Least-squares Morphing of Dynamic Meshes

André Medalha*, Lucas Pagliosa†, Afonso Paiva† and Paulo Pagliosa*

*FACOM – UFMS, Campo Grande
†ICMC – USP, São Carlos

***Abstract*—Morphing is a technique that smoothly transforms a shape onto another. In this paper, we present a method for morphing of two dynamic meshes: mesh sequences representing the keyframes of animated shapes over time. The pipeline of the proposed method comprises two main stages: template-based cross-parameterization and dynamic mesh interpolation. In the cross-parameterization stage, we use a variation of least-squares (LS) meshes to provide a coarse approximation of the geometry of the source mesh onto the target mesh. In our method, the possible candidates to control points of the LS-mesh are detected using an approach based on the Heat Kernel Signature (HKS). Then, an iterative process of fine fitting adds new constraints in the LS-mesh processing. The cross-parameterization is performed just once for any two frames in order to establish a full correspondence between vertices of the source and target meshes. Next, we use such a correspondence in the dynamic mesh interpolation stage to produce the morphing results. The method is entirely mesh-based and does not require the generation of skeletons, mesh segmentation or the use of any additional control structures. Moreover, it does not require the two input meshes to share the same number of vertices or triangles, or the have the same connectivity. The provided results show the robustness and effectiveness of our method.**

I. Introduction

Morphing, widely used in computer animations and geometry processing applications, is an effect that smoothly deforms a surface onto another over a certain time interval. Recently, a great challenge for the special effects industry was the morphing of the actor Mark Ruffalo onto his CGI (computer-generated imagery) character, The Incredible Hulk, in The Avengers movie. It illustrates the importance and the difficulty of the morphing of CGI characters in Computer Graphics.

Given two surfaces represented by triangular meshes, a prerequisite for morphing is to establish a one-to-one mapping between the source and target meshes. Such a mapping, or cross-parameterization, allows to transfer a function (e.g., spatial and/or texture coordinates) from the source to the target, with applications such as detail mapping [1], [2], mesh completion [3], [4] and deformation transfer [5], among others.

In general, cross-mapping methods require an initial correspondence between a number of pairs of features or *control points* on the source and target meshes, usually specified by the user, from which new correspondences between elements of the two meshes can be derived. Different techniques involve some kind of distortion, or deformation, driven by an objective function, or energy, that defines the similarity of such correspondences and satisfies some kind of boundary condition determined by the control points.

Once established a consistent cross-parameterization, morphing is directly obtained by interpolation of the corresponding points on the source and target meshes. Indeed, several cross-parameterization techniques published in the literature have used morphing of static meshes as an application to illustrate their respective implementations [6]–[8].

In this paper, we are concerned with the morphing of *dynamic meshes*. For our purposes, a dynamic mesh is a mesh sequence corresponding to the poses of an animated surface over a given period of time. In this context, Chen et al. [9] recently proposed a morphing framework that can simultaneously interpolate motions and geometries of two input mesh sequences. The framework requires the creation of compatible skeleton-driven cages for two rest poses of the source and target mesh sequences, which are used as control structures in the subsequent steps. These include a hybrid cross-mapping that depends on segmentations of the rest poses according to the cross-sections at the corresponding joints of the compatible skeletons. Two different methods can be employed: a quad-domain-based or a template-based fitting parameterization.

As the main contribution of this paper, we present a simpler alternative method for dynamic mesh morphing which is entirely mesh-based and does not demand the construction of skeletons, mesh segmentation or the use of any additional control structures. In addition, it does not require the two input meshes to share the same number of vertices or triangles or have the same connectivity.

For cross-parameterization, we have reviewed and then implemented a variation of the template-based fitting technique proposed by Yeh et al. [10], which is able to deform the geometry of meshes with arbitrary (but not distinct) genus while imposing fewer restrictions on their shape or pose. The technique comprises two main stages. The former uses initial control points provided by the user as constraints in a reconstruction surface technique based on the concept of least-squares (LS) meshes [11], and is responsible for a coarse approximation of the geometry of the source into the target mesh. The second stage is an iterative process of fine fitting where, at each step, more control points are automatically discovered and used as constraints in the LS-mesh processing.

Our method relies on a single cross-parameterization between any two poses taken from source and target, and uses it to guide the dynamic mesh interpolation of all poses of the input dynamic meshes. The results showed in the paper and the accompanying video attests the effectiveness of the method.

II. Related Work

Morphing is a technique widely employed in Computer Graphics. In the field of geometry processing, particularly,

morphing has been used as an application to demonstrate the effectiveness of several mesh parameterization methods published in the literature.

Informally, a mesh parameterization is a bijective map from a given mesh onto a standard parametric domain which varies according to the genus of the mesh. For instance, genus-0 meshes can be parameterized to spherical domains using extensions of planar techniques [12]–[15]. Mesh of an arbitrary genus can be mapped to the plane [16] or to a topologically equivalent base domain [13], [17], [18]. An indirect mapping between two meshes can be established by using a common standard parametric domain as an intermediate between the source and target meshes [19], [20]. Some authors use the term "domain-based parameterization" to denominate this kind of inter-mesh mapping.

Cross-parameterization techniques, on the other hand, consist of establishing correspondences between source and target meshes directly, without the intervention of a common domain. These usually expect the user provides initial pairwise correspondences, or control points, between features of the source and target meshes, especially in high curvature regions such as ears and fingers. Many of the existing approaches [6], [8], [21]–[23] require that the source and target be of the same genus. The technique presented in [6] is guaranteed to work only on genus-0 meshes, while the approach in [21] finds a maximal non-separated cut graph on each input mesh, ensuring the success of the algorithm on meshes of higher genus. Both methods operate on finding compatible base triangulations on the input models. The techniques introduced in [7], [24], [25] allow for the mapping between meshes of different genus, but with some limitations; in [24], for example, the user is required to manually specify control meshes with the same number of faces for both the source and target meshes. Approaches involving cross-parameterization and competitive learning based on self-organized maps (SOMs) are used in [26] for genus-0 meshes, and extended to higher genus parameterization in [27], [28]. Other methods [5], [8] propose solutions which deform the source mesh onto target mesh, requiring however these have similar orientation and position. In a more recent study, Yeh et al. [10] introduced a cross-parameterization method based on least-squares (LS) fitting [11]. Due to the reported efficiency in the cross-parameterization tests, this was the method we have enhanced in this paper.

Regarding the morphing of animated sequences, to the best or our knowledge, we have found just one method in the literature tackling this subject, proposed by Chen et al. [9]. In order to point out the differences between this method and our dynamic mesh morphing technique, we briefly describe important aspects of their framework, which comprises four steps: mesh sequence representation, hybrid cross-parameterization, motion blending, and dynamic shape interpolation.

