Computers & Graphics 53 (2015) 156–165

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

A shape deformation algorithm for constrained multidimensional scaling ☆

Yusuf Sahillioğlu *

Department of Computer Engineering, Middle East Technical University, Universiteler Mah. Dumlupinar Blv. No:1 06800, Ankara, Turkey

ARTICLE INFO

Article history:
Received 26 May 2015
Received in revised form
14 August 2015
Accepted 2 October 2015
Available online 23 October 2015

Keywords:
Detail-preserving MDS
High-resolution models
Canonical form
Sparse linear system
Retrieval
Segmentation

ABSTRACT

We present a new Euclidean embedding technique based on volumetric shape registration. Extrinsic representation of the intrinsic geometry of a shape is preferable in various computer graphics applications as it poses only a small degrees of freedom to deal with during processing. A popular Euclidean embedding approach to achieve such a representation is multidimensional scaling (MDS), which, however, distorts the original geometric details drastically. Our method introduces a constraint on the original MDS formulation in order to preserve the initial geometric details while the input shape is pulled towards its MDS pose using the perfectly accurate bijection in between. The regularizer of this registration framework is chosen in such a way that the system supports large deformations yet remains fast. Consequently, we produce a detail-preserving MDS pose in 90 s for a 53 K-vertex high-resolution mesh on a modest computer. We can also add pairwise point constraints on the deforming shape without any additional cost. Detail-preserving MDS is superior for non-rigid shape retrieval and useful for shape segmentation, as demonstrated.

1. Introduction

The geometry of the objects we deal with in computer graphics is non-Euclidean, that is, the distance between two points is a curvy path along the object surface instead of a flat line through the Euclidean space. In various computer graphics applications such as correspondence and classification, Euclidean geometry is preferable due to its smaller number of degrees of freedom to compare two objects, consisting only of rotation, translation, and uniform scaling. It is therefore plausible to embed the objects into the Euclidean space, which equips them with the desired Euclidean geometry.

It is, however, impossible to compute this embedding in the Euclidean space without introducing distortions (Section 3.1.1). We consequently added appealing features to our embedded poses without worrying too much about the inevitable distortion. In particular, we want to (i) preserve the geometric details and (ii) enable pairwise point constraints, while maintaining the original fast system designed for the embedding operation. The latter feature is a relatively new one in this context, whereas the former is shown to be advantageous for the shape retrieval application [1]. Our method outperforms [1] in terms of accuracy as well as execution time. We also show another advantage of the former feature in the context of shape segmentation.

The main idea of our algorithm is to exploit the bijection from the mesh model of the object to its embedded representation by means of a shape registration process. The matching term of the registration that pulls the mesh towards the distorted embedding pose of interest competes with the regularization terms that try to maintain the original geometric details. Additional pairwise point constraints can also be added to this framework efficiently. Thanks to the simple closed-form of our registration model, all our algorithm boils down to iteratively solving a sparse linear system, which makes it quite fast even with the high-resolution models, e.g., 90 s for the detail-preserved embedding computation of a 53 K-vertex mesh.

2. Contributions

- Computing a canonical pose for 3D models which preserves details and pairwise constraints.
- Combining as-rigid-as possible (ARAP) deformation energy with Tikhonov regularizer to handle large deformations with even less tetrahedra inversions.
  - Slight modification on ARAP using mesh fairing.
- Clear closed-forms of the (well-known) components of our deformation model with their derivations.

☆This paper was recommended for publication by Shi-Min Hu

* Tel.: +90 312 210 5563, +90 312 210 5544.
E-mail address: ysahillioglu@gmail.com

http://dx.doi.org/10.1016/j.cag.2015.10.003
0097-8493/

- Smooth blending of these components leading to an efficient linear solution system.

- A basic shape segmentation application and promising results on non-rigid shape retrieval that outperform some of the existing robust works.

We note that the source code, executable, and video for the method that we present in this paper will be made public.

3. Related work

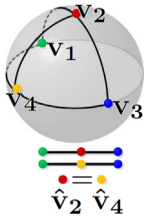
Our work aims to create a special pose, namely a detail-preserving canonical pose. We therefore investigate the related work on the creation of *special* poses in two subsections, first of which discusses canonical poses comprehensively in a survey fashion, whereas the second one focuses on the poses that are designed for purposes like beautification and fabrication.

3.1. Survey on canonical poses

Shape embedding is essentially the process of obtaining a *canonical* pose that can be used as an extrinsic representation of the intrinsic non-Euclidean geometry of an object. We discuss how the objects equipped with non-Euclidean geometry are embedded into a different domain where they gain invariance against the ubiquitous isometric deformations, such as rotation, translation, and bending transformations after which the pairwise distances along the surface of the object are preserved. An embedding is Euclidean or non-Euclidean if the embedding space is Euclidean, e.g., $\mathbb{R}^3$, or non-Euclidean, e.g., sphere, respectively. The distance in the definition of isometry is generally the geodesic distance which is more intuitive and accurate than the diffusion-based counterpart whose main advantage is robustness to the topological noise [2].

3.1.1. Euclidean embedding

When each pair of points on an object is embedded into the Euclidean space in such a way that the paired points are separated by an amount equal to the corresponding geodesic distance on the original shape, this Euclidean embedding is impossible to be distortionless, as illustrated at right with an example from [3].



After a distortionless Euclidean embedding of the sphere vertices $\mathbf{v}_{\{1,2,3,4\}}$ into the Euclidean space $\mathbb{R}^m$ for arbitrary m, the embedded vertices $\hat{\mathbf{v}}_{\{1,2,3,4\}}$ has $\|\hat{\mathbf{v}}_1-\hat{\mathbf{v}}_2\|_{L_m}=\|\hat{\mathbf{v}}_2-\hat{\mathbf{v}}_3\|_{L_m}=1$ and $\|\hat{\mathbf{v}}_1-\hat{\mathbf{v}}_3\|_{L_m}=2$, making the triangle $(\hat{\mathbf{v}}_1,\hat{\mathbf{v}}_2,\hat{\mathbf{v}}_3)$ flat. Moreover, $(\hat{\mathbf{v}}_1,\hat{\mathbf{v}}_4,\hat{\mathbf{v}}_3)$ is also flat implying that $\hat{\mathbf{v}}_2=\hat{\mathbf{v}}_4$ and consequently $\|\hat{\mathbf{v}}_2-\hat{\mathbf{v}}_4\|_{L_m}=0$ contradicting with the assumption that $\hat{\mathbf{v}}_2$ and $\hat{\mathbf{v}}_4$ were not overlapping in the first place. Geometric interpretation of this contradiction is that we are trying to embed a curvy spherical space into the flatty $\mathbb{R}^m$. Since it is impossible to find a distortionless embedding, researchers try to construct an approximate pose that distorts the embedding the least in the sense of some criterion.

A popular attempt to these distortion minimization approaches is an Euclidean embedding called multidimensional scaling (MDS) which comes in classical [4,5], least-squares [6–8], and landmark [9–12] forms. While the least-squares MDS minimizes the embedding distortion in least-squares sense, the classical MDS performs the minimization in the sense of Frobenius norm. Landmark method is merely an interpolation technique for fast embedding of dense set of points. All MDS models aim to represent the pairwise (dis)similarity data stored in the affinity matrix as Euclidean distances in a low-dimensional space in order to make these data accessible to visual inspection and further exploration. This mapping from the affinities a_{ij} to the m-dimensional MDS configuration $\hat{\mathbf{v}}$ is achieved by the transformation function $f: a_{ij}\rightarrow\hat{\mathbf{v}}$, where the particular choice of f specifies the MDS model.

Classical MDS is used extensively in computer graphics and vision applications such as shape correspondence [13,14], shape retrieval [15], and texture mapping [16]. Classical MDS essentially uses the m leading eigenvectors of the associated geodesic affinity matrix of the shape in order to transform the affinities to the m-dimensional configuration $\hat{\mathbf{v}}$ (see Section 5). This eigenanalysis leads to a low-dimensional spectral embedding with no danger of getting stuck in local minima, a problem that least-squares MDS model exhibits while minimizing the transformation error via gradient descent or Scaling by Maximizing a Convex Function (SMACOF) optimization algorithms [17]. Landmark MDS (LMDS), on the other hand, embeds a large number of points by further approximating the classical MDS. Given the embedded landmark points, LMDS interpolates the embedding coordinates for the remaining data points based on their distances from the landmark points. We employ LMDS [10] in order to guide our shape registration process.

