EngOpt 2012 - International Conference on Engineering Optimization
Rio de Janeiro, Brazil, 1-5 July 2012.

A Comprehensive Comparison of Shape Deformation Methods in Evolutionary Design Optimization

Daniel Sieger[1], Stefan Menzel[2], and Mario Botsch[1]

[1] Bielefeld University, Bielefeld, Germany, {dsieger,botsch}@techfak.uni-bielefeld.de
[2] Honda Research Institute Europe, Offenbach, Germany, stefan.menzel@honda-ri.de

Abstract
We introduce and investigate several state-of-the-art shape deformation methods for their use in evolutionary design optimization problems. Starting from classical free-form deformation (FFD) we broaden our investigation to more flexible methods such as direct manipulation FFD and deformations based on radial basis functions. We integrate the different methods into a framework for evolutionary design optimization and apply them in a passenger car optimization scenario improving the drag of a simplified Honda Civic model. We evaluate the aerodynamic performance of different design variations utilizing computational fluid dynamics simulations. Based on this scenario we analyze and compare the strengths and weaknesses of the individual methods.

Keywords: Shape deformation, evolutionary design optimization, computational fluid dynamics.

1. Introduction

Fully automatic simulation-based design optimization has become a crucial part of the product development cycle in a wide variety of domains such as automobile and aerospace engineering, electronics, or naval architecture. Its primary goal is the performance improvement of certain physical properties of real world objects. The success and efficiency of such an optimization process heavily depends on the carefully tuned interplay of three major components: the chosen optimization algorithm, the method to create design variations, and the design evaluation method.

While classical gradient-based methods have been extensively used for design optimization, the application of evolutionary algorithms becomes more and more feasible due to increasing computing power and availability of parallel computing environments. Evolutionary optimization techniques based on principles of biological evolution have several compelling advantages: the ability to find global optima, the potential to generate new and unexpected designs, robustness with regards to noise in the input data, the ability to deal with non-smooth or even discontinuous fitness functions, as well as the possibility to incorporate multiple objective fitness functions at once.

As for creating variations of an initial design, shape deformation methods originating from computer graphics and geometric modeling have been successfully integrated into a design optimization framework using evolutionary algorithms. In particular, the combination of free-form deformation (FFD) and evolution strategies (ES) has been proven to be a powerful tool for improving the aerodynamic performance of complex geometric designs. Within such an optimization framework the deformation method allows to manipulate a given shape by adjusting a set of control points that constitute the design parameters determined by the selected evolutionary algorithm.

As for design evaluation, typical choices include computational fluid dynamics (CFD) simulations based on finite volume methods for aerodynamic performance calculation, as well as finite element methods (FEM) for structural mechanics simulations. However, in order to produce meaningful results the fitness function is typically enhanced to include additional measures and constraints, e.g., to maintain geometric characteristics like minimum heights, curvatures, or distances.

The remainder of 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 several state-of-the-art shape deformation methods including FFD, direct manipulation FFD, and deformations based on radial basis functions. A generic evolutionary design optimization process as well as basic concepts of evolution strategies are introduced in Section 3. We apply this optimization process in a passenger car design optimization scenario utilizing CFD simulations for aerodynamic performance calculation in Section 4, which also serves us as a starting point for comparing the different deformation methods.

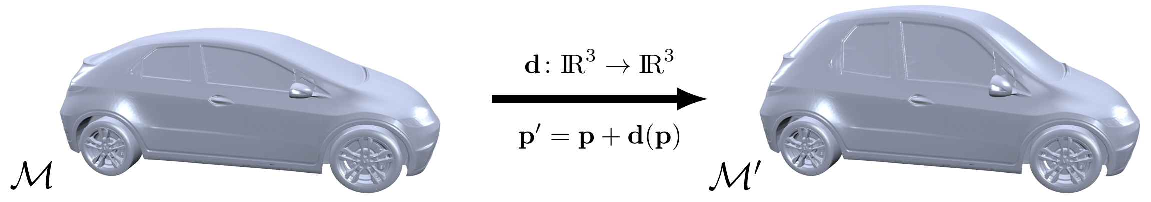
$\mathbf{d}\colon \mathbb{R}^3 \to \mathbb{R}^3$
$\mathbf{p}' = \mathbf{p} + \mathbf{d}(\mathbf{p})$
$\mathcal{M}$
$\mathcal{M}'$

