Pacific Graphics 2023
R. Chaine, Z. Deng, and M. H. Kim
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 42 (2023), Number 7

OptCtrlPoints: Finding the Optimal Control Points for Biharmonic 3D Shape Deformation

Kunho Kim[*1] Mikaela Angelina Uy[*2] Despoina Paschalidou[2] Alec Jacobson[3] Leonidas J. Guibas[2] Minhyuk Sung[1]

[1]KAIST [2]Stanford University [3]University of Toronto

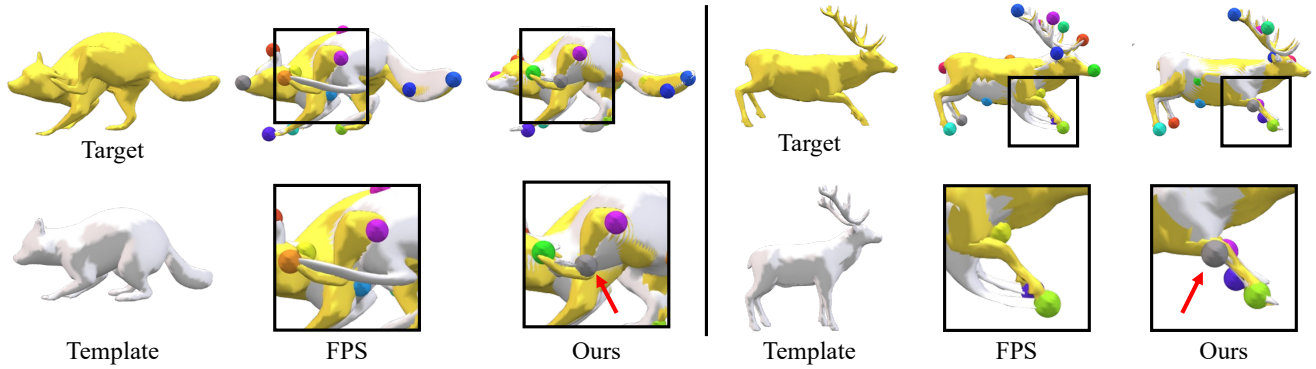

Figure 1: OPTCTRLPOINTS, *a data-driven method to search for the optimal set of control points, enables accurate replication of the possible variations of a shape through biharmonic deformation. In contrast to control points obtained through Farthest Point Sampling, the control points discovered by* OPTCTRLPOINTS *are strategically positioned, such as at the knees (see the red arrows in the black boxes), resulting in a superior fit of the template to the targets during deformation.*

Abstract

We propose OPTCTRLPOINTS, *a data-driven framework designed to identify the optimal sparse set of control points for reproducing target shapes using biharmonic 3D shape deformation. Control-point-based 3D deformation methods are widely utilized for interactive shape editing, and their usability is enhanced when the control points are sparse yet strategically distributed across the shape. With this objective in mind, we introduce a data-driven approach that can determine the most suitable set of control points, assuming that we have a given set of possible shape variations. The challenges associated with this task primarily stem from the computationally demanding nature of the problem. Two main factors contribute to this complexity: solving a large linear system for the biharmonic weight computation and addressing the combinatorial problem of finding the optimal subset of mesh vertices. To overcome these challenges, we propose a reformulation of the biharmonic computation that reduces the matrix size, making it dependent on the number of control points rather than the number of vertices. Additionally, we present an efficient search algorithm that significantly reduces the time complexity while still delivering a nearly optimal solution. Experiments on SMPL, SMAL, and DeformingThings4D datasets demonstrate the efficacy of our method. Our control points achieve better template-to-target fit than FPS, random search, and neural-network-based prediction. We also highlight the significant reduction in computation time from days to approximately 3 minutes.*

CCS Concepts

• ***Computing methodologies*** → ***Mesh models; Mesh geometry models; Shape analysis;***

1. Introduction

The demand for high-quality 3D models is growing rapidly, especially with recently emerging applications in virtual and augmented realities, gaming, robotics, animation, etc. However, creating and designing high-quality 3D models is a tedious and difficult process even for expert designers. Shape deformation is thus an important technique that enables producing plausible variations of existing high-quality, artist-generated 3D assets.

[*] denotes equal contribution.

Deforming a 3D model is, however, a highly non-trivial task. A straightforward approach is to parameterize deformation as positions of all the mesh vertices [GFK*19, SJA*20, WCMN19], although it is difficult for users to edit the shape and also can lead to unrealistic outputs due to its large degree of freedom. To circumvent this conundrum, existing works in geometry processing [JBPS11, WJBK15, LLCO08, JSW05, JMD*07, HS08, WBGH11, LH13, WSLG07, BP07] leverage a *sparse* set of *deformation handles* to constrain and parameterize deformation within a lower degree of freedom, facilitating more intuitive editing via interactions with the users.

For handle-based deformation to be effective and useful, there are several desirable properties, such as identity (i.e. preserving the shape under zero handle movement), locality, smoothness, closed-form expression, and flexibility for the representation of shapes. Due to such desirable properties, many existing deformation methods [JBPS11, WJBK15] use a set of *points* or *regions* in the mesh, which is typically a subset of the mesh vertices, as the handles (Fig. 1) while defining the shape deformation function from the handles using *biharmonic* weights. Given any input shape, point and region handles can be directly selected, and their biharmonic weights can also be directly computed, offering flexibility and convenience. This is in contrast to other types of deformation handles, such as cages [LLCO08, JSW05, JMD*07, HS08, WBGH11, LH13] and skeletons [WSLG07, BP07]. Cages require the manual construction of a closed polyhedral envelope for the shapes, while skeletons require rigging, where the skeleton structure needs to be delicately constructed to produce detailed deformation and typically necessitate manual weight painting.

Given a mesh and a sparse subset of the mesh vertices representing the control points, the biharmonic weights defining the linear map from the control point positions to the mesh vertex positions are calculated by solving a convex quadratic optimization problem [JBPS11], which was shown to be equivalent to solving a linear system [WJBK15] when relaxing some constraints. To obtain a wide range of plausible variations of the shape from a sparse set of control points, it is crucial to find the *optimal* set of control points. For an articulated 3D shape, for instance, the control points near the joints would be able to produce more realistic deformations by bending the joints properly (see control points at knees on the left of Fig. 1). Also, more control points would be needed in the regions with more detailed variations (see the additional control points on the leg on the right of Fig. 1).

In this work, we present a *data-driven* method for finding the optimal set of control points, coined OptCtrlPoints. Given a template mesh and its variant shapes (e.g. different poses in an animation), we find the ideal subset of the template mesh vertices as control points that can best fit the template to all the variant shapes via deformation. Our method can thus provide optimized deformation handles to the users and enables easier shape editing. In contrast to previous works [YAK*20, JTM*20] that utilize neural networks to learn deformation handles in a data-driven manner, our approach focuses on discovering a set of control points rather than fitting a sphere cage to the template mesh. The limitation of using a sphere cage is that it is unable to accommodate large deformations, making it unsuitable for non-rigid shapes like human or animal bodies. By identifying control points instead, our method enables more flexible and effective deformation modeling for such shapes.

Finding the optimal set of vertices poses a challenge due to the substantial amount of computation time involved, which can span several hours or more than a day. Two primary factors contribute to this extended computation duration. Firstly, the process of deforming the template mesh to accurately align with the target shapes is time-consuming. While deforming the mesh using *fixed* biharmonic weights can be performed quickly by prefactorizing a matrix within a linear system, the need to test different sets of control points prevents prefactorization of the matrix, leading to a considerably slower process. Secondly, solving a combinatorial optimization problem to determine the ideal subset of vertices becomes intractable when dealing with thousands or more vertices. When N is the number of vertices and K is the number of control points, a straightforward exhaustive search requires a time complexity of $\mathcal{O}(N^K)$, further contributing to the computational challenges.

To overcome these challenges and address the issue of intractable computation, we propose a novel algorithm that incorporates two key ideas. Firstly, we introduce a reformulation of the biharmonic weight computation, which significantly reduces the time required to solve the linear system. This is achieved by introducing a new linear system where the size of the matrix is not dependent on the number of vertices N, but rather on the number of control points K. This reformulation proves particularly effective in cases where the set of control points varies, as in our scenario where we search for the best set. Additionally, we present an efficient search algorithm that leverages the new biharmonic weight computation. This algorithm operates by iteratively updating control points one by one while simultaneously traversing local partitioned regions for each control point. This simple yet effective approach enables us to reduce the time complexity from $\mathcal{O}(N^K)$ to $\Theta(N+K^2)$, which is linear order of the number of the vertices while still providing a nearly optimal solution.

In our experiments, we assess the performance of our method by evaluating the alignment of the template to the target through deformation, comparing it with other baselines: Farthest Point Sampling (FPS), random search, and KeypointDeformer [JTM*20], a neural-network-based method for predicting keypoints for deformation. We conduct these evaluations on three datasets: SMPL [LMR*15], SMAL [ZKJB17], and DeformingThings4D [LTT*21]. Our results demonstrate that the control points discovered by our OptCtrl-Points algorithm offer a better fit of the template to the target shapes, thanks to their ideal locations for producing the desired variations. Additionally, our approach significantly reduces computation time, especially when compared to random search. In approximately 3 minutes, our method can find a good set of control points, whereas without the new biharmonic weight formulation and efficient search, it would take days to achieve a similar outcome. Furthermore, with the DeformingThings4D [LTT*21] dataset, we illustrate that our data-driven control point search method can discover an optimized set of control points tailored to the given set of target shapes. We conduct experiments using two setups: targeting all motions or focusing on a specific motion within the animations. The consistent lower fitting errors observed in the specific motion case compared

to the all motion case highlight the effectiveness of our data-driven approach.

