obtained by integrating 32 deformed local meshes.

Step 3 \textit{Fine local mapping}: \( \mathbf{R}^{(2)} \) and \( \Pi \) are divided into multiple local meshes and their corresponding local clouds based on the distribution of \( \Pi \). For each local mesh, an MLP is trained to fit the local mesh to its corresponding local cloud. The final resulting mesh \( \mathbf{R}^{(3)} \) is obtained by integrating the deformed local meshes.

#### A. Reference Mesh

The initial reference mesh \( \mathbf{R}^{(0)} \) consists of the set \( \mathbf{V}^{(0)} \) of vertices and set \( \mathbf{T} \) of triangular patches as follows:

\[
\mathbf{R}^{(0)} = \{ \mathbf{V}^{(0)} , \mathbf{T} \}.
\]

The vertex in \( \mathbf{V}^{(0)} \) is represented by the vector of its 3D coordinates. The patch in \( \mathbf{T} \) is a set of indices of the three vertices constituting the patch. The initial reference mesh \( \mathbf{R}^{(0)} \) is generated by subdividing an icosahedron with a circumscribed sphere of radius 1 into an arbitrary resolution. Subsequently, \( \mathbf{R}^{(0)} \) is divided into 32 local meshes (Fig. 4) to train the MLPs in coarse local mapping as follows.

A total of 32 vertices, called anchor vertices, are selected from \( \mathbf{V}^{(0)} \). Twelve vertices of the icosahedron in \( \mathbf{V}^{(0)} \) are used as anchor vertices. Moreover, using the 20 triangular elements of the icosahedron, the vertex closest to the line passing through the origin of the icosahedron and the centroid of each triangular element is selected. The selected 20 vertices are also used as anchor vertices.

Second, \( \mathbf{R}^{(0)} \) is divided into 32 local meshes centred at 32 anchor vertices. For each anchor vertex, the vertices and patches of \( \mathbf{R}^{(0)} \) are collected such that the geodesic distance from the anchor on \( \mathbf{R}^{(0)} \) is less than threshold \( \tau_a \). The set of collected vertices and patches is defined as a local mesh, \( \mathbf{C}_i^{(0)} \) \((i = 1, 2, \ldots, 32) \). Because we allow the division of \( \mathbf{R}^{(0)} \) such that an overlapping area occurs between two or three adjacent local meshes, the vertices of \( \mathbf{V}^{(0)} \) in the overlapping area are included in multiple local meshes. Therefore, \( \mathbf{R}^{(0)} = \mathbf{C}_1^{(0)} \bigcup \cdots \bigcup \mathbf{C}_{32}^{(0)} \). The threshold \( \tau_a \) used in the division of \( \mathbf{R}^{(0)} \) is set such that all vertices in \( \mathbf{V}^{(0)} \) are always included in at least one local mesh. Based on preliminary experiments, we set \( \tau_a \) to 0.55.

In the coarse local mapping, the deformation of the entire reference mesh is obtained by deforming local meshes independently and integrating the deformed local meshes into a single mesh with smooth surface. To achieve smooth mesh integration of overlapping areas among adjacent local meshes, we define the weight of the vertex, which indicates the importance of the
vertex for the integration. In the proposed method, the vertex is crucial for integration when the geodesic distance from the vertex in local mesh \( C_i^{(0)} \) to the anchor vertex \( a_i \) of \( C_i^{(0)} \) is short. Thus, the weight \( w_C^i \) of vertex \( v^{(0)} \) in \( C_i^{(0)} \) is calculated by subtracting the geodesic distance between \( a_i \) and \( v^{(0)} \) divided by the threshold \( \tau_i \) from 1. Here, weight \( w_C^i \) is calculated once before starting the iMG algorithm; the calculated weights are used during the algorithm.

B. Global Mapping

Fig. 5 reveals that in global mapping, the initial reference mesh \( R^{(0)} \) is deformed to approximately fit the input point cloud \( \Pi \) by training an MLP. Here, \( \Pi \) is aligned in advance, such that \( \Pi \) is enclosed by \( R^{(0)} \). The MLP in iMG uses the 3D coordinates of the points obtained from \( R^{(0)} \) as its input.

The MLP used in global mapping consists of six layers, namely one input, four intermediate, and one output. These six layers have 3, 128, 256, 512, 512, and 3 nodes, respectively. The set \( S^{(0)} \) of 3D points is obtained by sampling uniformly from \( R^{(0)} \) such that the number of sampled points is the same as that of the points in \( \Pi \). Point sampling was performed using the Fibonacci sphere algorithm. The input for the MLP comprises the 3D coordinates of the sampled points in \( S^{(0)} \). The output of the MLP is the 3D coordinates of the points after deformation.

The MLP is optimised to approximate the mapping function \( f^{(1)} \) from point \( s^{(0)} \in S^{(0)} \) to its corresponding point, \( p \in \Pi \). Using the mapping function \( f^{(1)} \), when \( s^{(0)} \) is provided as the input to the MLP, the output of the MLP is expressed as \( f^{(1)}(s^{(0)}) \). Similar to DGP, the loss function of an MLP is expressed as the mean squared error loss between \( s^{(0)} \) and its corresponding point \( p \):

\[
L^{(1)}(S^{(0)}, \Pi) = \frac{1}{|\Pi|} \sum_{s^{(0)} \in S^{(0)}, p \in \Pi} \| f^{(1)}(s^{(0)}) - p \| ,
\]

where \(|\Pi|\) represents the number of points in the set \( \Pi \). Here, \( \beta_i \) is the scale factor determined for each target object through preliminary experiments. For each sampled point \( s^{(0)} \), the corresponding point \( p \) is determined using the Sinkhorn-regularised distance [72] between \( S^{(0)} \) and \( \Pi \). The correspondence is redefined whenever the parameters of the MLP are updated.

After global mapping, we obtain the first reference mesh \( R^{(1)} \) by changing \( v^{(0)} \in V^{(0)} \) to vertex \( v^{(1)} = f^{(1)}(\beta_i v^{(0)}) \). The set \( v^{(1)} \) is denoted as \( V^{(1)} \).

C. Coarse Local Mapping

Coarse local mapping consists of three steps (Fig. 6). In the first step, the input point cloud \( \Pi \) is divided into 32 local clouds \( X_i \) corresponding to 32 local meshes \( C_i^{(1)} \). Here, \( C_i^{(1)} \) is obtained by replacing the vertices in \( C_i^{(0)} \) with their corresponding vertices in \( R^{(1)} \). In the division of \( \Pi \), for each point \( p \in \Pi \), we find the closest vertex to \( p \) from the first reference mesh \( R^{(1)} \) and regard it as the corresponding vertex of \( p \). The local cloud \( X_i \) is determined by collecting the points whose corresponding vertices are included in \( C_i^{(1)} \).

In the second step, each local mesh \( C_i^{(1)} \) is deformed to fit its corresponding local cloud \( X_i \). Finally, the second reference mesh \( R^{(2)} \) is obtained by integrating 32 deformed local meshes \( C_i^{(2)} \). The details of the second and third steps of coarse local mapping are as follows.

1) Deformation of Each Local Mesh: In local mesh deformation, the MLP is optimised to approximate the mapping function from the local mesh \( C_i^{(1)} \) to the corresponding local cloud \( X_i \) (Fig. 6). Before deformation, the coordinate system of \( C_i^{(1)} \) is transformed from the global coordinate system of the reference mesh to a local coordinate system, the origin of which coincides with the centroid of \( C_i^{(1)} \). Moreover, \( X_i \) is transformed into a local coordinate system by applying the transformation matrix of \( C_i^{(1)} \).

Next, 3D points are uniformly sampled from the surface of \( C_i^{(1)} \) such that the number of sampled points is the same as that of \( X_i \). Sampling is performed using Lloyd’s algorithm. The set \( S_i^{(1)} \) of the sampled points is used as the training dataset for the MLP during coarse local mapping.