Figure 1: Deformation of a simplified Honda Civic. The model $\mathcal{M}$ is warped by the space deformation function $\mathbf{d}$. Each point $\mathbf{p} \in \mathcal{M}$ is transformed to updated locations $\mathbf{p}' \in \mathcal{M}'$, where $\mathbf{p}' = \mathbf{p} + \mathbf{d}(\mathbf{p})$.

2. Shape Deformation Methods

In this section we introduce several state-of-the-art shape deformation methods for their use in evolutionary 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 [5]. 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 the context of design optimization is given in [25]. Linear variational surface deformation methods have been investigated in detail in [6].

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 computational fluid dynamics (CFD) or finite element method (FEM) simulations. In this case the volumetric grids required for the simulation typically are very time-consuming to generate. For highly complex geometries this process might even involve manual interaction by the engineer, which is prohibitive for a fully automatic design optimization process. In order to avoid the costly mesh generation process for each design variation created during optimization, one typically aims at deforming an initial simulation grid alongside with the object. A second requirement is the ability to robustly deal with defects in the input geometry. Especially models originating from an automatic conversion process of a CAD system may contain degeneracies such as badly shaped triangles, non-manifold meshes, 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 $\mathbf{p}' = \mathbf{p} + \mathbf{d}(\mathbf{p})$ for each original point $\mathbf{p} \in \mathcal{M}$. The basic procedure is illustrated in Figure 1. Naturally, space deformation techniques differ in how the function $\mathbf{d}$ is constructed. Since the deformations applied during optimization are typically relatively small, we focus on linear methods, i.e., techniques that require solving a linear system in one form or another. A common denominator for all methods is the use of some form of control structure, be it a volumetric lattice, a triangular surface mesh, or a set of points in space. The deformation function is typically constructed by blending the elements of the control structure in conjunction with some form of basis functions, as we will show in detail for the individual methods in the sections that follow.

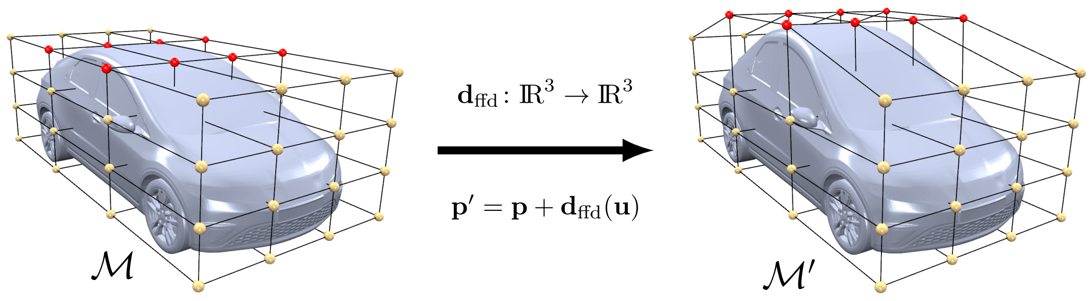
$\mathbf{d}_{\text{ffd}} : \mathbb{R}^3 \to \mathbb{R}^3$
$\mathbf{p}' = \mathbf{p} + \mathbf{d}_{\text{ffd}}(\mathbf{u})$
$\mathcal{M}$
$\mathcal{M}'$

Figure 2: Free-form deformation applied to a simplified Honda Civic. 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 $\mathbf{p}'$ are computed by evaluating the FFD space deformation function $\mathbf{d}_{\text{ffd}}$ for the local coordinates $\mathbf{u}$ of the point $\mathbf{p}$.