2. Related work

2.1. 3D Shape Deformation

3D shape deformation has been a long-standing problem in computer graphics and geometry processing. The problem is to find the best vertex positions for a given mesh in order to obtain a new shape that best fits a target while preserving the local geometric details of the original shape. Previous approaches include free-form deformation [KSSCO06, SP86] that define a smooth deformation function by interpolating the weights of the voxel grids enclosing the surface, and vertex-based approaches [SCOL*04, IMH05, SA07, LSC*04, LSLCO05] where vertex positions are directly optimized through a target-fitting objective function. Regularization losses are also used, such as mesh Laplacian [SCOL*04, LSC*04] and local rigidity [LSLCO05, IMH05, SA07] to preserve the geometric details of the original shape. Learning-based approaches have also been introduced for both free-form [YM16, HFW*18, JPS*18, KJG*18] and vertex-based [GFK*19, WCMN19] deformation approaches. More recently, neural implicit functions [DYT21, ZYDL21] explore defining the deformation offset on the full coordinate space, instead of only on the surface of the mesh. These works, however, may lead to unrealistic outputs due to their large degrees of freedom and also do not exert intuitive control over the shape, as editing operations are performed in the implicit space.

2.2. Traditional Handle-based Shape Deformation

Deformation handles are commonly used to address the need for low-dimensional control on shape deformation and have been well studied in the computer graphics literature [SB09]. Earlier works use volumetric prisms [BPGK06, BS08] or off-surface handles [BPWG07] to compute detail-preserving shape deformation through variational methods. These variational methods typically require optimization at each time of deformation. Cage [JSW05, JMD*07, HS08, WBGH11, LH13, LLCO08] is another form of shape handles where a shape is enclosed in a coarse polytope, and the mesh vertices are defined as a linear combination of the cage vertices through generalized barycentric coordinates. Skeletons [WSLG07, BP07] also define a linear map from the joints and bones to the mesh vertices via linear blend skinning, while now the handles appear inside the shape. Both cages and skeletons allow for shape deformation to be expressible in a closed form, but require manual construction of the source cage or rigging. Jacobson et al. [JBPS11] introduced a handle-based deformation function based on solving the *biharmonic equation* over the mesh surface with boundary constraints that can use a set of points or regions in the mesh as handles to define shape deformation using biharmonic weights. Unlike cages and skeletons, these handles can directly be computed for any source shape and are thus flexible and versatile. Wang et al. [WJBK15] then introduced a closed-form formulation to the original constrained quadratic optimization formulation. In our method, we leverage this versatile deformation handle with an efficient reformulation to enable tractable gradient-based optimization for handle *discovery*, in contrast to existing works that assume shape handles to be given.

2.3. Learning Handle-based Shape Deformation

Recently, handle-based shape deformations have been revisited in the context of deep learning. DeformSyncNet [SJA*20] uses rigid bounding boxes as shape handles for learning a latent shape difference deformation space. Wang et al. [YAK*20] introduced Neural Cages, a neural network for cage-based deformations that predicts a source-dependent cage used to deform a source shape to match a given target. KeypointDeformer [JTM*20] leverages Neural Cages to learn keypoints that can be used for deformation. However, the degree of plausible output shape deformations yielded by the neural cage-based deformation methods is limited since they start from a sphere-based cage, making highly non-rigid deformations on non-sphere-like shapes difficult. In contrast, we leverage a deformation function defined with biharmonic coordinates that enable flexibility for any given source shape. Liu et al. [LSMS21] also use biharmonic coordinates as their deformation function. In contrast to our work, where the goal is the *discovery* of the control points, they assume them to be given and instead learn a latent space of *meta-handles* for the given set of control points. Moreover, we discover *explicit* handles, which are 3D mesh vertices, that allow direct user interpretability and controllability, instead of meta-handles in latent space.

3. Background

3.1. Shape Deformation and Deformation Handles

The creation of high-quality 3D models is a tedious process that requires manual expertise, thus making shape deformation an important task as it enables converting an existing 3D model to a new shape while preserving their fine details. However, naive mesh deformation through moving individual mesh vertices is cumbersome as it is both difficult for users and can easily lead to unrealistic outputs. Existing work [JBPS11, WJBK15, LLCO08, JSW05, JMD*07, HS08, WBGH11, LH13, WSLG07, BP07] thus introduce intuitive *deformation handles*, e.g. control points, cages, skeletons, etc., to constrain and parameterize deformation with a low degree of freedom. Several properties are critical for handle-based deformation to be effective and useful:

1. **Identity**: The original shape must be reconstructed under zero movement of shape handles.
2. **Locality**: The deformation produced by each individual handle must be local and smooth.
3. **Closed-form**: The output deformed shape must be expressed in a closed-form given the transformations of the deformation handles.
4. **Flexibility**: The deformation handles and function must be defined without any constraints or additional information about the shape (e.g., a cage, a skeleton, or bounding primitives).

For these reasons, many existing shape deformation methods [LSMS21, LSKK22, JBPS11, WJBK15] use biharmonic weights as the deformation function. Jacobson et al. [JBPS11] first introduced bounded biharmonic coordinates that solve biharmonic equations defined over a mesh to compute the linear map from the handles to the mesh vertices. Wang et al. [WJBK15] later introduced a closed-form formulation for the biharmonic coordinate-based deformation, which we base our work on. Below, we explain the details

about the closed-form formulation of the biharmonic coordinate deformation function.

3.2. Biharmonic Coordinates

Given a 3D volumetric mesh with N vertices and K control points, which is a sparse *subset* of the mesh vertices ($K \ll N$) represented with a binary selector matrix $\mathbf{S} \in \{0,1\}^{K\times N}$, the biharmonic deformation function [JBPS11, WJBK15] from the positions of the control points $\mathbf{C} \in \mathbb{R}^{K\times 3}$ to the positions of mesh vertices $\mathbf{V} \in \mathbb{R}^{N\times 3}$ is defined as a *linear* function: $\mathbf{V} = \mathbf{WC}$. (The original biharmonic deformation [JBPS11, WJBK15] supports region handles, while we limit our scope to point handles for simplicity.) Here, $\mathbf{W} \in \mathbb{R}^{N\times K}$ called *biharmonic weights* is derived from the solution of the following optimization for the vertex positions $\mathbf{V}$ with respect to the equality constraints on the control point positions:

$$\mathbf{V} = \underset{\mathbf{X}\in\mathbb{R}^{N\times 3}}{\operatorname{argmin}} \frac{1}{2} \operatorname{trace}\left(\mathbf{X}^\top \mathbf{A}\mathbf{X}\right) \text{ subject to } \mathbf{SX} = \mathbf{C}, \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{N\times N}$ denotes the discrete Bilaplacian matrix of the mesh. This optimization finds the positions of the mesh vertices $\mathbf{V}$ minimizing squared Laplacian energy when the positions of the selected control points are fixed to be $\mathbf{C}$. Since the Bilaplacian matrix $\mathbf{A}$ is positive semi-definite, the optimization is a convex quadratic programming problem that can be solved as a linear system as described in [WJBK15]. The solution thus has a form of $\mathbf{V} = \mathbf{WC}$ where

$$\mathbf{W}(\mathbf{S};\mathbf{A}) = \mathbf{S}^\top - \mathbf{T}^\top \left(\mathbf{TAT}^\top\right)^{-1} \mathbf{TAS}^\top, \tag{2}$$

and $\mathbf{T} \in \{0,1\}^{(N-K)\times N}$ is the complementary selector matrix of $\mathbf{S}$ indicating the mesh vertices that are *not* selected as control points. By the definition of the Bilaplacian matrix $\mathbf{A}$ including the original positions of the vertices $\overline{\mathbf{V}}$ in its null space, the given pose of the mesh $\overline{\mathbf{V}}$ becomes the solution of the optimization when the control points are not moved ($\mathbf{C} = \mathbf{S}\overline{\mathbf{V}}$), satisfying the identity condition.

To leverage the expressivity of biharmonic coordinates for shape deformation, we need to find the *optimal* set of control points (a sparse subset of the mesh vertices) that allows us to achieve a wide variety of plausible variants. In what follows, we introduce our *data-driven* approach of finding the optimal set of control points efficiently given possible variants of the shape.

4. OptCtrlPoints

4.1. Problem Definition

Given a template volumetric mesh with N vertices which Bilaplacian matrix is $\mathbf{A} \in \mathbb{R}^{N\times N}$, and a set of M target shapes $\{\mathcal{X}_i\}_{i=1}^M$ that are the possible variants of the template, our goal is to find the optimal K-subset of the N source vertices as the control points, denoted as $\tilde{\mathbf{S}} \in \{0,1\}^{K\times N}$, which can best fit all the target shapes with their corresponding positions in the target (see Fig. 2):

$$\begin{aligned}\tilde{\mathbf{S}} = \underset{\mathbf{S}\in\{0,1\}^{K\times N}}{\operatorname{argmin}} & \sum_{i=1}^{M} d(\mathcal{X}_i, \mathbf{W}(\mathbf{S};\mathbf{A})\mathbf{C}(\mathbf{S};\mathcal{X}_i)) \\ \text{s.t.} \quad & \sum_{j=1}^{N} \mathbf{S}_{ij} = 1 \quad \text{for all } i,\end{aligned} \tag{3}$$

where $d(\cdot,\cdot)$ is a shape-to-shape distance function, $\mathbf{W}(\mathbf{S};\mathbf{A})$ is the biharmonic weight function in Eq. 2 for the control point selector matrix $\mathbf{S}$ and the given Bilaplacian matrix $\mathbf{A}$, and $\mathbf{C}(\mathbf{S};\mathcal{X}_i)$ is a function computing the corresponding positions of the control points $\mathbf{S}$ in the target $\mathcal{X}_i$. Assuming that the point-wise map from each target shape $\mathcal{X}_i$ to the template is given or estimated with an off-the-shelf shape correspondence method (e.g., functional maps [OBCS*12], or its neural variants [LRR*17, HLR*19, DSO20, ZQZ*21, APO21, MRSHO22]; see a survey [Sah20]), let $f : \{\mathcal{X}_i\}_{i=1}^M \to \mathbb{R}^{N\times 3}$ denote a function returning corresponding points of each vertex of the template in the same order. Then $\mathbf{C}(\mathbf{S};\mathcal{X}_i) = \mathbf{S}f(\mathcal{X}_i)$.