In the mesh sequence representation stage, the two input mesh sequences are encoded as a skeleton-driven cage-based control structure. More precisely, kinematic skeletons for a source and a target rest pose mesh are semi-automatically generated using consistent user-specified vertices near to the main joints of both meshes, as detailed in [29]. If those skeletons are not compatible (e.g., they have different segments incident to a joint), then virtual joints are added interactively such that the corresponding joints have the same valence. Next, the rest pose meshes are segmented according to the cross-sections at the corresponding joints of the compatible skeletons. Each segment is then classified as an S2 section, if it is like a cylinder, or an S1 section, otherwise. In the hybrid cross-parameterization stage, mappings between corresponding S1 sections in source and target rest pose meshes are obtained by using a domain-based parameterization similar to [30], whereas corresponding S2 sections are mapped following the template-based fitting process of [8]. The motion blending stage adopts the time warping curve method [31] to find optimal frame pairs of two motions by comparing the similarity of two frames, where the similarity is measured as the distances between corresponding joints. Then, the in-between motion is produced by interpolating the corresponding joint angles and bone lengths. Finally, the dynamic shape interpolation stage generates interpolated geometries for an in-between motion via a skeleton-driven cage-based deformation transfer scheme. Because the motion blending and shape interpolation stages can be processed singly or sequentially, the framework can be used for different purposes, including deformation transfer.

Our method is a much simpler, purely mesh-based alternative specifically designed for dynamic mesh morphing. Following, we present the fundamentals and a complete description of the proposed method.

III. Least-squares Morphing of Dynamic Meshes

In this paper, a mesh is described as a pair $\mathcal{M} = (\mathcal{V}, \mathcal{F})$, where the set $\mathcal{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{|\mathcal{V}|}\}$ represents the geometry and $\mathcal{F} = \{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_{|\mathcal{F}|}\}$ is the topology of $\mathcal{M}$. The coordinates $(x_i, y_i, z_i) \in \mathbb{R}^3$ of the position vector $\mathbf{v}_i \in \mathcal{V}$ define the spatial location of the i-th mesh vertex. (The symbol $\mathbf{v}$ is used throughout to denote both the position of a vertex and the vertex itself.) A mesh triangle $\mathbf{t}_i \in \mathcal{F}$ is defined by $(i_1, i_2, i_3) \in \mathbb{I}^3$ such that $\{\mathbf{v}_{i_1}, \mathbf{v}_{i_2}, \mathbf{v}_{i_3}\} \subset \mathcal{V}$ are the triangle vertices. A *dynamic mesh* is a mesh sequence $\mathcal{A}_\mathcal{M} = \{\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_n\}$ corresponding to the animation keyframes of the surface represented by $\mathcal{M}$ over a time period $T = [t_0, t_n]$. The mesh $\mathcal{M}_i \in \mathcal{A}_\mathcal{M}$ has topology $\mathcal{F}_i \equiv \mathcal{F}$ and represents the pose of the surface at time $t_i = t_0 + i \times d_T$, where $d_T = (t_n - t_0)/(n-1)$.

The proposed method takes as input two mesh sequences $\mathcal{A}_\mathcal{S} = \{\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_n\}$ and $\mathcal{A}_\mathcal{T} = \{\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_n\}$, where $\mathcal{S}$ and $\mathcal{T}$ denote the *source* and the *target* surfaces, respectively. It is assumed that both source and target surfaces are two-manifold and have the same genus. In addition, the method takes two index sets $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ and $\mathcal{R} = \{r_1, r_2, \ldots, r_m\}$. The index pair (c_i, r_i) indicates that the c_i-th vertex of every source mesh in $\mathcal{A}_\mathcal{S}$ is a *control point* corresponding to the r_i-th vertex of every target mesh in $\mathcal{A}_\mathcal{T}$. For each time step $t_i \in T$, $i = 0, 1, \ldots, n$, the method first generates $\mathcal{S}'_i$, a mesh with the same topology of the source mesh $\mathcal{S}_i$ but whose geometry is deformed to closely

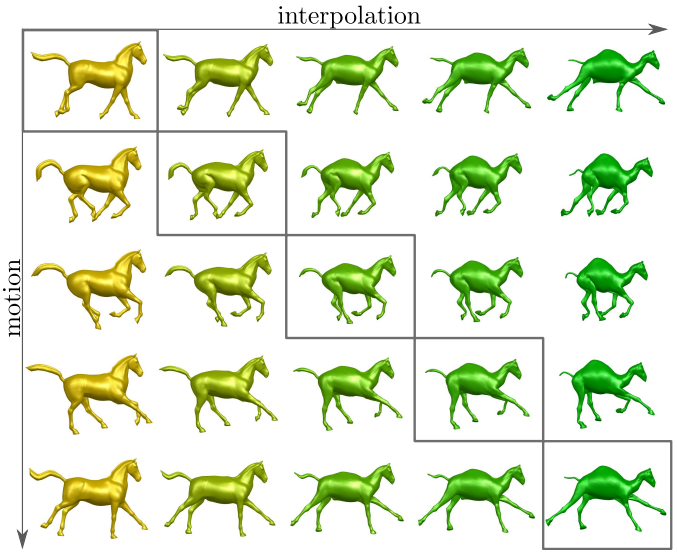

Fig. 1. Method overview. The left column shows the poses $\{\mathcal{S}_i\}$ of the source surface (horse). The right column shows the deformed poses $\{\mathcal{S}'_i\}$, which closely approximate the poses $\{\mathcal{T}_i\}$ of the target surface (camel). The method considers a time period T between $\mathcal{S}_i$ and $\mathcal{S}'_i$ to interpolate the original and deformed source poses at each time step t_i. The resulting dynamic mesh is formed by the interpolated poses $\{\mathcal{D}_i\}$ in the diagonal. The intermediate off-diagonal frames are merely illustrative and not generated by the method.

approximate the geometry of the target mesh $\mathcal{T}_i$. Such deformation is based on a cross-parameterization scheme which establishes a correspondence between a vertex of the source mesh and a point on the target mesh, and is driven by the initial correspondence defined by the input control points. Next, the method computes $\mathcal{D}_i$, a mesh whose geometry is given by the interpolation of the geometries of $\mathcal{S}_i$ and $\mathcal{S}'_i$ at time t_i (supposing $\mathcal{S}_i$ at t_0 and $\mathcal{S}'_i$ at t_n). Note that $\mathcal{D}_0 \equiv \mathcal{S}_0$ and $\mathcal{D}_n \equiv \mathcal{S}'_n \approx \mathcal{T}_n$. The method output is the dynamic mesh $\{\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_n\}$, as illustrated in Figure 1.

The method pipeline encompasses three stages: preprocessing (including dynamic mesh regularization, feature vertex detection, and initial vertex correspondence), cross-parameterization, and dynamic mesh interpolation.

