SShaDe: scalable shape deformation via local representations

F. Maggioli[1], D. Baieri[2], Z. Lähner[3], S. Melzi[1]

[1]University of Milano-Bicocca, Italy
[2]Sapienza - University of Rome, Italy
[3]University of Bonn, Germany

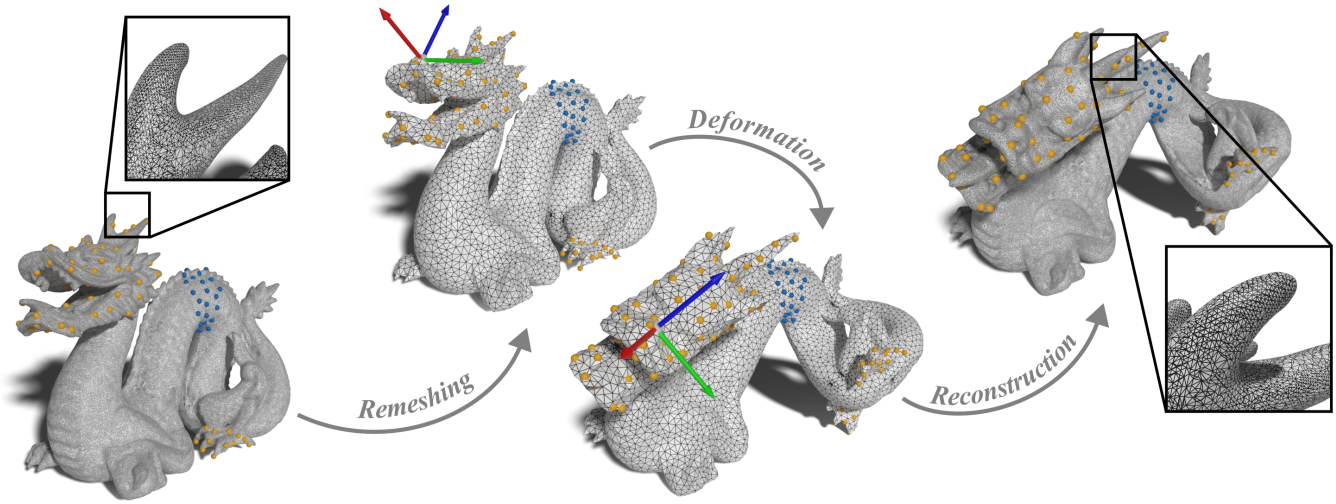


Figure 1: *A visual representation of our pipeline on handle-guided deformations. The input mesh (first) is remeshed to a lower resolution (second). The low resolution shape is then deformed (third), and the deformation of the original geometry (fourth) is reconstructed through a representation in a deformation-invariant local reference frame (second and third). Static handles are colored in blue, while dynamic handles are in yellow.*

Abstract

With the increase in computational power for the available hardware, the demand for high-resolution data in computer graphics applications increases. Consequently, classical geometry processing techniques based on linear algebra solutions are starting to become obsolete. In this setting, we propose a novel approach for tackling mesh deformation tasks on high-resolution meshes. By reducing the input size with a fast remeshing technique and preserving a consistent representation of the original mesh with local reference frames, we provide a solution that is both scalable and robust in multiple applications, such as as-rigid-as-possible deformations, non-rigid isometric transformations, and pose transfer tasks. We extensively test our technique and compare it against state-of-the-art methods, proving that our approach can handle meshes with hundreds of thousands of vertices in tens of seconds while still achieving results comparable with the other solutions.

CCS Concepts

• ***Computing methodologies*** → ***Shape modeling;*** *Computer graphics;* • ***Mathematics of computing*** → *Geometric topology;*

1. Introduction and related work

Deforming objects under the preservation of specific properties is one of the cornerstones of computer graphics, having countless applications in the field. Skeleton-based deformations [MLT88] are long-standing tools with widely applicability in surface animation. The simulation of deformable bodies is key in representing rubber-like materials [DBB11], cloth animations [BWD13], and natural processes and phenomena [MKH*23]. Shape deformation has also been successfully used in shape matching tasks [ELC19], and the interpolation of shapes has diverse useful applications in the field of geometry processing [SDGP*15]. Additionally, recent advances in neural fields and Gaussian splatting drew some attention on the deformation [YBHK21, BML*24], editing [PSBR24, WYZ*24], and simulation [ZYWL24] with these types of representations.

One of the most famous and widely used deformation types is as-rigid-as-possible (ARAP), in which the target configuration should be reached while preserving the local rigid structure at all the surface points as much as possible [SA07]. In the discrete setting, this translates to the simple constraint of preserving the edge lengths of the mesh. The powerful features that made ARAP deformations so popular are its simple formulation and the possibility of defining a small set of handle points, which can be used to drive the deformation. By combining the handle constraints and the local rigidity constraints, it is possible to obtain a visually plausible and natural looking global deformation. Over the years many applications and extensions to this energy have been proposed; from a smoothness prior [LG15] to ARAP regularization terms [HHS*21] and incorporation into neural fields [YSL*22, BML*24]. However, optimizing for an optimal global ARAP deformation does not scale well to high-resolution shapes. Consequently, these types of algorithm are unsuitable for meshes with an high vertex count – deforming meshes with more than 100k vertices with classical approaches can take several minutes on modern hardware.

A common workaround is to introduce a low-resolution reference graph on which the ARAP energy can be computed efficiently [ZSS97, HLX*20]. This leads to two challenges: 1) how to efficiently and accurately subsample the geometry while preserving the topology of the high-resolution mesh and 2) how to transfer the deformation computed on low-resolution vertices to the high-resolution geometry without introducing artifacts and errors. The task of obtaining a good sampling of the mesh is often a trade-off between efficiency and keeping correspondence information between the high- and low-resolution version. The front propagation method proposed by Peyré *et al.* [PC06] proved to be very effective in computing a farthest point sampling of the surface. Maggioli *et al.* [MBRM24] refined this method by constructing a sampling of the surface that induces a low-resolution triangle mesh which is topologically equivalent to the input shape. Obtaining a meaningful transfer of information from the low-resolution to the high-resolution is more challenging. For applications like avatar animation this is not a problem as a single deformation graph is sufficient [HLX*20] but for general shapes retaining this information is not trivial [JSZP20]. However, this information is crucial for transferring the low-resolution information to the high-resolution vertices – the high-resolution deformation should be obtained by mixing the deformations of the surrounding low-resolution vertices, but if not handled correctly, the interpolation might cause artifacts on boundaries and when the surface orientation changes. Solutions for this problem in multi-resolution settings have been proposed for specific energies [SYBF06, YSL*22] and data types [ZHS*05, YAK*20], but they are not suitable for a more general setting where one has to deal with arbitrary meshes. Another solution, more robust and general, has been proposed by Melzi *et al.* [MST*19] by using local reference frames at vertices. By using intrinsic quantities like surface normals and gradient vectors, the authors are able to define a local reference frame at each vertex that is invariant to isometric deformations of the shape. More similar in spirit to our work is the contribution from Morsucci *et al.* [MCS*18], where the authors propose the construction of a low-resolution graph connectivity on which edge constraints can be easily defined and efficiently solved. The paper also proposes an approach for extending the deformation by interpolating the rigid transformations of the control points via a distance-based weighting. While this method proves to be both efficient and effective, it is only suitable for deformations that are defined via edge lengths, excluding a variety of possible applications.

