Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 00 (2014) 000–000

Procedia Engineering

www.elsevier.com/locate/procedia

23rd International Meshing Roundtable (IMR23)

Constrained Space Deformation for Design Optimization

Daniel Sieger[a,*], Stefan Menzel[b], Mario Botsch[a]

[a]*Bielefeld University, Computer Graphics & Geometry Group, Postfach 100 131, D-33501 Bielefeld, Germany*
[b]*Honda Research Institute Europe GmbH, Carl-Legien-Straße 30, D-63073 Offenbach/Main, Germany*

Abstract

We present a novel shape deformation method for its use in design optimization tasks. Our space deformation technique based on moving least squares approximation improves upon existing approaches in crucial aspects: It offers the same level of modeling flexibility as surface-based deformations, but it is independent of the underlying geometry representation and therefore highly robust against defects in the input data. It overcomes the scalability limitations of existing space deformation techniques based on globally supported triharmonic radial basis functions while providing the same high level of deformation quality. Finally, unlike existing space deformation approaches, our technique directly incorporates geometric constraints—such as preservation of critical feature lines, circular couplings, or planar construction parts—into the deformation, thereby fostering the exploration of more favorable and producible shape variations during the design optimization process.

Keywords: mesh deformation; geometric constraints; moving least squares; design optimization

1. Introduction

Design optimization is a key component of the product development process of automotive industry, aircraft construction, and naval architecture. The overall goal is to discover alternative designs with improved physical or aesthetic properties. The development process typically starts with the creation of an initial prototype using computer aided design (CAD) tools. Subsequent steps generate a polygon surface mesh from the CAD model as well as a volumetric simulation mesh in order to evaluate the physical performance of the design, e.g., based on aerodynamics or structural mechanics simulations. Design variations are then created based on physical performance during simulation.

A challenging task within the optimization process is to develop effective means to create alternate designs. Changing the CAD model directly is typically prohibitive, since repeated surface and volume meshing is highly time-consuming, and for complex geometries might even require manual interaction by an expert. An alternative is to use *shape deformation techniques* to adapt both the surface and the volume mesh of the initial design prototype directly. This way, the design optimization can be performed in a fully automatic and parallel manner, which is of particular importance 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.

* Corresponding author. Tel.: +49-521-106-12145 ; fax: +49-521-106-12448.
E-mail address: dsieger@techfak.uni-bielefeld.de

Even though shape deformation techniques drastically simplify the creation of design variations, their successful application within practical design optimization tasks comes with a number of challenges:

1. Severe defects in the input data or varying element types in the simulation's surface and volume meshes prohibit surface-based or mesh-based deformation techniques and typically require space deformation methods.
2. The results obtained from the deformation might not be of sufficient quality, as illustrated in the comparisons of Staten and colleagues [1] and our recent investigations [2,3], which suggest the use of triharmonic radial basis functions (RBFs) for high quality shape deformations.
3. In terms of performance the method might not scale to complex optimization scenarios. For example, the RBFs proposed in [2,3] offer high deformation quality due to their built-in minimization of fairness energies, but the involved dense linear system restrict the method to moderately sized problems.
4. The method might not offer a sufficient level of modeling flexibility, e.g., to simulate inhomogeneous material behavior during deformation. RBFs, which implicitly minimize bending-type energies, fail to simulate stretching-dominant materials.
5. Critical features required for functionality and realization of design prototypes might not be properly preserved during deformation. The typical solution to this wide-spread problem in design optimization is to incorporate additional penalty terms into the optimization process. This strategy, however, results in costly creation and evaluation of unfavorable design variations.

In this paper, we present a shape deformation technique based on moving least squares (MLS) discretization [4] that improves upon existing approaches in virtually all of the above aspects: Since we follow a space deformation approach our method is independent of the underlying geometry representation and highly robust towards defects in the input data. In terms of deformation quality, our method is competitive to global triharmonic RBFs. We drastically improve on the latter in terms of scalability, having to solve sparse linear systems only. By incorporating explicit stretching and bending energies, we offer the same level of modeling flexibility as surface-based methods. Finally, our technique directly incorporates geometric constraints into the deformation, thereby fostering the exploration of more meaningful and producible shape variations during the design optimization process.

2. Related Work

In this paper, we are concerned with high-quality shape deformation techniques for their use in design optimization tasks. Such techniques typically incorporate the minimization of physically-inspired energies in order to perform smooth and physically plausible deformations, as exemplified by *mesh-based variational methods* computing smooth harmonic or biharmonic deformations by solving Laplacian or bi-Laplacian systems [5,6]. The finite element-based FEMWARP technique [5,7], which computes a harmonic deformation, was generalized from tetrahedra to hexahedra in [1], and turned out to be highly successful in comparison to other methods. While the deformations 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), and they depend on the element quality of the underlying mesh.

In contrast, *meshless deformation techniques* avoid these 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 the mesh. Spline-based free-form deformation (FFD) techniques [8] have been widely used in both the graphics and engineering communities [9]. After its initial conception numerous extensions have been proposed, and we refer the reader to the survey papers [10–12] for a more comprehensive overview. However, spline-based FFD does not offer the same degree of smoothness as harmonic or biharmonic deformations, and it requires a rather tedious control lattice setup, as we investigate in detail in [3].

In [2] we successfully combined the advantages of meshless approaches and mesh-based variational methods by employing *radial basis functions* (RBFs) for mesh deformation. RBF space warps can handle arbitrary polyhedral meshes and offer a degree of smoothness comparable to mesh-based variational techniques. However, an inherent limitation of this approach is that the implicit energy minimization is built-in by construction and therefore offers no choice in terms of energy minimization. Furthermore, due to the global support of their basis functions, the resulting linear systems are dense and therefore limited in terms of scalability.

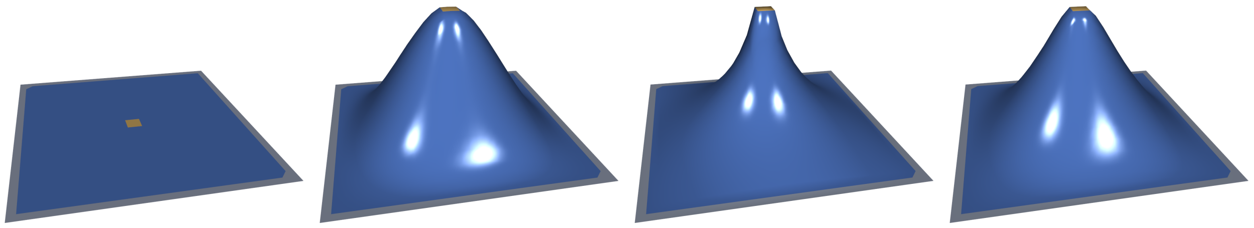

Fig. 1. Handle-based surface deformation of a plane (4000 vertices). From left to right: Undeformed model, minimization of pure bending, pure stretching, and a mixture thereof with parameters $w_b = 0.6$ and $w_s = 1.0$. We choose $w_f = 100$ in order to ensure prescribed handle and fixed constraints are satisfied.

In this paper, we propose to overcome these limitations by employing *moving least squares* (MLS) methods [4,13] for mesh deformation. These techniques have been successfully used in meshless physics simulation and computer animation, and offer the same high level of deformation quality as RBF warps, but they also come with increased flexibility with regards to energy minimization. Furthermore, the linear systems resulting from MLS-based discretization are generally sparse and therefore offer a drastically increased level of scalability compared to approaches based on globally supported RBFs.