The initial step of the preprocessing ensures that the input animations have the same number of keyframes and/or duration. For cyclic animations with a different number of frames n_s and n_t (where the indices s and t denote source and target, respectively), but equal time interval between consecutive frames, we can set $n = \text{lcm}(n_s, n_t)$, i.e., we replicate the frames of one or both animations to achieve a common number of poses for the input dynamic meshes. If the number of frames is equal, but the durations T_s and T_t are distinct, then we set $d_T = \min(T_s, T_t)/(n-1)$ and interpolate new frames at every d_T in the animation with smaller duration. This results in dynamic meshes with a different number of frames, which are then handled as described above. Other strategies as clipping frames are also possible. In this step we can also subdivide the source and/or target meshes, as discussed in Section V.

The remaining preprocessing steps and stages are described below. Before, it is reviewed the concept of least-squares meshes, a surface reconstruction technique proposed by Sorkine and Cohen-Or [11] and the main tool in which our method is based on (hence the name least-squares morphing).

A. Least-squares Meshes

A LS-mesh $(\mathcal{V}, \mathcal{F})$ is constructed from the connectivity provided by the triangles in $\mathcal{F}$ such that the position of every vertex $\mathbf{v}_i \in \mathcal{V}$ is enforced to be closest to the centroid of its 1-ring vertex neighborhood, $\mathcal{N}_1(\mathbf{v}_i)$, i.e.,

$$\mathbf{L}\,\mathbf{V} = \begin{bmatrix}\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0}\end{bmatrix}^\top, \tag{1}$$

where $\mathbf{V} = \begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2 & \ldots & \mathbf{v}_{|\mathcal{V}|}\end{bmatrix}^\top$ and $\mathbf{L}$ is the Tutte Laplacian of the mesh:

$$[\mathbf{L}]_{ij} = \begin{cases} 1 & \text{if } i = j, \\ -\dfrac{1}{|\mathcal{N}_1(\mathbf{v}_i)|} & \text{if } j \in \mathcal{N}_1(\mathbf{v}_i), \\ 0 & \text{otherwise.} \end{cases}$$

Since $\mathbf{L}$ is singular, it is necessary to provide the coordinates of some *control vertices* in order to obtain a nontrivial solution of Equation (1). Let $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ be the set of indices of m control vertices, and $\{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_m\}$ the prescribed vertex coordinates. One strategy is to employ a scheme of reduction [32] for removing from the system the lines and columns corresponding to the control vertices. The solution of such a reduced system gives the position of the remaining vertices, while the control vertices are positioned exactly at the given coordinates, i.e., the prescribed positions of the control vertices are *hard* constraints in Equation (1).

LS-meshes adopt a distinct approach: the control vertices are positioned closest, in the least-squares sense, to the given coordinates, i.e., the prescribed positions of the control vertices are *soft* constraints. These restrictions are imposed on Equation (1) by adding m rows to the system, which becomes rectangular with dimension $(|\mathcal{V}| + m) \times |\mathcal{V}|$:

$$\mathbf{A}\,\mathbf{V} = \mathbf{B} \Longleftrightarrow \begin{bmatrix}\mathbf{L} \\ \mathbf{F}\end{bmatrix} \mathbf{V} = \begin{bmatrix}\mathbf{b}_1 & \mathbf{b}_2 & \ldots & \mathbf{b}_{|\mathcal{V}|+m}\end{bmatrix}^\top, \tag{2}$$

where

$$[\mathbf{F}]_{ij} = \begin{cases} 1 & \text{if } j = c_i \in \mathcal{C}, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

and

$$\mathbf{b}_i = \begin{cases} \mathbf{0} & \text{if } i \leq |\mathcal{V}|, \\ \mathbf{r}_{c_{i-|\mathcal{V}|}} & \text{if } |\mathcal{V}| < i \leq |\mathcal{V}| + m. \end{cases} \tag{4}$$

System (2) is solved by finding $\mathbf{V}$ that minimizes:

$$\|\mathbf{A}\,\mathbf{V} - \mathbf{B}\|^2 = \|\mathbf{L}\,\mathbf{V}\|^2 + \sum_{k=1}^{m} \|\mathbf{v}_{c_k} - \mathbf{r}_{c_k}\|^2, \tag{5}$$

where $||\cdot||$ is the Frobenius norm. It is also possible to assign individual weights w_k to the control vertices; in that case, the energy to be minimized is:

$$\|\mathbf{L}\,\mathbf{V}\|^2 + \sum_{k=1}^{m} w_k^2 \|\mathbf{v}_{c_k} - \mathbf{r}_{c_k}\|^2.$$

As the value of w_k increases, $\mathbf{v}_{c_k}$ becomes closer to $\mathbf{r}_{c_k}$, at the expense of compromising the fairness conditions.

B. Feature Vertex Detection

Feature vertices can be used as candidates to control points. Motivated by the performance tests conducted in [33], [34], we have adopted for feature vertex detection an approach based on the concept of *Heat Kernel Signature* (HKS), proposed by Sun et al. [35].

For a triangle mesh $\mathcal{M}$, the HKS of a vertex $\mathbf{v}_i \in \mathcal{V}$ is written in terms of the eigenpairs (λ_k, ϕ_k) resulting from the spectral decomposition of the discrete Laplace-Beltrami operator (LBO) of $\mathcal{M}$:

$$\mathrm{HKS}_t(\mathbf{v}) = \sum_{k=1}^{n_v} e^{-\lambda_k t} \phi_k(\mathbf{v}_i)^2, \tag{6}$$

where $n_v \leq |\mathcal{V}|$. (In particular, we used the cotangent discretization [36] of the LBO.) The vertex $\mathbf{v}_i$ is considered a feature vertex if $\mathrm{HKS}_t(\mathbf{v}_i) > \mathrm{HKS}_t(\mathbf{v}_j)$, for every vertex $\mathbf{v}_j$, $j \in \mathcal{N}_2(\mathbf{v}_i)$, where $\mathcal{N}_2(\mathbf{v}_i)$ is the 2-ring vertex neighborhood of $\mathbf{v}_i$, for a fixed, large value of t (we used $t = 4 \ln 10/\lambda_2$ and $n_v = \min(300, |\mathcal{V}|)$ [35] in our experiments). Let $\mathcal{P}_\mathcal{M}$ be the index set of all feature vertices of $\mathcal{M}$. In the proposed method, the feature detection can be applied once for all poses of the input dynamic meshes, resulting $\mathcal{P}_{\mathcal{A}_\mathcal{S}} = \cup_{i=0}^{n} \mathcal{P}_{\mathcal{S}_i}$ and $\mathcal{P}_{\mathcal{A}_\mathcal{T}} = \cup_{i=0}^{n} \mathcal{P}_{\mathcal{T}_i}$, or only in a number of poses indicated by the user. This approach tries to capture, for example, those vertices that are in plain regions in some poses but can be in high curvature regions in others.

C. Initial Vertex Correspondence

In the last preprocessing step, the indices (c_k, r_k) of the initial pairs of corresponding vertices are specified, i.e., the sets $\mathcal{C}$ and $\mathcal{R}$ which are used in the cross-parameterization to guide the initial mapping process. In the current implementation (as in other methods, see Section II), this task is performed by the user, who is responsible for identifying some source and target vertices with similar semantic correspondence.