The MLP is optimised to approximate mapping function \( f_i^{(2)} \) from the sampled point \( s_i^{(1)} \in S_i^{(1)} \) to its corresponding point in \( X_i \). The architecture of the MLP is the same as that of the MLP used in global mapping. The loss function is expressed using (2). Before training the MLP, Sinkhorn-regularised distances are used to determine the correspondence between \( X_i \) and \( S_i^{(1)} \). This correspondence is used with no updates during training to prevent self-intersection between adjacent local meshes.
As mentioned in Section III-A, an overlapping area exists between adjacent local meshes, and their corresponding adjacent local clouds overlap. Therefore, multiple sampled points correspond to a single point in the overlapping area of the local clouds. The output of the various MLPs corresponding to the point typically has distinct positions. This difference leads to a discontinuity between adjacent local surfaces estimated using MLPs. To fuse adjacent surfaces into one continuous surface, the MLPs should be trained to recover the shape of the local cloud while minimising the difference. MLP training is a two-step process based on DGP.

First, each MLP is trained using the local cloud \( X_i \). After completion of the training, the outputs of the trained MLPs are calculated using the sampled points in set \( S_i^{(1)} \). The outputs of the MLPs are used as the new positional coordinates of the points in \( \Pi \) to retrain MLPs.

In the overlapping area, multiple output points of the MLPs are integrated into a new point by calculating the weighted average of the positional coordinates. Here, a point in \( X_i \) corresponds to a vertex in local mesh \( C_i^{(1)} \). Hence, we consider that the output point of the MLPs that use the sampled point paired with the point also corresponds to the vertex. Owing to this correspondence between the output point and the vertex, the weight \( w_i^f \) of its corresponding vertex in \( C_i^{(1)} \) is used as the weight of the output point.

Using the new positional coordinates of the points in \( \Pi \), the MLPs are additionally trained using the loss function (2). After retraining, the deformed local mesh \( C_i^{(2)} \) is obtained by replacing the 3D coordinates of vertex \( v_i^{(1)} \) in \( C_i^{(1)} \) with the mapped 3D coordinates \( f_i^{(2)}(\beta_2 v_i^{(1)}) \). Here, \( \beta_2 \) is the scale factor used in (2) instead of \( \beta_1 \). The value of \( \beta_2 \) is the same for all local meshes.

2) Integration of the Deformed Local Mesh: The deformation result of \( R^{(1)} \), called the second reference mesh \( R^{(2)} \), is generated by integrating the deformed local meshes \( C_i^{(2)} \). In the overlapping area, the output points are fused into one vertex \( v_i^{(2)} \) using the same method described in Section III-C1. After fusion, the second reference mesh \( R^{(2)} \) is obtained by changing \( v_i^{(1)} \in V^{(1)} \) into vertex \( v_i^{(2)} \). The set \( V^{(2)} \) is denoted as \( V^{(2)} \).

D. Fine Local Mapping

Fine local mapping consists of three steps (Fig. 7). The first step is to divide the second reference mesh \( R^{(2)} \) and \( \Pi \) into \( N_F \) local meshes \( F_j^{(2)} \) and \( N_F \) local clouds \( \Phi_j \). To consider the shape complexity of the input point cloud \( \Pi \), this division is performed using the curve fitting results for \( \Pi \). Consequently, the shape of each local cloud becomes simple. In the second step, using an MLP, each local mesh \( F_j^{(2)} \) is deformed to fit its corresponding local cloud \( \Phi_j \). Finally, the reference mesh \( R^{(3)} \) is obtained by integrating all deformed local meshes. Similar to the method described in Section III-C2, the integration fuses points in the overlapping area between the deformed local meshes in the vertex. Point fusion is performed by calculating the weighted average of the points. The weights of the points are weight \( w_i^f \) as defined in Section III-D1. The details of the division and deformation steps in the fine local mapping are as follows.

1) Division of the Second Reference Mesh and Input Point Cloud: The division of the second reference mesh \( R^{(2)} \) and input point cloud \( \Pi \) primarily comprises two processes. In the first process, the input point cloud \( \Pi \) is divided using a set of multiple blocks that cover \( \Pi \) (top left figure in Fig. 8). The point cloud in each block is referred to as a temporary local cloud. Furthermore, \( \Pi \) is divided such that the shape of the temporary local cloud is represented by a simple polynomial surface function. The use of temporary local clouds enables the estimation of the mapping function of the temporary local cloud using the MLP with fewer parameters, as described in Section III. Subsequently, for each temporary local cloud, we collect vertices in \( R^{(2)} \) that have the closest euclidean distance to each point in the temporary local cloud (middle-left figure in Fig. 8). The set of collected vertices and patches comprising the collected vertices is called a temporary local mesh.

The local mesh, which is the input to the MLP, is assumed to be a continuous area with no holes. However, depending on the
shape of the second reference mesh, $\mathbf{R}^{(2)}$, the temporary local mesh is typically discontinuous and/or has holes. Therefore, in the second process, using the initial reference mesh $\mathbf{R}^{(0)}$ and temporary local meshes (bottom-left and right figures in Fig. 8), $\mathbf{R}^{(2)}$ is redivided into $N_F$ continuous areas called local meshes (middle-right figure in Fig. 8). Moreover, for each of the local meshes obtained in $\mathbf{R}^{(2)}$, the corresponding local cloud is redefined as the set of points in $\Pi$ that are closest to the vertices in the local mesh (top-right figure in Fig. 8). The details of each process are as follows.

a) Generation of a temporary local mesh: To divide the input point cloud $\Pi$ into temporary local points, we use a bounding box containing $\Pi$, which is a set of blocks of equal size. For each block, the fitting error $E$ of the points in the block is defined as the maximum distance between each point $p$ and a point on the polynomial surface. In the proposed method, a fifth-order polynomial is used as a function of the polynomial surface. When $E$ exceeds the threshold $\tau_c$, the block is divided evenly into subblocks, adjusting the number of subblocks until $E$ for all subblocks falls below $\tau_c$. After the division process, all subblocks are regarded as blocks.

The computational time required for the fine local mapping increases depending on the number of blocks. To reduce the computational time while maintaining shape-recovery accuracy, adjacent blocks are repeatedly merged into a new block until no pairs of adjacent blocks with $E$ less than $\tau_c$ exist.

Using the merged blocks, the input point cloud $\Pi$ is divided into $N_F$-temporary point clouds. For each block, the corresponding temporary point cloud is obtained by collecting points within the block. Moreover, for each temporary point cloud, the corresponding temporary local mesh is obtained by collecting the vertices in $\mathbf{R}^{(2)}$ that are closest to the points in the temporary point cloud.

b) Generation of a local mesh and local point cloud: To simplify the re-division process of generating local meshes with no holes, we perform the process on the initial reference mesh $\mathbf{R}^{(0)}$ represented by the simplest spherical surface. First, the positional coordinates of the vertices in the temporary local meshes are replaced by those of their corresponding vertices in $\mathbf{R}^{(0)}$ (bottom-left figure in Fig. 8). Next, the smallest elliptical cylinder that includes all vertices in the temporary local mesh on the spherical surface is calculated. The portion of the initial reference mesh $\mathbf{R}^{(0)}$ within the elliptical cylinder is defined as a local mesh. By applying the elliptical cylinder calculation to all temporary local meshes, $\mathbf{R}^{(0)}$ is divided into $N_F$-local meshes $\mathbf{F}_j^{(0)}$ (bottom-right figure in Fig. 8). Finally, we obtain the local mesh $\mathbf{F}_j^{(2)}$ by replacing the coordinates of vertices in $\mathbf{F}_j^{(0)}$ with those of their corresponding vertices in $\mathbf{R}^{(2)}$ (middle-right figure in Fig. 8).

Moreover, for each point $p$ in the input cloud $\Pi$, the closest vertex to $p$ from $\mathbf{R}^{(2)}$ is used as the corresponding vertex of $p$. Using $\mathbf{F}_j^{(2)}$, the local cloud $\Phi_j$ is determined by collecting the corresponding points of the vertices in $\mathbf{F}_j^{(2)}$ (top-right figure in Fig. 8).