The embedding space for all MDS models is $\mathbb{R}^m$. These embeddings are all geodesic-based which arises the problem of sensitivity to local topology changes. Another drawback is the arbitrary reflections due to the sign ambiguities in the eigenvectors.

Another popular Euclidean embedding method is the Laplacian embedding, which replaces the usage of geodesic distances with the graph Laplacian that encodes local geometric and topological properties of the mesh. This appealing property renders Laplacian embedding more robust to topological noise than MDS embedding. As another advantage over MDS that deals with dense affinity matrices, Laplacian embedding performs eigenanalysis on the sparse graph Laplacian defined as follows:

$$L_{ij}=\begin{cases}-1 & \text{if vertex } v_i \text{ and } v_j \text{ adjacent,}\\ \text{degree}(v_i) & \text{if } i=j,\\ 0 & \text{otherwise.}\end{cases} \tag{1}$$

Such a spectral analysis on this combinatorial Laplacian or its variants [18–20] results in numerous applications such as natural vertex ordering for mesh streaming [21], calculation of number of spanning trees and connected components [22], and isometry-invariant embedding, the last one being our particular interest.

By properly adjusting the weights in Eq. (1) using discrete differential geometry, one not only injects more geometry to the combinatorial Laplacian but also decreases the sensitivity to the peculiarities in the triangulation of the input mesh. The resulting discrete Laplacian is linked with the Laplace–Beltrami operator that appears in the wave equation [23,24] and has the following entries:

$$L_{ij}=\begin{cases}-(\cot\alpha_{ij}+\cot\beta_{ij})/2 & \text{if } i \text{ and } j \text{ adjacent,}\\ \sum_k(\cot\alpha_{ik}+\cot\beta_{ik})/2 & \text{if } i=j,\\ 0 & \text{otherwise,}\end{cases} \tag{2}$$

where the edge (v_i,v_j) is shared by two triangles whose angles facing (v_i,v_j) are α and β [25].

Laplacian embedding, as well as the classical MDS embedding, are also known as spectral embeddings since the embedding

coordinates are computed as eigenvectors or eigenfunctions of some matrix.

3.1.2. Non-Euclidean embedding

In non-Euclidean embedding, a non-Euclidean space is used as an embedding space with the aim of finding a more comfortable accommodation to the objects than the flatty Euclidean space does. With this type of embedding the distance between pair of points can be measured via more sophisticated paths than the simple line paths of the Euclidean space, which in turn decreases the embedding distortion.

A popular and relatively easy choice of the non-Euclidean embeddings is spherical embedding, which parameterizes a triangle mesh onto the unit sphere in such a way that the spherical triangles induced by the mesh connectivity are not too distorted and do not overlap [26,27]. A common way to perform this embedding is to cut the closed genus-zero mesh into two pieces, parameterize each over a planar disk with a common boundary using any planar parameterization method [28,29], and then map each disk to a hemisphere to be merged along the common boundary. This embedding, however, is not preferable for shape comparison purposes due to the computational load of the arched distances over sphere surface as well as the difficulty of rigid surface matching in the spherical spaces.

Generalized MDS (GMDS), on the other hand, proposes to embed one shape into the surface of another from the same class [30,31]. Since the curved rooms of a surface embedding space is more suitable than a flat embedding space for housing a similar surface of the same class, this embedding generally performs better than the pure Euclidean embedding as well as the spherical one in the expense of minimization of a non-convex stress function that is difficult and expensive to optimize. The non-convex optimization proposed in the original GMDS is based on many geodesic distance approximations and can get stuck in a local minima [32]. Nevertheless, the invaluable idea of embedding one shape directly into the other removes unnecessary representation errors stemming from embedding into an intermediate space such as a sphere.

We finally mention Möbius embedding which conformally maps shapes with sphere topology into the extended complex plane with a rational linear function [33,34]. This conformal embedding that preserves angles is, however, sensitive to peculiarities of the particular triangulation and restricted to genus-zero surfaces.

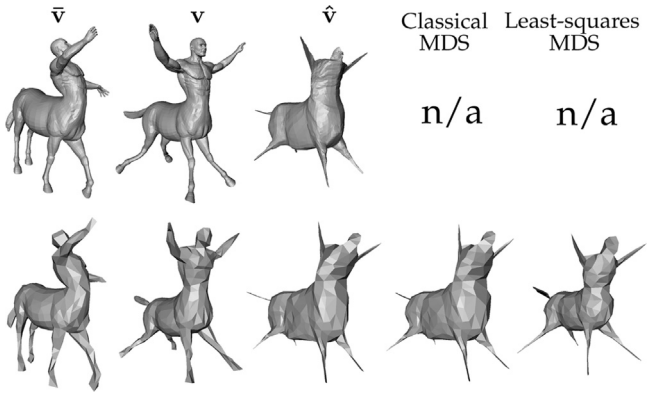


Fig. 1. High- (top) and low-resolution (bottom) models. Left to right: Input shape $\bar{\mathbf{v}}$, resulting deformed shape $\mathbf{v}$, target canonical shape $\hat{\mathbf{v}}$ based on LMDS [10], alternative targets based on classical [13] and least-squares MDS [6]. Note that the computation of the alternatives is intractable for the high-resolution input. Our algorithm, on the other hand, can be efficiently applied to high-resolution models to yield $\mathbf{v}$.

3.2. Other special poses

Since the distortions being minimized in Section 3.1 do not even attempt to preserve the geometric details of the original shape, the resulting poses all lack visual quality, as illustrated in Fig. 1 (see also Figs. 2–3 and 6–7). The poses we discuss in this subsection, on the other hand, are detail-preserving as they aim to achieve completely different objectives, ranging from beautification to fabrication. Symmetrization, for instance, is the task of enhancing approximate symmetries of an object by computing optimal displacement vectors that pulls the shape towards symmetry [35]. For applications of animation control, [36] finds a pose that approximately compensates the mesh sagging effect under gravity by estimating rest-length parameters of a spring-mass system. Refs. [37,38] aim to solve another physically based static equilibrium equation for hair animation. Ref. [39] iterates between carving and deformation to bring the object into a balanced pose that makes it stand after 3D printing. Inverse design methods [40,41] compute a special resting pose for 3D printing, which can

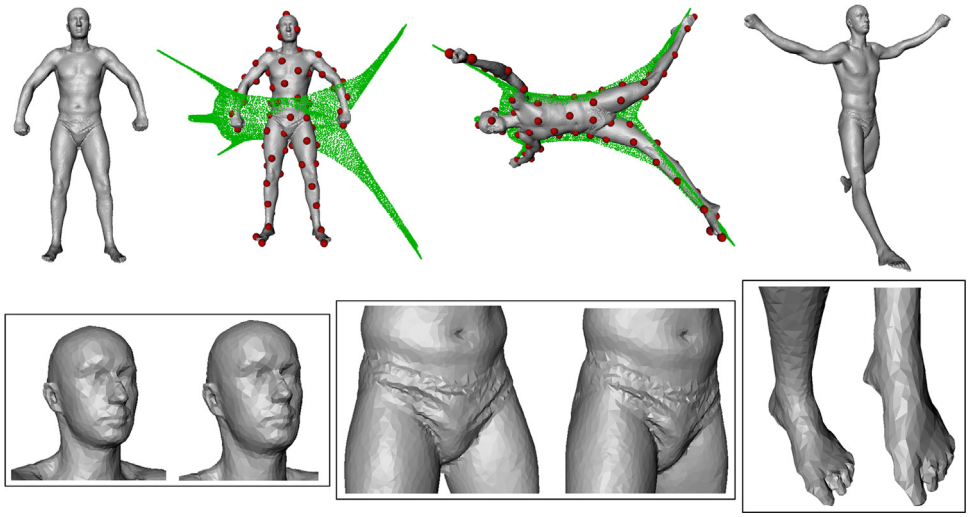


Fig. 2. (Top) Input mesh (leftmost). Classical MDS on landmarks (spheres) is interpolated to the dense mesh [10] to yield the target pose $\hat{\mathbf{v}}$ (green). Mesh is then registered to $\hat{\mathbf{v}}$ with our algorithm. (Bottom) Details before (left of each box) and after (right of each box) registration.