In order to help the user, we provide a graphical tool, called ILSM (Interactive LS-Mesh), which helps he/she interactively add, remove, and modify pairs of corresponding vertices while viewing the result of the coarse fitting on-the-fly, as shown in Figure 2. The user can choose any pairs of source and target poses to deal with. The feature vertices detected previously are available in ILSM and can be used as hints of control points.

D. Cross-Parameterization

In the cross-parameterization stage, it is employed a variation of the template-based fitting technique proposed by Yeh et al. [10]. It takes as input a source mesh $\mathcal{S} = (\mathcal{V}_s, \mathcal{F}_s)$ and a target mesh $\mathcal{T} = (\mathcal{V}_t, \mathcal{F}_t)$, where the indices s e t indicate source and target elements, respectively. (The meshes $\mathcal{S}$ and $\mathcal{T}$ can be resting poses of the source and target surfaces or any poses taken from $\mathcal{A}_\mathcal{S}$ and $\mathcal{A}_\mathcal{T}$, respectively. In the last case, the pose indices are omitted for sake of clarity.) The main goal is to obtain a mapping between the meshes from which it is possible to deform the source mesh in order to closest approximate the target geometry. The technique consists of

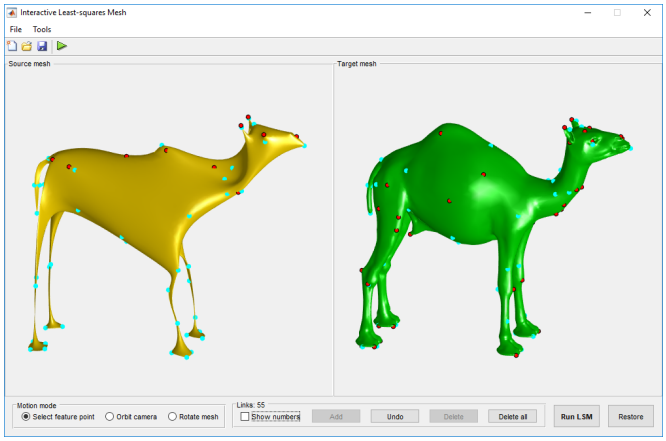

Fig. 2. ILSM: a graphical tool for interactive specification of pairs of corresponding vertices (drawn as cyan circles) to be used in the coarse fitting. Red circles are feature points detected with HKS. The left panel shows the coarse fitting of the source mesh. The right panel shows the target mesh.

two stages: coarse and fine fitting. The main improvements we have made in our implementation are in the last stage, more precisely, in the reliable correspondence criteria, curvature measurement, and surface detail fitting, as described below.

1) Coarse fitting: The coarse fitting stage reconstructs the source mesh $\mathcal{S}$ as a LS-mesh by solving Equation (5) to determine $\mathcal{V}_s$, where the soft restriction for the position of the c_i-th control vertex is $\mathbf{r}_{c_i} = \mathbf{v}^t_{r_i}$ ($c_i \in \mathcal{C}$ and $r_i \in \mathcal{R}$), i.e.,

$$\|\mathbf{A}_s\,\mathbf{V}_s - \mathbf{B}\|^2 = \|\mathbf{L}_s\,\mathbf{V}_s\|^2 + \sum_{i=1}^{m} \|\mathbf{v}^s_{c_i} - \mathbf{v}^t_{r_i}\|^2. \tag{7}$$

Since the positions of the control vertices in $\mathcal{S}$ are constrained by the corresponding vertices in $\mathcal{T}$, the reconstructed source mesh approximates more or less roughly the target.

2) Fine fitting: The fine fitting iteratively computes *reliable correspondence points* between $\mathcal{T}$ and the LS-mesh obtained in the coarse fitting, and then updates the geometry of $\mathcal{S}$ accordingly, essentially adding extra soft constraints to the LS-mesh processing. At each iteration step, reliable points between $\mathcal{T}$ and the transformed $\mathcal{S}$ are determined as follows: each vertex $\mathbf{v}^t_i$ is projected onto $\mathcal{S}$ along $\mathbf{n}^t_i$, the normal to target surface at $\mathbf{v}^t_i$. The intercepted triangle $\mathbf{t}^s_j$ (if any) and $\mathbf{v}^t_i$ make a *reliable pair* $(\mathbf{v}^t_i, \mathbf{t}^s_j)$ if: 1) the angle between $\mathbf{n}^t_i$, the unit normal to target surface at $\mathbf{v}^t_i$, and $\mathbf{N}^s_j$, the unit normal of the triangle $\mathbf{t}^s_j$, is less than 90^o, i.e., $\mathbf{n}^t_i \cdot \mathbf{N}^s_j > 0$; and 2) at least one vertex $\mathbf{v}^t_j$, $j \in \mathcal{N}_1(\mathbf{v}^t_i)$, projects onto a source triangle belonging to the 2-ring face neighborhood of $\mathbf{t}^s_i$. This approach leads to an increase in the number of reliable correspondences with few interactions, without requiring that the target mesh is much more refined that the source, as in the original method.

Next, it is performed a *dual relaxation* in order to compute the geometry of the dual mesh $\mathcal{S}^* = (\mathcal{V}^*_s, \mathcal{F}^*_s)$, where $\mathcal{V}^*_s$ and $\mathcal{F}^*_s$ are the dual vertices and dual faces of the primal mesh $\mathcal{S}$, respectively. (Each vertex $\mathbf{v}^{s*}_i \in \mathcal{V}^*_s$ corresponds to the centroid of the triangle $\mathbf{t}^s_i \in \mathcal{F}_s$, and two dual vertices are connected if their corresponding primal triangles are neighbors, i.e., share an edge in the primal mesh.) For each primal triangle

$\mathbf{t}_i^s$ that received one or more reliable vertices, the centroid $\mathbf{c}_i^t$ of these vertices is used as hard constraint for the corresponding "matched" dual vertex $\mathbf{v}_i^{s*}$ in the linear system

$$\mathbf{L}_s^* \mathbf{V}_s^* = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}_{|\mathcal{F}_s| \times 3}^\top . \quad (8)$$

The solution of Equation (8) (e.g. by reduction) gives the positions of the unmatched dual vertices which are uniformly flattened in the regions surrounded by the nearby matched ones, i.e., with reliable correspondences.

Once the position of each vertex $\mathbf{v}_i^{s*} = \mathbf{c}_i^{s*}$ in dual mesh is obtained, it is used as a soft constraint in the LS-mesh setup to refit the dual source mesh $\mathcal{S}^*$. More precisely, we compute $\mathbf{V}_s^*$ that minimizes

$$\omega^2 \|\mathbf{L}_s^* \mathbf{V}_s^*\|^2 + \sum_{i=1}^{|\mathcal{F}_s|} {w_i}^2 \|\mathbf{v}_i^{s*} - \mathbf{c}_i^{s*}\|^2, \quad (9)$$

where the smoothness term is controlled by the weight ω (we used $\omega = 3$ in our experiments) and the individual dual vertex constraints are controlled by the weights w_i, which depend on a curvature measure on $\mathcal{T}$. We adopt as curvature measure for every vertex $\mathbf{v}_i^t$:

$$h_i^t = 1 - \frac{|\kappa_{1_i}^t| + |\kappa_{2_i}^t|}{\sum_{j=1}^{|\mathcal{V}_t|} \left(|\kappa_{1_j}^t| + |\kappa_{2_j}^t| \right)}, \quad (10)$$

where $\kappa_{1_i}^t$ and $\kappa_{2_i}^t$ are the principal curvatures [37] at $\mathbf{v}_i^t$. In our experiments, we consider the vertex is in a high-curvature region if $h_i^t \geq 0.5$.