2.1. Free-form Deformation

Free-form deformation 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 [17, 26] and evolutionary design optimization [20, 19, 18] 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 [28]. Local deformations using B-spline basis functions have been introduced in [10]. An extension to more flexible control lattices, in particular cylindrical ones, has been proposed in [7]. This approach was later extended to control lattices of arbitrary topology in [16]. Free-form deformations using non-uniform rational B-splines are described in [15]. A highly flexible but computationally involved variant of FFD based on a 3D-Delaunay triangulation, its Voronoi dual, and Sibson coordinates [29] has been presented in [21]. A variant of FFD using T-splines [27] as basis functions—thereby allowing for local refinement of the control lattice—has been presented in [30].

The basic idea of FFD is based upon 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 $\mathbf{p} \in \mathcal{M}$ can be expressed as a linear combination of lattice control points $\mathbf{c}_{ijk}$ and basis functions N_i:

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

where (u_1, u_2, u_3) are the local coordinates of $\mathbf{p}$ 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

$$\mathbf{u} := (u_1, u_2, u_3), \quad N_p(\mathbf{u}) := N_i(u_1) N_j(u_2) N_k(u_3), \quad \text{and} \quad \delta\mathbf{c}_p := \delta\mathbf{c}_{ijk} = \mathbf{c}'_{ijk} - \mathbf{c}_{ijk},$$

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

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

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 a numerical technique such as a Newton method [23] for computing the local coordinates, the important advantage is the capability to perform deformations with local support.

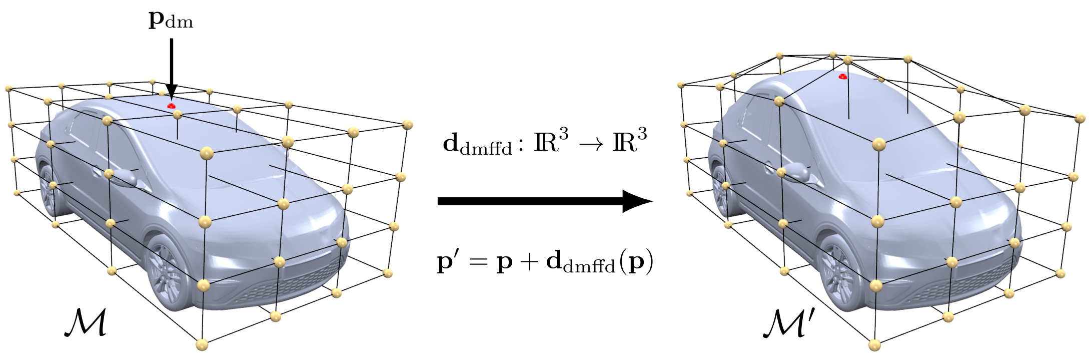
$\mathbf{p}_{\text{dm}}$
$\mathbf{d}_{\text{dmffd}}: \mathbb{R}^3 \to \mathbb{R}^3$
$\mathbf{p}' = \mathbf{p} + \mathbf{d}_{\text{dmffd}}(\mathbf{p})$
$\mathcal{M}$
$\mathcal{M}'$

Figure 3: Direct manipulation FFD of a simplified Honda Civic. A single object point $\mathbf{p}_{\text{dm}}$ (marked red) is selected from the vertices of the model $\mathcal{M}$. After specifying a displacement for $\mathbf{p}_{\text{dm}}$ the system computes control point displacements so that $\mathbf{p}'_{\text{dm}}$ is satisfied and the updated object point locations $\mathbf{p}'$ are computed.