A rather recent innovation in the development of shape deformation techniques is the integration of additional constraints into the deformation [14], as exemplified by the feature-preserving surface deformation technique of Masuda [15], or by the iWires system [16] for deformation of man-made objects. More recently, the latter approach was generalized to component-wise controllers [17], and the work of Habbecke [18] presents an efficient technique for the linear analysis of non-linear constraint in geometric modeling systems. However, all of the above methods are inherently surface-based. Therefore, their applicability to design optimization tasks is rather limited. A notable exception in this regard is the projection-based Shape-Up technique of Bouaziz and colleagues [19], since it allows for general constraints on arbitrary geometric data sets. We integrate this approach for constraint preservation into our MLS-based space deformation technique, thereby fostering the creation of more feasible design variations during design optimization.

In the following sections we describe our deformation technique in detail, going from the fundamentals to the specifics. We begin with a description of a general deformation model suitable for design optimization (Section 3). We describe our approach to space deformation based on subspace techniques in Section 4, where we also analyze and compare different choices of subspaces. In order to make our technique fully independent from the underlying geometry representation, we describe a spatial discretization of deformation energies in Section 5. Finally, we describe how to integrate constraints into the deformation in Section 6.

3. Mesh-Based Surface Deformation

In this section, we describe a mesh-based deformation model that is suitable for a design optimization framework. Since the most common targets for design optimization are sheet metal surfaces, such as car bodies, aircraft wings, or ship hulls, we concentrate on *surface deformation* models first. The resulting model will then be extended to *subspace* surface deformations and true *volumetric space deformations* in the following sections.

The shape deformation will be controlled by an interface that specifies displacements for certain surface regions. In a design optimization context, we propose the use of a *direct manipulation* interface, where the user—being either a human designer or an optimization algorithm—directly manipulates certain regions of the surface mesh. In contrast to, e.g., the control point metaphor of lattice-based freeform deformation (FFD) [8], direct manipulation interfaces are preferable for design optimization, since the direct coupling between optimization parameters and the effect on the design variation leads to improved convergence rates [20,21].

Furthermore, we employ the so-called *handle metaphor* [22], where we distinguish three types of surface regions on the mesh: The handle region $\mathcal{H}$ is directly displaced by the user. The fixed region $\mathcal{F}$ stays in place. The deformable region $\mathcal{D}$ is updated according to the physical deformation method while satisfying the Dirichlet constraints given by $\mathcal{H}$ and $\mathcal{F}$. An example of this modeling metaphor is given in Figure 1, with the handle region in gold, the fixed region in gray, and the deformable region in blue.

The deformable region $\mathcal{D}$ should behave in a physically-plausible manner, i.e., it should deform like a thin shell based on stretching and bending energies. The deformations occurring in design optimization tasks typically are rather small. Therefore, a linear deformation model will be sufficient, where stretching and bending are measured by first and second order partial derivatives of the displacement function $\mathbf{d}$, respectively.

In the continuous setting, the deformation $\mathbf{d}\colon \mathcal{S} \to \mathbb{R}^3$ of a surface $\mathcal{S}$ can be computed by minimizing the energy functional

$$E_{\text{shell}}[\mathbf{d}] = w_s \, E_{\text{stretch}}[\mathbf{d}] + w_b \, E_{\text{bend}}[\mathbf{d}] + w_f \, E_{\text{fix}}[\mathbf{d}], \tag{1}$$

consisting of weighted energy contributions for bending, stretching, and constraint deviation [23]:

$$E_{\text{stretch}}[\mathbf{d}] = \int_{\mathcal{D}} \|\nabla \mathbf{d}(\mathbf{x})\|^2 \, \mathrm{d}\mathbf{x}, \tag{2}$$

$$E_{\text{bend}}[\mathbf{d}] = \int_{\mathcal{D}} \|\Delta \mathbf{d}(\mathbf{x})\|^2 \, \mathrm{d}\mathbf{x}, \tag{3}$$

$$E_{\text{fix}}[\mathbf{d}] = \int_{\mathcal{H}\cup\mathcal{F}} \left\|\mathbf{d}(\mathbf{x}) - \bar{\mathbf{d}}(\mathbf{x})\right\|^2 \, \mathrm{d}\mathbf{x}, \tag{4}$$

where $\nabla\mathbf{d}$ denotes the Jacobian of $\mathbf{d}$, $\Delta\mathbf{d} = \nabla \cdot \nabla\mathbf{d}$ its Laplacian, $\|\cdot\|$ the Frobenius matrix norm or the Euclidean vector norm, and $\bar{\mathbf{d}}$ the prescribed Dirichlet constraints for the fixed and handle regions.

If we assume that the surface $\mathcal{S}$ is discretized by a proper triangle mesh $\mathcal{M}$ (non-degenerate triangles, one single two-manifold component), then the most flexible discretization of the above thin shell deformation is one whose degrees of freedom are the individual vertex positions $\mathbf{x}_1, \ldots, \mathbf{x}_n$, or the vertex displacements $\mathbf{d}_1, \ldots, \mathbf{d}_n$:

$$\mathbf{d}_h(\mathbf{x}) \;=\; \sum_{i=1}^{n} \mathbf{d}_i \psi_i(\mathbf{x}), \tag{5}$$

where ψ_i are the piecewise linear shape functions on the triangulation $\mathcal{M}$. Based on this discretization we can approximate the above energies [23,24] as

$$E_{\text{stretch}}[\mathbf{d}_h] = \sum_{\mathbf{x}_i \in \mathcal{D}} A_i \, \|\nabla \mathbf{d}_i\|^2 \,, \tag{6}$$

$$E_{\text{bend}}[\mathbf{d}_h] = \sum_{\mathbf{x}_i \in \mathcal{D}} A_i \, \|\Delta \mathbf{d}_i\|^2 \,, \tag{7}$$

$$E_{\text{fix}}[\mathbf{d}_h] = \sum_{\mathbf{x}_i \in \mathcal{H}\cup\mathcal{F}} A_i \left\|\mathbf{d}_i - \bar{\mathbf{d}}_i\right\|^2 , \tag{8}$$

where A_i denotes the Voronoi area of vertex i, and where we use the well-established discrete differential operators proposed in Meyer et al. [25]. These allow to write the discrete gradient $\nabla\mathbf{d}_i$ and discrete Laplacian $\Delta\mathbf{d}_i$ as a linear combination of neighboring vertices.

