A Comparison of Mesh Morphing Methods for 3*D* Shape Optimization*

Matthew L. Staten[1], Steven J. Owen[1], Suzanne M. Shontz[2], Andrew G. Salinger[1], and Todd S. Coffey[1]

[1] Sandia National Laboratories**, Albuquerque, NM, U.S.A.
{mlstate,sjowen,agsalin,tscoffe}@sandia.gov
[2] The Pennsylvania State University, University Park, PA, U.S.A.
shontz@cse.psu.edu

Summary. The ability to automatically morph an existing mesh to conform to geometry modifications is a necessary capability to enable rapid prototyping of design variations. This paper compares six methods for morphing hexahedral and tetrahedral meshes, including the previously published FEMWARP and LBWARP methods as well as four new methods. Element quality and performance results show that different methods are superior on different models. We recommend that designers of applications that use mesh morphing consider both the FEMWARP and a linear simplex based method.

Keywords: mesh morphing, mesh moving, mesh warping.

1 Introduction

The modeling and simulation process often involves many iterations of geometric design changes. Geometric parameters are driven automatically via an optimization procedure. The ability to automatically update an existing mesh to conform to a modified geometry is a necessary capability to enable rapid prototyping of many alternate geometric designs. In literature, this mesh update process is called mesh morphing [28, 21], mesh warping [17, 18], or mesh moving [22]. These algorithms first maintain constant mesh topology, while computing new locations for mesh nodes in order to conform to geometry changes. However, maintaining constant mesh topology has limitations based upon the magnitude of the geometric changes required. More advanced methods for tetrahedral meshes perform local mesh modifications once element quality degrades below a threshold [1, 4, 5, 14]. While

* The work of the third author is supported in part by NSF grant CNS-0720749 and NSF CAREER Award OCI-1054459.

** Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

modifying mesh topology maintains element quality, local modification methods are generally intractable for hexahedral meshes. In addition, mesh topology changes also introduce noise into the gradient computations of the optimization. Extending the range in which the mesh topology may be reused through many iterations of the design process is the desired outcome of this research.

In this paper, we propose four new mesh morphing methods: 1. smoothing alone, 2. weighted residuals, 3. simplex-linear, and 4. simplex-natural neighbor. In addition, two existing methods, FEMWARP [17] and LBWARP [18], which have been previously published for tetrahedral meshes, are extended to hexahedral meshes. We compare these six methods with eight example meshes (four hex, four tet) with numbers of nodes varying from 11k to 136k. We assume the geometric topology of the models remains constant through all iterations of the optimization, which allows the geometric ownership of mesh nodes and elements to remain unchanged.

Ideally, the element quality resulting from any morphing method should be optimal. In practice the element quality from a morphed mesh can still be improved by using the morphed node locations as the initial condition for a post-processing step of optimization based smoothing [3, 11]. The results presented in this paper are generated without post-processing smoothing in order to compare the resulting element quality of the morphing methods.

2 Background

Mesh morphing methods can be categorized as either mesh-based or meshless. Mesh-based methods use the element topology of the mesh being morphed to define a computational space to compute new node locations. Meshless methods ignore the element topology, in favor of other algebraic relationships.

Numerous mesh-based techniques have been developed based on solving partial differential equations (PDEs) assuming the boundary deformation has been defined. For example, spring model approaches have been developed based on solving Laplace's equation, variable diffusion, and biharmonic PDEs [2, 9, 18, 22]. One of these methods, LBWARP [18], is used in the comparison in this paper. A finite element based method (FEMWARP) for triangle and tetrahedral meshes has also been presented [17], and is also part of the comparison in this paper. Several elasticity-based approaches have been developed as well [19, 23, 25]. An optimization-based approach to mesh warping based on the target matrix paradigm recently appeared in [13].

A meshless approach using radial basis interpolation functions has been proposed in [15]. Simplex based meshless approaches have been used for morphing surfaces meshes [24, 28]. In Section 3.3, we extend these simplex methods to morph volume meshes, and compare the results with other methods.

The computer graphics and geometry processing communities use morphing for real-time deformations for animations [10] and geometric mappings between 3D surfaces and the cooresponding 2D parameter space [7].

3 Mesh Morphing Methods

We begin with a geometric domain, $\Omega_G^n = \{G^r | r = 0, 1, 2, 3\}$ at iteration n, with geometry entities of dimension r. An existing finite element mesh, $\Omega_M^n = \{M^r | r = 0, 1, 2, 3\}$, also at iteration n, is assumed to have been generated and fully associated with Ω_G^n. We seek a transformation $\Omega_M^n \Rightarrow \Omega_M^{n+1}$ given a new Ω_G^{n+1} such that the element quality in Ω_M^{n+1} is maximized and usable. We assume the topology of Ω_G^n and Ω_G^{n+1} are identical, which allows the mesh topology of Ω_M^n and Ω_M^{n+1} to also be identical. Therefore, we seek only the nodal ($r = 0$) transformation $\left[M^0\right]^n \Rightarrow \left[M^0\right]^{n+1}$.

In defining the nodal transformation $\left[M^0\right]^n \Rightarrow \left[M^0\right]^{n+1}$ we address the nodes on each geometric entity type independently. For example, nodes are first transformed from vertices ($r = 0$) of $\left[G^0\right]^n$ to vertices of $\left[G^0\right]^{n+1}$, followed by curves ($r = 1$), surfaces ($r = 2$) and finally volumes ($r = 3$). In each step, the locations of nodes associated to lower dimension entities are assumed to be fixed. We introduce the notation $\mathbf{X}_r(e)$ to represent the location of entity e, which is either a geometric vertex or a mesh node. If e is a geometric vertex, r is omitted. If e is a mesh node, r represents the dimension of the geometric entity to which the node is associated.

The new location of node k associated to geometric vertex j is simply:

$$\mathbf{X}_0\left(\left[M_k^0\right]^{n+1}\right) = \mathbf{X}\left(\left[G_j^0\right]^{n+1}\right). \tag{1}$$

For curves, we make the assumption that the parametric location, t_k, on curve j, for any node k, remains constant through the transformation $n \Rightarrow n + 1$, ($t_k = t_k^n = t_k^{n+1}$). We define the location for any interior node k on curve j as:

$$\mathbf{X}_1\left(\left[M_k^0\right]^{n+1}\right) = \left[G_j^1\right]^{n+1}(t_k) \tag{2}$$

where $\left[G_j^1\right]^{n+1}(t_k)$ is a simple parametric evaluation of curve j at t_k.

For nodes associated to surfaces we use either the smoothing or weighted residual method described below. Any of the methods described below could be used to morph surface nodes if the surface is planar, or with a post-morph 3D surface projection. However, our focus is on 3D morphing of volume mesh nodes, so we have only implemented the smoothing and weighted residual methods for general 3D surface node transformations.

