Constrained Space Deformation Techniques for Design Optimization

Daniel Sieger[a], Sergius Gaulik[a], Jascha Achenbach[a], Stefan Menzel[b], Mario Botsch[a]

[a] *Graphics & Geometry Group, Bielefeld University, Germany*
[b] *Honda Research Institute Europe GmbH, 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 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, planar or cylindrical 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—either manually or driven by an optimization algorithm—based on performance results during physical 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. We note that an alternative approach to avoid the costly remeshing of a CAD model is the use of isogeometric analysis [1]. In this work, however, we focus on more traditional design optimization and simulation scenarios.

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 [2] and our recent investigations [3, 4], 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 [3, 4] 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 cost or fitness function of the optimization process. While this strategy effectively excludes unfavorable designs as the outcome of the optimization process, it still requires the costly creation and evaluation of unfavorable design variations *during* the process since the penalty terms are applied in the fitness or cost function evaluation after the variations have been created and evaluated.

In this paper, we present a shape deformation technique based on moving least squares (MLS) discretization [5] 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.

We extend our previous work [6] in several key aspects: First, we extend the geometric primitives supported in our constrained deformation method to support cylindrical regions and rigid components. Second, we extend our comparison of subspace deformation techniques to include both global biharmonic and compact Gaussian RBFs, as well as more insightful mean curvature visualizations. Third, in order to make the setup procedure of our deformation method easier for the designer or engineer, we incorporate a technique for the automatic detection of geometric primitives into our system. Fourth, in order to boost the scalability of our constraint deformation, we introduce an alternative formulation of projective constraints ensuring sparsity of the resulting linear system. Fifth, we include additional deformation examples, including a combined volume and surface deformation of a practical CFD setup.

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 [7, 8]. The finite element-based FEMWARP technique [7, 9], which computes a harmonic deformation, was generalized from tetrahedra to hexahedra in [2], 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 limitations by computing a space deformation $\mathbf{d}\colon \mathbb{R}^3 \to \mathbb{R}^3$ that deforms the whole embedding space, thereby implicitly deforming the mesh. Spline-based free-form deformation (FFD) techniques [10] have been widely used in both the graphics and engineering communities [11]. After its initial conception numerous extensions have

been proposed, and we refer the reader to the survey papers [12–14] for a more comprehensive overview. However, spline-based FFD does not offer the same degree of fairness as harmonic or biharmonic deformations, and it requires a rather tedious control lattice setup, as we investigate in detail in [4].

In [3] we successfully combined the advantages of meshless approaches and mesh-based variational methods by employing *radial basis functions* (RBFs) for mesh deformation. RBF space deformations can handle arbitrary polyhedral meshes and offer a degree of fairness 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 which energy to minimize. Furthermore, due to the global support of their basis functions, the resulting linear systems are dense and therefore limited in terms of scalability.

In this paper, we propose to overcome these limitations by employing *moving least squares* (MLS) methods [5, 15] 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 deformations, 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 [16], as exemplified by the feature-preserving surface deformation technique of Masuda and colleagues [17], or by the iWires system [18] for deformation of man-made objects. More recently, the latter approach was generalized to component-wise controllers [19], and the work of Habbecke and Kobbelt [20] 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 technique of Bouaziz and colleagues [21], 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

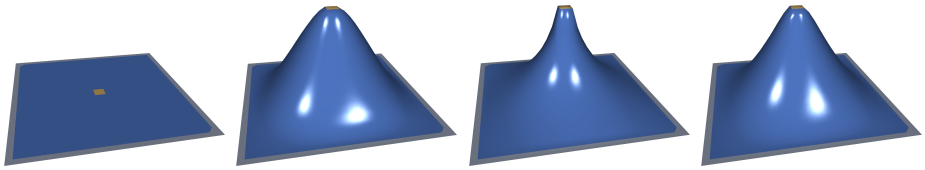


Figure 1: Handle-based surface deformation of a plane (1100 vertices). From left to right: Undeformed model, minimization of pure bending, pure stretching, and a mixture thereof with parameters $\gamma_b = 0.6$ and $\gamma_s = 1.0$. We choose $\gamma_f = 100$ in order to ensure that prescribed handle and fixed constraints are satisfied.

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) [10], 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 [22, 23].

Furthermore, we employ the so-called *handle metaphor* [24], 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

$$E_{\text{shell}}[\mathbf{d}] = \gamma_s\, E_{\text{stretch}}[\mathbf{d}] + \gamma_b\, E_{\text{bend}}[\mathbf{d}] + \gamma_f\, E_{\text{fix}}[\mathbf{d}], \tag{1}$$

consisting of weighted energy contributions for bending, stretching, and constraint deviation [25]:

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

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

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

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 energies 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 [25, 26] as

$$E_{\text{stretch}}[\mathbf{d}_h] = \sum_{t\in\mathcal{D}} A_t \left\|\nabla \mathbf{d}_t\right\|^2, \tag{6}$$

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

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

where A_i denotes the Voronoi area of vertex i, and A_t is the area of triangle t. We use the well-established discrete differential operators proposed in Meyer et al. [27], which allows us to write the discrete gradient $\nabla \mathbf{d}_t$ 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

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

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 [25, 26] 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 us to solve the normal equations of the linear least squares system

$$\left(\gamma_s \mathbf{G}^T\mathbf{G} + \gamma_b \mathbf{L}^T\mathbf{L} + \gamma_f \mathbf{F}^T\mathbf{F}\right) \mathbf{d} \;=\; \gamma_f \mathbf{F}^T\mathbf{F}\bar{\mathbf{d}}, \tag{10}$$

which we solve efficiently using sparse Cholesky factorization [28]. Note that in case the conditioning of the normal equations becomes a problem, we could also solve the system directly using a sparse QR factorization method. In order to ensure proper satisfaction of the Dirichlet boundary constraints, we typically choose γ_f to be one or two orders of magnitude larger than the smoothness weights γ_s and γ_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.

