Robust One-to-One Sweeping with Harmonic *S*-*T* Mappings and Cages

Shengyong Cai[1] and Timothy J. Tautges[2]

[1] University of Wisconsin-Madison, Madison, WI 53706, U.S.A.
shengyongcai@gmail.com
[2] MCS, Argonne National Laboratory, Madison, WI 53706, U.S.A.
tautges@mcs.anl.gov

Abstract. A sweeping algorithm can generate hexahedral meshes by sweeping an all-quad mesh on the *source* surface to the *target* surface. For one-to-one sweeping, the most difficult thing is to generate an all-quad mesh on the *target* surface which has the same mesh connectivity as that of the *source* surface. The traditional method is to use the affine transformation, like translation, rotation, scaling or combinations of them. This method works very well on the convex cases, while it fails for concave and multiply-connected surfaces. In this paper, *harmonic function* is used to map meshes from a *source* surface to its *target* surface. The result shows that it can generate an all-quad mesh on the *target* surface with good quality without any inverted elements and thus avoid expensive smoothing algorithm (*untangling*). In order to generate interior nodes between the source and target surface, cage-based deformation method is applied with good mesh quality as well.

Keywords: Harmonic, Sweeping, Hexahdral, Mesh Generation, One-to-One.

1 Introduction

In many applications such as *Computational Fluid Dynamics* (*CFD*) [1] and *Computational Structural Mechanics* (*CSM*) [2], a hexahedral mesh is preferred over a tetrahedral mesh. According to Ref. [3], there are two classes of methods for generating all-hexahedral meshes, namely, indirect methods which convert from a tetrahedral mesh to a hexahedral mesh [4], and direct methods. This latter may be further classified as *Grid-based* [5], *Medial Surface* [6, 7], *Plastering* [8], *Whisker Weaving* [9] and *Sweeping* [10]. Because it is difficult to combine or divide tetrahedral elements in such a way to guarantee the formation of all-hexahedra, the indirect methods are neither reasonable nor tractable for mesh generation [3]. For the *Grid-based* methods, mesh quality at the boundary of a volume is very poor and interior hex elements are not aligned with boundary hex elements (this is not

J. Sarrate & M. Staten (eds.), *Proceedings of the 22nd International Meshing Roundtable*,
DOI: 10.1007/978-3-319-02335-9_1,

good for *CFD*). The *Medial Surface* method generates a hexahedral mesh by decomposing volumes, which is an extension of *Medial Axis* method. The decomposed volumes are usually meshed with midside subdivision. However, this only works for geometry with 3-valent corner vertices, and less reliably for general geometry. *Plastering* is a 3D extension of the paving algorithm, and the *Whisker Weaving* method builds the dual of hexahedral meshes first then embeds meshes in 3D. Neither *Plastering* nor *Whisker Weaving* has been shown to be robust for general 3D models.

While all-hexahedral mesh generation on general three dimensional geometries remains an elusive goal, algorithms to mesh two-and-one-half dimensional geometries, generally referred to as sweeping or projection methods, continue to be important [10, 11, 12, 13]. In real-world applications, many geometry models can be constructed by sweeping in *CAD* software (*Pro/E*, *UG*, *Solidworks*, etc.) and subsequently hex-meshed using sweeping. More complicated models can often be decomposed into 2.5D pieces which are individually sweepable. Therefore, in practice, geometric decomposition followed by sweep-meshing remains the workhorse approach for generating hexahedral meshes.

The traditional one-to-one sweeping procedure for all-hexahedral meshes consists of four steps: (1) generate an all-quad mesh on a *source* surface; (2) project an all-quad mesh from a *source* surface to its *target* surface; (3) generate the structured all-quadrilateral meshes on the *linking* surface(s); (4) generate the hexahedral meshes, including interior nodes and elements, for volumes. The *source* and *target* surface may have different shapes, areas and/or curvatures, but they must be topologically equivalent homeomorphic. Of the above four steps, the two most difficult steps are to morph a *source* mesh to its *target* surface and locate interior nodes between them. This is especially true for volumes with concave features and non-simply connected *source*/*target* surfaces.

In this paper, we describe a mesh morphing method that uses the *harmonic function* to generate meshes on the *target* surface with adequate mesh quality, and a *cage-based* method for locating interior nodes that also achieves improved mesh quality. In combination, these methods are used to generate good-quality hexahedral meshes using sweeping for which previous sweeping methods fail. The remainder of this paper is structured as follows: Part 2 summarizes recent works about one-to-one sweeping and surface mesh morphing. Then *harmonic mapping* is introduced and surface correspondence establishment is described as well in Part 3. Finally, a *cage-based* method is applied in order to locate interior nodes between the *source* and *target* surface.

2 Previous Work

P. Knupp [10] devised two algorithms to locate interior nodes during sweeping: linear transformations between the bounding node loops and smoothing.

In his approach, a *source* surface was given, consisting of a layer of mesh elements with one or more bounding loops of vertices, along with a *target* surface with bounding loop(s), and a linear transformation was established between two surfaces. In order to avoid a singular transformation matrix, a set of point vectors on the bounding loops was redefined: $x_s^{i'} = x_s^i - (2x_s^c - x_t^c)$ (new positions of source nodes are redefined as their physical positions - two times source affine center + target affine center); $x_t^{i'} = x_t^i - x_s^c$ (new positions of target nodes are redefined as their physical positions - source affine center). Linear transformations for successive loops were computed using an *advancing front* method based on consecutive boundary loops derived from the *linking* surfaces. After locating interior nodes and connecting points with the same quadrilateral mesh connectivity as the *source* surface, this layer was smoothed independently of connections to nodes on the neighboring layers. However, this approach fails for moderately concave or multiply-connected *source*/*target* surfaces, and often does not produce smooth transitions between highly-curved *source*/*target* surfaces.

