arXiv:2005.11617v1 [cs.GR] 23 May 2020

MeshODE: A Robust and Scalable Framework for Mesh Deformation

JINGWEI HUANG*, Stanford University
CHIYU "MAX" JIANG*, University of California, Berkeley
BAIQIANG LENG, Tsinghua University
BIN WANG, Tsinghua University
LEONIDAS GUIBAS, Stanford University

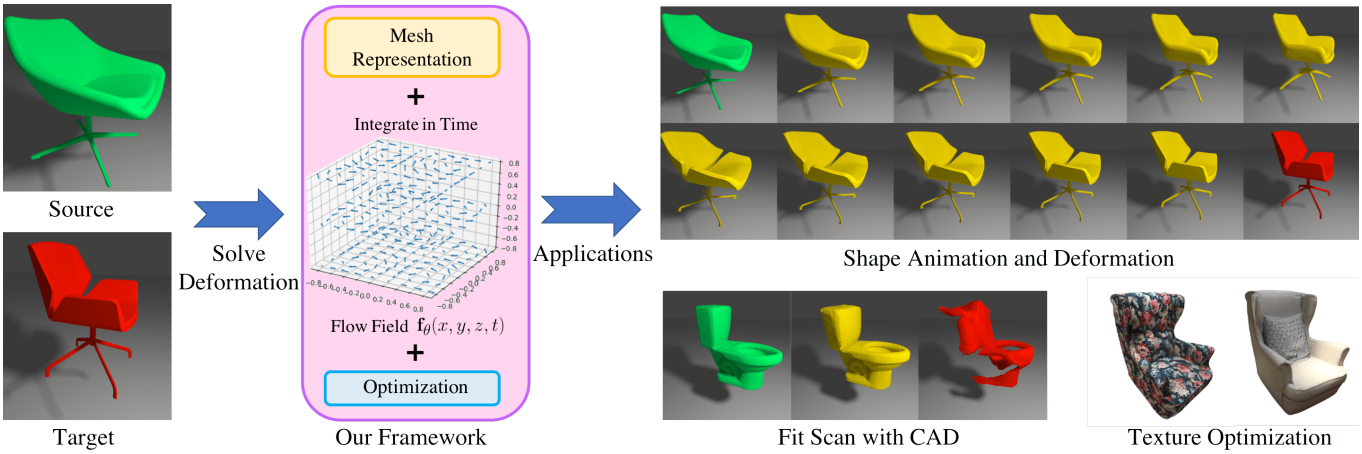

Fig. 1. MeshODE is a robust and scalable framework that addresses the problem of pairwise shape deformation without prespecified correspondences. It enables a range of computer graphics and computer vision applications including shape animation and creation, scan-to-CAD fitting and texture reconstruction.

We present MeshODE, a scalable and robust framework for pairwise CAD model deformation without prespecified correspondences. Given a pair of shapes, our framework provides a novel shape feature-preserving mapping function that continuously deforms one model to the other by minimizing fitting and rigidity losses based on the non-rigid iterative-closest-point (ICP) algorithm. We address two challenges in this problem, namely the design of a powerful deformation function and obtaining a feature-preserving CAD deformation. While traditional deformation directly optimizes for the coordinates of the mesh vertices or the vertices of a control cage, we introduce a deep bijective mapping that utilizes a flow model parameterized as a neural network. Our function has the capacity to handle complex deformations, produces deformations that are guaranteed free of self-intersections, and requires low rigidity constraining for geometry preservation, which leads to a better fitting quality compared with existing methods. It additionally enables continuous deformation between two arbitrary shapes without supervision for intermediate shapes. Furthermore, we propose a robust preprocessing pipeline for raw CAD meshes using feature-aware subdivision and a uniform graph template representation to address artifacts in raw CAD models including self-intersections, irregular triangles, topologically disconnected components, non-manifold edges, and nonuniformly distributed vertices. This facilitates a fast deformation optimization process that preserves global and local details.

On top of the methodological contributions, we create an evaluation benchmark for unsupervised pairwise shape deformation and find that our deformation results significantly outperform the state-of-the-art deformation algorithms with respect to fitting error. We show that our framework benefits various downstream applications including novel shape design and animation, scan-to-CAD fitting, and texture transfer. Our code is publicly available[1].

CCS Concepts: • **Computing methodologies → Shape modeling**; *Mesh geometry models*; Computer graphics.

Additional Key Words and Phrases: Shape Deformation, Neural Deformation

1 INTRODUCTION

The Shape Deformation Problem. 3D shape deformation is a fundamental problem in computer graphics and computer vision with various applications in these fields. For starters, shape deformation as a means of geometry editing can be useful for creating novel digital content by deforming a preexisting model. [Jiang et al. 2017; Sorkine and Alexa 2007] deform a source 3D model to a target shape given pairs of sparse correspondences between the source and the target to create novel shape designs. Such deformation can also be transferred to novel source shapes [Sumner and Popović 2004; Yifan et al. 2019]. Moreover, shape deformation can be used for computer animation [Anguelov et al. 2005; Jacobson and Sorkine 2011] by creating smooth interpolations between key-frames. Recently, [Wang et al. 2019; Yifan et al. 2019] addressed the more challenging

*Both authors contributed equally to the paper

Authors' addresses: Jingwei Huang, Stanford University; Chiyu "Max" Jiang, University of California, Berkeley; Baiqiang Leng, Tsinghua University; Bin Wang, Tsinghua University; Leonidas Guibas, Stanford University.

[1] https://github.com/hjwdzh/MeshODE

problem of deforming a source to an exemplar target shape without prespecified correspondences. It opens the opportunity for novel shape synthesis and animation with minimal human intervention, requiring only an exemplar target 3D model without any form of correspondence labels. Such unsupervised deformation also plays an important role in improving the fitting quality of deformed CAD models to match scanned models, which is a topic of growing interest in the vision community [Avetisyan et al. 2019a,b; Dahnert et al. 2019; Uy et al. 2020].

Challenges in shape deformation. Shape deformation is a conceptually challenging problem. A good shape deformation usually requires a good alignment of the deformed shape to the target model while preserving the shape features of the source model. The first criterion usually conflicts with the second since a perfect alignment to the target precludes the preservation of the source features. To balance the criteria, traditional methods [Jiang et al. 2017; Sorkine and Alexa 2007; Sorkine and Botsch 2009] jointly optimize for the rigidity of the source mesh and the fitting error of the correspondences specified by humans. For unsupervised shape deformation without explicit correspondences, iterative-closest-point (ICP) algorithm [Besl and McKay 1992] can be used to provide supervision for the optimization process. However, it can practically yield severe artifacts given complex CAD model structures. Recent works introduce deep learning methods to directly predict point-wise offsets [Wang et al. 2019] or to predict a control cage [Yifan et al. 2019] for more global deformations. However, direct predictions of vertex offsets do not offer sufficient regularization to the resulting shape, oftentimes leading to corrupt shape features such as self-intersections and non-physical distortions. On the other hand, predicting a coarse control cage adds too much regularization to the process, limiting the admissible degrees of freedom to allow a good fit between the deformed geometry and the target shape, leading to large fitting errors. Furthermore, these learning-based methods suffer from generalization issues and have limited capability for handling novel shapes.

Proposed formulation. We propose a novel and intuitive flow-based deformation framework that attempts to address all of the aforementioned challenges. In essence, we view the process of shape deformation as the process of convecting the geometric manifold of the source shape at time $t = 0$ along with a temporally varying flow $[v_x, v_y, v_z] = f_\theta(x, y, z, t)$ to time $t = 1$ to match the target geometry. The flow $f_\theta(x, y, z, t)$ can be parameterized using a neural network with trainable parameters θ that can be optimized via gradient descent. The deformation process can be integrated using a differentiable ODE solver [Chen et al. 2018]. We use a simple geometric matching loss (the 2-way squared Chamfer Distance computed using the ICP algorithm) between the deformed shape and the target shape together with a standard rigidity loss to optimize the neural flow model.

The flow-based formulation has various inherent advantages compared with competing approaches. First, we show that any mesh deformation-induced via a sufficiently smooth and continuous flow is guaranteed to prevent the self-intersection of the original surfaces. This is crucial in limiting the space of deformations to a physical and visually appealing one since it is one of the main challenges facing models that directly predict vertex offsets or optimize for vertex positions. For example, we guarantee that front and back faces of table tops do not collide during deformation. Second, we show that the flow naturally induces a bijective mapping $\mathcal{D} : \mathbb{R}^3 \to \mathbb{R}^3$ between the source and target geometries, which inherently guarantees the *source to target to source* cycle consistency for deformations, which further eases the optimization when we adopt the bidirectional fitting loss. Third, compared to a traditional parameterization of deformations using control cages or control points, the flow-based approach allows for a much higher degree of freedom for deformations due to the high representational capacity of neural networks. Last but not least, the continuous and smooth nature of the flow allows for seamless interpolation between the source and target states, allowing for smooth animations between shapes.

Mesh Preprocessing. To further extend our model to "CAD meshes in the wild" (e.g. ShapeNet [Chang et al. 2015]) that contain artifacts such as irregular triangles, non-uniform vertex distribution, non-watertight connectivity, topologically disconnected components, and self-intersections, we propose a robust computational pipeline for mitigating such irregularities, including a mesh subdivision operation, a virtual link construction, and a uniform skeleton template representation (Section 4). These steps aim to produce uniform vertex samples with geometry-aware connections for rigidity loss modeling, and thereby guarantee a fast and valid deformation optimization and preserve global and local details (e.g. geometric connectivity and sharp features). As a result, our ICP-based optimization does not have artifacts reported by [Yifan et al. 2019] and significantly outperforms the existing learning approaches based on fitting errors.

We provide a benchmark consisting of randomly selected pairs of shapes from the ShapeNet repository inside the chair, table, and sofa categories, where each pair is from the same category. We exhaustively evaluate the performance of different methods. The experiment shows that we significantly outperform the state-of-the-art according to fitting errors. We demonstrate that our framework is beneficial for many downstream applications including shape interpolation and animation, scan-to-CAD fitting, and texture reconstruction. In summary, our main contributions are:

- Introduction of a novel bijective flow-based deformation function with intrinsic smoothness and powerful representation capacity.
- Formulating a robust mesh preprocessing pipeline for dealing with complex structures of CAD models, thus enabling a practical, robust and scalable deformation framework.
- Evaluating deformation performance on a benchmark dataset, in which we significantly outperform the state-of-the-art.
- Illustration of the applicability of our framework to benefit various downstream applications including shape animation, scan-to-CAD fitting, and texture reconstruction.

2 RELATED WORKS

Mesh Deformation. Detail-preserving deformation is an important direction in geometry processing and has been studied for decades [Sorkine and Botsch 2009]. In supervised deformation, users

prespecify corresponding pairs of points in the source and target shape, upon which the deformation algorithm deforms the source shape to match the target on the prescribed points using some sort of interpolation [Chao et al. 2010; Huang et al. 2008; Levi and Gotsman 2014; Sorkine and Alexa 2007]. The key challenge is to ensure feature preservation by enforcing minimum distortion or rigidity. Without explicit correspondences, our deformation method optimizes the fitting and rigidity loss using correspondences from the iterative-closest-point (ICP) algorithm. Our novel neural deformation function is intrinsically smooth and does not require strong rigidity loss during optimization, which results in better fitting quality.

A different class of methods use a smooth parametric function so that the structure and the shape details are guaranteed to be preserved during the interpolation process. Such smooth functions can be implemented as the deformation of vertices on a coarse cage mesh template so that the deformation of the surface points are interpolated from the template [Calderon and Boubekeur 2017; Joshi et al. 2007; Ju et al. 2005; Lipman et al. 2008; Sacht et al. 2015; Thiery et al. 2012; Xian et al. 2012]. Other methods implement such functions with coarse regular or spherical grid structures to model the deformation in the image or camera space [Huang et al. 2017; Zhou and Koltun 2014; Zhou et al. 2013]. However despite successes in these methods to preserve geometric details, a deformation cage has limited degrees of freedom to describe subtle deformations. Although regular grids have more degrees of freedom, they require a strong rigidity constraint to avoid severe distortion and thus also has limited capacity for deformation. Our parametric function is a neural network-based function which has powerful representation capacity for complex deformations, while its intrinsic smoothness ensures a sufficient level of rigidity. Moreover, our deformation is a bijective mapping that enables unsupervised continuous deformations between shapes from both directions.

Recent works try to incorporate learning-based methods for shape deformation. For example, [Groueix et al. 2018; Wang et al. 2018] learns deformations of shapes based that share a common mesh template. This approach is limited by a strong assumption of common topology between the geometries that are modelled. A different set of methods model the deformation as a field stored on a 3D lattice grid [Hanocka et al. 2018; Jack et al. 2018; Yumer and Mitra 2016], where the resolution limits the capacity to accurately represent the source shapes. NeuralCage [Yifan et al. 2019] is able to preserve shape details by learning a deformation cage, but it suffers from limited representation capacity similar to the traditional cage-based deformation techniques. Finally, such deep-learning based methods are hard to generalize to shapes beyond the training set. In our proposal, the deformation function is a neural network, but we directly optimize for the parameters of the network and therefore do not suffer from generalization issues.