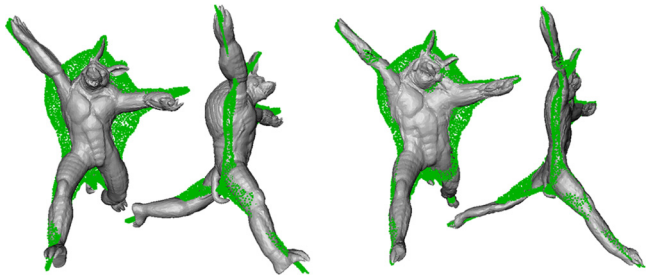


Fig. 3. Final deformed vertices $\mathbf{v}$ with (two views at left) and without (two views at right) sufficient regularization weights. Note that $\mathbf{v}$ at right matches $\hat{\mathbf{v}}$ (green point cloud) more tightly at the expense of losing original details. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

deform into the desired target shape under specified forces when fabricated.

The closest special pose to the pose we are aiming in this work is [1], which goes for a detail-preserving canonical pose after massive amount of computation and various heuristical stages that accumulate errors on the resulting pose. Their resulting pose, however, is shown to be superior for the non-rigid shape retrieval application. We outperform this method in terms of both accuracy and timing, which renders our solution as an even better alternative for non-rigid shape retrieval. This argument is supported further by additional comparisons in Section 8.2. Besides, our method is useful for the shape segmentation application, as demonstrated in Section 8.1.

4. Problem statement and overview

Our goal is to deform a given shape into its canonical pose that is invariant to non-rigid transformations such as bending. The key to this deformation is to preserve the geometric details as well as to enforce pairwise point constraints. To this end, we cast the problem as a shape registration between the input shape (source) and its canonical representation (target). While the matching term for this registration process is induced by the perfectly accurate bijection between the source and target, it definitely needs a good regularization effect to preserve the original details as the target shape is highly distorted from a visual point of view. Additional pairwise constraints can still be added to the linear closed-form framework designed for the detail-preserving shape registration task.

The input shape to our system is represented as a volumetric tetrahedral mesh whose vertices are denoted by $\overline{\mathbf{v}} \in \mathbb{R}^{3n\times 1}$ where n is the number of mesh vertices, and 3 elements are stored for each vertex, namely the x-, y-, and z-coordinates. The advantage of using tetrahedral meshes over the ubiquitous triangular meshes stems from the fact that the highly distorted configuration of the target shape effects the mesh much more severely if handled as a thin shell rather than a solid, as illustrated in Fig. 4 – right. Besides, it is known that tetrahedral meshes yield more realistic deformations than triangular ones do [42]. We consequently convert a potential triangular mesh input to a tetrahedral mesh via [42].

We also denote the vertices in target canonical pose as $\hat{\mathbf{v}} \in \mathbb{R}^{3n\times 1}$, and want to find the deformed vertices $\mathbf{v} \in \mathbb{R}^{3n\times 1}$ as the output of our algorithm (Fig. 1).

The following three sections describe our algorithm to compute the detail-preserving multidimensional scaling of the mesh with possible pairwise point constraints. Although most of the pieces are brought together from the existing works, we derive and formulate clean closed-form expressions to enable reproducibility. Another contribution is the smooth blending of these pieces to produce a solution that achieves the new state-of-the-art in the creation of detail-preserving canonical poses, an important problem in computer graphics and vision.

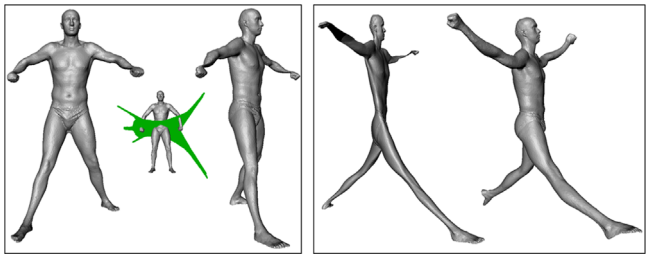


Fig. 4. (Left box) $E_{\text{Dirichlet}}$ does not support the large deformation to handle the registration (middle), e.g., arms do not rise sufficiently. (Right box) E_{ARAP} rises arms sufficiently. Unlike the tetrahedral mesh (right), the triangular surface mesh (left) squeezes, hence promoting the former.

5. Generation of target pose: $\hat{\mathbf{v}}$

We first need a fixed target pose towards which our tetrahedral mesh will be pulled. This fixed pose captures the intrinsic non-Euclidean geometry of the shape with an extrinsic representation, an appealing property for most geometry processing applications as this representation allows efficient and robust shape comparison using only a few degrees of freedom [1,13,14]. To this effect, we compute the target pose $\hat{\mathbf{v}}$ via multidimensional scaling (MDS).

The choice of the MDS model is an important issue. Since we want a scalable algorithm that can handle high-resolution models efficiently, we prefer the landmark-based interpolation method LMDS [10], which requires a seed embedding to be interpolated into a dense one. The seed embedding transforms only a subset of the vertex set $\overline{\mathbf{v}}$ to the Euclidean embedding space, which we call the landmark vertices (spheres in Fig. 2) and compute using the Farthest Point Sampling (FPS) algorithm [43]. Amongst two choices for this initial embedding, we prefer the classical approach [13] over the least-squares method [6] as the former is based on eigendecomposition, for which many efficient numerical algorithms are available. Besides, unlike the latter, it does not suffer from local convergence.

We start by forming an affinity matrix $A_{ij} = \exp(-g^2(i,j)/2\sigma^2)$ for the landmark vertex set of cardinality l with $g(\cdot,\cdot)$ being the geodesic distance between two points on a given surface. We set the kernel width σ to be half of the maximum geodesic distance over the surface, and use l=80. Note that computation of the geodesic affinity matrix does not bring any additional load to our overall algorithm as the corresponding geodesics are already prepared during the FPS process. $\mathbf{A} \in \mathbb{R}^{l\times l}$ is then eigendecomposed as $\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^{-1}$, where $\Lambda \in \mathbb{R}^{l\times l}$ is a diagonal matrix with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_l$ along the diagonal and $\mathbf{Q} = [\mathbf{q}_1 | \cdots | \mathbf{q}_l] \in \mathbb{R}^{l\times l}$ is the matrix of the corresponding eigenvectors, i.e., $\mathbf{q}_i \in \mathbb{R}^{l\times 1}$ is the ith eigenvector. We scale each of these eigenvectors with the inverse of the square root of the corresponding eigenvalue as suggested in [10], leading to $\hat{\mathbf{Q}} = \mathbf{Q}\Lambda^{-1/2}$. We truncate $\hat{\mathbf{Q}} \in \mathbb{R}^{l\times l}$ into $\hat{\mathbf{Q}}_m \in \mathbb{R}^{l\times m}$ by considering only the first m significant eigenvectors and obtain an m-dimensional spectral embedding $\hat{\mathbf{Q}}_m = [\hat{\mathbf{q}}_1 | \cdots | \hat{\mathbf{q}}_m]$, where the ith row of $\hat{\mathbf{Q}}_m$ gives the m-dimensional embedded coordinates of the ith landmark.

Given the seed embedded coordinates in $\hat{\mathbf{Q}}_m$, we employ LMDS [10] to embed the remaining $n-l$ vertices based on their distances from the l landmark vertices. Specifically, the m-dimensional embedded coordinates of the ith vertex is

$$\hat{\mathbf{v}}_i = \hat{\mathbf{Q}}_m^{\top}(\mathbf{g}_i - \mathbf{g}_\mu), \tag{3}$$

where $\mathbf{g}_i \in \mathbb{R}^{l\times 1}$ is the vector of kerneled geodesic distances from

the ith vertex to all the l landmarks and $\mathbf{g}_\mu = (\mathbf{g}'_1 + \cdots + \mathbf{g}'_l)/l$ is the mean where $\mathbf{g}'_j \in \mathbb{R}^{l\times 1}$ is the vector of kerneled geodesic distances from the jth landmark to all the landmarks. We set $m=3$ and concatenate $\hat{\mathbf{v}}_i \in \mathbb{R}^{3\times 1}$ for $i=1,..,n$ into $\hat{\mathbf{v}} \in \mathbb{R}^{3n\times 1}$ (green point cloud in Fig. 2).

6. Shape deformation: v

Our deformation algorithm aims to register the input mesh to the target $\hat{\mathbf{v}}$. The deformed vertices $\mathbf{v}$ are expected to match $\hat{\mathbf{v}}$ as much as possible while preserving the original geometric details present at $\overline{\mathbf{v}}$. Since $\hat{\mathbf{v}}$ is highly distorted in terms of geometric details, we need a volumetric deformation energy that competes well against this situation. To this end, we employ as-rigid-as possible (ARAP) energy which also supports large deformations [44]. In order to avoid overly large steps that could result in inversions, i.e., flipped tetrahedra, on the deforming mesh, we also introduce a Tikhonov regularization term that penalizes too large motion from the previous step. We therefore seek $\mathbf{v}$ that minimizes the following overall energy:

$$E_{\text{def}}(\mathbf{v}) = E_{\text{match}}(\mathbf{v}) + \alpha E_{\text{ARAP}}(\mathbf{v}) + \beta E_{\text{Tikhonov}}(\mathbf{v}) \tag{4}$$

We elaborate our deformation process in the sequel.