For nodes associated to volumes we use either the smoothing, weighted residual, simplex-linear, simplex-natural neighbor, FEMWARP, or LBWARP methods described below. Optionally, for structured hexahedral meshes, the interior node locations of the morphed models could be computed using transfinite interpolation. However, we seek methods which can be applied to unstructured hex mesh topologies.

3.1 Smoothing

Smoothing, a mesh-based method, utilizes smoothing techniques from the Mesquite [3] toolkit, and is used for morphing both surface and volume nodes ($r = 2, 3$). After the nodes on the bounding curves are morphed to their new

positions, the interior surface mesh will often be inverted. Surface smoothing is then performed using the nodal coordinates established on the boundary curves as fixed. Mesquite's mean ratio, condition number [11] and untangling smoothing procedures are used adaptively based on local mesh quality. The same smoothing techniques are subsequently employed for volumes, where the node locations established on the boundary surfaces are fixed.

3.2 Weighted Residuals

Weighted residual, a meshless method, borrows from hexahedral sweeping, which morphs quad meshes from "source" surfaces to intermediate or "target" surfaces [12]. The weighted residual method is used for surface and volume morphing ($r = 2, 3$). The initial node locations, $\left[M^0\right]^n$, are first transformed using an affine transformation, $\mathbf{\Gamma}$, computed to minimize the function:

$$F(\mathbf{\Gamma}) = \min \sum \left[\mathbf{X_{r-1}}\left(\left[M_k^0\right]^{n+1}\right) - \mathbf{\Gamma} \cdot \mathbf{X_{r-1}}\left(\left[M_k^0\right]^n\right)\right] \tag{3}$$

for all nodes, k, on the domain boundary. A correction is then applied based upon the weighted sum of the nearby residual vectors at boundary nodes. The location of an interior node k, can be represented as:

$$\mathbf{X}_r\left(\left[M_k^0\right]^{n+1}\right) = \mathbf{\Gamma} \cdot \mathbf{X}_r\left(\left[M_k^0\right]^n\right) + \sum_{i=1}^{npts} w_i R_i, \tag{4}$$

$$R_i = \mathbf{X}_{r-1}\left(\left[M_i^0\right]^{n+1}\right) - \mathbf{\Gamma} \cdot \mathbf{X}_{r-1}\left(\left[M_i^0\right]^n\right), \text{and} \tag{5}$$

$$w_i = \frac{d_i^{-2}}{\sum_{j=1}^{npts} d_j^{-2}} \tag{6}$$

where R_i in (4) represents the minimization error in (3) for boundary node i. The weight w_i in (4), is the normalized inverse distance squared from node k to the nearest boundary nodes. 0nly a subset of the boundary nodes influence the weight w_i. We identify the *npts* closest boundary nodes using a *kd-tree*. For the results in Section 4 we used $npts = 20$ for surface morphing and $npts = 80$ for volume morphing, however an adaptive method for determining the influencing nodes may be necessary.

3.3 Simplex-Linear Transformations

Simplex-linear is a meshless method based upon the BMSweep hex sweeping method [24], previously used for surface morphing [28]. In this study, simplex-linear was only implemented for 3D volume morphing, using weighted residual for surface morphing.

Simplex-linear creates a Delaunay tessellation, D^n, of the nodes on the fixed domain boundary. We use QMG[27] to generate D^n. No interior nodes are used. The enclosing tetrahedron, T_i^n, of each interior node i is determined. The barycentric coordinates, B_i, of $\mathbf{X}_3\left(\left[M_i^0\right]^n\right)$ with respect to T_i^n is computed. We require that the connectivity of D^n remains the same for

D^{n+1} and enforce the condition that the barycentic coordinates, B_i, for node i with respect to T_i^n and T_i^{n+1} will be the same. The location of interior node i at iteration $n+1$ is then:

$$\mathbf{X}_3\left(\left[M_i^0\right]^{n+1}\right) = \sum_{j=1}^{4} \left(B_i\right)_j \cdot \mathbf{X}_2\left(T_i^{n+1}\right)_j \tag{7}$$

where $\left(B_i\right)_j$ is defined as the j^{th} component of B_i and $\mathbf{X}_2\left(T_i^{n+1}\right)_j$ is the location of the j^{th} node of tetrahedron T_i^{n+1}.

3.4 Simplex-Natural Neighbor Transformations

The simplex-natural neighbor method uses a similar vertex-based weighting scheme as the simplex-linear method described in Section 3.3, but generalizes the selection of weighting vertices. If we consider C_i^n, the set of all tetrahedron, $T_i^n \in D^n$, whose circumsphere contains $\mathbf{X}_3\left(\left[M_i^0\right]^n\right)$, then all vertices, $(N_i^n)_j | j = 1, 2, ...nvert$, of tetrahedron C_i^n, are used to compute the natural neighbor coordinates, $(\lambda_i)_j$ for $[M_i^0]^n$. Similar to equation 3.3 we can define the new location $\mathbf{X}_3\left(\left[M_i^0\right]^{n+1}\right)$ as a weighted average of $(N_i^{n+1})_j$ using weights $(\lambda_i)_j$,

$$\mathbf{X}_3\left(\left[M_i^0\right]^{n+1}\right) = \sum_{j=1}^{nvert} \left(\lambda_i\right)_j \cdot \mathbf{X}_2\left(N_i^{n+1}\right)_j \tag{8}$$

where $(\lambda_i)_j$ are a function of the Voronoi volumes formed by temporarily inserting $[M_i^0]^n$ into D^n as described in [29] and [20]. Because the morphed locations of $[M_i^0]^n$ are influenced by the boundary points within its *natural neighborhood*, rather than only the four points defined by its enclosing tetrahedron, we hypothesize that the resulting morph quality will be improved.

3.5 Finite Elements

Finite element-based mesh warping (FEMWARP) was proposed by Baker [1] and developed by Shontz and Vavasis in [17] for tetrahedral meshes. We now adapt FEMWARP for hexahedral meshes. FEMWARP expresses the coordinates of each interior node, n_i, of the initial mesh as an affine combination of its neighbors with shape functions encapsulated in element stiffness matrices for each element. Extending FEMWARP to hexahedral meshes simply requires implementing hexahedral element stiffness matrices. We use standard bi-linear hex shape functions. Each n_i is then expressed as an affine combination of the nodes which share a common adjacent element with n_i.

The FEMWARP method proceeds as with tetrahedra. Let b and m be the numbers of boundary and interior nodes, $\left[M^0\right]^n$, in Ω_M^n, respectively. We form the $(m+b) \times (m+b)$ global stiffness matrix A by assembling the local element stiffness matrices for the boundary value problem $\triangle u = 0$ on $\left[G^0\right]^{n+1}$ with $u = u_0$ on $\partial\left[G^0\right]^{n+1}$. The result is a symmetric positive definite sparse global stiffness matrix where its nonzero entries correspond to pairs of neighboring nodes.