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 us to deform an existing volume mesh simultaneously with the surface, a feature of particular importance for design optimization. Furthermore, complex designs often consist of multiple *disconnected components* that space deformations can naturally deform at once, while mesh-based methods would require an additional coupling to propagate the deformation from one component to another. 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:

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

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

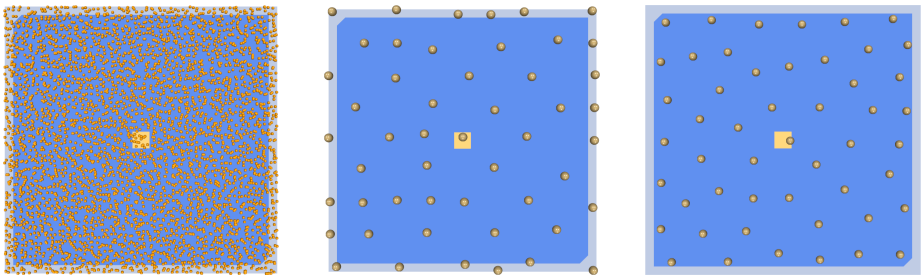


Figure 2: Surface sampling: Left: Dense random sampling of the mesh. From the dense samples we select a farthest point subset (center), perform iterative Lloyd relaxation on this subset (right), and use these points as RBF centers/MLS samples .

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 $\boldsymbol{w} = (\boldsymbol{w}_1^T, \ldots, \boldsymbol{w}_k^T)^T \in \mathbb{R}^{k\times 3}$ using a $n \times k$ subspace matrix $\boldsymbol{\Phi}$:

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

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

$$\boldsymbol{\Phi}^T \left(\gamma_s \mathbf{G}^T\mathbf{G} + \gamma_b \mathbf{L}^T\mathbf{L} + \gamma_f \mathbf{F}^T\mathbf{F}\right) \boldsymbol{\Phi}\, \boldsymbol{w} \;=\; \boldsymbol{\Phi}^T \left(\gamma_f \mathbf{F}^T\mathbf{F}\bar{\mathbf{d}}\right), \tag{11}$$

which has a drastically reduced complexity compared to the previous least squares system for surface deformation in (10).

In the following, we compare different choices for the shape functions φ_j. Motivated by our previous investigations [3, 4], we focus on meshless, kernel-based discretizations, and start with globally supported triharmonic and biharmonic RBFs, which however have the drawback of high computational cost and limited scalability. We then analyze compactly supported Gaussians and Wendland RBFs [29] as well as moving least squares discretization [5].

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 [30], 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: 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 [30].

Global RBFs. In our previous work [3, 31], we successfully employed global triharmonic RBFs for high quality mesh deformation. Following this approach, we can construct a subspace by using shape functions

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

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 scalability. Even though biharmonic basis functions $\varphi_j(\mathbf{x}) = \|\mathbf{x} - \mathbf{c}_j\|$ (see [29, 32] for the derivation of general polyharmonic RBFs) yield improved results for stretching minimization, they still suffer from the same scalability limitations as triharmonic basis functions.

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

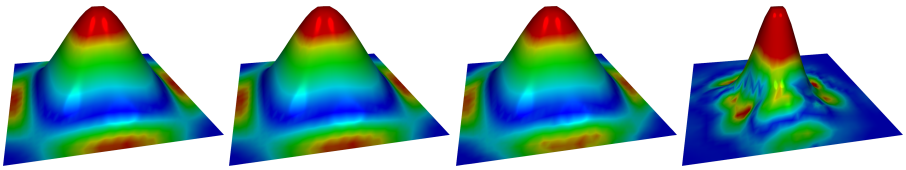
$$\begin{aligned}\varphi_j(\mathbf{x}) \;&=\; \varphi\left(\|\mathbf{x} - \mathbf{c}_j\|\right) \\ &=\; \varphi(r) \;=\; \begin{cases} (1-(r/\sigma))^4((4r/\sigma)+1)\,, & r < \sigma\,, \\ 0\,, & \text{otherwise}\,. \end{cases}\end{aligned}$$

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 depend on the chosen support radius. A small radius of $s = 5$ leads to artifacts in the deformation. Only with an large radius of $s = 50$ 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. Similar limitations apply to Gaussian RBFs

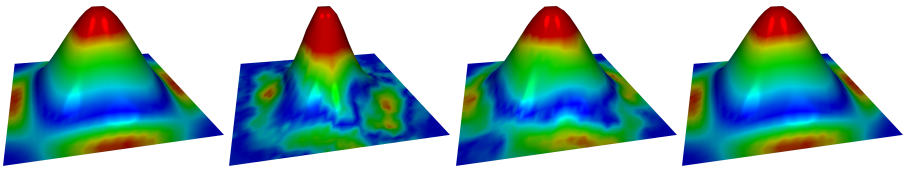
$$\varphi_j(\mathbf{x}) \;=\; e^{-\frac{\left\|\mathbf{x}-\mathbf{c}_j\right\|^2}{\sigma^2}}\,,$$

leading to a certain amount of smoothness artifacts in the deformed meshes.