Neural Bijectors as Normalizing Flows. In the machine learning community, generative models are usually considered as a mapping from a prior distrution (e.g., a normal distribution) to a data distribution. Normalizing Flows [Rezende and Mohamed 2015] is proposed as a bijective map in order to effectively model a complex distribution. Bijection allows such flows to be trained by learning the mapping from the data distribution to the prior distrubition by using the maximum likelihood loss, and the inverse can be used to sample new data from the prior. Among the first proposed approaches are the planar and radial flows whose Jacobian is easy to compute. Such models lack expressivity in higher dimensions, hence autoregressive flow models are proposed to address the issues [Dinh et al. 2016; Kingma et al. 2016; Papamakarios et al. 2017]. NeuralODE [Chen et al. 2018; Grathwohl et al. 2018] models the flow as an integration of velocity through the time, and allow continuous transformation between distributions. For example, PointFlow [Yang et al. 2019] develops a continuous normalizing flow based on it for shape generation. While all existing methods train the neural network to represent map a prior distribution to a target data distribution, we consider it as a bijective map with powerful presentation capacity that directly maps between two geometries in the physical space. We formulate it as a light-weight optimization problem without the need to pretrain on a large-scale dataset, thus avioding the generalization problem.

Shape Registration. Our optimization process can be viewed as obtaining a deformation for minimizing the difference between two shapes. A similar topic called shape registration has been researched for decades to align point clouds from scans. Iterative closest point (ICP) [Besl and McKay 1992; Chen and Medioni 1992] is an algorithm employed to solve this problem. An overview of the variants of this algorithm [Díez et al. 2015; Rusinkiewicz and Levoy 2001] suggests that the key challenges are finding better correspondences and performing outlier-tolerant optimization. [Fitzgibbon 2003] minimizes the distance between samples using the Levenberg-Marquardt solver. [Mitra et al. 2004] propose to identify the closest point efficiently using the distance field for shape optimization. Our problem is more like a non-rigid ICP problem [Li et al. 2008; Newcombe et al. 2015; Sumner et al. 2007]. The difference is that we are using neural ODE as a deep flexible template, and we deal with specific challenges of applying non-rigid ICP optimization for CAD models with complex geometry and topology.

Mesh Representation. Subdivision is a common technique to upsample a coarse mesh using interpolation to obtain a more accurate geometry representation [Catmull and Clark 1978; Doo 1978; Dyn et al. 1990; Kobbelt 2000; Stam 1998]. In contrast to direct interpolation, we aim to robustly upsample the mesh, preserve the original geometry, and obtain right-angle triangles with edges following principal curvatures. While quadrangulation algorithms [Huang et al. 2018b; Jakob et al. 2015] produce such mesh properties, they do not guarantee the preservation of all geometry details. Instead, we simply sample grid positions for each triangle along axes of computed orientation fields and remesh each triangle with a 2D delaunay triangulation [Lee and Schachter 1980]. Since CAD models have geometrically close but topologically disconnected components, several works repair models by manifold conversion [Chu et al. 2019; Huang et al. 2018a]. However, [Huang et al. 2018a] does not preserve sharp features and [Chu et al. 2019] produces unexpected shape boundaries. We do not rely on manifold topology, and simply build additional virtual links connecting close vertices to effectively model rigidity losses. To further enhance the performance, we used the uniform skeleton template as a proxy and transfer the

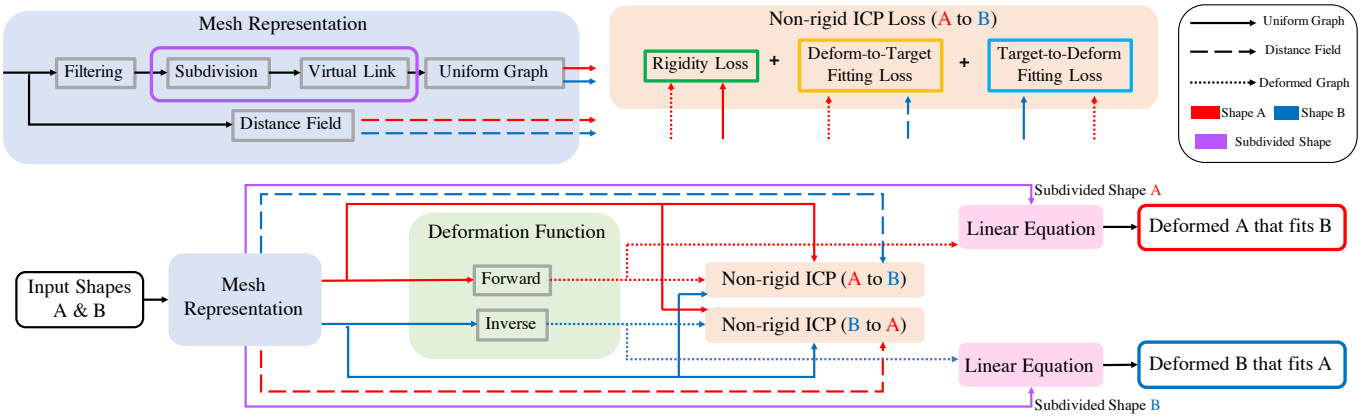

Fig. 2. Framework Pipeline. In the pre-processing stage, we perform a novel subdivision and build an additional uniform skeleton template. In the optimization stage, we solve for a deformation function that applies to the uniform skeleton. The optimization is guided by correspondences using iterative closest point algorithm with energy as the sum of squared distance and a rigidity loss [Sorkine and Alexa 2007]. Finally, we apply the deformation of the skeleton template to the original CAD model by solving a linear system.

deformation to the source mesh using a linear solver. Similar ideas are commonly used for skinning and skeletal animation [Capell et al. 2002] where linear interpolation is not favored. In our scenario, the template graph is a relatively dense approximation where a linear approximation is sufficiently accurate.

3 OVERVIEW

Figure 2 shows a schematic of the proposed deformation framework. In the pre-processing stage, we perform a subdivision and build a uniform skeleton template as a proxy. In the optimization stage, we solve for a deformation function that applies to the uniform skeleton. The optimization is guided by correspondences using the ICP algorithm with the sum of squared distances together with a rigidity loss [Sorkine and Alexa 2007] as the energy to minimize. Finally, we transfer the deformation of the skeleton template to the original CAD model by solving a linear system. We discuss the pre-processing step in section 4 and the mathematical formulation for the deformation problem in section 5. We discuss the creation of the benchmark in section 6 and evaluate our methods by comparison with the state-of-the-art methods. We further perform ablation studies to study the representation capacity using different deformation functions. Finally, we show that MeshODE benefits several downstream applications in section 7.

4 MESH REPRESENTATION

In this section, we aim at preprocessing the CAD model, so that most regions are isosceles right triangles with sufficient density to represent the deformation. Furthermore, we build virtual links and extract a uniform skeleton template as a deformation proxy for efficient optimization.

4.1 Mesh Filtering

Given a CAD model represented as a triangle mesh, we first normalize the mesh by applying a single translation and a uniform

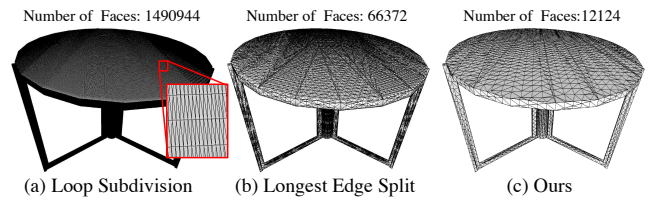

Fig. 3. Subdivision results with different approaches. Classical subdivisions result in meshes with big size and contains many irregular triangles. Our subdivision leads to fewer triangles that are less skew and more regularized.

scale to all vertices so that the bounding box of the mesh is centered at $(0.5, 0.5, 0.5)$ and the maximum length along three axes is 1. Next, we remove degenerate elements by first looping over all triangles and removing degenerate ones containing zero area. Then, we merge duplicate vertices, such that the cleaned mesh is free from degenerate vertices, edges, and faces.

Denote the cleaned mesh at this step as $\hat{\mathcal{M}} = \{\hat{\mathcal{V}}, \hat{\mathcal{E}}, \hat{\mathcal{F}}\}$ where $\hat{\mathcal{V}} = \{\hat{\mathbf{v}}_i\}$ is a set of vertices, $\hat{\mathcal{E}} = \{\hat{\mathbf{e}}_i\}$ a set of undirected edges, and $\hat{\mathcal{F}} = \{\hat{\mathbf{f}}_i\}$ a set of triangles. Specifically, $\hat{\mathbf{v}}_i$ is the 3D coordinate of the i-th vertex. $\hat{\mathbf{e}}_i =< \hat{e}_i^1, \hat{e}_i^2 >$ is set of two vertex indices of the i-th undirected edge in the mesh. $\hat{\mathbf{f}}_i =< \hat{f}_i^1, \hat{f}_i^2, \hat{f}_i^3 >$ is the set of three vertex indices of the i-th triangle in the mesh.

4.2 Mesh Subdivision

The next step involves subdividing the mesh such that the length of each edge is below a user-specified threshold: θ_l. There are multiple options for subdividing and upsampling the original mesh. For example, we can repeatedly subdivide the current longest edge and adjacent triangles until all edge lengths are below θ_l. We notice that this method tends to produce very large meshes while most new triangles subdivided from an irregular triangle are still irregular, as shown in Figure 3(a). Such an issue also holds with the standard loop

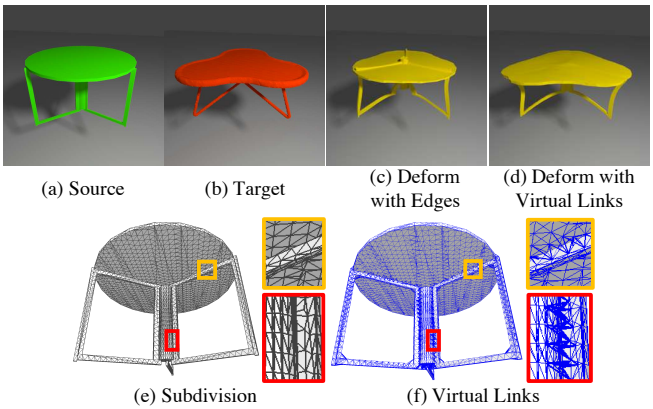

Fig. 4. Deformation directly using mesh edges breaks the geometry in (c), since CAD models do not guarantee watertight connectivities. In (d), deformation with virtual links addresses this issue. Virtual links additionally connect different table parts in (f) compared to mesh edges in (e).

subdivision strategy [Stam 1998] where at each step we subdivide each triangle into four by connecting the midpoints of the three edges (Figure 3(b)). Another important thing to consider is the edge direction. Since appealing deformations are usually considered to be ones that bend along the principal curvature, novel edges created from the subdivision should ideally follow the principal curvature. Such problems can potentially be addressed by the state-of-the-art quadrangulation methods [Huang et al. 2018b; Jakob et al. 2015]. However, such a remeshing process could change the original geometry and does not work well with disconnected components that exist in raw CAD models.

The key idea of our subdivision strategy is to upsample the mesh by sampling positions of a tangent lattice grid inside each triangle $\hat{\mathbf{f}}_i$ at the resolution of θ_l, where the lattice grid is fully determined by defining its origin as $\hat{f}_i^1$ and its x-axis as the orientation field computed from [Huang et al. 2018b]. In order to connect newly sampled points across different triangles sharing edges, we additionally subdivide each edge $\hat{\mathbf{e}}_i$ uniformly into segments such that the length of each segment is below θ_l. The number of segments can be determined as $N(\hat{\mathbf{e}}_i) = \lceil |\mathbf{v}_{\hat{e}_i^1} - \mathbf{v}_{\hat{e}_i^2}| / \theta_l \rceil$. Finally, each triangle can be remeshed by running a 2D Delaunay triangulation [Lee and Schachter 1980] algorithm to connect interior sampled grid positions and endpoints of the segments at the triangle boundary. As shown in Figure 3(c), our subdivision algorithm leads to fewer and more regular triangles. Additionally, internal edges follow principal curvatures which enable appealing bending deformations. In the following sections, we denote our subdivided mesh as $\mathcal{M} = \{\mathcal{V}, \mathcal{E}, \mathcal{F}\}$.