With this paper, we presents a novel flexible approach to efficiently compute deformations through a low-resolution mesh and apply them to the original high-resolution shape. We leverage the fast remeshing of [MBRM24] which provides a correspondence between the high- and low-resolution versions and perform an accurate reconstruction of geometric details by moving consistent local reference frames. Due to the knowledge of the correspondences this can be done very efficiently. Our experiments show a speed-up of 50x-100x in comparison to other ARAP implementations while keeping comparable metric scores. The method can also be used to tightly align isometrically deformed shapes and perform pose transfer between different classes.

Contributions. Our contributions can be summed up as follows:

- A highly efficient framework for a broad range of shape deformation tasks that works by transferring detailed rigid geometry through local reference frames in a lower resolution.
- An implementation that is 50 to 100 times faster than default ARAP deformation, which we applied to meshes with resolution up to 450k vertices with runtime performance of 10 seconds or less.
- Finally, we propose the first dataset for handle-guided deformations, automatically built from the well-known FAUST dataset by upsampling the original shapes and deriving motion handles from the ground truth correspondence.

2. Background and notation

Triangular meshes. We discretize a surface as a triangular mesh $\mathcal{M} = (V_{\mathcal{M}}, E_{\mathcal{M}}, T_{\mathcal{M}})$ embedded in $\mathbb{R}^3$, where: $V_{\mathcal{M}} \subset \mathbb{R}^3$ is a set of vertices in the 3D space; $T_{\mathcal{M}} \subset V_{\mathcal{M}}^3$ is a set of triangular faces among vertices (invariant under even permutations); $E_{\mathcal{M}} \subset V_{\mathcal{M}}^2$ is a set of edges induced by the triangles (*i.e.*, $(v_i, v_j, v_k) \in T_{\mathcal{M}} \implies (v_i, v_j) \in E_{\mathcal{M}}$).

Manifoldness. Through this paper, we assume our meshes to be 2-manifold. This means that we don't allow a mesh $\mathcal{M}$ to have

non-manifold edges (*i.e.*, edges that are incident on three or more triangles), nor non-manifold vertices (*i.e.*, vertices that are incident on two or more fans of triangles). We allow our meshes to have boundaries, that is to have some edges that are incident on a single triangle, but we assume them to be composed by a single connected component.

Deformation. When we refer to a deformation of a mesh $\mathcal{M} = (V_{\mathcal{M}}, E_{\mathcal{M}}, T_{\mathcal{M}})$, we indicate the creation of a second mesh $\mathcal{M}' = (V_{\mathcal{M}'}, E_{\mathcal{M}'}, T_{\mathcal{M}'})$ such that each vertex $v' \in V_{\mathcal{M}'}$ is obtained by altering the 3D position of a corresponding vertex $v \in V_{\mathcal{M}}$, but leaving unchanged the connectivity (*i.e.*, $E_{\mathcal{M}'} = E_{\mathcal{M}}$ and $T_{\mathcal{M}'} = T_{\mathcal{M}}$). When the correspondence between vertices of a mesh $\mathcal{M}$ and its deformation $\mathcal{M}'$ is not implied in the text, we explicitly refer to it as a bijective function $\pi : V_{\mathcal{M}} \to V_{\mathcal{M}'}$ that preserves the connectivity isomorphism between the two meshes. Namely

$$(v_i, v_j) \in E_{\mathcal{M}} \iff (\pi(v_i), \pi(v_j)) \in E_{\mathcal{M}'}\,, \tag{1}$$

$$(v_i, v_j, v_k) \in T_{\mathcal{M}} \iff (\pi(v_i), \pi(v_j), \pi(v_k)) \in T_{\mathcal{M}'}\,. \tag{2}$$

Intrinsic quantities. We use the term *geodesic* as a shorthand for indicating a geodesic shortest path, and the term *geodesic triangle* for indicating a convex region on a surface enclosed by the geodesics connecting three distinct surface points. Scalar functions over the surface are represented as functions $f : V_{\mathcal{M}} \to \mathbb{R}$ assuming real values at the vertices, while tangent vector fields $F : T_{\mathcal{M}} \to \mathbb{R}^3$ are defined on triangles. The surface normals are also defined at triangles, and for each triangle t the orientation of its normal n_t is defined by the orientation of its vertices using the right-hand rule. The gradient is an operator $\nabla : \mathcal{F}(\mathcal{M}, \mathbb{R}) \to \mathcal{F}(\mathcal{M}, \mathbb{R}^3)$ from the space $\mathcal{F}(\mathcal{M}, \mathbb{R})$ of scalar functions over $\mathcal{M}$ to the space $\mathcal{F}(\mathcal{M}, \mathbb{R}^3)$ of vector fields over $\mathcal{M}$. The gradient $\nabla f(t)$ of a scalar function f at a triangle t returns the tangent vector at t pointing towards the direction of the steepest increase in f.

3. Method

Our method achieves efficiency by optimizing a deformation on a low-resolution mesh but all information can be transferred accurately back to the original resolution. To that end, we first apply a topology-preserving remeshing to obtain a low-resolution version of the original shape while still preserving the overall geometry, see Section 3.1. We then apply some deformation pipeline to the low-resolution mesh. Finally, we use a robust and conservative local reference frame (Section 3.2) to reconstruct the original geometry in the deformed pose, see Section 3.3. An overview of our method can be found in Figure 1.

3.1. Remeshing

For the remeshing step, we require a procedure that can efficiently reduce the vertex count by orders of magnitude, while still preserving the original topology and providing robustness guarantees about the preservation of the underlying geometry. Maggioli *et al.* [MBRM24] proposed a remeshing algorithm that obtains a farthest point sampling of the surface and constructs an intrinsic Delaunay triangulation from the dual Voronoi decomposition. The method associates each vertex v and triangle t of the remeshed shape to, respectively, a Voronoi region R_v and a geodesic triangle t_g on the high-resolution mesh. Given the original mesh $\mathcal{M} = (V_{\mathcal{M}}, E_{\mathcal{M}}, T_{\mathcal{M}})$ and its remesh $\mathcal{N} = (V_{\mathcal{N}}, E_{\mathcal{N}}, T_{\mathcal{N}})$, the method also builds an approximately bijective linear mapping $\mathbf{U} \in \mathbb{R}^{V_{\mathcal{N}} \times V_{\mathcal{M}}}$ for upsampling scalar functions over $\mathcal{N}$ to scalar functions over $\mathcal{M}$. Nonetheless, the map is built using a raw projection of the vertices $V_{\mathcal{M}}$ onto the nearest surface points of $\mathcal{N}$. This operation has a $\mathcal{O}(|V_{\mathcal{M}}||T_{\mathcal{N}}|)$ cost, which can quickly become inefficient as the vertex count of $\mathcal{M}$ increases.