Bending

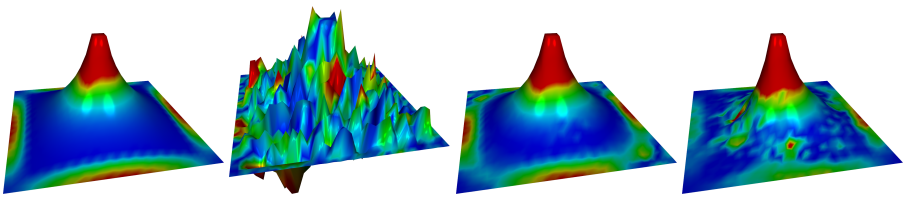


(a) surface (b) triharmonic (c) biharmonic (d) Wendland $s = 5$

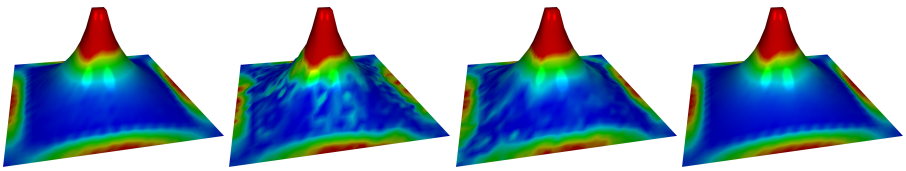


(e) Wendland s=50 (f) Gaussian $s = 5$ (g) Gaussian $s = 50$ (h) MLS $s = 5$

Stretching



(i) surface (j) triharmonic (k) biharmonic (l) Wendland $s = 5$



(m) Wendland s=50 (n) Gaussian $s = 5$ (o) Gaussian $s = 50$ (p) MLS $s = 5$

Figure 3: Subspace deformation of a plane (1100 vertices) minimizing bending (top rows) and stretching (bottom rows) energies. For each energy type we compare the mean curvature plot of the ground truth surface deformation, global triharmonic and biharmonic RBFs, compact Wendland and Gaussian RBFs with small ($s = 5$) and large ($s = 50$) support, MLS with small ($s = 5$) support. RBFs and MLS use 1000 shape functions.

Moving Least Squares. An alternative to RBFs is the meshless moving least squares (MLS) approximation method, which 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 already 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 [5] 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 $r > \sigma$. Other choices of smooth basis functions work equally well, see [5] for details. 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). In contrast to Martin and colleagues [15], who switch to more complex generalized MLS basis functions in order to handle degenerate sampling, we robustly handle this case by replacing the inverse $\mathbf{M}^{-1}$ by the pseudo-inverse $\mathbf{M}^{+}$ [33]. We compute $\mathbf{M}^{+}$ as $\mathbf{V}^T\mathbf{\Sigma}^{+}\mathbf{U}$ based on the singular value decomposition (SVD) $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, since this is the numerically most stable method [33, 34]. Using the SVD is also the computationally most expensive technique for computing the pseudo-inverse, but for our 4×4 matrices this turned out to not be crucial.

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 $\mathbf{\Phi}$, and hence all pseudo-inverse computations, 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.

We provide a performance comparison between global and compact RBFs as well as MLS basis functions in Figure 4. In this test, we perform deformations with an increasing number of basis function centers ranging from 1k to 10k degrees of freedom. We can observe that with an increasing number of basis functions both compact RBFs and MLS outperform the globally supported RBFs. In terms of memory usage the globally supported RBFs result in dense system matrices and therefore have quadratically growing storage requirements. In contrast, both compact methods have storage requirements growing only linear with a constant depending on the support radius.

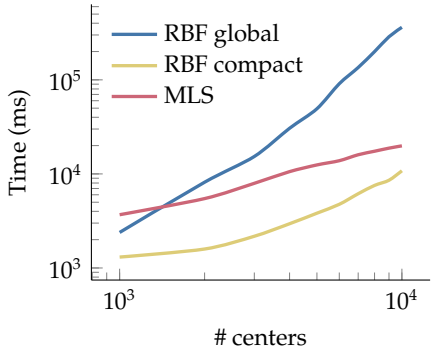


Figure 4: Performance comparison between global biharmonic RBFs, compact Wendland RBFs ($s = 50$), and MLS basis functions ($s = 5$). Computation time in milliseconds versus the number of basis function centers.

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. The flexibility of MLS deformations was for instance demonstrated in [15], where MLS shape functions were used to deform solids, shells, and rods based on mechanical/physical laws.

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—still employed the stretching and bending energies (6)–(8) based on a surface mesh. 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 sampling. 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 $\mathbf{c}_j$, $j = 1, \ldots, k$.

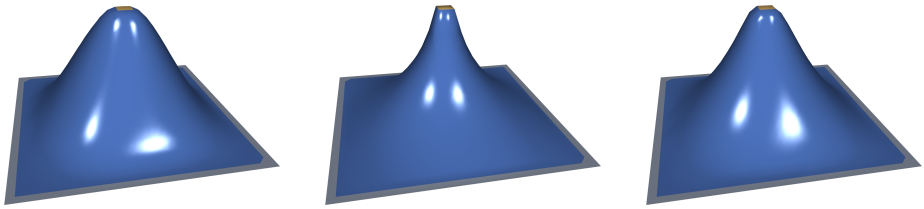


Figure 5: Handle-based deformation of a plane minimizing deformation energies using a spatial discretization based on MLS (1000 kernels). Pure bending (left), pure stretching (center), and a mixture thereof (right).