4.3 Virtual Links

Since CAD models created by artists do not guarantee watertight connectivity, it is common that different object parts are topologically disconnected while geometrically self-intersecting (e.g. table tops and legs). As a result, deformation based merely on topological connectivities from $\mathcal{E}$ breaks the geometry. Therefore, we need to additionally consider geometric relationships between vertices. We adopt a simple criterion that two vertices in $\mathcal{V}$ are geometrically related if their Euclidean distance is below a threshold θ_d.However, explicitly storing such geometrical relationships for all pairs of close vertices is quite inefficient. Specifically, the number of geometrical relationships is quadratic to the number of vertices in the local region, while some of them can have a large number of vertices in the CAD models.

In our implementation, we create a sparse 3D uniform voxel grid with the resolution as θ_d in a unit cube, where each non-empty voxel contains a list of vertices inside the voxel. We loop over each non-empty voxel, collect the vertices from this voxel and its subsequent three voxels along the three axes. We run a 3D Delaunay triangulation to connect the vertices and keep the edges whose lengths are smaller than θ_d. Such a formulation ensures the number of constraints is with $O(N)$ complexity in the local region. We collect all these edges together with $\mathcal{E}$ as a set of virtual links $\mathcal{L}$. We use the virtual links for enforcing the rigidity constraints, such that the deformation guarantees local geometrical feature preservation without a requirement for watertight manifold topology.This is illustrated in the comparison between Figure 4(c) and (d). Virtual links visualized in Figure 4(f) additionally connect different table parts in addition to mesh edges, as can be seen in Figure 4(e).

4.4 Uniform Skeleton Template

It is sufficient to solve the deformation for $\mathcal{V}$ with given constraints $\mathcal{L}$ such that the deformed mesh can be represented as the updated vertex positions connected by the triangles $\mathcal{F}$. However,though the upper bound of edge lengths through the previous steps can be guaranteed, the distribution of vertices is dense and non-uniform, rendering the optimization process inefficient. Short edges can also cause numerical issues and thereby require small time steps during optimization.

Therefore, we additionally build a uniform skeleton template as a graph to approximate the original mesh graph. Again, we create a sparse 3D uniform voxel grid at the resolution of θ_g and for each voxel we collect the vertices from $\mathcal{V}$ that belong to it. For each non-empty voxel, we create a node as the average of the vertices within. We define the set of created nodes as $\mathcal{V}_G$. For each virtual link in $\mathcal{L}$, we create a graph edge connecting to the corresponding node that vertices of the link belong to. Note that several virtual links correspond to the same edge in the graph template, and a single edge is kept for these links. We define the set of graph edges as $\mathcal{E}_G$. Finally, for the i-th graph node $\mathbf{v}_i \in \mathcal{V}_G$, we store the set of vertex indices in the original mesh $\mathcal{V}$ as s_i, and define the collection of s_i as $\mathcal{S}$. We denote such a uniform skeleton template as $\mathcal{G} = \{\tilde{\mathcal{V}}, \tilde{\mathcal{E}}, \mathcal{S}\}$.

5 MESH DEFORMATION

We aim at solving for the deformation map between a pair of two meshes A and B, preprocessed via the proposed approach in Sec. 4, that includes the subdivided vertices, triangles, virtual links and skeleton template. Specifically, we denote the two meshes as $\mathcal{M}_A = \{\mathcal{V}_A, \mathcal{F}_A, \mathcal{L}_A, \mathcal{G}_A\}$ and $\mathcal{M}_B = \{\mathcal{V}_B, \mathcal{F}_B, \mathcal{L}_B, \mathcal{G}_B\}$, where the skeleton templates are $\mathcal{G}_A = \{\tilde{\mathcal{V}}_A, \tilde{\mathcal{E}}_A, \mathcal{S}_A\}$ and $\mathcal{G}_B = \{\tilde{\mathcal{V}}_B, \tilde{\mathcal{E}}_B, \mathcal{S}_B\}$.

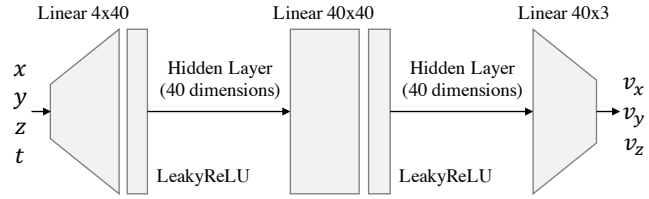

Fig. 5. Neural network architecture for velocity field prediction.

5.1 Energy Minimization

We solve for the deformation in two steps. First, we deform the source skeleton template to the target shape by minimizing fitting and rigidity energies. Second, we solve a linear system to transfer the deformation of the skeleton template to the source mesh.

Suppose shape A is the source and B is the target, the energy is as follows:

$$E(\mathcal{M}_A, \mathcal{M}_B, \mathcal{D}) = E_D(\mathcal{D}(\mathcal{G}_A), \mathcal{M}_B) + \lambda \cdot E_R(\mathcal{D}(\tilde{\mathcal{V}}_A), \tilde{\mathcal{V}}_A, \tilde{\mathcal{E}}_A). \quad (1)$$

The energy contains the fitting loss E_D (Section 5.3) between the deformed skeleton template $\mathcal{D}(\tilde{\mathcal{G}}_A)$ and the target mesh $\mathcal{M}_B$, and the rigidity loss E_R (Section 5.4) of the deformed skeleton template $\mathcal{D}(\tilde{\mathcal{G}}_A)$, given the source template $\tilde{\mathcal{G}}_A$.

Furthermore, in the case of solving for the bijective deformation mapping between A and B such that either of them can be the source or target, we require $\mathcal{D}$ to be a bijective mapping and propose the the two-way deformation energy to be:

$$E_2(\mathcal{M}_A, \mathcal{M}_B, \mathcal{D}) = E(\mathcal{M}_A, \mathcal{M}_B, \mathcal{D}) + E(\mathcal{M}_B, \mathcal{M}_A, \mathcal{D}^{-1}). \quad (2)$$

In either scenario, we optimize $\mathcal{D}$ to minimize the energy, and obtain the deformed skeleton $\mathcal{D}(\mathcal{G}_A)$ and optionally $\mathcal{D}^{-1}(\mathcal{G}_B)$. In the post processing step, we solve a linear system to apply the deformation to the original mesh (Section 5.5).

5.2 Deformation Function

The deformation function $\mathcal{D} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ maps every point in the source shape to the deformed shape.

First, our framework can incorporate traditional methods [Sorkine and Alexa 2007] and explicitly represent $\mathcal{D}$ as the set of sampled vertex coordinates in the deformed mesh, which we denote as $\mathcal{D}_e$, in which case the variables we are optimizing are the corresponding locations of these vertices in the deformed source skeleton, initialized as $\tilde{\mathcal{V}}_A$). Using the mesh preprocessing pipeline (detailed in Section 4), such an optimization process effectively preserves local geometry details for complex CAD models. However, we can only use such a functions for the A-to-B deformation problem by minimizing Equation 1, since there is no guarantee that $\mathcal{D}_e$ is invertible.

To find an invertible deformation function by minimizing the energy in Equation 2, the key is to find a parameterized function $\mathcal{D}$ which is flexible to handle complex deformation while ensuring invertibility. Neural ODE [Chen et al. 2018] satisfies both of these requirements. We first define the velocity field $\mathbf{u}(\mathbf{p}, t)$ given a 3D location $\mathbf{p}$ and time t. The field is parameterized by a neural network architecture shown in Figure 5. Then, we create the deformation path $\mathbf{p}(\mathbf{x}, t)$ for each source point $\mathbf{x}$ where instant velocity at time t is $\mathbf{u}(\mathbf{p}, t)$. $\mathcal{D}_{\text{ode}}$ maps all source points at time 0 to deformed points at time 1. The mathematical definition of $\mathcal{D}_{\text{ode}}$ is shown in Equation 3, where $\mathbf{u}$ is considered as the parameters of $\mathcal{D}_{\text{ode}}$.

$$\begin{aligned} \mathbf{p}(\mathbf{x}, 0) &= \mathbf{x} \\ \frac{d\mathbf{p}}{dt} &= \mathbf{u}(\mathbf{p}, t) \\ \mathcal{D}_{\text{ode}}(\mathbf{x}; \mathbf{u}) &= \mathbf{p}(\mathbf{x}, 1). \end{aligned} \quad (3)$$

The inverse function $\mathcal{D}_{\text{ode}}^{-1}$ can be inferred as

$$\begin{aligned} \mathcal{D}_{\text{ode}}^{-1}(\mathbf{x}; \mathbf{u}) &= \mathcal{D}_{\text{ode}}(\mathbf{x}; \mathbf{w}) \\ \mathbf{w}(*, t) &= -\mathbf{u}(*, 1 - t) \end{aligned} \quad (4)$$

We use the implementation provided by Neural ODE [Chen et al. 2018] to integrate $\mathbf{u}$ and obtain $\mathcal{D}_{\text{ode}}$ and $\mathcal{D}_{\text{ode}}^{-1}$ in a differentiable way. This allows us to optimize the parameters in $\mathbf{u}$ to achieve optimal deformation. In Section 7.1, we show smooth intermediate deformation results as an animation by computing $\mathbf{p}(\mathcal{V}_A, t)$ at different time $t \in [0, 1]$.

5.3 Fitting Loss

We consider the fitting loss between the deformed source shape and the target shape as the sum of the squared distances between vertices in the deformed source and the respective nearest vertex in target, which is similar to the ICP algorithm [Besl and McKay 1992]. Specifically, the fitting loss between a deformed skeleton $\mathcal{G}_A^* = \{\tilde{\mathcal{V}}_A^*, \tilde{\mathcal{E}}_A, \mathcal{S}_A\}$ and the target mesh $\mathcal{M}_B$ is defined as:

$$E_D^a(\mathcal{G}_A^*, \mathcal{M}_B) = \sum_{\mathbf{v}_i^* \in \tilde{\mathcal{V}}_A^*} \min_{\mathbf{p}_i \in \mathcal{M}_B} ||\mathbf{v}_i^* - \mathbf{p}_i||_2^2. \quad (5)$$

Since the target mesh $\mathcal{M}$ is a fixed shape, we precompute a distance field as a 64^3 uniform grid in a unit cube for fast retrieval of the nearest distance. By optimizing Equation 5, we are able to obtain the deformation such that vertices in the deformed source skeleton are well-covered by the target mesh. Since we additionally require the target mesh to be covered by the source skeleton, we define a fitting loss between the target skeleton vertices $\tilde{\mathcal{V}}_B$ of the mesh $\mathcal{M}_B$ and the deformed skeleton $\mathcal{G}_A^*$ as:

$$E_D^b(\mathcal{G}_A^*, \mathcal{M}_B) = \sum_{\mathbf{v}_i \in \tilde{\mathcal{V}}_B} \min_{\mathbf{p}_i^* \in \mathcal{G}_A} ||\mathbf{v}_i - \mathbf{p}_i^*||_2^2. \quad (6)$$

Since the source skeleton vertices are changed during optimization, we rebuild a KDtree for the deformed skeleton at every optimization step for efficiently querying the nearest distance between the target and the deformed source skeleton.

We define $E_D = E_D^a$ as a baseline ("ours-1way") in Section 6.1, and define $E_D = E_D^b$ for scan-to-cad fitting in Section 7.2 since it is unreasonable to cover all regions of the CAD by a partial scan. For all other experiments, we optimize the two-way fitting loss as $E_D = E_D^a + E_D^b$.

5.4 Rigidity Loss

Our rigidity loss E_R follows the definition in [Sorkine and Alexa 2007]. Specifically E_R measures the local distortions of the deformed point set $\mathcal{V}^*$ from the original point set $\mathcal{V}$ by aggregating the neighborhood information from an edge set $\mathcal{E}$. Since the edges are

undirected, $< j, i >$ is equivalent to $< i, j >$. The rigidity energy E_R is defined in Equation 7.

$$E_R(\mathcal{V}^*, \mathcal{V}, \mathcal{E}) = \min_{\{\mathbf{R}_i\}} \sum_{<i,j> \in \mathcal{E}} ||(\mathbf{v}_i^* - \mathbf{v}_j^*) - \mathbf{R}_i(\mathbf{v}_i - \mathbf{v}_j)||_2^2 \quad (7)$$

In order to optimize the rigidity loss, we use a three-channel Euler angle for each vertex i to represent its local rotation $\mathbf{R}_i$, and initialize it to be zero. These Euler angles are jointly optimized with the deformation function parameters.