Next, let A_I denote the $m \times m$ submatrix of A whose rows and columns are indexed by interior nodes, and let A_B denote the $m \times b$ submatrix of A whose rows are indexed by interior nodes and whose columns are indexed by boundary nodes. Let X_I^n be the $m \times 3$ matrix consisting of x, y, z-coordinates of the interior nodes at step n:

$$X_I^n = \{\mathbf{X}_3\left(\left[M_1^0\right]^n\right), \mathbf{X}_3\left(\left[M_2^0\right]^n\right), \ldots, \mathbf{X}_3\left(\left[M_m^0\right]^n\right)\}^T \tag{9}$$

and let X_B^n be the $b \times 3$ matrix consisting of x, y, z-coordinates of the boundary nodes at step n:

$$X_B^n = \{\mathbf{X}_2\left(\left[M_{m+1}^0\right]^n\right), \mathbf{X}_2\left(\left[M_{m+2}^0\right]^n\right), \ldots, \mathbf{X}_2\left(\left[M_{m+b}^0\right]^n\right)\}^T \tag{10}$$

where interior nodes are numbered first. Then it follows that:

$$A_I X_I^n = -A_B X_B^n \tag{11}$$

and

$$A_I X_I^{n+1} = -A_B X_B^{n+1}. \tag{12}$$

In (11) and (12), the only unknown is X_I^{n+1}, which represents the morphed locations of the interior nodes of M^{n+1} and can be found by solving (12) as a linear system of equations.

3.6 Log Barrier

The log barrier-based mesh warping (LBWARP) technique was proposed by Shontz and Vavasis in [18] for tetrahedral meshes. We now adapt LBWARP for hexahedral meshes. Let $\mathbf{X}_3\left(\left[M_i^0\right]^n\right)$ denote the x, y, z coordinates of the i^{th} interior node, n_i, in the initial mesh. In addition, let the x, y, z coordinates of n_i's adjacent nodes be given by $\{\mathbf{X}_{0,1,2,3}\left(\left[M_j^0\right]^n\right) : j \in N_i\}$, where N_i denotes the set of neighbors of n_i.

We define N_i as the set of all nodes which share an adjacent 3D element with n_i. For tetrahedra, this also means that n_i shares a common mesh edge with every n_j in N_i. However, this is not true for hexahedra because of the more complex hexahedral topology. We choose this definition of N_i to include more nodes in the interpolation and for consistency with FEMWARP.

In order to find the set of weights w_{ij}, where w_{ij} is the weight of node j on interior node i, we use the log barrier function from linear programming [16] to formulate the following optimization problem for each n_i:

$$\begin{array}{ll} \max_{w_{ij}, j \in N_i} & \sum_{j \in N_i} \log(w_{ij}) \\ \text{subject to} & w_{ij} > 0 \\ \sum_{j \in N_i} w_{ij} & = 1 \\ \mathbf{X}_3\left(\left[M_i^0\right]^n\right) & = \sum_{j \in N_i} w_{ij} \mathbf{X}_{0,1,2,3}\left(\left[M_j^0\right]^n\right) \end{array} \tag{13}$$

We solve for all w_{ij} using (13), a strictly convex optimization problem, for its unique optimum using the projected Newton method [30]. (See [18] for more details.)

Once the weights, w_{ij}, have been found, we assemble all w_{ij} into an $(m+b) \times (m+b)$ matrix A, and solve for the morphed interior node locations by decomposing it into A_I and A_B in the same manner used for FEMWARP. For LBWARP, A_I is an M-matrix.

4 Examples

We have implemented the six methods described in Section 3 using C++. For the FEMWARP and LBWARP methods, the Trilinos [26] implementation of the Amesos KLU sparse direct linear solver [6] was used to solve the linear systems. No matrix pre-conditioning was done.

Four example problems are illustrated in Figures 1, 3, 5, and 7. 3D morphing, using all six methods, was performed on a tetrahedral and a hexahedral mesh of each model, giving eight total test cases. To ensure a consistent baseline for the 3D methods, the boundary surface meshes of each example were morphed with the weighted residual method. Table 1 lists the number of nodes and elements in each example model. All test case data were run on an HP ProLiant linux workstation with two X5570 Intel quad cores with 24 GB RAM. The resulting quality and execution times were reported.

The tet meshes were generated with the GSH3D tet mesher [8]. The hex meshes on the bore and pipe models are completely structured and were generated with transfinite interpolation. In contrast, the hex meshes on the courier and canister models are unstructured, having been generated by decomposing the geometries into partitions each of which can be meshed with the pave and sweep method [24].

Table 1. Example Model Sizes

	Bore		Pipe		Canister		Courier	
	Hex	Tet	Hex	Tet	Hex	Tet	Hex	Tet
#nodes	11,904	21,859	11,520	17,174	136,462	83,562	101,817	136,106
#elems	15,190	109,535	8,532	81,304	128,269	472,924	83,934	699,867

Element quality is compared using the scaled Jacobian metric, which is the determinant of the elemental stiffness matrix. Normalization of element edges during formulation of stiffness matrices restricts the scaled Jacobian metric on the range [-1,1].

In each example, two sets of geometric parameters, $S_0 \in \{s_1^0, s_2^0, ...s_{nvar}^0\}$ and $S_F \in \{s_1^F, s_2^F, ...s_{nvar}^F\}$, define initial and final geometry configurations. Morphing is performed with N iterations between S_0 and S_F where the parameters, S_i, used for a given iteration i are defined as:

$$S_i = \frac{i}{N}(S_F - S_0), \qquad i \in \{0, 1, \ldots, N\}. \tag{14}$$

We used $N = 20$. Each model is morphed using both *relative* and *absolute* coordinates. With relative coordinates, the mesh is morphed from S_{i-1} to S_i, testing incremental morphing and iterative element quality degradation.

With absolute coordinates, the mesh is morphed from S_0 to S_i, testing how big a step each morphing method can take while maintaining element quality.

4.1 Bore Model

Figure 1 illustrates the "bore" model with its geometric parameters $S_0 \in \{h, r_1, r_2, r_3, \theta\}$ and $S_F \in \{h', r_1', r_2', r_3', \theta'\}$. The values used are $S_0 \in \{12.0, 2.0, 2.5, 3.0, 60°\}$ and $S_F \in \{12.0, 2.5, 3.5, 6.0, 30°\}$. The bore model tests scaling and rotations in each morphing method. The resulting element quality is illustrated in Figure 2. The hex mesh is a structured mesh generated with transfinite interpolation, while the tet mesh is completely unstructured. For this simple model, all morphing methods except smoothing provide near identical results for both relative and absolute morphing. The smoothing method does well for relative morphing, but quickly inverts element quality for absolute morphing. This suggests that smoothing should be restricted to small parametric changes. Table 2 lists the execution times for each method. It is notable that the execution times are roughly the same for the relative and absolute runs for all methods except for smoothing. Smoothing times more than double for absolute. This is because as the parametric changes get bigger, the displacement of the boundary nodes produces an inverted mesh that smoothing must work harder to untangle.