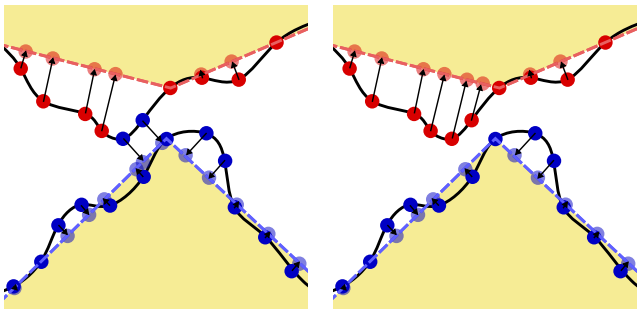
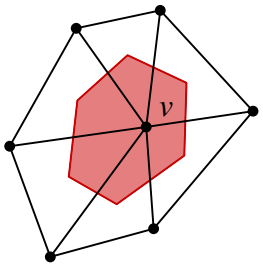


Figure 2: *The vertices of two Voronoi regions on the high-resolution shape (solid) are projected onto the low-resolution representation (dashed, shaded in yellow) with the approach from [MBRM24] (left) and our method (right). The color of the vertices (dark red or dark blue) reveals in which part of the low-resolution surface they are projected (respectively, light red or light blue).*

To solve this issue and further reduce the cost of the projection operation, we exploit the fact that each vertex $v \in V_{\mathcal{N}}$ is associated with a Voronoi region $R_v \subset V_{\mathcal{M}}$ onto $\mathcal{M}$, and that R_v is a topological 2-disk. In this scenario, the original region R_v must be represented by the dual Voronoi area of v on $\mathcal{N}$, meaning that it is contained inside the triangle fan around v (*i.e.*, the dual Voronoi area of v is the region of the mesh which is closer to v than to any other vertex, as shown in red the inset figure). Thus, for each vertex $v \in V_{\mathcal{N}}$, we project the region R_v onto the nearest surface points of the triangle fan around v. Statistically, the number of triangles incident on a vertex on a manifold mesh is always about 6-7, meaning that the projection operation for a Voronoi region has a $\mathcal{O}(|R_v|)$ cost. By summing across all the regions, we get



$$\sum_{v \in V_{\mathcal{N}}} \mathcal{O}(|R_v|) = \mathcal{O}(|V_{\mathcal{M}}|)\,, \tag{3}$$

achieving a significant improvement in the cost of the projection operation.

Furthermore, this novel approach for computing the projection of $\mathcal{M}$ onto $\mathcal{N}$ has the additional desired property of mitigating artifacts and errors in the projection. As shown in the example from Figure 2, the original method proposed in [MBRM24] does not account for regions of the surface that are geodesically far away, but close in the embedding space. Potentially, this situation results in

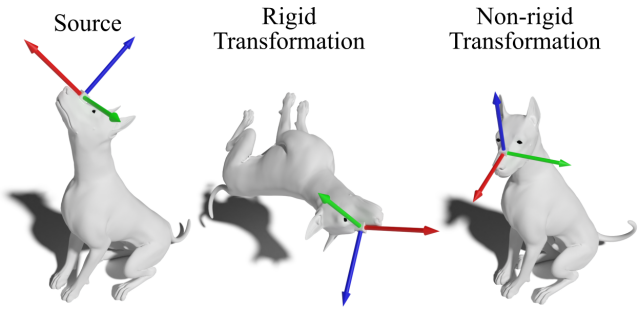


Figure 3: *An example of local reference frame at a triangle (left) and how it is affected by a rigid transformation (center) and a non-rigid deformation (right) of the surface. The blue vector is always aligned with the surface normal, the red vector points towards the snout, and the green vector towards the left of the dog's head.*

an incorrect mapping of the vertices of $\mathcal{M}$ onto $\mathcal{N}$. On the other hand, with our approach we ensure that the vertices of each Voronoi region of $\mathcal{M}$ are mapped into the correct region of $\mathcal{N}$.

3.2. Local reference frame

Given a mesh $\mathcal{M} = (V_{\mathcal{M}}, E_{\mathcal{M}}, T_{\mathcal{M}})$, the remeshing process produces a mesh $\mathcal{N} = (V_{\mathcal{N}}, E_{\mathcal{N}}, T_{\mathcal{N}})$, which represents $\mathcal{M}$ at a lower resolution, and a mapping $P : V_{\mathcal{M}} \to T_{\mathcal{N}}$ associating each vertex of $\mathcal{M}$ to the triangle of $\mathcal{N}$ it is projected onto. Since each triangle $t \in T_{\mathcal{N}}$ represents a geodesic triangle t_g onto $\mathcal{M}$, by building a local reference frame at t, we can completely represent the geometry of t_g in terms of local coordinates in t. Let $\mathcal{N}' = (V_{\mathcal{N}'}, E_{\mathcal{N}}, T_{\mathcal{N}})$ a connectivity-preserving deformation of the shape $\mathcal{N}$ (notice that edges and triangles are the same as $\mathcal{N}$), which may be obtained through an as-rigid-as-possible deformation, manual editing, or any other suitable deformation pipeline. If the local reference frame at triangle t is consistent across deformations, we can use it to reconstruct the global coordinates of the geodesic triangle t_g after the deformation, effectively deforming the original mesh $\mathcal{M}$ into a mesh $\mathcal{M}' = (V_{\mathcal{M}'}, E_{\mathcal{M}}, T_{\mathcal{M}})$. A visual representation of how local reference frames are deformed consistently through shapes is shown in Figure 3. As shown in the figure, the three basis vector always represent the same intrinsic directions and the same semantics across the shapes (*e.g.*, the blue vector is always the surface normal and the red vector always points toward the snout).

Since we want our local reference frame to be consistent across deformations and variations of pose, we must build it using intrinsic measures and quantities. For this task, we adapt the approach proposed by Melzi *et al.* [MST*19]. In their paper, the authors propose to use the normal vectors and the normalized gradient of some pose-invariant scalar function $f : \mathcal{N} \to \mathbb{R}$ on the surface (*e.g.*, the Laplacian eigenfunctions) to obtain a local reference frame at the vertices for shape correspondence tasks; the third vector can be easily obtained via a cross product between the other two. Differently from [MST*19], our setting provides us with a twofold advantage:

- we need our local reference frame to be defined over triangles rather than vertices;
- the correspondence between $\mathcal{N}$ and $\mathcal{N}'$ is already known.

Algorithm 1 Deformation transfer via a local reference frame.

procedure LRFT($\mathcal{M}, \mathcal{N}, \mathcal{N}', f : V_{\mathcal{N}} \to \mathbb{R}, P : V_{\mathcal{M}} \to T_{\mathcal{N}}$)
  for $t \in T_{\mathcal{N}}$ **do**
    $g_t, g'_t \leftarrow$ gradient of f at t on $\mathcal{N}$ and $\mathcal{N}'$, respectively
    $n_t, n'_t \leftarrow$ normal at t on $\mathcal{N}$ and $\mathcal{N}'$, respectively
    $w_t \leftarrow g_t \times n_t, \quad w'_t \leftarrow g'_t \times n'_t$
    $b_t, b'_t \leftarrow$ barycenter of t in $\mathcal{N}$ and $\mathcal{N}'$, respectively
    $\mathbf{L}_t \leftarrow (g_t \;\; w_t \;\; n_t), \quad \mathbf{L}'_t \leftarrow (g'_t \;\; w'_t \;\; n'_t)$
  end for
  $V_{\mathcal{M}'} \leftarrow \varnothing$
  for $v \in V_{\mathcal{M}}$ **do**
    $t \leftarrow P(v)$
    $v_t \leftarrow \mathbf{L}_t^{-1}(v - b_t)$
    $v' \leftarrow \mathbf{L}'_t v_t + b'_t$
    $V_{\mathcal{M}'} \leftarrow [V_{\mathcal{M}'}, v']$
  end for
  return $\mathcal{M}' = (V_{\mathcal{M}'}, E_{\mathcal{M}}, T_{\mathcal{M}})$
end procedure

Since triangles are the natural domain for normal and gradient vector, working on the mesh triangle allows us to reduce the number of operations to perform and improve numerical stability. Furthermore, given the correspondence between $\mathcal{N}$ and $\mathcal{N}'$ we can transfer any function accurately, thus enabling the usage of simple functions such as the mesh coordinates rather than pose-invariant functions, which are notoriously costly to compute and usually not robust on shapes at low-resolution. Employing gradient vectors as basis vectors for local reference frames requires caution, as by the Poincaré-Hopf ("Hairy-Ball") Theorem, not all surface topologies guarantee a non-null tangent vector field everywhere. In cases where we obtain tangent vectors with a very small magnitude at some triangles, we can introduce a second distinct function $f_2 : \mathcal{N} \to \mathbb{R}$ on the surface. For each triangle t where $\nabla f(t)$ has a very small magnitude, we can replace the basis vector with with $\nabla f_2(t)$. However, as mentioned in [MST*19], in the discrete setting this situation occurs rarely, and we never experienced similar cases in our experiments.