The main challenge in finding the optimal K-subset of template vertices as control points in Eq. 3 lies in the extensive computation time required. Note that when the control points and resulting biharmonic weights are *fixed*, the deformations can be computed quickly by prefactorizing $\mathbf{TAT}^\top$ in Eq. 2. However, when computing deformations with *different* sets of control points, it is not feasible to leverage the prefactorization since the complementary selector matrix $\mathbf{T}$ varies. Consequently, computing the biharmonic weight matrix $\mathbf{W}$ in Eq. 2 becomes the bottleneck, taking several seconds to compute even once (for detailed analysis, refer to Sec. 5). Moreover, identifying the optimal K-subset from a pool of N elements is an NP-complete combinatorial optimization problem, making exhaustive search impractical due to the typically high number of vertices, often in the thousands.

To address this challenge, we present our efficient control point search framework, called OptCtrlPoints. First, in Sec. 4.2, we introduce a reformulation of Eq. 2 that yields the same biharmonic weight matrix $\mathbf{W}$ while solving a linear system on a much smaller scale, significantly reducing computation time. Second, in Sec. 4.3, we propose an efficient search algorithm that reduces the time complexity of the search from $\mathcal{O}(N^K)$ (for exhaustive search) to $\Theta(N+K^2)$ in average, while effectively finding nearly optimal solutions in practice.

4.2. Reformulation of $\mathbf{W}(\mathbf{S};\mathbf{A})$ (Eq. 2)

Let $\mathbf{M}$ denote the linear system in $\mathbf{W}(\mathbf{S};\mathbf{A})$ (Eq. 2):

$$\mathbf{M} = \left(\mathbf{TAT}^\top\right)^{-1} \mathbf{TAS}^\top. \tag{4}$$

We begin by introducing a reformulation of $\mathbf{M}$ for the case when $\mathbf{A}$ is not a Bilaplacian matrix but rather an arbitrary invertible square matrix. However, in our specific scenario, $\mathbf{A}$ is the Bilaplacian matrix, which is non-invertible and positive-semidefinite. In Section 4.2.1, we elaborate on how we address the singularity of $\mathbf{A}$ in this new formulation.

We choose a permutation matrix $\mathbf{P} \in \mathbb{R}^{N\times N}$ such that the product of $\mathbf{P}$ with $\mathbf{S}$ and $\mathbf{T}$ becomes $\mathbf{SP} = \left[\mathbf{0}_{K\times(N-K)} \mid \mathbf{I}_{K\times K}\right]$ and $\mathbf{TP} = \left[\mathbf{I}_{(N-K)\times(N-K)} \mid \mathbf{0}_{(N-K)\times K}\right]$, respectively. Moreover, using $\mathbf{P}$, we define $\mathbf{B} = \mathbf{P}^\top\mathbf{AP}$ and its inverse $\mathbf{D} = \mathbf{P}^\top\mathbf{A}^{-1}\mathbf{P}$ as follows:

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{12}^\top & \mathbf{B}_{22} \end{bmatrix}, \mathbf{D} = \begin{bmatrix} \mathbf{D}_{11} & \mathbf{D}_{12} \\ \mathbf{D}_{12}^\top & \mathbf{D}_{22} \end{bmatrix}, \tag{5}$$

where $\mathbf{B}_{11}, \mathbf{D}_{11} \in \mathbb{R}^{(N-K)\times(N-K)}$, $\mathbf{B}_{12}, \mathbf{D}_{12} \in \mathbb{R}^{(N-K)\times K}$, and

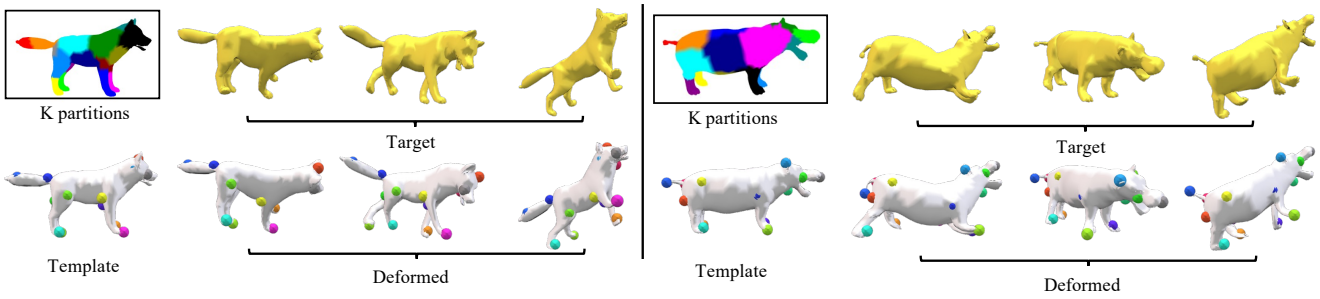

Figure 2: *Examples of the template and target shapes, along with the fitting results obtained through biharmonic deformations from the template to the target. The colored points indicate the control points used for the deformation computation. The segmented shapes within the black boxes at the top left corner illustrates the partitioned volume of the source tetrahedral mesh, enabling a level-of-detail search for the optimal placement of each control point.*

$\mathbf{B}_{22}, \mathbf{D}_{22} \in \mathbb{R}^{K\times K}$ are block matrices. Now taking into account that a permutation matrix is an orthogonal matrix (i.e. $\mathbf{P}^{-1} = \mathbf{P}^\top$) and $\mathbf{P}\mathbf{P}^\top = \mathbf{I}$, we can rewrite Eq. 4 as follows:

$$\begin{aligned}
\mathbf{M} &= \left(\mathbf{T}\mathbf{A}\mathbf{T}^\top\right)^{-1}\mathbf{T}\mathbf{A}\mathbf{S}^\top \\
&= \left((\mathbf{T}\mathbf{P})\left(\mathbf{P}^\top\mathbf{A}\mathbf{P}\right)(\mathbf{T}\mathbf{P})^\top\right)^{-1}(\mathbf{T}\mathbf{P})\left(\mathbf{P}^\top\mathbf{A}\mathbf{P}\right)(\mathbf{S}\mathbf{P})^\top \\
&= \left((\mathbf{T}\mathbf{P})\mathbf{B}(\mathbf{T}\mathbf{P})^\top\right)^{-1}(\mathbf{T}\mathbf{P})\mathbf{B}(\mathbf{S}\mathbf{P})^\top \\
&= \left(\begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{B}_{11} & \mathbf{B}_{12}\\ \mathbf{B}_{12}^\top & \mathbf{B}_{22}\end{bmatrix}\begin{bmatrix}\mathbf{I}\\ \mathbf{0}\end{bmatrix}\right)^{-1} \\
&\quad \begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{B}_{11} & \mathbf{B}_{12}\\ \mathbf{B}_{12}^\top & \mathbf{B}_{22}\end{bmatrix}\begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix} \\
&= \mathbf{B}_{11}^{-1}\mathbf{B}_{12}.
\end{aligned} \tag{6}$$

By using the Schur complement [Hay68], we can express $\mathbf{D}$, the inverse of $\mathbf{B}$ as follows:

$$\mathbf{D}^{-1} = \left[\begin{array}{c|c}\mathbf{D}_{11} & \mathbf{D}_{12}\\ \hline \mathbf{D}_{12}^\top & \mathbf{D}_{22}\end{array}\right]^{-1} = \left[\begin{array}{c|c}\mathbf{B}_{11} & \mathbf{B}_{12}\\ \hline \mathbf{B}_{12}^\top & \mathbf{B}_{22}\end{array}\right], \tag{7}$$

where

$$\begin{aligned}
\mathbf{B}_{11} &= (\mathbf{D}_{11} - \mathbf{D}_{12}\mathbf{D}_{22}^{-1}\mathbf{D}_{12}^\top)^{-1}, \\
\mathbf{B}_{12} &= -\left(\mathbf{D}_{11} - \mathbf{D}_{12}\mathbf{D}_{22}^{-1}\mathbf{D}_{12}^\top\right)^{-1}\mathbf{D}_{12}\mathbf{D}_{22}^{-1}, \text{ and} \\
\mathbf{B}_{22} &= \mathbf{D}_{22}^{-1} + \mathbf{D}_{22}^{-1}\mathbf{D}_{12}^\top\left(\mathbf{D}_{11} - \mathbf{D}_{12}\mathbf{D}_{22}^{-1}\mathbf{D}_{12}^\top\right)^{-1}\mathbf{D}_{12}\mathbf{D}_{22}^{-1}.
\end{aligned} \tag{8}$$

Thus, setting $\mathbf{Q} = \mathbf{D}_{11} - \mathbf{D}_{12}\mathbf{D}_{22}^{-1}\mathbf{D}_{12}^\top$, we have $\mathbf{B}_{11}^{-1} = \mathbf{Q}$ and $\mathbf{B}_{12} = -\mathbf{Q}^{-1}\mathbf{D}_{12}\mathbf{D}_{22}^{-1}$. Hence we get

$$\mathbf{M} = \mathbf{B}_{11}^{-1}\mathbf{B}_{12} = -\mathbf{D}_{12}\mathbf{D}_{22}^{-1}. \tag{9}$$