6.1. Minimization of $E_{\text{match}}(\mathbf{v})$

The first energy term E_{match} penalizes the spatial difference between $\mathbf{v}$ and $\hat{\mathbf{v}}$. Thanks to the one-to-one mapping from the input shape to its LMDS representation $\hat{\mathbf{v}}$, this difference is measured trivially by subtracting the coordinate entries at the same indices:

$$E_{\text{match}}(\mathbf{v}) = \|\mathbf{v}-\hat{\mathbf{v}}\|^2 = (\mathbf{v}-\hat{\mathbf{v}})^\top(\mathbf{v}-\hat{\mathbf{v}}) = \|\mathbf{v}\|^2 - 2\mathbf{v}^\top\hat{\mathbf{v}} + \|\hat{\mathbf{v}}\|^2 \tag{5}$$

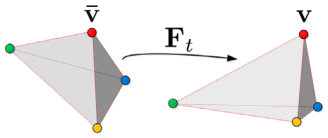
Since this mapping is also perfectly accurate, Eq. (5) does not suffer from the local minima problem of the classical ICP approaches even if the initial positioning of $\mathbf{v}$ and $\hat{\mathbf{v}}$ is wide apart [45]. Minimization of Eq. (5) is, however, not very interesting as it would set $\mathbf{v}=\hat{\mathbf{v}}$ which would return the exact target LMDS pose, defeating our detail-preservation purpose. To remedy this problem, we next introduce our regularization terms that keep $\mathbf{v}$ in good shape.

6.2. Minimization of $E_{\text{ARAP}}(\mathbf{v})$

The second energy term E_{ARAP} has a regularization effect by protecting the supposedly good initial shape of each tetrahedron via

$$E_{\text{ARAP}}(\mathbf{v}) = \sum_t \overline{\lambda}_t \|\mathbf{F}_t - \mathbf{R}_t\|_F^2, \tag{6}$$

which penalizes the deviation of the deformation gradient $\mathbf{F}_t$ for tetrahedron t from the closest rigid rotation $\mathbf{R}_t$ [44].



Deformation gradient $\mathbf{F}_t \in \mathbb{R}^{3\times 3}$ is a transformation shown at right that maps the input-pose edges to their deformed pose as follows. Let $\overline{\mathbf{v}}_{1,2,3,4}$ and $\mathbf{v}_{1,2,3,4}$ be the input and deformed coordinates of tetrahedron t, respectively. Representing the edges via $\mathbf{E}_t = [\mathbf{v}_1 - \mathbf{v}_4 | \mathbf{v}_2 - \mathbf{v}_4 | \mathbf{v}_3 - \mathbf{v}_4] \in \mathbb{R}^{3\times 3}$ and $\overline{\mathbf{E}}_t = [\overline{\mathbf{v}}_1 - \overline{\mathbf{v}}_4 | \overline{\mathbf{v}}_2 - \overline{\mathbf{v}}_4 | \overline{\mathbf{v}}_3 - \overline{\mathbf{v}}_4] \in \mathbb{R}^{3\times 3}$, we obtain

$$\mathbf{E}_t = \mathbf{F}_t \overline{\mathbf{E}}_t, \tag{7}$$

which implies $\mathbf{F}_t = \mathbf{E}_t \overline{\mathbf{E}}_t^{-1}$, with $\overline{\mathbf{E}}_t^{-1}$ being pre-computed for efficiency.

The closest rigid rotation $\mathbf{R}_t$, on the other hand, optimally aligns tetrahedron t with the corresponding 4 points on $\hat{\mathbf{v}}$ in the least-squares sense, hence deforming $\mathbf{v}$ towards the target. We employ the alternating iterative algorithm in [44] with a slight modification to minimize Eq. (6): (1) Fixing $\mathbf{v}$, we first solve for rotations using singular value decomposition [44]. (2) Fixing rotations, we then solve for $\mathbf{v}$ that minimizes Eq. (6) using the matrix algebra discussed in the sequel. We alternate between these two steps until mesh stops moving. Our slight modification is that in step (1) we use $\hat{\mathbf{v}}$ only in the first iteration. For the remaining iterations, we use the faired [46] version of $\mathbf{v}$ instead as it approximates $\hat{\mathbf{v}}$ with a better shape, hence improving the calculations of rotations.

We need the quadratic form of Eq. (6) to minimize it efficiently. We begin by rewriting Eq. (6) as

$$\begin{aligned} E_{\text{ARAP}}(\mathbf{v}) &= \sum_t \overline{\lambda}_t (\mathbf{G}_t\mathbf{v} - \mathbf{r}_t)^\top (\mathbf{G}_t\mathbf{v} - \mathbf{r}_t) = \sum_t \overline{\lambda}_t ((\mathbf{G}_t\mathbf{v})^\top \mathbf{G}_t\mathbf{v} \\ &\quad - 2\mathbf{r}_t^\top (\mathbf{G}_t\mathbf{v}) + r_t^\top r_t), \end{aligned} \tag{8}$$

where $\overline{\lambda}_t$ is the volume of the tetrahedron t at the input-pose $\overline{\mathbf{v}}$ and the deformation gradient operator $\mathbf{G}_t \in \mathbb{R}^{9\times 3n}$ extracts the vectorized version of the desired gradient $\mathbf{f}_t = \text{vec}(\mathbf{F}_t) \in \mathbb{R}^{9\times 1}$ when multiplied by $\mathbf{v}$, i.e., $\mathbf{f}_t = \mathbf{G}_t\mathbf{v}$ [47]. Note that rotation $\mathbf{R}_t$ is also exposed to vectorization, i.e., $\mathbf{r}_t = \text{vec}(\mathbf{R}_t) \in \mathbb{R}^{9\times 1}$. Using the $(\mathbf{AB})^\top = \mathbf{B}^\top\mathbf{A}^\top$ identity,

$$\begin{aligned} E_{\text{ARAP}}(\mathbf{v}) &= \sum_t \overline{\lambda}_t ((\mathbf{v}^\top \mathbf{G}_t^\top)(\mathbf{G}_t\mathbf{v}) - 2\mathbf{r}_t^\top \mathbf{G}_t\mathbf{v} + r_t^\top r_t) \\ &= \mathbf{v}^\top \underbrace{\left(\sum_t \overline{\lambda}_t \mathbf{G}_t^\top \mathbf{G}_t\right)}_{\mathbf{L}} \mathbf{v} - 2\underbrace{\left(\sum_t \overline{\lambda}_t \mathbf{r}_t^\top \mathbf{G}_t\right)}_{\mathbf{r}^\top\mathbf{K}} \mathbf{v} + \sum_t \overline{\lambda}_t \mathbf{r}_t^\top \mathbf{r}_t, \end{aligned} \tag{9}$$

where $\mathbf{L}$ is the mesh Laplacian and $\mathbf{r}^\top = [\mathbf{r}_1^\top | \cdots | \mathbf{r}_h^\top] \in \mathbb{R}^{1\times 9h}$ concatenates the rotations of each of the h tetrahedra in the mesh column by column. Similarly, $\mathbf{K} \in \mathbb{R}^{9h\times 3n}$ concatenates the scaled deformation gradient operators $\overline{\lambda}_t\mathbf{G}_t$ row by row. The final quadratic form is given by

$$E_{\text{ARAP}}(\mathbf{v}) = \mathbf{v}^\top\mathbf{L}\mathbf{v} - 2\mathbf{r}^\top\mathbf{K}\mathbf{v} + \sum_t \overline{\lambda}_t \mathbf{r}_t^\top \mathbf{r}_t \tag{10}$$

6.3. Minimization of $E_{\text{Tikhonov}}(\mathbf{v})$

The third energy term E_{Tikhonov} is another regularizer that helps keep the deforming mesh in good shape. Since E_{ARAP} energy does not sufficiently penalize element inversions, especially in the case of large deformations that we intend to handle, we control the deformation further via E_{Tikhonov} which penalizes the deviation from the previous step:

$$E_{\text{Tikhonov}}(\mathbf{v}) = \|\mathbf{v}-\check{\mathbf{v}}\|^2 = \|\mathbf{v}\|^2 - 2\mathbf{v}^\top\check{\mathbf{v}} + \|\check{\mathbf{v}}\|^2, \tag{11}$$

where $\check{\mathbf{v}}$ is the pose of the previous iteration.