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:

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

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

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

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

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

with a $N \times k$ Laplacian matrix $\mathbf{L}_{i,j} = \sqrt{V_i}\Delta\varphi_j(\mathbf{q}_i)$. For the computation of analytical basis function derivatives we refer the reader to [5].

For the prescribed Dirichlet constraints we keep the subspace formulation $\left\|\mathbf{F}\boldsymbol{\Phi}\boldsymbol{w} - \mathbf{F}\bar{\mathbf{d}}\right\|^2$ 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

$$\left(\gamma_s \mathbf{G}^T\mathbf{G} + \gamma_b \mathbf{L}^T\mathbf{L} + \gamma_f \boldsymbol{\Phi}^T\mathbf{F}^T\mathbf{F}\boldsymbol{\Phi}\right)\boldsymbol{w} \;=\; \gamma_f \boldsymbol{\Phi}^T\mathbf{F}^T\mathbf{F}\bar{\mathbf{d}}. \quad (14)$$

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 5, 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 and cylindrical 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 constrained deformation approaches have been proposed during recent years [16]. 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 [21] 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 [21].

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 $\mathcal{C}$, 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 [21].

Let m be the number of different constraint sets $\mathcal{C}_t$, $t = 1, \dots, m$. The Shape-Up method then measures deviation from the constraints as squared distance from constraint projections $P_t(\mathbf{x})$:

$$E_{\text{constr}}(\mathbf{x}) \;=\; \sum_{t=1}^{m} \|\mathbf{x} - P_t(\mathbf{x})\|^2 \,. \tag{15}$$

Since the projections $P_t(\mathbf{x})$ typically are nonlinear functions of $\mathbf{x}$, E_{constr} is minimized by *alternating optimization* (also called block coordinate descent): First $\mathbf{x}$ is kept fixed and all projections $\mathbf{x}_t = P_t(\mathbf{x})$ are computed (local step). Then the projected target positions $\mathbf{x}_t$ are held fixed and $\mathbf{x}$ is updated by a simultaneous least squares fit to the target positions $\mathbf{x}_t$ (global step). This process is iterated until convergence, and converges to a feasible solution if it exists (see [21] for details). We choose this type of minimization strategy for the sake of simplicity and robustness. However, other variants such as a non-linear conjugate gradient method or a trust-region variant of Newton's method [35] are applicable as well.

In each iteration, the global step requires the solution of a linear least squares system of the form

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

where $\bar{\mathbf{x}}$ is the vector of the stacked projections $\mathbf{x}_t$. The matrix $\mathbf{C}$ contains the stacked constraint matrices $\mathbf{C}_t$, which combine the mean-centered positions of constrained points, i.e., for each constraint set $\mathcal{C}_t$ involving n_t points $\mathbf{C}_t$ is a $n_t \times n_t$ matrix with entries

$$\mathbf{C}_t(i,j) \;=\; \begin{cases} 1 - \frac{1}{n_t}\,, & i = j\,, \\ -\frac{1}{n_t}\,, & \text{otherwise}\,. \end{cases}$$

When combining the individual constraint matrices $\mathbf{C}_t$ into the global matrix $\mathbf{C}$, we adjust the columns of $\mathbf{C}_t$ such that they match the corresponding indices in the global vertex set $\mathbf{x}$. The mean-centering of positions allows for translation of constraint sets during optimization, thereby improving the overall convergence rate of the iterative alternating optimization (see also Section 2.2 and Figure 10 in [21]).

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 γ_c) and also perform the above alternating optimization procedure. In each iteration, we first find the constraint projections $P(\mathbf{x})$ (local step) and combine them into the target vector $\bar{\mathbf{x}}$, which we rewrite in terms of displacements $\mathbf{d}$ instead of position $\mathbf{x}$. The global minimization of constraint deviation is then integrated into the previous least squares system

$$\begin{aligned}\left(\gamma_s\mathbf{G}^T\mathbf{G} + \gamma_b\mathbf{L}^T\mathbf{L} + \gamma_f\mathbf{\Phi}^T\mathbf{F}^T\mathbf{F}\mathbf{\Phi} + \gamma_c\mathbf{\Phi}^T\mathbf{C}^T\mathbf{C}\mathbf{\Phi}\right)\boldsymbol{w} \;=& \\ \mathbf{\Phi}^T\left(\gamma_f\mathbf{F}^T\mathbf{F}\bar{\mathbf{d}} + \gamma_c\mathbf{C}^T\bar{\mathbf{x}}\right),& \end{aligned} \tag{17}$$

which we again solve efficiently using sparse Cholesky factorization [28]. We iterate this alternating optimization procedure until convergence, which typically happens after 1k–10k iterations in our examples—depending on the complexity of the constraints involved. Note that the convergence rate is independent from the chosen sampling density which only affects approximation accuracy.

6.1. Alternative constraint formulation

Applying the above constraint formulation in modeling setups with large constraint regions such as those obtained from automatic primitive detection (see Section 6.3) reveals a severe limitation in the above formulation: Due to the mean-centering of constraints, the constraint matrix $\mathbf{C}$ can become dense, such that the resulting linear system can no longer be solved efficiently. The reason for this is the following: Let n_t be the number of points involved in a particular constraint set $\mathcal{C}_t$. Then in the original formulation this leads to $n_t \times n_t$ non-zero entries in the constraint matrix $\mathbf{C}$, leading to a fully dense matrix in the worst case scenario of a constraint covering the whole mesh, e.g., a planarity constraint on a planar mesh.

