On Shape Deformation Techniques for Simulation-based Design Optimization

Daniel Sieger and Stefan Menzel and Mario Botsch

Abstract We present an in-depth analysis and benchmark of shape deformation techniques for their use in simulation-based design optimization scenarios. We first introduce classical free-form deformation, its direct manipulation variant, as well as deformations based on radial basis functions. We compare the techniques in a series of representative synthetic benchmarks, including computational performance, numerical robustness, quality of the deformation, adaptive refinement, as well as precision of constraint satisfaction. As an application-oriented benchmark we investigate the ability to adapt an existing volumetric simulation mesh according to an updated surface geometry, including unstructured tetrahedral, structured hexahedral, and arbitrary polyhedral example meshes. Finally, we provide a detailed assessment of the methods and give concrete advice on choosing a suitable technique for a given optimization scenario.

1 Introduction

Simulation-based design optimization is a key aspect of the product development process of automotive industry, aircraft construction, and naval architecture. The overall goal is to explore alternative and novel designs with improved physical or aesthetic properties. The development process typically starts with the creation of

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

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

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

an initial design prototype using a computer aided design (CAD) tool. Subsequent steps create a polygon surface mesh from the CAD model as well as a volumetric simulation mesh for physical performance evaluation, e.g., using computational fluid dynamics (CFD) simulations for aerodynamic performance calculation, or finite element methods (FEM) for structural mechanics simulations. Design variations are then created based on physical performance during simulation. In this paper, we are concerned with efficient means to create such alternate designs.

The obvious approach of changing the CAD model directly is prohibitive in many cases, since both the surface and volume meshing steps would have to be repeated. For complex geometries and precise physical simulations the meshing process 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.

This paper is organized as follows: We begin with an investigation of the fundamental requirements a deformation method should satisfy in order to be suitable for common design optimization scenarios (Section 2). Based on these requirements we introduce state-of-the-art shape deformation methods including classical free-form deformation (FFD), direct manipulation FFD (DM-FFD), and deformations based on radial basis functions (RBFs). We compare the different methods in a series of synthetic and application oriented benchmarks in Section 3. Finally, we perform a detailed assessment of the methods in Section 4 and give concrete guidance for choosing a suitable technique for a particular design optimization scenario.

2 Shape Deformation Methods

In this section we introduce state-of-the-art shape deformation methods for their use in simulation-based design optimization. Before describing the individual techniques in detail, we briefly review related work, motivate our selection of methods, and introduce the concept of a space deformation. Shape deformation methods have been an area of continuous and extensive research within the fields of computer graphics and geometric modeling. Consequently, a wide variety of techniques has been proposed during recent years. Since providing an overview of the complete field is beyond the scope of this work we refer to existing introductions and surveys. Detailed references for the individual methods covered in this work are provided in the corresponding sections. A general introduction to shape deformation techniques is provided by [4]. Surveys on space deformation techniques have been presented in [1] and [8]. While the former concentrates on building a mathematical formalism for the different methods, the latter is focused on the interactive manipulation of a model by a designer. In contrast, a survey of shape parametrization techniques in

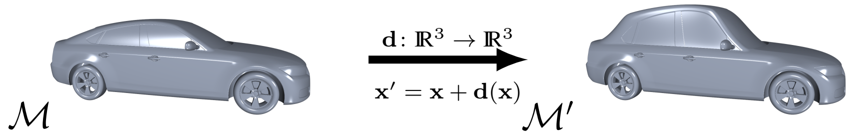
$\mathbf{d}\colon \mathbb{R}^3 \to \mathbb{R}^3$
$\mathbf{x}' = \mathbf{x} + \mathbf{d}(\mathbf{x})$
$\mathcal{M}$
$\mathcal{M}'$

Fig. 1 Deformation of the DrivAer model. The model $\mathcal{M}$ is warped by the space deformation function $\mathbf{d}$. Each point $\mathbf{x} \in \mathcal{M}$ is transformed to updated locations $\mathbf{x}' = \mathbf{x} + \mathbf{d}(\mathbf{x}) \in \mathcal{M}'$.

the context of design optimization is given in [27]. Staten and coworkers recently proposed and evaluated several mesh morphing techniques, which they compared with respect to computational performance and element quality on different tetrahedral and hexahedral meshes [38]. This evaluation was later extended by Sieger and colleagues [35, 36]. Linear variational surface deformation methods have been investigated in detail in [5].

The selection of deformation methods considered in our comparison is highly driven by our application domain—design optimization. In this context, one may formulate several requirements a deformation technique should satisfy. A fundamental one is the ability to transparently deal with different object representations such as triangular or quadrilateral surface meshes, volumetric meshes, polygon soups, as well as point-based representations. On the one hand, this requirement stems from the desire to be able to optimize a wide variety of designs. On the other hand, it is particularly important when the evaluation of the objective function involves computationally expensive CFD or FEM simulations. As already outlined in Section 1, the volumetric simulation meshes typically are very time-consuming to generate. Therefore, in order to avoid the costly mesh generation process for each design variation created during optimization, one typically aims at adapting an initial simulation mesh alongside with the surface. A second requirement is the ability to robustly deal with defects in the input geometry. Especially when the surface mesh is the result of an automatic conversion process from the CAD model the resulting mesh might contain degeneracies such as badly shaped triangles, non-manifold configurations, or disconnected components.

A type of deformation methods that naturally fulfills the above requirements are so-called *space deformations*. The fundamental idea behind these methods is to deform the embedding space around an object, thereby deforming the object implicitly. From a mathematical point of view a space deformation is a function $\mathbf{d}\colon \mathbb{R}^3 \to \mathbb{R}^3$ that maps each point in space to a certain displacement. Given a deformation function, a model $\mathcal{M}$ can be transformed to a deformed model $\mathcal{M}'$ by computing updated point locations $\mathbf{x}' = \mathbf{x} + \mathbf{d}(\mathbf{x})$ for each original point $\mathbf{x} \in \mathcal{M}$. The basic procedure is illustrated in Figure 1, where a space deformation for the DrivAer body [12] is shown. Naturally, space deformation techniques differ in how the function $\mathbf{d}$ is constructed. Typically, a control structure such as a volumetric lattice or a set of points is blended with some form of basis functions, as we will describe for the individual methods in the sections that follow. Since the deformations applied during optimization are typically relatively small, we focus on linear space deformation methods.

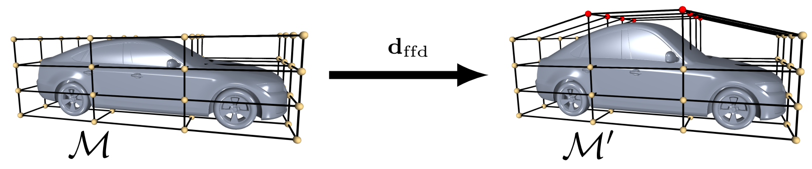

Fig. 2 Free-form deformation applied to the DrivAer model. The original model $\mathcal{M}$ is embedded in a regular lattice of $4 \times 4 \times 4$ control points (golden). After moving the selected control points (red) the updated object point locations $\mathbf{x}'$ are computed by evaluating the FFD space deformation function $\mathbf{d}_{\mathrm{ffd}}$ for the local coordinates $\mathbf{u}$ of the point $\mathbf{x}$.