Here, an overlapping area exists between adjacent elliptic cylinders. Therefore, vertices in the overlapping area are included in multiple local meshes. Similar to coarse local mapping, to achieve smooth mesh integration, we define a weight $w_j^{(s)}$ of the vertex in $\mathbf{F}_j^{(2)}$. For each vertex in $\mathbf{F}_j^{(2)}$, we define $w_j^{(s)}$ as the geodesic distance on $\mathbf{R}^{(0)}$ from the centroid of $\mathbf{F}_j^{(0)}$ to the vertex in $\mathbf{F}_j^{(2)}$. To ensure that the weights around the boundary of $\mathbf{F}_j^{(0)}$ are zero, the distance is normalised using the lengths of the long and short sides of the approximated ellipse for $\mathbf{F}_j^{(0)}$.

2) Deformation of the Local Mesh: In fine local mapping, MLP is optimised to approximate the mapping function from local mesh $\mathbf{F}_j^{(2)}$ to the corresponding local cloud $\Phi_j$. Before deformation, the coordinate systems of $\mathbf{F}_j^{(2)}$ and $\Phi_j$ are converted to a local coordinate system obtained using the method described in Section III-C1.

The training dataset for MLP consists of two sets of 3D coordinates. Using the same method as in coarse local mapping (Step 2 of the iMG algorithm), the 3D points are sampled uniformly from the surface of $\mathbf{F}_j^{(2)}$ such that the number of sampled points is the same as that of $\Phi_j$. We set $\mathbf{S}_j^{(2)}$ for the sampled points as the first set. Subsequently, the local cloud $\Phi_j$ is used as the ground truth for $\mathbf{S}_j^{(2)}$.

When the number $|\Phi_j|$ of points in $\Phi_j$ is large, the MLP can be efficiently optimised using only $\mathbf{S}_j^{(2)}$. However, because the rugged area cannot be represented by a single polynomial surface, the area is divided into multiple local clouds with a small number of points. This indicates that many local clouds with a small number of points are generated around the rugged area. Therefore, using only $\mathbf{S}_j^{(2)}$ is sometimes insufficient for training the MLP.

To stably train MLP using a small number of points in $\Phi_j$, set $\mathbf{V}_j^{(2)}$ of vertices in $\mathbf{F}_j^{(2)}$ is added to the training dataset as the second set. Moreover, set $\mathbf{S}_j^{(2)}$ is generated using the polynomial surface obtained in Section III-D1 for $\Phi_j$, and used as the ground truth of $\mathbf{V}_j^{(2)}$. The 3D points in $\mathbf{S}_j^{(2)}$ are randomly sampled from the polynomial surface such that the number of sampled points is the same as the number of vertices in $\mathbf{V}_j^{(2)}$.

The MLP is optimised to approximate mapping function $f_j^{(3)}$ from the sampled point $s_j^{(2)} \in \mathbf{S}_j^{(2)}$ to its corresponding point $p \in \Phi_j$ and from $v_j^{(2)} \in \mathbf{V}_j^{(2)}$ to $s_j^{(2)} \in \mathbf{S}_j^{(2)}$. The MLP architecture is the same as that of the MLP used in global mapping. The loss function is expressed as follows:

$$L^{(3)}(\mathbf{S}_j^{(2)}, \Phi_j, \mathbf{V}_j^{(2)}, \mathbf{S}_j^{(2)})$$

$$= \frac{1}{|\Phi_j|} \sum_{s_j^{(2)} \in \mathbf{S}_j^{(2)}, p \in \Phi_j} ||f_j^{(3)}(\beta_j s_j^{(2)}) - p||$$

$$+ \frac{\alpha}{|\mathbf{V}_j^{(2)}|} \sum_{v_j^{(2)} \in \mathbf{V}_j^{(2)}, s_j^{(2)} \in \mathbf{S}_j^{(2)}} ||f_j^{(3)}(\beta_j v_j^{(2)}) - s_j^{(2)}||, \quad (3)$$
where $\beta_3$ is the scale factor determined for each target object through preliminary experiments. The value of $\beta_3$ is the same for all local meshes. The first term of (3) represents an input point-cloud-based loss that evaluates the mapping accuracy from the sampled points on the reference mesh $R^{(2)}$ to the input point cloud. The second term of (3) represents a reference mesh-based loss that evaluates the mapping accuracy from the reference mesh to the sampled points $s^p$ on a polynomial surface. Parameter $\alpha$ is a weight coefficient that determines whether the input point-cloud-based loss or reference mesh-based loss is emphasised. Based on preliminary experiments, we set $\alpha = 0.5$.

Before training the MLP, the Sinkhorn-regularised distance is used to determine the correspondence between $\Phi_j$ and $S^{(2)}_j$ and between $S^{(2)}_j$ and $V^F_j$. Similar to coarse local mapping, correspondence is used with no updates during training.

MLP is trained and retrained in the same manner as in coarse local mapping (Section III-C1). In the retraining of MLPs, the weight $w^F_j$ is used for calculating the new positional coordinates of the points in $\Pi$. Moreover, in the retraining, we set $\alpha = 0$ in (3) to increase the effect of mapping from the sampled points on the reference mesh to the input point cloud. After the retraining, a deformed local mesh $F^{(3)}_j$ is obtained by replacing the 3D coordinates of vertex $v^{(2)}$ in $F^{(2)}_j$ with their mapped 3D coordinates $f^{(3)}(\beta_3 v^{(2)})$.

IV. EXPERIMENT

To confirm the effectiveness of the proposed iMG, we conducted simulations and experiments to generate isomorphic meshes using simulation and measurement data. Two types of simulation data, namely point-cloud data with and without noise, were used. The measurement data were acquired using the TrueDepth camera of an iPhone 11 Pro (Apple Inc., USA).

In iMG, even if given the same input point cloud, the shape of the generated model differs for each optimisation time. Therefore, for evaluation, we generated isomorphic meshes from each input point cloud five times. The shape-recovery accuracy of iMG was evaluated using the average and standard deviation of the shape-recovery accuracies of the five generated models. A GPU (MSI Radeon RX Vega 56 Air Boost 8 G OC) was used in all the experiments.

A. Data

In the simulation, we used three surface meshes: a rabbit, a replica of the face of Max Planck, and the box displayed in Fig. 9(a–c). The vertices of the meshes were used as simulation data without noise (Fig. 9(d–f)), whereas the original surface meshes were used as ground truth meshes of the isomorphic meshes generated from the simulation data. The sizes of the three ground truth meshes were $75.2 \times 35.7 \times 98.0$ [mm], $72.5 \times 59.2 \times 98.9$ [mm], and $100.0 \times 100.0 \times 100.0$ [mm], respectively.

To obtain measurement data, a 3D printer was used to create three real models of three meshes. A TrueDepth camera was used to measure the shapes of the models. During measurements, the mobile phone was fixed on a tripod, whereas the model was placed on a rotating table. The entire surface of the model was scanned by rotating the table. Based on these measurements, the surface meshes of the real models were obtained using an STL Maker (Scandy LLC, New Orleans, LA, USA).

To execute an iMG with only one GPU, the number of vertices of each surface mesh was reduced to approximately 2,500 using a down-sampling process. A surface mesh obtained using the down-sampling process was used as the measurement mesh (Fig. 13(a–c)). The vertices of the measurement mesh were used as the measurement data (Fig. 13(d–f)). The measurement data contained noise. As displayed in Fig. 13(a–c), parts were missing in the measurement data because of the occlusion of the object. For example, no points were observed on the jaw of...
Fig. 11. (a–c) and (g–i) are the simulation data with noise of the rabbit and box models, respectively. From left to right, the noise parameters (δ, σ) are (20, 0.01), (40, 0.03), and (60, 0.05). (d–f) and (j–l) are isomorphic meshes generated by iMG from the simulation data with noise.