X. Roca et al. [12] used the *least-squares* approximation of an affine mapping for projecting a *source* surface mesh onto its *target* surface. The mapping was defined between the parametric spaces on the *source* and *target* surface, using only boundary nodes. In order to avoid the skewn and flattening effects when locating interior nodes on the *target* surface, several functions with the *least-squares* forms [13] were introduced to perform the *least-squares* approximation. However, this approach still suffers from poor mesh quality on the *target* surface which is concave and/or multiply-connected. In addition, this method could not be used for the *source*/*target* surfaces with no parameterization, which sometimes arises when meshing a discrete (i.e. facet-based) geometry.

The *BoundaryError* method [14, 15] was introduced to place nodes using a linear affine algorithm and a subsequent residual error correction. In order to successfully capture curvatures of *source* and *target* surface, the *BoundaryError* method calculated the residual error twice, which was then interpolated for final interior node location. This method is useful for locating interior nodes inside a volume between the *source* and *target* surface but not suitable for mapping a *source* surface mesh onto its *target* surface.

M. L. Staten et al. [11] developed an algorithm called *BMSweep* to place interior nodes on the *target* surface and those between the *source* and *target* surface. The background mesh, which was generated by tessellating boundary nodes on the *source* surface in the parametric space, was needed to provide a framework for computing interior nodes locations on each layer. However, if there is a volume with distorted holes and the same background mesh connectivity is used for all the layers during sweeping, inverted elements will be introduced in the background mesh.

M. L. Staten and S. J. Owen et al. [16] also described six mesh morphing techniques for 3D shape optimization: smoothing, weighted residuals (*BoundaryError*), simplex-linear transformation (*BMSweep*), simplex-natural

neighbor transformation, finite element and *Log Barrier* method. The simplex-linear transformation and finite element Warping were recommended for a mesh morphing system. However, simplex-linear transformation based on *BMSweep* suffers from the same drawbacks as *BMSweep*. The finite element-based mesh warping algorithm expresses coordinates of each interior node as an affine combination of its neighbors with shape functions encapsulated in the element stiffness matrix for each element. It is very expensive to solve the stress equilibrium equations.

S. M. Shontz et al. [17] presented a mesh warping algorithm for tetrahedral meshes based on weighted *Laplacian* smoothing. A set of local weights for each interior node, which described relative distances of a node to its neighbors, was determined. After deforming boundaries, a system of linear equations based on weights was solved to determine final locations of interior nodes. However, it is an extension of smoothing and only works for morphing surfaces with smaller deformation.

R. Vurputoor et al. [18] proposed a mesh morphing technique for geometrically dissimilar tessellated surfaces. A topologically conforming background template mesh on the *target* surface was created by using the same triangle mesh connectivity as the *source* surface. Hence, those background meshes were used to map interior nodes between the *source* and *target* surface. However, due to the same triangle mesh connectivity used on the *source* and *target* surface, inverted elements may be introduced in the background mesh on the *target* surface if there is a surface with twisted holes and constant outmost boundary.

I. Sigal et al. [19] presented two morphing algorithms, namely, automated wrapping and manual landmarks, and applied them to prepare specimen-specific models of caudal rat vertebrae. The basic idea of automated wrapping was to find mappings from the *source* and *target* surfaces to an auxiliary surface instead of finding a mapping between the complicate *source* and *target* surfaces. However, this kind of mesh morphing works very well only for closed surfaces and volume morphing, while the *source* and *target* surfaces are generally open during sweeping.

One problem in morphing from one shape to another is to establish the correspondence map [20]. Fortunately, many scholars have solved this problem from their perspectives. T. Kanai et al. [21] used *harmonic maps* for morphing triangle meshes with any arbitrary topology. The basic idea was to define reference shapes by using vertex-to-vertex correspondences between two meshes. The partition of a mesh was defined by the reference shape and partitioned meshes were embedded into a polygonal region in the plane through *harmonic maps*. By overlapping two embedded meshes, the correspondence was established between them. A. Lee et al. [20] presented a method for user-controlled morphing of two homeomorphic triangle meshes of any arbitrary topology. The *MAPS* algorithm (Multiresolution Adaptive Parameterization of Surfaces [22]) was employed to parameterize both meshes over simple base

domains and an additional *harmonic mapping* brought the latter into correspondence. Feature pairs of points were required to be specified by users. Z.W. Fan et al. [23] applied the polycube-based cross-parameterization on mesh morphing. Takashi et al. [24] proposed a multiresolution-based shape representation for 3D mesh morphing. Two types of subdivision fitting scheme were used to calculate the interpolation mesh.

In order to avoid problems from the above methods, *morphing* techniques are used to project a *source* mesh onto the *target* surface in this paper. *Morphing* techniques aim at transforming a given *source* shape into a *target* shape [25]. Afterwards, an all-quad mesh on a given *source* shape can be embedded into its *target* shape by using *barycentric coordinates* efficiently. Therefore, based on the concept of morphing, we can map the *source* surface and *target* surface onto a common domain (usually convex polygon such as 2D unit disk). In this paper, we use *harmonic function* to map the *source* surface and *target* surface onto a 2D unit disk. *Harmonic mapping* has many merits which are valuable for surface mapping [26, 27]: (a) compute through the global optimization; (b) it is diffeomorphism; (c) it is determined by the metric, not the embedding.

In addition, one-to-one sweeping is easier than a general morphing problem. First, it does not require users interaction to specify feature pairs of vertices because the *source* and *target* surfaces are connected by the *linking* surfaces. All the nodes on the boundaries of *source* surface have their corresponding locations on those of *target* surface when morphing from a *source* surface to its *target* surface. Moreover, there is no complicated topology transformation because a *source* surface must have the same topology as its *target* surface in one-to-one sweeping.

3 Harmonic Mapping

In this paper, an application of harmonic function on projecting a source mesh onto its target surface is proposed. In graphics, surfaces are represented with graphics triangle meshes. During one-to-one sweeping, the general approach is to map the source and target graphics triangulations first to separate unit disks, then associate them together. Afterwards, an all-quad mesh from the source surface can be located in its graphics mesh. Because the unit disks from the source and target surface are associated together, any node from quadrilateral meshes on the source surface can be located in the graphics mesh of target surface as well. Therefore, an all-quad mesh on the source surface can be mapped back to the target surface.