For implementation convenience and easier extensibility in the following sections, we write the discrete shell energy (6)–(8) as

$$E_{\text{shell}}[\mathbf{d}_h] \;=\; w_s \, \|\mathbf{G}\mathbf{d}\|^2 + w_b \, \|\mathbf{L}\mathbf{d}\|^2 + w_f \, \left\|\mathbf{F}(\mathbf{d} - \bar{\mathbf{d}})\right\|^2 , \tag{9}$$

where $\mathbf{d} = (\mathbf{d}_1^T, \ldots, \mathbf{d}_n^T)^T$ is the $(n \times 3)$ matrix of per-vertex displacements, and $\mathbf{G}$ and $\mathbf{L}$ are gradient and Laplace matrices containing the required cotangent weights in each row and having their rows weighted by $\sqrt{A_i}$, respectively (see [23,24] for details). $\mathbf{F}$ is a diagonal matrix with $\mathbf{F}_{i,i} = \sqrt{A_i}$ if $\mathbf{x}_i \in \mathcal{F} \cup \mathcal{H}$ and $\mathbf{F}_{i,i} = 0$ otherwise. The minimization of the shell energy (9) then requires to solve the normal equations of the linear least squares system

$$\left(w_s \mathbf{G}^T\mathbf{G} + w_b \mathbf{L}^T\mathbf{L} + w_f \mathbf{F}^T\mathbf{F}\right) \mathbf{d} \;=\; w_f \mathbf{F}^T\mathbf{F}\bar{\mathbf{d}}, \tag{10}$$

which can be done efficiently using a sparse Cholesky solver [26]. In order to ensure proper satisfaction of the Dirichlet boundary constraints, we typically choose w_f to be one or two orders of magnitude larger than the smoothness weights w_s and w_b. This mesh-based surface deformation approach, depicted in Figure 1, is our ground truth technique, which we try to reproduce using (more robust and more general) space deformation methods.

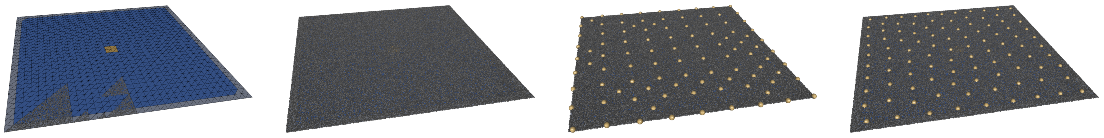
Fig. 2. Surface sampling: Incremental random sampling of each mesh face (left, samples in gray). This step finishes when the complete surface is densely covered with random samples (center left). From the dense samples we select a farthest point subset (center right, subset in gold). Then we perform Lloyd relaxation on this subset and use these points as RBF centers/MLS samples (right).

4. Subspace Surface Deformation

The deformation model described in the previous section offers high flexibility, since it uses the degrees of freedom of the mesh as degrees of freedom for the surface deformation. As motivated above, we are aiming at a *space deformation* approach, which deforms not only the given surface $\mathcal{S}$, but the whole space Ω embedding the object.

One advantage of space deformations is that they are *independent from the underlying geometry representation*, i.e., the same technique is applicable to point-sets, polygonal surface meshes, and polyhedral volume discretizations. This also allows to deform an existing volume mesh simultaneously with the surface, a feature of particularly importance for design optimization. Furthermore, complex designs often consist of multiple *disconnected components* that space deformations can naturally deform at once. Finally, the *robustness* against defects in the input data (e.g., degenerate triangles) is another compelling argument for space deformations, which are neither affected by the complexity nor by the quality of the input meshes.

In contrast to the previous section, we are looking for a deformation function $\mathbf{d}\colon \Omega \subset \mathbb{R}^3 \to \mathbb{R}^3$ that deforms the embedding space Ω around the model, while at the same time offering a comparable flexibility and deformation quality:

$$\mathbf{d}_h(\mathbf{x}) \;=\; \sum_{j=1}^{k} \mathbf{w}_j \varphi_j(\mathbf{x}),$$

where $\varphi_1, \ldots, \varphi_k$ are coarser shape functions ($k \ll n$) and $\mathbf{w}_j \in \mathbb{R}^3$ their coefficients.

In the following, we will analyze the modeling flexibility of different subspaces corresponding to different shape functions φ_j. In order to make the experiments more comparable to the mesh-based deformation, and to avoid any dependence on potentially insufficient numerical quadrature, we minimize the same vertex-based discrete shell energy (9), but replace the per-vertex displacements $\mathbf{d}_i$ by $\mathbf{d}_h(\mathbf{x}_i)$. We can then express the $n \times 3$ matrix $\mathbf{d}$ of vertex displacements in terms of the coefficients $\mathbf{w} = (\mathbf{w}_1^T, \ldots, \mathbf{w}_k^T)^T \in \mathbb{R}^{k \times 3}$ using a $n \times k$ subspace matrix $\mathbf{\Phi}$:

$$\mathbf{d} = \mathbf{\Phi}\mathbf{w} \quad \text{with} \quad \mathbf{\Phi}_{i,j} = \varphi_j(\mathbf{x}_i).$$

Inserting this into the discrete shell energy (9) leads to the $k \times k$ least squares system

$$\mathbf{\Phi}^T \left(w_s \mathbf{G}^T\mathbf{G} + w_b \mathbf{L}^T\mathbf{L} + w_f \mathbf{F}^T\mathbf{F} \right) \mathbf{\Phi}\,\mathbf{w} \;=\; \mathbf{\Phi}^T \left(w_f \mathbf{F}^T\mathbf{F}\bar{\mathbf{d}} \right). \tag{11}$$

In the following, we compare different choices for the shape functions φ_j. Motivated by our previous investigations [2,3], we focus on meshless, kernel-based discretization, and start with globally supported triharmonic RBFs, which however disqualify due to their high computational cost and limited scalability. We then analyze compactly supported Wendland RBFs [27] as well as moving least squares discretization [4].

For these kernel-based discretizations, we first need an efficient method to place the basis functions φ_j on the surface $\mathcal{S}$. To this end we employ a sampling strategy based on iterative Lloyd-relaxation [28], which we illustrate in Figure 2. Starting from the initial mesh, we create a dense sampling of the surface by computing random points within each polygonal face of the mesh. We then select a subset of k samples from the dense sampling by means of farthest point selection, i.e., we start with a random sample and iteratively add new samples based on maximizing the minimum distance between the new and the previously chosen samples. Finally, in order to maximize uniformity of the sampling we perform Lloyd-relaxation, i.e., we iteratively move each sample to the barycenter of the dense sample points being closest to the sample.

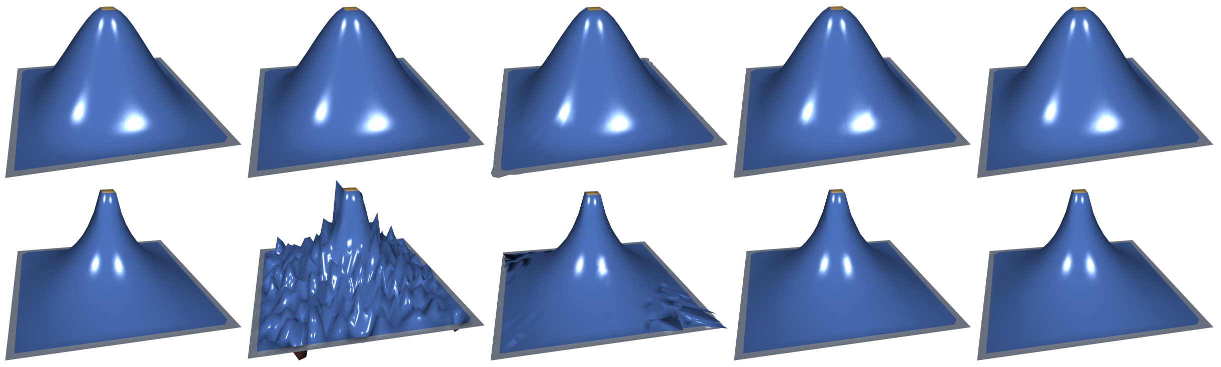

Fig. 3. Deformation of a plane (4000 vertices) minimizing bending (top row) and stretching (bottom row) energies. From left to right: Surface deformation, global RBFs, compact RBFs with medium ($s = 50$) and large ($s = 200$) support, MLS with small ($s = 5$) support. RBFs and MLS use 1000 shape functions.

