Engineering with Computers manuscript No.
(will be inserted by the editor)

RBF Morphing Techniques for Simulation-based Design Optimization

Daniel Sieger · Stefan Menzel · Mario Botsch

Received: date / Accepted: date

Abstract Morphing an existing simulation mesh according to updated geometric parameters in the underlying computer-aided design model is a crucial technique within fully automatic design optimization. By avoiding costly automatic or even manual meshing, it enables the automatic and parallel generation and evaluation of new design variations, e.g., through finite element or computational fluid dynamics simulations. In this paper, we present a simple yet versatile method for high quality mesh morphing. Building upon triharmonic radial basis functions, our shape deformations minimize distortion and thereby implicitly preserve shape quality. Moreover, the same unified code can morph tetrahedral, hexahedral, or arbitrary polyhedral meshes. We compare our method to other recently proposed techniques and show that ours yields superior results in most cases. We analyze how to explicitly prevent inverted mesh elements by successively splitting the deformation into smaller steps. Finally, we investigate the performance of different linear solvers as well as the use of an incremental least squares solver for the sake of improved scalability.

Keywords Mesh morphing · Design optimization · Radial basis functions

A preliminary version of a portion of these results appeared in shortened form in the Proceedings of the 2012 International Meshing Roundtable

D. Sieger
Bielefeld University, Postfach 100 131, D-33501 Bielefeld, Germany
E-mail: dsieger@techfak.uni-bielefeld.de

S. Menzel
Honda Research Institute Europe GmbH, Carl-Legien-Str. 30,
D-63073 Offenbach/Main, Germany
E-mail: stefan.menzel@honda-ri.de

M. Botsch
Bielefeld University, Postfach 100 131, D-33501 Bielefeld, Germany
E-mail: botsch@techfak.uni-bielefeld.de

1 Introduction

Simulation-based design optimization is becoming a cornerstone of the product development process of the automotive industry, aircraft construction, and naval architecture. The adaptation of an existing volumetric simulation mesh according to an updated computer-aided design (CAD) geometry is a key component for performing the optimization in a fully automatic and parallel manner. The importance of such a component further increases when dealing with complex geometries that prohibit automatic mesh generation and require manual interaction by an expert instead, or when using stochastic optimization techniques—such as evolutionary algorithms—which typically require the creation and evaluation of a large number of design variations in order to find a feasible solution.

The adaptation of an existing simulation mesh is accomplished by using a mesh morphing or mesh warping technique: Given an initial CAD surface $\mathcal{G}$ and a volumetric mesh $\mathcal{M}$ of that geometry, a shape variation $\mathcal{G} \mapsto \mathcal{G}'$ is generated by changing the geometric embedding of $\mathcal{G}$ while keeping its topology fixed. The mesh morphing technique then adapts the mesh $\mathcal{M}$ such that the updated version $\mathcal{M}'$ conforms to the updated boundary surface $\mathcal{G}'$. Analogously to the geometric changes in the CAD model, only the geometric embedding of $\mathcal{M}$ (i.e., its node positions) is updated in this process, while the mesh topology (i.e., its connectivity) stays fixed.

Mesh morphing techniques aim at preserving the element quality as much as possible, thereby allowing for as large as possible geometric changes before inevitably requiring some remeshing due to element inversion. Staten and coworkers recently proposed and evaluated several mesh morphing techniques, which they compared with respect to computational performance and element quality on different tetrahedral and hexahedral meshes [35].

Motivated by the work of Staten et al. [35], building on their results, and contributing to their benchmarking comparisons, we present and evaluate a meshless morphing technique based on triharmonic radial basis functions (RBFs). Our method yields highly smooth space warps that minimize distortion and thereby preserve element quality. While being computationally more expensive, our method offers the following compelling advantages: It is easy to understand and straightforward to implement; it can be applied to tetrahedral, hexahedral, or general polyhedral meshes; and finally, it more robustly achieves higher quality results.

2 Related Work

The recent comparison of mesh morphing methods published by Staten and colleagues [35] constitutes the starting point for our investigation. Based on a set of test scenarios involving varying complexity and topology the authors benchmark six techniques for warping volume meshes. Besides in the meshing community, mesh morphing or mesh deformation methods have also been a subject of intensive research in computer graphics. Even though many of the techniques stemming from the graphics community are typically only applied to surface meshes, we concentrate our discussion on methods which are applicable to volumetric meshes as well. We can roughly classify morphing approaches from within both fields into four categories: methods based on generalized barycentric coordinates, mesh smoothing techniques, mesh-based variational methods that minimize certain smoothness energies, and meshless warping approaches. Most techniques take the updated boundary node positions as input and compute the new locations of interior nodes from these boundary constraints.

Approaches based on *barycentric coordinates* determine the interior nodes as a linear (affine or convex) combination of the boundary nodes through a generalization of linear barycentric interpolation [37]. Examples are Wachspress coordinates [39], mean value coordinates [11], harmonic coordinates [20], and maximum entropy coordinates [17, 36]. The simplex-linear method of [35], being a generalization of BMSWEEP [34], as well as its extension to natural neighbor interpolation [33], also belong to this category. While these approaches typically have simple geometric constructions and therefore are easy to implement and efficient to compute, the resulting morphs might not be smooth enough to reliably preserve element quality.

Mesh smoothing methods adjust interior node locations in order to explicitly optimize mesh element quality [21, 22, 30], where the Mesquite framework [7] offers implementations based on mean ratio, untangling, and matrix condition number [21]. In the context of mesh warping, the updated boundary nodes act as fixed constraints while the optimization process determines the interior nodes. The mesh smoothing methods evaluated in [35] worked well for small geometric changes, but were lacking robustness for larger deformations. In comparison, the LBWARP method [30], a weighted Laplacian smoothing based on the log-barrier technique, gives considerably better results, but is computationally more complex.

Mesh-based variational methods compute smooth harmonic or biharmonic deformations by solving Laplacian or bi-Laplacian systems [3, 16], which is numerically more robust than most mesh smoothing techniques. The finite element-based FEMWARP technique [3], which computes a harmonic deformation, was generalized from tetrahedra to hexahedra in [35], and turned out to be the most successful approach in Staten's benchmarks. Note that harmonic coordinates [20] (see above) are closely related to these approaches, since they are also derived by solving a Laplacian system. The boundary nodes' harmonic coordinate functions can be thought of as "response functions" of the Laplacian PDE. While the morphs produced by mesh-based variational methods tend to preserve element quality well, they have to be custom-tailored to each mesh type (e.g., tetrahedral or hexahedral).

In contrast, *meshless morphing techniques* avoid this limitation by computing a space warp $\mathbf{d}\colon \mathbb{R}^3 \to \mathbb{R}^3$ that deforms the whole embedding space, thereby implicitly deforming each node of the mesh $\mathcal{M}$. After the initial freeform deformation (FFD) paper [29], many variants and extensions have been proposed. We refer the reader to the survey papers [4, 13, 27], which focus on mathematical formalisms for the different methods, on the interactive manipulation by a designer, and on shape deformation in the context of aerodynamic design optimization, respectively. However, spline-based FFD does not offer the same degree of smoothness as harmonic or biharmonic morphs, and it requires a rather tedious lattice setup.

We propose to combine the advantages of meshless approaches and mesh-based variational methods by using *radial basis functions* (RBFs) for mesh morphing [5, 6, 19, 25]. Our RBF space warps are easy to setup and compute, can handle arbitrary polyhedral meshes, and offer a degree of smoothness equivalent or even superior to mesh-based variational techniques. Moreover, we show how to employ the same approach not only for morphing volumetric simulation meshes (Section 3), but also for morphing the updated surface node locations (Section 4).

3 RBF Volume Morphing

In this section we describe how to morph volume meshes using radial basis functions. We note, however, that our technique is not restricted to this particular application setting. In fact, the RBF approach is capable to handle arbitrary geometries whenever it is possible to specify the deformation as a set of displacements give at certain positions in space.

The input for the volume morphing consists of three sets of mesh vertices: *Surface nodes* $\{\mathbf{s}_1, \ldots, \mathbf{s}_m\}$ and interior *volume nodes* $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ of the initial mesh $\mathcal{M}$, as well as the updated *surface nodes* $\{\mathbf{s}'_1, \ldots, \mathbf{s}'_m\}$ of $\mathcal{M}'$, where the $\mathbf{s}_i$ and $\mathbf{s}'_i$ conform to the CAD geometries $\mathcal{G}$ and $\mathcal{G}'$, respectively. The goal is to find updated *volume node* positions $\{\mathbf{v}'_1, \ldots, \mathbf{v}'_n\}$, such that the element quality of the morphed mesh $\mathcal{M}'$ is as good as possible.