2.1 Free-form Deformation

Free-form deformation (FFD) is a well-established deformation technique that has been widely used in both academia and industry. Since it also has been employed within shape optimization [18, 28] and simulation-based design optimization [19, 20, 21, 34] it forms the basis for our comparison. Before describing the method in detail we first review important variants of the technique. Free-form deformations using Bézier basis functions have been originally introduced in [31]. Local deformations using B-spline basis functions have been introduced in [11]. An extension to more flexible control lattices, in particular cylindrical ones, has been proposed in [6]. This approach was later extended to control lattices of arbitrary topology [17]. Free-form deformations using non-uniform rational B-splines are described in [16]. A highly flexible but computationally involved variant of FFD based on a 3D-Delaunay triangulation, its Voronoi dual, and Sibson coordinates [33] has been presented in [23]. A variant of FFD using T-splines [32] as basis functions—thereby allowing for local refinement of the control lattice—has been presented in [37].

The basic idea of FFD is based on embedding the object to be deformed in a parallelepiped lattice and deforming it using a trivariate tensor-product Bézier or B-spline function. The deformation procedure to perform free-form deformation of an object can be divided into several steps. First, a control lattice has to be generated and adapted to the deformation scenario at hand. Then the local coordinates with respect to the control lattice have to be computed for each point to be deformed. After this embedding each object point $\mathbf{x} \in \mathcal{M}$ can be expressed as a linear combination of lattice control points $\mathbf{c}_{ijk}$ and basis functions N_i:

$$\mathbf{x} \;=\; \sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n} \mathbf{c}_{ijk} N_i(u_1) N_j(u_2) N_k(u_3), \tag{1}$$

where (u_1, u_2, u_3) are the local coordinates of $\mathbf{x}$ with respect to the control lattice, and l, m, n are the numbers of control points in each direction. For the sake of simplicity we define

$$\mathbf{u}(\mathbf{x}) \coloneqq (u_1, u_2, u_3), \quad N_p(\mathbf{u}(\mathbf{x})) \coloneqq N_i(u_1) N_j(u_2) N_k(u_3),$$

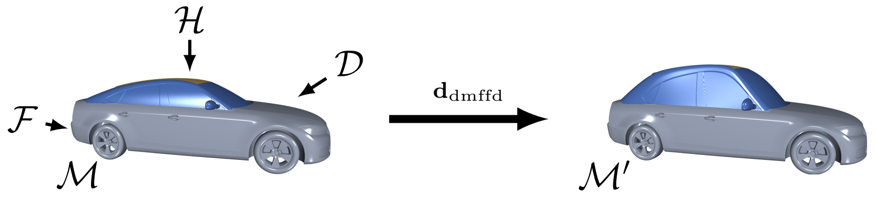

Fig. 3 Direct manipulation FFD on the DrivAer model using a handle-based direct manipulation interface. The vertices of the model $\mathcal{M}$ are classified into three distinct sets: Handle vertices ($\mathcal{H}$, golden) can be directly displaced, fixed vertices ($\mathcal{F}$, gray) are kept in place, and deformable vertices ($\mathcal{D}$, blue) are updated according to the deformation method.

as well as

$$\delta\mathbf{c}_p \coloneqq \delta\mathbf{c}_{ijk} = \mathbf{c}'_{ijk} - \mathbf{c}_{ijk},$$

where $\mathbf{c}'_{ijk}$ denotes an updated control point location. We can then define the FFD space deformation function as

$$\mathbf{d}_{\text{ffd}}(\mathbf{x}) \;=\; \sum_p \delta\mathbf{c}_p N_p(\mathbf{u}(\mathbf{x})). \tag{2}$$

Finally, the deformation is performed by moving the control points and computing the updated object point locations. An example deformation using FFD is illustrated in Figure 2.

In our implementation of FFD we use cubic B-splines with a uniform knot vector. While this type of basis functions requires an iterative root-finding technique such as a Newton method [26] for computing the local coordinates, the important advantage is the capability to perform deformations with local support.

2.2 Direct Manipulation FFD

In an interactive modeling system the manipulation of control points to perform a deformation becomes a tedious task—especially when using a complex control lattice with a large number of control points. A more flexible and intuitive interface for controlling a deformation is offered by *direct manipulation* approaches, which have been introduced for FFD in [13], referred to as DM-FFD throughout this paper. Instead of moving control points, the user *directly* moves the object points. The modeling system then computes control point displacements so that the new object point positions are matched as precise as possible. An example deformation of the DrivAer model using direct manipulation is shown in Figure 3.

Direct manipulation interfaces are not only beneficial within an interactive modeling scenario, they can also be used effectively within simulation-based design optimization, as has been shown for direct manipulation FFD in [20]. Due to the more direct influence of the parameters determined during optimization on the design, using such an interface can result in a drastically faster convergence of the optimization. In contrast to classical FFD, the ability to choose an arbitrary object point for optimization offers a higher degree of flexibility. Furthermore—due to the automatic computation of control point displacements—this approach also reduces the need to pre-deform the control lattice to a certain degree.

Within a direct manipulation interface the user—be it an engineer or an optimization algorithm—prescribes a set of m displacement constraints at so-called *handle points* $\{\mathbf{h}_1, \ldots, \mathbf{h}_m\}$, where the deformation function has to attain certain displacement values $\mathbf{d}(\mathbf{h}_i) = \mathbf{h}_i' - \mathbf{h}_i = \delta\mathbf{h}_i$. The displacements $\{\delta\mathbf{c}_1, \ldots, \delta\mathbf{c}_n\}$ of the n control points satisfying the prescribed displacements can be computed by solving the linear system

$$\underbrace{\begin{bmatrix} N_1(\mathbf{u}(\mathbf{h}_1)) & \ldots & N_n(\mathbf{u}(\mathbf{h}_1)) \\ \vdots & \ddots & \vdots \\ N_1(\mathbf{u}(\mathbf{h}_m)) & \ldots & N_n(\mathbf{u}(\mathbf{h}_m)) \end{bmatrix}}_{\mathbf{\Phi}} \underbrace{\begin{pmatrix} \delta\mathbf{c}_1^T \\ \vdots \\ \delta\mathbf{c}_n^T \end{pmatrix}}_{\mathbf{C}} = \underbrace{\begin{pmatrix} \delta\mathbf{h}_1^T \\ \vdots \\ \delta\mathbf{h}_m^T \end{pmatrix}}_{\mathbf{H}} . \tag{3}$$

Since the linear system (3) can be over-determined as well as under-determined, it is typically solved by computing the *pseudo-inverse* $\mathbf{\Phi}^+$ of the basis function matrix $\mathbf{\Phi}$. This is typically done by performing a singular value decomposition (SVD) [13, 10] so that $\mathbf{\Phi} = \mathbf{U\Sigma V}^T$, where $\mathbf{U}$ is a $m \times m$ orthogonal matrix, $\mathbf{\Sigma}$ is a $m \times n$ diagonal matrix containing the singular values of $\mathbf{\Phi}$, and $\mathbf{V}^T$ is a $n \times n$ orthogonal matrix. The pseudo-inverse of $\mathbf{\Phi}$ then is $\mathbf{\Phi}^+ = \mathbf{V\Sigma}^+\mathbf{U}^T$, where the pseudo-inverse of the diagonal matrix $\mathbf{\Sigma}$ can be computed as

$$\mathbf{\Sigma}_{ij}^+ = \begin{cases} \frac{1}{\sigma_i}, & \text{if } i = j \wedge \sigma_i \neq 0, \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where σ_i is the i-th singular value of $\mathbf{\Phi}$. We note that for values close to zero σ_i has to be clamped in order to prevent numerical instabilities. Once $\mathbf{\Phi}^+$ has been computed the control point displacements can be computed by

$$\mathbf{C} = \mathbf{\Phi}^+\mathbf{H}, \tag{5}$$

where $\mathbf{C}$ is the matrix of control point displacements and $\mathbf{H}$ is the matrix of constraint displacements. However, solving for $\mathbf{C}$ using the pseudo-inverse has its drawbacks. If the system is under-determined, a *least-norm* solution is found, i.e., the amount of movement of the control points $||\delta\mathbf{c}||$ is minimized. If the system is overdetermined, a *least-squares* solution is found, i.e., the error in satisfying the specified constraints is minimized. This means that depending on the resolution of the control lattice the system might not be able to satisfy the constraints specified by the user in an exact manner. In both cases, however, the solution does not necessarily result in a physically plausible deformation.