Triharmonic RBFs. In our previous work [2,29], we successfully employed global triharmonic RBFs for high quality mesh morphing. Following this approach, we can construct a subspace by using shape functions

$$\varphi_j(\mathbf{x}) = \left\|\mathbf{x} - \mathbf{c}_j\right\|^3$$

located at centers $\mathbf{c}_j$. In Figure 3 we provide a comparison between the purely surface-based deformation and a subspace deformation using global triharmonic RBFs. While triharmonic RBFs work well for minimizing bending (which they do by construction), they fail to model stretching-dominant materials. Furthermore, due to their global support the matrix $\mathbf{\Phi}$ is dense, posing a serious limitation in terms of performance and scalability.

Wendland RBFs. An alternative to globally supported RBFs are compactly supported RBFs, such as the C^2-continuous Wendland functions

$$\varphi_j(\mathbf{x}) = \varphi\left(\left\|\mathbf{x} - \mathbf{c}_j\right\|\right) = \varphi(r) = \begin{cases} (1-r)^4(4r+1)\,, & r < \sigma\,, \\ 0\,, & \text{otherwise}\,. \end{cases}$$

The choice of the support radius σ is critical for the quality of the resulting subspace. In our implementation, we set support radii so that at least s shape functions φ_j cover each geometry point $\mathbf{x}_i$. As illustrated in Figure 3, the results with compact RBFs heavily depends on the chosen support radius. A moderately small radius of $s = 50$ leads to severe artifacts in the deformation. Only with an overly large radius of $s = 200$ the subspace produces results comparable to the surface deformation. In this case, however, the resulting linear system is not sufficiently sparse anymore, so that the compact RBFs are not an alternative in terms of scalability.

Moving Least Squares. An alternative to RBFs is the meshless moving least squares (MLS) approximation method. This allows for the construction of high quality and scalable subspaces, as we illustrate in Figure 3. In contrast to compact RBFs, MLS yield high quality results with a cover of $s = 5$. Since a reasonably comprehensive introduction to MLS is beyond the scope of this paper we refer the reader to the detailed introduction of [4] and only provide the required basic facts. The MLS shape functions $\varphi_j(\mathbf{x})$ are defined as

$$\varphi_j(\mathbf{x}) = \mathbf{p}(\mathbf{x})^T \mathbf{M}^{-1}(\mathbf{x})\mathbf{p}(\mathbf{c}_j) w(\mathbf{x} - \mathbf{c}_j)\,,$$

where $\mathbf{p}(\mathbf{x})$ is the vector of monomials $\mathbf{p}(x, y, z) = (1, x, y, z)^T$ and the spatially varying matrix $\mathbf{M}(\mathbf{x}) \in \mathbb{R}^{4\times4}$ is the so-called *moment matrix*

$$\mathbf{M}(\mathbf{x}) = \sum_{j=1}^{k} w(\mathbf{x} - \mathbf{c}_j)\mathbf{p}(\mathbf{c}_j)\mathbf{p}(\mathbf{c}_j)^T\,.$$

The weighting function $w(\cdot)$ is *compactly supported* and of sufficient smoothness. In our implementation, we use $w(r) = \frac{1}{2}\cos(r/\sigma \cdot \pi) + \frac{1}{2}$, with $w(r) = 0$ for $r > \sigma$. Unlike RBFs, the MLS basis functions do not have a simple analytic form, but require the inversion of the moment matrix for function evaluation. Note that the moment matrix becomes singular if the MLS samples $\mathbf{c}_j$ lie in the kernel of a linear polynomial (coplanar samples). We robustly handle this case by replacing the inverse $\mathbf{M}^{-1}$ by the pseudo-inverse $\mathbf{M}^{+}$ [30].

Even though MLS basis functions are significantly more expensive to evaluate than RBFs, this is not a problem in design optimization, since the MLS matrix $\mathbf{\Phi}$ can be pre-computed and re-used throughout the design optimization loop. More importantly, the MLS discretization scales well to complex models due to the sparsity of $\mathbf{\Phi}$, and the evaluation of φ_j is trivial to parallelize.

In summary, an MLS subspace yields a deformation that combines the strengths of the three approaches: the flexible energy minimization of mesh-based surface deformations, the high quality of global RBFs, and the scalability of compactly supported basis functions.

5. Volumetric Space Deformation

The previous section motivated the use of MLS shape functions as a flexible subspace for high quality deformation. However, the above comparisons—while using a space deformation function $\mathbf{d}\colon \mathbb{R}^3 \to \mathbb{R}^3$—still employed the stretching and bending energies based on a surface mesh (6)–(8). In this section we generalize the MLS deformation to true volumetric space deformations, which can then robustly process defect-laden, highly complex, and multi-component input meshes. To this end, we have to (i) place MLS kernels not only on the surface, but also in the embedding space Ω, and (ii) replace the vertex-based quadrature for integrating gradients and Laplacians over the surface $\mathcal{S}$ by a numerical cubature for integration over the embedding space Ω.

The volumetric sampling is a simple extension of the surface sampling shown in Figure 2. We first perform a dense sampling of the volume elements and then choose a subset by means for farthest point selection. We add this subset to the initial farthest point sampling of the surface $\mathcal{S}$ and then perform a combined Lloyd clustering of both the surface and volume samples, where we give a higher weight or density to the surface, leading to a slightly higher sampling density of the surface compared to the volume. As before, we denote the resulting MLS samples by $\mathbf{c}_j$, $j = 1, \ldots, k$.

We perform exactly the same sampling strategy to determine integration points $\mathbf{q}_i$, $i = 1, \ldots, N$, but make sure that the sampling density of the integration points $\mathbf{q}_i$ is sufficiently larger than the density of the MLS samples $\mathbf{c}_j$ (we use $N \approx 4k$).

Discretizing the stretching energy (2) in space amounts to evaluating the basis function derivatives at integration points:

$$E_{\text{stretch}}[\mathbf{d}_h] \;=\; \sum_{i=1}^{N} V_i \left\|\nabla \mathbf{d}(\mathbf{q}_i)\right\|^2 \;=\; \sum_{i=1}^{N} V_i \left\| \sum_{j=1}^{k} \mathbf{w}_j \nabla\varphi_j(\mathbf{q}_i) \right\|^2 \;=\; \left\|\mathbf{G}\mathbf{w}\right\|^2, \tag{12}$$

where V_i is the (approximate) Voronoi volume of integration point $\mathbf{q}_i$, and $\mathbf{G}$ is a $3N \times k$ gradient matrix with

$$\mathbf{G}_{3i,j} \;=\; \sqrt{V_i} \cdot \frac{\partial\, \varphi_j(\mathbf{q}_i)}{\partial\, x}, \quad \mathbf{G}_{3i+1,j} \;=\; \sqrt{V_i} \cdot \frac{\partial\, \varphi_j(\mathbf{q}_i)}{\partial\, y}, \quad \mathbf{G}_{3i+2,j} \;=\; \sqrt{V_i} \cdot \frac{\partial\, \varphi_j(\mathbf{q}_i)}{\partial\, z}.$$

Similarly, discretizing the bending energy (3) in space leads to

$$E_{\text{bend}}[\mathbf{d}_h] \;=\; \sum_{i=1}^{N} V_i \left\|\Delta \mathbf{d}(\mathbf{q}_i)\right\|^2 \;=\; \sum_{i=1}^{N} V_i \left\| \sum_{j=1}^{k} \mathbf{w}_j \Delta\varphi_j(\mathbf{q}_i) \right\|^2 \;=\; \left\|\mathbf{L}\mathbf{w}\right\|^2, \tag{13}$$

with a $N \times k$ Laplacian matrix $\mathbf{L}_{i,j} \;=\; \sqrt{V_i}\Delta\varphi_j(\mathbf{q}_i)$. For the computation of basis function derivatives we refer the reader to [4].