2.2. Direct Manipulation Free-form Deformation (DMFFD)

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 the 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 [12]. 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 a Civic 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 evolutionary design optimization, as has been shown for direct manipulation FFD in [19]. 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 the context of evolutionary optimization this aspect is also referred to as strong causality [24]. 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 displacement constraints, i.e., the deformation function has to attain certain values $\mathbf{d}(\mathbf{u}_i) = \bar{\mathbf{d}}_i$ for the constraints $\{\mathbf{p}_1, \ldots, \mathbf{p}_m\}$. The control point displacements $\delta\mathbf{c}$ satisfying the prescribed displacements can be computed by solving the linear system

$$\begin{bmatrix} N_1(\mathbf{u}_1) & \ldots & N_n(\mathbf{u}_1) \\ \vdots & \ddots & \vdots \\ N_1(\mathbf{u}_m) & \ldots & N_n(\mathbf{u}_m) \end{bmatrix} \begin{pmatrix} \delta\mathbf{c}_1^T \\ \vdots \\ \delta\mathbf{c}_n^T \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{d}}_1^T \\ \vdots \\ \bar{\mathbf{d}}_m^T \end{pmatrix}. \tag{3}$$

Since the matrix in (3) is singular, the system can be solved using the *pseudo-inverse* [12, 9]. However, solving for $\delta\mathbf{c}$ using the pseudo-inverse has its drawbacks. When 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. When 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, the solution does not necessarily result in a physically plausible deformation.

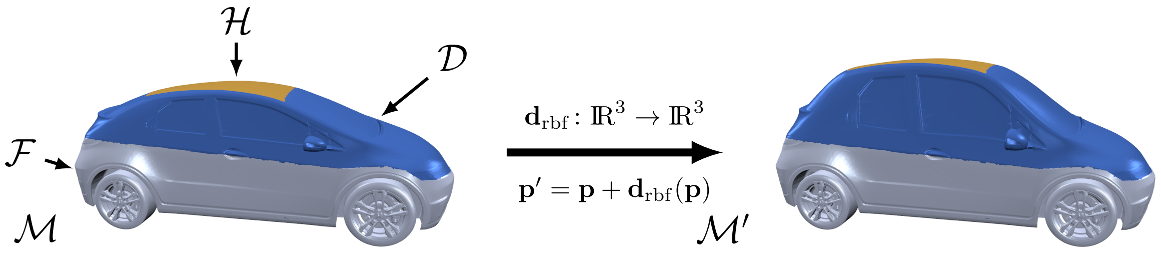
$\mathcal{H}$
$\mathcal{D}$
$\mathcal{F}$
$\mathcal{M}$
$\mathbf{d}_{\text{rbf}}\colon \mathbb{R}^3 \to \mathbb{R}^3$
$\mathbf{p}' = \mathbf{p} + \mathbf{d}_{\text{rbf}}(\mathbf{p})$
$\mathcal{M}'$

Figure 4: Deformation of a simplified Honda Civic using a handle-based direct manipulation interface for RBFs. The vertices of the model $\mathcal{M}$ are classified into three distinct sets: Handle vertices ($\mathcal{H}$, golden) can be directly displaced, fixed vertices ($\mathcal{F}$, grey) are kept in place, and deformable vertices ($\mathcal{D}$, blue) are updated according to the deformation method.

2.3. Deformations Based on Radial Basis Functions

A space deformation technique based on radial basis functions (RBFs) has been introduced in [4]. This method improves upon FFD and DMFFD in two significant aspects: First, due to its point-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 physically inspired energies—resulting in a smooth and physically plausible deformation. Earlier instances of RBF deformations [3] lack this crucial property. A conceptually similar approach that enforces additional properties on the deformation has been introduced by [14]. An example deformation using RBFs as well as an explanation of our handle-based interface is given in Figure 4.