5.5 Post processing

Once we obtain the deformation of the skeleton template, we can apply the deformation to the original mesh. First, we estimate the local rotation of the local region for each vertex as the optimized rotation of the nearest skeleton node. For $\mathcal{M} = \{\mathcal{V}, \mathcal{F}, \mathcal{L}, \mathcal{G}\}$, we deform vertices $\mathcal{V}$ to $\mathcal{V}^*$ to minimize the following energy:

$$E_L(\mathbf{V}^*; \mathcal{M}, \mathcal{D}) = \sum_i ||\mathcal{D}(\tilde{\mathbf{v}}_i) - \frac{1}{|s_i|}(\sum_{j \in s_i} \mathbf{v}_j^*)||_2^2 + \lambda \sum_{<i,j> \in \mathcal{L}} ||(\mathbf{v}_i^* - \mathbf{v}_j^*) - \mathbf{R}_i(\mathbf{v}_i - \mathbf{v}_j)||_2^2 \quad (8)$$

The first term requires that the average of vertices in s_i that are close to the i-th skeleton node is close to the deformed node after the deformation. The second term is the rigidity loss that the deformed edge offset should be consistent with the estimated local rotation of the vertices (similar to Equation 7). Since E_L is quadratic to $\mathcal{V}^*$, we can solve the massive number of vertices in the original CAD models efficiently with a linear solver.

5.6 Implementation details

We set $\theta_l = 0.02$ for subdivision, $\theta_d = 0.015$ for building virtual links, and $\theta_g = 0.01$ as the resolution of the skeleton node. We set $\lambda = 1$ for Equation 7 when optimizing $\mathcal{D}_E$ and $\lambda = 0.1$ when optimizing $\mathcal{D}_{\text{ODE}}$. We adopt weaker rigidity loss since $\mathcal{D}_{\text{ODE}}$ is intrinsically smooth and easier to preserve shape features.

We solve $\mathcal{D}_E$ with the efficient Levenberg-Marquardt algorithm [Moré 1978] implemented in the Ceres Solver[2]. We solve parameters in $\mathcal{D}_{\text{ODE}}$ with an Adam solver [Kingma and Ba 2014] implemented in PyTorch[3]. In the next section, we evaluate the difference between all three versions and compare them with the state-of-the-art methods in a controlled experiment.

6 EVALUATION

We create our benchmark dataset by randomly selecting 3625 pairs of shapes as sources and targets from ShapeNet. Our pairs are selected from the chair, the table, and the sofa categories and we guarantee that the two shapes from each pair belong to the same category. In Section 6.1, we evaluate our deformation framework and compare them with the state-of-the-art methods. We additionally do ablation studies for different mesh representations in Section 6.2 and different deformation functions in Section 6.3.

	Chair	Table	Sofa	All
Source	6.232/6.232	6.580/6.580	4.805/4.805	6.155/6.155
NeuralCage	5.781/7.396	5.760/21.36	3.542/29.60	5.381/17.64
3DN	3.000/7.523	3.325/10.843	2.952/9.482	3.489/10.44
Ours-1way	2.914/2.914	2.990/2.990	2.622/2.622	2.901/2.901
Ours-2way	2.053/2.053	2.073/2.073	1.990/1.990	2.052/2.052
Ours-neural	**1.136/1.136**	**1.297/1.297**	**1.092/1.092**	**1.203/1.203**

Table 1. Evaluation using the two-way Chamfer distance ($\times 10^{-2}$). For each entry, we report two numbers, one where the shapes are canonically aligned with the training data and the other with a 90-degree rotation for both the source and the target around the the z-axis.

NeuralCage	3DN	Ours-1way	Ours-2way	Ours-neural
1.17/4.81	0.58/2.12	-/1.24	-/3.79	3.73/3.96

Table 2. Average time (seconds) used to process each pair of shapes on a server with 8 Titan X GPUs or 32 2.20GHz CPUs. For NeuralCage, 3DN and ours-neural, we report two numbers, running it with or without GPUs.

6.1 Comparison of Deformation Quality

We compare our framework with the state-of-the-art methods based on the two-way Chamfer distance between the deformed source model and the target model The state-of-the-art methods include 3DN [Wang et al. 2019] and NeuralCage [Yifan et al. 2019]. Our framework enables valid CAD deformation by optimizing Equation 1 using $\mathcal{D}_E$ with traditional one-way ($E_D = E_D^a$) or two-way ($E_D = E_D^a + E_D^b$) squared Chamfer loss, and our newly proposed neural deformation $\mathcal{D}_{\text{ODE}}$ with two-way squared Chamfer loss. They are shortened as "ours-1way", "ours-2way" and "ours-neural" respectively. Note that all these experiments are performed under a single-direction scenario (Equation 1), while we additionally discuss bidirectional optimization (Equation 2) in Section 6.5 and utilize use it in Section 7.1.

In Table 1, we report two numbers for each entry, one with the shape canonically aligned with the training data and the other with a 90-degree rotation for both the source and the target around the z-axis. All three versions of ours significantly outperform the existing state-of-the-art methods according to the fitting error. Further, the drop in performance for the deep-learning-based baselines show that they do not generalize to novel shapes or poses outside the training regime with a simple rotation variation. Visual comparisons are shown in Figure 6: Our method achieves better fitting quality and visual appeal compared to 3DN [Wang et al. 2019] and NeuralCage [Yifan et al. 2019]. Ours-1way is not optimized for the coverage of the target, while ours-2way ensures coverage of both the deformed and the target shape. Ours-neural additionally provides more flexible deformation with better fitting quality. Based on the experiments, our optimization-based methods significantly outperform the existing learning-based methods with respect to the fitting errors, and our novel neural function achieves the best performance.

[2]http://ceres-solver.org
[3]https://pytorch.org

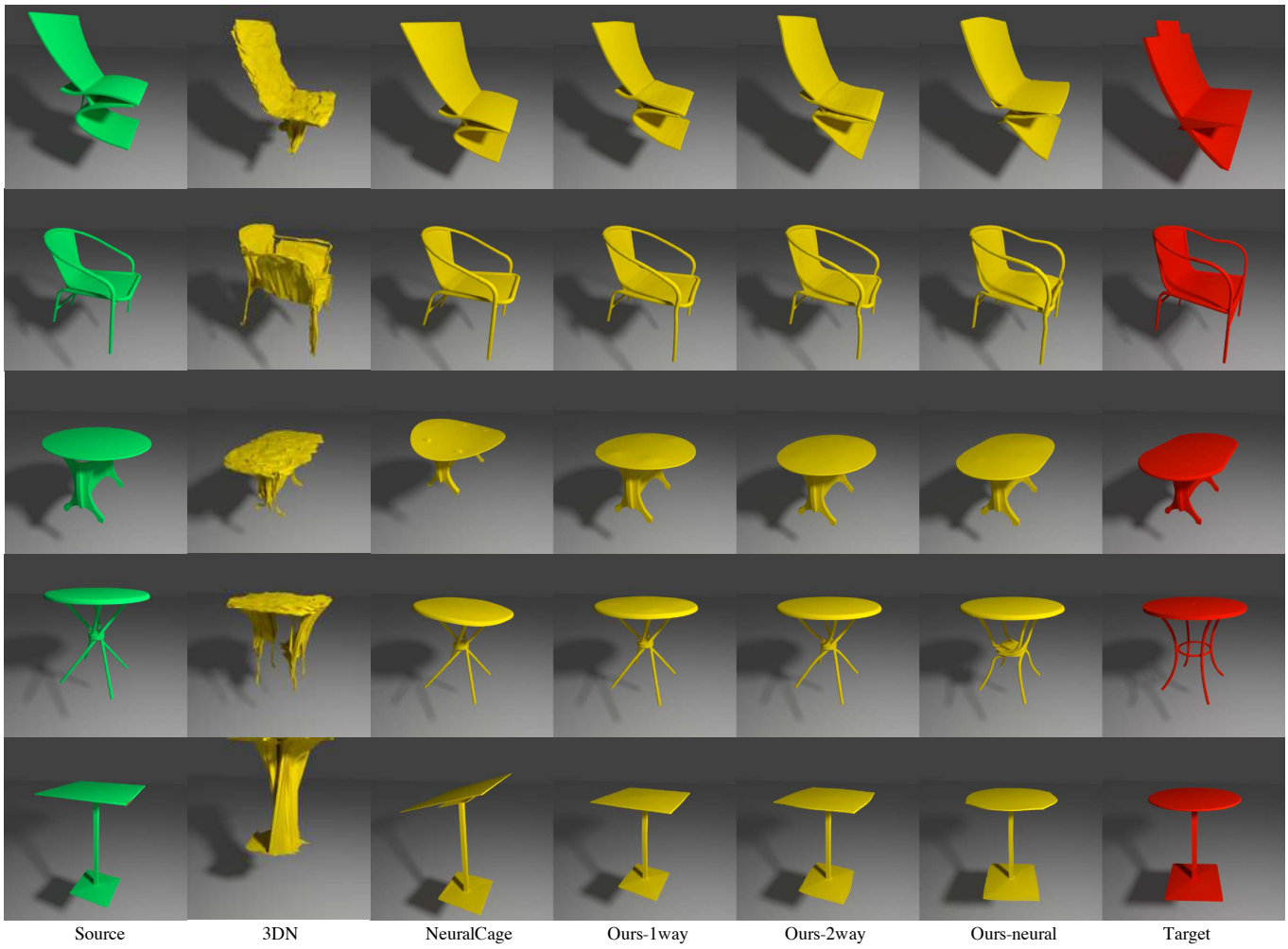

Fig. 6. Visual comparison on shape deformation between different approaches. 3DN [Wang et al. 2019] and NeuralCage [Yifan et al. 2019] do not fit the target as good as our framework do. Ours-1way is not optimized for the coverage of the target, while ours-2way ensures coverage of both the deformed and the target shape. Ours-neural also ensures the coverage given by the bijective mapping.

Efficiency is another important factor to consider. We process all models on a server with 8 Titan X GPUs and 32 2.20GHz CPUs. The average processing time for each pair of shapes for different methods is shown in Table 2. Regarding efficiency, we find that our optimization based method: $\mathcal{D}_E$ is on par with the forward pass of a trained network (NeuralCage) which relies on GPUs to do efficient computation with a big network. Our NeuralODE-based optimization uses a significantly more lightweight neural network which is easy to optimize. The gap between GPU and CPU of our NeuralODE is small, due to our reliance on a CPU implementation of Equation 1. This suggests that our GPU version has the potential to be further accelerated. Finally, we directly obtain a bijective and smooth intermediate deformation animation (Section 7.1) with infinite time resolution once our single-pass optimization is done.

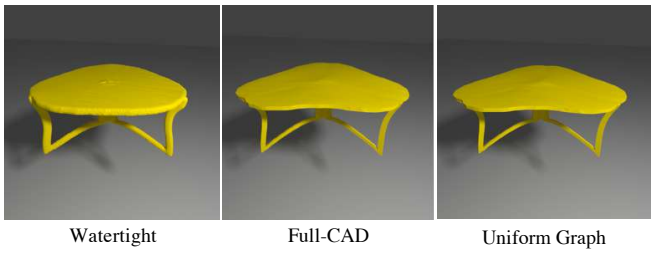

Fig. 7. Deformation results with different mesh representations. Although watertight approximation [Huang et al. 2018a] can achieve good fitting to the target, the geometry is different and sharp features can not be well-preserved. Our skeleton template representation is a good approximation which generates quite similar results comparing with direct optimization of the original CAD model.

	Loop	Longest	Our-CAD	Watertight	Skeleton
Error ($\times 10^{-2}$)	2.043	2.286	1.925	1.823	2.052
Time (s)	141.7	51.6	7.14	5.73	3.79

Table 3. Performance comparison for different mesh representations. "Loop" stands for loop subdivision [Stam 1998]. "Longest" stands for longest-edge subdivision. "Our-CAD" uses our subdivision method. "Watertight" uses [Huang et al. 2018a] to convert the CAD model into a watertight shape. "Skeleton" uses our uniform skeleton approximation based on our subdivision strategy.

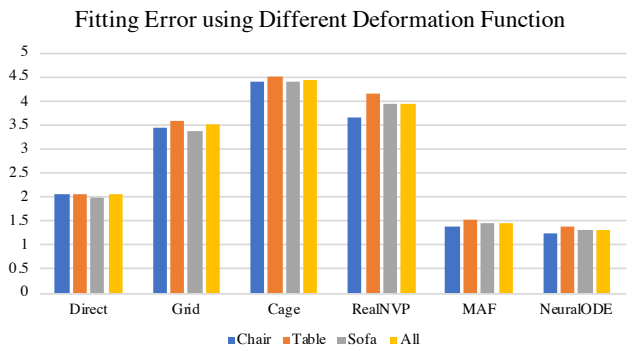

Fig. 8. Representation capacity using different deformation functions. We optimize Equation 1 for all deformation functions. We set rigidity loss as 1 for $\mathcal{D}_e$ and $\mathcal{D}_{\text{grid}}$ and 0.1 for other methods. $\mathcal{D}_{\text{ode}}$ has the closest representation capacity to $\mathcal{D}_e$ and require small rigidity and thus achieves the best results comparing to other neural-network based methods. $\mathcal{D}_e$ and regular grid requires a big rigidity loss to avoid unpleasant distortion, which also limits their representation capacities. Cage-based deformation is more global and is not suitable for aligning subtle geometry structures.