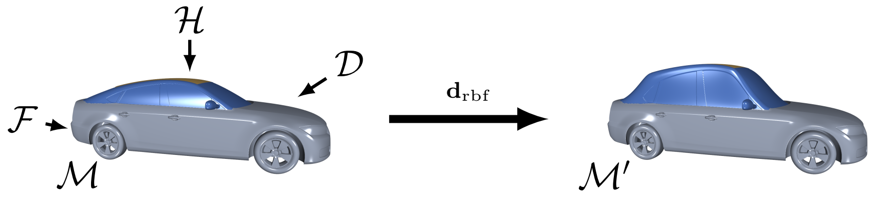

Fig. 4 Deformation of the DrivAer model using a handle-based direct manipulation interface for RBFs.

2.3 RBF Deformation

Mesh deformation using radial basis functions (RBFs) has been proposed by several authors [2, 3, 15, 22]. This method improves upon FFD and DM-FFD in two significant aspects: First, due to its point-based or kernel-based nature, introducing additional degrees of freedom in regions of interest is highly flexible, without the need to maintain a complicated control structure. Second, the deformation function can be constructed in such a way that it directly minimizes a physically inspired energy—resulting in a smooth and physically plausible deformation. An example deformation based on radial basis functions using the same handle-based interface as described in Figure 3 for DM-FFD is given in Figure 4.

On a more abstract level, we can treat mesh deformation as a scattered data interpolation problem: We search for a function $\mathbf{d}\colon \mathbb{R}^3 \to \mathbb{R}^3$ that (i) exactly interpolates the prescribed displacements $\mathbf{d}(\mathbf{h}_i) = \delta\mathbf{h}_i$ and (ii) smoothly interpolates these displacements through space. Radial basis functions are well known to be suitable for solving this type of problem [39]. Using RBFs we define the deformation function as a linear combination of radially symmetric kernel functions $\varphi_j(\mathbf{x}) = \varphi(\|\mathbf{x} - \mathbf{c}_j\|)$, located at centers $\mathbf{c}_j \in \mathbb{R}^3$ and weighted by $\mathbf{w}_j \in \mathbb{R}^3$, plus a linear polynomial to guarantee linear precision:

$$\mathbf{d}(\mathbf{x}) \;=\; \sum_{j=1}^{m} \mathbf{w}_j \varphi_j(\mathbf{x}) \;+\; \sum_{k=1}^{4} \mathbf{q}_k \pi_k(\mathbf{x})\,, \tag{6}$$

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

The choice of the kernel function $\varphi\colon \mathbb{R} \to \mathbb{R}$ basically determines the shape of the interpolant. Commonly used kernels include Gaussians, (inverse) multiquadrics, and polyharmonic splines (see Table 1 for an overview). In our application we aim for high quality deformations minimizing the distortion of mesh elements. To meet this goal, we have to use a sufficiently smooth kernel function. While Gaussian and multiquadric basis functions provide infinite smoothness, i.e., they are C^∞, they require the choice of an additional shape parameter (the ϵ in Table 1). Small values

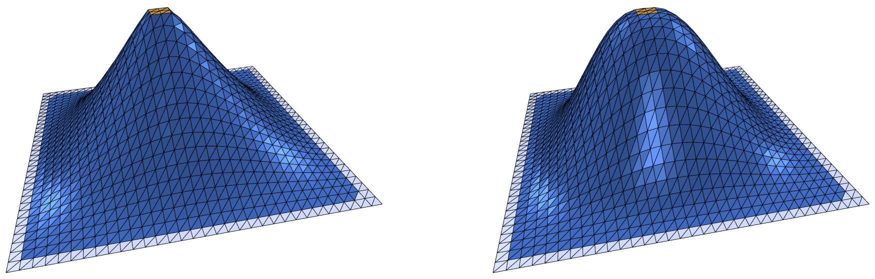

Fig. 5 Comparison between a biharmonic (left) and a triharmonic (right) deformation of a plane. We displace the golden region, keep the gray region fixed, and deform the blue region. We place RBF kernels on all vertices in the golden and gray regions.

of ϵ increase approximation accuracy, but lead to numerically instabilities, and *vice versa*. Therefore, finding the optimal shape parameter for a given radial basis function and the particular application is a non-trivial task on its own (see [7] for an overview of different strategies).

In contrast, polyharmonic splines are free of shape parameters, but only of finite smoothness. Depending on the application scenario, we have to choose a sufficiently high degree of smoothness. In $\mathbb{R}^3$ the polyharmonic spline $\varphi_k(r) = r^{2k-3}$ is a fundamental solution of the k-th order Laplacian Δ^k, such that also the RBF deformation (6) is k-harmonic, i.e., $\Delta^k \mathbf{d} = 0$. Being the strong form of a variational energy minimization, this is equivalent [39] to $\mathbf{d}$ minimizing the weak form

$$\iiint_{\mathbb{R}^3} \left\| \frac{\partial^k \mathbf{d}}{\partial x^k} \right\|^2 + \left\| \frac{\partial^k \mathbf{d}}{\partial x^{k-1} \partial y} \right\|^2 + \ldots + \left\| \frac{\partial^k \mathbf{d}}{\partial z^k} \right\|^2 \mathrm{d}x\, \mathrm{d}y\, \mathrm{d}z. \tag{7}$$

In order to preserve mesh quality during deformation, we should construct a deformation function that at least minimizes the change of first-order derivatives of the mesh elements [38], and therefore the first-order derivatives of the deformation function. With $k = 1$ in (7), this is achieved by the harmonic RBF $\varphi(r) = 1/r$, but those basis functions are singular at their centers. The biharmonic spline $\varphi(r) = r$ is well defined, but not differentiable at the center and therefore not smooth enough

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

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

for our application (see Figure 5). By choosing $\varphi(r) = r^3$, we obtain a deformation function that is triharmonic, therefore penalizes third-order derivatives in (7), and is globally C^2 smooth. With these properties, it is the lowest-order polyharmonic RBF suitable for our application. Since for numerical robustness a low order is preferable, we eventually chose triharmonic RBFs for our deformation method.