The RBF deformation technique is motivated by treating space deformation as an abstract interpolation problem: Given a set of displacements $\bar{\mathbf{d}}_i$ at positions $\mathbf{p}_i$, the goal is to smoothly interpolate displacements through space. It is well-known from the field of scattered data interpolation and approximation that RBFs are highly suitable for solving this type of problem [31]. Within the RBF deformation approach the space deformation function is constructed as a linear combination of radially symmetric kernels $\varphi_j(\mathbf{p}) = \varphi(\|\mathbf{c}_j - \mathbf{p}\|)$, located at centers $\mathbf{c}_j \in \mathbb{R}^3$ and weighted by $\mathbf{w}_j \in \mathbb{R}^3$:

$$\mathbf{d}_{\text{rbf}}(\mathbf{p}) \;=\; \sum_{j=1}^{m} \mathbf{w}_j \varphi_j(\mathbf{p}) + \boldsymbol{\pi}(\mathbf{p}), \tag{4}$$

where $\boldsymbol{\pi}(\mathbf{p})$ is a polynomial of low degree used to guarantee polynomial precision. The choice of the kernel function $\varphi\colon \mathbb{R} \to \mathbb{R}$ has a significant influence on the quality of the deformation. In our implementation we choose $\varphi(r) = r^3$, which results in a tri-harmonic deformation function, i.e., $\Delta^3\mathbf{d} = 0$, where Δ is the Laplace operator. Therefore, the deformation function $\mathbf{d}$ minimizes a fairness energy [4, 5]:

$$\int_{\mathbb{R}^3} \left\| \frac{\partial^3 \mathbf{d}}{\partial x^3} \right\|^2 + \left\| \frac{\partial^3 \mathbf{d}}{\partial x^2 \partial y} \right\|^2 + \ldots + \left\| \frac{\partial^3 \mathbf{d}}{\partial z^3} \right\|^2 \mathrm{dV}.$$

A second important aspect of RBF deformations is the kernel placement. By directly placing kernels on all handle and fixed vertices we can easily implement a handle-based direct manipulation interface satisfying all constraints specified by the user easily and *exactly*. If required, the performance of this approach can be drastically improved by using only a reduced set of constraints, namely those that are close to the deformable region. Finally, in order to compute the RBF deformation $\mathbf{d}_{\text{rbf}}$ the basis function weights $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_m)^T \in \mathbb{R}^{m\times 3}$ and polynomial coefficients $\mathbf{Q} = (\mathbf{q}_1, \ldots, \mathbf{q}_4) \in \mathbb{R}^{4\times 3}$ have to be computed by solving the dense symmetric linear system

$$\begin{pmatrix} \boldsymbol{\Phi} & \boldsymbol{\Pi} \\ \boldsymbol{\Pi}^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{W} \\ \mathbf{Q} \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{D}} \\ \mathbf{0} \end{pmatrix}, \tag{5}$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{m\times m}$ with $\boldsymbol{\Phi}_{ij} = \varphi_j(\mathbf{p}_i)$, $\boldsymbol{\Pi} \in \mathbb{R}^{m\times 4}$ with $\boldsymbol{\Pi}_{ij} = \pi_j(\mathbf{p}_i)$, $\bar{\mathbf{D}} = (\bar{\mathbf{d}}_1, \ldots, \bar{\mathbf{d}}_m)^T \in \mathbb{R}^{m\times 3}$, and $\{\pi_1, \ldots, \pi_4\}$ being a basis of the space of trivariate linear polynomials Π.

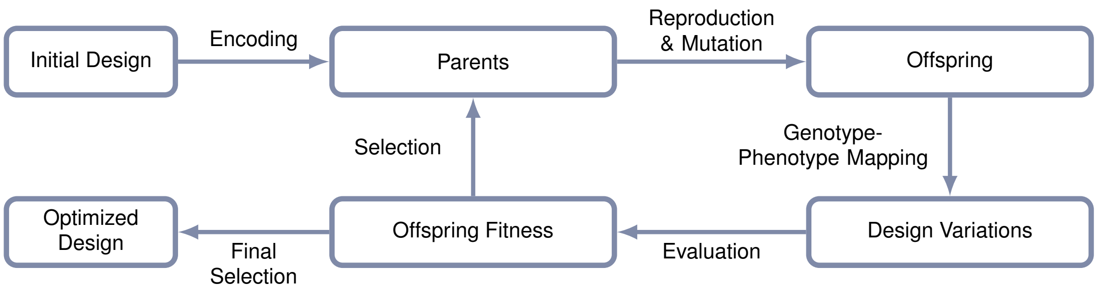

Figure 5: Overview about a generic evolutionary design optimization process.