On a more abstract level, we can treat the morphing problem as a scattered data interpolation problem: We search for a function $\mathbf{d}\colon \mathbb{R}^3 \to \mathbb{R}^3$ that (i) exactly interpolates the prescribed boundary displacements $\mathbf{d}(\mathbf{s}_i) = (\mathbf{s}'_i - \mathbf{s}_i)$ and (ii) smoothly interpolates these displacements into the mesh interior. Radial basis functions (RBFs) are well known to be suitable for solving this type of problem [41]. Using RBFs we define the deformation function as a linear combination of radially symmetric kernel functions $\varphi_j(\mathbf{x}) = \varphi(\|\mathbf{x} - \mathbf{x}_j\|)$, located at centers $\mathbf{x}_j \in \mathbb{R}^3$ and weighted by $\mathbf{w}_j \in \mathbb{R}^3$, plus a linear polynomial to guarantee linear precision:

$$\mathbf{d}(\mathbf{x}) \;=\; \sum_{j=1}^{m} \mathbf{w}_j \varphi_j(\mathbf{x}) \;+\; \sum_{k=1}^{4} \mathbf{q}_k \pi_k(\mathbf{x})\,, \tag{1}$$

where $\{\pi_1, \pi_2, \pi_3, \pi_4\} = \{x, y, z, 1\}$ is a basis of the space of linear trivariate polynomials, weighted by coefficients $\mathbf{q}_k \in \mathbb{R}^3$. Note that the polynomial term is important, since it guarantees to find the optimal affine motion (translation, rotation, scaling) contained in the prescribed displacements $\mathbf{s}_i \mapsto \mathbf{s}'_i$.

The choice of the kernel function $\varphi\colon \mathbb{R} \to \mathbb{R}$ basically determines the shape of the interpolant. Commonly used kernels include Gaussians, (inverse) multiquadrics, and polyharmonic splines (see Table 1 for an overview). In our application we aim for high quality deformations minimizing the distortion of mesh elements. To meet this goal, we have to use a sufficiently smooth kernel function. While Gaussian and multiquadric basis functions provide infinite smoothness, i.e., they are C^∞ at the center, they require the choice of an additional shape parameter (the ϵ in Table 1). Small values of ϵ increase approximation accuracy, but lead to numerically instabilities, and *vice versa*. Therefore, finding the optimal shape parameter for a given radial basis function and the particular application is a non-trivial task of its own (see [10] for an overview of different strategies).

Gaussian	$\varphi(r) = \mathrm{e}^{-(\epsilon r)^2}$
Multiquadric	$\varphi(r) = \sqrt{1 + (\epsilon r)^2}$
Inverse multiquadric	$\varphi(r) = 1/\sqrt{1 + (\epsilon r)^2}$
Polyharmonic spline in $\mathbb{R}^d$	$\varphi_k(r) = \begin{cases} r^{2k-d}, & d \text{ odd}, \\ r^{2k-d}\log(r), & d \text{ even}. \end{cases}$

Table 1 Commonly used radial basis functions. For Gaussians and (inverse) multiquadrics ϵ denotes the shape parameter. For the polyharmonic splines k denotes the order of smoothness.

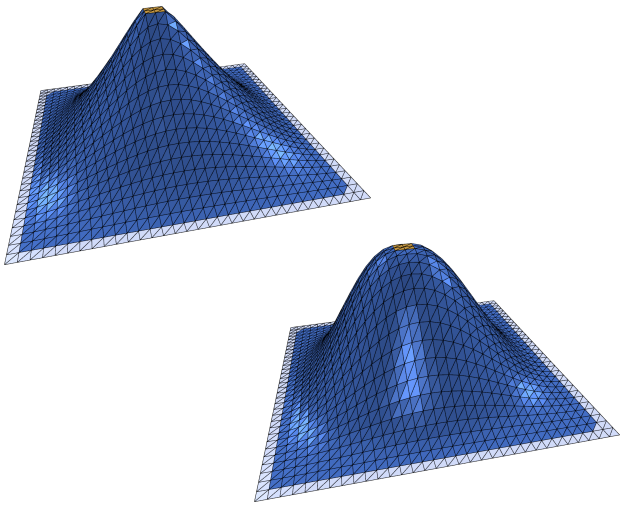


Fig. 1 Comparison between a biharmonic (top) and a triharmonic (bottom) deformation of a plane. We displace the golden region, keep the gray region fixed, and deform the blue region. We place RBF kernels on all vertices in the golden and gray regions.

In contrast, polyharmonic splines are free of shape parameters, but only of finite smoothness. Depending on the application scenario, we have to choose a sufficiently high degree of smoothness. In $\mathbb{R}^3$ the polyharmonic spline $\varphi_k(r) = r^{2k-3}$ is a fundamental solution of the k-th order Laplacian Δ^k, such that also the RBF deformation (1) is k-harmomic, i.e., $\Delta^k \mathbf{d} = 0$. Being the strong form of a variational energy minimization, this is equivalent to $\mathbf{d}$ minimizing [41]

$$\iiint_{\mathbb{R}^3} \left\| \frac{\partial^k \mathbf{d}}{\partial x^k} \right\|^2 + \left\| \frac{\partial^k \mathbf{d}}{\partial x^{k-1} \partial y} \right\|^2 + \ldots + \left\| \frac{\partial^k \mathbf{d}}{\partial z^k} \right\|^2 \mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z. \tag{2}$$

Following [35] we measure element quality by means of the minimum scaled Jacobian, i.e., by first-order partial derivatives. Hence, in order to preserve element quality during morphing, we should minimize the change of first-order derivatives of the elements, and therefore the first-order derivatives of the deformation function. With $k = 1$ in (2), this is achieved by the harmonic RBF $\varphi(r) = 1/r$, but those basis functions are singular at their centers. The biharmonic spline $\varphi(r) = r$ is well defined, but not differentiable at the center and therefore not smooth enough for our application (see Figure 1). By choosing $\varphi(r) = r^3$, we obtain a deformation function that is triharmonic, therefore penalizes third-order derivatives in (2), and is globally C^2 smooth. With these properties, it is the lowest-order polyharmonic RBF suitable for our application. Since for numerical robustness a low order is preferrable, we eventually chose triharmonic RBFs for volume morphing. Nevertheless we also give comparisons to other basis functions in terms of element quality and numerical conditioning in Section 5.4.

Satisfying the interpolation constraints $\mathbf{d}(\mathbf{s}_i) = (\mathbf{s}'_i - \mathbf{s}_i)$ amounts to placing RBF kernels at the constraint positions (i.e., $\mathbf{x}_j = \mathbf{s}_j$) and finding the coefficients $\mathbf{w}_j$ and $\mathbf{q}_k$ by solving the $(m+4) \times (m+4)$ linear system

$$\mathbf{A}\mathbf{X} = \mathbf{B}, \tag{3}$$

where

$$\mathbf{A} = \begin{pmatrix} \varphi_1(\mathbf{s}_1) & \cdots & \varphi_m(\mathbf{s}_1) & \pi_1(\mathbf{s}_1) & \cdots & \pi_4(\mathbf{s}_1) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \varphi_1(\mathbf{s}_m) & \cdots & \varphi_m(\mathbf{s}_m) & \pi_1(\mathbf{s}_m) & \cdots & \pi_4(\mathbf{s}_m) \\ \pi_1(\mathbf{s}_1) & \cdots & \pi_1(\mathbf{s}_m) & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \pi_4(\mathbf{s}_1) & \cdots & \pi_4(\mathbf{s}_m) & 0 & \cdots & 0 \end{pmatrix},$$

$$\mathbf{X} = \Big(\mathbf{w}_1, \dots, \mathbf{w}_m, \mathbf{q}_1, \dots, \mathbf{q}_4\Big)^T,$$

and

$$\mathbf{B} = \Big((\mathbf{s}'_1 - \mathbf{s}_1), \dots, (\mathbf{s}'_m - \mathbf{s}_m), \mathbf{0}, \dots, \mathbf{0},\Big)^T.$$

After solving (3) we can compute the morphed mesh $\mathcal{M}'$ by simply evaluating the RBF deformation at each volume node: $\mathbf{v}'_i = \mathbf{v}_i + \mathbf{d}(\mathbf{v}_i)$. This part can easily be parallelized and therefore is highly efficient. The computationally most expensive part is the solution of the linear system (3), which is dense due to the global support of $\varphi(r)$. We discuss the performance and the scalability of our method in Section 7.

4 Surface Morphing

In order to establish a common baseline for comparison, the benchmark tests of Staten and colleagues [35] were all based on the same surface morphing, i.e., they all shared the same boundary surface nodes for $\mathcal{M}$ and $\mathcal{M}'$. However, while the quality of our RBF volume morphing alone is already promising, we point out that the surface morph entirely determines the volume morph, thereby constituting an upper bound on the maximum quality achievable. In order to further improve the quality of our results we propose a high quality surface morphing method that is also based on polyharmonic radial basis functions.