Finally, by considering the orthogonality of the permutation matrix $\mathbf{P}$ and that $\mathbf{P}^\top = \begin{bmatrix}\mathbf{T}\\ \mathbf{S}\end{bmatrix}$ based on its definition, we can show that $\mathbf{D}_{12} = \mathbf{T}\mathbf{A}^{-1}\mathbf{S}^\top$ and that $\mathbf{D}_{22} = \mathbf{S}\mathbf{A}^{-1}\mathbf{S}^\top$ as follows:

$$\begin{aligned}
\mathbf{D} &= \mathbf{P}^\top\mathbf{A}^{-1}\mathbf{P} \\
&= \begin{bmatrix}\mathbf{T}\\ \mathbf{S}\end{bmatrix}\mathbf{A}^{-1}\begin{bmatrix}\mathbf{T}^\top \mid \mathbf{S}^\top\end{bmatrix} \\
&= \left[\begin{array}{c|c}\mathbf{T}\mathbf{A}^{-1}\mathbf{T}^\top & \mathbf{T}\mathbf{A}^{-1}\mathbf{S}^\top\\ \hline \mathbf{S}\mathbf{A}^{-1}\mathbf{T}^\top & \mathbf{S}\mathbf{A}^{-1}\mathbf{S}^\top\end{array}\right] \\
&= \left[\begin{array}{c|c}\mathbf{D}_{11} & \mathbf{D}_{12}\\ \hline \mathbf{D}_{12}^\top & \mathbf{D}_{22}\cdot\end{array}\right],
\end{aligned} \tag{10}$$

Then, Eq. 9 becomes:

$$\mathbf{M} = -\mathbf{T}\mathbf{A}^{-1}\mathbf{S}^\top\left(\mathbf{S}\mathbf{A}^{-1}\mathbf{S}^\top\right)^{-1}. \tag{11}$$

By replacing Eq. 11 in our initial expression from Eq. 2, we obtain the following reformulation:

$$\mathbf{W}(\mathbf{S};\mathbf{A}) = \mathbf{S}^\top + \mathbf{T}^\top\left(\mathbf{T}\mathbf{A}^{-1}\mathbf{S}^\top\left(\mathbf{S}\mathbf{A}^{-1}\mathbf{S}^\top\right)^{-1}\right). \tag{12}$$

Note that Eq. 12 includes a linear system with a significantly smaller matrix, $\mathbf{S}\mathbf{A}^{-1}\mathbf{S}^\top \in \mathbb{R}^{K\times K}$, where $K \ll N$. Also, $\mathbf{A}^{-1}$ can be pre-computed to speed up the computation at each iteration.

4.2.1. Handling the Singularity of the Bilaplacian Matrix A

The reformulation of $\mathbf{M}$ (Eq. 11) cannot be directly used in our case since the Bilaplacian matrix $\mathbf{A}$ is a singular matrix. One possible approach to handling the singularity of $\mathbf{A}$ is to leverage the shaving-off technique by Jacobson [Jac14] while fixing a single control point during the control point search. Namely, when rewriting the optimization problem in Eq. 1 into a system of linear equations as follows:

$$\begin{bmatrix}\mathbf{A} & \mathbf{S}^\top\\ \mathbf{S} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{X}\\ \Lambda\end{bmatrix} = \begin{bmatrix}\mathbf{0}\\ \mathbf{C}\end{bmatrix}, \tag{13}$$

where $\Lambda \in \mathbb{R}^K$ is a vector of Lagrange multipliers, and assuming the fixed control point occupies the last index of vertices without loss of generality, the matrix on the left side of the linear system can be re-split in a way to shave off the last row and column of the Bilaplacian matrix $\mathbf{A}$ while expanding the selector matrix $\mathbf{S}$ and the zero matrix

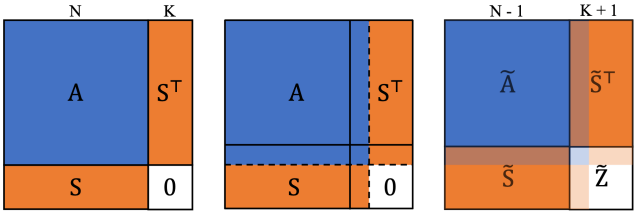

Figure 3: *Shaving-off technique handling the singularity of the Bilaplacian matrix* $\mathbf{A}$. *When assuming the last vertex is fixed as one of the control points, we define a new matrix* $\tilde{\mathbf{A}}$ *by taking the last row and column from* $\tilde{\mathbf{A}}$ *and also expand the selector matrix* $\mathbf{S}$ *and the zero matrix region. Since the Bilaplacian matrix is rank* $N-1$, *the new matrix* $\tilde{\mathbf{A}}$ *has full rank, and thus the Schur complement trick in Sec. 4.2 can be used.*

region, resulting in $\tilde{\mathbf{A}} \in \mathbb{R}^{(N-1)\times(N-1)}$, $\tilde{\mathbf{S}} \in \mathbb{R}^{(K+1)\times(N-1)}$, and $\tilde{\mathbf{Z}} \in \mathbb{R}^{(K+1)\times(K+1)}$ as follows (see Fig. 3):

$$\begin{bmatrix} \tilde{\mathbf{A}} & \tilde{\mathbf{S}}^\top \\ \tilde{\mathbf{S}} & \tilde{\mathbf{Z}} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ \Lambda \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \tilde{\mathbf{C}} \end{bmatrix}. \tag{14}$$

where $\tilde{\mathbf{C}} \in \mathbb{R}^{(K+1)\times 3}$ is the concatenation of a zero vector and $\mathbf{C} \in \mathbb{R}^{K\times 3}$. Since the Bilaplacian matrix $\mathbf{A}$ has rank $N-1$, $\tilde{\mathbf{A}}$ obtained by removing the last row and column of $\mathbf{A}$ has full rank. Thus, the Schur complement trick in Sec. 4.2 can now be directly utilized. One difference is that the bottom-right block matrix on the left side of the new linear system is not a zero matrix but $\tilde{\mathbf{Z}}$. Hence, Eq. 11 needs to be modified as follows:

$$\mathbf{M} = -\tilde{\mathbf{T}}\tilde{\mathbf{A}}^{-1}\tilde{\mathbf{S}}^\top \left(\tilde{\mathbf{S}}\tilde{\mathbf{A}}^{-1}\tilde{\mathbf{S}}^\top - \tilde{\mathbf{Z}}\right)^{-1}, \tag{15}$$

where $\tilde{\mathbf{T}} \in \{0,1\}^{(N-K)\times(N-1)}$ is the complement of the selector matrix $\mathbf{T}$ without the last column. Note that the last vertex represents the fixed control point, and thus, the complementary set of the control point selection in $\tilde{\mathbf{T}}$ is not changed. $\mathbf{W}(\mathbf{S};\mathbf{A})$ in Eq. 12 is also reformulated again as follows:

$$\tilde{\mathbf{W}}(\mathbf{S};\mathbf{A}) = \tilde{\mathbf{S}}^\top + \tilde{\mathbf{T}}^\top \left(\tilde{\mathbf{T}}\tilde{\mathbf{A}}^{-1}\tilde{\mathbf{S}}^\top \left(\tilde{\mathbf{S}}\tilde{\mathbf{A}}^{-1}\tilde{\mathbf{S}}^\top - \tilde{\mathbf{Z}}\right)^{-1}\right). \tag{16}$$

The positions of the vertices, excluding the fixed single control point, can be computed as $\tilde{\mathbf{W}}\tilde{\mathbf{C}}$; note that the position of the control point is given.

Alternatively, one can simply consider regularizing the Bilaplacian matrix $\mathbf{A}$ by adding a small-weighted identity matrix (e.g., $\mathbf{A}+\varepsilon\mathbf{I}$), approximating the solution while achieving numerical stability. In our experiments, we empirically find that this simple approach, which does not even require fixing any control points, performs well in practice for identifying the best set of control points. As a result, we use this regularization approach in our implementation.

4.3. Control Point Search Algorithm

Although the computation of the biharmonic weight matrix $\mathbf{W}$ in Eq. 12 is fast, finding the K optimal control points that best align the template mesh to the target shapes via deformation remains computationally infeasible when an exhaustive search of $\binom{N}{K}$ computations

Algorithm 1: Pseudocode of OPTCTRLPOINTS.

```
/* Input: The ordered list of initial vertex
   indices, the ordered list of sets of the
   partitioned vertex indices, and the set of
   target shapes.                                */
/* Output: The ordered list of control point
   vertex indices.                               */
Inputs: (s_k^(0))_{k=1}^N, (V_k)_{k=1}^N, {X_i}_{i=1}^M.
Outputs: (s_k)_{k=1}^N
Function FittingDist((s_k)_{k=1}^N):
    S ← S((s_k)_{k=1}^N);      // Update the binary matrix.
    d ← Σ_i d(X_i, W(S;A)C(S;X_i));                // Eq. 3
    return d;
Function FindRegion((s_k)_{k=1}^N, k, d_min):
    l_min ← k;
    for l = 1,...,K do
        /* Resample s' if it is already selected.
           */
        do
            s' ~ U(V_l);          // Draw a sample vertex.
        while s' ∈ {s_k}_{k=1}^N;
        s_k ← s';
        d' ← FittingDist((s_k)_{k=1}^N));
        if d' < d_min then
            d_min ← d';
            l_min ← l;
    return l_min, d_min;
Function FindVertex((s_k)_{k=1}^N, k, d_min, V):
    s'_min ← s_k;
    for s' ∈ V do
        s_k ← s';
        d' ← FittingDist((s_k)_{k=1}^N));
        if d' < d_min then
            d_min ← d';
            s'_min ← s';
    return s'_min, d_min;
d_min ← inf
for k = 1,...,K do
    s_k ← s_k^(0);          // Initialize with geodesic FPS.
for k = 1,...,K do
    l, d_min ← FindRegion((s_k)_{k=1}^N, k, d_min));
    s', d_min ← FindVertex((s_k)_{k=1}^N, k, d_min, V_l));
    s_k ← s';          // Update the k-th control point.
```

is used. To address this issue, we propose an effective search algorithm that reduces the time complexity to asymptotically linear order of the number of vertices $\Theta(N+K^2)$ in average. Despite this reduction in complexity, our algorithm still manages to discover nearly optimal solutions in practice.