For the prescribed Dirichlet constraints we keep the subspace formulation $\left\|\mathbf{F\Phi w} - \mathbf{F\bar{d}}\right\|^2$ of (11). Combining this with the above spatial energies, i.e., with the MLS version of the gradient matrix $\mathbf{G}$ and the Laplace matrix $\mathbf{L}$, leads to the final $k \times k$ linear least squares system

$$\left(w_s \mathbf{G}^T\mathbf{G} + w_b \mathbf{L}^T\mathbf{L} + w_f \mathbf{\Phi}^T\mathbf{F}^T\mathbf{F}\mathbf{\Phi}\right)\mathbf{w} \;=\; w_f \mathbf{\Phi}^T\mathbf{F}^T\mathbf{F}\bar{\mathbf{d}}. \tag{14}$$

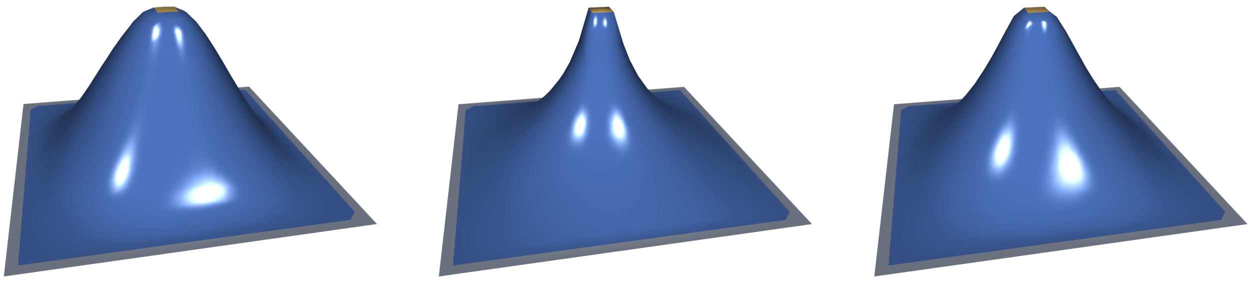

Fig. 4. Handle-based deformation of a plane (4000 vertices) minimizing deformation energies using a spatial discretization based on MLS (1000 kernels). Pure bending (left), pure stretching (center), and a mixture thereof (right) $w_b = 0.1$, $w_s = 3.0$.

Solving this system yields the desired MLS *space deformation*, which no longer depends on the complexity and quality of the input meshes. As demonstrated in Figure 4, the MLS deformation based on spatial energies provides the same deformation quality and flexibility as the surface-based energy discretization of Figure 3.

6. Constrained Space Deformation

A design prototype typically contains regions with important geometric properties such as planar components, characteristic feature lines, or circular couplings. Such geometric features are often essential for the design in order to fulfill its function or to meet production limitations. The classical approach to maintain such constraints during an optimization process is to penalize constraint violation by integrating additional penalty terms into the fitness or cost function. However, this approach has the severe drawback that infeasible designs are still created and *evaluated*, which is particularly unfavorable when the performance evaluation involves time-consuming CFD or FEM simulations.

In contrast, we propose to maintain constraints right from the start by incorporating them directly into the deformation method, thereby preventing the evaluation of infeasible designs. Within our method the user marks a particular region—probably guided by some mechanism for automatic detection of geometric primitives—as being of a particular constraint type such as, e.g., planarity. Then, when deforming the shape by manipulating the handle region, our method automatically makes sure that the corresponding constraint is satisfied *while still minimizing the deformation energy* of (1).

As already noted in Section 2, several approaches to constrained deformation have been proposed during recent years [14]. Most of them, however, are purely surface-based in nature and therefore too limited for general design optimization tasks. In contrast, the Shape-Up technique of Bouaziz and colleagues [19] maintains constraints on arbitrary geometric data sets, making it the method of choice for our application area. In the following, we will briefly describe the technique and show how we adopt it within our system. For a full treatment of the method, however, we refer the reader to the original paper [19].

The key ingredients of Shape-Up are projection operators for different types of constraints. Modeling a constraint (e.g., planarity) for a vertex set $\mathbf{x}$ requires the projection $P(\mathbf{x})$ of $\mathbf{x}$ onto the constraint set, i.e., the smallest change of $\mathbf{x}$ such that it satisfies the constraint. For a planarity constraint, for instance, $P(\mathbf{x})$ computes the projection onto a least squares fitting plane. For the most common constraints this projection can be computed quite easily [19].

The Shape-Up method then minimizes deviation from the constraints as squared distance from constraint projections:

$$E_{\text{const}}(\mathbf{x}) \;=\; \sum_{r=1}^{s} \|\mathbf{x} - P_r(\mathbf{x})\|^2 \,. \tag{15}$$

Since the projections $P_r(\mathbf{x})$ typically are nonlinear functions of $\mathbf{x}$, E_{const} is minimized by an alternating optimization procedure: First $\mathbf{x}$ is kept fixed and the projections $\mathbf{o}_r = P_r(\mathbf{x})$ are computed. Then the projected target positions $\mathbf{o}_r$ are held fixed and $\mathbf{x}$ is updated by a least squares fit to the $\mathbf{o}_r$, leading to a linear system of the form

$$\mathbf{C}^T\mathbf{C}\mathbf{x} \;=\; \mathbf{C}^T\bar{\mathbf{o}}, \tag{16}$$

where $\mathbf{C}$ denotes the relative position of vertices in constraint sets and $\bar{\mathbf{o}}$ is the vector of the stacked projections $\mathbf{o}_r$. This alternating procedure is iterated until convergence.