Fig. 12. Results of Experiment 2: PM distances of the isomorphic meshes after the fine local mapping of the rabbit, Max Planck and box models with noise in nine patterns (δ, σ).

the rabbit, the nose or neck of the Max Planck model, or the bottom of the mesh that was in contact with the rotating table during measurement. The ground truth meshes (Fig. 9(a)–(c)) for the simulation data were used as ground truth meshes for measurement data.

B. Accuracy Metric

The point-mesh (PM) distance, which is the average of the two distances from the isomorphic mesh \( R^{(3)} \) to its ground-truth mesh \( G \) and vice versa, was used to evaluate the shape-recovery accuracy of the isomorphic mesh.

The distance \( d(s^R, G) \) from a point \( s^R \) that is randomly sampled from \( R^{(3)} \) to \( G \) is defined as follows:

\[
d(s^R, G) = \min_{\gamma \in G} H(s^R, \gamma),
\]

where the function \( H(s^R, \gamma) \) returns the length of the perpendicular line from point \( s^R \) to patch \( \gamma \) in \( G \).

Similarly, distance \( d(s^G, R) \) from point \( s^G \) randomly sampled from \( G \) to \( R^{(3)} \) is defined as follows:

\[
d(s^G, R) = \min_{t \in R^{(3)}} H(s^G, t),
\]

where \( t \) denotes a triangular patch in the isomorphic mesh \( R^{(3)} \).

The PM distance \( D(R^{(3)}, G) \) is defined as follows:

\[
D(R^{(3)}, G) = \frac{1}{2} \left( \sum_{s^R \in S^R} d(s^R, G) + \sum_{s^G \in S^G} d(s^G, R^{(3)}) \right).
\]

In the experiments and simulations, \( S^R \) and \( S^G \) are sets of 30,000 points randomly sampled from isomorphic mesh \( R^{(3)} \) and its ground truth mesh \( G \), respectively. Here, |\( S^R \) denotes the number of points. The shape-recovery accuracy increases with the decrease in the PM distance.
TABLE I
RESULTS OF EXPERIMENT 1: THE SECOND TO FOURTH COLUMNS LIST THE PM DISTANCES OBTAINED FROM GLOBAL MAPPING, COARSE LOCAL MAPPING, AND FINE LOCAL MAPPING, RESPECTIVELY

|                | PM distance [mm] |
|----------------|------------------|
|                | Global | Coarse | Fine  |
| Rabbit         | 0.34 ± 0.01 | 0.10 ± 0.00 | 0.06 ± 0.00 |
| Max Planck     | 0.43 ± 0.02 | 0.19 ± 0.01 | 0.12 ± 0.01 |
| Box            | 0.35 ± 0.02 | 0.08 ± 0.00 | 0.11 ± 0.01 |

C. Experiment 1: Isomorphic Mesh Generation Using Simulation Data Without Noise

In the simulation using three simulated datasets without noise (Fig. 9(d)–(f)), the number of vertices in the reference mesh was 36,002. The hyperparameters $\beta_1$, $\beta_2$, $\beta_3$, and $\tau_c$ of iMG for each input point cloud were determined through preliminary experiments.

The iMG generated isomorphic meshes (Fig. 9(g)–(i)) using the three simulation data (rabbit, Max Planck, and box) as the input point clouds. The colour of each point of the mesh was determined by the direction of the normal at the point. Fig. 10 illustrates the isomorphic meshes after global mapping, coarse local mapping, and fine local mapping. Table I lists the PM distances for the three mappings. The colour map in Fig. 10 illustrates the distance $d(v, G)$ from the vertex $v$ of the isomorphic mesh $R(v)$ to its ground truth mesh $G$ in (4). The colour is close to blue when the value of $d(v, G)$ is close to zero, whereas the colour is close to red when the value is close to 0.5 [mm].

From Fig. 10 and Table I, for the rabbit and Max Planck models, the blue region increased, whereas the PM distance decreased as the deformation progressed from global mapping to fine local mapping. This phenomenon indicated that the distance between the isomorphic mesh and the ground truth mesh decreased. In particular, the shape-recovery accuracy improved after coarse local mapping. By contrast, the accuracy was low for shapes with large curvature changes, such as the tail, eyes, nose, and mouth at the feet of the rabbit, as well as the ears, eyes, and mouth at the Max Planck model. However, fine local mapping improved the shape-recovery accuracy in these regions.

The PM distance after the coarse local mapping of the box was the shortest among the three steps. After fine local mapping, the red region around the edges increased slightly.

D. Experiment 2: Isomorphic Mesh Generation Using Simulation Data With Noise

To confirm the robustness of iMG to noise, isomorphic meshes were generated from three simulation data containing artificial noise. These simulation data were then generated by adding uniformly distributed noise to simulation data without noise.

Noise was controlled by two parameters, namely $\delta$ and $\sigma$. The parameter $\delta$ [%] represents the ratio of the number of points, to which random values were added. The parameter $\sigma$ represents the range $[-\sigma L, \sigma L]$ of the random values, where $L$ is the length of the long side of the bounding box enclosing the target object. By combining $\delta = \{20, 40, 60\}$ and $\sigma = \{0.01, 0.03, 0.05\}$, we generated nine patterns of simulation data with noise. Fig. 11(a)–(c) and (g)–(i) show the simulation data with noise for the rabbit and box models when $(\delta, \sigma) = (20, 0.01), (40, 0.03),$ and $(60, 0.05)$.

Fig. 11(d)–(f) and (j)–(l) display the isomorphic meshes generated from the simulation data, and Fig. 12 details PM distances using final isomorphic meshes. From these figures, the shape-recovery accuracy decreased in inverse proportion to the increase in the ratio of added noise and noise variance. However, all the models recovered the approximate shapes of the target objects.

E. Experiment 3: Isomorphic Mesh Generation Using Measurement Data

In the third experiment, isomorphic meshes were generated from the measured data, as displayed in Fig. 13(d)–(f). Fig. 13(g)–(i) show the generated isomorphic meshes. Table II lists the PM distances for each isomorphic mesh.

Table II shows that the PM distance of the isomorphic mesh was almost the same as or smaller than that of the measurement mesh for all the data. From the results presented in Table II and Fig. 13, iMG enabled the generation of isomorphic meshes from an incomplete point cloud while recovering the shapes of the missing parts and retaining shape-recovery accuracy.

F. Experiment 4: Comparison to Baselines Using Simulation Data With or Without Noise

We used three baselines: Point2mesh [48], AtlasNet [59], and Voxel2mesh [67]. Unlike data-free iMG and Point2mesh, AtlasNet and Voxel2mesh are data-driven methods that generate an isomorphic mesh of a target object using a trained network. Therefore, if a class of objects is included in the training dataset, the shape of the target object can be reconstructed with high accuracy. Otherwise, the shape-recovery accuracy of the object is not guaranteed. Because of these characteristics of data-driven methods, the shape-recovery accuracy of AtlasNet and Voxel2mesh is influenced by whether the class of the object to be modelled is included in the training data. Therefore, the two cases were compared using the input point clouds of the trained and untrained classes.