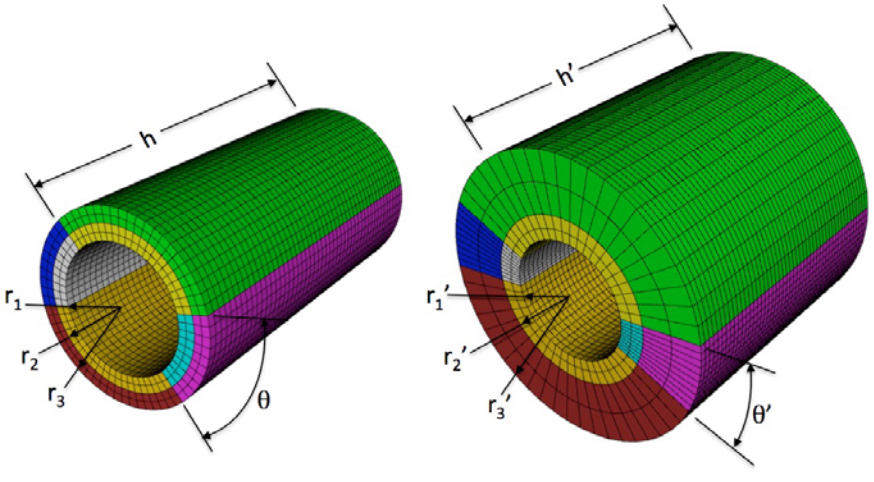


Fig. 1. Example problem "bore". (a) The parametric variables on the initial shape, (b) the final shape.

4.2 Pipe Model

Figure 3a illustrates the "pipe" model with its geometric parameters. Figures 3b and 3c illustrate the initial and ending geometric shapes. The pipe thickness and radius are decreased and the elbow radius is increased. The pipe model tests nonlinear stretching. Figure 4 again shows that the smoothing

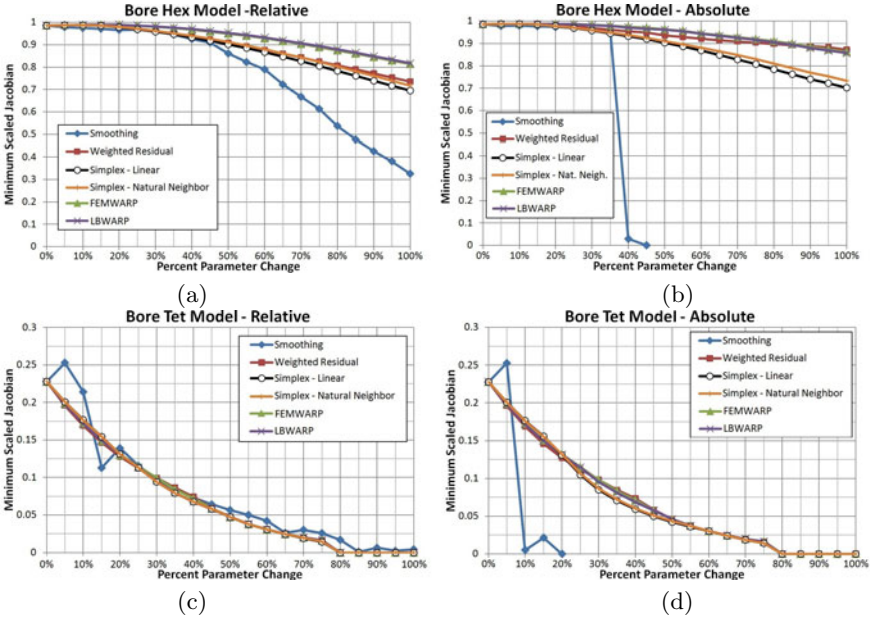


Fig. 2. Results of morphing on bore model. (a) Relative hex morph, (b) absolute hex morph, (c) relative tet morph, (d) absolute hex morph.

Table 2. Results for morphing bore model: CPU Time Per Step

Method	Ave Time Per Step (sec)			
	Hex-Relative	Hex-Absolute	Tet-Relative	Tet-Absolute
Smoothing	0.23	0.75	0.68	1.75
Weighted Residual	3.57	3.65	7.52	7.40
FEMWARP	0.30	0.29	0.56	0.58
LBWARP	1.55	1.36	1.97	1.96
Simplex-linear	0.33	0.40	0.39	0.36
Simplex-natural neighbor	0.46	0.38	0.94	0.56

method is unable to maintain a quality mesh for very long, especially in absolute morphing. For tet morphing the five other methods result in nearly the same element quality. Hex element quality from FEMWARP, simplex-linear, and simple-natural neighbor are roughly the same and superior to weighted residual, while LBWARP is somewhere in the middle.

Execution times for the pipe model are listed in Table 3. For this model, the FEMWARP, simplex-linear, and simplex-natural neighbor methods perform significantly better than LBWARP, smoothing, and weighted residual, while weighted residual is the most expensive method. Simplex-linear is by far the least expensive method.

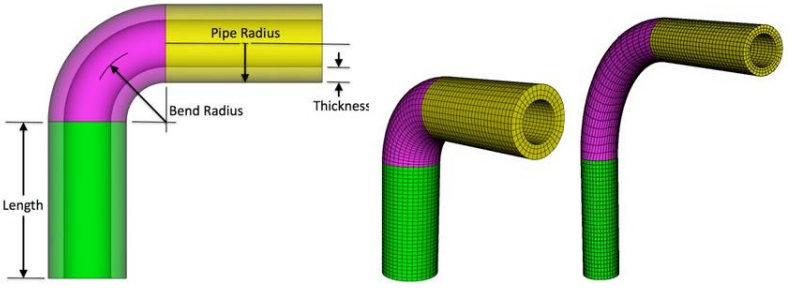


Fig. 3. Example problem "pipe". (a) The parametric variables, (b) the initial shape, (c) the final shape.