In our search algorithm, our objective is to iteratively refine a set of control points starting from an initial configuration. We utilize geodesic Farthest Point Sampling (FPS) over surface of the mesh to establish the initial set. Our algorithm incorporates two key ideas:

- Drawing inspiration from the coordinate descent approach in continuous optimization, we propose to determine the optimal

location for each control point individually, while keeping all other current control points fixed.
- We propose a level-of-detail approach where at each iteration of updating a single control point, we select one of the partitioned volumes of the template mesh first and then traverse each vertex in the selected partition.

Specifically, let $(\mathbf{s}_k)_{k=1}^N$ denote the ordered list of template vertex indices for the control points, where $\mathbf{s}_k \in [1,N]$ for all k, and $\mathbf{s}_k \neq \mathbf{s}_l$ for all distinct k and l. Let $\mathbf{S}((\mathbf{s}_k)_{k=1}^N)$ then represent the $K \times N$ binary matrix, with elements equal to one for the selected points and zero otherwise. The indices of the control points are initialized with the FPS point indices $(\mathbf{s}_k^{(0)})_{k=1}^N$. We construct the partition of the vertex indices $\{V_k\}_{k=1}^N$ based on their proximity to the initial set of control points $(\mathbf{s}_k^{(0)})_{k=1}^N$, as shown inside the black boxes of Fig. 2. Since our algorithm allows the selection of internal vertices as control points, we employ the distance over the volume mesh graph as a measure of proximity. We update each control point $\mathbf{s}_k$ sequentially using the following two steps for each point. (See Alg. 1 for the details.)

In the first step, we determine the partition to which the k-th control point $\mathbf{s}_k$ will move. We randomly sample a vertex from each partitioned volume V_l. Then, we select the one of the sampled vertices that provides the minimum sum of distances between the template mesh and all the target shapes after deformation (as shown in Eq. 3) when substituting $\mathbf{s}_k$. By finding the vertex with the minimum sum of fitting distances, we identify the corresponding partitioned region V for further exploration. (See `FindRegion` function in Alg. 1.) This approach allows each control point to explore different regions across the entire shape, mitigating the risk of falling into a local minimum through local search.

In the second step, within the selected region V, we find the best vertex, excluding those already chosen as control points, as a replacement for the k-th control point $\mathbf{s}_k$ using the same distance measurement in Eq. 3. The vertex selected during this step becomes the new k-th control point for the subsequent iteration. (See the `FindVertex` function in Alg. 1.)

Assuming an even partitioning of the template mesh with an equal number of vertices in each region, the average time complexity to update the entire set of control points once is asymptotically $\mathcal{O}(K^2)$ for the first step and $\Theta(N)$ for the second step. Therefore, the total complexity is $\Theta(N+K^2)$. This complexity is linear with respect to the number of vertices, and since $K \ll N$, it significantly reduces the computation time compared to the exhaustive search complexity of $\mathcal{O}(N^K)$.

5. Experiments

In this section, we present the results of our experiments, where we compare the performance of our proposed method, OptCtrlPoints, with baseline search methods and a neural-network-based keypoint prediction method. We evaluate the performance based on the fitting error to the target shapes after deformation and the computation time.

5.1. Datasets

We evaluate our method on three different datasets of human (SMPL [LMR*15]) and animal (SMAL [ZKJB17] and DeformingThings4D [LTT*21]) models. For each class of shapes, we take one template mesh and multiple target shapes covering a wide range of non-rigid deformations.

5.1.1. SMPL [LMR*15] and SMAL [ZKJB17]

For human models, we use synthetic shapes generated from SMPL [LMR*15]. We use the samples generated by Groueix et al. [GFK*18], which contains a large variety of body poses and shapes. For animal models, we use four classes of shapes including *fox*, *hippo*, *horse*, and *tiger* generated from SMAL [ZKJB17]. We follow Groueix et al. [GFK*18] to randomly draw the shape samples. We run our searching algorithm on each animal category separately. For both SMPL and SMAL, we use the rest pose as the template shape and take $M = 1000$ random shapes for each template as targets $\{\mathcal{X}_i\}_{i=1}^M$.

5.1.2. DeformingThings4D [LTT*21]

DeformingThings4D [LTT*21] contains characters from Adobe Mixamo and also multiple animated motions for each character. We evaluate our approach both with *all* motions for *each* character and also with each motion to demonstrate the effectiveness of our *data-driven* method of locating the ideal set of control points best fitting the given set of targets. We use seven characters in our experiments, namely *bear*, *deer*, *doggie*, *dragon*, *moose*, *procy* and *raccoon*. For each character, we randomly sample $M = 1000$ different targets from all motions for the per-category experiments, while we use all the frames of the motion as targets for the per-motion experiments. We use the first frame of specific animation sequence as the template shape. Refer to the appendix for more details about the data used in our experiments.

5.2. Experiment Setup

5.2.1. Data Preprocessing and Implementation Details

For all template meshes, we first convert each mesh into a watertight manifold using the method of Huang et al. [HSG18] and then into a tetrahedral mesh using TetWild [HZG*18]. We simplify and regularize each tetrahedral mesh to have 5000 vertices and also normalize it to fit in a unit sphere. We then precompute the Bilaplacian matrix $\mathbf{A}$ for each template mesh using the libigl [JP*18] and and its inverse with the regularization described in Sec. 4.2.1 for efficient searching.

To compute $\mathbf{C}(\mathbf{S};\mathcal{X}_i) = \mathbf{S}f(\mathcal{X}_i)$ (Sec. 4.1), we leverage vertex-wise correspondence of the meshes provided from SMPL [LMR*15], SMAL [ZKJB17], and DeformingThings4D [LTT*21], while the correspondence can also be found using the off-the-shelf shape correspondence methods (refer to a survey [Sah20] for the recent literature). Given the vertex-wise correspondence, we use the average of per-vertex L2 distance as our shape-to-shape distance function $d(\cdot,\cdot)$ (Sec. 4.1). We executed Alg. 1 only once in all the experiments, but we also demonstrate in Sec. 5.6 that iterating the algorithm further improves the selection of control points.