After this process, for each triangle t in the mesh $\mathcal{N}$, we obtain a matrix $\mathbf{L}_t$ defined as

$$\mathbf{L}_t = \begin{pmatrix} g_t & w_t & n_t \end{pmatrix}, \tag{4}$$

where n_t is the normal vector at triangle t, g_t is the gradient vector at t, and $w_t = g_t \times n_t$ is the cross product between the other two. By calling b_t the barycenter of the triangle t, the representation of a vertex $v \in V_{\mathcal{M}}$ in the local reference frame of t is given by

$$v_t = \mathbf{L}_t^{-1}(v - b_t). \tag{5}$$

Similarly, we can build a matrix $\mathbf{L}'_t$ representing the local reference frame of the same triangle t, but in $\mathcal{N}'$. The reconstruction of the global coordinates for the deformed vertex $v' \in V_{\mathcal{M}'}$ can be obtained through

$$v' = \mathbf{L}'_t v_t + b'_t. \tag{6}$$

The entire procedure is summarized in Algorithm 1, which ac-

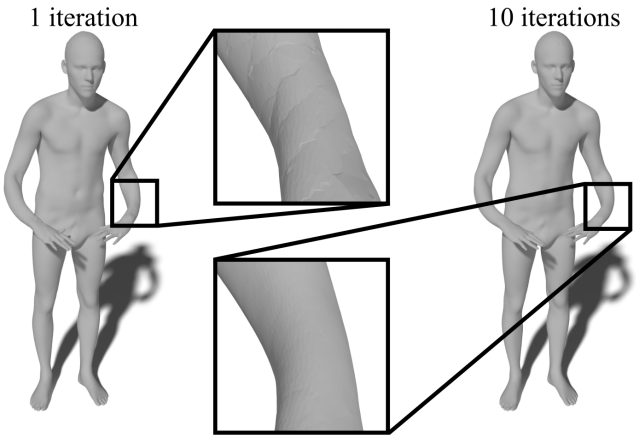


Figure 4: *Comparison between the direct application of our approach (left) and the averaging of multiple iterations (right). The closeups on the arms show the difference in smoothness between the two approaches.*

cepts in input the high-resolution shape $\mathcal{M}$, its remesh $\mathcal{N}$, the deformed remesh $\mathcal{N}'$, a scalar function f over $\mathcal{N}$, and the projection map P of the vertices of $\mathcal{M}$ onto the triangle of $\mathcal{N}$. The first **for** loop (lines 2-8) computes the local reference frames of all the triangles in both the remeshed shape $\mathcal{N}$ and its deformation $\mathcal{N}'$. These local reference frames are used in the second **for** loop (lines 10-15) for representing the vertices of $\mathcal{M}$ in local coordinates and deforming them according to $\mathcal{N}'$.

3.3. Refinement

The approach we described so far has a major drawback due to the linearity of the local reference frames. Indeed, representing a local reference frame as an affine 3D transformation allows for fast computation, but it also limits the representational power of the deformation. Each geodesic triangle t_g on $\mathcal{M}$ associated with a triangle $t \in T_{\mathcal{N}}$ is deformed according to the same linear transformation induced by the local reference frame. As a result, the deformed mesh at high-resolution presents visual artifacts and lines along the edges of the geodesic triangles, as can be seen in Figure 4. In order to mitigate this effect, we apply multiple iterations of our process using different samplings for computing the low-resolution mesh, resulting in different intrinsic triangulations of $\mathcal{M}$. Each of these iterations produces a slightly different version of the final mesh $\mathcal{M}'$, mostly differing only for the placement of the artifact lines. By averaging all these versions of $\mathcal{M}'$, we obtain a final result where the artifacts are smoothed out, but which still preserves all the details of the original surface. While in our experiments we notice that 5-10 iterations are generally enough, we acknowledge that estimating an optimal number of iterations could be worth of future investigations. We stress that, as each individual iteration is independent, the process can be easily implemented in a multi-threaded environment, averaging all the results at the end and achieving an almost perfect parallelism.

4. Results

We validate the performance of our method by testing its performance on handle-guided deformation tasks against state-of-the-art solutions. We also evaluate the pipeline on different types of deformations, to prove its applicability in different contexts. Finally, we show that our approach can be exploited for pose transfer tasks.

Our method is implemented in C++, using as backends the Eigen library for linear algebra routines [GJ*10] and the libigl library for basic geometric operations [JP*18]. All the experiments are performed on a machine equipped with a CPU Intel i7-10700K and 32GB of RAM.

4.1. As-rigid-as-possible deformations

As-rigid-as-possible (ARAP) deformations represent a core task in the field of geometry processing but most existing solutions are known to suffer from poor scalability to high-resolution meshes. To evaluate the performance of our pipeline in this setting, we compare it against the original ARAP implementation (ARAP) [SA07], the smooth rotations variant (ARAP-SR) [LG15], and the spokes and rims method (Elastic) [CPSS10]. The latter is also the method we use for deforming our low-resolution shapes, as it is the method that gives the best trade-off between quality and computational cost. Additionally, we consider the recent work from Baieri *et al.* (Implicit-ARAP) [BML*24], as it provides an efficient and more scalable solutions to previous approaches while still achieving comparable quality. Finally, we did not consider the work from Morsucci *et al.* [MCS*18] for the evaluation, as there is no publicly available implementation. For our experiments, we use some of the user-defined deformations used in [BML*24], which are based on models from the Stanford's 3D Scanning Repository (S3D) [sta]. Moreover, we test all the competitors on the DeFAUST dataset (see Appendix A for an in-depth presentation of the dataset), a novel dataset that we propose to validate accuracy and efficiency in this task on high-resolution meshes. This evaluation enables us to provide a quantitative comparison that was completely missing on this task and thus is an additional concrete contribution.

For evaluating the results, we follow the approach from [BML*24]. Given the original surface $\mathcal{M}$ and the deformed surface $\mathcal{M}'$, we consider the relative surface error E_{area} and volume error E_{vol}

$$E_{\text{area}} = \frac{|A_{\mathcal{M}} - A_{\mathcal{M}'}|}{A_{\mathcal{M}}}, \quad E_{\text{vol}} = \frac{|V_{\mathcal{M}} - V_{\mathcal{M}'}|}{V_{\mathcal{M}}}, \tag{7}$$

where $A_{\mathcal{S}}$ and $V_{\mathcal{S}}$ are, respectively, the surface area and the volume of a watertight surface $\mathcal{S}$. We also evaluate the distortion induced on the input geometry by considering the edge length error and the face angle error. Namely, the edge length error is the average difference in length between the same edge before and after the deformation, normalized by the maximum edge length. This can be computed as

$$E_{\text{edge}} = \frac{1}{|E_{\mathcal{M}}|} \sum_{(u,v) \in E_{\mathcal{M}}} \frac{|\|u - v\| - \|\pi(u) - \pi(v)\||}{\max_{e \in E_{\mathcal{M}}} \|e\|}, \tag{8}$$

where $\pi : V_{\mathcal{M}} \to V_{\mathcal{M}'}$ is the point-to-point correspondence between the vertices of the original mesh $\mathcal{M}$ and its deformation $\mathcal{M}'$.