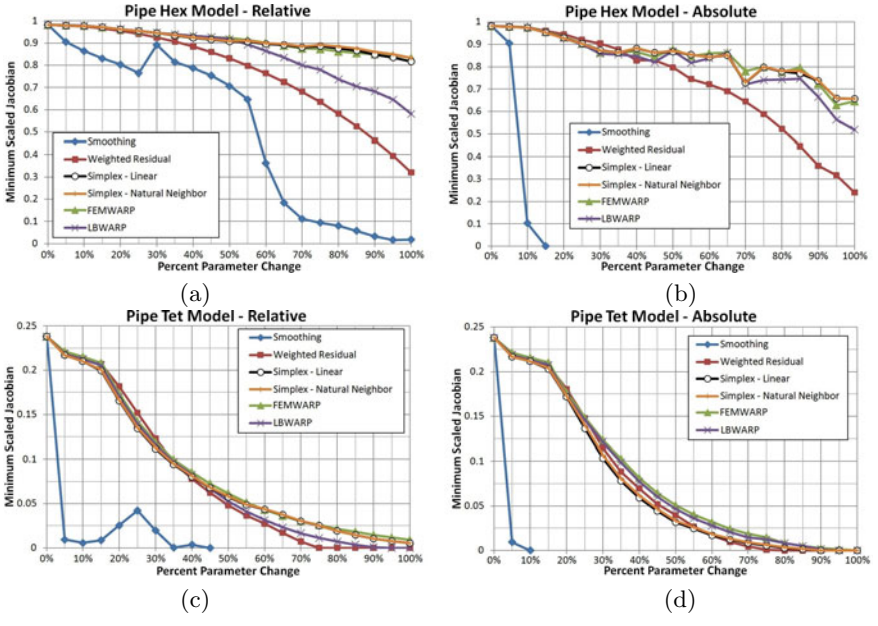


Fig. 4. Results of morphing on pipe model. (a) Relative hex morph, (b) absolute hex morph, (c) relative tet morph, (d) absolute hex morph.

Table 3. Results for morphing pipe model: CPU Time Per Step

Method	Ave Time Per Step (sec)			
	Hex-Relative	Hex-Absolute	Tet-Relative	Tet-Absolute
Smoothing	1.08	1.19	2.56	2.41
Weighted Residual	3.39	3.28	6.08	5.93
FEMWARP	0.31	0.32	0.55	0.56
LBWARP	1.53	1.50	1.88	1.85
Simplex-linear	0.16	0.28	0.31	0.28
Simplex-natural neighbor	0.29	0.35	0.53	0.54

4.3 Canister Model

Figure 5a illustrates the "canister" model which has only one geometric parameter (i.e. the offset of an inner cylindrical hole). Figures 5b and 5c illustrate the initial and ending geometric shapes. The canister model tests severe localized mesh warping, while the majority of the model remains unchanged. Figure 6 shows that for the canister model, FEMWARP and LBWARP provide the best quality. They maintain a positive scaled Jacobian mesh for longer than any other method. The exception is the smoothing algorithm, which performs well for hex-relative (Figure 6a), but poorly for hex-absolute and both tet morphs. The simplex and weighted residual methods also perform well, providing equivelant quality to FEMWARP and LBWARP for the first several steps of all four morphs, until eventually trailing off. Interestingly, simplex-linear and simplex-natural neighbor provide near identical results.

The increased model size of the canister model (∼100k nodes) over the pipe and bore models (10k-20k nodes) is reflected in the execution times in Table 4. For hex, FEMWARP and LBWARP are by far the most expensive, followed by weighted residual. For tet, FEMWARP and LBWARP did not show the spike in computation time. The simplex-linear method is by far the most efficient method, several orders of magnitude faster than the others.

Table 4. Results for morphing canister model: CPU Time Per Step

Method	Ave Time Per Step (sec)			
	Hex-Relative	Hex-Absolute	Tet-Relative	Tet-Absolute
Smoothing	5.10	20.90	3.88	4.02
Weighted Residual	76.24	74.86	46.00	46.52
FEMWARP	215.48	216.39	31.92	31.81
LBWARP	265.14	266.96	43.35	44.50
Simplex-linear	0.55	0.83	0.57	0.69
Simplex-natural neighbor	54.25	52.88	23.43	23.48

4.4 Courier Model

Figure 7 illustrates the "courier" model. Eight parametric variables are modified to test full design optimization. For hex-relative, the simplex methods consistently provide the best quality, followed by FEMWARP. LBWARP is the first to invert the mesh. For hex-absolute and the tet morphs, FEMWARP and the simplex methods provide the best element quality.

Table 5 shows that simplex-linear is again the most efficient method. Interestingly, FEMWARP and LBWARP are also very efficient when compared to the Canister model results. It is clear that the efficiency of FEMWARP and LBWARP are model dependent, likely controlled by the conditioning of the sparse global system being solved.

4.5 Performance and Scaling

Twenty-five variations of a brick model were used to obtain scaling data. An $N \times 20 \times 20, N \in \{20, 30, \ldots 260\}$, hex mesh was fit to a $N \times 20 \times 20$ box. The dimensions of the box were then scaled to $N \times 20 \times 40$, and the mesh morphed to fit. Figures 9a and 9b illustrate the model for $N = 20$ and $N = 30$ respectively. Each $N \times 20 \times 20$ mesh was morphed to the $N \times 20 \times 40$ model with each of the six morphing methods, testing $3D$ affine scaling. Different scaling would likely result with different geometric changes such as torsion or shear.

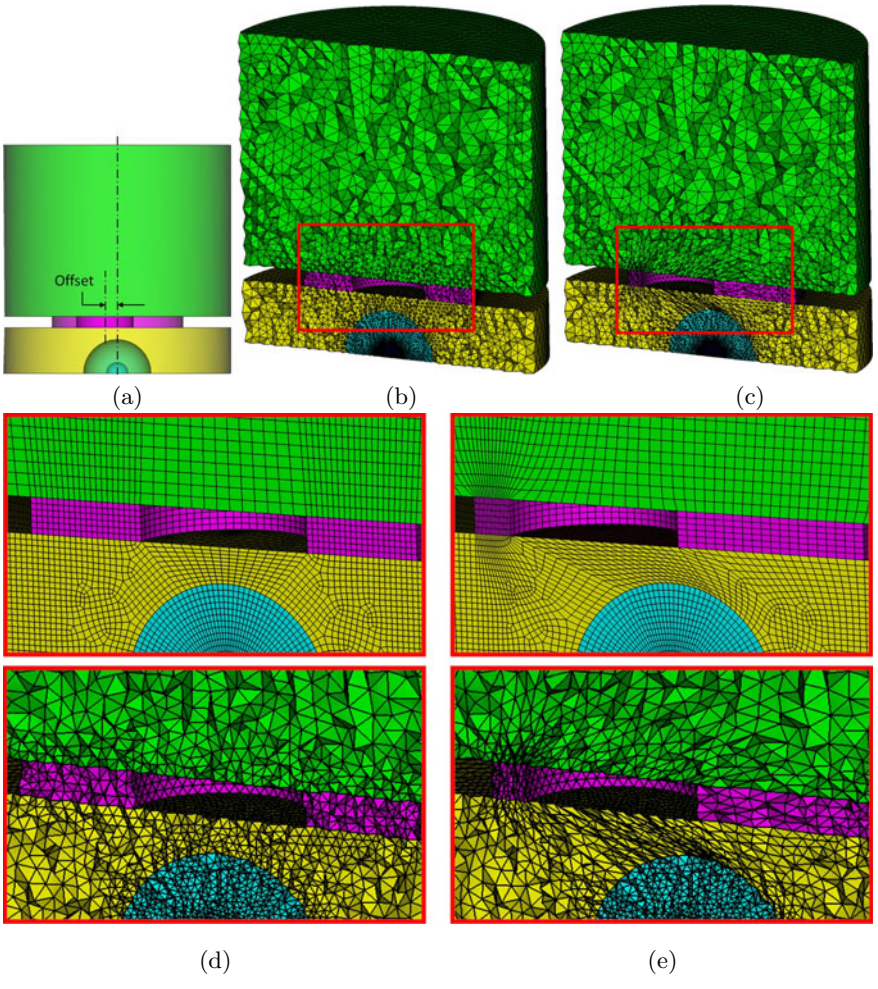


Fig. 5. Example problem "canister". (a) The offset parametric variable, (b) the initial shape, (c) the final shape, (d) zoom in of initial meshes, (e) zoom in of final meshes.