6.4. Minimization of $E_{\text{def}}(\mathbf{v})$

Having put all the three constituent energy terms into their quadratic forms after some matrix algebra (Eqs. (5), (10), (11)), it is now easy to minimize the overall energy term E_{def} by differentiating with respect to $\mathbf{v}$:

$$\begin{aligned} \frac{\partial E_{\text{def}}}{\partial \mathbf{v}} &= \frac{\partial E_{\text{match}}}{\partial \mathbf{v}} + \alpha\frac{\partial E_{\text{ARAP}}}{\partial \mathbf{v}} + \beta\frac{\partial E_{\text{Tikhonov}}}{\partial \mathbf{v}} = 2(\mathbf{v}-\hat{\mathbf{v}} + \alpha(\mathbf{L}\mathbf{v} - (\mathbf{r}^\top\mathbf{K})^\top) \\ &\quad + \beta(\mathbf{v}-\check{\mathbf{v}})) \end{aligned} \tag{12}$$

Setting this derivative to zero, we obtain a linear system:

$$(\alpha\mathbf{L}+(\beta+1)\mathbf{I})\mathbf{v}=\hat{\mathbf{v}}+\alpha(\mathbf{r}^\top\mathbf{K})^\top+\beta\check{\mathbf{v}}, \tag{13}$$

which we solve quickly using a sparse direct solver.

6.5. Notes on $E_{\text{def}}(\mathbf{v})$

Our deformation model is essentially a competition between the matching term that pulls $\mathbf{v}$ towards $\hat{\mathbf{v}}$ and regularization terms that favor a mesh whose tetrahedra are shaped similarly to the original ones induced by $\overline{\mathbf{v}}$. Consequently, the selection of α and β weights are important to achieve a good balance, as illustrated in Fig. 3. Although there is not any systematic protocol other than trial-and-error method for this selection, the optimal weights we choose for the production of our results are always the same, i.e., $\alpha=100$ and $\beta=10$.

We also emphasize the importance of our main regularizer E_{ARAP} by replacing it with another deformation measure from the same family, namely the Dirichlet energy $E_{\text{Dirichlet}}$, which penalizes the deviation of the deformation gradient $\mathbf{F}_t$ for tetrahedron t from the identity matrix $\mathbf{I}_3\in\mathbb{R}^{3\times3}$ [48]:

$$E_{\text{Dirichlet}}(\mathbf{v})=\sum_t\overline{\lambda}_t\|\mathbf{F}_t-\mathbf{I}_3\|_F^2 \tag{14}$$

Unlike E_{ARAP} that is invariant to rotations, $E_{\text{Dirichlet}}$ penalizes rotations as they are different from $\mathbf{I}_3$. Consequently, $E_{\text{Dirichlet}}$ cannot handle as large deformations as E_{ARAP} can, as illustrated in Fig. 4 – left.

We finally note that E_{ARAP} energy performs better on volumetric solid shapes represented by tetrahedra than it does on surfaces in 3D, as shown in Fig. 4 – right, especially for our specific registration problem which uses a squeezed target shape $\hat{\mathbf{v}}$ suffering from volume loss. A tetrahedron having more neighbors through its edges than a triangle in 3D is more constrained and has less tendency towards the squeezed target configuration, which in turn promotes a tetrahedral mesh for our work.

7. Additional pairwise constraints

As an additional feature to our deformation model, we can incorporate pairwise point constraints on the deforming $\mathbf{v}$ in quadratic form, hence still solving a fast linear system. To this end, we adapt the quadratic spring potential representation of [49]. The user-specified distances between the user-specified point pairs are induced on $\mathbf{v}$ via springs connecting those points:

$$E_{\text{def2}}(\mathbf{v})=E_{\text{spring}}(\mathbf{v})+E_{\text{def}}(\mathbf{v})=\tfrac{1}{2}\mathbf{v}^\top\mathbf{L}_s\mathbf{v}-\mathbf{v}^\top\mathbf{J}\mathbf{d}+E_{\text{def}}(\mathbf{v}), \tag{15}$$

which leads to the following linear system to solve after setting its derivative to zero:

$$(\mathbf{L}_s+2\alpha\mathbf{L}+(2\beta+2)\mathbf{I})\mathbf{v}=\mathbf{J}\mathbf{d}+2\hat{\mathbf{v}}+2\alpha(\mathbf{r}^\top\mathbf{K})^\top+2\beta\check{\mathbf{v}}, \tag{16}$$

where the matrices $\mathbf{L}_s$, $\mathbf{J}$, and the vector $\mathbf{d}$ encode the spring information (please refer to [49] for the derivation and content of these structures). The resulting pose after the constrained deformation is given in Fig. 5.

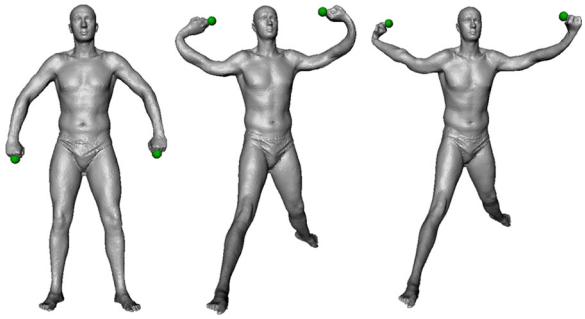


Fig. 5. Input mesh (left) with two user-specified points to be constrained (spheres). High (middle) and low (right) spring stiffness values yielding different poses $\mathbf{v}$. Unconstrained result can be seen in Fig. 2.

8. Results

We have tested our detail-preserving multidimensional scaling (MDS) computation method on three well-known 3D shape benchmark datasets: SCAPE [50], TOSCA [8], and Watertight [51]. The former is a reconstructed pose sequence of a human actor, which contains 71 non-uniformly sampled models exhibiting triangulation peculiarities. TOSCA, on the other hand, consists of 35 fourleg animals, 39 humans, and 6 centaurs, hence a total of 80 objects, each having approximately 50 K vertices. The part that we have used from the Watertight dataset consists of glasses, chair, spring, and armadillo classes of cardinality 20 each.

In addition to the visual evidence of our performance provided through Figs. 6–8, we also demonstrate our advantage over the only other existing work addressing the same detail-preserving MDS problem (Fig. 9). We also show the potential of our algorithm with a shape segmentation application (Section 8.1).

In Fig. 6, we visualize our deformation process on three different examples from the Watertight and TOSCA datasets through intermediate iterations (please also see the accompanying video for more results on this process).

Detail-preserving MDS poses of some SCAPE and Watertight models are given in Figs. 7 and 8, respectively. We observe that all models converge to their target poses with preserved details. SCAPE models possessing peculiarities in their triangulations

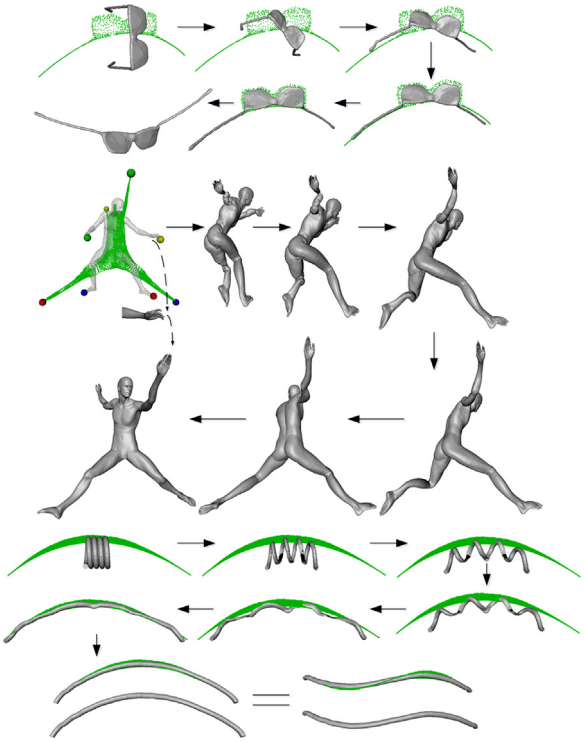


Fig. 6. Three example shapes (gray) are being deformed towards their LMDS poses (green) in the direction of arrows. Last steps show the resulting pose in a different view. Note that details as subtle as those in the fingers of the human figure (dashed arrows) are preserved. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