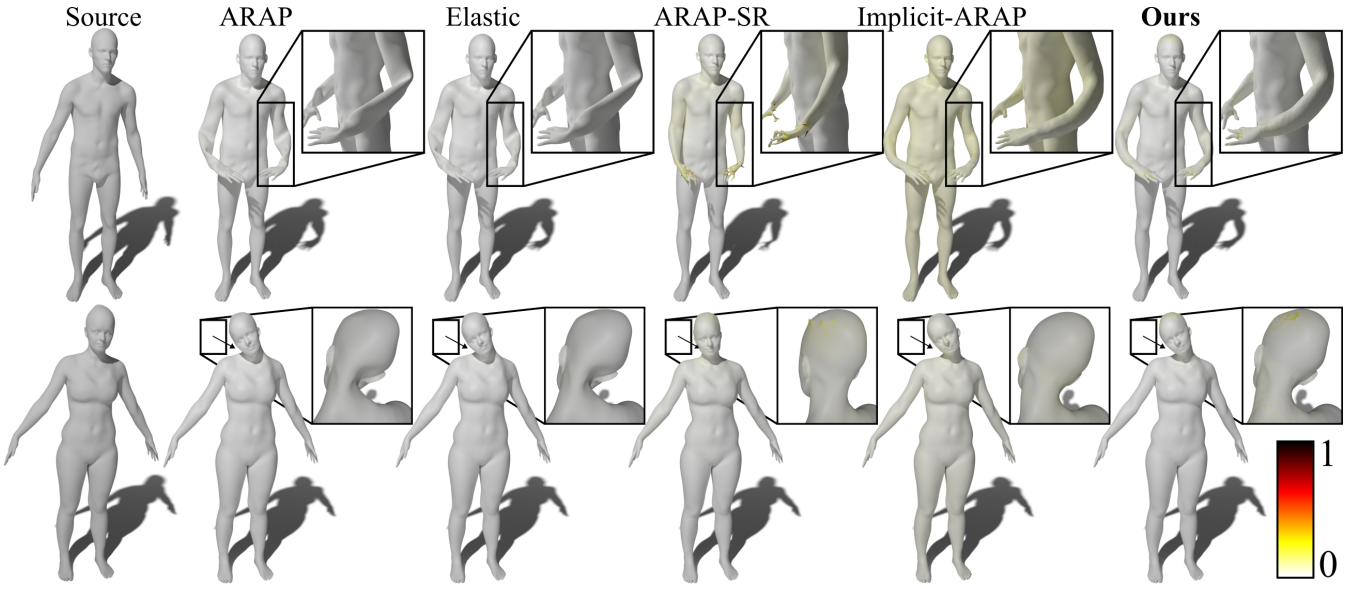


Figure 5: *Comparison of our solution with the baseline approaches on sample deformations from the DeFAUST dataset. The close-ups show the deformation artifacts, and the edge error as in Equation (8) is shown as a hot colormap.*

Method	Volume	Area	EL	FA	Time
ARAP	12.41%	**0.45%**	**0.37%**	**0.517°**	7m:15s
Elastic	12.25%	**0.45%**	**0.38%**	**0.523°**	8m:27s
ARAP-SR	11.69%	5.36%	3.39%	5.628°	**7m:11s**
Implicit-ARAP	**0.99%**	0.82%	1.06%	1.517°	8m:22s
Ours	**2.28%**	**0.59%**	1.28%	2.148°	**0m:8s**

Table 1: *Average errors and execution times for our method and the considered baseline approaches on the DeFAUST dataset. The angle error (FA) is expressed in degrees, while the edge length error (EL), the surface error, and the volume error are in percentage. For each column, the best and second best results are highlighted in light blue and light green, respectively.*

Similarly, the face angle error is the average difference between the same internal angles of the triangles before and after the deformation, which is given by

$$E_{\text{angle}} = \frac{1}{3|T_{\mathcal{M}}|} \sum_{t \in T_{\mathcal{M}}} \sum_{v \in t} \frac{|\angle(v,t) - \angle(\pi(v),t)|}{\angle(v,t)}, \tag{9}$$

where $\angle(v,t)$ is the internal angle of t at vertex v.

The results of our experiments on the DeFAUST dataset are summarized in Table 1 where we show the error metrics and the runtime averaged across all the 40 high-resolution examples in the dataset. The experiments are run until convergence with a cutoff of 1000 iterations. For our method, we remesh the original surface to about 6000 vertices, averaging the result across 5 parallel iterations (as discussed in Section 3.3). The configuration parameters of Implicit-ARAP are set to the values suggested by the authors in their paper. The experiments show that our approach can achieve comparable results to the other baseline approaches in terms of surface error, edge length error, and face angle error. However, with our method we can compute the deformation at a fraction of the computational cost, obtaining a 50x-60x speedup. Notably, both Implicit-ARAP and our solution achieve significantly better results in terms of volume error. This behavior can be better appreciated in the examples from Figure 5, which also shows the ability of our method to keep the per-edge error low. We speculate that this outcome on volume preservation comes from an intrinsic issue of ARAP and its variants, as the local rigidity constraints only translate to useful volume preservation in a low resolution setting. Clearly, this is not an issue for continuous implicit methods such as Implicit-ARAP, while our method may be seen specifically as a way to account for this issue, since we perform the deformation in a low-resolution representation of the original input. In support to this argument, we highlight that the volume errors for the baselines in the same exact set of experiments on the low-resolution FAUST shapes are much more competitive (ARAP: 1.28%, Elastic: 1.29%, ARAP-SR: 1.21%).

Lastly, the examples in Figure 6 depict a comparison between our method and the other baselines on some user-defined deformations from the original Implicit-ARAP paper. As shown in the figure, with our approach we are able to obtain results that are more visually plausible with few seconds of computation, instead of several minutes.

4.2. Non-rigid isometric deformations

Our pipeline is not bound to ARAP deformations only: as long as the deformation does not change the number of vertices and their connectivity, our method can be applied to other types of deformations as well. In this regard, we design a set of experiments to test the performance of our pipeline when tackling arbitrary non-rigid isometric deformations. We use the TOSCA dataset [BBK08]