6.2 Choice of Mesh Representation

In this ablation study, we investigate the deformation performance based on different mesh representations obtained through different mesh preprocessing pipelines. We optimize Equation 1 for all variants using our proposed method and report the fitting error and timing in Table 3. Direct optimization of the original CAD model pre-processed using our subdivision method achieves the minimum fitting error. Other subdivision methods achieve slightly worse fitting quality because of the irregularity of the triangles. Although the watertight approximation [Huang et al. 2018a] achieves good fitting to the target, the geometry is different and the sharp features can not be well-preserved, as shown in figure 7. The skeleton template representation achieves similar fitting error comparing to other subdivision methods. Efficiency-wise, deformation based on our subdivision is much faster than existing ones. Our skeleton template further improves efficiency. To balance quality and efficiency, we combine our subdivision method and the skeleton template for mesh representation.

6.3 Choice of Deformation Function

In this ablation study, we study the impacts of different choices of deformation functions on fitting errors. For traditional deformation functions, we test $\mathcal{D}_e$ along with two common choices for parameterizing the deformation space, including uniform grids or control cages. We use the 10^3 grids with 0.1 resolution and the convex hull of the source shape as the control cage. For learned bijective mapping, we explore the usage of several flow-based models including RealNVP [Dinh et al. 2016], MAF [Papamakarios et al. 2017], and our $\mathcal{D}_{\text{ode}}$.

To test the representation capacity, we optimize Equation 1 for all deformation functions. Based on Figure 8, deformation grids and cages have limited representation capacity compared to $\mathcal{D}_e$. $\mathcal{D}_{\text{ode}}$ is the best choice among flow-based models and achieves better fitting quality by requiring less explicit rigidity constraints.

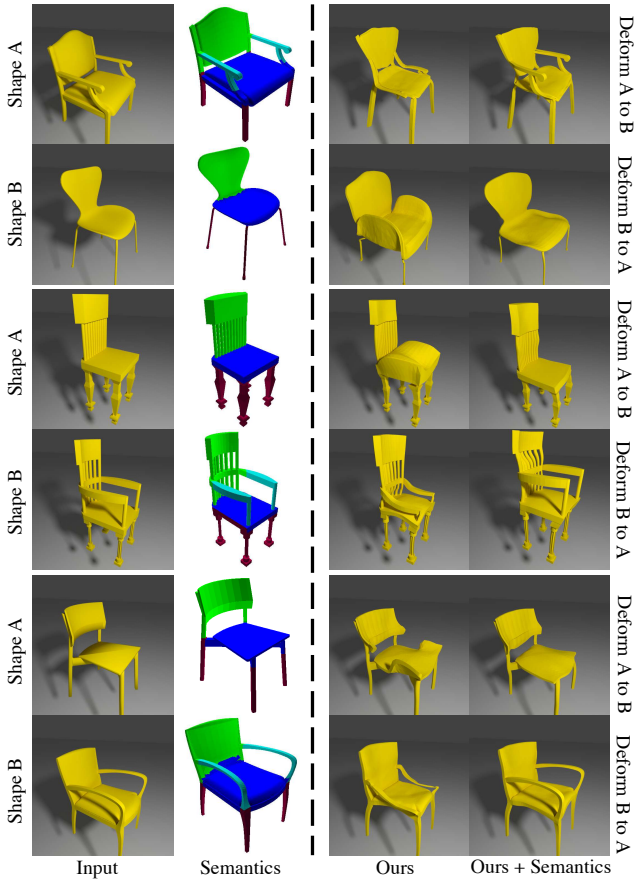

Fig. 9. Part-aware deformation. Each example contains two rows as a pair of deformations. Our input are meshes (first column) with part semantics predicted by PointNet [Qi et al. 2017] (second column). The third column is the pairwise deformation with our NeuralODE by optimizing the pure geometry fitting loss using Equation 2. We can obtain semantically meaningful deformation (fourth column) by measuring fitting loss separately for different semantic parts.

6.4 Part-aware Deformation

Furthermore, we explore the use of additional semantic information in our framework during pairwise shape deformations. Our inputs

Fig. 10. Schematics for the deformation of points on a tabletop. If the deformation maps two different points A and B to the same location and causes self-intersections, such deformation would violate the property of bijection, therefore bijective mappings are free of self-intersections.

are pairs of chairs with semantic segmentation labels produced using PointNet [Qi et al. 2017]. We separately measure the fitting loss (Section 5.3) for regions with different semantic labels and compute the sum of them as the total fitting loss. For certain semantic parts of one object that does not exist in another, we set the fitting loss as zero. In Figure 9, we show that such a modification can apply semantically meaningful deformations (e.g., chair handles are not deformed to seats).

6.5 Properties of MeshODE

We derive and evaluate several properties of MeshODE.

Bijection. If $\mathcal{D}$ is a bijective mapping, we expect it deforms any point $\mathbf{p}$ to a new location, while its inverse maps it back to the same point. We measure quality of bijection for a point $\mathbf{p}$ as

$$\epsilon(\mathbf{p}) = ||\mathbf{p} - \mathcal{D}^{-1}(\mathcal{D}(\mathbf{p}))|| \tag{9}$$

From our numerical evaluation, the maximum and the average of $\epsilon(\mathbf{p})$ for all mesh vertices in the dataset are 3×10^{-5} and 6×10^{-7} respectively, compared to the mesh vertex coordinate range from 0 to 1. Therefore, the numerical errors do not affect the bijectivity of our model.

Free of Self-Intersection. An intuition based illustration of proof by contradiction is given in Figure 10. Any deformation achieved through a bijective map does not induce self-interesction of any two points in the original space. This is important for dealing with thin structures (e.g. tabletop) without enforcing strong rigidity constraints.

Bidirectional Fitting. Optimization of the single-direction energy (Equation 1) targets a deformation that minimizes the distance between deformed shape A to shape B. In certain scenarios, we want to optimize for bidirectional deformation (Equation 2) so that either shape can be deformed to fit the other. In Figure 11, we evaluate the fitting loss from deformed shape A to B by plotting Equation 1 comparing two different optimizations using Equation 1 and 2. It is expected that optimization with Equation 1 achieves lower losses since the objective function is exactly the evaluation metric. However, slightly surprisingly, optimization with Equation 2 achieves a similar fitting quality, suggesting that bidirectional fitting optimization will not affect the fitting quality from either direction. This is due to the intrinsic bijection property of our deformation function.

7 APPLICATION

In this section, we demonstrate several applications that can benefit from our framework, including novel shape creation, animation,

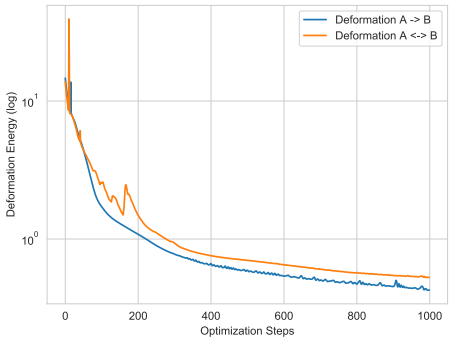

Fig. 11. We compare two optimizations using Equation 1 and 2, optimizing for single-directional and bi-directional fitting. We evaluate the fitting loss from deformed shape A to B by plotting Equation 1. While optimization with Equation 1 is expected to achieve better fitting quality, optimization with Equation 2 achieves similar performance.

	Chair	Table	Sofa	All
Human	1.715	2.832	1.947	2.119
AE	1.856	2.782	2.142	2.200
Uy *et al.*	1.706	2.813	1.570	2.069
$\mathcal{D}_{\text{ode}}$(Human)	1.343	1.177	0.845	1.236
$\mathcal{D}_{\text{ode}}$(AE)	1.401	1.104	1.051	1.264
$\mathcal{D}_{\text{ode}}$(Uy *et al.*)	1.402	1.173	1.002	1.283

Table 4. we report the fitting error of the deformed CAD model to the scan using different retrieval methods proposed in [Uy et al. 2020] including human retrieval (Human), nearest neighbor under autoencoder (AE), and their own proposed method (Uy *et al.*). The first three rows use the deformation function in [Uy et al. 2020] as baselines, and the last three are deformed using our method.

scan-to-CAD fitting, and texture reconstruction. For the shape animation and design, our ODE-based bijective function $\mathcal{D}_{\text{ode}}$ creates bidirectional deformations with smooth intermediate steps (Equation 2). For scan-to-CAD fitting and texture reconstruction, we choose to optimize for Equation 1. More results can be found in the supplemental material.

7.1 Shape Animation and Design

By optimizing the smooth deformation function $\mathcal{D}_{\text{ode}}$, we are able to obtain a time-dependent velocity fields $\mathbf{u}(\mathbf{p}, t)$. We can create intermediate steps of the deformation $\mathbf{p}(\mathbf{x}, t_i)$ as an animation, where $\mathbf{x} \in \mathcal{V}_{\text{src}}$ and t_i are uniformly and densely sampled in $[0, 1]$. This additionally allows the designer to interactively create new shapes by deciding how close the shape is to the target using the style of the source shape by specifying a single interpolation scalar t. Examples of such a novel shape creation process are shown in figure 12, where for every two rows our input is the two shapes shown in green and red, and yellow shapes are intermediate results via deformation.

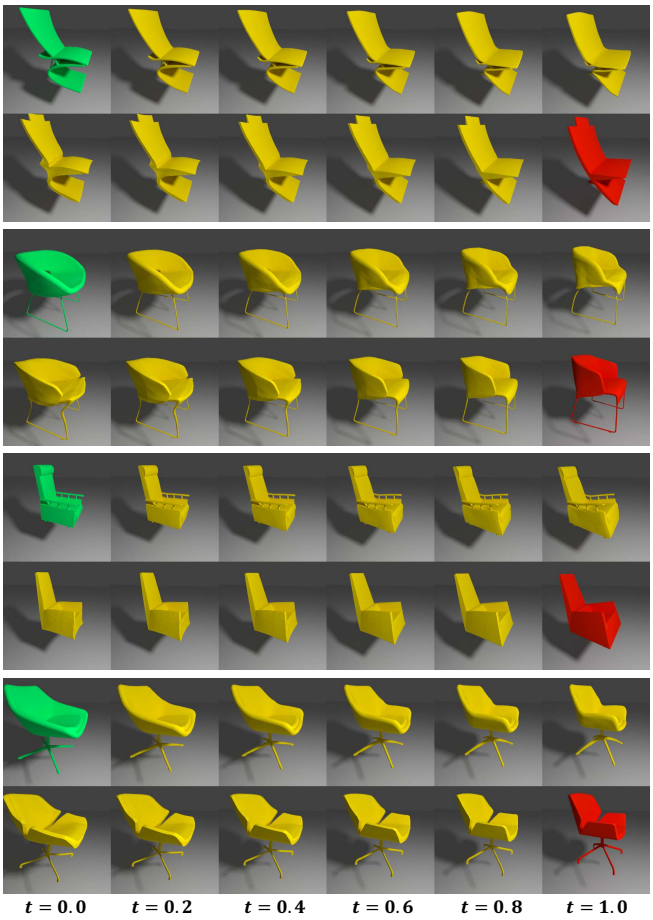

Fig. 12. Bidirectional shape animation. For every two rows, our input is the two shapes shown in green and red. Each shape can be deformed to the other with reasonable intermediate steps. Designers can interactively create novel shapes by controlling t with our bijective smooth function.

Fig. 13. Fitting errors between retrieved/deformed CAD models and scans in Scan2CAD [Avetisyan et al. 2019a]. Our deformation produces results that are more visually appealing with smaller fitting errors, compared to the state-of-the-art.

7.2 Scan-to-CAD

By setting the scanned mesh as the target and the CAD model as the source, we can deform the CAD to fit the scan. This is a promising direction to fix the scanning geometry as explored by [Avetisyan et al. 2019a,b; Dahnert et al. 2019; Uy et al. 2020]. Using our deformation that minimizes Equation 1, the CAD model can better fit the scanning geometry. In Table 4, we report the fitting error between the deformed CAD model and the scan using different retrieval methods proposed in [Uy et al. 2020] including human retrieval (Human), nearest neighbor under autoencoder (AE), the retrieval method in Uy *et al.*. The first three rows are use the deformation function in [Uy et al. 2020] as a baseline, and the last three are deformed using our method. As a result, our deformation produces better results with respect to the fitting error based on different retrieval methods. Figure 13 shows some visual comparisons, where we produce deformed source shapes with better fitting quality to scans.

7.3 Texture Optimization

Finally, we demonstrate that our deformation framework can help with transferring scanned textures to scanned CAD models that are aligned with the scans. This can be used for enhancing the appearance of the CAD fitting to the scans, as proposed in [Huang et al. 2020].