When using AtlasNet, we used pre-trained weights available on GitHub, as provided in [59]. To match the resolution of the spherical mesh used in AtlasNet, we used a spherical surface mesh with 2,562 vertices as the reference model of iMG. Furthermore, when using Point2mesh and Voxel2mesh to generate isomorphic meshes, we used the spherical surface mesh.
TABLE III
RESULTS OF EXPERIMENT 4: PM DISTANCES [$\times 10^{-2}$ MM] OF ISOMORPHIC MESHES OBTAINED USING IMG, POINT2MESH, ATLASNET, AND Voxel2mesh

| Mesh | Without noise | With noise |
|------|---------------|------------|
| Airplane | $0.25 \pm 0.19$ | $0.23 \pm 0.06$ | $0.33 \pm 0.14$ | $0.72 \pm 0.26$ | $0.54 \pm 0.23$ | $0.35 \pm 0.12$ | $0.83 \pm 0.27$ | $1.06 \pm 0.46$ |
| Bench | $0.38 \pm 0.25$ | $0.46 \pm 0.09$ | $0.58 \pm 0.17$ | $1.54 \pm 0.75$ | $0.97 \pm 0.54$ | $0.66 \pm 0.04$ | $1.28 \pm 0.22$ | $1.92 \pm 1.07$ |
| Cabinet | $0.24 \pm 0.20$ | $0.21 \pm 0.06$ | $0.33 \pm 0.09$ | $0.73 \pm 0.44$ | $0.41 \pm 0.23$ | $0.31 \pm 0.12$ | $0.56 \pm 0.11$ | $0.76 \pm 0.50$ |
| Car | $0.31 \pm 0.13$ | $0.37 \pm 0.04$ | $0.34 \pm 0.08$ | $1.17 \pm 0.95$ | $0.51 \pm 0.07$ | $0.3 \pm 0.07$ | $0.91 \pm 0.13$ | $1.31 \pm 1.09$ |
| Cellphone | $0.11 \pm 0.03$ | $0.17 \pm 0.02$ | $0.19 \pm 0.03$ | $0.67 \pm 0.52$ | $0.21 \pm 0.02$ | $0.24 \pm 0.05$ | $0.65 \pm 0.09$ | $0.90 \pm 0.62$ |
| Chair | $0.19 \pm 0.02$ | $0.30 \pm 0.05$ | $0.56 \pm 0.13$ | $0.86 \pm 0.51$ | $0.52 \pm 0.07$ | $0.54 \pm 0.06$ | $0.90 \pm 0.17$ | $0.93 \pm 0.64$ |
| Fireplace | $0.12 \pm 0.04$ | $0.16 \pm 0.05$ | $0.20 \pm 0.10$ | $1.23 \pm 0.45$ | $0.56 \pm 0.14$ | $0.31 \pm 0.11$ | $1.58 \pm 0.35$ | $2.13 \pm 0.82$ |
| Lamp | $0.09 \pm 0.04$ | $0.23 \pm 0.06$ | $0.48 \pm 0.27$ | $0.74 \pm 0.47$ | $0.36 \pm 0.13$ | $0.38 \pm 0.12$ | $0.94 \pm 0.29$ | $0.88 \pm 0.64$ |
| Monitor | $0.24 \pm 0.11$ | $0.18 \pm 0.05$ | $0.45 \pm 0.14$ | $0.68 \pm 0.27$ | $0.32 \pm 0.11$ | $0.33 \pm 0.12$ | $0.75 \pm 0.21$ | $0.84 \pm 0.56$ |
| Sofa | $0.15 \pm 0.05$ | $0.34 \pm 0.11$ | $0.55 \pm 0.22$ | $0.78 \pm 0.17$ | $0.48 \pm 0.11$ | $0.52 \pm 0.16$ | $0.89 \pm 0.24$ | $0.73 \pm 0.14$ |
| Speaker | $0.11 \pm 0.04$ | $0.18 \pm 0.07$ | $0.50 \pm 0.33$ | $0.83 \pm 0.39$ | $0.35 \pm 0.12$ | $0.39 \pm 0.12$ | $0.90 \pm 0.40$ | $0.91 \pm 0.60$ |
| Table | $0.23 \pm 0.09$ | $0.26 \pm 0.11$ | $0.51 \pm 0.27$ | $0.79 \pm 0.29$ | $0.40 \pm 0.12$ | $0.37 \pm 0.12$ | $0.82 \pm 0.32$ | $0.85 \pm 0.35$ |
| Watercraft | $0.12 \pm 0.03$ | $0.14 \pm 0.02$ | $0.26 \pm 0.07$ | $0.71 \pm 0.47$ | $0.25 \pm 0.03$ | $0.37 \pm 0.04$ | $0.54 \pm 0.04$ | $1.01 \pm 0.73$ |
| Average | $0.60 \pm 0.09$ | $0.34 \pm 0.05$ | $0.31 \pm 0.15$ | $0.58 \pm 0.26$ | $0.48 \pm 0.18$ | $0.29 \pm 0.15$ | $0.89 \pm 0.27$ | $1.09 \pm 0.72$ |

The second to fifth, and sixth to ninth columns list the PM distances for input point clouds without and with noise, respectively. The values in the forth column correspond to the PM distances of the isomorphic mesh obtained after applying AtlasNet to an input point cloud of 30,000 points.

### Meshes

As the reference mesh for all data and avoided the triangular patch subdivision process. Moreover, Voxel2mesh was trained using the same training dataset as AtlasNet. When the proposed method and the three baselines (Point2mesh, AtlasNet, and Voxel2mesh) were applied to one point cloud, the isomorphic mesh was generated only once.

In addition to the three methods above, many other neural-network-based methods exist for generating mesh models [43], [44], [46], [65], [71]. However, because directly generating isomorphic meshes using these methods is difficult, as mentioned in Section II, they were not included as baselines in the experiment.

#### 1) Comparison to Baselines Using Input Point Clouds

*Without Noise:* AtlasNet trained objects of 13 classes in the ShapeNet dataset [73] (airplane, bench, cabinet, car, cellphone, chair, fireplace, lamp, monitor, sofa, speaker, table, watercraft). Therefore, in cases where models of trained classes were used as the input point cloud, we selected five objects from each of the 13 classes. We used the three simulation data and five bottle objects from the ShapeNet dataset as the input point clouds of untrained classes. Here, AtlasNet was trained using point clouds with 30,000 points, although the three simulation data (rabbit, Max Planck, and box) consist of approximately 2,500 points. Therefore, in the cases without noise, for the input point clouds from the ShapeNet dataset, AtlasNet was applied to both types of input point clouds, those comprising 30,000 and those comprising approximately 2,500 points. However, other methods have been applied to input point clouds with approximately 2,500 points. All ShapeNet data were scaled to fit a sphere with a diameter of 1 [mm] (Bottle), whereas the scales of the three meshes of Rabbit, Max Planck, and Box were adjusted so that the meshes fit within a bounding box with a long side of approximately 100.0 [mm]. The last row of Table III shows the average PM distance of the 65 objects included in the trained classes. Here, the untrained classes include objects scaled to fit a sphere with a diameter of 1 [mm] (Bottle), whereas the scales of the three meshes of Rabbit, Max Planck, and Box were adjusted so that the meshes fit within a bounding box with a long side of approximately 100.0 [mm]. The last row of Table III shows the average PM distance of the 65 objects included in the trained classes. Here, the untrained classes include objects scaled to fit a sphere with a diameter of 1 [mm] (Bottle), whereas the scales of the three meshes of Rabbit, Max Planck, and Box were adjusted so that the meshes fit within a bounding box with a long side of approximately 100.0 [mm]. The last row of Table III shows the average PM distance of the 65 objects included in the trained classes.

**Table III** shows the PM distances of iMG and Point2mesh are smaller than those of other methods. Additionally, iMG and Point2mesh can recover the shape of the target object with high accuracy, whereas AtlasNet and Voxel2mesh produced incomplete shapes of the objects from untrained classes. The objects of untrained classes, Fig. 14 shows that AtlasNet and Voxel2mesh can reconstruct the approximate shape of the box or bottle nearly as accurately as with trained classes. However, even in this case, Table III shows that the shape-recovery accuracy of iMG is higher than that of other methods.

The red patches depicted in Fig. 15 denote the flipped patches in the isomorphic meshes generated by our iMG and Point2mesh. The flipped patches contribute to the generation of the isomorphic meshes with unnatural and noisy surfaces, as shown in the results of Point2mesh with noise in Fig. 14.
Fig. 14. Results of Experiment 4: Ground truth meshes, input point clouds, and isomorphic meshes generated using our iMG, Point2mesh, AtlasNet, and Voxel2mesh. For AtlasNet, isomorphic meshes were generated from the input point clouds of 30,000 points.