In Fig.1, *harmonic mapping* algorithm develops M_1 and M_2 to 2D unit disks, which we call $H1$ and H_2, respectively. If M_1 and M_2 have a graphical triangular mesh respectively, the same applied for H_1 and H_2. H_1 and H_2 have the same mesh connectivity as M_1 and M_2, respectively. For the *source* surface M_1 and *target* surface $M2$, H_1 and H_2 are created by mapping M_1

and M_2 onto 2D unit disk by using the *harmonic mapping*, respectively. In order to establish the correspondence between H_1 and H_2, a new common 2D unit disk H_c (which is replaced by H_2 later) needs to be created by adjusting nodes location on boundaries and combining both H_1 and H_2. Recall that H_c has both M_1 and M_2s connectivity. For the sake of simplicity, we keep 2D unit disk H_2 fixed. Without creating a new H_c and mapping from H_1 to H_c, the boundary nodes on H_1 are adjusted in order to make the boundary nodes between H_1 and H_2 correspond. After the boundary correspondence between H_1 and H_2 is made, 2D unit disk H_1 is mapped onto H_2 directly. Then the correspondence between M_1 and M_2 is established. Afterwards, an all-quad mesh on the *source* surface can be mapped back to the *target* surface. In order to guarantee that the *harmonic mapping* is one-to-one and well defined for geometries with large aspect ratios, the graphical triangular mesh may be smoothed on the geometry if possible before *harmonic mapping*.

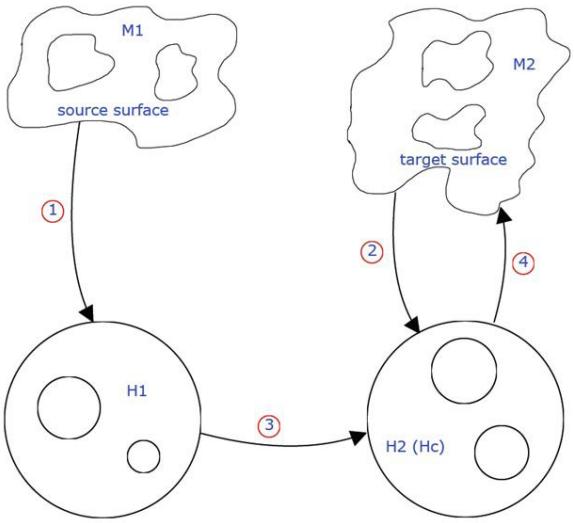

Fig. 1 Road map for mapping an all-quad mesh from a *source* surface to its *target* surface. (1) M_1 is mapped onto H_1. (2) M_2 is mapped onto H_2. (3) The boundary nodes of H_1 are adjusted and correspondence between H_1 and H_2 is made. H_1 is mapped onto H_2. (4) Quadrilateral meshes on the *source* surface are mapped back onto M_2.

3.1 Harmonic Mapping

Harmonic Map φ, $M \rightarrow H$ is mapping between two Riemannian manifolds. M and H are harmonic if the *Dirichlet* energy is minimized. *Harmonic Map* performs a mapping from a topological disk to a 2D unit disk. To construct

the *source* and *target* surfaces embeddings, the piecewise linear approximation method for mapping from M to H is used [21,28], which is established as follows: n vertices on the outmost boundary are distributed on the boundary of 2D disk. This could be based on the edge length between two adjacent boundary nodes. After vertices on the outmost boundary of *source* surface have been distributed, vertices on the outmost boundary of *target* surface should be fixed as well because they are corresponded with those of *source* surface and located through the *linking* surfaces. For a multiply-connected surface, the mapping remains one-to-one even when considering holes in the domain [29]. Based on the Ref. [30], if there is a genus zero surface S with multiple boundaries, and a Riemannian metric g, then there exists a conformal map $f : S \to D$, where D is a 2D unit disk with circular holes.

3.1.1 Discrete Harmonic 1-Form

Let $[v_i, v_j]$ be an interior edge on the triangular mesh, connecting two faces $[v_i, v_j, v_k]$ and $[v_i, v_j, v_l]$, the corner angle in $[v_i, v_j, v_k]$ against $[v_i, v_j]$ is θ_k^{ij}, the corner angle in $[v_i, v_j, v_l]$ against $[v_i, v_j]$ is θ_l^{ij}, the edge weight is defined as

$$\omega_{ij} = cot\theta_k^{ij} + cot\theta_l^{ij} \tag{1}$$

The discrete harmonic energy is defined as

$$E(f) = \sum_{[v_i,v_j]} \omega_{ij}(f(v_i) - f(v_j))^2 \tag{2}$$

The discrete harmonic function is the critical point of the harmonic energy, which satisfies the following discrete harmonic 1-form.

$$\delta f(v_i) = \sum_{[v_i,v_j]\in E} \omega_{ij}(f(v_i) - f(v_j)) = 0, \ \forall v_i \in V \tag{3}$$

Where V is a set of all the vertices.

3.1.2 Multiply Connected Domains

If a function is harmonic (that means it satisfies Laplace' equation over a particular space) and transformed via a conformal map to another space, the transformation is also harmonic. If there is a genus zero surface with multiple holes, the generalized *Koebe*'s method could be applied to compute the canonical conformal mappings [30] where the *harmonic mapping* is a particular *conformal mapping*. The conformal mapping of a multiply connected domain is equivalent to compute the conformal mapping of a topological annulus, which is reduced to compute a pair of conjugated harmonic 1-forms.