With good deformation, the texture is easier to optimize from observed images to the fitted CAD model, since our deformation enables better alignment and therefore less geometry alignment errors. Specifically, we optimize based on the two-way fitting (Equation 1) to deform the CAD to the scanned geometry and optimize the texture from images to the geometry according to to [Huang et al. 2020]. Compared to the result from [Huang et al. 2020] without any deformation, the texture appearance on our deformed CAD model has fewer artifacts, as shown in Figure 14. Such improvement mostly happen at the corners or object boundaries, where severely misaligned regions cannot be easily address by pure texture optimization.

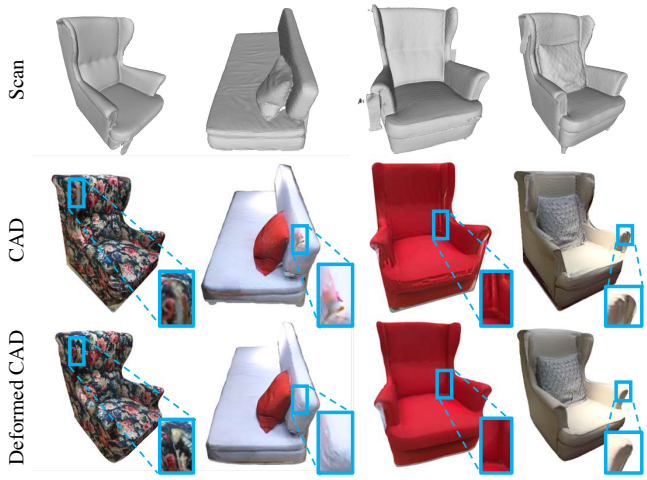

Fig. 14. Texture optimization comparisoned with [Huang et al. 2020]. Texture can be better optimized with our deformed CAD model since it fits better to the scan and has better geometry alignment with the images for the scan.

8 CONCLUSION

We develop a robust and scalable framework for CAD deformation. Our key contribution is a novel neural deformation function that enables bi-directional deformation between two shapes with continuous intermediate steps. Our mesh representation enables valid deformation with complex CAD model structures and our results significantly outperform existing state-of-the-art methods. Our deformation framework enables several applications related to shape animation and design, scanning, and texture reconstruction.

Our method can be further extended in several future directions. A multi-grid optimization could further improve the deformation in a global-local manner with better convergence. It is possible to enforce a divergence-free velocity field by applying a curl operator. This will be essential for applications that require volume-preserving deformation, e.g., human animation. Finally, it is possible to modify the velocity field of the neural function to accept additional parameters to describe deformations between multiple models and encode the models into a deformation space.

REFERENCES

Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. 2005. SCAPE: shape completion and animation of people. In *ACM SIGGRAPH 2005 Papers*. 408–416.

Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X Chang, and Matthias Nießner. 2019a. Scan2cad: Learning cad model alignment in rgb-d scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2614–2623.

Armen Avetisyan, Angela Dai, and Matthias Nießner. 2019b. End-to-End CAD Model Retrieval and 9DoF Alignment in 3D Scans. In *Proceedings of the IEEE International Conference on Computer Vision*. 2551–2560.

Paul J Besl and Neil D McKay. 1992. Method for registration of 3-D shapes. , 586–606 pages.

Stéphane Calderon and Tamy Boubekeur. 2017. Bounding proxies for shape approximation. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–13.

Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. 2002. Interactive skeleton-driven dynamic deformations. *ACM transactions on graphics (TOG)* 21, 3 (2002), 586–593.

Edwin Catmull and James Clark. 1978. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-aided design* 10, 6 (1978), 350–355.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. 2015. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012* (2015).

Isaac Chao, Ulrich Pinkall, Patrick Sanan, and Peter Schröder. 2010. A simple geometric model for elastic deformations. *ACM transactions on graphics (TOG)* 29, 4 (2010), 1–6.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural ordinary differential equations. In *Advances in neural information processing systems*. 6571–6583.

Yang Chen and Gérard G Medioni. 1992. Object modeling by registration of multiple range images. *Image Vision Comput.* 10, 3 (1992), 145–155.

Lei Chu, Hao Pan, Yang Liu, and Wenping Wang. 2019. Repairing man-made meshes via visual driven global optimization with minimum intrusion. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 158.

Manuel Dahnert, Angela Dai, Leonidas J Guibas, and Matthias Nießner. 2019. Joint Embedding of 3D Scan and CAD Objects. In *Proceedings of the IEEE International Conference on Computer Vision*. 8749–8758.

Yago Díez, Ferran Roure, Xavier Lladó, and Joaquim Salvi. 2015. A qualitative review on 3D coarse registration methods. *ACM Computing Surveys (CSUR)* 47, 3 (2015), 1–36.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803* (2016).

Daniel Doo. 1978. A subdivision algorithm for smoothing down irregularly shaped polyhederons. *Computer Aided Design* (1978), 157–165.

Nira Dyn, David Levine, and John A Gregory. 1990. A butterfly subdivision scheme for surface interpolation with tension control. *ACM transactions on Graphics (TOG)* 9, 2 (1990), 160–169.

Andrew W Fitzgibbon. 2003. Robust registration of 2D and 3D point sets. *Image and vision computing* 21, 13-14 (2003), 1145–1153.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. 2018. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367* (2018).

Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. 2018. 3d-coded: 3d correspondences by deep deformation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 230–246.

Rana Hanocka, Noa Fish, Zhenhua Wang, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. 2018. Alignet: Partial-shape agnostic alignment via unsupervised learning. *ACM Transactions on Graphics (TOG)* 38, 1 (2018), 1–14.

Jingwei Huang, Zhili Chen, Duygu Ceylan, and Hailin Jin. 2017. 6-DOF VR videos with a single 360-camera. In *2017 IEEE Virtual Reality (VR)*. IEEE, 37–44.

Jingwei Huang, Hao Su, and Leonidas Guibas. 2018a. Robust watertight manifold surface generation method for shapenet models. *arXiv preprint arXiv:1802.01698* (2018).

Jingwei Huang, Justus Thies, Angela Dai, Abhijit Kundu, Chiyu Max Jiang, Leonidas Guibas, Matthias Nießner, and Thomas Funkhouser. 2020. Adversarial Texture Optimization from RGB-D Scans. *arXiv preprint arXiv:2003.08400* (2020).

Jingwei Huang, Yichao Zhou, Matthias Niessner, Jonathan Richard Shewchuk, and Leonidas J Guibas. 2018b. Quadriflow: A scalable and robust method for quadrangulation. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 147–160.

Qi-Xing Huang, Bart Adams, Martin Wicke, and Leonidas J Guibas. 2008. Non-rigid registration under isometric deformations. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 1449–1457.

Dominic Jack, Jhony K Pontes, Sridha Sridharan, Clinton Fookes, Sareh Shirazi, Frederic Maire, and Anders Eriksson. 2018. Learning free-form deformations for 3d object reconstruction. In *Asian Conference on Computer Vision*. Springer, 317–333.

Alec Jacobson and Olga Sorkine. 2011. Stretchable and twistable bones for skeletal shape deformation. In *Proceedings of the 2011 SIGGRAPH Asia Conference*. 1–8.

Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. 2015. Instant field-aligned meshes. *ACM Trans. Graph.* 34, 6 (2015), 189–1.

Tao Jiang, Kun Qian, Shuang Liu, Jing Wang, Xiaosong Yang, and Jianjun Zhang. 2017. Consistent as-similar-as-possible non-isometric surface registration. *The Visual Computer* 33, 6-8 (2017), 891–901.

Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. 2007. Harmonic coordinates for character articulation. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 71–es.

Tao Ju, Scott Schaefer, and Joe Warren. 2005. Mean value coordinates for closed triangular meshes. In *ACM Siggraph 2005 Papers*. 561–566.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Diederik P. Kingma, Tim Salimans, and Max Welling. 2016. Improving Variational Inference with Inverse Autoregressive Flow. *CoRR* abs/1606.04934 (2016). arXiv:1606.04934 http://arxiv.org/abs/1606.04934

Leif Kobbelt. 2000. $\sqrt{3}$-subdivision. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 103–112.

Der-Tsai Lee and Bruce J Schachter. 1980. Two algorithms for constructing a Delaunay triangulation. *International Journal of Computer & Information Sciences* 9, 3 (1980),

219–242.

Zohar Levi and Craig Gotsman. 2014. Smooth rotation enhanced as-rigid-as-possible mesh animation. *IEEE transactions on visualization and computer graphics* 21, 2 (2014), 264–277.

Hao Li, Robert W Sumner, and Mark Pauly. 2008. Global correspondence optimization for non-rigid registration of depth scans. In *Computer graphics forum*, Vol. 27. Wiley Online Library, 1421–1430.

Yaron Lipman, David Levin, and Daniel Cohen-Or. 2008. Green coordinates. *ACM Transactions on Graphics (TOG)* 27, 3 (2008), 1–10.

Niloy J Mitra, Natasha Gelfand, Helmut Pottmann, and Leonidas Guibas. 2004. Registration of point cloud data from a geometric optimization perspective. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. 22–31.

Jorge J Moré. 1978. The Levenberg-Marquardt algorithm: implementation and theory. In *Numerical analysis*. Springer, 105–116.

Richard A Newcombe, Dieter Fox, and Steven M Seitz. 2015. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 343–352.

George Papamakarios, Theo Pavlakou, and Iain Murray. 2017. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*. 2338–2347.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 652–660.

Danilo Jimenez Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770* (2015).

Szymon Rusinkiewicz and Marc Levoy. 2001. Efficient variants of the ICP algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*. IEEE, 145–152.

Leonardo Sacht, Etienne Vouga, and Alec Jacobson. 2015. Nested cages. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–14.

Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, Vol. 4. 109–116.

Olga Sorkine and Mario Botsch. 2009. Interactive shape modeling and deformation. (2009).

Jos Stam. 1998. Evaluation of loop subdivision surfaces. In *SIGGRAPHâĂŹ98 CDROM Proceedings*. Citeseer.

Robert W Sumner and Jovan Popović. 2004. Deformation transfer for triangle meshes. *ACM Transactions on graphics (TOG)* 23, 3 (2004), 399–405.

Robert W Sumner, Johannes Schmid, and Mark Pauly. 2007. Embedded deformation for shape manipulation. In *ACM SIGGRAPH 2007 papers*. 80–es.

Jean-Marc Thiery, Julien Tierny, and Tamy Boubekeur. 2012. CageR: cage-based reverse engineering of animated 3D shapes. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 2303–2316.

Mikaela Angelina Uy, Jingwei Huang, Minhyuk Sung, Tolga Birdal, and Leonidas Guibas. 2020. Deformation-Aware 3D Model Embedding and Retrieval. *arXiv preprint arXiv:2004.01228* (2020).

Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. 2018. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 52–67.

Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. 2019. 3dn: 3d deformation network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1038–1046.

Chuhua Xian, Hongwei Lin, and Shuming Gao. 2012. Automatic cage generation by improved obbs for mesh deformation. *The Visual Computer* 28, 1 (2012), 21–33.

Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. 2019. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE International Conference on Computer Vision*. 4541–4550.

Wang Yifan, Noam Aigerman, Vladimir Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. 2019. Neural Cages for Detail-Preserving 3D Deformations. *arXiv preprint arXiv:1912.06395* (2019).

M Ersin Yumer and Niloy J Mitra. 2016. Learning semantic deformation flows with 3d convolutional networks. In *European Conference on Computer Vision*. Springer, 294–311.

Qian-Yi Zhou and Vladlen Koltun. 2014. Simultaneous localization and calibration: Self-calibration of consumer depth cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 454–460.

Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. 2013. Elastic fragments for dense scene reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision*. 473–480.

APPENDIX

A SHAPE DEFORMATION

Following Section 6.1 in the main paper, we provide additional visual comparisons on shape deformation in Figure 15 and 16. 3DN [Wang et al. 2019] and NeuralCage [Yifan et al. 2019] do not fit the target as good as our methods do. Ours-1way is not optimized for the coverage of the target, while ours-2way ensures coverage of both the deformed and the target shape. Ours-neural additionally provides more flexible deformation with better fitting quality.

B SHAPE CREATION AND ANIMATION

Following Section 7.1 in the main paper, we provide additional examples for shape animation and design in Figure 17, 18 and 19. By optimizing the smooth deformation function $\mathcal{D}_{\text{ode}}$, we are able to obtain a time-dependent velocity field $\mathbf{u}(\mathbf{p}, t)$. We can create intermediate steps of the deformation $\mathbf{p}(\mathbf{x}, t_i)$ as an animation, where $\mathbf{x} \in \mathcal{V}_{\text{src}}$ and t_i are uniformly and densely sampled in $[0, 1]$. Such intermediate steps can be used for interactive novel shape design.