Table 5. Results for morphing courier model: CPU Time Per Step

Method	Ave Time Per Step (sec)			
	Hex-Relative	Hex-Absolute	Tet-Relative	Tet-Absolute
Smoothing	5.13	12.27	37.61	67.40
Weighted Residual	39.70	39.16	69.26	72.76
FEMWARP	4.75	4.89	14.92	14.78
LBWARP	18.55	18.71	28.44	28.49
Simplex-linear	1.07	1.50	1.95	2.55
Simplex-natural neighbor	6.69	7.45	13.23	12.67

The execution times are plotted in Figure 10. Figure 10 shows that simplex-linear, simplex-natural neighbor and weighted residual scale nearly linearly, while LBWARP, FEMWARP and smoothing scale to a power of ∼1.50. The simplex methods scale the best and are the most efficient. LBWARP and FEMWARP are the most expensive since they require a solution to a global sparse matrix system.

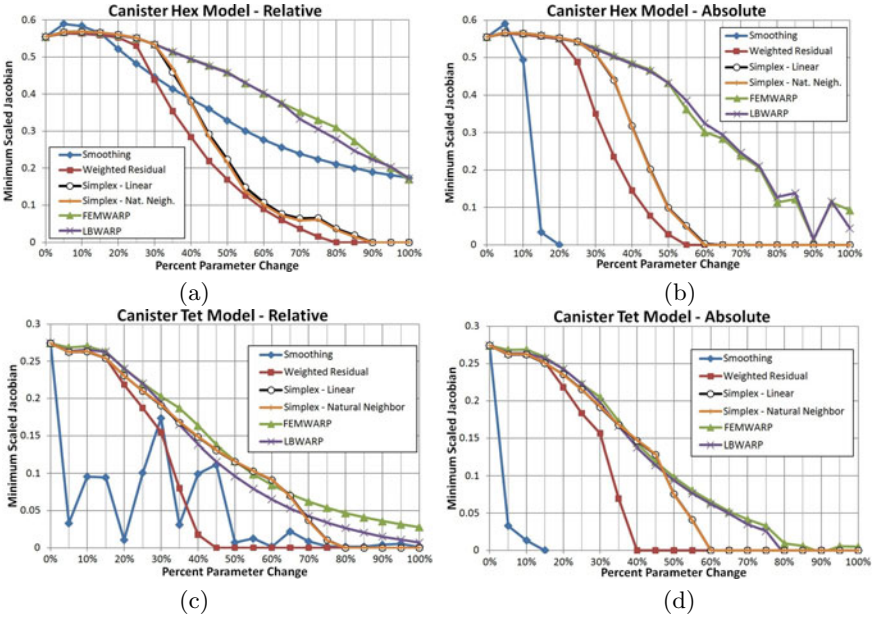
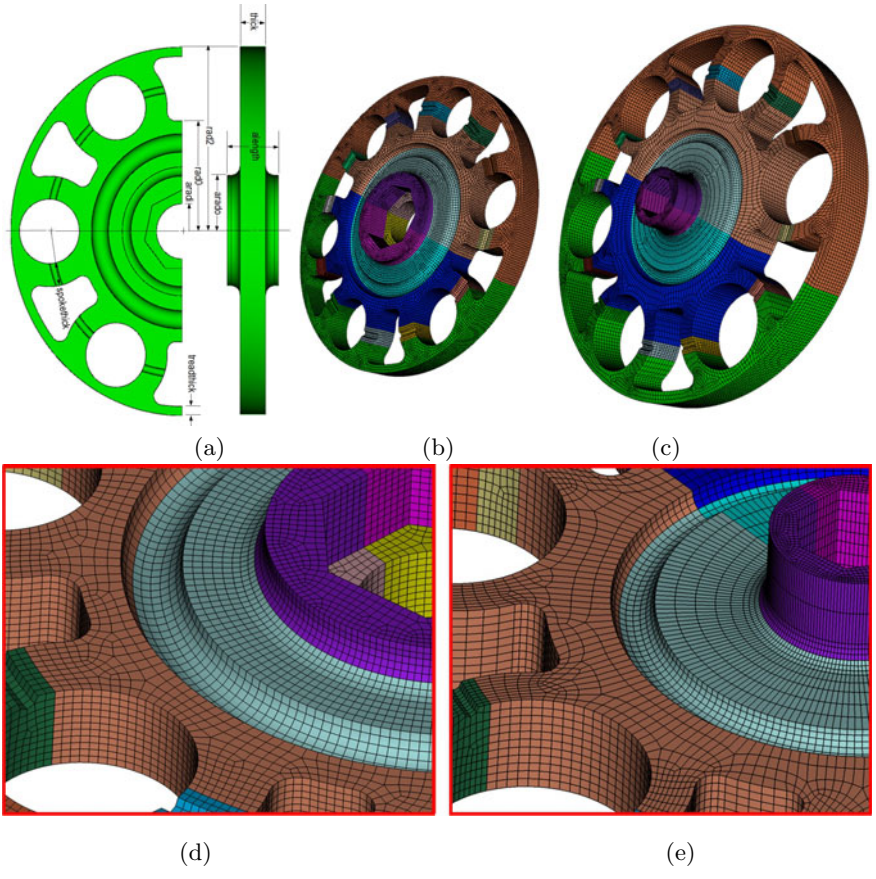


Fig. 6. Results of morphing on canister model. (a) Relative hex morph, (b) absolute hex morph, (c) relative tet morph, (d) absolute hex morph.



(a) (b) (c)

(d) (e)

Fig. 7. Example problem "courier". (a) The parametric variables, (b) the initial shape, (c) the final shape, (d) zoom in of initial mesh, (e) zoom in of final mesh.