Since in our setting the topology of the CAD surface $\mathcal{G}$ stays constant, there is a one-to-one correspondence between the faces, curves, and corner vertices of $\mathcal{G}$ and $\mathcal{G}'$. Staten and colleagues exploit this fact for morphing curves: For each *curve node* $\mathbf{c}_i \in \mathcal{M}$ belonging to a curve $\mathbf{f}\colon \mathbb{R} \to \mathbb{R}^3$ of the initial geometry $\mathcal{G}$, they find the parameter value u such that $\mathbf{c}_i = \mathbf{f}(u)$ and compute the morphed node as $\mathbf{c}'_i = \mathbf{f}'(u)$, where $\mathbf{f}' \subset \mathcal{G}'$ is the morphed curve corresponding to $\mathbf{f} \subset \mathcal{G}$.

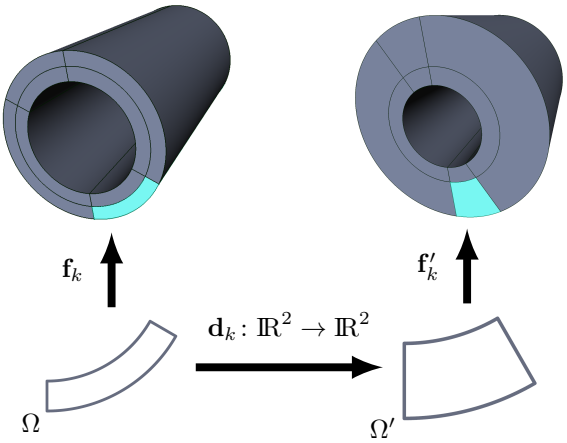


Fig. 2 Overview of the surface morphing for one face.

The morphed curves then act as boundary constraints for morphing the surface nodes, which Staten et al. performed using either mesh smoothing or the weighted residual technique. Both, however, lead to a certain amount of distortion or even inverted surfaces triangles, which in turn negatively impacts the volume morphing. We note, however, that Staten and colleagues did not focus on the lower-dimensional surface warp but on a comparison of volume morphing techniques. It is also worth noting that Shontz and Vavasis show how to employ lower-dimensional variants of their LBWARP technique for mesh construction and warping in [30].

In our approach we extend the curve morphing idea to the surface case: For each surface node $\mathbf{s}_i$ we find its corresponding face $\mathbf{f}\colon \mathbb{R}^2 \to \mathbb{R}^3$ and its (u, v)-parameters, and define the morphed surface node as the corresponding point on the morphed face $\mathbf{f}'(u, v)$. We first describe how to find the (u, v)-parameters of a surface node $\mathbf{s}_i$, before explaining the actual mapping from $\mathbf{f}$ to $\mathbf{f}'$.

Given a surface node $\mathbf{s}_i$, finding its corresponding face $\mathbf{f}$ and (u, v) parameters in theory amount to projecting $\mathbf{s}_i$ onto each face $\mathbf{f}_k \in \mathcal{G}$ and selecting the closest one. Although most CAD kernels (Open CASCADE [26] in our case) provide this functionality, in practice these projections are both computationally intensive and numerically instable for complex, trimmed faces. We address both problems by densely sampling the CAD surface $\mathcal{G}$, which requires only robust and efficient evaluations and results in samples $\mathbf{p}_j = \mathbf{f}_k(u_j, v_j)$. For each surface node $\mathbf{s}_i$ we then find its closest sample point $\mathbf{p}_j$ and project $\mathbf{s}_i$ onto $\mathbf{f}_k$ with (u_j, v_j) as initial guess. We perform this search efficiently through space partitioning: when storing the samples $(\mathbf{p}_j, u_j, v_j, k)$ in a kD-tree [28], finding the closest sample for a given $\mathbf{s}_i$ takes less than 0.01ms for a highly dense sampling of about 15M sample points. In our experiments, this approach drastically improved the efficiency and robustness of the projections.

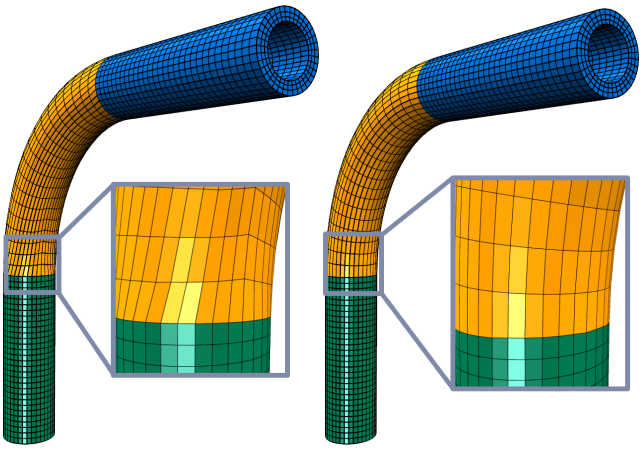


Fig. 3 Comparison of element quality in the morphed surface mesh. While there are distortions in the meshes of Staten et al. [35] (left), our surface (right) is perfectly aligned to the updated CAD geometry.

After finding the face $\mathbf{f}$ and the (u, v) parameters, we need to move the node $\mathbf{s}_i$ to the corresponding point on the morphed CAD face $\mathbf{f}'$. This part is more challenging than for the curve case, since the geometric embedding of a face $\mathbf{f}\colon \Omega \to \mathbb{R}^3$ can change in two ways: (i) an adjustment of its geometric parameters, e.g., spline control points or cylinder radii, and (ii) a change of its parameter domain Ω, e.g., due to adjusted trimming curves. While (i) simply amounts to evaluating $\mathbf{f}'$ instead of $\mathbf{f}$, (ii) requires to morph the parameter values $(u, v) \in \Omega$ to $(u', v') \in \Omega'$.

In order to morph the parameter values to the updated parametric domain, we exploit the versatility of our approach and construct a 2D RBF deformation function $\mathbf{d}_k\colon \mathbb{R}^2 \to \mathbb{R}^2$ for each face $\mathbf{f}_k$ of the CAD model (see Figure 2). To this end we uniformly sample the (u, v)-boundary curves of the faces $\mathbf{f}_k$ and $\mathbf{f}'_k$, resulting in 2D point samples $\{\mathbf{c}_1, \ldots, \mathbf{c}_n\} \in \Omega$ and $\{\mathbf{c}'_1, \ldots, \mathbf{c}'_n\} \in \Omega'$. Constructing a suitable 2D RBF warp requires only minor changes to the 3D formulation (1). In contrast to the 3D case, the 2D biharmonic basis function $\varphi_2(r) = r^2 \log(r)$ is differentiable at the center and therefore smooth enough for our scenario. The polynomial part consists of the basis $\{\pi_1, \pi_2, \pi_3\} = \{x, y, 1\}$, and the coefficients $\mathbf{w}_j, \mathbf{q}_k$ are two-dimensional. With these changes, and replacing $\mathbf{s}_i$ by $\mathbf{c}_i$, we solve a linear system analogous to (3) for computing the RBF warp $\mathbf{d}_k$. After performing the parameter warp $(u', v') = (u, v) + \mathbf{d}_k(u, v)$ we compute the morphed surface node as $\mathbf{s}'_i = \mathbf{f}'_k(u', v')$.

This method is easy to compute and produces high quality surface warps of minimal parametric distortion, as shown in Figure 3, which compares our surface morphing to that of [35]. Thanks to its meshless nature, we can apply our method to all kinds of faces, such as simple non-trimmed rectangular faces, trimmed faces with curved boundaries, as well as faces with trimmed holes (see Bore and Pipe examples in the next section).

5 Evaluation of Morphing Quality

In this section, we compare our mesh morphing technique to the results recently published in [35]. The examples include meshes of varying topology, including structured and unstructured hexahedral as well as tetrahedral meshes. The complexity of the models ranges from ∼10-15k vertices for the Bore and Pipe models up to ∼130k vertices for the Courier model. The Canister model used in [35] was not available to us. Since Staten et al. provide a detailed description of the different geometric parameters in the CAD models and how they adjust them we do not reproduce them here.

For the sake of a clear representation, we restrict our comparison to those methods that either delivered the best results (FEMWARP and LBWARP), or that have been recommended by the authors for sake of simplicity and efficiency (Simplex-linear). In order to ensure comparability of the results, we also measure element quality based on the scaled Jacobian as described in [21]. For our RBF volume morphing, we include results for both the original surface node locations from [35] (denoted RBF) as well as those obtained by our surface morphing (denoted RBF-S). The results denoted by RBF-IQR were computed using the incremental QR solver described in Section 7.

Following the benchmarks of [35], we investigate two different types of morphing: *relative* and *absolute* morphing. In the former case, we incrementally update the mesh from the initial design to the full parameter change. In the latter case, we directly warp the initial mesh to the corresponding parameter change. As in the benchmarks of [35] we use $N = 20$ steps for both types of morphing. In the following subsections, we present detailed results for the individual test cases. In order to give the reader an impression of where in the meshes the element quality becomes particularly low, we present selected cut-views of the morphed volume meshes in Figure 4. After performing an absolute morph to the full parameter change on both hexahedral and tetrahedral meshes, we highlight the worst 5% of the elements in red.