However, the primary reason for mean-centering is the improved convergence rate due to translation invariance of each constraint set. We propose an alternative constraint formulation sharing the same improved convergence behavior while ensuring sparsity of the constraint matrix. Instead of mean-centering the points involved in a constraint set $\mathcal{C}_t$, we may also subtract an arbitrary point $\mathbf{x}_p \in \mathcal{C}_t$, e.g., the point closest to the mean, thereby leading to n_t rows with only two entries

$$\mathbf{C}_t(i,j) \;=\; \begin{cases} -1\,, & j = p\,, \\ 1\,, & j = i\,. \end{cases}$$

Using the above formulation effectively ensures that the constraint matrix $\mathbf{C}$ is sparse even in case of large constraint regions $\mathcal{C}_t$.

6.2. Constraint Types

In our current system, we implement five basic geometric constraint types of fundamental nature and general use: Planarity, circularity, cylinder, feature lines, and rigid shape constraints. For planarity and circularity constraints we employ projection operators described in [21]. For the cylinder constraint, we use the cylinder fitting method described in [36] and project the points back onto the fitted cylinder. Our feature line constraint is modeled as a conformal matching of the points on the initial feature line, which therefore might translate, rotate, and uniformly scale. Similar to the feature line constraint, we also support shape constraints based on rigid matching, allowing for translation and rotation only. This constraint can be useful in a number of settings, e.g., in order to keep complete components of a design rigid, or to rigidly maintain the shape of

selected mesh elements—such as boundary layer cells in a CFD mesh—which are of particular importance for accurate physical performance calculation. In Figure 6, we show synthetic examples demonstrating the effect of each constraint type. Note that due to the iterated nature of the alternating optimization, the resulting meshes do indeed minimize the deformation energy while satisfying the geometric constraints.

6.3. Automatic Constraint Detection

The manual selection of geometric primitives in a mesh can be a tedious and time-consuming task. In order to speed up the setup process and guide the designer towards meaningful constraints, we employ a procedure for the automatic detection of primitives in the surface mesh which then can be used as constraints during deformation. Note that even though in some design optimization scenarios it is possible to transfer information about geometric primitives from a corresponding CAD model, this is not necessarily the case in purely mesh-based modeling or reverse engineering scenarios.

The automatic detection of geometric primitives in point sets is a well-known problem in the context of surface reconstruction and segmentation. One of the most widely used techniques is the random sample consensus (RANSAC) algorithm [37]. The core idea of this technique is to repeatedly draw random samples defining a geometric primitive from given point data and then to evaluate how well the primitive approximates the rest of the data set, see [38] for details. Our primitive detection is based on a combination of the efficient RANSAC algorithm described in [39] and a forward search technique [40, 41] to further refine the estimated primitives.

A key question when using the above techniques on a surface mesh is the choice of input data. Simply using mesh vertices and vertex normals leads to unsatisfactory results, since the normals at vertices on sharp feature edges are not aligned with the normal direction of the (multiple) geometric primitives such a vertex belongs to, see Figure 7 left. We resolve this issue by using face barycenters and face normals as input instead. The detected primitives are then assigned to all vertices belonging to the corresponding faces, thereby allowing a vertex to belong to multiple primitives. Our system currently supports plane, cylinder, sphere, and cone primitives, see Figure 7 for examples. From the set of detected primitives the user then selects those to be preserved during deformation.

7. Results

In this section, we present different deformation results using our constrained space deformation technique. We use the Eigen [42] library for efficient matrix operations and the sparse Cholesky decomposition of CHOLMOD [28] for solving linear systems. We parallelize the evaluation of MLS basis functions