3. Evolutionary Design Optimization

Evolutionary algorithms employ principles of biological evolution such as reproduction and mutation for solving global optimization problems in a stochastical manner. Widely used techniques include genetic algorithms, genetic programming, evolutionary programming, and evolution strategies. The general approach behind these techniques is to successively improve the fitness of a starting population by adapting its parameters. On a general level, evolutionary algorithms have several advantages, such as the ability to find global optima, the capability to generate novel and unexpected designs, robustness to noise and uncertainty, as well as the ability to deal with non-smooth, discontinuous, and multi-objective fitness functions.

An overview about a generic evolutionary design optimization process is illustrated in Figure 5. The initial design is encoded into a parent chromosome. A set of offspring chromosomes is created by means of reproduction and mutation. By mapping the offspring's genotype to its phenotype a set of design variations is created. The new designs are evaluated with regards to a specific fitness function. The most successful offspring are selected to be the parents of the next generation and the evolution cycle starts anew. This process is repeated until a desired fitness value is reached, the optimization converges, or a maximum number of generations is reached.

Evolution strategies (ES) are a prominent class of evolutionary algorithms that can be employed to solve a wide variety of optimization problems. Potential solutions are typically represented using vectors of real numbers. After copying the parent chromosome for reproduction the offspring chromosomes are mutated by adding a normally distributed random vector with zero mean. In comparison to other evolutionary algorithms ES have two significant advantages: First, the strategy parameters can be adapted during optimization (self-adaptation)—leading to faster convergence. Second, due to its simple vector-based encoding, incorporating constraints on the parameters is much simpler than in other methods. A comprehensive introduction to evolution strategies and its variants is given in [2].

A state-of-the-art variant of ES that provides high convergence rates and is able to deal with small population sizes is the covariance matrix adaptation evolution strategy (CMA-ES) [11]. This property is particularly important when using a computationally expensive fitness functions such as large-scale CFD or FEM simulations. The characteristics that enable CMA-ES to provide fast convergence on small populations are utilization of one source for object and strategy parameter variation, cumulative step size adaptation, and adaptation of the full covariance matrix for adjusting the random distribution of candidate solutions towards previously successful ones, as illustrated schematically in Figure 6.

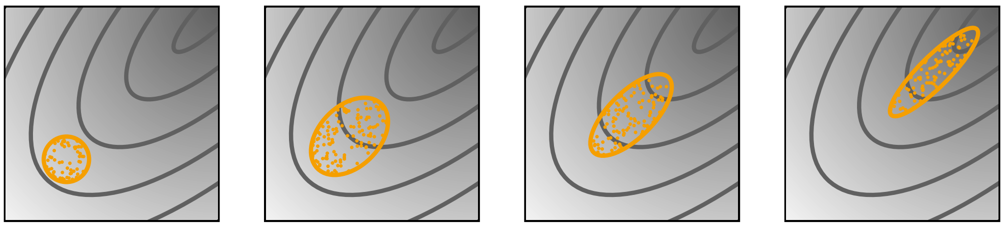

Figure 6: Incremental adaptation of the random distribution of candidate solutions in CMA-ES.