5.1 Bore Model

The change of parameters in the Bore model tests the ability of the different methods to deal with scaling and rotation. An example morph from the initial mesh to the full parameter change is shown in Figure 5. As can be seen from this figure, our method is on par with or better than the FEMWARP and LBWARP methods. It is important to note that the element inversion after 75% parameter change in case of the tetrahedral model is due to a defect in the morphed surface mesh of [35]. By using our more robust surface morph we are able to perform both the relative and the absolute morph up to a parameter change of 100% without any inverted elements.

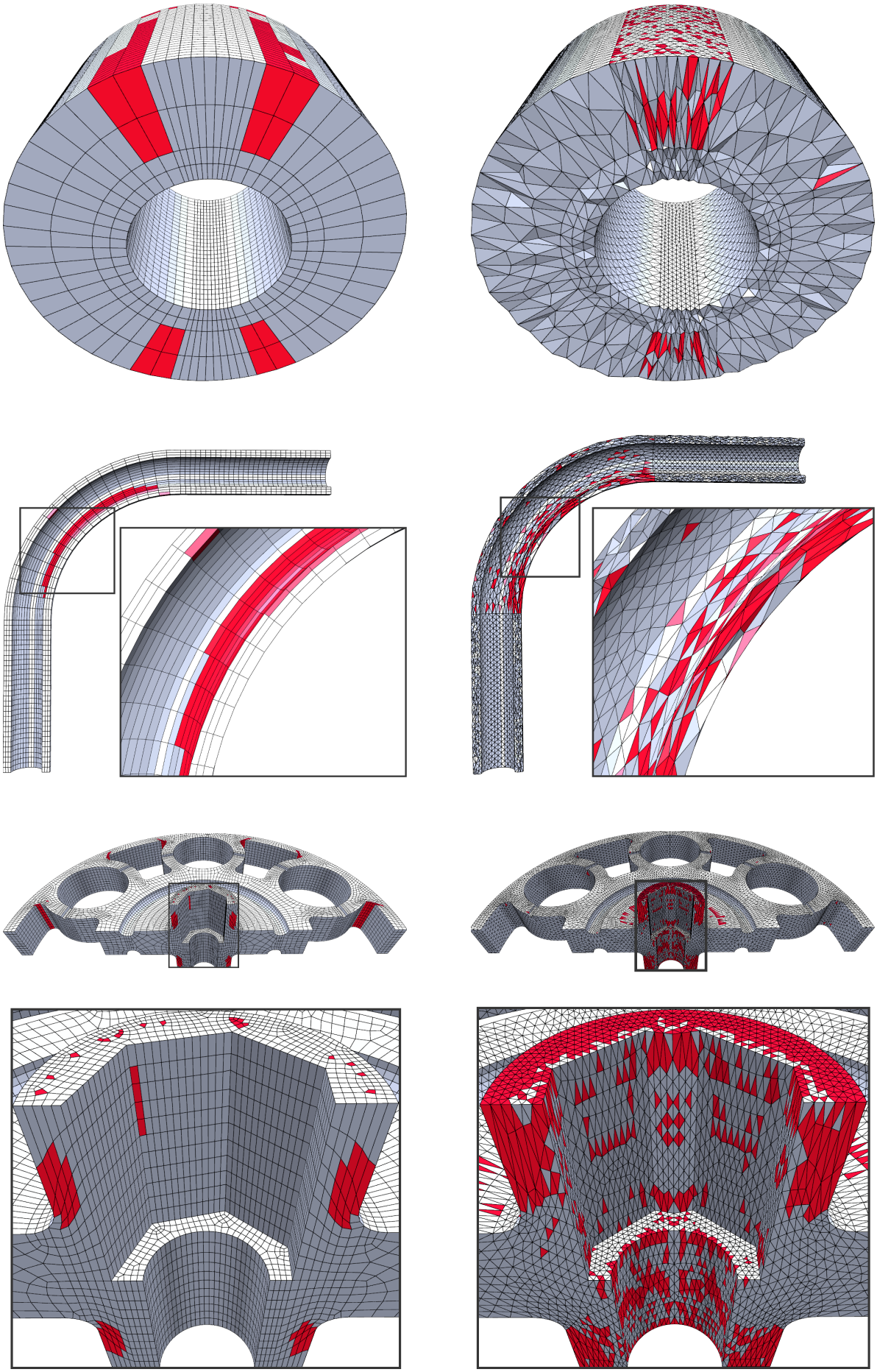
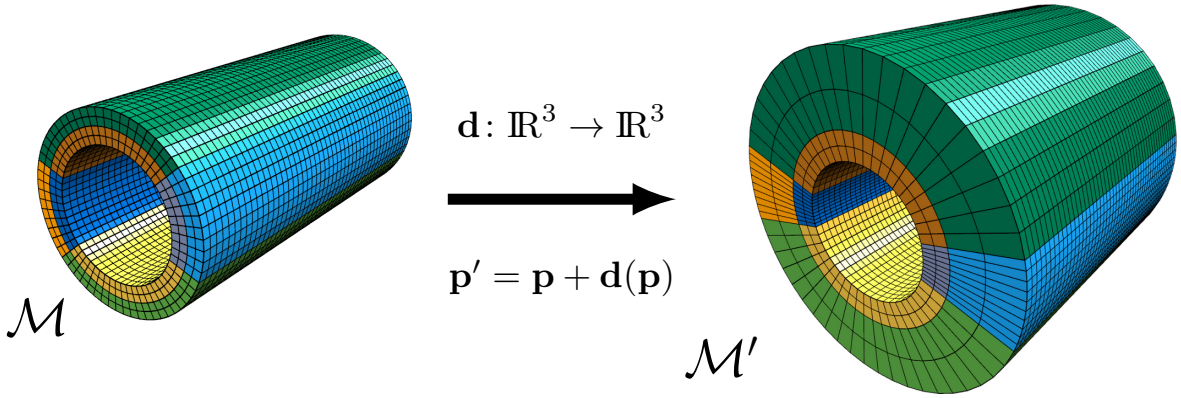
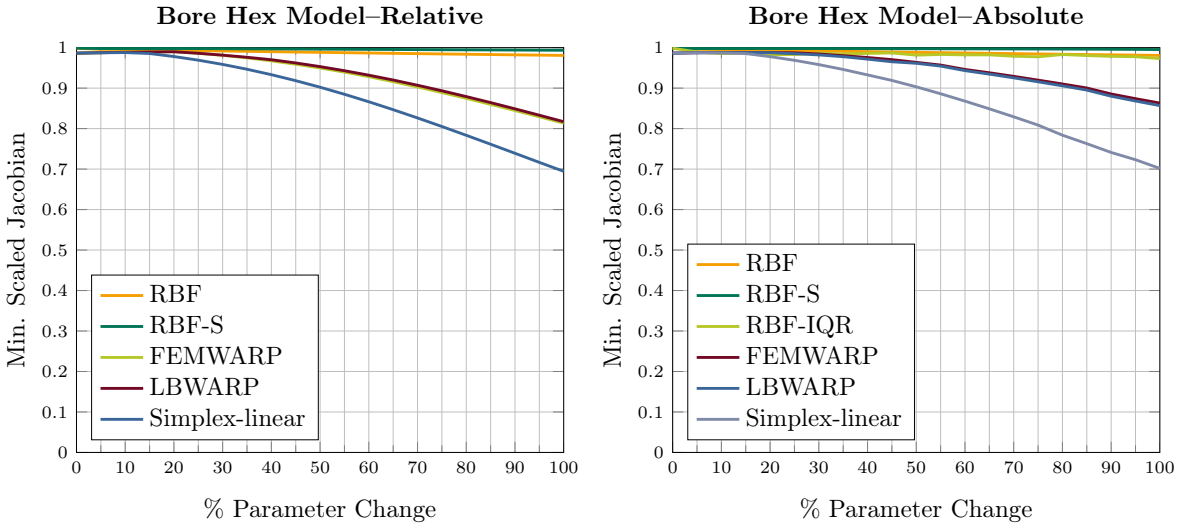
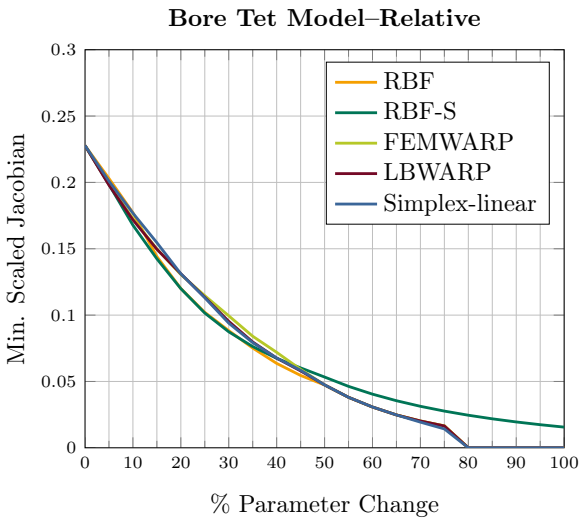


Fig. 4 Cut views of the hexahedral (left) and tetrahedral (right) meshes for the Bore (top), Pipe (middle), and Courier (bottom) models after performing an absolute morph to the full parameter change. We highlight the worst 5% of the elements in red.