We can *exactly* satisfy the interpolation constraints $\mathbf{d}(\mathbf{h}_i) = \delta\mathbf{h}_i$ by placing RBF kernels at the constraint positions (i.e., $\mathbf{c}_j = \mathbf{h}_j$) and finding the coefficients $\mathbf{w}_j$ and $\mathbf{q}_k$ by solving the $(m+4) \times (m+4)$ linear system

$$\underbrace{\begin{pmatrix} \varphi_1(\mathbf{h}_1) & \cdots & \varphi_m(\mathbf{h}_1) & \pi_1(\mathbf{h}_1) & \cdots & \pi_4(\mathbf{h}_1) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \varphi_1(\mathbf{h}_m) & \cdots & \varphi_m(\mathbf{h}_m) & \pi_1(\mathbf{h}_m) & \cdots & \pi_4(\mathbf{h}_m) \\ \pi_1(\mathbf{h}_1) & \cdots & \pi_1(\mathbf{h}_m) & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \pi_4(\mathbf{h}_1) & \cdots & \pi_4(\mathbf{h}_m) & 0 & \cdots & 0 \end{pmatrix}}_{\mathbf{\Phi}} \cdot \underbrace{\begin{pmatrix} \mathbf{w}_1^T \\ \vdots \\ \mathbf{w}_m^T \\ \mathbf{q}_1^T \\ \vdots \\ \mathbf{q}_4^T \end{pmatrix}}_{\mathbf{W}} = \underbrace{\begin{pmatrix} \delta\mathbf{h}_1^T \\ \vdots \\ \delta\mathbf{h}_m^T \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix}}_{\mathbf{H}} . \qquad (8)$$

After solving (8) we can compute the morphed mesh $\mathcal{M}'$ by simply evaluating the RBF deformation at each mesh vertex: $\mathbf{x}_i' = \mathbf{x}_i + \mathbf{d}(\mathbf{x}_i)$. The computationally most expensive part is the solution of the linear system (8), which is dense due to the global support of $\varphi(r)$. We discuss the performance and the scalability of our method in Section 3.1.

3 Benchmarks

In this section we evaluate the different deformation methods based on a set of synthetic benchmarks. The goal of these benchmarks is to capture basic properties of the different deformation methods which are relevant for the use in design optimization scenarios. We perform our evaluation based on the following criteria: computational performance, numerical robustness, adaptivity and precision, as well as quality of the deformation. For each criterion we first describe our tests and methodology, and then present the results for the individual deformation methods.

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

In many of the benchmarks a direct comparison with FFD based on control point manipulation is not really feasible, i.e., it is not possible to compare the methods on a solid and representative basis. In these cases, we only compare DM-FFD and RBFs.

3.1 Performance

While the impact of the performance of a deformation method is often negligible when used within an design optimization loop, it is still an important and fundamental characteristic. Furthermore, it is crucial for usage in an interactive modeling application. Within control point-based FFD, the only performance-critical component is the computation of the local coordinates of each object point with respect to the control lattice. When using B-spline basis functions, this computation requires the use of a numerical technique such as a golden section search or a Newton method [26]. However, since the local coordinate computation is independent for each object point, this part is trivial to parallelize.

Naturally, direct manipulation FFD also requires the local coordinate computation discussed above. In addition, however, the linear system (3) has to be solved. The standard approach for this is based on computing the pseudo-inverse using singular value decomposition, which has a computational cost of $4m^2n + 22n^3$ floating point operations [10]. Additional computational costs come from the matrix multiplications required to actually compute the pseudo-inverse $\mathbf{\Phi}^+ = \mathbf{V}\mathbf{\Sigma}^+\mathbf{U}^T$ from the SVD.

Within the RBF deformation technique the most expensive part is the solution of the linear system (8), which is dense due to the global support of the chosen radial basis functions $\varphi(r) = r^3$. The resulting asymptotic complexity is of $O(m^3)$ when using standard solvers for dense linear systems. Since the linear system (8) is symmetric but not positive definite, efficient Cholesky-type solvers are not applicable. However, the system can still be solved efficiently by using a LDL^T factorization, which has computational costs of $\frac{1}{3}m^3$ floating point operations. For a more comprehensive investigation of different solvers for RBF deformation we refer to [36].

However, benchmarking the performance by simply measuring the time it takes to deform a given mesh is not really meaningful since the methods pre-compute different amounts of information. Comparing the performance of control point-based FFD to directly manipulated DM-FFD or RBFs is not feasible, since there is no way to perform the same deformation with all three methods. In order to facilitate a representative and objective comparison between DM-FFD and RBFs, we present an alternative formulation of both deformation methods which allows us to fully pre-compute the deformation. The deformation methods we investigated are linear, i.e., they require solving a linear problem in one form or another. Therefore, the deformations can be pre-computed by solving a sequence of m linear system (see, e.g., [3]). Even more, the methods can be handled in a uniform manner by expressing the deformation in terms of handle basis functions.

The m displacement constraints $\delta\mathbf{h}_i$ are given as prescribed values of the deformation function $\mathbf{d}(\mathbf{h}_i) = \delta\mathbf{h}_i$. In case of DM-FFD, the control point displacements $\delta\mathbf{c}_j$ satisfying these constraints are found by solving the linear system of (3). In case of RBFs, we find the weights $\mathbf{w}_j$ for the deformation function (6) by solving (8). What we are searching for are the displacements $\mathbf{x}'_i - \mathbf{x}_i = \delta\mathbf{x}_i$ for each deformable vertex $\mathbf{x}_i$. Written in matrix form this becomes $\mathbf{X} = (\delta\mathbf{x}_1, \ldots, \delta\mathbf{x}_k)^T$, where k is the number of deformable vertices. In case of DM-FFD, $\mathbf{X}$ can be computed using

$$\mathbf{X} \;=\; \mathbf{M}\cdot\mathbf{C}, \quad \mathbf{M}_{ij} = N_j(\mathbf{u}(\mathbf{x}_i)), \tag{9}$$

where $N_j(\mathbf{u}(\mathbf{x}_i))$ is the trivariate tensor-product B-spline basis function of control point $\mathbf{c}_j$ evaluated at point $\mathbf{x}_i$, and $\mathbf{C}$ is the matrix of control point displacements $\delta\mathbf{c}_j$. By substituting $\mathbf{C}$ using (5) we can rewrite (9) as

$$\mathbf{X} \;=\; \underbrace{\mathbf{M}\cdot\mathbf{\Phi}^{+}}_{\mathbf{B}}\cdot\mathbf{H},$$

where $\mathbf{H}$ is the $m \times 3$ matrix of prescribed handle displacements. Using the $k \times m$ matrix $\mathbf{B}$ we can then directly evaluate the vertex displacements in terms of handle displacements.

The corresponding formulation for RBFs is similar: $\mathbf{X}$ can be computed by

$$\mathbf{X} \;=\; \mathbf{M}\cdot\mathbf{W}, \quad \mathbf{M}_{ij} = \varphi_j(\mathbf{x}_i), \tag{10}$$

where $\mathbf{W}$ is the matrix of radial basis function weights. Based on (8) the weight matrix $\mathbf{W}$ can be computed by inverting $\mathbf{\Phi}$, i.e., as $\mathbf{W} = \mathbf{\Phi}^{-1}\mathbf{H}$. This yields

$$\mathbf{X} \;=\; \underbrace{\mathbf{M}\cdot\mathbf{\Phi}^{-1}}_{\mathbf{B}}\cdot\mathbf{H}.$$

Then $\mathbf{B}$ is the desired $k \times m$ basis function matrix that can be used to compute the vertex displacements from the given handle displacements.

Based on this formulation, we compare the performance of the methods by precomputing a deformation with 427 constraints. In Figure 6 we present the results comparing both FFD variants in their serial and parallel (using OpenMP [25]) versions as well as RBFs. As to be expected from theory, DM-FFD offers the worst performance. While parallel local coordinate computation clearly improves (DM-)FFD performance, RBFs require the same amount of time to solve the full problem.