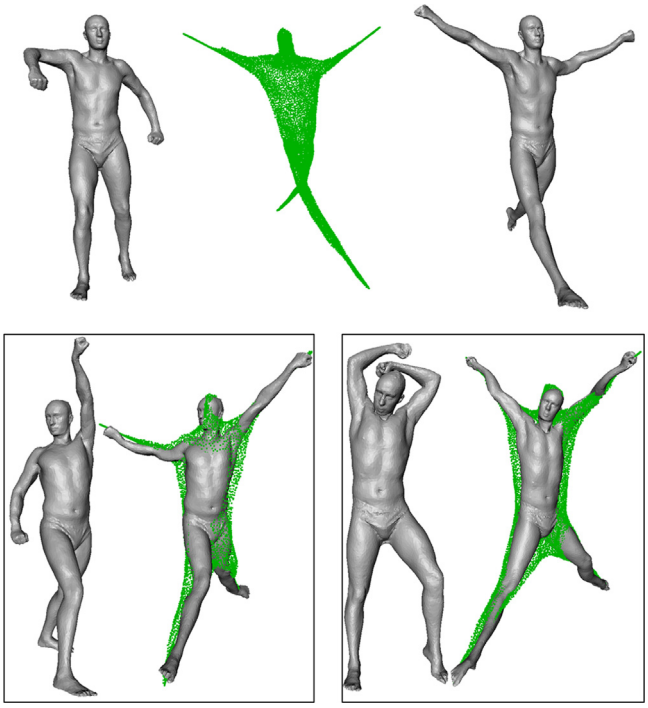


Fig. 7. (Top) Input (left) and its LMDS representation (middle). Resulting model by our algorithm (right). (Bottom) Input and resulting models as left and right sides of each box.

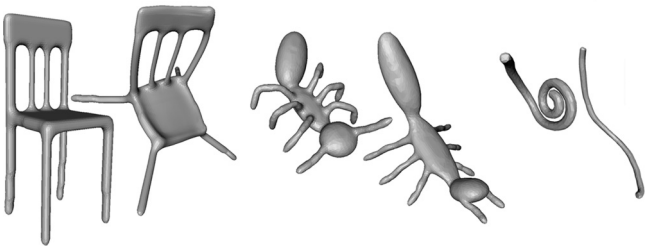


Fig. 8. Input and resulting poses as left and right sides of three pairs.

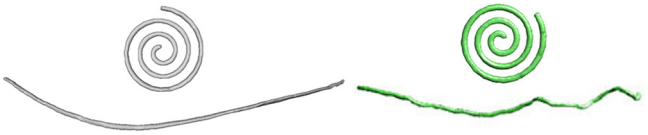


Fig. 9. Input models (top) and outputs (bottom) produced by our algorithm (left) and [1] (right).

deform as nice and smooth as the TOSCA human with uniform triangulation (Fig. 6 – middle), which shows the insensitivity of our framework to the tessellation quality.

The execution time of our algorithm on a 2.53 GHz PC with 8 GB RAM for a SCAPE model of 12.5 K vertices and 44 K tetrahedra is 20 s, whereas a TOSCA model of 53 K vertices and 185 K tetrahedra requires 53 s. Chair (10 K vertices), ant (5 K vertices), and spring (752 vertices) models from the Watertight dataset take 15, 7, and 1 s, respectively. While $\sim$10% of the execution time is used for the sampling and landmark MDS embedding operations, the remaining time belongs to the deformation process. Considering the 243 s execution time of [1] for a 5 K-vertex mesh on a faster machine (3.19 GHz, 12 GB RAM), our method is much more fast, i.e., compare with our corresponding 7 s on our slower computer. The reason is that while we are solving one closed-form energy function E_{def} as a sparse linear system at each iteration, [1] solves several disjoint energy minimization problems for their assembling and smoothing operations, preceded by other heuristical steps such as voxelization and segmentation. Unlike our closed-form equation that can be easily implemented (Eq. (13)), the descriptions of their steps are not clear and sufficient for reproduction. We were also unable to find a public repository (code, executable, resulting mesh files) for [1]. We consequently directly copy their figure in our Fig. 9 and run our algorithm with the same input. Our resulting pose obtained in much less time is smoother and more accurate.

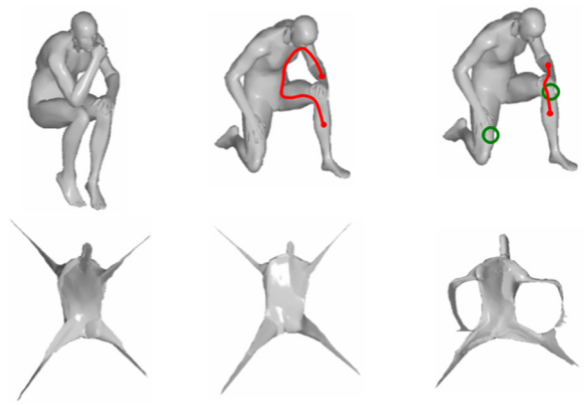


Fig. 10. MDS embeddings with (right) and without (left and middle) topological noise. Image taken from [8].

Table 1
Hausdorff distances between unfolded models when the target poses are obtained with and without noise. Models are scaled such that maximum geodesic distance is one. The amount of displacement is set to a random number in $[0, \rho\epsilon]$, where ϵ is the average edge length in the original mesh.

Distance	$\rho=1$	$\rho=5$	$\rho=10$	$\rho=20$	$\rho=50$
Hausdorff	0.09	0.12	0.17	0.70	1.96

We finally evaluate the performance of our algorithm under noise. The quality of the MDS-based target pose (Section 5) depends on the geodesic distances, which are affected at various levels in the presence of topological and/or geometrical noises.

With topological noise, e.g., hand is connected to the knee via a single noisy edge (Fig. 10 – right), geodesic distances change drastically, which in turn messes up the MDS-based embedding.

With geometric noise, e.g., random perturbations of vertices, geodesic distances change in proportion to the amount of vertex displacements. To quantify this affect, we add a combination of noise (random amount of displacement in normal direction) and shotnoise (random direction of displacement) to the male TOSCA models [52], and then use the resulting embeddings as the target pose of the original models, which are unfolded towards these targets. We also unfold the original models towards the target embeddings obtained without any noise. We then quantify the similarity of these unfolded meshes by computing their Hausdorff distances. Small distance values in Table 1 show that our unfolding operation is insensitive to the geometric noise when the power of displacement is sufficiently low, e.g., $\rho \leq 10$.

8.1. An application: shape segmentation

As mentioned in Section 1, Euclidean geometry simplifies solutions to many graphics applications. After equipping a shape with the Euclidean geometry by creating the extrinsic representation of its intrinsic geometry via new coordinates $\mathbf{v}$, we take the shape segmentation application into consideration to demonstrate this claim.

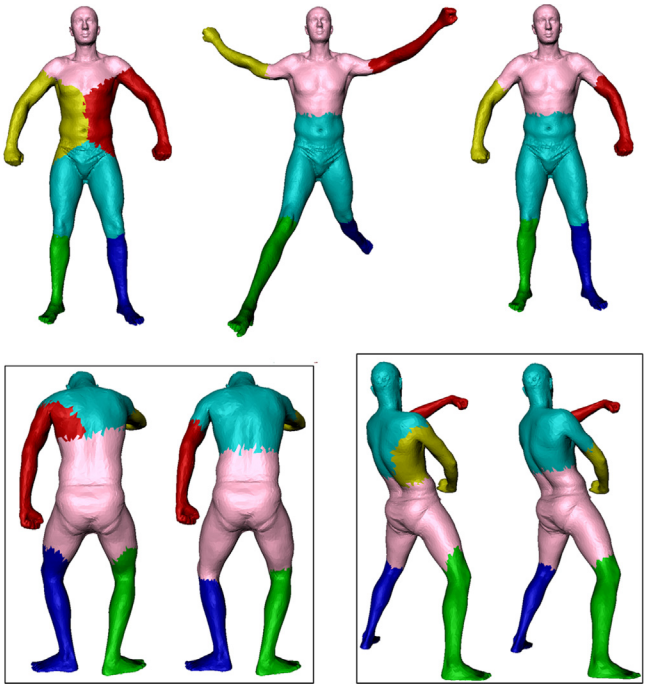


Fig. 11. (Top) Input mesh segmented via [53] based on the non-Euclidean geometry (left). Our segmentation computed on $\mathbf{v}$ based on the Euclidean geometry (middle). $\mathbf{v}$ is rolled-back to the input coordinates $\bar{\mathbf{v}}$ with the new segmentation labels (right). (Bottom) Segmentation results of [53] and our work as left and right sides of each box.