Fig. 15. Isomorphic meshes generated by (a) our iMG and (b) Point2mesh from input point clouds (left) without noise and (right) with noise. Flipped patches are displayed in red.

Processing accuracy using noisy isomorphic meshes is consequently compromised. Hence, reducing the number of flipped patches in the meshes is important.

As displayed in Fig. 15, in the case of the data-free methods, the results obtained with Point2mesh (Fig. 15(b)) include many flipped patches. On the contrary, our iMG enables the generation of isomorphic meshes with fewer flipped patches while capturing detailed shape features (Fig. 15(a)). In addition, Table III indicates that the PM distances of iMG are equal to or lower than Point2mesh in all classes except the airplane. These results reveal that iMG has a higher shape-recovery accuracy than Point2mesh.

2) Comparison to Baselines Using Input Point Clouds With Noise: To assess robustness to noise, uniform noise was added to the input point clouds comprising approximately 2,500 points. In Experiment 2, robustness was verified by adding nine patterns of noise, generated by varying the values of parameters \( \delta \) and \( \sigma \),
to three simulation data. Experiment 4 used input point clouds with noise generated for $(\delta, \sigma) = (40, 0.03)$, which is one of the nine patterns. The 40 and 0.03 were the median values of the parameters used in Experiment 2. In the eighth to eleventh columns of Fig. 14, the first to 13th and 14th to 17th rows show the isomorphic meshes obtained by applying our iMG and the three baselines to the noisy point clouds from both trained and untrained classes. In the sixth to ninth columns of Table III, the third to 16th, and 17th to 21st rows list the PM distances. The isomorphic mesh obtained using iMG was selected as the one with the shortest PM distance among the global, coarse local, and fine local mapping results.

The eighth to eleventh columns in Fig. 14 show that compared with the third to sixth columns in Fig. 14, the shape-recovery accuracy of iMG generally decreased. However, as Table III shows, in the sixth to ninth columns, the PM distances of iMG and Point2mesh were shorter than those of AtlasNet and Voxel2mesh. Although Point2mesh has the shortest PM distance, as displayed in Fig. 15, the isomorphic meshes generated by Point2mesh (Fig. 15(b)) include many flipped patches. By contrast, our iMG with the second-shortest PM distance can generate an isomorphic mesh that represents the overall shape features of the target object with fewer flipped patches.

V. DISCUSSION

In this section, we discuss the shape-recovery accuracy of iMG and its robustness to noise based on the experimental results in Section IV.

A. Shape-Recovery Accuracy

The results of Experiment 1 in Figs. 9 and 10 and Table I indicate that iMG enabled the generation of isomorphic meshes for input point clouds without noise and simultaneously maintained the mesh structure of the reference mesh. In particular, iMG achieved the highest accuracy in recovering the shapes of curved surfaces with large curvatures.

By contrast, as indicated in the box model results, the accuracy of recovering edge shapes in fine local mapping (Fig. 10(f)) were inferior to that of coarse local mapping (Fig. 10(e)). In fine local mapping, the reference mesh was divided into local meshes based on the shape of the input point cloud. In the box model, local meshes were generated around edges, as described in Section III-D1. To smoothly integrate the deformed local mesh, the weighted average of vertex positions in the overlapping local meshes was calculated as the vertex position in the overlapping region. When many overlapping regions exist around an edge, the shapes as well as the edges of the regions are recovered smoothly through the integration process. This explains why the accuracy of recovering edge shapes is higher after coarse local mapping than after fine local mapping.

In Experiments 1 and 4, the isomorphic meshes were generated using two types of reference meshes with 36,002 and 2,562 vertices. Table I and the 18th to 20th rows of Table III show the PM distances obtained using the two reference meshes with 36,002 and 2,562 vertices, respectively. In comparison, the PM distances with 36,002 vertices were 0.05–0.08 [mm] lower than those with 2,562 vertices. This result indicates that iMG improved shape-recovery accuracy with an increase in the number of vertices in the reference mesh.

In this study, because we targeted the generation of isomorphic meshes, the mesh resolution of the reference mesh was fixed regardless of the object shape. However, for applications that require a higher shape recovery accuracy, it will be necessary to adaptively change the mesh resolution of the reference mesh according to object shapes.

Figs. 16(a) and (b) detail the isomorphic meshes generated from input point clouds with small and large coarse areas, respectively. As displayed in Fig. 16, the coarseness of the isomorphic mesh is proportional to that of the input point cloud. Generating an isomorphic mesh with a unified mesh structure from a point cloud with large coarse areas can be achieved by preprocessing the point-cloud distribution to make it uniform. Moreover, in a point cloud measured by a sensor, occlusions often cause areas where the point cloud density differs significantly, as shown in Fig. 13(d)–(f). Even when such a point cloud is input, Fig. 13(g)–(i) show that the proposed method generates an isomorphic mesh as a continuous surface.

To verify the accuracy of the semantic correspondence, we compared two correspondence methods. One is the semantic correspondence determined using only the vertex ID number. In the other method, each vertex is mapped toward the direction of the vector connecting the centre of gravity of one mesh and the vertex. The correspondence is found by using polar coordinates of the mapped vertices on a unit sphere. The latter is considered as the general correspondence.

To show the semantic correspondence, in Fig. 17(a), the isomorphic meshes generated by iMG are displayed by assigning a colour to the vertex in the mesh according to the vertex ID number. To show the general correspondence, in Fig. 17(b), the isomorphic meshes of the sofas are colour-coded based on the polar coordinates of the vertices mapped on the unit sphere. In these figures, the areas with the same colour can be seen as corresponding. Moreover, the vertices having similar colours indicate that the positional relationship between the vertices is close. In addition, the red of Fig. 17(c) indicates regions where two or more locations on the surface of each sofa are represented by the same polar coordinate in Fig. 17(b).

In Fig. 17(a), all vertices in each mesh are represented by a different colour, and no vertices have the same colour except those in correspondence within the two meshes. This indicates that the one-to-one vertex correspondence is satisfied between the two meshes. Moreover, common parts, such as the tyres of the...
B. Robustness to Noise

Experiments 2 and 3 were conducted to verify whether iMG could generate isomorphic meshes from the two types of input point clouds with noise (Fig. 11) produced by adding artificial noise and that (Fig. 13) obtained by scanning real models by using a TrueDepth camera.

The results of Experiment 2 (Figs. 11 and 12) show that the overall shape-recovery accuracy tended to decrease in inverse proportion to the increase in the ratio $\delta$ of the added noise and the noise variance $\sigma$. The maximum lengths of the bounding boxes covering the rabbit, Max Planck, and box models were 98.0, 98.9, and 100 [mm], respectively. Fig. 12 indicates that when the maximum length is used as the object size, because all object sizes are almost 100 [mm], the PM distances for all the objects are less than 2.0% of the object size. This indicates that iMG is robust to noise.

Fig. 12 also reveals that in the case of the box model, the PM distance changed depending on the value of $\sigma$ only when $\delta$ was fixed. By contrast, when $\sigma$ is fixed, the PM distance is almost the same, regardless of the value of $\delta$. In the rabbit and Max Planck models, the PM distance changed depending on the values of $\sigma$ and $\delta$. Here, because the box model has only edges and planes, even when $\delta$ is increased, the effect of noise on the target object remains almost the same. Alternatively, the rabbit and Max Plank models consist of curved surfaces of various shapes and if $\delta$ increases, noise is added to the curved surfaces. Because recovering a smooth surface from a noise point cloud is difficult compared to plane shape recovery, the PM distances of the two models increased.