The weights w_i in Equation (9) are set as follows. If $\mathbf{v}_i^{s*}$ is a matched dual vertex, then w_i is the average of the curvature measure of all reliable vertices projecting onto the primal triangle $\mathbf{t}_i^s$; otherwise, w_i is the average of the curvature measure of all target vertices. This scheme allows for a tighter approximation in high-curvature areas of $\mathcal{T}$ (where the weights are higher) and good triangle quality in flat areas.

The fitting result obtained in Equation (9) is transformed back to the primal domain by finding $\mathbf{V}_s$ that minimizes the following LS-system:

$$\|\mathbf{L}_s \mathbf{V}_s\|^2 + \alpha C_1(\mathbf{V}_s) + C_2(\mathbf{V}_s). \quad (11)$$

The first energy term C_1 is related to the dual mesh fitting (we adopted $\alpha = 3.0$ in our experiments). Since each dual vertex $\mathbf{v}_i^{s*}$ corresponds to a primal triangle $\mathbf{t}_i^s$, this term forces the centroid of the primal vertices $\{\mathbf{v}_{i_1}^s, \mathbf{v}_{i_2}^s, \mathbf{v}_{i_3}^s\}$ of $\mathbf{t}_i^s$ to fit the dual vertex $\mathbf{v}_i^{s*}$:

$$C_1(\mathbf{V}_s) = \sum_{i=1}^{|\mathcal{F}|_s} w_i^2 \left\| \frac{1}{3}(\mathbf{v}_{i_1}^s + \mathbf{v}_{i_2}^s + \mathbf{v}_{i_3}^s) - \mathbf{v}_i^{s*} \right\|^2, \quad (12)$$

where the weights w_i are computed using the surface curvature measure as described above. To improve the fitting of surface details, the energy term C_2 is added. For each high-curvature vertex $\mathbf{v}_i^t$ in $\mathcal{T}$, the reliable projection criteria described earlier is used to find a corresponding position in the transformed source mesh $\mathcal{S}$. Let $(\mathbf{v}_i^t, \mathbf{t}_j^s)$ be the reliable pair and $\mathbf{p}_i^s$ the point resulting from the projection of $\mathbf{v}_i^t$ onto $\mathbf{t}_j^s$. This point can be written as a linear combination of the $\mathbf{t}_j^s$'s vertices using barycentric coordinates: $\mathbf{p}_i^s = b_1 \mathbf{v}_{j_1}^s + b_2 \mathbf{v}_{j_2}^s + b_3 \mathbf{v}_{j_3}^s$. For a set $\mathcal{B}$ of matched high-curvature vertices on $\mathcal{T}$:

$$C_2(\mathbf{V}_s) = \sum_{i \in \mathcal{B}} {h_i^t}^2 \left\| (b_1 \mathbf{v}_{j_1}^s + b_2 \mathbf{v}_{j_2}^s + b_3 \mathbf{v}_{j_3}^s) - \mathbf{v}_i^t \right\|^2. \quad (13)$$

The fine fitting process is repeated several times, each step making the source mesh closest to the target. We stop the iterations when the relation between the number of reliable and projected vertices is less than 4%.

3) Cross-Parameterization of Dynamic Meshes: In the proposed method, the cross-parameterization scheme is applied only once to a single pair $\mathcal{S}$ and $\mathcal{T}$ (as commented above, the source and target meshes can be resting poses or any poses from input dynamic meshes, e.g., $\mathcal{S}_0$ and $\mathcal{T}_0$). Since the cross-parameterization is the most time-consuming stage, such an approach leads to an increase in the computational performance of the method. In order to ensure that this single cross-parameterization can then be appropriately used for shape interpolation of *all* poses of the input dynamic meshes, we extended the fine fitting as follows. After ending the iterations (less than 25 in all our tests), for each vertex $\mathbf{v}_j^t \in \mathcal{V}_t$ we find the nearest vertex $\mathbf{v}_i^s \in \mathcal{V}_s$, and set (i, j) as a pair of corresponding points. Next, we use this full vertex correspondence as soft constraints to reconstruct the source as an LS-mesh. In fact, since the topology of all poses of a dynamic mesh is the same, we can use such a correspondence to deform every pose $\mathcal{S}_i \in \mathcal{A}_\mathcal{S}$ into the corresponding $\mathcal{T}_i \in \mathcal{A}_\mathcal{T}$. Moreover, this scheme helps to eliminate some artifacts that are sensitive to threshold adopted to detect high curvature regions.

E. Dynamic Mesh Interpolation

The last stage generates the output dynamic mesh $\mathcal{A}_\mathcal{D}$. For each pair $\mathcal{S}_i \in \mathcal{A}_\mathcal{S}$, $\mathcal{T}_i \in \mathcal{A}_\mathcal{T}$, a new LS-mesh, $\mathcal{S}_i'$, is constructed with an extra coarse fitting step using the full vertex correspondence obtained in the cross-parameterization stage. This extra step is required to smoothly position the vertices of $\mathcal{S}_i'$, since distinct source vertices can correspond to the same target vertex. Then, a new mesh, $\mathcal{D}_i$, is created and added into $\mathcal{A}_\mathcal{D}$. We consider the time interval T to morph $\mathcal{S}_i$ into $\mathcal{S}_i'$, i.e., the position of a vertex of $\mathcal{D}_i$ is given by the linear interpolation of the corresponding vertices of $\mathcal{S}_i$ and $\mathcal{S}_i'$ at time t_i. Results of the method are presented in Section V.

IV. Aspects of Implementation

The source code was almost all implemented in MATLAB, including the graphical tool ILSM introduced in Section III. The main reason for using MATLAB is the number of mathematical functions available and the ease of manipulating matrix data; as a result, MATLAB provides an environment in which it is possible to implement numerical solutions faster than using traditional languages, such as C++. We used the Toolbox Graph by Gabriel Peyre[1] to compute the principal curvatures and LBO matrix; the remainder of the MATLAB code was developed by the authors.

[1]Available at mathworks.com/matlabcentral/fileexchange/5355.