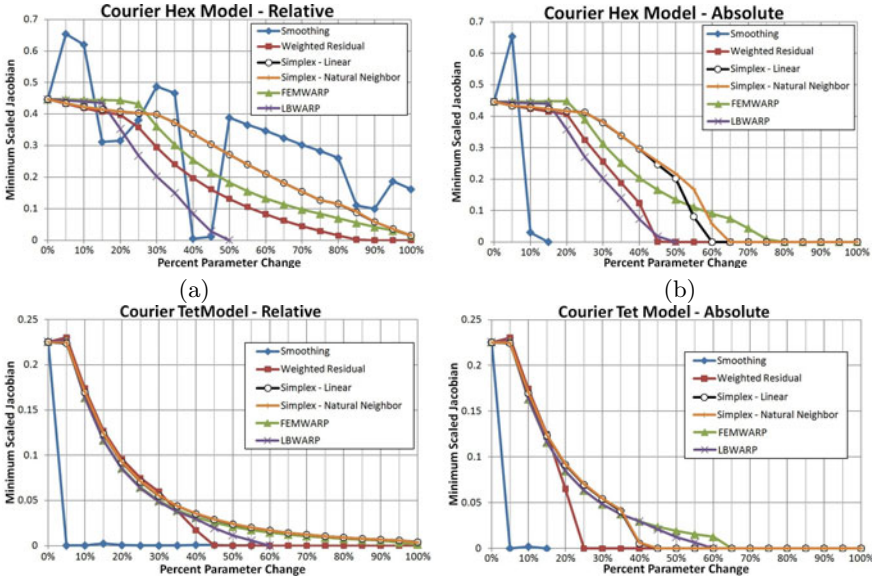


Fig. 8. Results of morphing on courier model. (a) Relative hex morph, (b) absolute hex morph, (c) relative tet morph, (d) absolute hex morph.

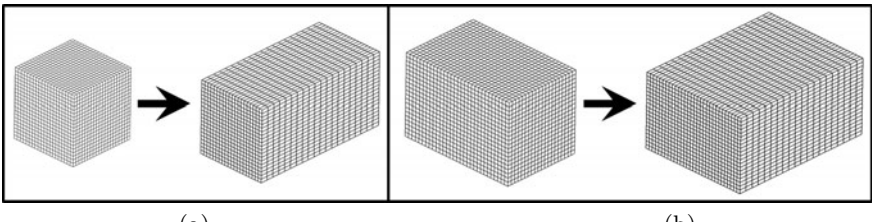


Fig. 9. Model for scaling study, (a) $N = 20$, (b) $N = 30$.

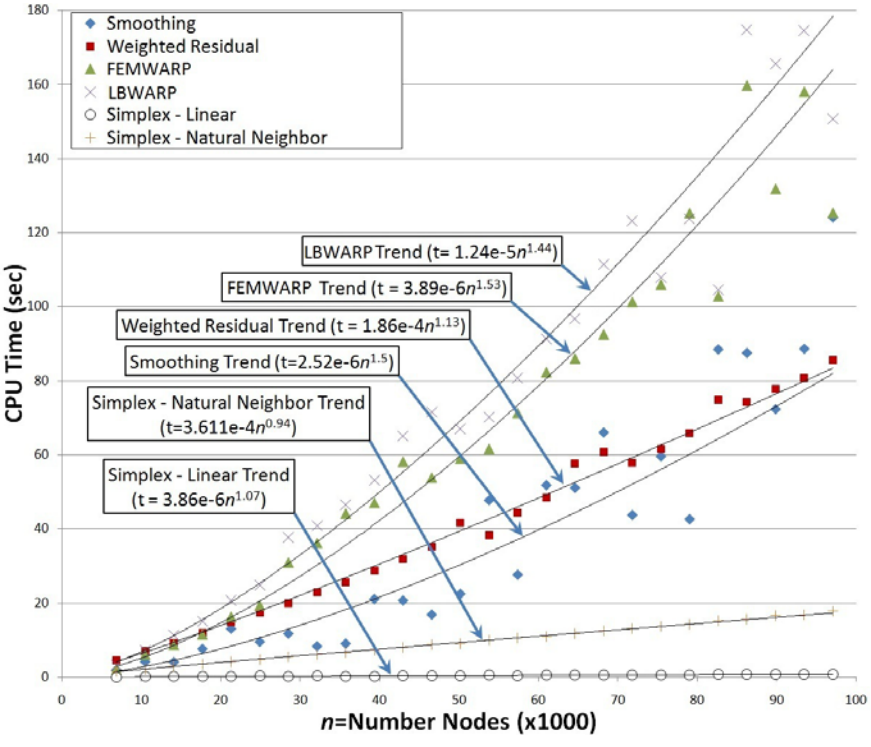


Fig. 10. Scaling (time per step).

5 Conclusions

We have compared six $3D$ mesh morphing methods on eight example models. Here we make conclusions based on the results of this comparison.

Smoothing: The smoothing method tracks small linear changes in geometry very well, as illustrated by the bore model. However, absolute (large step) morphing and nonlinear transformations, such as those in the pipe, courier, and canister models, proved impossible to capture. The morph of the model boundary produces a severely tangled mesh for large transformations, which a smoother cannot always untangle. Likely additional smoothing iterations are needed. However, in general, smoothing is not a reliable morphing method.

Weighted Residual: The weighted residual method performed on par with the other methods for simple models such as the bore model. However, as model complexity increased, weighted residual and smoothing are generally the first to produce poor mesh quality. Quality might be improved by increasing *npts* in equations (4) and (6), at the cost of decreased efficiency.

Simplex-Linear: This method proved effective for all example models including those with large transformations and complex geometry such as

the courier and canister models. Performance and scaling is excellent due to the ability to localize the location of containing tets and the computation of barycentric coordinates. The most expensive part, however minimal, is the generation of the background Delaunay tesselation. However, there are good quality Delaunay tesselators available [27]. Even with the cost of generating the tesselation, simplex-linear is still by far the least computationally expensive method. Considering reasonable element quality and excellent performance, simplex-linear should be considered for any morphing implementation.

Simplex-Natural Neighbor: This method was included in the study in anticipation of improved element quality over simplex-linear by using a more judicious selection of weighting vertices. Using our implementation, both barycentric and natural neighbor weighting exhibited very similar results. As suggested by [29], augmenting the natural neighbor interpolant to use a nodal based gradient function may provide further improvement. However, with our current implementation, we observed little improvement in element quality and decreased performance over simplex-linear. Given that this method is more difficult to implement than simplex-linear, further studies would be needed before this method could be recommended over simplex-linear.

LBWARP: LBWARP provided excellent element quality on all test cases, except for the courier model with hexes. On the courier-hex model, element quality is still good, but less than the other methods. However, scaling of LBWARP is a concern. LBWARP requires solving an $n \times n$ sparse linear system, where n is the number of interior nodes being morphed. An efficient sparse linear solver is critical for any LBWARP implementation. Our example models show that LBWARP can morph meshes with ∼100k elements in a few minutes. Computation times may become unacceptable as model sizes increase beyond a few hundred thousand nodes. Efficiency also varies significantly, for similarly sized models, as shown by a comparison of Tables 4 and 5. LBWARP is also complicated to implement. Tolerance and convergence issues must be considered carefully in the computation of the initial weights, w_{ij}, particularly with hexahedral meshes which tend to have large sets of coplanar element faces. Finally, LBWARP requires that the initial mesh be non-inverted (i.e. all elements being convex). Computation of initial weights, w_{ij}, requires that each point lie in the convex hull of its neighbors, which is not guaranteed in inverted meshes. The other five methods do not have this restriction.