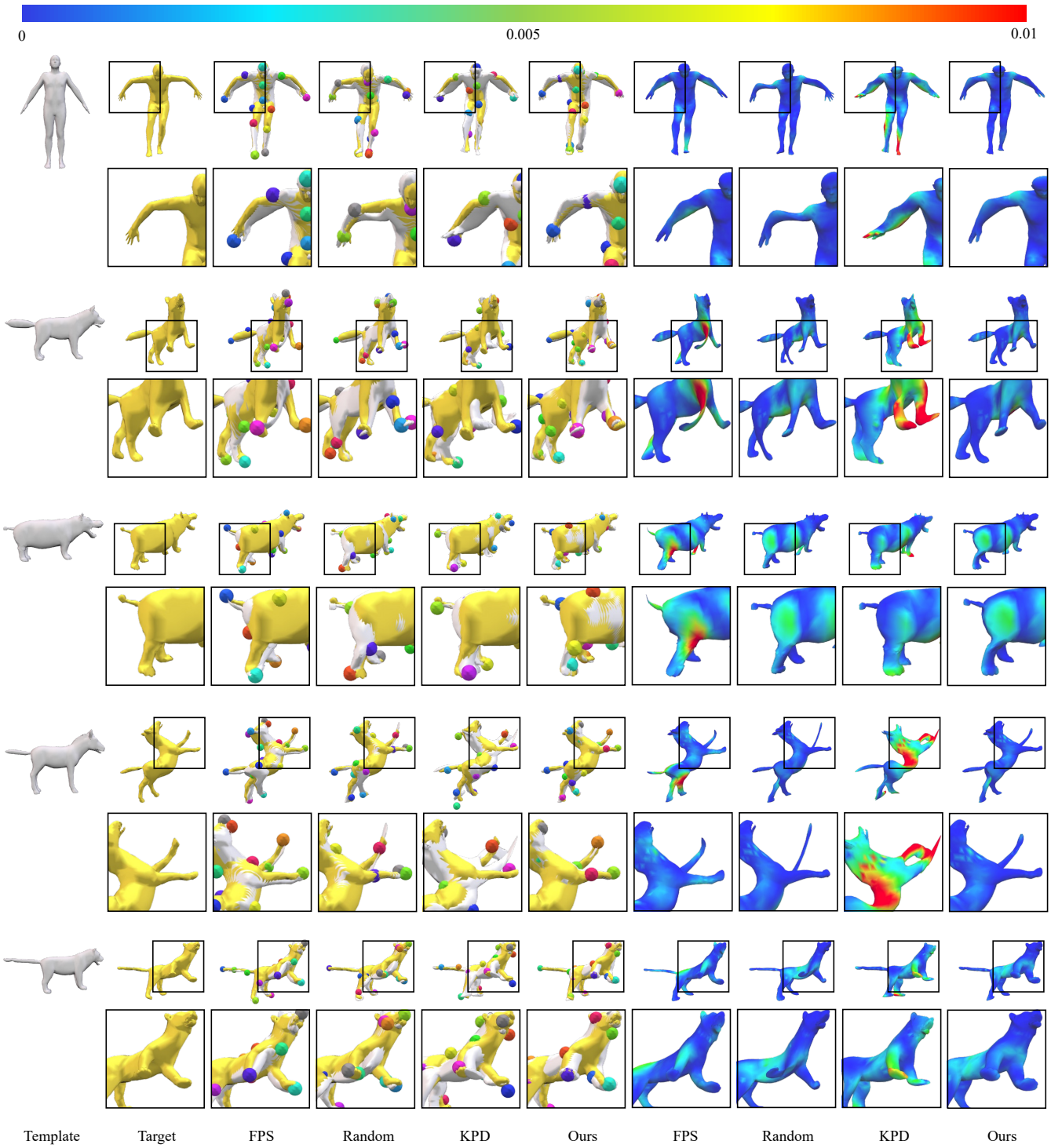

Figure 4: ***Qualitative Results on the SMPL [LMR*15] and SMAL [ZKJB17] datasets.*** *We show qualitative comparisons of our approach compared to FPS, random search and KPD, respectively. For each example: (Left) We show each method's output control points, the corresponding deformed template (white) overlayed over the desired target (yellow) to illustrate the alignment of the deformed source to the target shapes using the output control points. Notice that our approach finds better control points near joints as shown in the shoulder of the human, legs of the fox, horse, hippo and tiger. (Right) We also show the raw output deformed source shape colored with vertex-to-vertex alignment to target error map for visualization. We see that apart from achieving better fitting error, our approach achieves less distortions especially on the limbs. In all examples, the bottom row is a zoomed in version of the top row.* ***Best viewed in zoom and color. Refer to the appendix for additional results.***

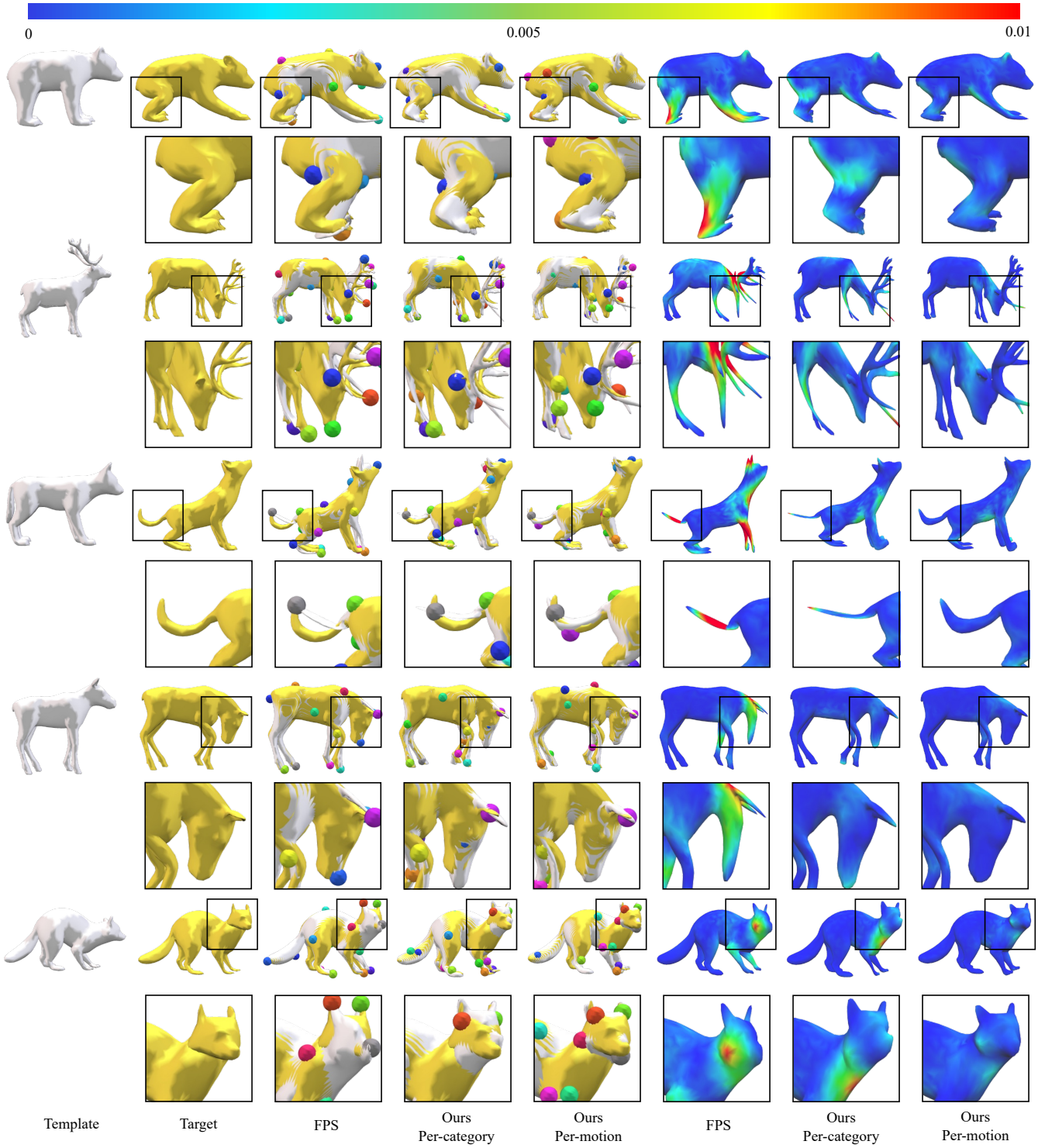

Figure 5: ***Qualitative Results on Deform4DThings [LTT*21].*** *We show qualitative comparisons of our approach in both per-category and per-motion settings compared to FPS. (Left) We show the output control points together with the corresponding deformed source shape (white) overlayed with the desired target (yellow). We see that our approach leads to better fitting compared to the FPS baseline. Moreover, our per-motion setting outputs more specialized control points that leads to better fitting to the specific motion, as shown by the legs of the bear (first row), head of the deer and raccoon (second and last row), tail of the doggie (third row), back and head of the horse (fourth row). (Right) We similarly also show a colored visualization of target shape that corresponds to the vertex-to-vertex error map. We see significantly less distortions of our output shapes compared to the baseline.* ***Best viewed in zoom and color. Refer to the appendix for additional results.***

Methods	K	Average Fitting Distance ($\times 10^{-4}$)				
		SMPL [LMR*15]	SMAL [ZKJB17]			
			fox	hippo	horse	tiger
KPD [JTM*20]	16	17.09	23.12	23.41	34.82	21.53
	24	17.16	25.63	23.65	28.42	22.92
	32	14.49	25.70	20.75	32.47	20.15
FPS	16	11.28	10.16	12.57	10.37	8.47
	24	5.70	4.41	8.32	3.92	3.76
	32	3.60	2.97	5.38	2.52	2.60
Random Search	16	8.38	8.18	9.20	7.88	6.78
	24	3.91	4.78	5.38	4.52	3.80
	32	2.51	3.40	3.63	2.97	2.59
Ours	16	**5.16**	**5.07**	**5.49**	**4.45**	**4.15**
	24	**2.25**	**2.61**	**3.06**	**2.13**	**2.21**
	32	**1.36**	**1.89**	**1.83**	**1.57**	**1.42**

Table 1: *Average fitting distance between corresponding vertices of target shape and deformed template shape multiplied by* 10^4.

5.2.2. Baselines

We compare our method, OPTCTRLPOINTS, with three different baselines:

1. **Farthest Point Sampling (FPS)**: This is the case of directly using the geodesic Farthest Point Sampling vertices, our initial set of control points $(\mathbf{s}_k^{(0)})_{k=1}^N$, as the final set without searching. We demonstrate in our experiments that our efficient searching method discovers a much better set of control points, which greatly reduces the fitting error.
2. **Random Search**: Instead of the computationally infeasible exhaustive search that tests all possible $\binom{N}{K}$ cases, we compare our method with a random search approach. In random search, we randomly select K control points from the N vertices multiple times and choose the set with the lowest fitting distance. We iterate the random set sampling process $N \times K$ times, which is significantly larger than the number of cases in our method.
3. **KeypointDeformer (KPD) [JTM*20]**: We additionally compare our method with KPD, which leverages a neural-network-based approach to predict keypoints on shapes and align the template to the target through cage-based deformation (not biharmonic). We trained KPD for each template shape and its corresponding set of M targets using the released codebase. The cage created by warping a *sphere* mesh often fails to disentangle the deformations of different parts, especially in shapes with articulating parts (such as limbs of a human body) or complex topological structures. In our experiments, we demonstrate that our method, employing biharmonic deformation, achieves a better fit of the template shape to the targets with the ideal set of control points.

5.3. Fitting Error Comparisons with SMPL [LMR*15] and SMAL [ZKJB17]

We evaluate our OPTCTRLPOINTS and other baselines by assessing how well the template mesh aligns with the target shapes using the output deformation handles. We conducted experiments with all the methods while varying the number of keypoints K to be 16, 24, and 32.

Methods	Time (mins)		
	$K = 16$	$K = 24$	$K = 32$
Random Search	48.8	70.4	97.0
Ours	**2.8**	**2.9**	**3.0**

Table 2: *Execution time profiling of our* OPTCTRLPOINTS *compared to random search. We show that our method demonstrates significantly faster performance compared to random search as our time complexity is reduced to* $\Theta(N + K^2)$*, whereas the time requirement increase linearly with the number of control points for random search.*

First, we compare our method to FPS and random search, where control points are used as handles for biharmonic deformation. Tab. 1 presents a comparison of the average fitting distances between the template and target shapes. Compared to FPS, where our initial set of control points is used directly, our method demonstrates a significant improvement, reducing the fitting distance by more than half in most cases. Additionally, when compared to random search, which explores a much larger number of control point sets, our efficient search yields substantially lower fitting errors.