The function responsible for computing the reliable correspondences in the fine fitting stage was implemented in C++ due to computational performance. In order to find the projection of a vertex $\mathbf{v}_i^t$ onto the deformed source mesh $\mathcal{S}$, we shoot a ray from $\mathbf{v}_i^t$ along the direction $\mathbf{n}_i^t$ and verify if the ray hits a triangle $\mathbf{t}_j^s$ of $\mathcal{S}$. The search is performed inside a sphere centered at $\mathbf{v}_i^t$ to avoid the intersection with regions of $\mathcal{S}$ far from the target vertex, as depicted in Figure 3. More precisely, we find $\mathbf{p}_i = \mathbf{v}_i^t + d_i\,\mathbf{n}_i^t$ (closest to $\mathbf{v}_i^t$) such that $|d_i| \leq r_{k_i}$, where r_{k_i} is the search radius:

$$r_{k_i} = \max_{j \in \mathcal{N}_k(\mathbf{v}_i^t)} ||\mathbf{v}_i^t - \mathbf{v}_j^t||^2.$$

In our experiments, we adopted $r_{k_i} \approx k\, r_{1_i}$, with $k = 7$. To accelerate the ray/triangle intersection calculations, we employed a bounding volume hierarchy (BVH) [38] in each iteration of the fine fitting for partitioning the triangles of the deformed source mesh $\mathcal{S}$. After computing the intersection point, if any, the vertex $\mathbf{v}_i^t$ is classified as reliable or unreliable as explained in Section III. The C++ function for reliable correspondence was made available to MATLAB by using the `mex` interface (see mathworks.com/help/matlab/ref/mex.thml).

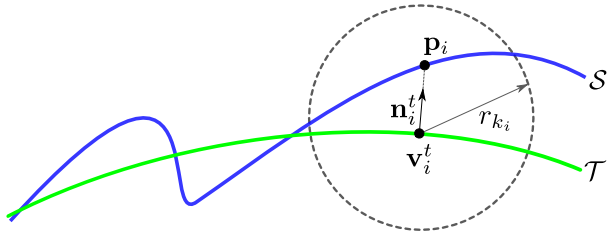

Fig. 3. Ray-casting to find $\mathbf{p}_i$, the projection onto $\mathcal{S}$ of the target vertex $\mathbf{v}_i^t$ along its normal $\mathbf{n}_i^t$, inside a sphere of radius r_{k_i}.

V. Results and Discussions

Table I summarizes the data for the models we have used to test our implementation. For each model are shown the name of the dynamic mesh, the number of vertices ($|\mathcal{V}|$), the number of triangles ($|\mathcal{F}|$), and the number of feature points ($|\mathcal{P}|$) detected by the HKS-based approach described in Section III.

TABLE I
Data for the models used for tests.

Mesh	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{F}\rvert$	$\lvert\mathcal{P}\rvert$
Allosaurus	7,102	13,192	23
Horse	8,431	16,843	43
Man	52,609	95,040	55
Camel	21,887	43,814	63
Whale	18,055	34,880	85

We apply our pipeline to the test cases presented in Table II, with $d_T = 1/24$ seconds. For each case, we identify the source and target dynamic meshes, the number of poses (n), the number of initial control points (m) used in the coarse fitting, and the processing time for the cross-parameterization and dynamic mesh interpolation stages, CP and DMI, respectively (in seconds). The processing time for the HKS-based feature point detection, the only fully automatic preprocessing step, varies between 9.0 (for the horse) and 20.0 (for the whale) seconds per frame. All the tests were performed on a computer equipped with CPU Intel Core i7-5500U and 16 GB of RAM.

TABLE II
Test cases.

#	Source	Target	n	m	CP (s)	DMI (s)
1	Allosaurus	Horse	48	40	34.0	10.0
2	Allosaurus	Man	50	45	35.2	8.5
3	Horse	Camel	48	37	19.2	8.3
4	Horse	Whale	80	35	17.0	13.6

In test case 1, we refined the target mesh (horse) by using a scheme which is described below, resulting in a mesh with $\mathcal{V} = 75,822$ and $\mathcal{F} = 151,587$. Frames of test cases 1, 2, and 4 are depicted in Figure 4, Figure 5, and Figure 6, respectively (see also the accompanying video). In these figures, the first and second rows show poses of the source and target dynamic meshes, $\{\mathcal{S}_i\}$ and $\{\mathcal{T}_i\}$, respectively, and the third row show the morphing results, $\{\mathcal{D}_i\}$. Frames of the test case 3 were used in the method overview, as illustrated in Figure 1.

Our last test case is the morphing of the dinosaur in all other models: whale, horse, man, camel and back to the dinosaur. Frames resulting from morphing can be seen in Figure 7.

A. Discussion

Experiments have shown the proposed method is robust to deal with models with a distinct number of articulated parts, as can be seen, for example, in the tails that arise and disappear in the morphing of Figure 7.

For the test cases 1-4, the total processing time for the cross-parameterization and dynamic mesh interpolation varies between 27.5 seconds and 45.0 seconds, as reported in Table II. While this time seems to be quite appropriate for our MATLAB-based implementation, we argue it is not feasible to make a fair comparison with the running time presented in [9], since the test platforms, models, and supposedly programming languages are distinct.

Regarding the size of the meshes, the target mesh $\mathcal{T}$ must provide more details, and, therefore, be a bit more refined than the source mesh $\mathcal{S}$. That is necessary once the neighboring vertices of $\mathbf{v}_i^t$ should project onto neighboring faces of $\mathbf{t}_j^s$ to establish a reliable correspondence. However, this would not be possible if the set $\mathcal{V}_t$ is very small or irregularly distributed in space. To deal with this issue, we implement a MATLAB function that refines a triangle mesh by splitting the mesh edges in a number of semi-edges, which are used to tessellate the interior of the mesh triangles as illustrated in Figure 8.

B. Limitations

The importance of the initial vertex correspondence is key to achieve a correct cross-parameterization with direct impact on the fine fitting stage. In our experiments, we have observed that the fine fitting is not always capable to properly handle poor coarse fitting meshes reconstructed from badly chosen control points. Selecting adequate initial control points can be

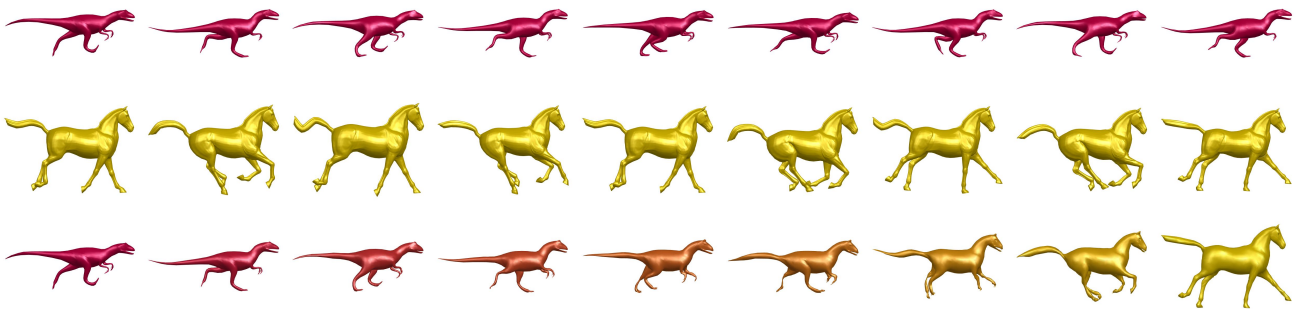

Fig. 4. Test case 2: Allosaurus to horse.