Suppose there is a surface S with n holes γ_i, $i = 1, \cdots, n$ and boundaries D_i, $i = 1, \cdots, n$. The outmost boundary which bounds the surface S can be

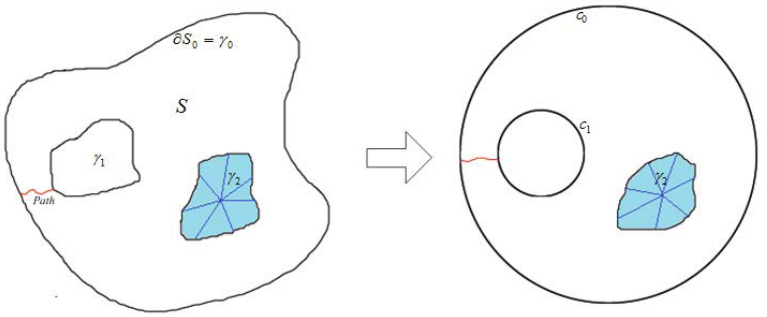

Fig. 2 An example of conformal mapping of S to the canonical annulus ($\gamma_0 \to c_0$, $\gamma_1 \to c_1$)

denoted as ∂S_0. The conformal mapping could be computed in the following steps [30].

(1) Remove a hole D_k from a surface S by computing a path which connects D_k and ∂S_0. Then the boundary of a hole becomes a part of ∂S_0. All other holes are filled by connecting the hole mass center with the boundary vertices on each individual hole. An example is shown in Fig. 2.
(2) Conformally map the surface S (annulus) to the canonical annulus, such that the boundary (D_k) of γ_k is mapped to a circle c_k.

$$\varphi_k : S - D_k \to \textit{unit disk}$$

such that $\varphi_k(\gamma_k) = c_k$
(3) Compute a harmonic mapping of D_k, with the boundary condition that the boundary of D_k is mapped to c_k.

$$f_k : D_k \to \textit{a disk inside 2D unit disk}, \Delta f_k = 0, f_k|_{\gamma_k} = c_k$$

(4) Update the whole mesh S

$$S \leftarrow \varphi_k(S - D_k) \cup f_k(D_k)$$

(5) Process the remaining holes $D_i (i = 1, \cdots, n)$ individually using the above steps. Then the boundary of each disk is mapped to a circular curve, compute the center and radii as (c_k,r_k).
(6) Repeat step 5 until it converges.

$$\sum_{k=1}^{3} |c_k^0 - c_k^1|^2 + |r_k^0 - r_k^1|^2 < \epsilon$$

where (c_k^0,r_k^0) and (c_k^1,r_k^1) are the center and radius of D_i of two consecutive iterations.

3.2 Establish Surface Correspondence

In the Sec. 3.1, an embedding H_1 or H_2 has already been created from M_1 or M_2. In this section, two embeddings are associated together (H_2 is fixed and kept constant), which has the combined mesh connectivity from both *source* surface M_1 and *target* surface M_2. After establishing the correspondence between H_1 and H_2, any point from the *source* surface M_1 corresponds with a position on the *target* surface M_2.

An all-quad mesh generation on the *target* surface mapped from a *source* surface mesh consists of four steps.

(1) Based on Sect. 3.1.2, map the *target* surface M_2 onto a 2D unit disk H_2.
(2) Perform a mapping from the *source* surface M_1 onto 2D unit disk H_1 and H_2: first, adjust vertices on the boundaries in H_1 so that those on H_1 are moved to their corresponding positions on H_2 (corresponding positions can be obtained from the *linking* sides between M_1 and M_2). Recall that M_1 and M_2 may have different triangle mesh nodes and connectivity, it is not necessary to make all the vertices on the outmost boundary overlap. However, they should be corresponded. In other words, those vertices on the boundaries of H_1 are fixed once H_2 is fixed because the *linking* surfaces guide the correspondence between M_1 and M_2. Then the *harmonic mapping* is performed and M_1 and M_2 are mapped onto H_2.
(3) Compute the *barycentric coordinates* for every mesh node from the quadrilateral meshes on the source surface through the graphical triangular mesh M_1. Then calculate 2D positions of every mesh node from the quadrilateral meshes of *source* surface M_1 at H_1 by using the same barycentric coordinates. In this step, every node from the quadrilateral meshes of source surface on the graphical triangle mesh M_1 could be placed on H_1. Because there is an all-quad mesh on the *source* surface M_1, a triangular face is searched at H_1 on the *source* surface where each mesh node n_m^1 is located (See Fig. 3(a)). When n_m^1 is located in a face $\{v_i^1, v_j^1, v_k^1\}$ of H_1, the barycentric coordinates (i_1^1, i_2^1, i_3^1) can be computed as follows.

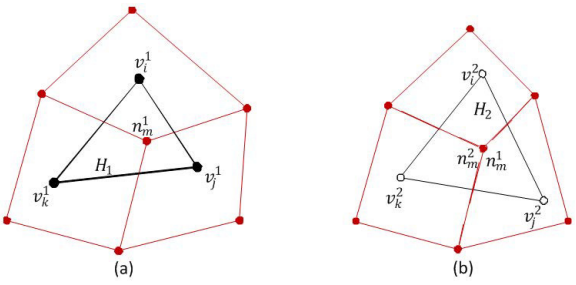

Fig. 3 Mapping vertex n_m^1 in H_1 to H_2

$$n_m^1 = i_1^1 v_i^1 + i_2^1 v_j^1 + i_3^1 v_k^1 \tag{4}$$

$$i_1^1 + i_2^1 + i_3^1 = 1 \tag{5}$$

Where n_m^1 is a mesh node on the *source* surface and v_i^1,v_j^1 and v_k^1 are three vertices of a triangle from the graphical triangulations.

(4) Calculate corresponding 3D positions on the *target* surface M_2 of each mesh node n_m^2 at H_2. Search a triangular face at H_2 where a node n_m^1 in H_1 is included in order to compute 3D locations of a node n_m^2 on the *target* surface M_2 (see Fig. 3(b)). When n_m^1 is located in a triangular face $\{v_i^2,v_j^2,v_k^2\}$ at H_2, the barycentric coordinates (i_1^2,i_2^2,i_3^2) could be computed as follows.

$$n_m^2 = i_1^2 v_i^2 + i_2^2 v_j^2 + i_3^2 v_k^2 \tag{6}$$

$$i_1^2 + i_2^2 + i_3^2 = 1 \tag{7}$$

Where n_m^2 is a mesh node on the *target* surface. 3D position on the *target* surface for a mesh node n_m^2 is computed based on the barycentric coordinate on the face (i_1^2, i_2^2, i_3^2).