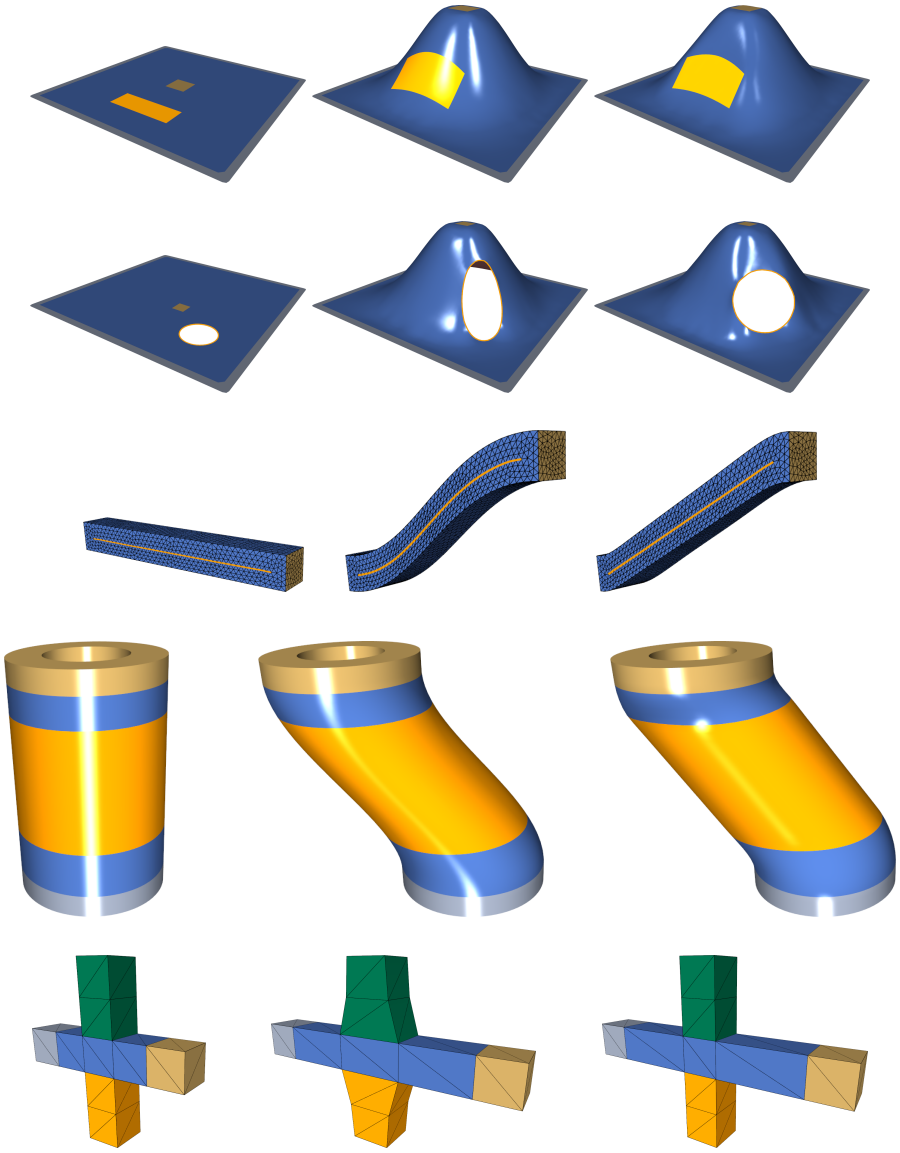


Figure 6: Synthetic constraint examples. For each constraint type (planarity, circularity, feature line, cylinder, rigidity) we show the original mesh, the deformation without constraint, and the deformation minimizing bending and constraint energies using $\gamma_b = 1.0$ and $\gamma_c = 10$ as weights.

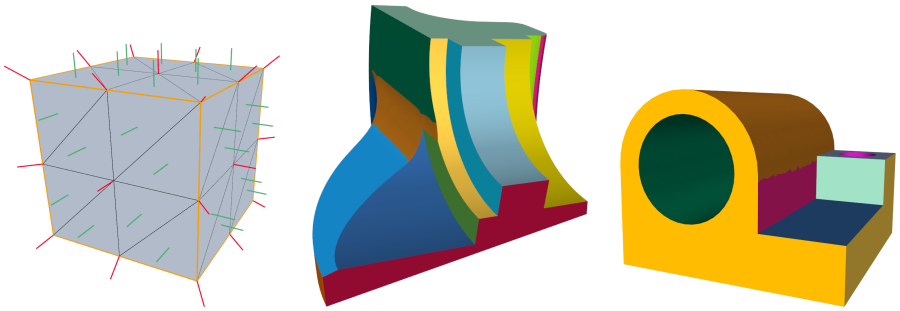


Figure 7: Left: Cube with sharp feature edges (orange), face (green) and vertex normals (red). Center/right: Automatically detected primitives in the fandisk and joint models. Each colored region corresponds to a plane or cylinder primitive.

and their analytical derivatives using OpenMP [43]. Furthermore, we use the `Surface_mesh` data structure [44] 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 $\gamma_f > \gamma_c > \gamma_{s/b}$, where $\gamma_f \approx 1000$, $\gamma_c \approx 10$, $\gamma_{s/b} \approx 1$. We also note that we normalize the different weights by the number of constraints prescribed for a given type and region.

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 8. 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 9. We keep the bottom fixed, 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 9 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. In Figure 10 we present a deformation example of the joint model using constraints determined through our automatic geometric primitive detection. We keep the bottom fixed again and use the top cylinder region as a handle.

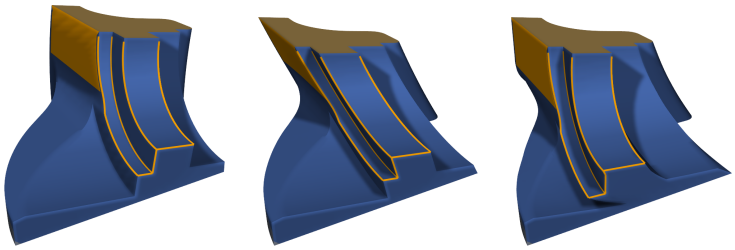


Figure 8: Deformation of the fandisk model. From left to right: Original model, deformation without constraints, with feature line constraint, and with additional planarity constraint.

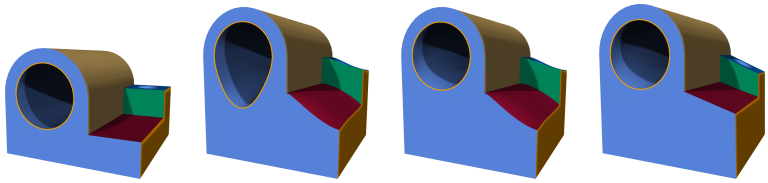


Figure 9: Deformation of the joint model. From left to right: Original model, deformation without constraints, with a circularity constraint, and with a additional planarity constraints.

Figure 10: Deformation of the joint model using automatically detected geometric primitives. Left: Original setup keeping the bottom fixed and using the top cylinder region (golden) as a handle. Right: Deformed model.