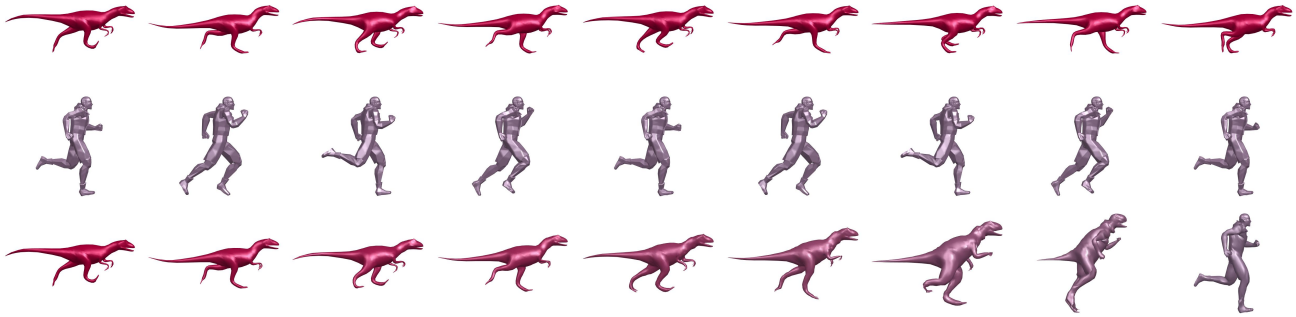

Fig. 5. Test case 3: Allosaurus to man.

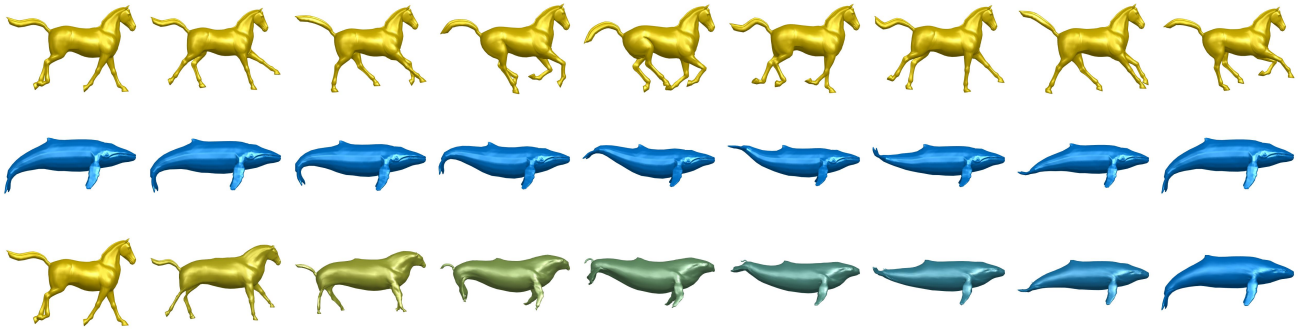

Fig. 6. Test case 3: horse to whale.

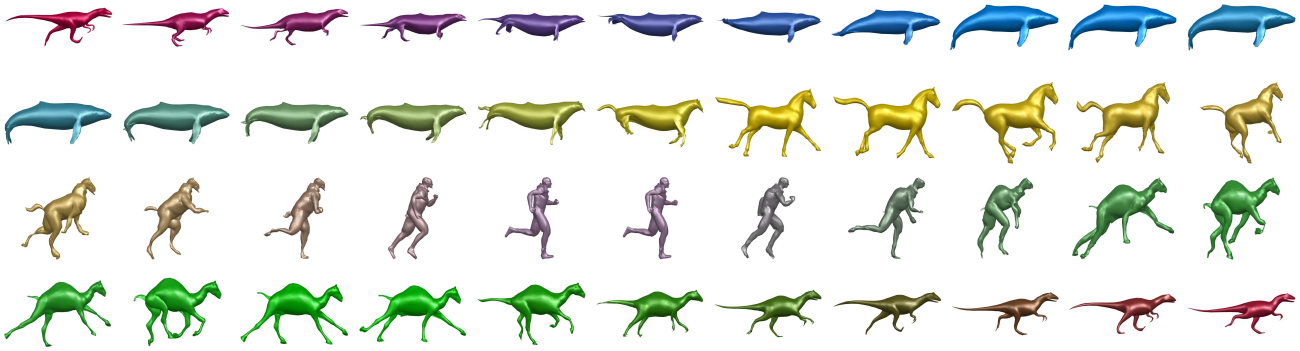

Fig. 7. Allosaurus morphing onto all other models and back to Allosaurus.

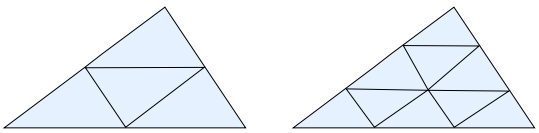

Fig. 8. Mesh refinement. First, each mesh edge is split into a number $e > 1$ of semi-edges specified by the user. Next, each mesh triangle is tessellated into e^2 new triangles by connecting the new vertices in segments parallel to the triangle edges. In these examples, we used $e = 2$ (left) and $e = 3$ (right).

a repetitive, trial-and-error process. For this, our graphical tool ILSM is a very valuable and helpful resource.

VI. Conclusion and Future Work

In this paper, we have presented an alternative method for dynamic mesh morphing. Our method is simpler than the most recent related method published in the literature [9], since it is purely mesh-based and does not require skeletons, mesh subdivision or any control structures. Moreover, the proposed method relies on a unique template-based fitting cross-parameterization technique, centered at the concept of least-squares (LS) meshes and adapted for dealing with dynamic meshes. The cross-parameterization stage establishes a full vertex correspondence between the source and target meshes from which we can perform dynamic mesh interpolations to obtain the morphing results. The results presented in the paper show the effectiveness of the proposed method.

The main drawback of the method is the manual setup of some initial vertex correspondence used to guide the cross-parameterization stage. In addition, the method can be only applied to meshes with the same genus. We will also address these issues in future work.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. The authors were supported by CAPES (Brazilian Federal Agency for Support and Evaluation of Graduate Education, DS-1206477/M) and FAPESP (São Paulo Research Foundation).

References

[1] A. Sheffer, E. Praun, and K. Rose, "Mesh parameterization methods and their applications," *FTCGV*, vol. 2, no. 2, pp. 105–171, 2006.

[2] K. Hormann, K. Polthier, and A. Sheffer, "Mesh parameterization: theory and practice," in *SIGGRAPH Asia Courses*, 2008, pp. 12:1–12:87.