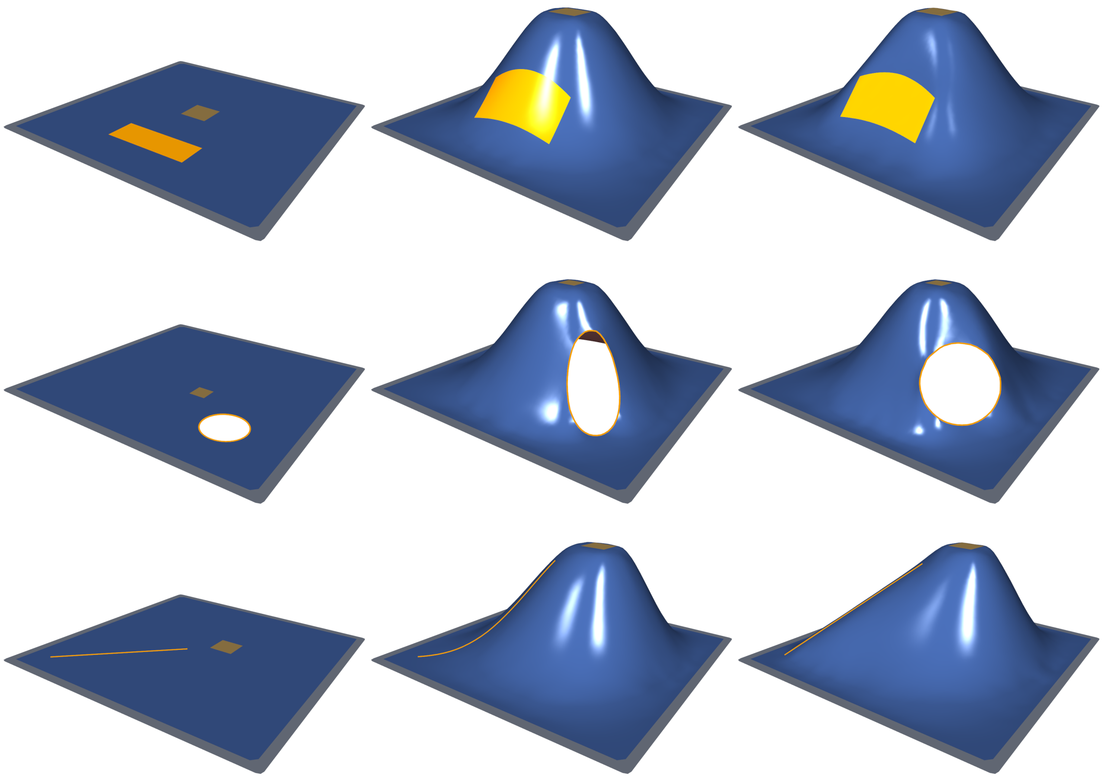
Fig. 5. Synthetic constraint examples. For each constraint type (planarity, circularity, feature line) we show the original mesh, the deformation without constraint, and the deformation minimizing bending and constraint energies using $w_b = 1.0$ and $w_c = 10$ as weights.

In order to integrate this approach into our framework, we add a constraint energy similar to (15) to our discrete shell energy (9) (weighted by w_c) and also perform the above alternating optimization. We first find the constraint projections $P(\mathbf{x})$ and combine them into the target vector $\bar{\mathbf{o}}$, which we rewrite in terms of deformation $\mathbf{d}$ instead of position $\mathbf{x}$. The minimization of constraint deviation is then integrated into the previous least squares system:

$$\left(w_s \mathbf{G}^T\mathbf{G} + w_b \mathbf{L}^T\mathbf{L} + w_f \mathbf{\Phi}^T\mathbf{F}^T\mathbf{F}\mathbf{\Phi} + w_c \mathbf{\Phi}^T\mathbf{C}^T\mathbf{C}\mathbf{\Phi}\right)\mathbf{w} \;=\; \mathbf{\Phi}^T\left(w_f \mathbf{F}^T\mathbf{F}\bar{\mathbf{d}} + w_c \mathbf{C}^T\bar{\mathbf{o}}\right). \tag{17}$$

In our current system, we implement three basic geometric constraint types of fundamental nature and general use: Planarity, circularity, and feature lines. However, additional constraints such as rigidly deforming regions or constraints on the shape of individual mesh elements can be easily added by employing suitable projection operators. For planarity and circularity constraints we employ projection operators described in [19]. Our feature line constraint is modeled as a conformal matching of the initial feature line, which therefore might translate, rotate, and uniformly scale. In Figure 5 we show synthetic examples for each constraint type.

7. Results

In this section, we present different deformation results using our constrained space deformation technique. We use the Eigen [31] library for efficient matrix operations and the sparse Cholesky decomposition of CHOLMOD [26] for solving linear systems. We parallelize the evaluation of MLS basis functions and their derivatives using OpenMP [32]. Furthermore, we use the `Surface_mesh` data structure [33] for efficient surface mesh deformation. In a typical modeling scenario satisfying the prescribed fixed and handle constraints is of highest importance and geometric constraints satisfaction is typically more important than smoothness minimization. Therefore, we select the weights balancing the individual constraint contributions such that $w_f > w_c > w_{s/b}$, where $w_f \approx 1000$, $w_c \approx 10$, $w_{s/b} \approx 1$. We also note that we normalize the different weights by the number of constraints prescribed for a given type.

7.1. Surface Deformation

In this section, we present examples for constrained deformations on surface models of typical mechanical parts such as they can occur within design optimization scenarios. We begin with example deformations of the fandisk model in Figure 6. In this setup, we keep the bottom part of the model fixed and translate the handle region to the left. We select a subset of the sharp edges of the model as feature lines, and an additional planar constraint area in the upper left area. As becomes clear from the illustration, deforming the model without constraints distorts both feature lines and the planar region, whereas with constraints both of them are nicely preserved.

Example deformations of the joint model are illustrated in Figure 7. We keep the bottom fixed again, lift the top handle region, and impose a circularity constraint on the pipe-like opening. Without the constraint the opening would no longer fit with connecting parts, with the constraint, it does. The rightmost image in Figure 7 shows the use of an additional planarity constraint. In this case, the initially already planar region deforms in such a way that the resulting mesh minimizes both the smoothness and planarity energies.

Finally, as a more complex example, we show a deformation of the DrivAer [34] reference shape for car body aerodynamics in Figure 8. The mesh contains 465k vertices, and we use 4k MLS samples to discretize our deformation energies. As can be seen from the illustration, the circular shape of the wheelhouse is nicely preserved.

7.2. Volume Deformation

In this section, we compare the volume mesh morphing quality of our new method to that of our previously proposed RBF technique. We show an example deformation of a tetrahedral volume mesh containing 13k vertices in Figure 9. In this setup, we keep the outer boundary fixed and use the interior sphere-shaped boundary as handle. We can see that both techniques allow for rather large deformations without resulting in inverted mesh elements. For the sake of comparison with our previous results [29] we analyze mesh quality in terms of minimum scaled Jacobian. Our new method results in even slightly increased mesh quality (0.05) compared to our previous RBF morphs (0.03).

As an additional comparison with the results from [29], we include a morphing example of the pipe model as shown in Figure 10. The original mesh has a min. scaled Jacobian of 0.98. After performing one step of absolute morphing to the full parameter change the RBF morph results in a mesh quality of 0.951, and our new method yields 0.954.

8. Conclusion & Outlook

In this paper, we presented a novel space deformation technique based on MLS methods for its use in design optimization scenarios. Our method offers similarly high quality deformations as our previous RBF morphing technique, but with significantly increased scalability. Our space-based energy discretization allows for flexible modeling operations typically only provided by mesh-based techniques. Finally, by incorporating geometric constraints into the deformation, we not only increase modeling capabilities but also its usefulness for design optimization tasks.

Even though our technique provides increased flexibility and scalability compared to RBF morphing, the implementation complexity increases as well. While RBF deformations simply require the solution of a dense linear system, our new technique involves Lloyd-relaxation in 3-space, numerical integration, more complex basis functions and derivatives, as well as the selection of several parameters such as the constraint weights, the number of sample points, or the basis functions support radii.