After the above four steps, an all-quad mesh on the *target* surface can be created. An example of mapping between them is shown in Fig. 4.

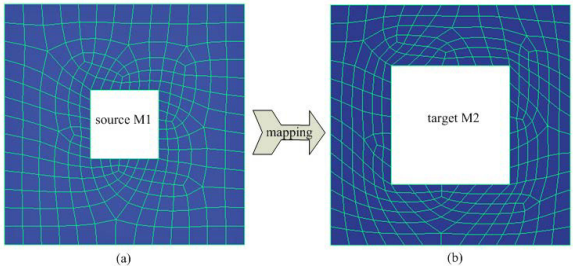

Fig. 4 An all-quadrilateral mesh mapping from a *source* surface M_1 to its *target* surface M_2

4 Interior Nodes'Placement Based on Cage-Based Method

In one-to-one sweeping, *linking* sides connect the *source* and *target* surface. They usually have four logical sides or corners. However, they may consist of one or several geometric curves. Therefore, *transfinite interpolation* (*TFI*) can be used to generate the structured all-quadrilateral meshes on the *linking* sides. After all the surfaces are meshed, cage-based method can be used to place interior nodes.

4.1 Introduction

The cage-based deformation allows an arbitrary closed mesh to act as a deformation cage around another mesh. Figure 5 is an example of deformed object (gray color) inside a deformed mesh cage (black wireframe). There is only one requirement for a cage that the deformed mesh cage could be any shape of mesh but it must be closed. In Fig. 5, the cage has been altered using proportional editing, as a result the sphere alters its shape in response. The object (gray color) is bound with its cage mesh. When the cage mesh is deformed, the object is told to use the deformed cage to deform itself. Basically,

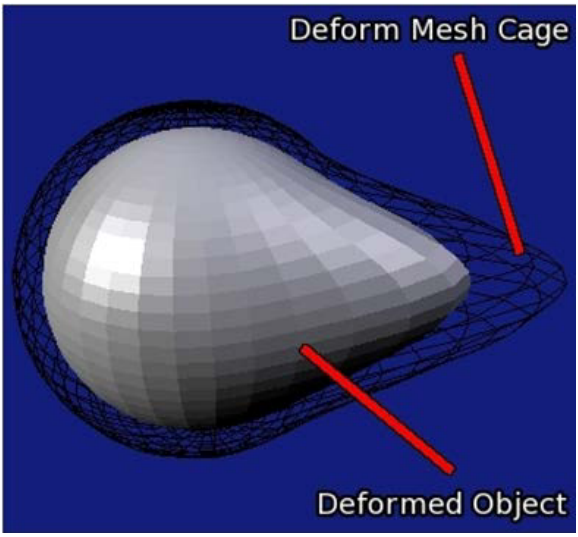

Fig. 5 A 3D example of deformed sphere when its enclosing cage is deformed

there are four steps for a cage-based deformation: (1) automatically or manually create a cage to enclose an object to be deformed; (2) bind an object with its cage (cage vertices). In this step, the geometry info of an enclosed object is bound with its cage vertices. That means every vertex on an object is a function of its cage vertices. If any cage vertex is altered, the object will deform itself by using its deformed cage. (3) deform a cage in order to deform an object; (4) interpolate new object in response to the deformed cage. In the above four steps, step (2) is the most difficult one. Current approaches for step (2) include *Mean Value Coordinates*, *Harmonic Coordinates* and *Green Coordinates*. Current cage methods express a point η inside a cage P as an affine sum of its cage vertices $V = \{v_i\}_{i \in I_V} \subset R^3$. Let i be the cage vertex index, v_i be 3D location of a cage vertex i and I_V be a set of cage vertices, then we have

$$\eta = F(\eta; P) = \sum_{i \in I_V} \phi_i(\eta) v_i \tag{8}$$

Where $\phi_i(\eta)$ is the weight for representing the deformation influence and often referred as "coordinates". Then the deformation defined by a deformed cage P' is defined by

$$\eta = F(\eta; P') = \sum_{i \in I_V} \phi_i(\eta) v_i' \tag{9}$$

4.2 Framework for Locating Interior Nodes

During one-to-one sweeping, the cage-based deformation can be applied to place interior nodes inside volumes because interior nodes are enclosed by their bounding surfaces (*source*, *target* and *linking* surfaces). An example is shown in Fig. 6 with all the bounding surfaces as a deformed cage. Our methods proceed using the following steps:

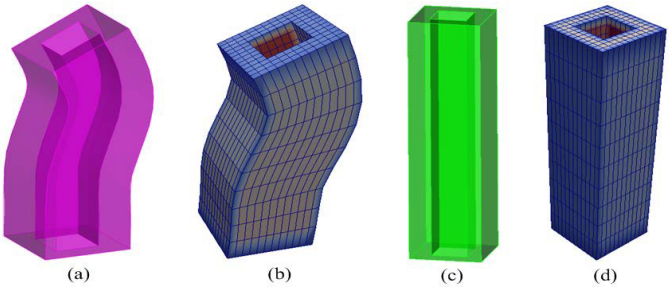

Fig. 6 The physical and topological model for locating interior nodes inside a volume: (a)a physical model; (b)meshed bounding surfaces for(a); (c)a topological model from (a); (d)hexahedral meshes generated for (c).

(1) **Create the Topological Model:** start from a *target* surface and translate its quadrilateral meshes in the inverse sweeping direction until its *source* surface is reached.
(2) **Mesh Conversion:** convert quadrilateral meshes on the bounding surfaces into triangular meshes both in the physical model and topological model. All these triangular meshes (from the physical model and topological model) should have the same mesh connectivity in both physical model and topological model and they correspond each other.
(3) **Binding:** in the topological model, positions of interior nodes have already been placed by simple translation (see Fig. 6(d)). Because all the vertices on the bounding surfaces in the topological model have already been located as well, the cage-based deformation techniques could be

used to bind interior nodes with respect to their cage vertices on the bounding surfaces in the topological model. That means: every interior node location is a function of its cage vertices on the bounding surfaces.