Moreover, using the results of Experiments 2 and 3, we analysed the influence of noise and missing parts on the
shape-recovery accuracy of the iMG in detail. Fig. 18 shows the PM distances between the isomorphic mesh obtained using iMG and its input mesh. In Fig. 18, the input mesh was the surface mesh with noise corresponding to the input point cloud in Experiment 2 and a measurement mesh in Experiment 3. When the noise was small, the PM distance of the isomorphic mesh was longer than that of the input mesh. By contrast, when the noise was large, e.g. $\delta = 60$ and $\sigma \geq 0.03$, the PM distance of the isomorphic mesh was shorter than that of the input mesh in all models.

Because iMG is a data-free method, an approximation error occurs when fitting curved surfaces to the input point cloud with and without noise. When noise is low, approximation errors influence the shape recovery accuracy. Therefore, the shape recovery accuracy of the isomorphic mesh was lower than that of the input mesh. However, when the amount of noise is large, the approximation by the curved surfaces reduces the effect of noise. Because the effect of noise reduction was greater than that of the approximation error, the shape-recovery accuracy of the isomorphic mesh was higher than that of the input mesh when the amount of noise was large.

Fig. 18 reveals that the noise in the measurement mesh of the rabbit, Max Planck, and the box was close to $(\delta, \sigma) = (60, 0.03)$, $(60, 0.03)$, and more than $(60, 0.05)$, respectively. However, the PM distances of isomorphic meshes generated from the measurement data are longer than those of isomorphic meshes generated from the simulation data with the settings of $\delta$ and $\sigma$ because the measurement mesh of Experiment 3 contains not only noise but also missing parts, which renders mesh generation more difficult than the generation from simulation data. However, as displayed in Fig. 13, iMG generated isomorphic meshes from the point cloud with noise while filling the missing parts. Furthermore, the PM distances of the isomorphic meshes were almost equal to or shorter than those of the measured meshes for all models. Therefore, our iMG can generate isomorphic meshes from a scanned point cloud using mobile sensors without introducing mesh-generation errors larger than measurement errors.

In Experiment 4, the shape-recovery accuracy of iMG was compared with those of AtlasNet, Point2mesh, and Voxel2mesh. As presented in the tenth column of Fig. 14, because the data-driven AtlasNet learned the shape characteristics of the edges in advance, the edge and plane shape of the box was represented even though some noise was added to the input point cloud. On the contrary, the data-free iMG generates isomorphic meshes only from the input point cloud. Therefore, as shown in the 16th row and eighth column in Fig. 14, the edge shapes of the obtained isomorphic mesh of the box became smoother than the ground truth shape (the 16th row and first column in Fig. 14). From these results, to improve the recovery accuracy of edge shapes, we intend to combine data-driven and data-free frameworks in the future.

C. Limitations

Because a spherical surface is used as the reference mesh, iMG always outputs isomorphic meshes with genus zero. Therefore, the generation of isomorphic meshes using iMG is possible while filling the missing parts of an input point cloud. However, iMG also fills the holes that the objects originally contained. If a genus-$\gamma$ surface is provided as a reference mesh, there is a high possibility of generating an isomorphic mesh of genus-$\gamma$. However, depending on the number of genera in the reference mesh, the division methods in Steps 2 and 3 need to be changed.

When the number of points in the local cloud is low, overfitting to the local cloud occurs because only a few points can be used for MLP training. Overfitting often results in flipped patches in each local mesh. Therefore, although iMG generates an isomorphic mesh with fewer flipped patches than Point2mesh, flipped patches typically occur in isomorphic meshes obtained by iMG. Moreover, in the missing parts where no points exist, the distribution of vertices in the isomorphic mesh became sparse.

Finally, the advantage of data-free methods including iMG is that they do not require large amounts of training data, while their disadvantage is that they require more computation time than data-driven methods. However, it is essential to speed up iMG to apply to real-world scenarios. Therefore, one of our future works is to accelerate iMG by introducing a data-driven framework for iMG.
VI. CONCLUSION

We proposed iMG for generating an isomorphic mesh from a point cloud containing noise and missing parts. In the proposed iMG, a reference mesh is mapped to the input point cloud in a step-by-step manner. This process enables flexible deformation and simultaneously maintains the structure of the reference mesh. Moreover, it is data-free and does not require training data.

In the experiments, isomorphic meshes were generated using point clouds with and without noise. The experimental results revealed that the proposed iMG had a higher shape-recovery accuracy than conventional methods. Moreover, isomorphic meshes were generated using point clouds acquired using a mobile sensor. We experimentally confirmed that iMG could output an isomorphic mesh with a shape close to the ground truth while filling in missing parts.

ACKNOWLEDGMENT

Special thanks are extended to Kyo Itaya at Okayama University for his assistance with the experiments.