FEMWARP: FEMWARP maintained the best element quality for all test cases and is easy to implement. Computation/assembly of local element stiffness matrices is the most difficult coding task. Although, FEMWARP requires solving an $n \times n$ global sparse system matrix, where n is the number of interior nodes, and efficiency varies significantly for similarly sized models, easy parallelization should minimize scaling concerns. We recommend FEMWARP based on its excellent element quality and easy of implementation.

In summary we recommend that both simplex-linear and FEMWARP should be considered when implementing a mesh morphing system. FEMWARP provides excellent element quality and easily extended to parallel for large models. The simplex-linear method also provides reasonable element quality, with excellent performance even for very large models. Additional studies would be required to identify the context in which the other morphing methods presented in this paper would be a reasonable option.

Finally, we observe that hexahedral meshes tend to maintain higher element quality longer than tetrahedral meshes during morphing, particularly with structued hex meshes, likely due to the more flexible element topology of hexahedral elements. For example, in the bore and pipe models, minimum element quality degrades only slightly for the hex meshes, while the quality of tet meshes on the same model become inverted (compare Figures 2a, 2b, 4a, and 4b with Figures 2c, 2d, 4c, and 4d). This same effect is seen to a lesser effect with the unstructured hex meshes on the canister and courier models. Maintaining high element quality during morphing is more important in hexahedral meshes because local remeshing of hexahedra is generally considered intractable, while straightforward for tetrahedral meshes.

References

1. Baker, T.J.: Mesh movement and metamorphosis. In: Proc. 10th Int. Meshing Roundtable, pp. 387–396 (2001)
2. Branets, L., Carey, G.F.: A local cell quality metric and variational grid smoothing algorithm. Eng. Comput. 21, 19–28 (2005)
3. Brewer, M., Diachin, L., Knupp, P., Leurent, T., Melander, D.: The Mesquite mesh quality improvement toolkit. In: Proc. 12th Int. Meshing Roundtable, pp. 239–250 (2003)
4. Cardoze, D., Miller, G., Olah, M., Phillips, T.: A Bézier-based moving mesh framework for simulation with elastic membranes. In: Proc. 13th Int. Meshing Roundtable, 71-80 (2004)
5. Dai, M., Schmidt, D.P.: Adaptive tetrahedral meshing in free-surface flow. J. Comp. Phys. 208, 228–252 (2005)
6. Davis, T.A., Natarajan, E.P.: Algorithm 8xx:KLU, a direct sparse solver for circuit simulation problems. ACM Transactions on Mathematical Software, 1-17 (2011)
7. Floater, M.S.: One-to-one piecewise linear mappings over triangulations. Math. Comp. 72, 685–696 (2003)
8. GSH3D, INRIA, `http://www-roc.inria.fr/gamma/gamma/ghs3d/ghs.php`
9. Helenbrook, B.T.: Mesh deformation using the biharmonic operator. Int. J. Num. Meth. Engr. 56, 1007–1021 (2003)
10. Jacobson, A., Baran, I., Popović, J., Sorkine, O.: Bounded Biharmonic Weights for Real-Time Deformation. To Appear In ACM Transactions On Graphics (Proc. Of ACM SIGGRAPH) 30(4) (2011), `http://igl.ethz.ch/projects/bbw/`
11. Knupp, P.: Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities, part I. Int. J. Num. Meth. Engr. 48, 401–420 (2000)

12. Knupp, P.: Next-generation sweep tool: a method for generating all-hex meshes on two-and-one-half dimensional geomtries. In: Proc. 7th Int. Meshing Roundtable, pp. 505–513 (1998)
13. Knupp, P.: Updating meshes on deforming domains: An application of the target-matrix paradigm. Commun. Num. Meth. Engr. 24(6), 467–476 (2007)
14. Li, R., Tang, T., Zhang, P.: Moving mesh methods in multiple dimensions based on harmonic maps. J. Comput. Phys. 170, 562–588 (2001)
15. Michler, A.K.: Aircraft control surface deflection using RBF-based mesh deformation. Int. J. Num. Meth. Engr. (2011), doi: 10.1002/nme.3208
16. Nocedal, J., Wright, S.J.: Numerical optimization, 2nd edn. Springer, Heidelberg (2006)
17. Shontz, S.M., Vavasis, S.A.: Analysis of and workarounds for element reversal for a finite element-based algorithm for warping triangular and tetrahedral meshes. BIT, Numerical Mathematics 50, 863–884 (2010)
18. Shontz, S.M., Vavasis, S.A.: A mesh warping algorithm based on weighted Laplacian smoothing. In: Proc. 12th Int. Meshing Roundtable, pp. 147–158 (2003)
19. Shontz, S.M., Vavasis, S.A.: A robust solution procedure for hyperelastic solids with large boundary deformation. Eng. Comput. (2011), doi: 10.1007/s00366-011-0225-y
20. Sibson, R.: A brief description of natural neighbor interpolation. In: Interpretting Multivariate Data, pp. 21–36. John Wiley and Sons, New York (1981)
21. Sigal, I.A., Hardisty, M.R., Whyne, C.M.: Mesh-morphing algorithms for specimen-specific finite element modeling. J. Biomech. 41(7), 1381–1389 (2008)
22. Stein, K., Tezduyar, T., Benney, R.: Mesh moving techniques for fluid-structure interactions with large displacements. Trans. ASME 2003 70, 58–63 (2003)
23. Stein, K., Tezduyar, T., Benney, R.: Automatic mesh update with the solid-extension mesh moving technique. Comput. Meth. Appl. M. 193, 2019–2032 (2004)
24. Staten, M.L., Canann, S.A., Owen, S.J.: BMSWEEP: locating interior nodes during sweeping. Eng. Comput. 15(3), 212–218 (1999)
25. Tezduyar, T., Behr, M., Mittal, S., Johnson, A.A.: Computation of unsteady incompressible flows with the finite element methods – Space-time formulations, iterative strategies and massively parallel implementations. New Methods in Transient Analysis, PVP-vol. 246/AMD-vol. 143, ASME, New York, 7-24 (1992)
26. The Trilinos Project, Sandia Nat. Lab., `http://trilinos.sandia.gov/`
27. Vavasis, S.: QMG: mesh generation and related software (2010), `http://www.cs.cornell.edu/home/vavasis/qmg-home.html`
28. Vurputoor, R., Mukherjee, N., Cabello, J., Hancock, M.: A mesh morphing technique for geometrically dissimilar tesselated surfaces. In: Proc. 16th Int. Meshing Roundtable, pp. 315–334 (2007)
29. Watson, D.F.: nngridr: an implementation of natural neighbor interpolation. University of Western Australia (1994)
30. Wright, S.J.: Primal-dual interior-point methods. SIAM (1997)