Our proposed segmentation algorithm leveraging an extrinsic representation of intrinsic geometry is much more simple than existing algorithms [53,54], yet perform sufficiently accurate for isometric shapes that are identical up to rotation, translation, and bending deformations, e.g., a human figure in standing and sitting poses. Our algorithm simply runs the k-means clustering algorithm on the resulting coordinates $\mathbf{v}$ to obtain a pose-invariant segmentation. Unlike [53] which considers the intrinsic non-Euclidean geometry, i.e., geodesic distances along the surface, during the k-means process, we consider the Euclidean geometry, i.e., the L_2 distances over $\mathbf{v}$, which in turn yields a better segmentation illustrated in Fig. 11.

We also perform quantitative comparisons with [53]. Note that [53] is one of the seven algorithms evaluated by Princeton segmentation benchmark [55], which uses the Watertight models as its dataset. We employ two metrics from [55], namely the boundary-based Cut Discrepancy (CD) and the region-based Rand Index (RI), where the former measures how close segment boundaries of the computer-generated and human-generated results are to one another, and the latter measures the consistency of segment interiors. For human-generated results we use the ones provided by [55]. Since CD measures dissimilarities, we report 1-RI (as [55] does) to quantify dissimilarities rather than similarities, thus lower values are better on both metrics. The average CD and 1-RI over all models in the benchmark for our method are 0.37 and 0.21, respectively. The corresponding pair of numbers for [53] is slightly worse, being 0.41 and 0.25. We also give these numbers over the Human class instead of over all the models. In this case we obtain 0.31 and 0.18, whereas [53] achieves 0.33 and 0.22. These results show that utilizing our Euclidean distances instead of geodesics [53] inside the same clustering framework yields better performance. We therefore think that this is a good showcase of the Euclidean distances produced by our detail-preserving MDS method. We should however note that our preliminary segmentation algorithm requires further optimization in order to compete with the recent sophisticated algorithms that are designed specifically for the shape segmentation problem [56–58].

8.2. An application: non-rigid shape retrieval

As another computer graphics application, we note that [1] has reported a superior performance of detail-preserving MDS poses over the conventional MDS poses for the non-rigid shape retrieval problem. This is a reasonable result as semantically similar shapes with varying details could not be distinguished well with conventional canonical poses that lack details. Our detail-preserving canonical poses, being much more accurate than those of [1], imply that our algorithm improves the performance of a non-rigid shape retrieval application as well. We verify this hypothesis by the experiments on the TOSCA benchmark in the sequel. We also note that, in a parallel and independent work, [59] also computes detail-preserving MDS poses and demonstrates their superiority for the non-rigid shape retrieval application.

There is not a single state-of-the-art algorithm on non-rigid shape retrieval. As reported recently in SHREC'15 track [60], some of the older MDS-based methods [6] are still as effective as the more recent algorithms, but have been surpassed in terms of efficiency. We therefore pick [10,61] for comparisons, where the first one is an MDS-based method that interpolates the sparse results of [6] to dense meshes efficiently, and the second one is a recent belief function based approach that already employs the TOSCA dataset, which makes it easier and healthier to import their results into our paper. Note that [10] is the target pose (Section 5) of our method (Figs. 1 and 7) towards which our input tetrahedral mesh is pulled with preserved details.

Quantitative analysis of our experimental results is depicted in Fig. 12, which shows the Precision–Recall curves of our algorithm, LMDS [10,61]. For the plots in Fig. 12, the vertical axis is the Precision, which is the ratio of the relevant matches to the number of retrieved models, whereas the horizontal axis is the Recall, which is the ratio of relevant matches to the size of the query class. Ideally, this curve should be a horizontal line at unit precision. Based on these plots, we find that our approach provides the best retrieval precision in this experiment.

9. Limitations and discussion

The main limitation of our algorithm is its dependence on geodesic distances which brings sensitivity to the topological noise as a small topological change on the mesh may alter most of the geodesics drastically (Fig. 10 – right). One may alleviate this problem by introducing diffusion-based distances [62,2] to the system. Another limitation concerns our additional feature, namely the pairwise point constraining, in that we currently can only control the distance between the constraint pair of points, not the

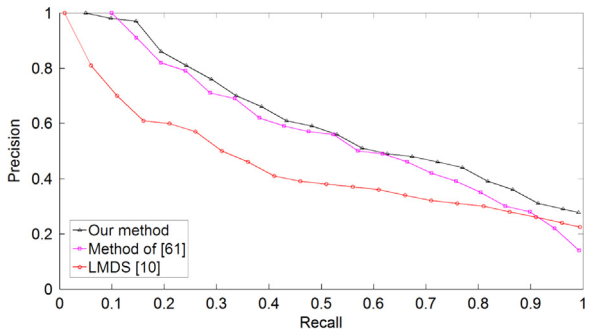


Fig. 12. Precision–Recall plots of the retrieval methods on TOSCA dataset.

direction. We therefore cannot quite obtain a special pose, such as the Vitruvian man pose, as the distances may be satisfied in arbitrary configurations. This arbitrary behavior may even cause self-intersections. The potential of our algorithm is demonstrated through two applications, shape segmentation and retrieval.

Finally we note that there are methods that handle volumetric deformation without requiring any volumetric tessellation [63,64]. Ref. [63] rather constructs a volumetric graph whose vertices are derived from a lattice embedded inside the mesh. This deformation, however, only roughly preserves the volume. Ref. [64], on the other hand, links the volume of an object to its surface curvature, which in turn cannot preserve simultaneously the surface details and the volume. We use the recent advances in volumetric tessellation [42] to tetrahedralize any given mesh, and consequently preserve the volume better.

10. Conclusion and future work

We presented an algorithm that brings a given shape into the detail-preserving multidimensional scaling (MDS) pose by means of an efficient and robust volumetric shape registration process. The input tetrahedral mesh is pulled towards its MDS representation using the perfectly accurate matching in between. The original geometric details on the input is preserved by adapting the closed-forms of as-rigid-as-possible and Tikhonov deformation energies. Thanks to the quadratic harmony of the energy terms, all the deformation process boils down to solving a sparse linear system. Additional pairwise point constraints can also be added without sacrificing the linear structure of the solution.

As the future work, apart from analyzing the trade-off between the accuracy of the geodesic metric currently in use and the topological noise robustness of the diffusion-based metric to be tested, one may also investigate a way to fine-tune the deformation weights in a more principled way than the trial-and-error approach. We should note that, though, once tweaked, we used the same weights for all the results in this paper. Our preliminary segmentation solution may also benefit from more advanced clustering techniques as well as boundary smoothing via morphology. Improving the pairwise point constraints to handle special poses, such as the Vitruvian man pose or A-pose, may also be interesting.

Appendix A. Supplementary material

Supplementary data associated with this paper can be found in the online version at http://dx.doi.org/10.1016/j.cag.2015.10.003.