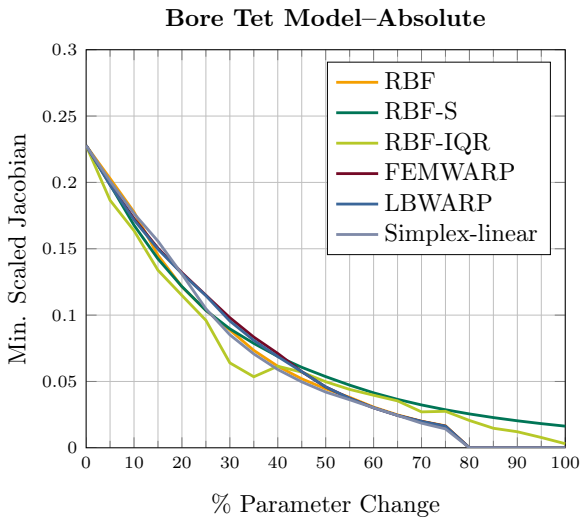


Fig. 5 Morphing results of the Bore model.

	Bore		Pipe		Courier
	Hex	Tet	Hex	Tet	Hex
biharmonic	0.985	0.015	0.92	0.018	0.09
triharmonic	0.995	0.016	0.95	0.019	0.045
quadharmonic	0.995	0.016	0.96	0.017	-0.99

Table 2 A comparison of RBF warps showing the min. scaled Jacobian after an absolute morph to the full parameter change.

5.2 Pipe Model

The change of parameters in the Pipe model tests the ability of the different methods to deal with nonlinear stretching. We present the initial and final shapes as well as detailed results in Figure 6. While our method provides superior results on the hex model, those on the tetrahedral model are comparable. However, in contrast to other methods ours does not result in inverted elements at 95% parameter change for the absolute morphing of the tetrahedral model. Again, the results obtained using our combined volume and surface morphing are superior.

5.3 Courier Model

The Courier model is the most complex model in our comparison. In contrast to previous examples, the hexahedral mesh of this model is an unstructured one. Especially in case of the absolute tetrahedral mesh morphing, all methods presented in [35] result in inverted elements as soon as reaching a change of parameter values of 65%. In contrast, our method results in inverted elements only after a parameter change of 75% (see the results in Figure 7). Unfortunately, due to a mismatch between CAD geometry and initial tetrahedral mesh, we could not apply our surface morphing method for the tetrahedral Courier model.

5.4 Comparison of RBF Warps

In Table 2 we report the resulting element quality for different RBF warps after performing an absolute morph to the full parameter change. In all but one case the triharmonic warp delivers better results than the biharmonic one. The higher-order quadharmonic RBF (using $\varphi_4(r) = r^5$) does not result in noteworthy improvements. Even worse, in case of the Courier hex model it even leads to inverted elements due to numerical instabilities. Investigating the condition number indeed reveals a drastic increase with the order of the basis function, being 2.3^6 for the biharmonic, 2.3^{11} for the triharmonic, and 6.3^{15} for the quadharmonic warp.

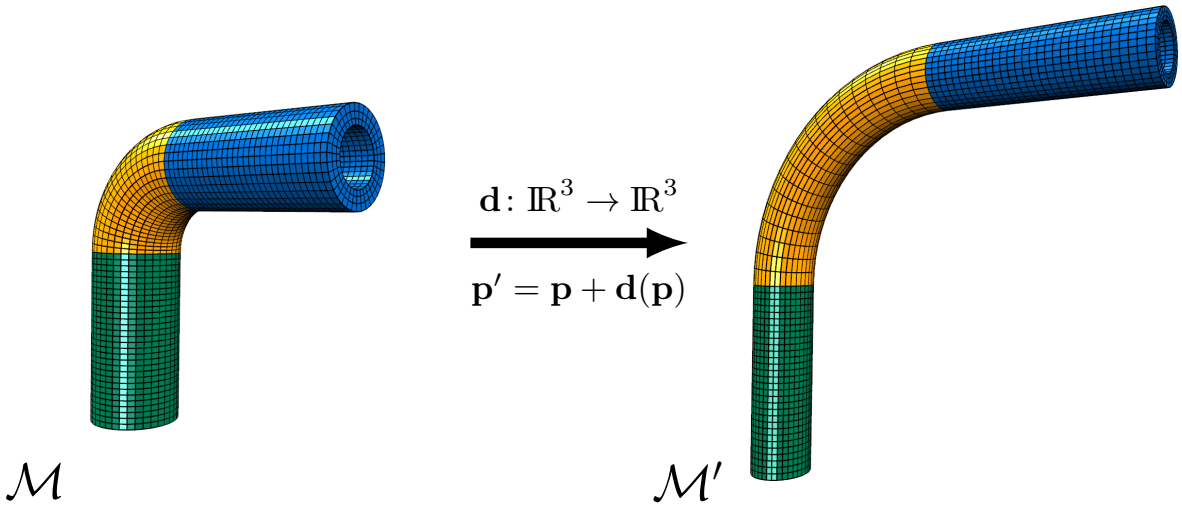
6 Inversion-free Morphing

In this section we investigate several approaches for preventing inverted elements in the deformed mesh. As already observed by Staten and colleagues, relative morphing tends to better preserve element quality compared to absolute morphing. Therefore, one could be inclined trying to avoid inverted elements by using smaller and smaller steps of relative morphing. However, within their approach they use the relative geometric parameters in the CAD model to generate the intermediate boundary nodes for the volume morphing. Therefore, preventing element inversions in a fully automatic manner is rather complicated in this approach. Another approach to improve the resulting element quality and to eventually prevent inversions is to use a more powerful nonlinear deformation method such as [32].

A simple and efficient approach for preventing inversions is to iteratively split the deformation. In [14] that a space deformation is guaranteed to be free of self-intersections if (i) the it has continuous first partial derivatives and (ii) the determinant of its Jacobian is larger than zero. The first criterion is naturally fulfilled by our smooth RBF warps. The second one is fulfilled if the displacements to be interpolated are sufficiently small. We therefore use the following procedure to prevent inversions: We initially perform the full deformation. If the deformation results in at least one inverted element, we uniformly split the deformation into n steps, where n is the current iteration. We repeat this process until no more inversions occur. We illustrate this approach schematically in Figure 8. We note that Shontz and Vavasis follow a similar procedure in the context of FEMWARP in [31].

The prevention of self-intersections and element inversions under deformation has also been subject to substantial research [2, 14, 15, 31]. A particularly powerful approach is the construction and integration of a smooth space-time vector field, which theoretically guarantees the absence of intersections and element inversions [2, 12, 24]. In the limit case of arbitrarily small displacements our splitting approach is roughly equivalent to vector field-based approaches. In contrast, however, our method avoids the increased computational costs for the construction of the space-time vector field [24], making our method more practical in the current volume mesh morphing scenario.

We finally emphasize that even a smooth, inversion-free space warp does not necessarily guarantee the absence of inverted elements. Applying the deformation to all mesh nodes eventually turns the smooth space warp into a piecewise linear C^0 per-element deformation. Therefore, a fundamental requirement for an inversion-free deformation is a mesh resolution sufficiently high to faithfully represent the deformation field. For a detailed investigation of element inversion and mesh discretization in the context of the FEMWARP technique we refer to Shontz and Vavasis [31].