C SCAN-TO-CAD FITTING

Following Section 7.2 in the main paper, we provide additional examples for scan-to-CAD deformation fitting comparisons in Figure 20. Comparing with [Uy et al. 2020], we produce better fitting results without collapsing parts which are missing in scans.

D TEXTURE RECONSTRUCTION

Following Section 7.3 in the main paper, we provide additional examples in Figure 21 comparing texture reconstruction using the roughly-aligned original and deformed CAD models using [Huang et al. 2020]. Deformed CAD models fit scans better and leads to better texture reconstruction from aligned images.

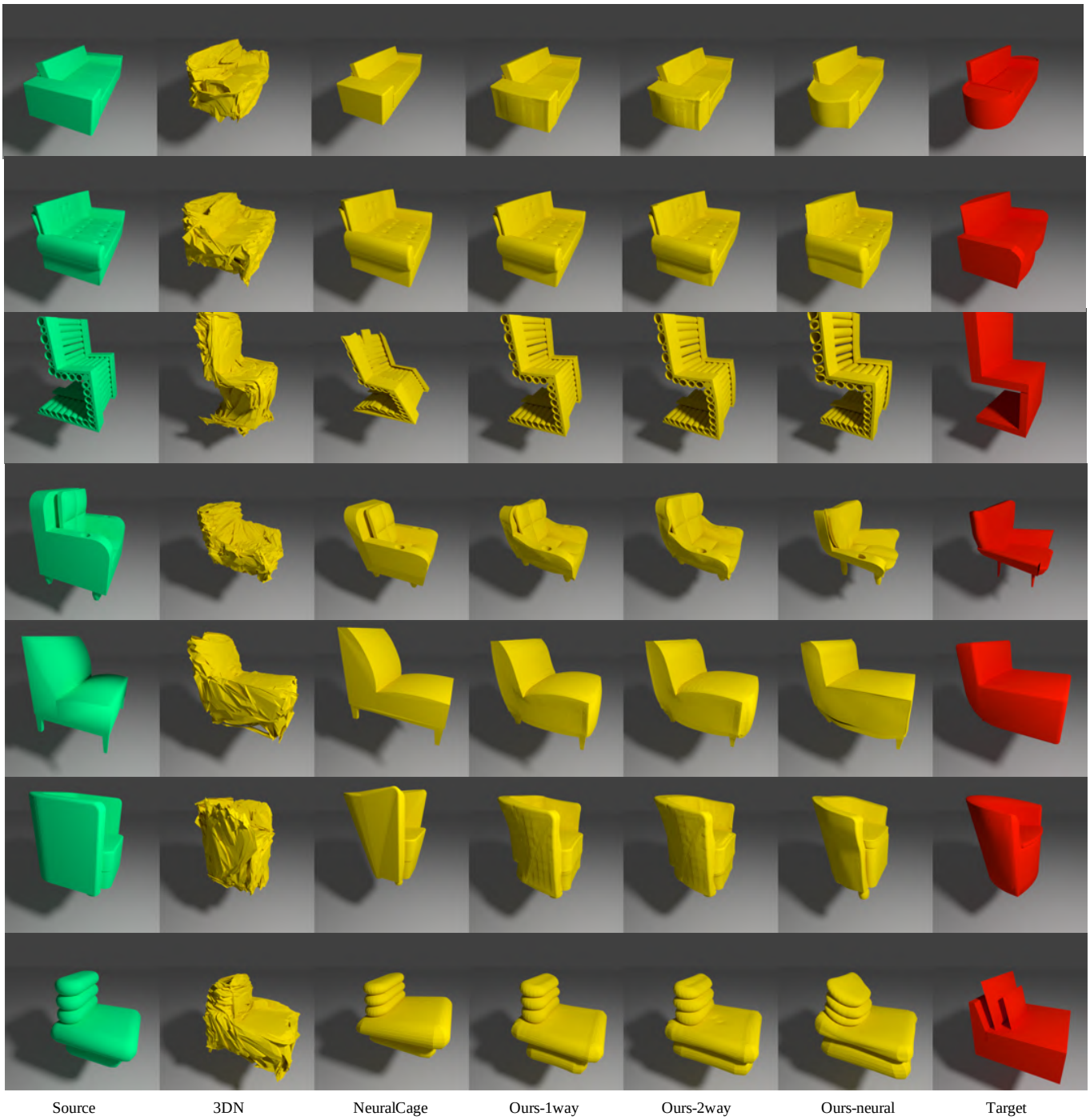

Fig. 15. Visual comparison on shape deformation between different approaches (examples set 2).

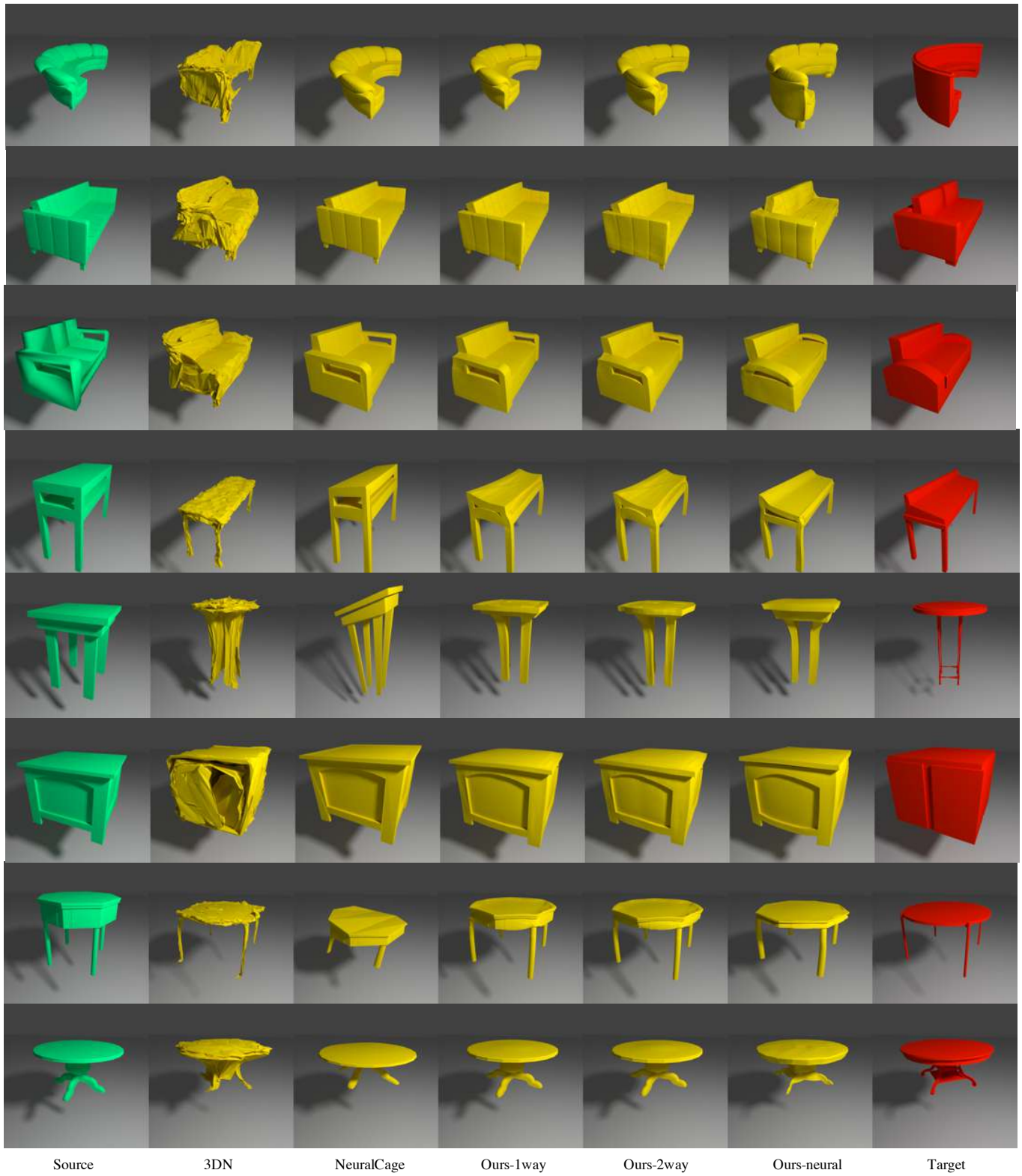

Fig. 16. Visual comparison on shape deformation between different approaches (examples set 2).

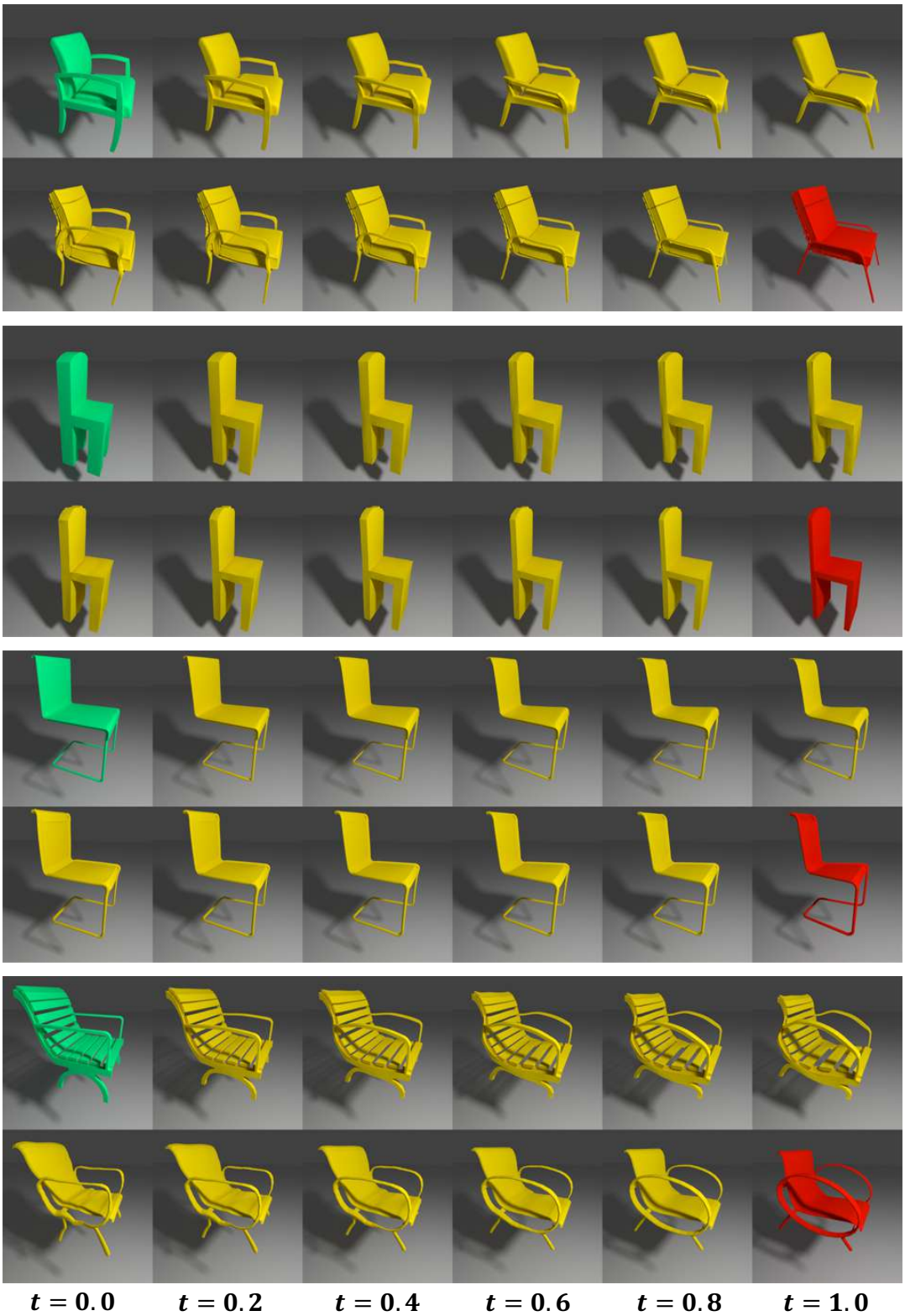

Fig. 17. Bidirectional shape animation (examples set 1). For every two rows, our input is the two shapes shown in green and red. Each shape can be deformed to the other with reasonable intermediate steps. Designers can interactively create novel shapes by controlling t with our bijective smooth function.