(4) **Interpolation:** the triangle meshes converted from quad meshes on the bounding surfaces in physical model are used as a deformed cage. Because the binding process is done in step 3, Eqn. 8 with a deformed cage as inputs could be used to interpolate final location of interior nodes inside the deformed cage.

4.3 Harmonic Coordinates

The main problem to solve the desired cage-object relationship (binding process in Sect. 4.2) starts with a theoretical problem. Let a cage C be a polyhedron in d dimensions (it is a closed planar polygon in 2D and a closed region bound by planar faces in 3D). For each cage vertex C_i, a function $h_i(p)$ (*harmonic coordinates*) defined on the cage C should satisfy the following conditions (p is an interior node inside its cage) [31]: (1) interpolation $h_i(C_j)=\delta_(i,j)$; (2)$h_i(p)$ should have at least C^1 smooth inside a cage; (3) non-negativity $h_i(p) \geq 0$, for all the interior points $p \in C$; (4)interior locality: interior locality holds, if, in addition to non-negativity, the coordinate functions have no interior extrema; (5) linear reproduction Given an arbitrary function $f(p)$, it should satisfy $H[f](p)=\sum_i h_i(p)f(C_i)$; (6)affine invariance $\sum_i h_i(p)=1$ for all the interior points $p \in C$; (7) strict generalization of barycentric coordinates $h_i(p)$ is a barycentric coordinate of p with respect to a cage vertex C_i. The coordinate functions satisfying all seven properties could be solutions to the *Laplace*s equation.

$$\bigtriangledown^2 h_i(p) = 0,\ p \in Interior(C) \tag{10}$$

Let ∂p denotes a point on the boundary of ∂C of C. Then

$$h_i(\partial p) = \phi_i(\partial p) \quad for\ all\ \partial p \in \partial C \tag{11}$$

Where $\phi_i(\partial p)$ is the (univariate) piecewise linear function such that $\phi_i(C_j) = \delta_{i,j}$.

The approximation of harmonic functions by piecewise linear functions over triangulations on the bounding surfaces (used as the cage), in such a way that the injective property is preserved. *Harmonic Coordinates* could be computed as follows [31]:

(1) Allocate a regular grid which is large enough to enclose the whole cage. Each grid cell contains a value and a tag. A tag could be one of *UNTYPED*, *BOUNDARY*, *INTERIOR*, or *EXTERIOR*.
(2) Initialize the grid by marking all the cells as *UNTYPED*.
(3) Scan-convert boundary conditions into the grid, marking each scan converted cell with the *BOUNDARY* tag. In 3D, it is restricted to triangular

faces, meaning that the boundary values varying in a piecewise linear fashion. The *BOUNDARY* cells could be marked with harmonic coordinate value equal to 1.

(4) Start with one of corner cells, flood fill the exterior, mark each visited cells with *EXTERIOR* tag. It stops when *BOUNDARY* cells are reached. During this step, only the exterior cells are visited in that bounding surfaces are closed.
(5) Mark the remaining *UNTYPED* cells as *INTERIOR* with harmonic coordinate value equal to 0.
(6) *Laplacian* Smoothing: for each *INTERIOR* cell, replace cells value with an average of its neighbors.

5 Results

All the development and testing were done on Ubuntu 13.04, running on Intel Core-2 processors with 4GB RAM. In order to assess mesh quality of new sweeping algorithm based on *harmonic mapping*, several examples are provided. The new sweeping algorithm works for the geometry: cap surfaces with different shapes and curvature, but with the same topology. For the geometry with non-constant cross section along the sweeping direction, nonlinear sweeping trajectories, non-parallel cap surfaces or non-simply connected cap surfaces, it works as well. The new sweeping algorithm based on *harmonic mapping* can avoid inverted elements and produce good-quality meshes when an all-quadrilateral mesh is mapped onto the *target* surface. After all the bounding surfaces are meshed, the cage-based deformation method is used to place interior nodes inside volumes in order to deal with the complicated internal structures inside volumes. The results show the hexahedral meshes with good quality are produced as well.

The first example in Fig. 7 is a blocky volume where there is an increasing hole between the *source* and *target* surface. The interior boundary is randomly curved. A simple linear affine transformation will produce 170 inverted elements when a *source* mesh is swept towards the *target* surface. After the surface mesh projection based on *harmonic mapping* is used, inverted elements could be eliminated (see Fig. 7(d) for details). However, *Cubit13.2* generates some inverted elements shown in Fig. 7(c) when an all-quad mesh on the *source* surface is swept towards the *target* surface. For comparison, the mesh quality histograms from our method and *Cubit13.2* are plotted in Fig. 7(e) and Fig. 7(f). The result shows that an all-hex mesh produced by our method has better mesh quality.

Figure 8 shows a blocky volume with a concave feature and an increasing twisted hole between the *source* and *target* surface. An all-quad mesh on the *source* surface is shown in Fig. 8(b). After our method is applied, an all-quad mesh on the *target* surface without any inverted element is produced in Fig. 8(d). For comparison, this example is run on *Cubit13.2* and mesh quality

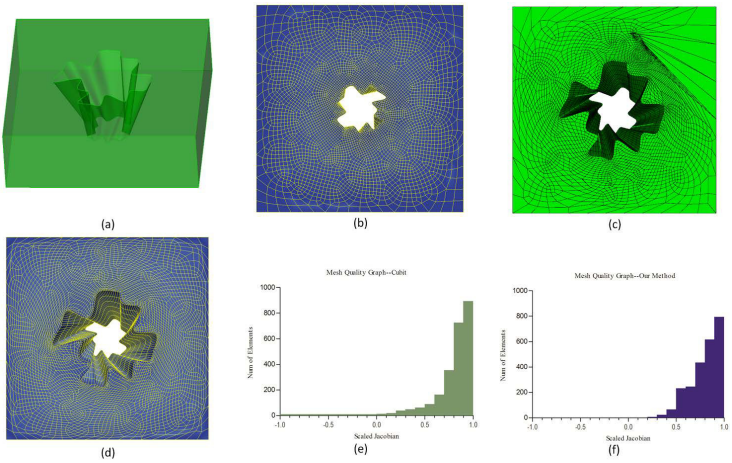

Fig. 7 Swept volume with a varying hole of random boundary. (a) a geometry model; (b) quadrilateral mesh on the *source* surface; (c) quadrilateral on the *target* surface generated by *Cubit*; (d) quadrilateral mesh on the *target* surface generated by our method; (e) a quality histogram from *Cubit*; (f) a quality histogram from our method.