There are multiple directions for future work: So far, we only included basic constraints into our system. Therefore, a natural direction for future work would be the integration of additional constraint types such as the maintenance of mutual distances between parts or the adherence to maximal or minimal widths and heights. Additional constraints on the volume mesh are conceivable as well, such as explicit constraints on the size and shape of boundary layer elements. More advanced constraints could include relations between multiple parts, such as symmetry, orthogonality, or co-planarity. Another direction is the automatic detection of geometric constraints using an approach similar to [35].

Finally, we look forward to evaluating our technique within an actual design optimization setup including physics simulations, e.g., the aerodynamic performance optimization of a passenger car.

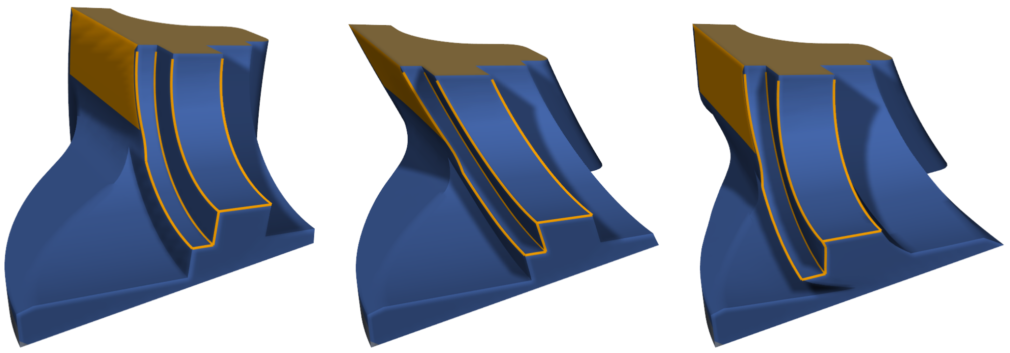
Fig. 6. Deformation of the fandisk model. From left to right: Original model, deformation without constraints, with feature line constraint, and with additional planarity constraint.

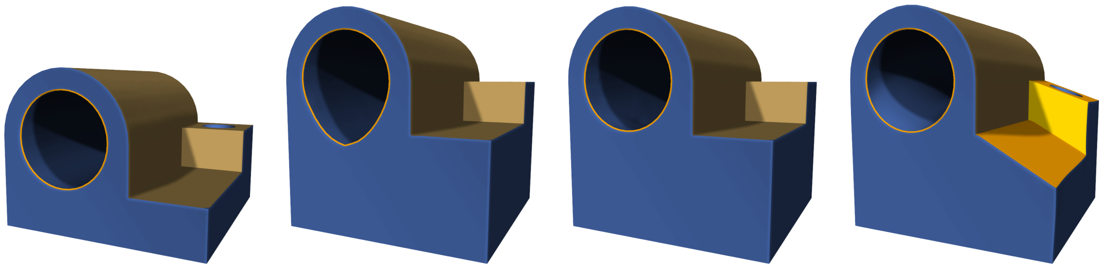
Fig. 7. Deformation of the joint model. From left to right: Original model, deformation without constraints, with a circularity constraint, and with an additional planarity constraint. Note that in the first three images the planar region only stays planar since it is part of the rigidly transformed handle region.

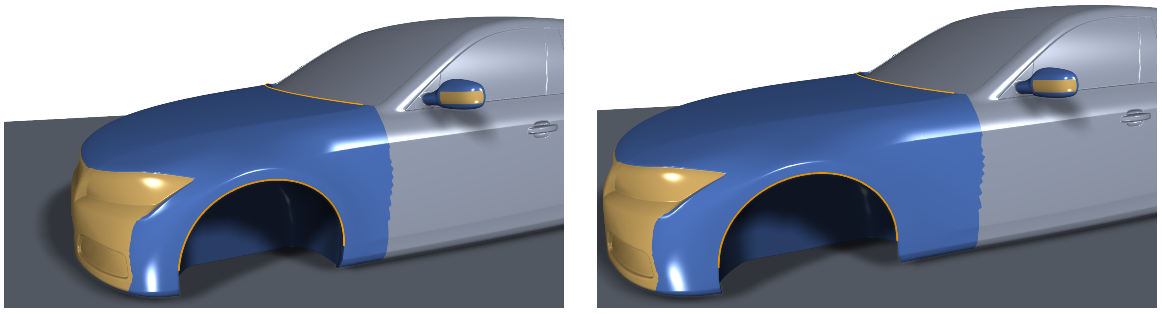
Fig. 8. Deformation of the DrivAer model. Left: Original setup. Right: Stretching the front while keeping the wheelhouse circular.

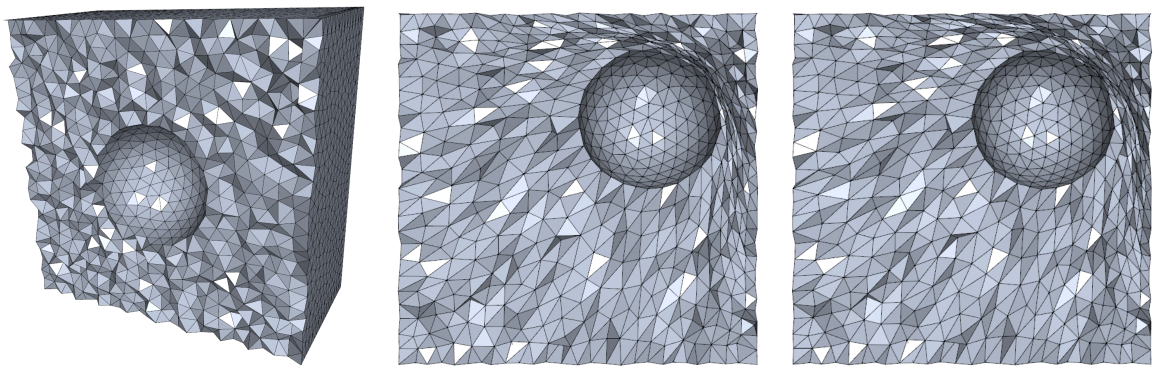
Fig. 9. Comparison of volume mesh morphing quality in terms of min. scaled Jacobian. From left to right: The original mesh (0.12), a triharmonic RBF morph (0.03), and our technique (0.05).

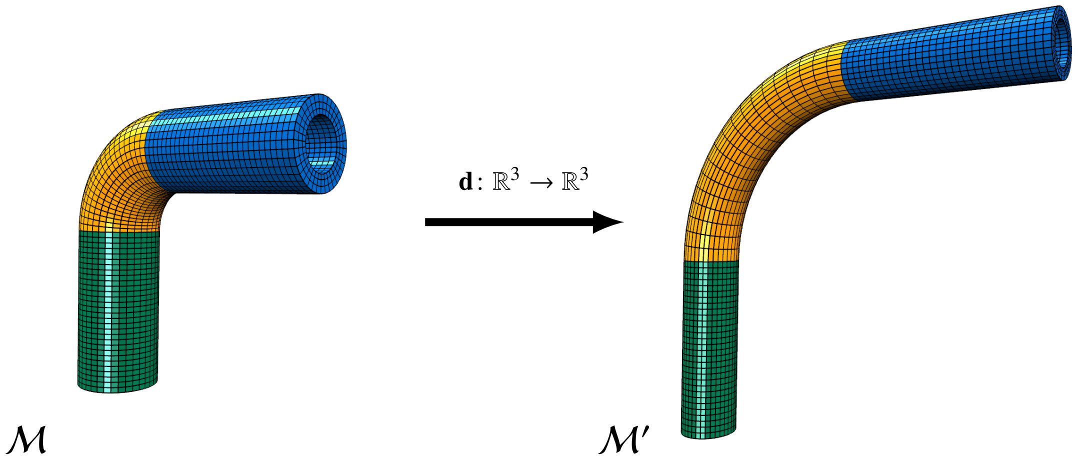

Fig. 10. Example deformation of the pipe model. Left: Original model. Right: Deformed model.