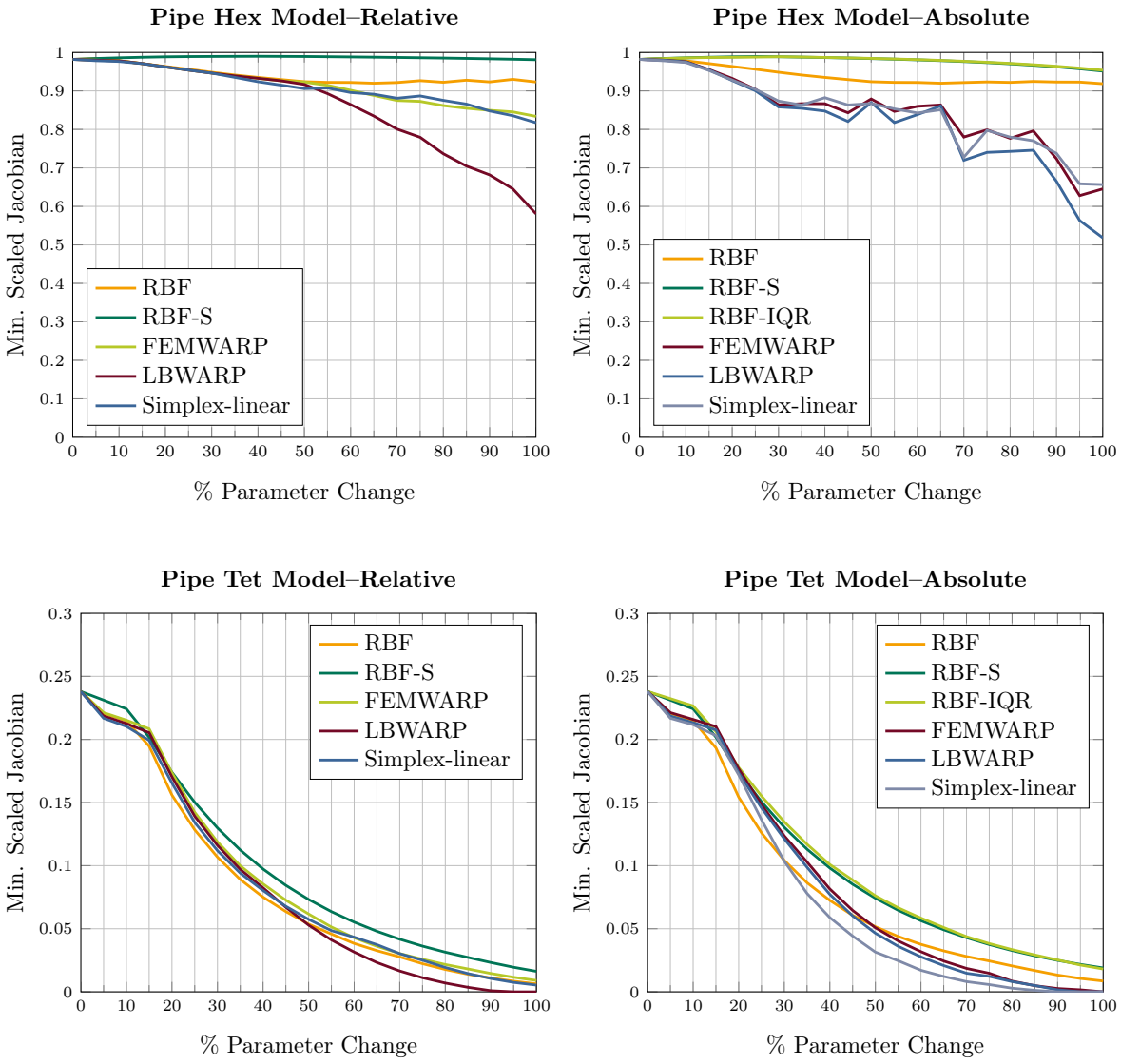


Fig. 6 Morphing results of the Pipe model.

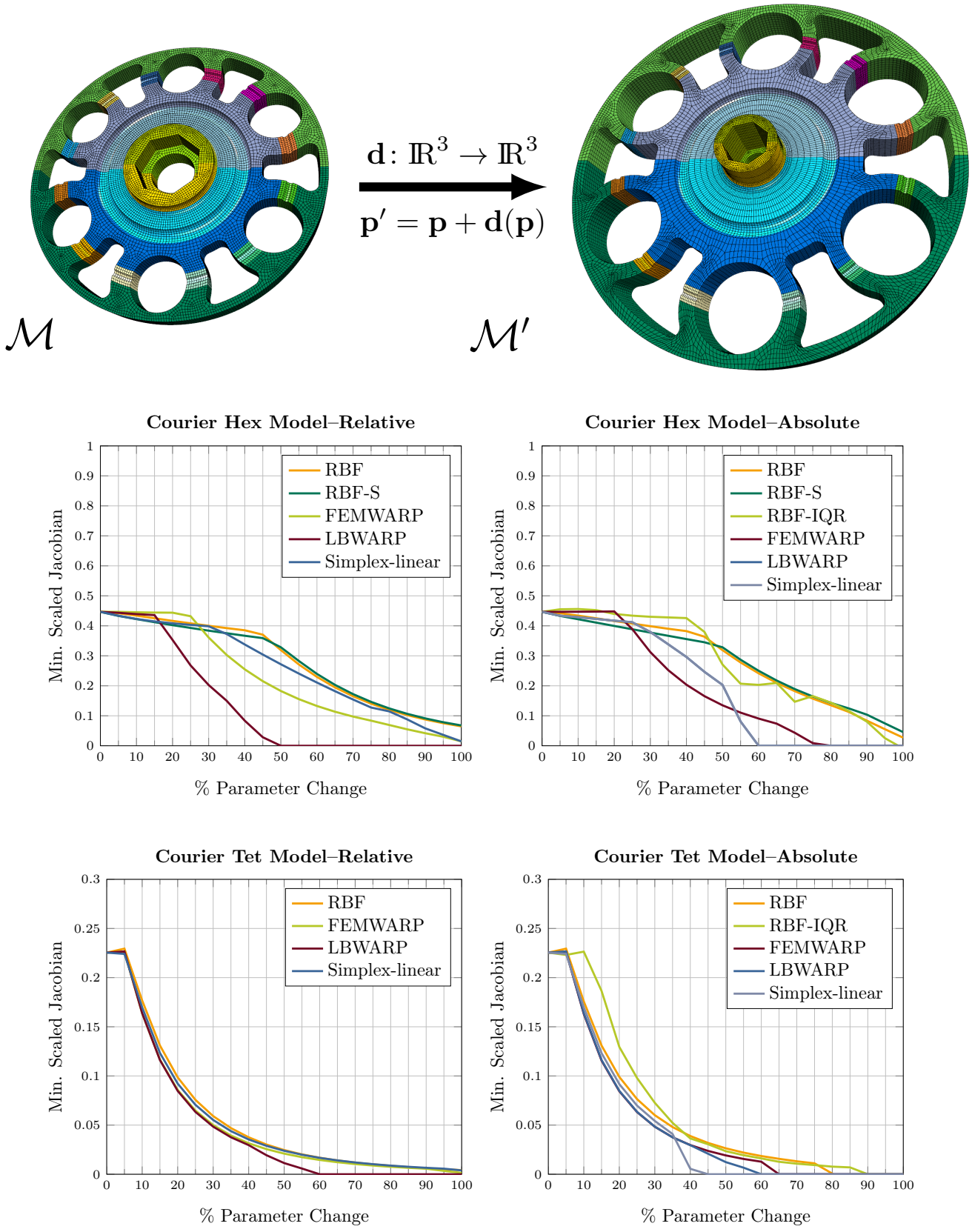


Fig. 7 Morphing results of the Courier model.

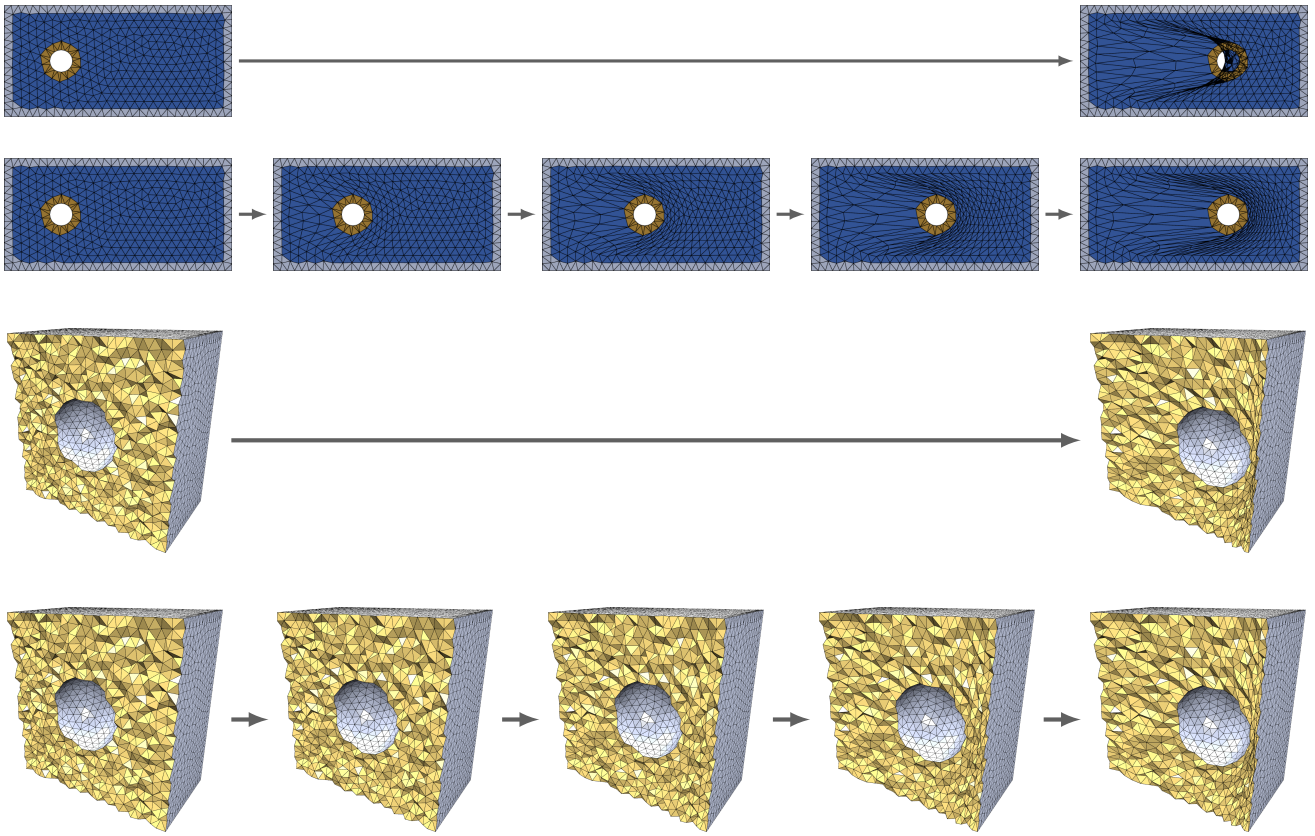
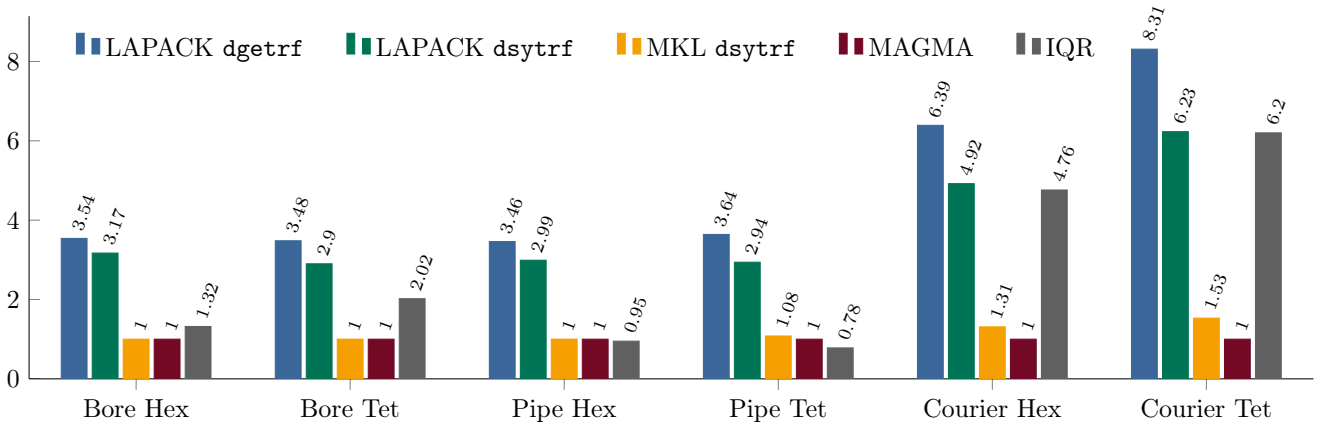