REFERENCES

[1] H. Liu, J. Jia, and N. Z. Gong, “PointGuard: Provably robust 3D point cloud classification,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 6186–6195.
[2] C. Wen, X. Li, X. Yao, L. Peng, and T. Chi, “Airborne LiDAR point cloud classification with global-local graph attention convolution neural network,” ISPRS J. Photogrammetry Remote Sens., vol. 173, pp. 181–194, 2021.
[3] X. Liu, Z. Han, Y.-S. Liu, and M. Zwicker, “Fine-grained 3D shape classification with hierarchical part-view attention,” IEEE Trans. Image Process., vol. 30, pp. 1744–1758, 2021.
[4] F. A. Azcona et al., “Interpretation of brain morphology in association to Alzheimer’s disease dementia classification using graph convolutional networks on triangulated meshes,” in Proc. Int. Workshop Shape Med. Imag., Springer, 2020, pp. 95–107.
[5] Z. Lakhili, A. El Alami, A. Mesbah, A. Berrahou, and H. Qidraa, “Deformable 3D shape classification using 3D racah moments and deep neural networks,” Procedia Comput. Sci., vol. 148, pp. 12–20, 2019.
[6] D. George, X. Xie, Y. Lai, and G. K. Tam, “A deep learning driven active framework for segmentation of large 3D shape collections,” Comput.-Aided Des., vol. 144, Art. no. 103179.
[7] H. Huang, X. Li, L. Wang, and Y. Fang, “3D-metaconnet: Meta-learning for 3D shape classification and segmentation,” in Proc. Int. Conf. 3D Vis., 2021, pp. 982–991.
[8] S. Qiu, S. Anwar, and N. Barnes, “Dense-resolution network for point cloud classification and segmentation,” in Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis., 2021, pp. 3813–3822.
[9] X. Wang, X. Sun, X. Cao, K. Xu, and B. Zhou, “Learning fine-grained segmentation of 3D shapes without part labels,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 10276–10285.
[10] X. Xu and G. H. Lee, “Weakly supervised semantic point cloud segmentation: Towards 10x fewer labels,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 13706–13715.
[11] L. Han, T. Zheng, L. Xu, and L. Fang, “OccuSeg: Occupancy-aware 3D instance segmentation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 2940–2949.
[12] L. Yi, W. Zhao, H. Wang, M. Sung, and L. J. Guibas, “GSPN: Generative shape proposal network for 3D instance segmentation in point cloud,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 3947–3956.
[13] L. Wang, Y. Huang, Y. Hou, S. Zhang, and J. Shan, “Graph attention convolution for point cloud semantic segmentation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 10296–10305.
[14] F. Yu, K. Liu, Y. Zhang, C. Zhu, and K. Xu, “PartNet: A recursive part decomposition network for fine-grained and hierarchical shape segmentation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 9491–9500.
[15] H.-Y. Meng, L. Gao, Y.-K. Lai, and D. Manocha, “VV-Net: Voxel VAE net with group convolutions for point cloud segmentation,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2019, pp. 8500–8508.
[16] J. Zhang, Y. Wu, T. Zhao, Z. Zhao, and C. Lu, “PointSIFT: A SIFT-like network module for 3D point cloud semantic segmentation,” 2018, arXiv:1807.00652.
[17] H. Su et al., “SPLATNet: Sparse lattice networks for point cloud processing,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 2530–2539.
[18] R. Hanocka et al., “Meshcnn: A network with an edge,” ACM Trans. Graph., vol. 38, no. 4, pp. 1–12, 2019.
[19] J. Chibane, T. Alldieck, and G. Pons-Moll, “Implicit functions in feature space for 3D shape reconstruction and completion,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 6970–6981.
[20] K. Genova, F. Cole, A. Sud, A. Sarna, and T. Funkhouser, “Local deep implicit functions for 3D shape,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 4857–4866.
[21] R. Chabara et al., “Deep local shapes: Learning local SDF priors for detailed 3D reconstruction,” in Proc. Eur. Conf. Comput. Vis., Springer, 2020, pp. 608–625.
[22] Z. Chen, A. Tagliasacchi, and H. Zhang, “BSP-Net: Generating compact meshes via binary space partitioning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 45–54.
[23] L. Gao et al., “SDM-Net: Deep generative network for structured deformable mesh,” ACM Trans. Graph., vol. 38, no. 6, pp. 1–15, 2019.
[24] Z. Chen and H. Zhang, “Learning implicit fields for generative shape modeling,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 5939–5949.
[25] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep learning on point sets for 3D classification and segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 652–660.
[26] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet: Deep hierarchical feature learning on point sets in a metric space,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 5099–5108.
[27] J. Xie, Y. Xu, Z. Zheng, S.-C. Zhu, and Y. N. Wu, “Generative pointnet: Deep energy-based learning on unordered point sets for 3D generation, reconstruction and classification,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 14976–14985.
[28] N. Luo, Y. Wang, Y. Gao, Y. Tian, Q. Wang, and C. Jing, “KNN-based feature learning network for semantic segmentation of point cloud data,” Pattern Recognit. Lett., vol. 152, pp. 365–371, 2021.
[29] S. V. Sheshpannavar and K. Kambhamettu, “A novel local geometry capture in pointnet for 3D classification,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops, 2020, pp. 262–263.
[30] D. Casas, M. Tejera, J.-Y. Guillemaut, and A. Hilston, “Parametric control of captured mesh sequences for real-time animation,” in Proc. Motion Games: 4th Int. Conf., Edinburgh, U.K., Springer, 2011, pp. 242–253.
[31] R. W. Sumner and J. Popović, “Deformation transfer for triangle meshes,” ACM Trans. Graph., vol. 23, no. 3, pp. 399–405, 2004.
[32] T. Heimann and H.-P. Meinzer, “Statistical shape models for 3D medical image segmentation: A review,” Med. Image Anal., vol. 13, no. 4, pp. 543–563, 2009.
[33] K. Morooka, R. Matsubara, S. Miyazaki, T. Fukuda, S. Sugii, and R. Karrasume, “Ancient pelvic reconstruction from collapsed component bones using statistical shape models,” Mach. Vis. Appl., vol. 30, pp. 59–69, 2019.
[34] M. Kazhdan, M. Bolitho, and H. Hoppe, “Poisson surface reconstruction,” in Proc. 4th Eurographics Symp. Geometry Process., 2006.
[35] F. Williams, M. Trager, J. Bruna, and D. Zorin, “Neural splines: Fitting 3D surfaces with infinitely-wide neural networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 9949–9958.
[36] C. Jiang et al., “Local implicit grid representations for 3D scenes,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 6001–6010.
[37] B. Ummenhofer and V. Koltun, “Adaptive surface reconstruction with multiscale convolutional kernels,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 5651–5660.
[38] J. Zhang, Y. Yao, and L. Quan, “Learning signed distance field for multi-view surface reconstruction,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 6525–6534.
[60] S. Xiong, J. Zhang, J. Zheng, J. Cai, and L. Liu, “Robust surface reconstruction algorithm,” in Proc. 25th Ann. Conf. Comput. Assist. Intervention, vol. 157, pp. 237–250, 2018.
[61] M. Liu, X. Zhang, and H. Su, “Meshting point clouds with predicted intrinsic-extrinsic ratio guidance,” 2020, pp. 18583–18592.
[62] Z. Mi, Y. Luo, and W. Tao, “SSRNet: Scalable 3D surface reconstruction network,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 970–979.
[63] N. Sharp and M. Ovsjanikov, “PointTriNet: Learned triangulation of 3D point sets,” 2020, pp. 2005–2013.
[64] B. Ma, Z. Han, Y.-S. Liu, and M. Zwicker, “Neural-pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces,” 2020, arXiv:2011.13495.
[65] M. Yang, Y. Wen, W. Chen, Y. Chen, and K. Jia, “Deep optimized priors for 3D shape modeling and reconstruction,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 3269–3278.
[66] S. Peng, C. Jiang, Y. Liao, M. Niemeyer, M. Pollefeys, and A. Geiger, “Shape as points: A differentiable point solver,” in Proc. Adv. Neural Inf. Process. Syst., 2021, pp. 13032–13044.
[67] A. Boulch and R. Marlet, “POCO: Point convolution for surface reconstruction,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2022, pp. 6302–6314.
[68] U. Wickramasinghe, E. Renelli, G. Knott, and P. Fua, “Voxel2Mesh: 3D mesh model generation from volumetric data,” in Proc. Med. Image Comput. Comput. Assist. Intervention: 23rd Int. Conf., Lima, Peru, Springer, 2020, pp. 299–308.
[69] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, “Occupancy flow: 4D reconstruction by learning particle dynamics,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2019, pp. 5379–5389.
[70] Y. Duan, H. Zhu, H. Wang, L. Yi, R. Nevatia, and L. J. Guibas, “Curriculum DeepSDF,” in Proc. Eur. Conf. Comput. Vis., Springer, 2020, pp. 51–67.
[71] C.-L. Li, T. Simon, J. Saragih, B. Póczos, and Y. Sheikh, “LBS ball-pivoting algorithm for surface reconstruction,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 970–979.
[72] R. A. Potamias, S. Ploumpis, and S. Zafeiriou, “Neural mesh simplification,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2022, pp. 2024–2033.
[73] M. Cuturi, “Sinkhorn distances: Lightspeed computation of optimal transport,” in Proc. Adv. Neural Inf. Process. Syst., 2013, pp. 2292–2300.
[74] A. Y. Chang et al., “ShapeNet: An information-rich 3D model repository,” 2015, arXiv:1512.03012.

Shoko Miyauuchi (Member, IEEE) received the PhD degree in engineering from Kyushu University, in 2018. She is an assistant professor with the Graduate School of Information Science and Electrical Engineering, Kyushu University, Japan. Her research interests include medical image processing and 3D shape modelling.

Ken’ichi Morooka (Member, IEEE) received the BE degree in electronic engineering and the PhD degree from Kyushu University, in 1995 and 2000, respectively. He was associated with the Institute of Systems & Information Technologies/KYUSHU. From 2000 to 2006, he was a research associate with Imaging Science and Engineering Laboratory, Tokyo Institute of Technology. He was an Associate Professor with Digital Medicine Initiative, and with the Graduate School of Information Science and Engineering, Kyushu University from 2006 to 2020. He joined as a professor with the Graduate School of Natural Science and Technology, Okayama University, in 2020-2023. Since 2023, he is a professor with Kumamoto University, Japan. His research interests cover computer vision, computer graphics, medical imaging and pattern recognition.

Ryo Kurazume (Senior Member, IEEE) received the MEng and PhD degree in mechanical engineering from the Tokyo Institute of Technology, in 1989 and 1998. He is a professor with the Graduate School of Information Science and Electrical Engineering, Kyushu University. He was a director of the Robotics Society of Japan (RSJ), a director of the Society of Instrument and Control Engineers (SICE), and a chairperson of the Japan Society of Mechanical Engineers (JSME) Robotics and Mechatronics Division. He received JSME Robotics and Mechatronics Academic Achievement Award, SICE System Integration Division Academic Achievement Award, and JSME Robotics and Mechatronics Division Robotics and Mechatronics Award. He is RSJ fellow, JSME fellow, and SICE fellow. His current research interests include legged robot control, computer vision, multiple mobile robots, service robots, care technology, and biometrics.