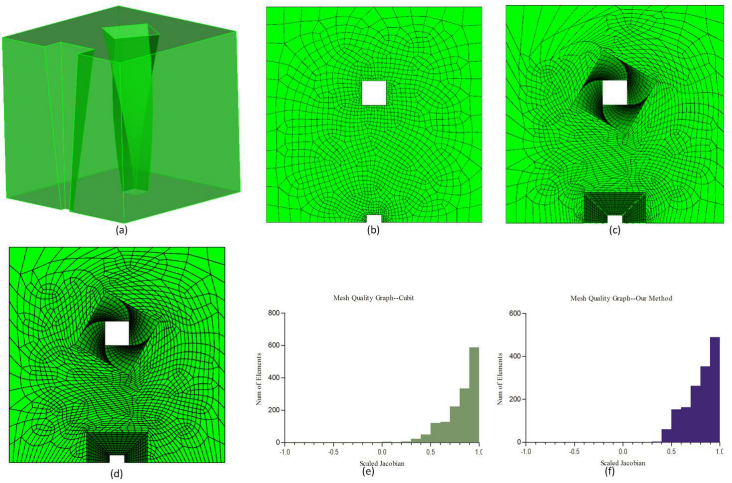

Fig. 8 Volumes with concavities and increasing twisted hole. (a) a geometry model; (b) an all-quad mesh on the *source* surface; (c) an all-quad mesh generated by *Cubit 13.2*; (d) an all-quad mesh on the *target* surface generated by our method; (e) a quality histogram from *Cubit13.2*; (f) a quality histogram from our method.

histograms from our method and *Cubit13.2* are plotted in Fig. 8(f) and Fig. 8(e). The results show that our method can produce all-hexahedral meshes without any inverted element and with better mesh quality.

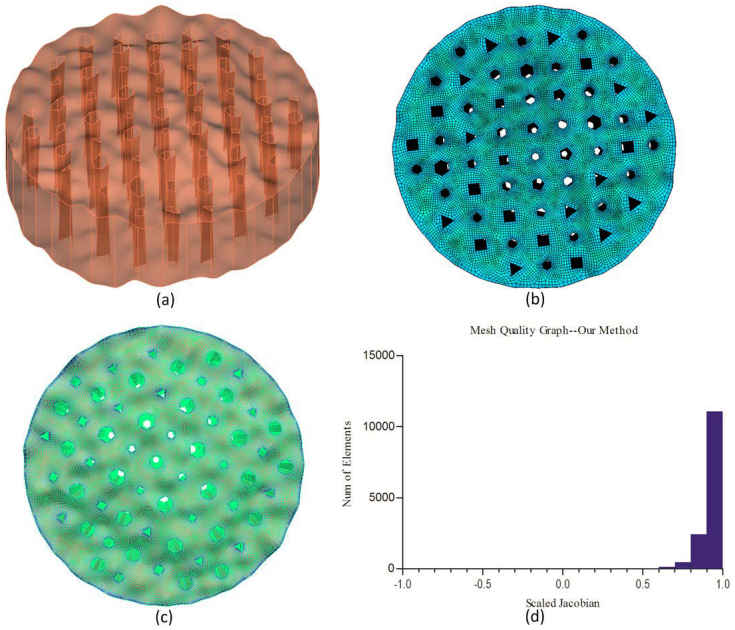

Fig. 9 A spongecake meshed with sweeping. (a) a geometry model; (b) quadrilateral mesh on the *source* surface; (c) quadrilateral mesh on the *target* surface generated by our method; (d) a quality histogram.

The third example in Fig. 9 is a spongecake which is similar to cylinder with a lot of varying holes inside the volume (increasing hole sizes and decreasing hole sizes between the *source* and *target* surface). The cap surfaces are curved. Simple linear affine transformation would create 709 inverted elements inside holes. After surface mesh projection based on *harmonic mapping* is used, no inverted elements are created. The final all-hexahedral meshes are shown in Fig. 9(c). The mesh quality histogram in Fig. 9(d) shows that an all-hex mesh produced by our method has better mesh quality.

6 Conclusion

In this paper, a new algorithm to project an all-quad mesh on *source* surface to *target* surface based on *harmonic mapping* has been presented. It has been successfully implemented in *MeshKit*[35]. The projection between two topologically equivalent surfaces is determined by making the correspondence between them based on *harmonic mapping*. First generate 2D unit disk based on harmonic mapping. Then make the correspondence between two unit disks. Finally, an all-quad mesh on the *source* surface is mapped back to the *target* surface. In order to locate interior nodes between the *source* and *target*

surfaces, the cage-based method is used to produce good-quality hexahedral meshes compared to *Cubit13.2*.

Acknowledgement. This work was funded under the auspices of the Nuclear Energy Advanced Modeling and Simulation (NEAMS) program of the Office of Nuclear Energy, and the Scientific Discovery through Advanced Computing (SciDAC) program funded by the Office of Science, Advanced Scientific Computing Research, both for the U.S. Department of Energy, under Contract DE-AC02-06CH11357. We also thank all the Fathom members (from both Univ. of Wisconsin-Madison and Argonne National Lab) for their efforts on the CGM [32], MOAB [33], Lasso [34] and MeshKit [35] libraries, which were used heavily to support this work.