[3] B. Allen, B. Curless, and Z. Popović, "Articulated body deformation from range scan data," *ACM TOG*, vol. 21, no. 3, pp. 612–619, 2002.

[4] B. Allen, B. Curless, and Z. Popović, "The space of human body shapes: reconstruction and parameterization from range scans," *ACM TOG*, vol. 22, no. 3, pp. 587–594, 2003.

[5] R. W. Sumner and J. Popović, "Deformation transfer for triangle meshes," *ACM TOG*, vol. 23, no. 3, pp. 399–405, 2004.

[6] V. Kraevoy and A. Sheffer, "Cross-parameterization and compatible remeshing of 3d models," *ACM TOG*, vol. 23, no. 3, pp. 861–869, 2004.

[7] T.-Y. Lee, C.-Y. Yao, H.-K. Chu, M.-J. Tai, and C.-C. Chen, "Generating genus-n-to-m mesh morphing using spherical parameterization," *CAVW*, vol. 17, no. 3-4, pp. 433–443, 2006.

[8] L. Zhang, L. Liu, Z. Ji, and G. Wang, "Manifold parameterization," *LNCS*, vol. 4035, pp. 160–171, 2006.

[9] X. Chen, J. Feng, and D. Bechmann, "Mesh sequence morphing," *CGF*, pp. 179–190, 2015.

[10] I.-C. Yeh, C.-H. Lin, O. Sorkine, and T.-Y. Lee, "Template-based 3d model fitting using dual-domain relaxation," *IEEE TVCG*, vol. 17, no. 8, pp. 1178–1190, 2011.

[11] O. Sorkine and D. Cohen-Or, "Least-squares meshes," in *SMI*, 2004, pp. 191–199.

[12] C. Gotsman, X. Gu, and A. Sheffer, "Fundamentals of spherical parameterization for 3d meshes," in *I3D '06*, 2006, pp. 28–29.

[13] E. Praun and H. Hoppe, "Spherical parametrization and remeshing," *ACM TOG*, vol. 22, no. 3, pp. 340–349, 2003.

[14] A. Sheffer, C. Gotsman, and N. Dyn, "Robust spherical parameterization of triangular meshes," *Computing*, vol. 72, no. 1-2, pp. 185–193, 2004.

[15] I. Friedel, P. Schröder, and M. Desbrun, "Unconstrained spherical parameterization," in *SIGGRAPH Sketches*, 2005.

[16] X. Gu, S. J. Gortler, and H. Hoppe, "Geometry images," *ACM TOG*, vol. 21, no. 3, pp. 355–361, 2002.

[17] V. Kraevoy and A. Sheffer, "Template-based mesh completion," in *SGP '05*, 2005, pp. 13–22.

[18] A. Khodakovsky, N. Litke, and P. Schröder, "Globally smooth parameterizations with low distortion," *ACM TOG*, vol. 22, no. 3, pp. 350–357, 2003.

[19] M. Alexa, "Recent advances in mesh morphing," *CGF*, vol. 21, no. 2, pp. 173–198, 2002.

[20] V. Nivoliers, D. Yan, and B. Lvy, "Fitting polynomial surfaces to triangular meshes with Voronoi squared distance minimization," in *Meshing Roundtable*, 2011, pp. 601–617.

[21] J. Schreiner, A. Asirvatham, E. Praun, and H. Hoppe, "Inter-surface mapping," *ACM TOG*, vol. 23, no. 3, pp. 870–877, 2004.

[22] C. Carner, M. Jin, X. Gu, and H. Qin, "Topology-driven surface mappings with robust feature alignment," in *VIS '05*, 2005, pp. 543–550.

[23] A. Lee, D. Dobkin, W. Sweldens, and P. Schrder, "Multiresolution mesh morphing," in *SIGGRAPH 99*, 1999, pp. 343–350.

[24] D. DeCarlo and J. Gallier, "Topological evolution of surfaces," in *Graphics Interface*, 1996, pp. 194–203.

[25] J. Bennett, V. Pascucci, and K. Joy, "Genus oblivious cross parameterization: Robust topological management of inter-surface maps," in *PG '07*, 2007, pp. 238–247.

[26] C.-Y. Liou and Y.-T. Kuo, "Conformal self-organizing map for a genus-zero manifold," *TVC*, vol. 21, no. 5, pp. 340–353, 2005.

[27] S. Matsui, K. Aoki, and H. Nagahashi, "3d triangular mesh parameterization with semantic features based on competitive learning methods," *IEICE*, vol. E91-D, no. 11, pp. 2718–2726, 2008.

[28] K. Morooka, S. Matsui, and H. Nagahashi, "Self-organizing deformable model for mapping 3d object model onto arbitrary target surface," in *3DIM '07*, 2007, pp. 193–200.

[29] X. Chen and J. Feng, "Adaptive skeleton-driven cages for mesh sequences," *CAVW*, vol. 25, no. 3-4, pp. 445–453, 2014.

[30] C.-Y. Yao, H.-K. Chu, T. Ju, and T.-Y. Lee, "Compatible quadrangulation by sketching," *CAVW*, vol. 20, no. 2-3, pp. 101–109, 2009.

[31] M. Gleicher, H. J. Shin, L. Kovar, and A. Jepsen, "Snap-together motion: assembling run-time animations," *ACM TOG*, vol. 22, no. 3, pp. 702–702, 2003.

[32] O. Zienkiewicz, R. Taylor, and J. Zhu, *The Finite Element Method: Its Basis and Fundamentals*, 7th ed. Elsevier, 2013.

[33] A. M. Bronstein, M. M. Bronstein, B. Bustos, U. Castellani, M. Crisani, B. Falcidieno, L. J. Guibas, I. Kokkinos, V. Murino, M. Ovsjanikov, G. Patané, I. Sipiran, M. Spagnuolo, and J. Sun, "SHREC'10 Track: feature detection and description," in *3DOR '10*, 2010, pp. 79–86.

[34] E. Boyer, A. M. Bronstein, M. M. Bronstein, B. Bustos, T. Darom, R. Horaud, I. Hotz, Y. Keller, J. Keustermans, A. Kovnatsky, R. Litman, J. Reininghaus, I. Sipiran, D. Smeets, P. Suetens, D. Vandermeulen, A. Zaharescu, and V. Zobel, "SHREC 2011 Track: robust feature detection and description benchmark," in *3DOR '11*, 2011, pp. 71–78.

[35] J. Sun, M. Ovsjanikov, and L. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," *CGF*, vol. 28, no. 5, pp. 1383–1392, 2009.

[36] B. Lévy and H. R. Zhang, "Spectral mesh processing," in *SIGGRAPH Courses*, 2010, pp. 8:1–8:312.

[37] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun, "Anisotropic polygonal remeshing," *ACM TOG*, vol. 22, no. 3, pp. 485–493, 2003.

[38] M. Pharr, W. Jakoband, and G. Humphreys, *Physically Based Rendering: From Theory To Implementation*, 3rd ed. Morgan Kaufmann, 2016.