Fig. 8 Using splitting to prevent element inversion. In 2D (top) we displace the golden region, keep the gray region fixed, and deform the blue region. In 3D (bottom) we move a sphere within the tetrahedral mesh of a box. In both cases performing a large deformation in a single step leads to inverted elements while splitting the deformation into smaller steps leads to a mesh without inverted elements.



	Bore Hex	Bore Tet	Pipe Hex	Pipe Tet	Courier Hex	Courier Tet
LAPACK `dgetrf`	8.0	12.0	4.5	8.5	140	483
LAPACK `dsytrf`	7.1	10.0	3.9	6.9	108	362
MKL `dsytrf`	2.3	3.4	1.3	2.5	29	89
MAGMA	2.3	3.4	1.3	2.3	22	58
IQR	3.0	7.0	3.3	1.8	104	360

Table 3 Performance comparison of the RBF volume morphing using different solvers for the linear system (3), averaged over five runs. The table reports morphing times in seconds. The chart depicts performance differences relative to the GPU-based MAGMA solver.

7 Performance and Scalability

In this section we investigate the performance and the scalability of our method. As already noted in Section 3, the computationally most expensive part within our technique is the solution of the linear system (3) for the volume morphing. This linear system is dense due to the global support of the chosen radial basis functions $\varphi(r) = r^3$, resulting in an asymptotic complexity of $O(m^3)$ when using standard solvers for dense linear systems.

While there exist sophisticated techniques for efficiently solving RBF-based systems like (3), such as multipole expansion, multi-level approximation, or greedy center selection [8, 25, 41], this was not necessary in all our test cases. Since most CAD geometries $\mathcal{G}$ and their corresponding volume meshes $\mathcal{M}$ are constructed from multiple solid components, we can simply perform the volume morphing individually for each of these (reasonably small) components. In all our examples this could be done using a standard linear solver, e.g., the LU factorization of the LAPACK library [1].

Nevertheless, we investigate in the following how to (i) improve the performance by using efficient implementations of standard solvers and (ii) improve the scalability to larger models using an incremental least squares solver.

Since the linear system (3) is symmetric but not positive definite, efficient Cholesky-type solvers are not applicable, leaving us with solvers based on LU and LDL^T factorizations as the default choices. We compared four solver implementations:

- The general LU decomposition (`dgetrf`) of LAPACK,
- the LDL^T factorization for symmetric matrices (`dsytrf`) of LAPACK,
- the multi-core LDL^T decomposition (`dsytrf`) of the Intel Math Kernel Library (MKL) [18],
- the GPU-accelerated LU decomposition of MAGMA [38].

We performed all tests on a Dell T7500 workstation with an Intel Xeon E5645 2.4 GHz CPU and 18GB RAM running Ubuntu Linux 12.04 x86_64. We compiled all code with `gcc` 4.6.3, optimization turned on (using `-O3`) and debugging checks disabled (`-DNDEBUG`). In order to rule out caching and power saving issues, we averaged the timings over five morphing steps.

The results in Table 3 show that the performance differs significantly between solvers. The largest differences exist in case of the Courier model where the MAGMA-based solver is up to eight times faster than other implementations. The results for the MKL are almost identical to the MAGMA results for smaller models, but the difference increases with the model size. The comparison between LAPACK's general LU and symmetric LDL^T factorizations also shows considerable differences. Unfortunately, specialized symmetric factorizations were not available in MAGMA.

Despite the impressive performance improvements the scalability of all these methods is still limited by their time complexity $O(m^3)$ and memory consumption $O(m^2)$, preventing their use for high resolution meshes. One can observe, however, that even for densely tessellated models the geometric morphs are still rather simple and smooth, so that a moderate number of RBF kernels is sufficient to represent the deformation [6]. We can therefore compute the morph by using only a subset of the surface nodes $\mathbf{s}_j$ as centers $\mathbf{x}_j$. This turns the interpolation problem (3) into a least squares approximation problem, where the required number of centers depends on the complexity of the deformation only—instead of on the complexity of the mesh.

As an implementation of this concept we use our incremental QR solver (IQR) initially presented in [6]. This solver starts by fitting the polynomial term of (1) only and then incrementally adds more and more RBF kernels φ_j until the least squares error falls below a user-prescribed threshold. Below we sketch the main ideas of [6] for completeness.

Using just k basis functions (polynomial and RBFs) instead of the full set of $(m+4)$ basis functions corresponds to replacing the quadratic $(m+4) \times (m+4)$ linear system $\mathbf{AX} = \mathbf{B}$ of (3) by the reduced $(m+4) \times k$ system $\mathbf{A}_k\mathbf{X}_k = \mathbf{B}$, where $\mathbf{A}_k$ is composed from the k columns of $\mathbf{A}$ corresponding to the k selected basis functions, and $\mathbf{X}_k$ are their respective coefficients. This over-determined system can be solved robustly using the QR factorization:

$$\mathbf{A}_k = \mathbf{Q}_k\mathbf{R}_k, \quad \mathbf{X}_k = \mathbf{R}_k^{-1}\mathbf{Q}_k^T\mathbf{B}. \tag{4}$$

The main observation of [6] is that computing the QR factorization of the *full* matrix $\mathbf{A}$ iteratively processes column by column for $k = 1, \ldots, m+4$, and that iteration k basically computes $\mathbf{Q}_k$ and $\mathbf{R}_k$. In addition, the least squares error $\|\mathbf{A}_k\mathbf{X}_k - \mathbf{B}\|^2$ can be determined almost for free without actually having to compute $\mathbf{X}_k$. The IQR solver therefore works similar to a standard QR solver, but can stop as soon as at iteration k the first k columns of $\mathbf{A}$ yield a sufficiently accurate least squares solution. Since this algorithm simply chooses the first k columns of $\mathbf{A}$, a suitable re-ordering of the columns (i.e., of the corresponding basis functions) is performed in a pre-process. We use a farthest point center selection strategy, since this can be computed at negligible cost and guarantees a good matrix condition number [6].

The complexity of solving the least squares system (4) is $O(mk^2)$. Hence, in the worst case that all $m+4$ columns have to be used the complexity still is $O(m^3)$ as for all other solvers. Since the computational overhead compared to a standard QR solver is negligible, the performance is on par with standard (CPU-based) solvers even if the morph is complex and requires a large number of centers (see Courier example in Table 3). However, for simple deformations, such as the Bore and Pipe examples in Table 3, the incremental solver even outperforms the GPU-based MAGMA solver. We note

that the user-prescribed error might negatively influence the resulting element quality if it is not small enough, thereby offering a trade-off between performance and quality.