Qualitative results in Fig. 4 also show that our method achieves more meaningful fine-grained deformations than FPS and random search, thanks to the optimal placement of control points in regions with greater variations or articulations. The first column shows the templates, the second column displays the targets, the next four columns demonstrate the alignment results through deformation, and the last four columns exhibit the fitting error maps over the deformed template shapes. Notably, failure cases of FPS and random search are observed, resulting in distortion in the deformation, such as in the *second row* where the legs of the fox are distorted, and in the *last row* where the legs and body of the tiger deviates from the target shapes. In contrast, our method's control points produce much better deformation results.

Furthermore, compared to KPD [JTM*20], which employs cage-based deformation instead of biharmonic deformation, our method excels in fitting the template to the targets through deformation. Tab. 1 clearly illustrates a substantial gap between the average fitting errors of KPD and our method. Moreover, Fig. 4 vividly showcases the qualitative difference, especially in the arms and legs of the human (*first row*) and the hind legs of the fox and the horse (*second* and *fourth rows*).

Figures are best viewed with zoom and in color. Additional qualitative results can be found in the supplementary material.

5.4. Computation Time Analysis

Our proposed reformulation for the biharmonic weights matrix computation (Sec. 4.2) and the efficient search algorithm (Sec. 4.3) enable us to determine the optimal control point locations in a matter of minutes, even when dealing with 1000 target shapes. Without both of these advancements, achieving this task computationally would be challenging.

Methods	K	Average Fitting Distance ($\times 10^{-4}$)													
		Bear (3EP)		Deer (OMG)		Doggie (MN5)		Dragon (OF2)		Moose (1DOG)		Procy (STEM)		Raccoon (VGG)	
		Cat.	Mot.	Cat.	Mot.	Cat.	Mot.	Cat.	Mot.	Cat.	Mot.	Cat.	Mot.	Cat.	Mot.
FPS	16	18.79	16.06	15.89	17.40	20.13	16.97	45.33	44.35	14.72	14.99	26.24	24.57	30.83	27.88
	24	8.78	7.72	5.33	5.59	7.48	6.24	26.35	26.31	6.66	6.57	15.24	14.40	15.68	15.65
	32	6.82	6.05	3.10	3.14	6.32	5.31	16.85	17.06	4.94	4.86	9.89	9.70	10.87	11.35
Random Search	16	16.17	12.59	8.35	7.14	14.76	11.27	21.57	19.91	14.04	12.25	20.16	17.86	27.75	24.99
	24	9.83	7.77	3.74	3.32	8.63	6.65	16.04	15.41	6.77	6.37	12.19	10.36	16.54	15.12
	32	5.95	5.05	2.54	2.31	5.86	4.56	14.02	13.50	4.52	4.17	7.82	6.92	11.96	11.23
Ours	16	9.67	**7.65**	5.23	**4.19**	8.72	**6.67**	17.80	**16.76**	6.62	**6.37**	13.14	**11.10**	16.13	**14.56**
	24	5.16	**4.06**	1.72	**1.56**	4.57	**3.54**	10.68	**10.31**	3.78	**3.51**	6.46	**5.50**	9.10	**8.42**
	32	3.47	**2.73**	1.28	**1.15**	2.95	**2.31**	9.29	**8.95**	2.74	**2.44**	4.35	**3.73**	5.80	**5.37**

Table 3: *Target-driven shape deformation results for DeformingThings4D [LTT*21] dataset. The control points identified by our* OPTCTRLPOINTS *method achieve better alignment of the template to the targets compared to FPS and random search. Moreover, when specific motion targets (per-motion, denoted as Mot.) are provided instead of general targets across all motions (per-category, denoted as Cat.), the control points are further tailored, resulting in even lower fitting distances. The L2-loss across corresponding vertices is multiplied by* 10^4.

Iteration	SMPL [LMR*15]		
	$K = 16$	$K = 24$	$K = 32$
1	5.14	2.31	1.34
2	4.77	1.95	1.17
3	4.74	1.80	1.15

Table 4: *Results from iterating Alg. 1. Fitting distance is improved through iterative execution of Alg. 1 on the SMPL dataset across different numbers of control points. The L2-loss across corresponding vertices is multiplied by* 10^4.

We first present profiling results comparing the computation time of the fitting loss (Eq. 3) using the original biharmonic weight formation (Eq. 2) and our reformulation (Eq. 12). When computing the fitting loss with 16 control points, utilizing a single NVIDIA RTX 3090 and parallelization, our PyTorch implementation takes 1.188 seconds for the original formulation, while our reformulation with precomputation completes the calculation in only 0.024 seconds, which is **49 times faster** than the original formulation.

We also demonstrate the efficiency of our method compared to the naive random search approach for finding a better solution. We conducted profiling to compare the overall execution time of our OPTCTRLPOINTS and random search. Tab. 2 presents the average runtime of OPTCTRLPOINTS compared to random search. In random search, we sample subsets of vertices $N \times K$ times, whereas our method has a linear order complexity with respect to the number of vertices. Consequently, we achieve approximately K times more speedup, as shown in Tab. 2, resulting in a significant reduction in computation time from about an hour to approximately **3 minutes**, while also obtaining a better set of control points (as demonstrated in the quantitative results in Tab. 1).

These findings emphasize the crucial role played by both the new biharmonic weight formation and the efficient search algorithm, as **without them, it would take days** to find a satisfactory set of control points. For instance, when the number of control points K is 16, $48.8\,\text{mins} \times 49 = 2391.2\,\text{mins} = 1.66\,\text{days}$.

5.5. Results with DeformingThings4D [LTT*21]

In the experiment with the DeformThings4D dataset, we highlight the effectiveness of our OPTCTRLPOINTS in discovering the optimal set of control points for a given set of targets in a *data-driven* manner. We present two distinct experimental setups: per-category and per-motion.

Quantitative results are presented in Table 3, where our approach outperforms both FPS and random search in both the per-category and per-motion setups. Notably, our OPTCTRLPOINTS consistently achieves a lower fitting error in the per-motion setup compared to the per-category setup. This outcome is attributed to our *data-driven* approach, which enables us to discover control points that are tailored to the given set of targets.

Fig. 5 presents qualitative results. In the *third row*, FPS fails to preserve the geometry of the dog's legs and tail, while our approach successfully retains the leg geometry during the howl motion by identifying control points on the joints. Moreover, our per-motion approach significantly outperforms the FPS baseline in scenarios involving larger motions and wider variations in the targets. For example, in the *second row*, our method achieves much better results in recovering the geometry of the deer's face and legs, whereas FPS falls short in this regard. Also, in the *fifth row*, our approach excels at capturing head deformations by identifying additional control points on the raccoon's head and neck.

Figures are best viewed with zoom and in color. Additional qualitative results can be found in the supplementary material.

5.6. Refinement through Iteration of Algorithm 1

We show that while executing Algorithm 1 only once can achieve desirable results, iterating through the algorithm can yield improved outcomes, as shown in Tab. 4. We see that through iterative execution of the Alg. 1, the average fitting distance is further reduced, leading to better results on the SMPL dataset across different numbers of control points.

6. Conclusion

We introduced OPTCTRLPOINTS, a data-driven method for determining the optimal set of control points to replicate target shapes as biharmonic deformations of the template mesh. To address the computational challenges associated with finding the best k-subset out of N vertices while solving a large-scale linear system at each trial, we proposed a reformulation of the biharmonic weights that significantly speeds up the computation. Additionally, we developed an efficient search algorithm that significantly outperforms random search in terms of both quality and time efficiency. In future work, we plan to extend our method to identify *region* handles of 3D shapes. This extension will allow us to handle more localized and specific deformations in a controlled and data-driven manner.

Acknowledgements. This work was partly supported by NRF grant (RS-2023-00209723) and IITP grant (2022-0-00594, RS-2023-00227592) funded by the Korean government (MSIT), Technology Innovation Program (20016615) funded by the Korea government(MOTIE), and grants from ETRI, KT, NCSOFT, and Samsung Electronics. Leonidas Guibas acknowledges support from an ARL grant W911NF-21-2-0104, a Vannevar Bush Faculty Fellowship, and gifts from the Adobe and Snap corporations. Despoina Paschalidou acknowledges support from the Swiss National Science Foundation under grant number P500PT 206946.