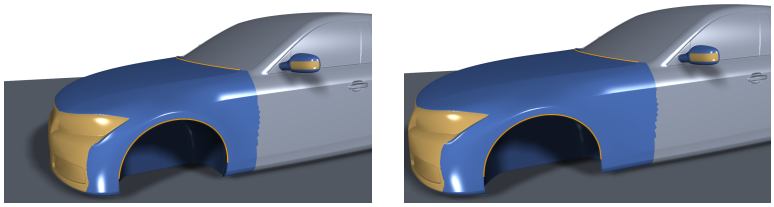


Figure 11: Deformation of the DrivAer model. Left: Original setup. Right: Stretching the front while keeping the wheelhouse circular.

Finally, as a more complex example, we show a deformation of the DrivAer [45] reference shape for car body aerodynamics in Figure 11. The mesh contains 465k vertices, and we use 4k MLS samples to discretize the displacements and about 16k cubature points to discretize the deformation energies. As can be seen from the illustration, the circular shape of the wheelhouse is nicely preserved. We also note that deformations using global RBFs would not be easily applicable to this scenario: The number of user-prescribed handle and fixed constraints is much too high to be feasible for dense linear systems solvers. However, advanced RBF methods such as specialized incremental QR solvers ([31, 46]) or fast multipole methods ([29, 47]) might still be applicable.

7.2. Volume Deformation

In this section, we compare the quality of our new volumetric mesh deformation method to that of our previously proposed RBF technique. We show an example deformation of a tetrahedral volume mesh containing 13k vertices in Figure 12. 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 large deformations without resulting in inverted mesh elements. In order to provide a quantitative comparison to our previous results [31], 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 deformations (0.03). Both methods produce similar results, and we refer to [31] for a quantitative evaluation of mesh quality and element inversion of the RBF technique as well as other state-of-the-art deformation methods.

As an additional comparison to our previous results [31], we include a deformation example of the hexahedral pipe model containing 11k vertices and 8.5k cells. We show the original and deformed meshes in Figure 13. The original mesh has a minimum scaled Jacobian of 0.98. After performing one step of absolute deformation to the full parameter change—see [31] for a description of absolute and relative deformation—the RBF deformation results in a mesh quality of 0.951, and our new method yields 0.954.

7.3. Combined Surface and Volume Deformation

As already noted in the introduction, an important feature of space deformation methods is the ability to deform an existing volumetric simulation mesh *along* with a surface. Therefore, our final example is a combined surface and volume mesh deformation of the DrivAer model, as illustrated in Figure 14. In this case, we use a hex-dominant polyhedral volume mesh generated using OpenFOAM's `snappyHexMesh` utility, containing 1.3M vertices and 970k cells. We use 5k MLS samples on the surface as well as 2k samples in the volume to discretize displacements. Correspondingly, we use about 28k integration points to discretize our deformation energies. In order to better preserve the shape of boundary layer cells, we apply rigid shape constraints on those elements. For the

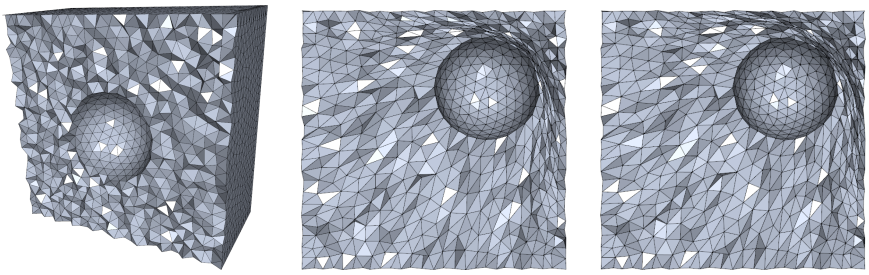


Figure 12: Comparison of volume mesh deformation quality in terms of min. scaled Jacobian. From left to right: The original mesh (0.12), a triharmonic RBF deformation (0.03), and our technique (0.05). Note that both methods did not produce inverted elements, as indicated by their still positive minimum scaled Jacobians.

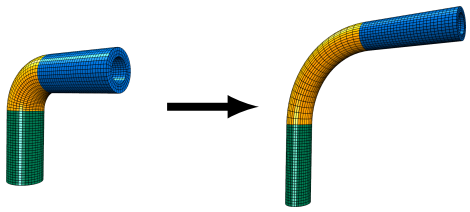


Figure 13: Deformation of the pipe model, comparison in terms of minimum scaled Jacobian. Left: Original model (0.98). Right: Deformed model (0.954).

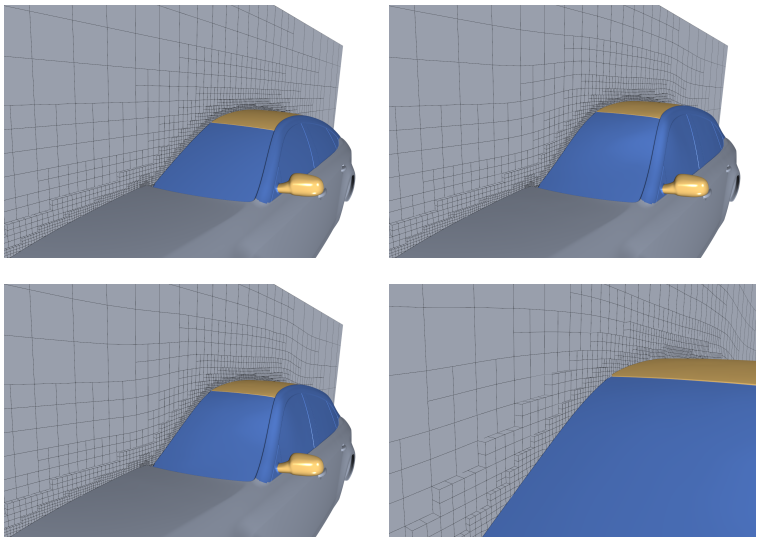


Figure 14: Combined volume and surface deformation of the DrivAer model. Top row: Original setup and deformation without constraints. Bottom row: Deformation with rigid shape constraints, close-up of boundary layer elements.