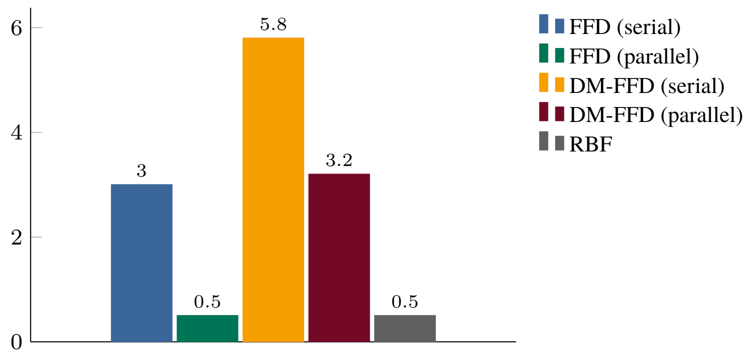

Fig. 6 Performance comparison of the deformation methods. Times in seconds. For FFD methods a control grid of resolution 8^3 was used, which results in a comparable number of DoFs as the RBF setup.

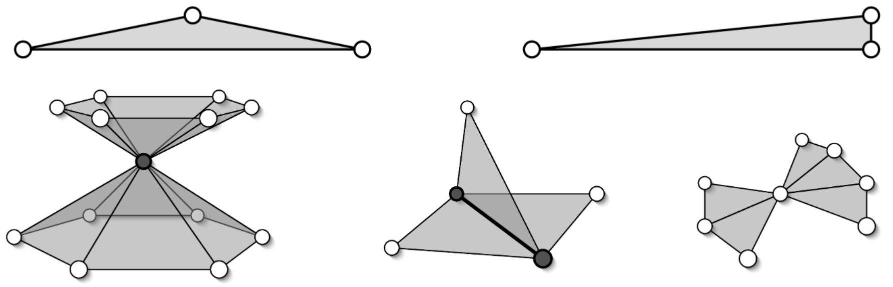

Fig. 7 Examples of common surface mesh degeneracies. Top row: low quality triangles. Bottom row: non-manifold connectivity.

3.2 Robustness

The robustness of a deformation method describes its robustness towards defects in the input data. Such defects can include low-quality triangles with very large or very small angles, such as caps or needle elements, non-manifold configurations, or self-intersections in the input mesh (see Figure 7 some for common examples). Due to their space-based nature, FFD and RBFs are highly robust with respect to defects in the input data. However, in the direct manipulation variant of FFD the singular value decomposition used to compute the pseudo-inverse might also be a source for numerical instabilities. In order to prevent division by zero, artifacts in the deformation, as well as extreme distortions of the control lattice, one has to clamp the singular values σ_i in (4). Figure 8 presents an example of the unwanted artifacts in the deformation depending on different clamping values. Since a suitable clamping value for a given deformation setup is not known a-priori, it has to be determined heuristically by the user—thereby constituting a source of increased effort and potential failure.

Non-manifold configurations are problematic in general, since in this case the 1-ring neighborhood of a vertex can no longer be traversed reliably. While this does not pose a direct problem for the methods investigated, it might prevent the use of a more efficient reduced constraint direct manipulation interface as described in [3].

3.3 Quality of Deformation

The quality of the deformation includes several aspects. On the most general level, the deformation should be free of any unexpected oscillations or artifacts. Following the principle of *simplest shape* [29], the deformation function should be smooth, fair, and physically plausible. Furthermore, we want the deformation to maintain mesh element quality as much as possible in order to allow for as large as possible deformations. We note, however, that the methods we consider do not incorporate additional mesh optimization procedures that are eventually required for particularly large deformations.

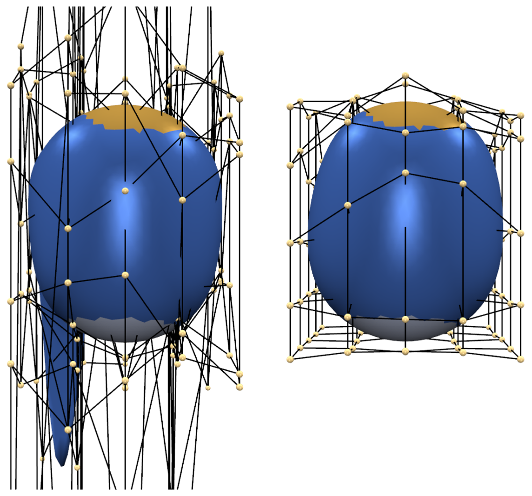

Fig. 8 Artifacts in the deformation of a sphere due to lack of clamping of singular values. DM-FFD with a $5 \times 5 \times 5$ control lattice. Different clamping values: 10^{-10} (left) and 10^{-2} (right). The handle displacements are exactly the same in both examples.

As a first benchmark we investigate the smoothness of the deformation techniques by analyzing the curvature of a surface mesh after deformation. More specifically, we consider mean curvature defined as

$$H = \frac{\kappa_1 + \kappa_2}{2},$$

where κ_1 and κ_2 are the principal (maximum and minimum) curvatures of the surface. Using the cotangent weight discretization of the Laplace-Beltrami operator [4] we compute the mean curvature on for a given vertex $\mathbf{x}_i$ of the mesh by

$$H(\mathbf{x}_i) = \frac{1}{2} \left\| \Delta \mathbf{x}_i \right\| .$$

For more details we refer the reader to [4]. A color-coded mean curvature visualization is shown in Figure 9 after performing a pre-defined deformation with both RBFs and DM-FFD. As can be seen from the visualization, DM-FFD suffers from aliasing artifacts due to its lattice-based nature. The same artifacts occur in control point FFD. In contrast, RBFs result in highly smooth deformations due to their built-in minimization of physically-inspired energies as described in Section 2.3.

FFD results in deformations that are not necessarily physically plausible. Especially its direct manipulation variant does not optimize for a high quality deformation, but for minimization of control point movement. In general, using a lattice-based method the shape of the deformation strongly depends on the resolution and form of the control lattice, as shown in Figure 10. Therefore, it becomes highly difficult to predict the shape resulting from a particular deformation setup in advance.

Another problem with lattice-based methods is the continuity in case of partial control grids. If the control grid covers only a subset of the object, non-smooth transitions between object points inside the control volume and those outside may occur (see Figure 11, center). In such cases additional sheets of control points have to be inserted in order to assure a smooth transition (Figure 11, right). This not only complicates the setup process of FFD, it also introduces unnecessary degrees of freedom due to bad adaptivity (see Section 3.4).

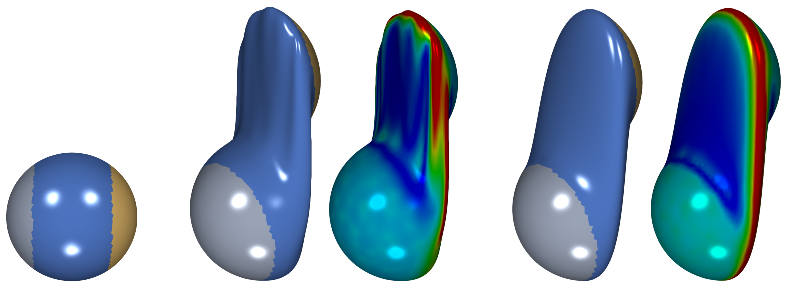

Fig. 9 Comparison of mesh smoothness after deformation based on mean curvature visualizations. Red indicates high curvature, blue low curvature. From left to right: Setup, deformed mesh and curvature visualizations for DM-FFD (729 control points) and RBFs (792 kernels).