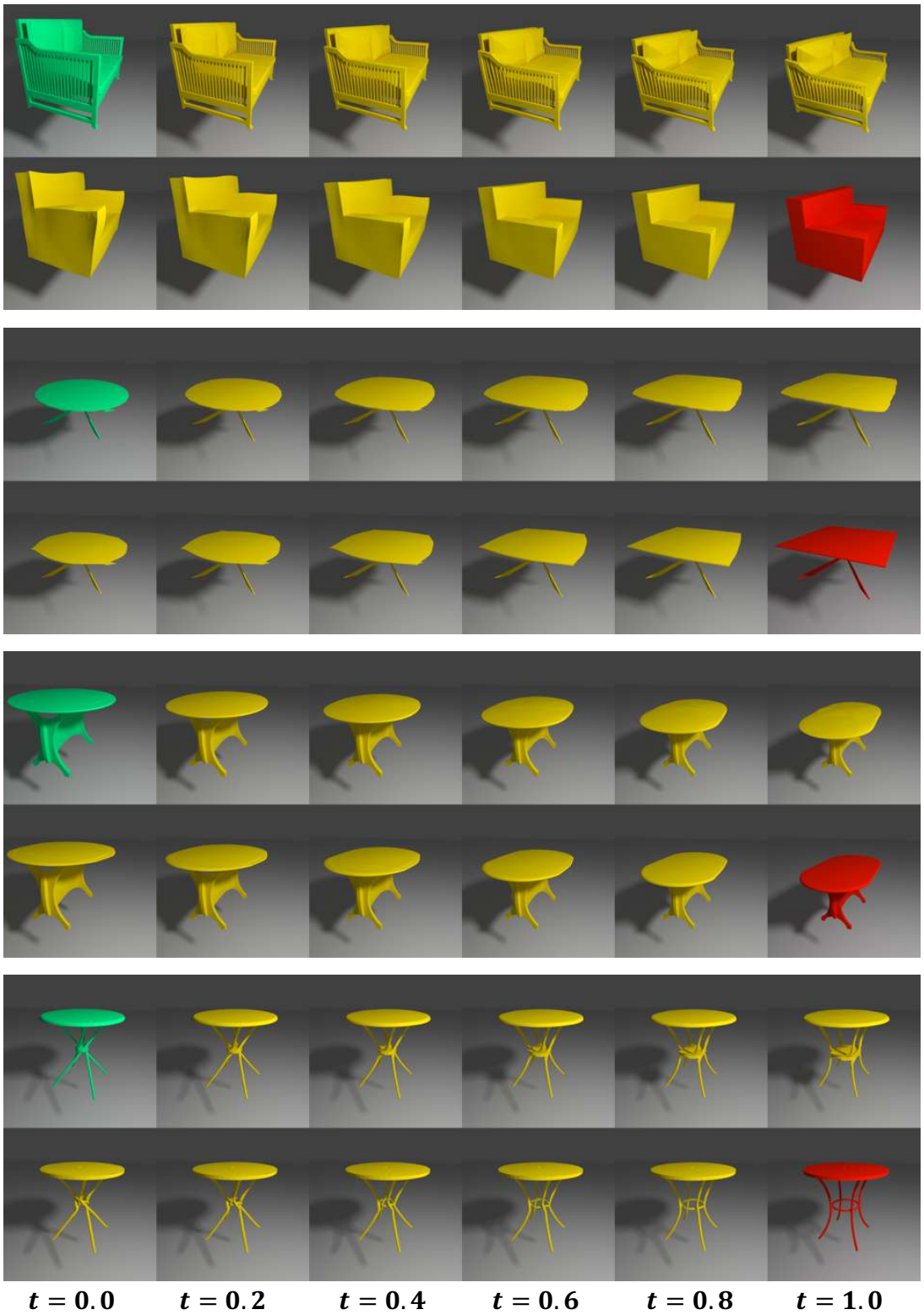

Fig. 18. Bidirectional shape animation (examples set 2). For every two rows, our input is the two shapes shown in green and red. Each shape can be deformed to the other with reasonable intermediate steps. Designers can interactively create novel shapes by controlling t with our bijective smooth function.

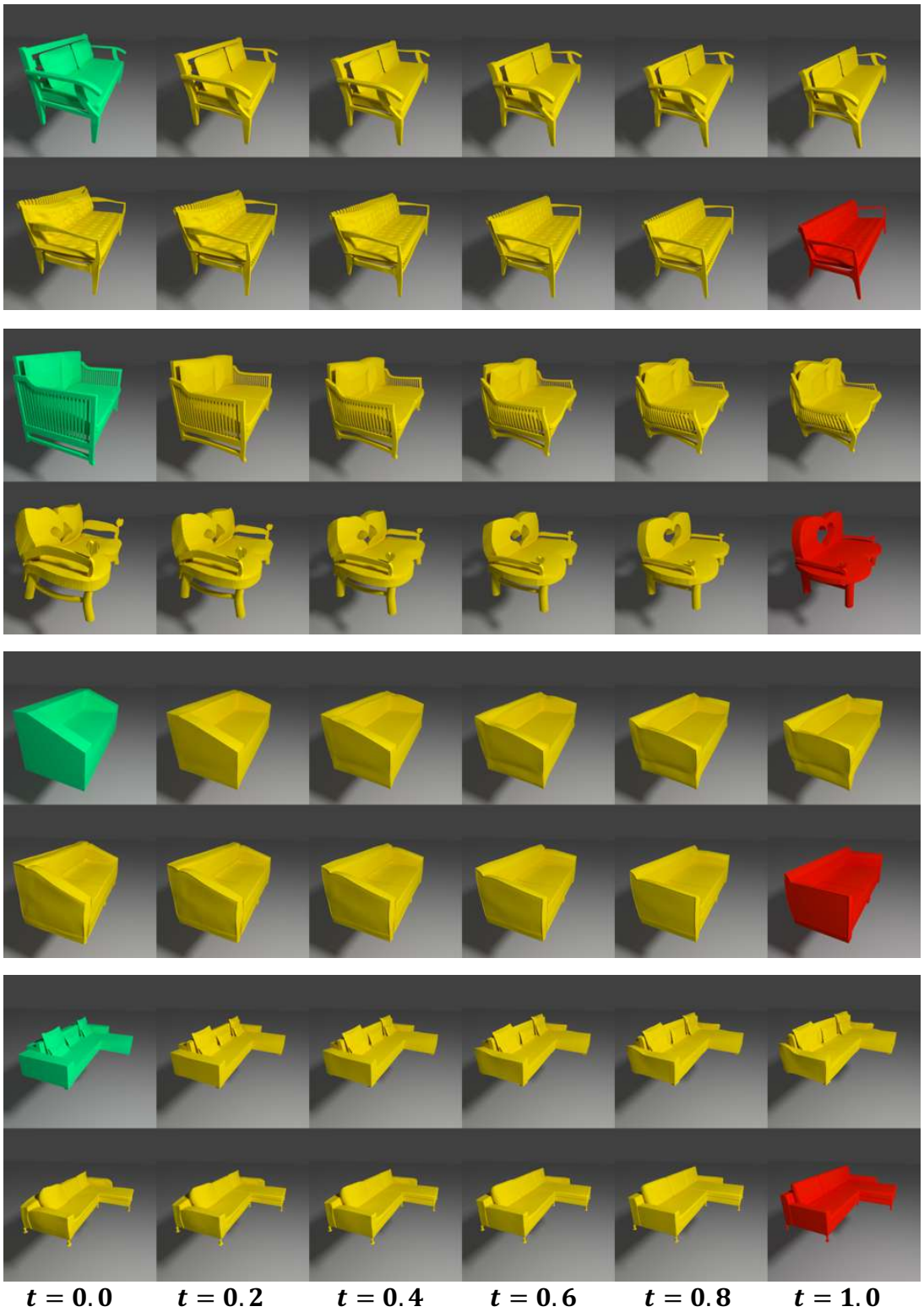

Fig. 19. Bidirectional shape animation (examples set 3). For every two rows, our input is the two shapes shown in green and red. Each shape can be deformed to the other with reasonable intermediate steps. Designers can interactively create novel shapes by controlling t with our bijective smooth function.

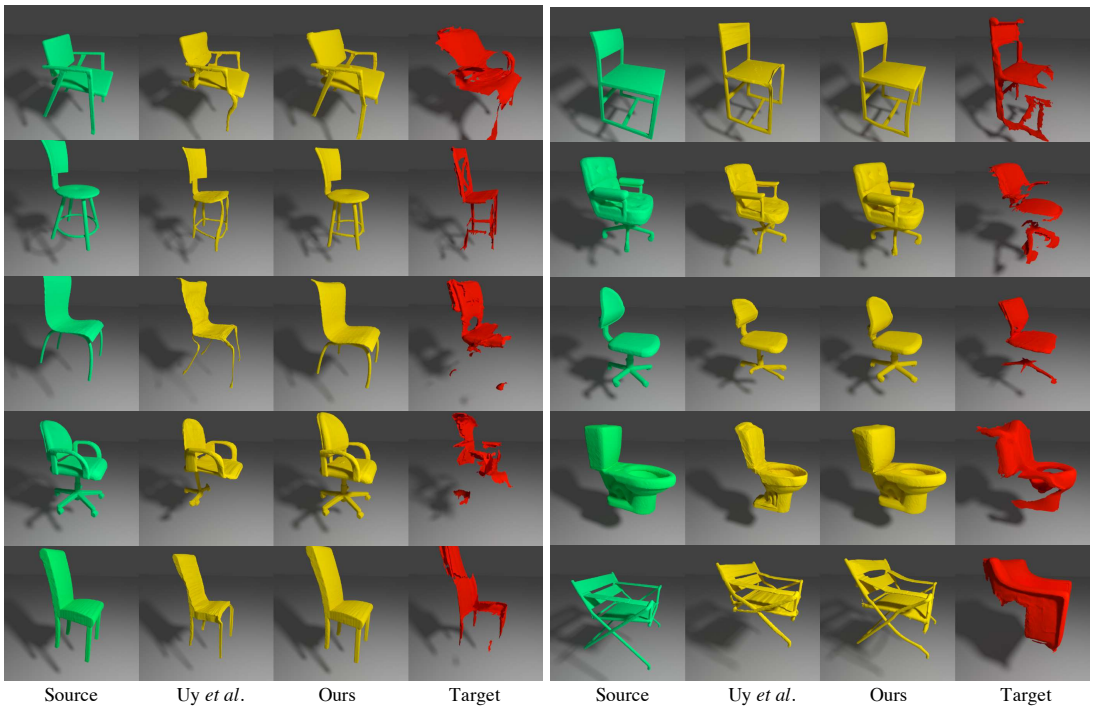

Fig. 20. Fitting errors between retrieved/deformed CAD models to scans in Scan2CAD [Avetisyan et al. 2019a]. Our deformation produces less fitting errors and more visually appealing results compared to the state-of-the-art.

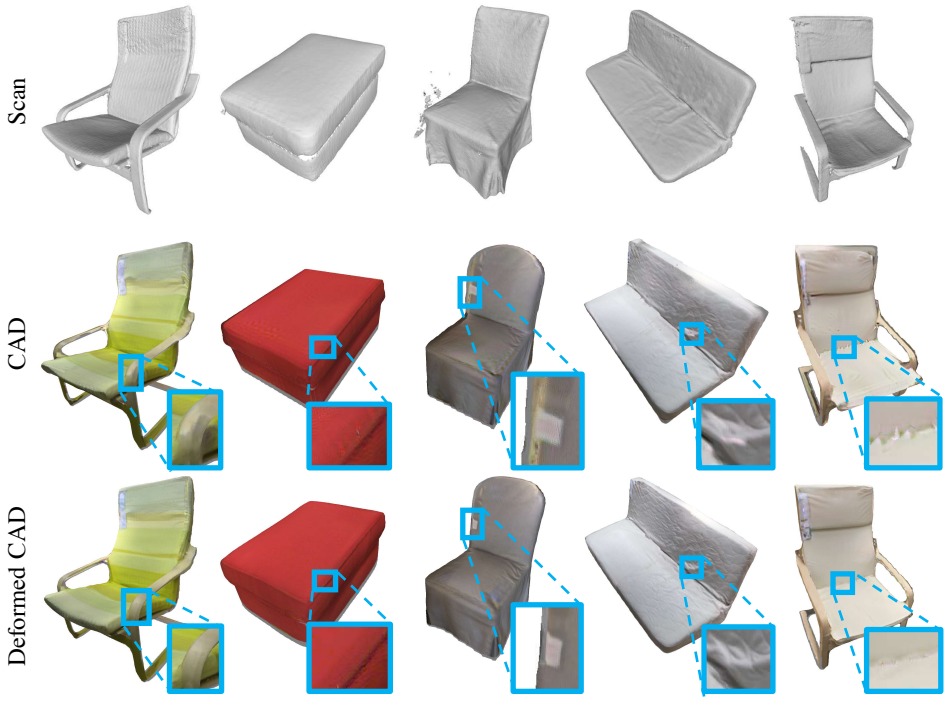

Fig. 21. Texture optimization comparison with [Huang et al. 2020]. Texture can be better optimized with our deformed CAD model since it fits better to the scan and has better geometry alignment with the scanning images.