deformation we lift the roof of the car model, which is a standard deformation in optimizing car body aerodynamics. After deformation the overall volume mesh quality is nicely preserved and the mesh is still usable for simulation, as we evaluated using OpenFOAM's `checkMesh` utility. The different meshes yield the following results for the important cell orthogonality check: The original mesh has a maximum value of 53.81, the deformed mesh without constraints has 54.52, and the mesh with rigid shape constraints enabled yields a value of 54.42 (smaller value indicating higher mesh quality). The less critical aspect ratio and faces skewness checks do not report significant differences.

8. Conclusion and 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 deformation technique, but with significantly increased scalability. Our space-based energy discretization allows for flexible modeling operations typically only provided by mesh-based deformation techniques. Finally, by incorporating geometric constraints into the deformation, we not only increase modeling capabilities but also its usefulness for design optimization tasks. Our alternative formulation of projective constraints increases the scalability of the technique to modeling scenarios involving large constraint regions, and our automatic detection of geometric primitives aids the designer or engineer during setup process.

Even though our technique provides increased flexibility and scalability compared to RBF deformation, 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 function support radii.

There are multiple directions for future work: Even though we currently support a wide variety of constraints, 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. More advanced constraints could include relations between multiple parts, such as symmetry, orthogonality, or co-planarity, including methods for the automatic analysis of constraints using an approach similar to [48].

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

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. The DrivAer model is courtesy of the Institute for Aerodynamics and Fluid Mechanics at TU Munich.

References

[1] J. A. Cottrell, T. J. Hughes, Y. Bazilevs, Isogeometric Analysis: Toward Integration of CAD and FEA, John Wiley & Sons, 2009.

[2] 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.

[3] 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.

[4] D. Sieger, S. Menzel, M. Botsch, On shape deformation techniques for simulation-based design optimization, in: S. Perotto, L. Formaggia (Eds.), New Challenges in Grid Generation and Adaptivity for Scientific Computing, Vol. 5 of SEMA SIMAI Springer Series, Springer International Publishing, 2015, pp. 281–303.

[5] 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).

[6] D. Sieger, S. Menzel, M. Botsch, Constrained space deformation for design optimization, Procedia Engineering 82 (0) (2014) 114–126, proceedings of the 23rd International Meshing Roundtable. doi:http://dx.doi.org/10.1016/j.proeng.2014.10.377.

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

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

[9] 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.

[10] T. W. Sederberg, S. R. Parry, Free-form deformation of solid geometric models, in: Proc. of ACM SIGGRAPH, 1986, pp. 151–159.

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

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

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

[14] 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.

[15] 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.

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

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

[18] 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.

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.

[24] M. Botsch, L. Kobbelt, An intuitive framework for real-time freeform modeling, ACM Transaction on Graphics 23 (3) (2004) 630–634.

[25] M. Botsch, O. Sorkine, On linear variational surface deformation methods, IEEE Transactions on Visualization and Computer Graphics 14 (1) (2008) 213–230.

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

[27] 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.

[28] 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.

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

[30] S. Lloyd, Least square quantization in PCM, IEEE Transactions on Information Theory 28 (2) (1982) 129–137.

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

[32] J. Duchon, Spline minimizing rotation-invariant semi-norms in Sobolev spaces, in: W. Schempp, K. Zeller (Eds.), Constructive Theory of Functions of Several Variables, no. 571 in Lecture Notes in Mathematics, Springer Verlag, Berlin, 1977, pp. 85–100.

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

[34] L. N. Trefethen, D. Bau, Numerical Linear Algebra, SIAM, Philadelphia, 1997.

[35] J. Nocedal, S. Wright, Numerical Optimization, Springer, 2006.

[36] P. Schneider, D. H. Eberly, Geometric Tools for Computer Graphics, Morgan Kaufmann, 2002.

[37] M. A. Fischler, R. C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM 24 (6) (1981) 381–395.

[38] G. Roth, M. D. Levine, Extracting geometric primitives, CVGIP: Image Understanding 58 (1) (1993) 1–22.

[39] R. Schnabel, R. Wahl, R. Klein, Efficient RANSAC for point-cloud shape detection, Computer Graphics Forum 26 (2) (2007) 214–226.

[40] A. C. Atkinson, M. Riani, A. Cerioli, Exploring Multivariate Data with the Forward Search, Springer Series in Statistics, Springer, 2004.

[41] S. Fleishman, D. Cohen-Or, C. T. Silva, Robust moving least-squares fitting with sharp features, ACM Transaction on Graphics 24 (3) (2005) 544–552.

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

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

[44] 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.

[45] 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.

[46] M. Botsch, L. Kobbelt, Real-time shape editing using radial basis functions, Computer Graphics Forum (Proc. Eurographics) 24 (3) (2005) 611–21.

[47] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, T. R. Evans, Reconstruction and representation of 3D objects with radial basis functions, in: Proc. of ACM SIGGRAPH, ACM, New York, 2001, pp. 67–76.

[48] 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.