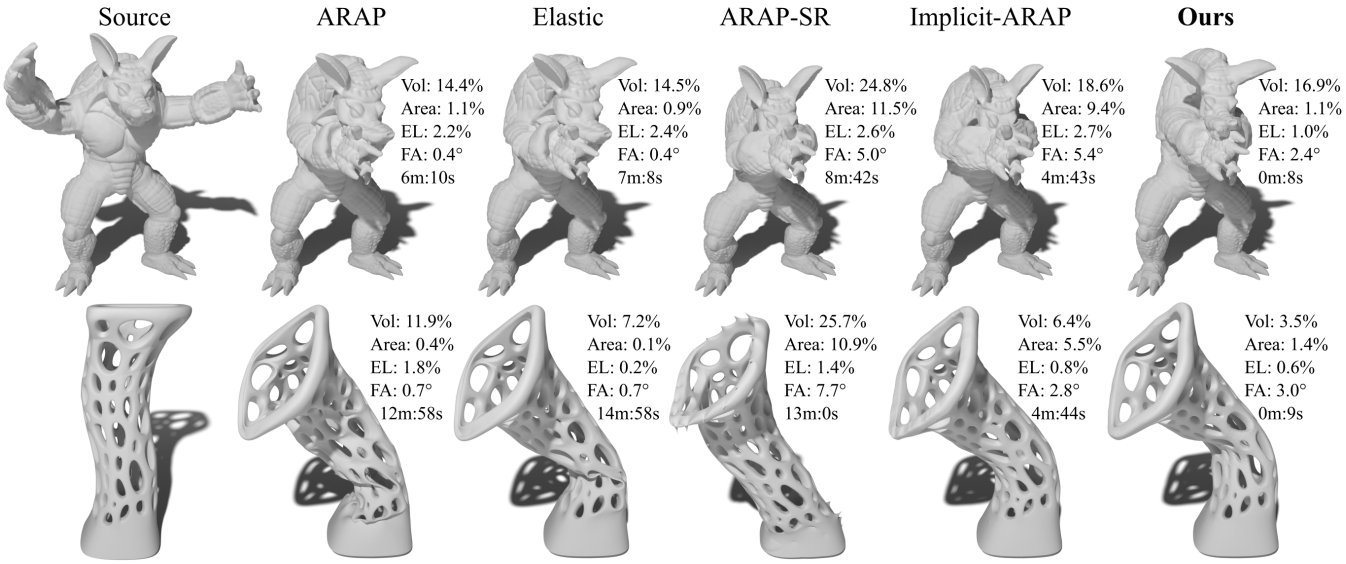


Figure 6: *Comparison of our solution with the baseline approaches on some user-defined deformations from [BML*24].*

which is composed by synthetic meshes representing 9 human and animal subjects in several different poses, including a neutral rest pose. By using the provided correspondence and starting from the rest pose, we reconstruct all the other poses from a low-resolution deformation. Given the rest pose $\mathcal{M} = (V_{\mathcal{M}}, E_{\mathcal{M}}, T_{\mathcal{M}})$ of a subject and another arbitrary pose $\mathcal{M}' = (V_{\mathcal{M}'}, E_{\mathcal{M}'}, T_{\mathcal{M}'})$ of the same subject, we apply the following procedure:

1. we remesh $\mathcal{M}$ to a low-resolution version $\mathcal{N} = (V_{\mathcal{N}}, E_{\mathcal{N}}, T_{\mathcal{N}})$;
2. since $V_{\mathcal{N}}$ is a subset of $V_{\mathcal{M}}$, we use the correspondence provided by the TOSCA dataset to obtain the corresponding vertices $V_{\mathcal{N}'}$ in the target pose;
3. we infer the remesh of the target pose as $\mathcal{N}' = (V_{\mathcal{N}'}, E_{\mathcal{N}}, T_{\mathcal{N}})$ using the same connectivity as $\mathcal{N}$;
4. using the representation of $\mathcal{M}$ in the local reference frame of the triangles of $\mathcal{N}$, we transfer the local coordinates onto $\mathcal{N}'$ and reconstruct the deformed mesh $\mathcal{M}'$.

The approach is summarized by the diagram in Figure 7a. In Figure 8 we provide visual comparisons between the original meshes from the TOSCA dataset and those reconstructed with our approach. Here we remesh the surfaces to about 6000 vertices and average the result across 10 deformations. We apply this pipeline to the entire TOSCA dataset, comparing the original TOSCA meshes and our reconstruction. We evaluate the results by computing the Hausdorff distance d_H and the Chamfer distance d_C, which are defined as

$$d_H(\mathcal{X},\mathcal{Y}) = \max\left(\max_{x\in V_{\mathcal{X}}} d(x,\mathcal{Y}),\ \max_{y\in V_{\mathcal{Y}}} d(y,\mathcal{X})\right), \quad (10)$$

$$d_C(\mathcal{X},\mathcal{Y}) = \frac{1}{|V_{\mathcal{X}}|}\sum_{x\in V_{\mathcal{X}}} d^2(x,\mathcal{Y}) + \frac{1}{|V_{\mathcal{Y}}|}\sum_{y\in V_{\mathcal{Y}}} d^2(y,\mathcal{X}), \quad (11)$$

where $d(v,\mathcal{S})$ represents the Euclidean distance between the point v and the surface $\mathcal{S}$. For a meaningful comparison, we center all

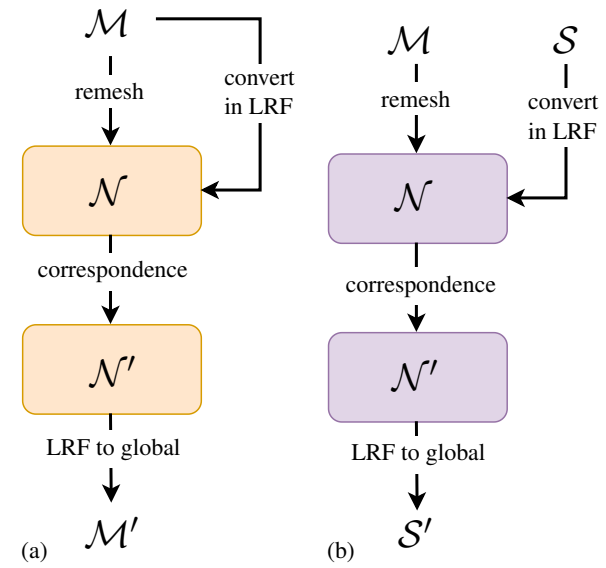


Figure 7: *The pipelines used for transferring the non-isometric deformations from the TOSCA dataset (a) and for the pose transfer tasks on the manually edited meshes (b).*

the shapes at the origin of the axes and rescale them to fit inside the unit sphere. Furthermore, we exclude from the experiments the `wolf` shape for its low vertex count (less than 5000 vertices) and the `gorilla` shape for its non-manifoldness and high number of connected components. By averaging the results across the entire dataset, we obtain a Hausdorff distance $d_H = 5.66 \cdot 10^{-3}$ and a Chamfer distance $d_C = 3.76 \cdot 10^{-8}$. Figure 8 shows how tight the alignment is.

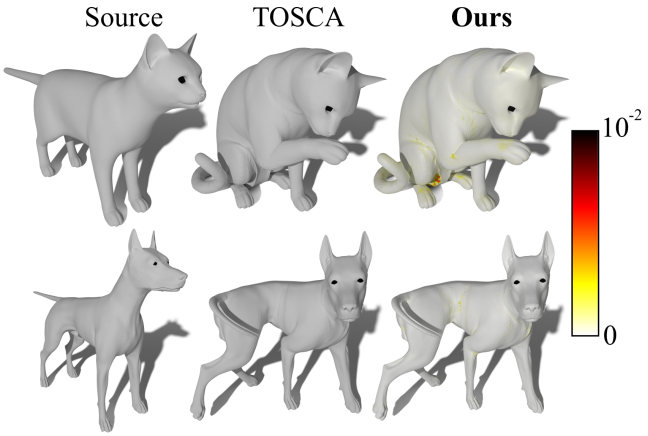


Figure 8: *From left to right: a mesh from the TOSCA dataset representing a subject in its rest pose; a mesh from the TOSCA dataset representing the same subject in another pose; the mesh obtained by deforming the rest pose with our pipeline. The Euclidean distance between the deformed mesh and the nearest surface point of the TOSCA mesh is shown as a function on the surface.*