References

[APO21] ATTAIKI S., PAI G., OVSJANIKOV M.: DPFM: Deep partial functional maps. In *3DV* (2021). 4

[BP07] BARAN I., POPOVIĆ J.: Automatic rigging and animation of 3D characters. *ACM Transactions on Graphics* (2007). 2, 3

[BPGK06] BOTSCH M., PAULY M., GROSS M. H., KOBBELT L.: PriMo: coupled prisms for intuitive surface modeling. In *Eurographics Symposium on Geometry Processing* (2006). 3

[BPWG07] BOTSCH M., PAULY M., WICKE M., GROSS M.: Adaptive space deformations based on rigid cells. *Computer Graphics Forum* (2007). 3

[BS08] BOTSCH M., SORKINE O.: On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics* (2008). 3

[DSO20] DONATI N., SHARMA A., OVSJANIKOV M.: Deep Geometric Functional Maps: Robust feature learning for shape correspondence. In *CVPR* (2020). 4

[DYT21] DENG Y., YANG J., TONG X.: Deformed Implicit Field: Modeling 3D shapes with learned dense correspondence. In *CVPR* (2021). 3

[GFK*18] GROUEIX T., FISHER M., KIM V. G., RUSSELL B., AUBRY M.: 3D-CODED : 3D correspondences by deep deformation. In *ECCV* (2018). 7

[GFK*19] GROUEIX T., FISHER M., KIM V. G., RUSSELL B. C., AUBRY M.: Deep self-supervised cycle-consistent deformation for few-shot shape segmentation. In *Eurographics Symposium on Geometry Processing* (2019). 2, 3

[Hay68] HAYNSWORTH E. V.: *On the Schur complement*. Tech. rep., BASEL UNIV (SWITZERLAND) MATHEMATICS INST, 1968. 5

[HFW*18] HANOCKA R., FISH N., WANG Z., GIRYES R., FLEISHMAN S., COHEN-OR D.: ALIGNet: Partial-shape agnostic alignment via unsupervised learning. *ACM Transactions on Graphics* (2018). 3

[HLR*19] HALIMI O., LITANY O., RODOLA E., BRONSTEIN A. M., KIMMEL R.: Unsupervised learning of dense shape correspondence. In *CVPR* (2019). 4

[HS08] HORMANN K., SUKUMAR N.: Maximum entropy coordinates for arbitrary polytopes. *Computer Graphics Forum* (2008). 2, 3

[HSG18] HUANG J., SU H., GUIBAS L.: Robust watertight manifold surface generation method for shapenet models. *CoRR abs/1802.01698* (2018). 7

[HZG*18] HU Y., ZHOU Q., GAO X., JACOBSON A., ZORIN D., PANOZZO D.: Tetrahedral meshing in the wild. *SIGGRAPH* (2018). 7

[IMH05] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As-rigid-as-possible shape manipulation. In *SIGGRAPH* (2005). 3

[Jac14] JACOBSON A.: *Schur Complement Trick for Positive Semi-definite Energies*. Tech. rep., Columbia University, 2014. 5

[JBPS11] JACOBSON A., BARAN I., POPOVIC J., SORKINE O.: Bounded biharmonic weights for real-time deformation. In *SIGGRAPH Asia* (2011). 2, 3, 4

[JMD*07] JOSHI P., MEYER M., DEROSE T., GREEN B., SANOCKI T.: Harmonic coordinates for character articulation. In *SIGGRAPH* (2007). 2, 3

[JP*18] JACOBSON A., PANOZZO D., ET AL.: libigl: A simple C++ geometry processing library, 2018. https://libigl.github.io/. 7

[JPS*18] JACK D., PONTES J. K., SRIDHARAN S., FOOKES C., SHIRAZI S., MAIRE F., ERIKSSON A.: Learning free-form deformations for 3D object reconstruction. In *ICCV* (2018). 3

[JSW05] JU T., SCHAEFER S., WARREN J.: Mean value coordinates for closed triangular meshes. *ACM Transactions on Graphics* (2005). 2, 3

[JTM*20] JAKAB T., TUCKER R., MAKADIA A., WU J., SNAVELY N., KANAZAWA A.: KeypointDeformer: Unsupervised 3D keypoint discovery for shape control. In *CVPR* (2020). 2, 3, 10

[KJG*18] KURENKOV A., JI J., GARG A., MEHTA V., GWAK J., CHOY C. B., SAVARESE S.: DeformNet: Free-form deformation network for 3D shape reconstruction from a single image. In *WACV* (2018). 3

[KSSCO06] KRAEVOY V., SHEFFER A., SHAMIR A., COHEN-OR D.: Non-homogeneous resizing of complex models. In *SIGGRAPH Asia* (2006). 3

[LH13] LI X.-Y., HU S.-M.: Poisson coordinates. *IEEE Transactions on Visualization and Computer Graphics* (2013). 2, 3

[LLCO08] LIPMAN Y., LEVIN D., COHEN-OR D.: Green coordinates. In *SIGGRAPH* (2008). 2, 3

[LMR*15] LOPER M., MAHMOOD N., ROMERO J., PONS-MOLL G., BLACK M. J.: SMPL: A skinned multi-person linear model. In *SIGGRAPH Asia* (2015). 2, 7, 8, 10, 11

[LRR*17] LITANY O., REMEZ T., RODOLA E., BRONSTEIN A., BRONSTEIN M.: Deep Functional Maps: Structured prediction for dense shape correspondence. In *CVPR* (2017). 4

[LSC*04] LIPMAN Y., SORKINE O., COHEN-OR D., LEVIN D., ROSSI C., SEIDEL H. P.: Differential coordinates for interactive mesh editing. In *Shape Modeling Applications* (2004). 3

[LSKK22] LEE J., SUNG M., KIM H., KIM T.-K.: Pop-Out Motion: 3D-aware image deformation via learning the shape laplacian. In *CVPR* (2022). 3

[LSLCO05] LIPMAN Y., SORKINE O., LEVIN D., COHEN-OR D.: Linear rotation-invariant coordinates for meshes. In *SIGGRAPH* (2005). 3

[LSMS21] LIU M., SUNG M., MĚCH R., SU H.: DeepMetaHandles: Learning deformation meta-handles of 3D meshes with biharmonic coordinates. In *CVPR* (2021). 3

[LTT*21] LI Y., TAKEHARA H., TAKETOMI T., ZHENG B., NIESSNER M.: 4DComplete: Non-rigid motion estimation beyond the observable surface. In *ICCV* (2021). 2, 7, 9, 11, 13

[MRSHO22] MAGNET R., REN J., SORKINE-HORNUNG O., OVSJANIKOV M.: Smooth non-rigid shape matching via effective dirichlet energy optimization. In *3DV* (2022). 4

[OBCS*12] OVSJANIKOV M., BEN-CHEN M., SOLOMON J., BUTSCHER A., GUIBAS L.: Functional Maps: A flexible representation of maps between shapes. *ACM Transactions on Graphics* (2012). 4

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Eurographics Symposium on Geometry Processing* (2007). 3

[Sah20] SAHILLIOĞLU Y.: Recent advances in shape correspondence. *The Visual Computer* (2020). 4, 7

[SB09] SORKINE O., BOTSCH M.: Interactive shape modeling and deformation. In *Eurographics Tutorials* (2009). 3

[SCOL*04] SORKINE O., COHEN-OR D., LIPMAN Y., ALEXA M., RÖSSL C., SEIDEL H.-P.: Laplacian surface editing. In *Eurographics Symposium on Geometry Processing* (2004). 3

[SJA*20] SUNG M., JIANG Z., ACHLIOPTAS P., MITRA N. J., GUIBAS L. J.: DeformSyncNet: Deformation transfer via synchronized shape deformation spaces. In *SIGGRAPH Asia* (2020). 2, 3

[SP86] SEDERBERG T. W., PARRY S. R.: Free-form deformation of solid geometric models. In *SIGGRAPH* (1986). 3

[WBGH11] WEBER O., BEN-CHEN M., GOTSMAN C., HORMANN K.: A complex view of barycentric mappings. *Computer Graphics Forum* (2011). 2, 3

[WCMN19] WANG W., CEYLAN D., MECH R., NEUMANN U.: 3DN: 3D deformation network. In *CVPR* (2019). 2, 3

[WJBK15] WANG Y., JACOBSON A., BARBIC J., KAVAN L.: Linear subspace design for real-time shape deformation. *ACM Transactions on Graphics* (2015). 2, 3, 4

[WSLG07] WEBER O., SORKINE O., LIPMAN Y., GOTSMAN C.: Context-aware skeletal shape deformation. *Computer Graphics Forum* (2007). 2, 3

[YAK*20] YIFAN W., AIGERMAN N., KIM V. G., CHAUDHURI S., SORKINE-HORNUNG O.: Neural cages for detail-preserving 3D deformations. In *CVPR* (2020). 2, 3

[YM16] YUMER E., MITRA N. J.: Learning semantic deformation flows with 3D convolutional networks. In *ECCV* (2016). 3

[ZKJB17] ZUFFI S., KANAZAWA A., JACOBS D., BLACK M. J.: 3D Menagerie: Modeling the 3D shape and pose of animals. In *CVPR* (2017). 2, 7, 8, 10

[ZQZ*21] ZENG Y., QIAN Y., ZHU Z., HOU J., YUAN H., HE Y.: CorrNet3D: Unsupervised end-to-end learning of dense correspondence for 3D point clouds. In *CVPR* (2021). 4

[ZYDL21] ZHENG Z., YU T., DAI Q., LIU Y.: Deep implicit templates for 3D shape representation. In *CVPR* (2021). 3

A. Appendix

We include more dataset details in this section. Please refer additional qualitative results in the supplementary material.

A.1. DeformingThings4D [LTT*21] Data Used in Our Experiments

We select seven characters in our experiments namely *bear, deer, doggie, dragon moose, procy* and *raccoon*, which have more than 1,000 target shapes. For the per-motion experiments, we use all motions that have more than 100 target shapes for each character, which are namely:

- bear3EP: **Agression**, Drink, Eat1, Eat2, Hide, Idle1, Idle2, Idle3, Lie, Sleep
- deerOMG: **drink**, eat1, hide, Idle1, Idle2, Idle3, lie, sleep
- doggieMN5: **drink**, eat1, eat2, Howl, idle1, idle2, idle3, Lie
- dragonOF2: **act3**, act30, act31, act38, act46, act49, act57
- moose1DOG: **drink**, eat1, Idle1, Idle2, Idle3, lie, Liesleep
- procySTEM: **Actions2**, Idle1, Idle8, Idle9, Idle11, SleepLieSeat0, SleepLieSeat2
- raccoonVGG: **Actions1**, Climb15, Idle0, Idle2, Idle6, SleepLieSeat1

The template shape of each category is set to the first frame (bold) of the first animation in the above list.