Comparing the performance of our RBF-based technique with those investigated by Staten and colleagues shows that our method is computationally more expensive. However, in all but one case our method allows to perform an absolute morph to the full parameter change without resulting in inverted elements. Other methods might only reach this goal by falling back to several steps of relative morphing, thereby becoming computationally more expensive than our approach.

8 Conclusions and Future Work

In this paper, we presented a simple and versatile method for high-quality mesh morphing of both surface and volume meshes using RBFs. The smoothness of our triharmonic RBF morphs leads to similar or superior element quality compared to all techniques evaluated in [35]. The implementation of our method is straightforward and essentially requires setting up the linear system (3) and solving it using a standard solver. Therefore, it can be considered significantly easier to implement than the LBWARP method. While being similarly straightforward to implement as our method, the FEMWARP technique has to be derived explicitly for each element type. In contrast, our RBF morphs are highly flexible, since the same unified code can morph arbitrary geometries in arbitrary dimensions.

We also investigated solutions to potential issues arising in mesh morphing for design optimization. We presented a simple and efficient technique for constructing inversion-free deformations. Furthermore, we have illustrated how the performance of our method can be drastically improved by employing efficient GPU-based linear solvers or incremental least squares solvers. Another direction for performance improvements is parallelization. The basic requirement for a parallel execution of our method is a decomposition of the mesh into separate components with matching interfaces, which is widespread task in parallel computing.

A promising direction for future work is to compare the results of our globally supported RBFs with those of compactly supported basis functions [41]. While these basis functions have the advantage that the linear systems become sparse and hence allows for more efficient storage and solvers, one has to explicitly incorporate the minimization of a smoothness energy like (2) in order to obtain high-quality deformations. As shown in [40], such an approach is capable to produce high-quality results while being computationally more efficient than methods based on globally supported basis functions.

Acknowledgments

The authors kindly thank Matthew Staten from Sandia National Laboratories for providing us with the Bore, Pipe, and Courier models from [35]. We also thank the anonymous reviewers for their useful comments and suggestions. Daniel Sieger gratefully acknowledges the financial support from Honda Research Institute Europe (HRI-EU). Mario Botsch is supported by the German National Research Foundation (DFG CoE 277: CITEC).

References

1. Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Croz, J.D., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.: LAPACK Users' Guide, third edn. SIAM, Philadelphia, PA (1999)
2. Angelidis, A., Cani, M.P., Wyvill, G., King, S.: Swirling-Sweepers: Constant volume modeling. Graph Models **68**(4), 324–32 (2006)
3. Baker, T.J.: Mesh movement and metamorphosis. In: Proceedings of the 10th International Meshing Roundtable, pp. 387–396 (2001)
4. Bechmann, D.: Space deformation models survey. Comput Graph **18**(4), 571 – 586 (1994)
5. de Boer, A., van der Schoot, M., Bijl, H.: Mesh deformation based on radial basis function interpolation. Comput Struct **85**, 784–795 (2007)
6. Botsch, M., Kobbelt, L.: Real-time shape editing using radial basis functions. Comput Graph Forum **24**(3), 611–621 (2005)
7. Brewer, M., Diachin, L.F., Knupp, P., Leurent, T., Melander, D.: The Mesquite mesh quality improvement toolkit. In: Proceedings of the 12th International Meshing Roundtable, pp. 239–250 (2003)
8. Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C., Evans, T.R.: Reconstruction and representation of 3D objects with radial basis functions. In: Proceedings of ACM SIGGRAPH, pp. 67–76. ACM, New York (2001)
9. Duchon, J.: Spline minimizing rotation-invariant seminorms in Sobolev spaces. In: W. Schempp, K. Zeller (eds.) Constructive Theory of Functions of Several Variables, no. 571 in Lecture Notes in Mathematics, pp. 85–100. Springer Verlag, Berlin (1977)
10. Fasshauer, G.E.: Meshfree approximation methods with MATLAB. World Scientific Publishing (2007)
11. Floater, M.S., Kos, G., Reimers, M.: Mean value coordinates in 3D. Comput Aided Geom D **22**, 623–631 (2005)
12. von Funck, W., Theisel, H., Seidel, H.P.: Vector field-based shape deformations. ACM T Graphic **25**(3), 1118–1125 (2006)

13. Gain, J., Bechmann, D.: A survey of spatial deformation from a user-centered perspective. ACM T Graphic **27**, 107:1–107:21 (2008)
14. Gain, J., Dodgson, N.: Preventing self-intersection under free-form deformation. IEEE T Vis Comput Gr **7**(4), 289 –298 (2001)
15. Harmon, D., Panozzo, D., Sorkine, O., Zorin, D.: Interference aware geometric modeling. ACM T Graphic **30**(6), 137:1–137:10 (2011)
16. Helenbrook, B.T.: Mesh deformation using the biharmonic operator. Int J Numer Meth Eng **56**, 1007–1021 (2003)
17. Hormann, K., Sukumar, N.: Maximum entropy coordinates for arbitrary polytopes. Comput Graph Forum **27**(5), 1513–1520 (2008)
18. Intel: Intel Math Kernel Library version 11.0. http://software.intel.com/en-us/intel-mkl (2013)
19. Jakobsson, S., Amoignon, O.: Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization. Comput Fluids **36**(6), 1119–1136 (2007)
20. Joshi, P., Meyer, M., DeRose, T., Green, B., Sanocki, T.: Harmonic coordinates for character articulation. ACM T Graphic **26**(3) (2007)
21. Knupp, P.: Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. Part I. Int J Numer Meth Eng **48**(3), 401–420 (2000)
22. Knupp, P.: Updating meshes on deforming domains: An application of the target-matrix paradigm. Commun Num Method Eng **24**(6), 467–476 (2008)
23. Martin, S., Kaufmann, P., Botsch, M., Wicke, M., Gross, M.: Polyhedral finite elements using harmonic basis functions. Comput Graph Forum **27**(5), 1521–1529 (2008)
24. Martinez Esturo, J., Rössl, C., Fröhlich, S., Botsch, M., Theisel, H.: Pose correction by space-time integration. In: Proceedings of Vision, Modeling, Visualization, pp. 33–40 (2011)
25. Michler, A.K.: Aircraft control surface deflection using RBF-based mesh deformation. Int J Numer Meth Eng **88**(10), 986–1007 (2011)
26. Open CASCADE: Open CASCADE Technology, 3D modeling & numerical simulation (2012). URL `http://www.opencascade.org/`
27. Samareh, J.A.: A survey of shape parameterization techniques. Tech. Rep. NASA/CP-1999-209136/PT1, NASA Langley Research Center (1999)
28. Samet, H.: The Design and Analysis of Spatial Data Structures. Addison Wesley, Reading, MA (1994)
29. Sederberg, T.W., Parry, S.R.: Free-form deformation of solid geometric models. In: Proceedings of ACM SIGGRAPH, pp. 151–159. ACM, New York (1986)
30. Shontz, S.M., Vavasis, S.A.: A mesh warping algorithm based on weighted Laplacian smoothing. In: Proceedings of the 12th International Meshing Roundtable, pp. 147–158 (2003)
31. 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 Numer Math **50**(4), 863–884 (2010)
32. Shontz, S.M., Vavasis, S.A.: A robust solution procedure for hyperelastic solids with large boundary deformation. Eng Comput **28**(2), 135–147 (2012)
33. Sibson, R.: Interpreting Multivariate Data, vol. 21, chap. A brief description of natural neighbor interpolation. John Wiley & Sons (1981)
34. Staten, M.L., Canann, S.A., Owen, S.J.: BMSweep: Locating interior nodes during sweeping. Eng Comput **15**(3), 212–218 (1999)
35. Staten, M.L., Owen, S.J., Shontz, S.M., Salinger, A.G., Coffey, T.S.: A comparison of mesh morphing methods for 3D shape optimization. In: Proceedings of the 20th International Meshing Roundtable, pp. 293–311 (2011)
36. Sukumar, N.: Construction of polygonal interpolants: A maximum entropy approach. Int J Numer Meth Eng **61**(12), 2159–2181 (2004)
37. Sukumar, N., Malsch, E.A.: Recent advances in the construction of polygonal finite element interpolants. Arch Comput Method E **13**(1), 129–163 (2006)
38. Tomov, S., Nath, R., Ltaief, H., Dongarra, J.: Dense linear algebra solvers for multicore with GPU accelerators. In: Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on, pp. 1–8. IEEE (2010)
39. Wachspress, E.L.: A Rational Finite Element Basis. Academic Press (1975)
40. Walder, C., Schölkopf, B., Chapelle, O.: Implicit surface modelling with a globally regularised basis of compact support. Comput Graph Forum **25**(3), 635–644 (2006)
41. Wendland, H.: Scattered Data Approximation. Cambridge University Press, Cambridge, UK (2005)