References

[1] Lian Z, Godil A, Xiao J. Feature-preserved 3d canonical form (IJCV). Int J Comput Vis 2013;102(1–3):221–38.
[2] Lipman Y, Rustamov R, Funkhouser T. Biharmonic distance. ACM Trans Graph 2010;29(3).
[3] Linial N. Finite metric spaces combinatorics, geometry, and algorithms. In: Proceedings ICM, vol. 3; 2002. p. 573–86.
[4] Cox T, Cox M. Multidimensional scaling. 2nd ed.. London: Chapman & Hall/CRC; 2000.
[5] Rustamov R, Lipman Y, Funkhouser T. Interior distance using Barycentric coordinates. Comput Graph Forum 2009;28(5).
[6] Elad A, Kimmel R. On bending invariant signatures for surfaces. IEEE Trans PAMI 2003;25:1285–95.
[7] Borg I, Groenen P. Modern multidimensional scaling–theory and applications. 2nd ed.. Berlin: Springer; 2005.
[8] Bronstein AM, Bronstein MM, Kimmel R. Numerical geometry of non-rigid shapes. New York: Springer; 2008.
[9] Faloutsos C, Lin K. A fast algorithm for indexing, datamining and visualisation of traditional and multimedia datasets. ACM SIGMOD 1995:163–74.
[10] de Silva V, Tenenbaum J. Global versus local methods for nonlinear dimensionality reduction. In: Proceedings NIPS; 2002. p. 705–12.
[11] Aflalo Y, Kimmel R. Spectral multidimensional scaling. Proc Natl Acad Sci 2013;110(45):18052–7.
[12] Panozzo D, Baran I, Diamanti O, Sorkine-Hornung O. Weighted averages on surfaces. In: Proceedings SIGGRAPH, 32 (4) (2013) 60:1–60:12.
[13] Jain V, Zhang H. Robust 3d shape correspondence in the spectral domain. In: IEEE international conference on shape modeling and applications (SMI); 2006. p. 118–29.
[14] Sahillioğlu Y, Yemez Y. Minimum-distortion isometric shape correspondence using EM algorithm. IEEE Trans PAMI 2012;34(11):2203–15.
[15] Jain V, Zhang H. A spectral approach to shape-based retrieval of articulated 3d models. Comput Aided Des 2007;39(5):398–407.
[16] Zigelman G, Kimmel R, Kiryati N. Texture mapping using surface flattening via multidimensional scaling. IEEE Trans Vis Comput Graphics 2002;8:198–207.
[17] Groenen P, van de Velden M. Multidimensional scaling. Econom Inst Rep EI 2004:04–15.
[18] Zhang H. Discrete combinatorial Laplacian operators for digital geometry processing. Proc SIAM Conf Geom Des Comp 2004:575–92.
[19] Sorkine O. Differential representations for mesh processing. Comput Graph Forum 2006;25(4):789–807.
[20] Mateus D, Horaud R, Knossow D, Cuzzolin F, Boyer E. Articulated shape matching using Laplacian eigenfunctions and unsupervised point registration. In: Proceedings of computer vision and pattern recognition (CVPR); 2008.
[21] Isenburg M, Lindstrom P. Streaming meshes. IEEE Vis 2005:30–7.
[22] Feidman J. Computing betti numbers via combinatorial Laplacians. Proc 28th Symp Theory Comput 1996:386–91.
[23] Levy B. Laplace–Beltrami eigenfunctions: towards an algorithm that understands geometry. In: IEEE international conference on shape modeling and applications (SMI); 2006.
[24] Wardetzky M, Saurabh M, Kalberer F, Grinspun E. Discrete Laplace operators: no free lunch. In: Sympsium on geometry processing; 2007. p. 33–37.
[25] Rustamov R. Laplace–Beltrami eigenfunctions for deformation invariant shape representation. Comput Graph Forum (Proc SGP) 2007:225–33.
[26] Gotsman C, Gu X, Sheffer A. Fundamentals of spherical parameterization for 3D meshes. ACM Trans Graph 2003;22(3):358–63.
[27] Haker S, Angenent S, Tannenbaum A, Kikinis R, Sapiro G. Conformal surface parameterization for texture mapping. IEEE Trans Vis Comp Graph 2000;6(2):1–8.
[28] Desbrun M, Meyer M, Alliez P. Intrinsic parameterizations of surface meshes. Comput Graph Forum 2002;21(3):210–8.
[29] Sheffer A, Sturler E. Parameterization of faceted surfaces for meshing using angle based flattening. Eng Comput 2001;17(3):326–37.
[30] Bronstein AM, Bronstein MM, Kimmel R. Generalized multidimensional scaling: a framework for isometry invariant partial surface matching. Proc Natl Acad Sci 2006;103(5):1168–72.
[31] Bronstein AM, Bronstein MM, Kimmel R. Efficient computation of isometry-invariant distances between surfaces. SIAM J Sci Comput 2006;28(5):1812–36.
[32] Sahillioğlu Y, Yemez Y. Coarse-to-fine isometric shape correspondence by tracking symmetric flips. Comput Graph Forum 2013;32(1):177–89.
[33] Lipman Y, Funkhouser T. Möbius voting for surface correspondence. ACM Trans Graph 2009;28(3).
[34] Zeng, Y., Wang, C., Wang, Y., Gu, X., Samaras, D., Paragios, N. Dense non-rigid surface registration using high-order graph matching. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR): 2010. p. 382–389.
[35] Mitra N, Guibas L, Pauly M. Symmetrization. Proc SIGGRAPH 2007;26(63):1–8.
[36] Twigg C, Kacic-Alesic Z. Optimization for sag-free simulations. Proc SCA 2011.
[37] Derouet-Jourdan A, Bertails-Descoubes F, Thollot J. Stable inverse dynamic curves. ACM Trans Graph 2010;29(6).
[38] Derouet-Jourdan A, Bertails-Descoubes F, Daviet G, Thollot J. Inverse dynamic hair modeling with frictional contact. ACM Trans Graph 2013;32(6):159.
[39] Prévost R, Whiting E, Lefebvre S, Sorkine-Hornung O. Make It stand: balancing shapes for 3D fabrication. Proc SIGGRAPH 2013;32(4):81:1–10.
[40] Skouras M, Thomaszewski B, Bickel B, Gross M. Computational design of rubber balloons. Comput Graph Forum 2012;31(2):835–44.
[41] Chen X, Zheng C, Xu W, Zhou K. An asymptotic numerical method for inverse elastic shape design. Proc SIGGRAPH 2014;33(4).
[42] Jacobson A, Kavan L, Sorkine O. Robust inside–outside segmentation using generalized winding numbers. Proc SIGGRAPH 2013;32(4):33:1–12.
[43] Eldar Y, Lindenbaum M, Porat M, Zeevi Y. The farthest point strategy for progressive image sampling. IEEE Trans Image Process 1997;6:1305–15.
[44] Sorkine O, Alexa M. As-rigid-as-possible surface modeling. Comput Graph Forum (SGP) 2007:109–16.
[45] Rusinkiewicz S., Levoy M. Efficient variants of the ICP algorithm. In: Third International Conference on 3-D Digital Imaging and Modeling; 2001. p. 145–152.
[46] Taubin G. A signal processing approach to fair surface design. Proc SIGGRAPH 1995:315–58.
[47] Sumner RW. Mesh Modification Using Deformation Gradients [Ph.D. thesis]. MIT; 2005.
[48] Schuller C, Kavan L, Panozzo D, Sorkine-Hornung O. Locally injective mappings. Comput Graph Forum (Proc SGP) 2013;32(5).
[49] Liu T, Bargteil A, O'Brien J, Kavan L. Fast simulation of mass-spring systems. Proc SIGGRAPH Asia 2013;32(6):209:1–7.
[50] Anguelov D, Srinivasan P, Koller D, Thrun S, Rodgers J, Davis J. Scape: shape completion and animation of people. Trans Graph 2005;24(3):408–16.

[51] Giorgi D, Biasotti S, Paraboschi L. Shape retrieval contest 2007: watertight models track. SHREC Compet 2007.
[52] Sahillioğlu Y, Yemez Y. Multiple shape correspondence by dynamic programming. Comput Graph Forum (Proc Pac Graph) 2014;33(7):121–30.
[53] Shlafman S, Tal A, Katz S. Metamorphosis of polyhedral surfaces using decomposition. Comput Graph Forum 2002;21(3):219–28.
[54] Shamir A. A survey on mesh segmentation techniques. Comput Graph Forum 2008;27(6):1539–56.
[55] Chen X, Golovinskiy A, Funkhouser T. A benchmark for 3d mesh segmentation. Proc SIGGRAPH 2009;28(3).
[56] Shapira L, Shamir A, Cohen-Or D. Consistent mesh partitioning and skeletonisation using the shape diameter function. Vis Comput 2008;24(4):249–59.
[57] Golovinskiy A, Funkhouser T. Randomized cuts for 3d mesh analysis. Proc SIGGRAPH Asia 2008.
[58] Kalogerakis E, Hertzmann A, Singh K. Learning 3d mesh segmentation and labeling. Proc SIGGRAPH 2010.
[59] Pickup D, Sun X, Rosin P, Martin R. Euclidean-distance-based canonical forms for non-rigid 3d shape retrieval. Pattern Recognit 2015;48(8):2500–12.
[60] Pickup D, Sun X, Rosin P, Martin R, Cheng Z, Nie S. et al. Shrec'15 track: canonical forms for non-rigid 3d shape retrieval; 2007.
[61] Tabia H, Daoudi M, Vandeborre J, Colot O. A new 3d-matching method of nonrigid and partially similar models using curve analysis. IEEE Trans PAMI 2011;33(4):852–8.
[62] Coifman R, Lafon S. Diffusion maps. Appl Comput Harmonic Anal 2006; 21(1):5–30.
[63] Zhou K, Huang J, Snyder J, Liu X, Bao H, Guo B, et al. Large mesh deformation using the volumetric graph Laplacian. ACM Trans Graph 2005;24(3):496–503.
[64] Lipman Y, Cohen-Or D, Gal R, Levin D. Volume and shape preservation via moving frame manipulation. ACM Trans Graph 2007;26(5).