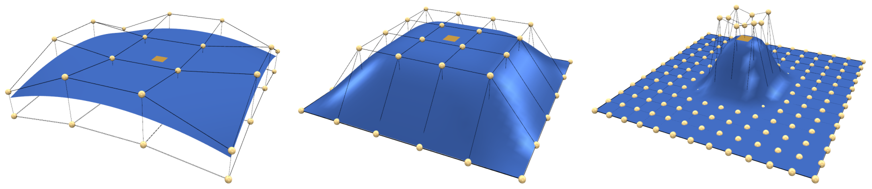

Fig. 10 Dependency of the deformation on the control lattice resolution. For all examples the same handle region was moved by the same translation.

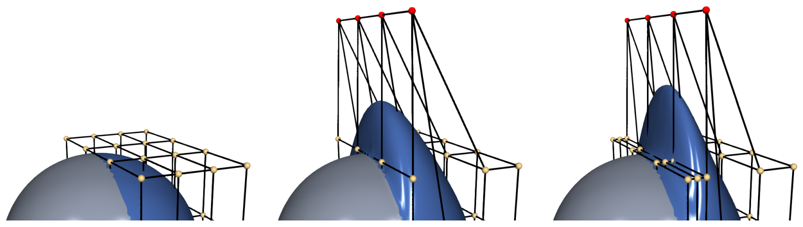

Fig. 11 Continuity problems in FFD in case of partial control volumes. From left to right: Original setup, non-smooth transition, smooth transition.

3.4 Adaptivity

In general, the adaptivity of a deformation method describes how well the method is capable of approximating a certain shape with an as low as possible number of degrees of freedom (DoFs). In the context of shape optimization the ability to dynamically add additional DoFs in regions of high interest is particularly important.

In order to evaluate the adaptivity we implemented a benchmark test that matches a source shape to a given target shape. In this test case, all vertices are prescribed as constraints, and the deformation method has to match the shape as closely as possible. For each of the methods we start with a low number of DoFs and successively refine the method to include more and more DoFs. We stop refinement once the number of DoFs is equal to the number of constraints.

Adaptivity can be measured best when approximating a target shape that is identical to the source shape for most vertices while having sharp local features in another region. The target shape is shown in Figure 12. This shape is particularly demanding since the transition from the plane to the feature area is very steep.

In case of DM-FFD we perform adaptive refinement by inserting additional control point planes in $x-$ and $y-$directions in those cells containing the vertex with the largest error. We do not perform refinement in z-direction, since in our example this would only result in wasted degrees of freedom. As becomes clear from Figure 12, the adaptivity of DM-FFD is generally poor, since it depends on the resolution of the lattice being used. While increasing the resolution of the lattice leads to sufficient degrees of freedom to approximate fine details as well, at the same time the insertion process also alters the deformation itself. An alternative to the current control point refinement would be to use knot insertion.

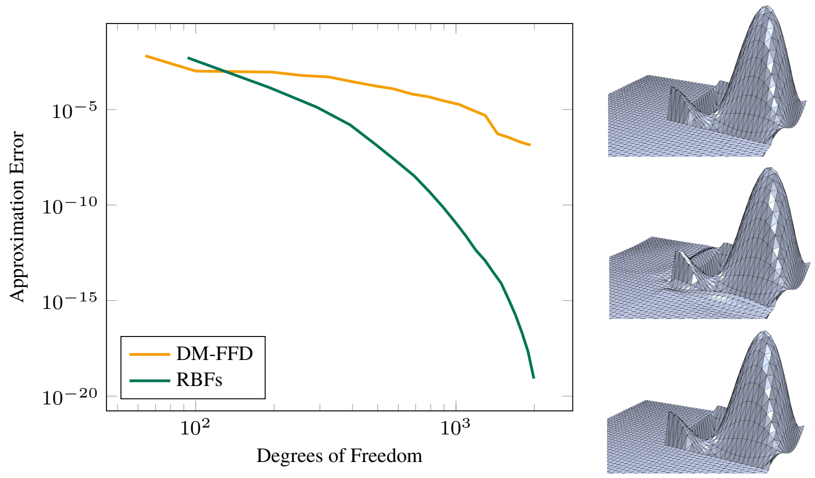

Fig. 12 Adaptive refinement benchmark results. Left: degrees of freedom vs. approximation error. Right: example results. From top to bottom: target shape, DM-FFD (900 DoF), and RBFs (993 DoF).

In case of RBFs we use straightforward adaptive greedy refinement [30]. Initially, we uniformly sample the plane with a given number of kernels. We then successively add additional kernels at the vertices of the mesh having the largest errors. The results in Figure 12 clearly confirm that RBFs provide superior approximation accuracy compared to DM-FFD.

3.5 Precision

The precision of a deformation method describes the accuracy in satisfying the positional constraints as prescribed by the user or optimization method. Typically, the accuracy is either *exact*, only provided in a *least-squares* sense, or only in a *qualitative* manner. Manipulating control points of a lattice as in case of FFD can only provide qualitative precision. Directly manipulated FFD improves on this by providing precision in a least-squares sense through the solution of (3). Finally, by solving (8) RBFs allow for exact satisfaction of constraints, thereby offering the highest level of precision. The quantitative results of Section 3.4 underline the differences in precision.

3.6 Volume Mesh Morphing

The ability to morph an existing volumetric simulation mesh according to a updated CAD geometry or alongside with a changed surface mesh is a crucial feature for deformation techniques in simulation-based design optimization: By avoiding costly (re-)meshing, it drastically reduces the computational cost, and it enables the construction of fully automatic optimization loops. This benchmark is particularly meaningful, since it accumulates results from the previous synthetic benchmarks. Even though both FFD and RBFs allow for volume mesh morphing due to their space-based nature, there are significant differences in the resulting mesh element quality. We present two different test scenarios involving three different mesh types: We investigate morphing of unstructured tetrahedral and structured hexahedral meshes according to an updated CAD geometry in Section 3.6.1. Finally, present a test-case of the DrivAer model involving an arbitrary polyhedral mesh to used CFD computations in Section 3.6.2.

In case of DM-FFD we generate a uniform control lattice enclosing the complete volume mesh. Unfortunately, the resolution required to satisfy given deformation constraints as precisely as possible is not known in advance and heavily depends on the complexity of the deformation and the geometry to be deformed. To accommodate for this, we investigate different grid resolutions, namely from 5^3, 10^3, 15^3, and 25^3 control points (referred to as DM-FFD-5/10/15/25 below). Therefore, the problem of automatic control grid generation is largely unsolved and a serious obstacle for fully automatic optimization procedures.

3.6.1 Pipe Model

In this section we investigate mesh quality based on the morphing benchmarks introduced in [36, 38]. Given an initial CAD surface and a volume mesh, shape variations are created by changing geometric parameters of the CAD model and computing updated surface nodes to match the new geometry. The surface nodes are then used as input to the morphing technique computing updated interior volume nodes. For a more detailed description of the benchmarks we refer to [36, 38]. We choose absolute morphing of the unstructured tetrahedral and structured hexahedral Pipe meshes as a representative example and present the results by means of minimum Scaled Jacobian in the volume mesh vs. percentage of parameter change in the CAD model in Figure 14. The plots show that RBFs better preserve element quality. Note that while the low resolution DM-FFD-5 test case results in more or less reasonable mesh quality, the constraints are not fulfilled exactly, i.e., the boundary nodes of the volume mesh do not match the update CAD surface (see Figure 13, right for an example). In contrast, higher resolutions are not capable of dealing with large changes due to the increasing locality of the deformation.