Acknowledgments

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). We thank Sebastian Martin from VM Research for providing us code for the computation of MLS basis function derivatives. The fandisk model is courtesy of Hughes Hoppe. The joint model is courtesy of Pierre Alliez. The pipe model was kindly provided by Matthew Staten from Sandia National Laboratories.

References

[1] M. L. Staten, S. J. Owen, S. M. Shontz, A. G. Salinger, T. S. Coffey, A comparison of mesh morphing methods for 3D shape optimization, in: Proceedings of the 20th International Meshing Roundtable, 2011, pp. 293–311.

[2] D. Sieger, S. Menzel, M. Botsch, High quality mesh morphing using triharmonic radial basis functions, in: Proceedings of the 21st International Meshing Roundtable, Springer-Verlag, Berlin, 2012, pp. 1–15.

[3] D. Sieger, S. Menzel, M. Botsch, On shape deformation techniques for simulation-based design optimization, in: New Challenges in Grid Generation and Adaptivity for Scientific Computing, SEMA SIMAI Springer Series, Springer Milan, 2014, to appear.

[4] T.-P. Fries, H.-G. Matthies, Classification and overview of meshfree methods, Tech. Rep. Informatikbericht 2003-3, Institute of Scientific Computing, Technical University Braunschweig (2003).

[5] T. J. Baker, Mesh movement and metamorphosis, in: Proceedings of the 10th International Meshing Roundtable, 2001, pp. 387–396.

[6] B. T. Helenbrook, Mesh deformation using the biharmonic operator, International Journal for Numerical Methods in Engineering 56 (7) (2003) 1007–1021.

[7] S. M. Shontz, S. A. Vavasis, Analysis of and workarounds for element reversal for a finite element-based algorithm for warping triangular and tetrahedral meshes, BIT Numer Math 50 (4) (2010) 863–884.

[8] T. W. Sederberg, S. R. Parry, Free-form deformation of solid geometric models, in: Proc. of ACM SIGGRAPH, 1986, pp. 151–59.

[9] S. Menzel, B. Sendhoff, Representing the change - free form deformation for evolutionary design optimization, Studies in Computational Intelligence 88 (2008) 63–86.

[10] D. Bechmann, Space deformation models survey, Computers & Graphics 18 (4) (1994) 571 – 586.

[11] J. A. Samareh, A survey of shape parameterization techniques, Tech. Rep. NASA/CP-1999-209136/PT1, NASA Langley Research Center (1999).

[12] J. Gain, D. Bechmann, A survey of spatial deformation from a user-centered perspective, ACM Transaction on Graphics 27 (4) (2008) 107:1–107:21.

[13] S. Martin, P. Kaufmann, M. Botsch, E. Grinspun, M. Gross, Unified simulation of elastic rods, shells, and solids, ACM Transaction on Graphics 29 (4) (2010) 39:1–39:10.

[14] N. J. Mitra, M. Wand, H. Zhang, D. Cohen-Or, M. Bokeloh, Structure-aware shape processing, in: Eurographics 2013 STARs, 2013, pp. 175–197.

[15] H. Masuda, Y. Yoshioka, Y. Furukawa, Preserving form features in interactive mesh deformation, Computer-Aided Design 39 (5) (2007) 361 – 368.

[16] R. Gal, O. Sorkine, N. J. Mitra, D. Cohen-Or, iWIRES: an analyze-and-edit approach to shape manipulation, ACM Transaction on Graphics 28 (3) (2009) 33:1–33:10.

[17] Y. Zheng, H. Fu, D. Cohen-Or, O. K.-C. Au, C.-L. Tai, Component-wise controllers for structure-preserving shape manipulation, Computer Graphics Forum 30 (2) (2011) 563–572.

[18] M. Habbecke, L. Kobbelt, Linear analysis of nonlinear constraints for interactive geometric modeling, Computer Graphics Forum 31 (2) (2012) 641–650.

[19] S. Bouaziz, M. Deuss, Y. Schwartzburg, T. Weise, M. Pauly, Shape-up: Shaping discrete geometry with projections, Computer Graphics Forum 31 (5) (2012) 1657–1667.

[20] S. Menzel, M. Olhofer, B. Sendhoff, Direct manipulation of free form deformation in evolutionary design optimisation, in: International Conference on Parallel Problem Solving From Nature (PPSN), 2006, pp. 352–361.

[21] D. Sieger, S. Menzel, M. Botsch, A comprehensive comparison of shape deformation methods in evolutionary desgin optimization, in: Proceedings of the 3rd International Conference on Engineering Optimization, 2012.

[22] M. Botsch, L. Kobbelt, An intuitive framework for real-time freeform modeling, ACM Transaction on Graphics 23 (3) (2004) 630–34.

[23] M. Botsch, O. Sorkine, On linear variational surface deformation methods, IEEE Transactions on Visualization and Computer Graphics 14 (1) (2008) 213–30.

[24] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, B. Levy, Polygon Mesh Processing, AK Peters, 2010.

[25] M. Meyer, M. Desbrun, P. Schröder, A. H. Barr, Discrete differential-geometry operators for triangulated 2-manifolds, in: H.-C. Hege, K. Polthier (Eds.), Visualization and Mathematics III, Springer-Verlag, Heidelberg, 2003, pp. 35–57.

[26] Y. Chen, T. A. Davis, W. W. Hager, S. Rajamanickam, Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate, ACM Transactions on Mathematical Software 35 (3) (2008) 1–14.

[27] H. Wendland, Scattered Data Approximation, Cambridge University Press, Cambridge, UK, 2005.

[28] S. Lloyd, Least square quantization in PCM, IEEE Transactions on Information Theory 28 (2) (1982) 129–37.

[29] D. Sieger, S. Menzel, M. Botsch, RBF morphing techniques for simulation-based design optimization, Engineering with Computers 30 (2) (2014) 161–174.

[30] G. H. Golub, C. F. van Loan, Matrix Computations, Johns Hopkins University Press, Baltimore, 1989.

[31] G. Guennebaud, B. Jacob, et al., Eigen v3, http://eigen.tuxfamily.org (2010).

[32] OpenMP Architecture Review Board, OpenMP application program interface version 3.1 (2011).
URL `http://www.openmp.org/mp-documents/OpenMP3.1.pdf`

[33] D. Sieger, M. Botsch, Design, implementation, and evaluation of the `Surface_mesh` data structure, in: Proceedings of the 20th International Meshing Roundtable, 2011, pp. 533–550.

[34] A. I. Heft, T. Indinger, N. A. Adams, Introduction of a new realistic generic car model for aerodynamic investigations, in: SAE 2012 World Congress, 2012.

[35] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, N. J. Mitra, Globfit: Consistently fitting primitives by discovering global relations, ACM Transaction on Graphics 30 (4) (2011) 52:1–52:12.