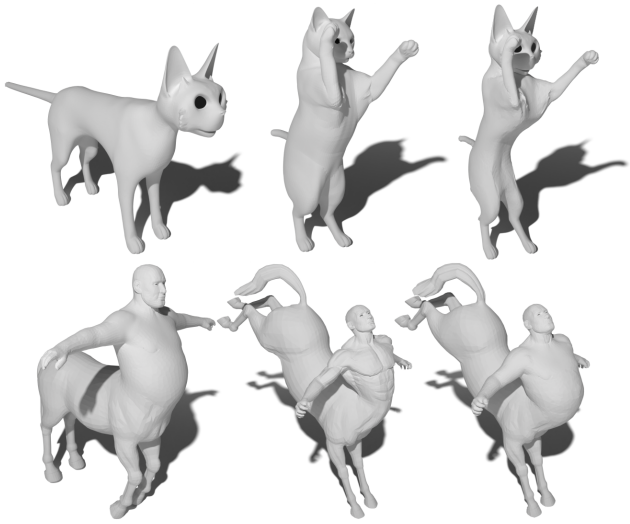


Figure 9: *From left to right: a mesh obtained from the rest pose of a TOSCA mesh by manual editing; a mesh from the TOSCA dataset; the manually edited mesh deformed with our method to the pose of the TOSCA mesh.*

4.3. Pose transfer

Another possible use-case scenario for our method is the transfer of the pose of a shape into the one of another shape. To demonstrate the effectiveness of our method for this task, we manually edit the rest pose of the `centaur` mesh and the `cat` mesh from the TOSCA dataset to obtain two new meshes that are semantically similar to the original but present different features. Notice that the new meshes are in one-to-one correspondence with the original shapes, share the same connectivity, and are aligned in 3D space. Given the rest pose $\mathcal{M}$ from the TOSCA dataset, the edited mesh $\mathcal{S}$ in the rest pose, and a mesh $\mathcal{M}'$ representing the subject $\mathcal{M}$ in a different pose, we want to compute a deformation $\mathcal{S}'$ of $\mathcal{S}$ representing the edited subject in the pose provided by $\mathcal{M}'$. We remesh the original mesh $\mathcal{M}$ into a low-resolution shape $\mathcal{N}$, and follow the process summarized by Figure 7b. The procedure is similar to the one we describe in Section 4.2 but instead of computing the local representation of $\mathcal{M}$ onto $\mathcal{N}$, we use the one-to-one correspondence between $\mathcal{M}$ and $\mathcal{S}$ to compute the local coordinates of the vertices of $\mathcal{S}$ onto the triangles of $\mathcal{N}$. Then, we reconstruct the mesh $\mathcal{S}'$ representing the edited mesh $\mathcal{S}$ into the target pose provided by $\mathcal{M}'$.

In Figure 9 we show two examples where we deform our edited `cat` and `centaur` meshes from the rest pose to another pose defined by a mesh representing the corresponding original subject. Despite some minor visual artifact, we achieve visually plausible results in ~2 seconds of computation for each example. Additionally, as mentioned in Section 3, our method allows us to define the local reference frames onto the triangles, instead of the vertices. By performing the same experiment onto the full-resolution meshes and defining the local reference frames at the vertices (as in [MST*19]) we obtained meshes of inferior visual quality, showing the additional benefits in using our pipeline.

5. Conclusions

We introduced a novel pipeline for efficiently tackling the problem of deforming high-resolution meshes. We exploited a scalable remeshing algorithm to accurately represent a geometry at a lower resolution and improved the existing approach for mapping vertices at high-resolution onto the low-resolution shape. By taking advantage of the correspondence between a mesh and its deformed counterpart, we implemented an efficient local reference frame at triangles using it for representing and deforming the original high-resolution geometry. We validated our approach on multiple sets of data, including a novel dataset we introduced for as-rigid-as-possible deformation tasks. Our extensive set of experiments demonstrates the efficiency and efficacy of our approach in multiple settings, including ARAP deformations, non-rigid isometric transformations, and pose transfer tasks.

In the future, we plan exploring other possible applications of our pipeline, like shape matching and other types of mesh editing tasks. We also intend to explore the possibility of softening the requirements of our method, like the assumption of manifoldness and the graph isomorphism constraint between the low-resolution mesh before and after the deformation. Finally, we mean to probe other approaches for representing the high-resolution geometry with local descriptors, eventually increasing their expressive power and enabling more drastic deformations like topological changes.

References

[BBK08] BRONSTEIN A., BRONSTEIN M., KIMMEL R.: *Numerical Geometry of Non-Rigid Shapes*. Springer, New York, NY, 2008. 6

[BGS10] BOYÉ S., GUENNEBAUD G., SCHLICK C.: Least squares subdivision surfaces. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 2021–2028. 10

[BML*24] BAIERI D., MAGGIOLI F., LÄHNER Z., MELZI S., RODOLÀ E.: Implicit-arap: Efficient handle-guided deformation of high-resolution meshes and neural fields via local patch meshing. *arXiv preprint arXiv:2405.12895* (2024). 2, 5, 7

[BRLB14] BOGO F., ROMERO J., LOPER M., BLACK M. J.: FAUST: Dataset and evaluation for 3D mesh registration. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (Piscataway, NJ, USA, June 2014), IEEE. 9

[BWD13] BENDER J., WEBER D., DIZIOL R.: Fast and stable cloth simulation based on multi-resolution shape matching. *Computers & Graphics 37*, 8 (2013), 945–954. 2

[CPSS10] CHAO I., PINKALL U., SANAN P., SCHRÖDER P.: A simple geometric model for elastic deformations. *ACM Trans. Graph. 29*, 4 (jul 2010). doi:10.1145/1778765.1778775. 5

[DBB11] DIZIOL R., BENDER J., BAYER D.: Robust real-time deformation of incompressible surface meshes. In *Proceedings of the 2011 ACM SIGGRAPH/eurographics symposium on computer animation* (2011), pp. 237–246. 2

[ELC19] EISENBERGER M., LÄHNER Z., CREMERS D.: Divergence-free shape correspondence by deformation. In *Computer Graphics Forum* (2019), vol. 38, Wiley Online Library, pp. 1–12. 2

[GJ*10] GUENNEBAUD G., JACOB B., ET AL.: Eigen v3. http://eigen.tuxfamily.org, 2010. 5

[HDD*93] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Mesh optimization. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (1993), pp. 19–26. 10

[HHS*21] HUANG Q., HUANG X., SUN B., ZHANG Z., JIANG J., BAJAJ C.: Arapreg: An as-rigid-as possible regularization loss for learning deformable shape generators. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021). 2

[HLX*20] HABERMANN M., LIU L., XU W., ZOLLHOEFER M., PONS-MOLL G., THEOBALT C.: Real-time deep dynamic characters. *Transactions on Graphics (Proc. of SIGGRAPH)* (2020). 2

[JP*18] JACOBSON A., PANOZZO D., ET AL.: libigl: A simple C++ geometry processing library, 2018. https://libigl.github.io/. 5

[JSZP20] JIANG Z., SCHNEIDER T., ZORIN D., PANOZZO D.: Bijective projection in a shell. *ACM Transactions on Graphics (TOG) 39*, 6 (2020). 2

[LG15] LEVI Z., GOTSMAN C.: Smooth rotation enhanced as-rigid-as-possible mesh animation. *IEEE Transactions on Visualization and Computer Graphics 21*, 2 (2015), 264–277. doi:10.1109/TVCG.2014.2359463. 2, 5