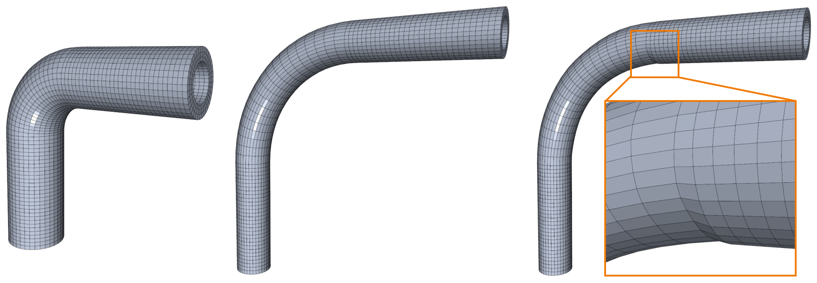

Fig. 13 Pipe model morphing examples. From left to right: Initial mesh, RBF, DM-FFD-5.

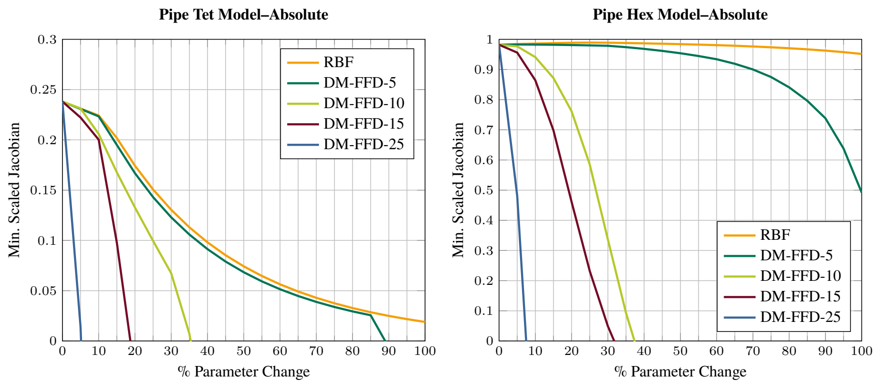

Fig. 14 Pipe model morphing results. Mesh quality vs. parameter change.

3.6.2 DrivAer Model

As an application-oriented benchmark we investigate an exemplary CFD test case for the DrivAer model. We use OpenFOAM [24] for the CFD setup and generate the volume mesh using the `snappyHexMesh` utility. The resulting arbitrary polyhedral mesh contains 1.2M cells and 1.6M points. To investigate the resulting mesh quality we use OpenFOAM's `checkMesh` tool, which analyzes general mesh properties, such as connectivity, ordering, and orientation, but also essential mesh quality characteristics, such as cell orthogonality, aspect ratio, and face skewness. For the deformation setup we select three handle vertices on the car roof while keeping the outer boundary of the volume mesh fixed. For the deformation itself we simply lift the handle vertices upwards.

A cut view of the resulting volume mesh and car surface patch is shown in Figure 15. A summary of results as obtained by OpenFOAM's `checkMesh` tool is given in Table 2. For a detailed description of the individual mesh checks we refer to the OpenFOAM [24] documentation. In case of RBFs the volume mesh is still usable and all mesh quality checks succeed. In case of DM-FFD-10 and DM-FFD-15 the meshes are still usable, but the cell orthogonality check warns about one non-orthogonal cell that might spoil the accuracy and/or convergence of the simulation. Furthermore, we note that for more complex deformations the 10^3 and 15^3 resolutions might not be sufficient to satisfy the displacement constraints with acceptable precision. In case of the higher resolution DM-FFD-25 setup several mesh quality checks fail and the mesh is no longer usable for simulation at all: The mesh contains 151 high aspect ratio cells, 1353 non-orthogonal faces, 1414 incorrectly oriented face pyramids, and 62 highly skewed faces. For all DM-FFD setups, Figure 15 again demonstrates the strong dependence of the resulting shape on the chosen control grid resolution.

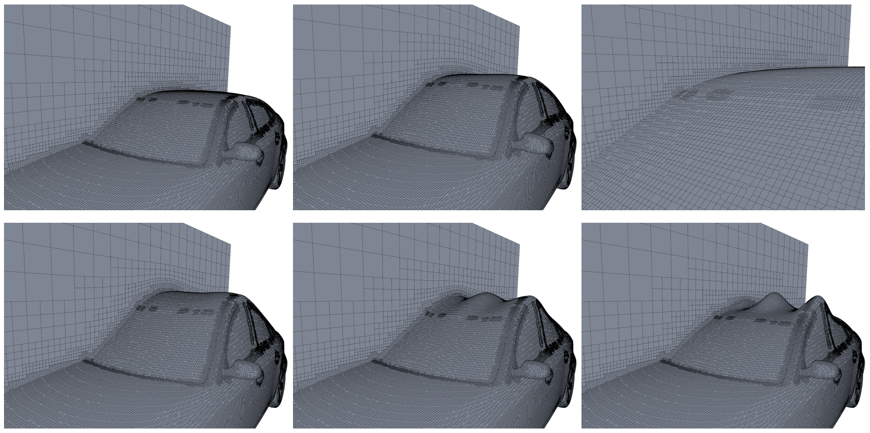

Fig. 15 Cut-view and car surface patch of the resulting volume mesh after deformation. Original (top left), RBF (top center), zoom to the surface (top right), DM-FFD (bottom row). In case of DM-FFD the results for control grid resolutions 10^3, 15^3, and 25^3 are shown.

	Aspect Ratio	Cell Orthogonality	Face Skewness	Face Pyramids
Original	6.9 ✓	64.7 ✓	3.4 ✓	✓
RBF	6.6 ✓	68.6 ✓	3.7 ✓	✓
DM-FFD-10	7.0 ✓	71.3 !	3.6 ✓	✓
DM-FFD-15	7.0 ✓	70.7 !	3.4 ✓	✓
DM-FFD-25	2.5e+195 ✗	179.7 ✗	1031.8 ✗	✗

Table 2 Results reported by OpenFOAM's `checkMesh`. Successful tests are indicated by a ✓, warnings by !, and errors by ✗. Numbers are given for the worst quality element in the mesh.

4 Summary & Conclusion

The results of the individual benchmarks show that there are significant differences between the deformation methods. A compact and simplified summary of the results is presented in Table 3. For each of the benchmarks and methods we assign either a positive (+), neutral (○), or a negative (−) assessment. Both FFD and DM-FFD achieve mixed results. While DM-FFD improves upon FFD in terms of precision, computational costs increase and robustness issues might occur. Both FFD techniques expose significant weaknesses with regards to adaptive refinement and quality of the deformation. In contrast, RBFs score the largest number of positive and only one neutral assessment.

However, the choice of a deformation method heavily depends on the needs of a given design optimization scenario. If the scenario neither demands for precise constraint satisfaction nor for adaptive refinement but only aims for general exploration of the design space, FFD offers a simple and robust deformation technique. In many cases, however, exact control is highly important in order obtain valid designs that meet production limitations such as keeping critical components fixed or deforming them only rigidly. In such cases, we clearly recommend RBFs over both FFD and its direct manipulation variant.