References

1. Biswas, R., Strawn, R.C.: Tetrahedral and hexahedral mesh adaptation for CFD problems. Applied Numerical Mathematics 26(1-2), 135–151 (1988)
2. Samareh, J.A.: Geometry and grid/mesh generation issues for CFD and CSM shape optimization. Optimization and Engineering 6(1), 21–32 (2005)
3. Owen, S.J.: A Survey of Unstructured Mesh Generation Technology. In: 7th IMR, pp. 239–267 (1998)
4. Taniguchi, T., Goda, T., Kasper, H., et al.: Hexahedral Mesh Generation of Complex Composite Domain. In: 5th International Conference on Grid Generation in Computational Field Simmulations, pp. 699–707 (1996)
5. Schneiders, R.: A Grid-based Algorithm for the Generation of Hexahedral Element Meshes. Engineering with Computers 12(3-4), 168–177 (1996)
6. Price, M.A., Armstrong, C.G.: Hexahedral Mesh Generation by Medial Surface Subdivision: Part I. IJNME 38(19), 3335–3359 (1995)
7. Price, M.A., Armstrong, C.G.: Hexahedral Mesh Generation by Medial Surface Subdivision: Part II. IJNME 40, 111–136 (1997)
8. Blacker, T.D., Myers, R.J.: Seams and Wedgers in Plastering: A 3D Hexahedral Mesh Generation Algorithm. Engineering With Computers 2, 83–93 (1993)
9. Tautges, T.J., Blacker, T.D., Mitchell, S.A.: The Whisker-Weaving Algorithm: A Connectivity Based Method for Constructing All-Hexahedral Finite Element Meshes. IJNME 39, 3327–3349 (1996)
10. Patric, M.K.: Next-Generation Sweep Tool: A Method for Generating All-Hex Meshes on Two-And-One-Half Dimensional Geometries. In: 7th IMR, pp. 505–513 (1998)
11. Staten, M.L., Canann, S.A., Owen, S.J.: BMSweep: Locating Interior Nodes During Sweeping. In: 7th IMR, pp. 7–18 (1998)
12. Roca, X., Sarrate, J., Huerta, A.: Surface Mesh Projection for Hexahedral Mesh Generation by Sweeping. In: 13th IMR, pp. 169–180 (2004)
13. Roca, X., Sarrate, J., Huerta, A.: A new least-squares approximation of affine mappings for sweep algorithms. In: 14th IMR, pp. 433–448 (2005)
14. White, D.R., Lai, M.W., et al.: Automated Hexahedral Mesh Generation by Virtual Decomposition. In: 4th IMR, pp. 165–176 (1995)
15. Scott, M.A., Earp, M.N., Benzley, S.E., et al.: Adaptive Sweeping Techniques. In: 14th IMR, pp. 417–432 (2005)

16. Staten, M.L., Owen, S.J., Shontz, S.M., Salinger, A.G., Coffey, T.S.: A comparison of mesh morphing methods for 3*D* shape optimization. In: Quadros, W.R. (ed.) Proceedings of the 20th International Meshing Roundtable, vol. 90, pp. 293–311. Springer, Heidelberg (2011)
17. Shontz, S.M., Vavasis, S.A.: A mesh warping algorithm based on weighted Laplacian smoothing. In: 12th IMR, pp. 147–158 (2003)
18. Vurputoor, R.M., et al.: A Mesh Morphing Technique for Geometrically Dissimilar Tessellated Surfaces. In: 16th IMR, pp. 315–334 (2008)
19. Sigal, I.A., Hardisty, M.R., Whyne, C.M.: Mesh-morphing algorithms for specimen-specific finite element modeling. Journal of Biomechanics 41(7), 1381–1389 (2008)
20. Lee, A., Dobkin, D., Sweldens, W., Schroder, P.: Multiresolution Mesh Morphing. In: Proceedings of SIGGRAPH 1999, pp. 343–350 (1999)
21. Kanai, T., Suzuki, H., Kimura, F.: Three-dimensional geometric metamorphosis based on Harmonic Maps. The Visual Computer 14(4), 166–176 (1998)
22. Lee, A.W.F., Sweldens, W., Schroder, P., et al.: MAPS: Multiresolution Adaptive Parameterization of Surfaces. In: SIGGRAPH 1998 Proceedings, pp. 95–104 (1998)
23. Fan, Z.W., Jin, X.G., Feng, J.Q.: Mesh Morphing using polycube-based cross-parameterization. Computer Animation and Virtual Worlds 16, 499–508 (2005)
24. Kanai, T., Fujita, M., Chiyokura, H.: Multiresolution interpolation meshes. In: 9th Pacific Graphics International Conference, vol. 10, pp. 60–69 (2001)
25. Marc, A.: Recent Advances in Mesh Morphing. CG Forum, 1–23 (2002)
26. Wang, Y., Gupa, M., Gu, X.F., et al.: High Resolution Tracking of non-Rigid 3D Motion of Densely Sampled Data Using Harmonic Maps. In: IEEE International Conference on Computer Vision (2005)
27. Joshi, P., Meyer, M., et al.: Harmonic Coordinates for Character Articulation. ACM Transactions on Graphics 26(3(7)) (2007)
28. Zhang, D., Hebert, M.: Harmonic maps and their applications in surface matching. In: IEEE Conference on Computer Vision and Pattern Recognition (1999)
29. Remacle, J.F., Geuzaine, C., Compere, G., et al.: High Quality Surface Remeshing Using Harmonic Maps. IJNME 83(4), 403–425 (2009)
30. Zeng, W., Yin, X.T., Zhang, M., Luo, F., Gu, X.F.: Generalized Koebe's method for conformal mapping multiply connected domains. In: SIAM/ACM Joint Conference on Geometric and Physical Modeling, pp. 89–100. ACM (2009)
31. Joshi, P., Meyer, M., DeRose, T., et al.: Harmonic coordinates for character articulation. ACM Transactions on Graphics (TOG) 26(3) (2007)
32. CGMA, `https://trac.mcs.anl.gov/projects/ITAPS/wiki/CGM`
33. MOAB, `https://trac.mcs.anl.gov/projects/ITAPS/wiki/MOAB`
34. Lasso, `https://trac.mcs.anl.gov/projects/ITAPS/wiki/Lasso`
35. MeshKit, `https://trac.mcs.anl.gov/projects/fathom/wiki/MeshKit`