[MBRM24] MAGGIOLI F., BAIERI D., RODOLÀ E., MELZI S.: Rematching: Low-resolution representations for scalable shape correspondence. In *European Conference on Computer Vision* (2024), Springer. 2, 3

[MCS*18] MORSUCCI A., CENTIN M., SIGNORONI A., ET AL.: Fast centroidal deformation for large mesh models. In *Italian Chapter Conference 2018-Smart Tools and Apps in Computer Graphics, STAG 2018* (2018), Eurographics Association, pp. 97–106. 2, 5

[MKH*23] MAGGIOLI F., KLEIN J., HÄDRICH T., RODOLÀ E., PAŁUBICKI W., PIRK S., MICHELS D. L.: A physically-inspired approach to the simulation of plant wilting. In *SIGGRAPH Asia 2023 Conference Papers* (2023), pp. 1–8. 2

[MLT88] MAGNENAT T., LAPERRIÈRE R., THALMANN D.: Joint-dependent local deformations for hand animation and object grasping. In *Proceedings of Graphics Interface'88* (1988), Canadian Inf. Process. Soc, pp. 26–33. 2

[MST*19] MELZI S., SPEZIALETTI R., TOMBARI F., BRONSTEIN M. M., STEFANO L. D., RODOLA E.: Gframes: Gradient-based local reference frame for 3d shape matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 4629–4638. 2, 4, 8

[PC06] PEYRÉ G., COHEN L. D.: Geodesic remeshing using front propagation. *International Journal of Computer Vision 69* (2006), 145–156. 2

[PSBR24] PALANDRA F., SANCHIETTI A., BAIERI D., RODOLÀ E.: Gsedit: Efficient text-guided editing of 3d objects via gaussian splatting. *arXiv preprint arXiv:2403.05154* (2024). 2

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing* (2007), pp. 109–116. 2, 5

[SDGP*15] SOLOMON J., DE GOES F., PEYRÉ G., CUTURI M., BUTSCHER A., NGUYEN A., DU T., GUIBAS L.: Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (ToG) 34*, 4 (2015), 1–11. 2

[sta] The stanford 3d scanning repository. https://graphics.stanford.edu/data/3Dscanrep/. 5

[SYBF06] SHI L., YU Y., BELL N., FENG W.-W.: A fast multigrid algorithm for mesh deformation. *ACM Trans. Graph. 25*, 3 (2006). 2

[WYZ*24] WU T., YUAN Y.-J., ZHANG L.-X., YANG J., CAO Y.-P., YAN L.-Q., GAO L.: Recent advances in 3d gaussian splatting. *Computational Visual Media 10*, 4 (2024), 613–642. 2

[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

[YBHK21] YANG G., BELONGIE S., HARIHARAN B., KOLTUN V.: Geometry processing with neural fields. In *Advances in Neural Information Processing Systems* (2021), Ranzato M., Beygelzimer A., Dauphin Y., Liang P., Vaughan J. W., (Eds.), vol. 34, Curran Associates, Inc., pp. 22483–22497. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/bd686fd640be98efaae0091fa301e613-Paper.pdf. 2, 10

[YSL*22] YUAN Y.-J., SUN Y.-T., LAI Y.-K., MA Y., JIA R., GAO L.: Nerf-editing: geometry editing of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 18353–18364. 2

[ZHS*05] ZHOU K., HUANG J., SNYDER J., LIU X., BAO H., GUO B., SHUM H.-Y.: Large mesh deformation using the volumetric graph laplacian. *ACM Trans. Graph. 24*, 3 (2005). 2

[ZSS97] ZORIN D., SCHRÖDER P., SWELDENS W.: Interactive multiresolution mesh editing. *Transactions on Graphics (Proc. of SIGGRAPH)* (1997). 2

[ZYWL24] ZHONG L., YU H.-X., WU J., LI Y.: Reconstruction and simulation of elastic objects with spring-mass 3d gaussians. *European Conference on Computer Vision (ECCV)* (2024). 2

Appendix A: DeFAUST dataset

To the best of our knowledge, no dataset exists for quantitative evaluations on deformation methods. For a more exhaustive validation of our method, we introduce the **De**forming **FAUST** (DeFAUST) dataset, which we derive from the well-known FAUST dataset for shape matching [BRLB14]. The FAUST dataset contains 3D scans of real humans in various poses. Among these shapes, 10 subjects in 10 different poses (including a rest pose for each subject) are registered to a template with about 7000 vertices, meaning that all the 100 meshes are in one-to-one correspondence and share the same connectivity.

For each subject, we select the rest pose as the starting pose for the deformation and 4 target poses from which we extract the deformation handles. We also handpick some landmarks which we use for driving the handle extraction. Figure 10 depicts the rest pose

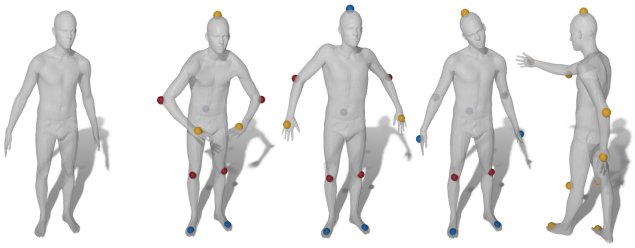


Figure 10: *from left to right: the rest pose used in the dataset as the starting position for the deformation; the four target poses used for handle extarction. The landmarks used for the handle generation are color-coded: blue for static handles; yellow for dynamic handles; red for unused handles.*

and the 4 selected poses, as well as the landmarks that we use for the deformation. Starting from each landmark ℓ on the rest pose $\mathcal{M}$, we select 30 random vertices in a neighborhood of ℓ, obtaining the set H_ℓ of handles that drive the deformation of the region associated with the landmark. Then, by using the provided correspondence $\pi : \mathcal{M} \to \mathcal{M}'$ between $\mathcal{M}$ and another target pose $\mathcal{M}'$, we select the handles $H'_\ell = \{\pi(v) : v \in H_\ell\}$ on $\mathcal{M}'$. We compute the difference

$$\Delta_\ell = \sum_{v \in H_\ell} \|v - \pi(v)\|, \tag{12}$$

and if Δ_ℓ is smaller than some threshold, we mark the positions of the handles H_ℓ as static. Otherwise, we mark them as dynamic handles, using the corresponding positions in H'_ℓ as their target positions. For the last pose on the right of Figure 10 we use more landmarks. Differently from the other poses, this pose is not aligned in 3D space with the rest pose, and more information could be needed to solve for a meaningful deformation.

As a result, we obtain a dataset containing 10 different human shapes and 4 sets of deformation handles for each shape. The deformation handles are provided as 3D positions instead of vertex indices, so that they can be used with remeshed and subdivided shapes. Additionally, since we want our dataset to be used also for the evaluation of deformation algorithms on high-resolution meshes, we produce an high-resolution version of the 10 rest pose shapes using the LS3 Loop algorithm [BGS10] to obtain synthetic shapes with an high vertex count. As noted by Yang *et al.* [YBHK21], most deformation algorithms (especially those designed for as-rigid-as-possible deformation tasks) do not perform well on grid-like triangulations like Marching Cubes meshes or shapes obtained by means of regular subdivisions. For this reason, we apply a step of isotropic remeshing as in [HDD*93] to the subdivided surfaces, obtaining meshes with a vertex count in the range [150k, 200k].