In some optimization scenarios the locality of the deformation might also be an important aspect. Both FFD methods allow for local deformations—depending on control grid resolution and setup as well as basis function degree. In contrast, our RBF deformations are global due to our choice of triharmonic basis functions. While there exist compactly supported RBFs [39], these basis functions lack the built-in energy minimization of (7). However, in many cases a proper setup of fixed and handle regions in the direct manipulation interface eventually provides a sufficient degree of locality.

Naturally, all of three methods can be enhanced in several ways. In case of both FFD methods the use of more flexible basis functions such as T-splines [32] or truncated hierarchical B-splines [9] would drastically improve the adaptivity of the respective methods. As for RBFs, constraining the deformation function to be positive—similar to the bounded biharmonic weights introduced in [14]—offers an interesting perspective for future work.

	Performance	Robustness	Quality	Adaptivity	Precision
FFD	○	+	○	−	−
DM-FFD	−	○	○	−	○
RBF	○	+	+	+	+

Table 3 Summary of results. For each benchmark test and deformation method we assign a negative (−), neutral (○), or a positive (+) assessment.

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). The authors kindly thank Matthew Staten from Sandia National Laboratories for originally providing us with Pipe models from [38].

References

1. Bechmann, D.: Space deformation models survey. Computers & Graphics **18**(4), 571 – 586 (1994)
2. de Boer, A., van der Schoot, M., Bijl, H.: Mesh deformation based on radial basis function interpolation. Computers & Structures **85**, 784–795 (2007)
3. Botsch, M., Kobbelt, L.: Real-time shape editing using radial basis functions. Computer Graphics Forum (Proc. Eurographics) **24**(3), 611–21 (2005)
4. Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., Levy, B.: Polygon Mesh Processing. AK Peters (2010)
5. Botsch, M., Sorkine, O.: On linear variational surface deformation methods. IEEE Transactions on Visualization and Computer Graphics **14**(1), 213–30 (2008)
6. Coquillart, S.: Extended free-form deformation: A sculpturing tool for 3D geometric modeling. In: Proc. of ACM SIGGRAPH, pp. 187–96 (1990)
7. Fasshauer, G.E.: Meshfree approximation methods with MATLAB. World Scientific Publishing (2007)
8. Gain, J., Bechmann, D.: A survey of spatial deformation from a user-centered perspective. ACM Transaction on Graphics **27**, 107:1–107:21 (2008)
9. Giannelli, C., Jüttler, B., Speleers, H.: THB–splines: The truncated basis for hierarchical splines. Computer Aided Geometric Design **29**, 485–498 (2012)
10. Golub, G.H., van Loan, C.F.: Matrix Computations. Johns Hopkins University Press, Baltimore (1989)
11. Griessmair, J., Purgathofer, W.: Deformation of Solids with Trivariate B-Splines. In: Proceedings of Eurographics (1989)
12. Heft, A.I., Indinger, T., Adams, N.A.: Introduction of a new realistic generic car model for aerodynamic investigations. In: SAE 2012 World Congress (2012)
13. Hsu, W.M., Hughes, J.F., Kaufman, H.: Direct manipulation of free-form deformations. In: Proc. of ACM SIGGRAPH, pp. 177–84 (1992)
14. Jacobson, A., Baran, I., Popović, J., Sorkine, O.: Bounded biharmonic weights for real-time deformation. ACM Transaction on Graphics **30**, 78:1–78:8 (2011)

15. Jakobsson, S., Amoignon, O.: Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization. Computers & Fluids **36**(6), 1119–1136 (2007)
16. Lamousin, H., Waggenspack, N.: NURBS-based free-form deformations. Computer Graphics and Applications, IEEE **14**(6), 59–65 (1994)
17. MacCracken, R., Joy, K.I.: Free-form deformations with lattices of arbitrary topology. In: Proc. of ACM SIGGRAPH, pp. 181–88 (1996)
18. Manzoni, A., Quarteroni, A., Rozza, G.: Shape optimization for viscous flows by reduced basis methods and free-form deformation. International Journal for Numerical Methods in Fluids **70**, 646–670 (2011)
19. Menzel, S., Olhofer, M., Sendhoff, B.: Application of free form deformation techniques in evolutionary design optimisation. In: Proceedings of the 6th World Congress on Structural and Multidisciplinary Optimization (2005)
20. Menzel, S., Olhofer, M., Sendhoff, B.: Direct manipulation of free form deformation in evolutionary design optimisation. In: International Conference on Parallel Problem Solving From Nature (PPSN), pp. 352–361 (2006)
21. Menzel, S., Sendhoff, B.: Representing the change–free form deformation for evolutionary design optimization. Studies in Computational Intelligence **88**, 63–86 (2008)
22. Michler, A.K.: Aircraft control surface deflection using RBF-based mesh deformation. International Journal for Numerical Methods in Engineering **88**(10), 986–1007 (2011)
23. Moccozet, L., Thalmann, N.: Dirichlet free-form deformations and their application to hand simulation. In: Computer Animation '97, pp. 93–102 (1997)
24. OpenFOAM: Open Source Field Operation and Manipulation C++ libraries. `http://www.openfoam.org` (2012)
25. OpenMP Architecture Review Board: OpenMP application program interface version 3.1 (2011). URL `http://www.openmp.org/mp-documents/OpenMP3.1.pdf`
26. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipes: The Art of Scientific Computing, 2nd edn. Cambridge University Press (1992)
27. Samareh, J.A.: A survey of shape parameterization techniques. Tech. Rep. NASA/CP-1999-209136/PT1, NASA Langley Research Center (1999)
28. Samareh, J.A.: Aerodynamic shape optimization based on free-form deformation. In: Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference (2004)
29. Sapidis, N.S.: Designing Fair Curves and Surfaces: Shape Quality in Geometric Modeling and Computer-Aided Design. Society for Industrial and Applied Mathematics (1994)
30. Schaback, R., Wendland, H.: Adaptive greedy techniques for approximate solution of large RBF systems. Numerical Algorithms **24**(3), 239–254 (2000)
31. Sederberg, T.W., Parry, S.R.: Free-form deformation of solid geometric models. In: Proc. of ACM SIGGRAPH, pp. 151–59 (1986)
32. Sederberg, T.W., Zheng, J., Bakenov, A., Nasri, A.: T-splines and T-NURCCs. ACM Transaction on Graphics **22**(3), 477–484 (2003)
33. Sibson, R.: A vector identity for the dirichlet tessellation. In: Mathematical Proceedings of the Cambridge Philosophical Society, vol. 87, pp. 151–155 (1980)
34. Sieger, D., Menzel, S., Botsch, M.: A comprehensive comparison of shape deformations in evolutionary design optimization. In: Proceedings of the 3rd International Conference on Engineering Optimization (2012)
35. Sieger, D., Menzel, S., Botsch, M.: High quality mesh morphing using triharmonic radial basis functions. In: Proceedings of the 21st International Meshing Roundtable (2012)
36. Sieger, D., Menzel, S., Botsch, M.: RBF morphing techniques for simulation-based design optimization. Engineering with Computers (2013). To appear
37. Song, W., Yang, X.: Free-form deformation with weighted T-spline. The Visual Computer **21**, 139–151 (2005)
38. Staten, M.L., Owen, S.J., Shontz, S.M., Salinger, A.G., Coffey, T.S.: A comparison of mesh morphing methods for 3D shape optimization. In: Proceedings of the 20th International Meshing Roundtable, pp. 293–311 (2011)
39. Wendland, H.: Scattered Data Approximation. Cambridge University Press, Cambridge, UK (2005)