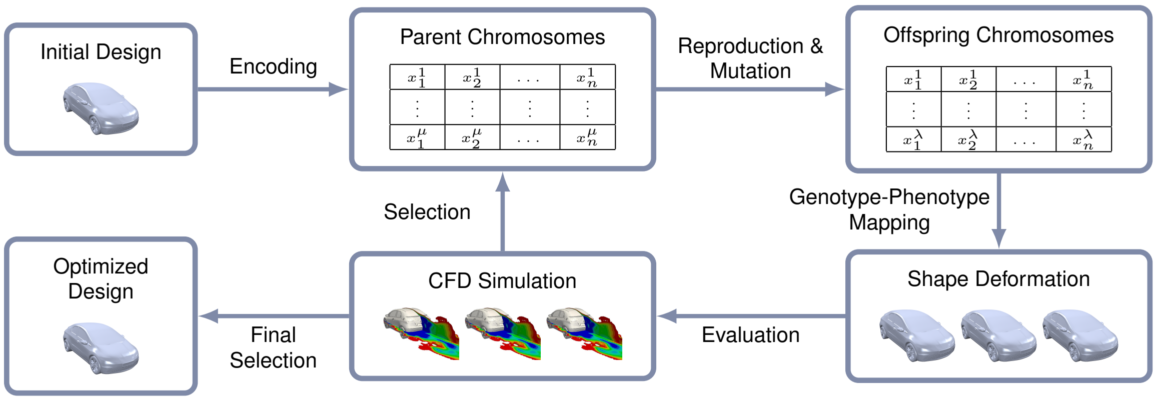

Figure 7: Overview about the evolutionary design optimization process.

4. Application: Passenger Car Design Optimization

We apply the different deformation methods in a passenger car design optimization scenario improving the aerodynamic performance of a simplified Honda Civic. This scenario allows us to illustrate and compare the individual strengths and weaknesses of the different deformation methods. The setup of our evolutionary design optimization loop is illustrated in Figure 7. We use the CMA-ES provided by the Shark machine learning library [13] and employ a $(2, 15)$-strategy, i.e., two parents and 15 offspring individuals are considered. The selection is only performed on the offspring individuals. We encode the design using $n = 23$ parameters. The feasible range of each parameter was constrained in order to prevent the design to become too flat or too far stretched out.

4.1. Deformation Setups

The different deformation methods were set up in order to deform the back part of the Civic model. In case of FFD a control grid was generated and pre-deformed to match the shape of the car. Several control point groups were defined to be deformed according to the same parameter. Certain control points on the boundary of the grid have been kept fixed in order to prevent discontinuities in the boundary region. The same control grid was used in DMFFD. However, instead of using individual control point groups all non-constrained control points were allowed to move in order to satisfy the prescribed object point displacements. The object points that were displaced were chosen to approximately reflect the FFD control point groups. Similar to DMFFD, a set of 20 handle regions was selected for the RBF deformation method. The initial setups of the different deformation methods are shown in Figure 8. The number of vertices being deformed was about 54k for all deformation methods.

4.2. Genotype-Phenotype Mapping

The genotype of an offspring chromosome is mapped to its phenotype by establishing a correspondence between each component of the chromosome and a displacement into one spatial direction. Depending on the deformation method a given control point group (FFD), group of object points (DMFFD), or a handle region (RBF) is displaced by the parameter. In case of RBFs, e.g., the first parameter corresponds to the vertical displacement of the top roof handle region.

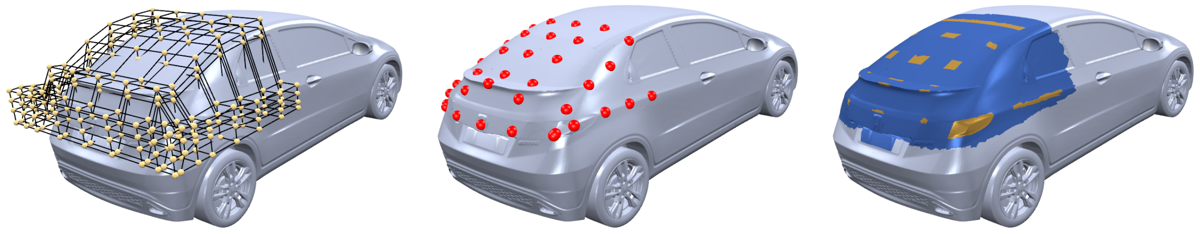

Figure 8: Initial setups of the different deformation methods: FFD, DMFFD, RBFs.

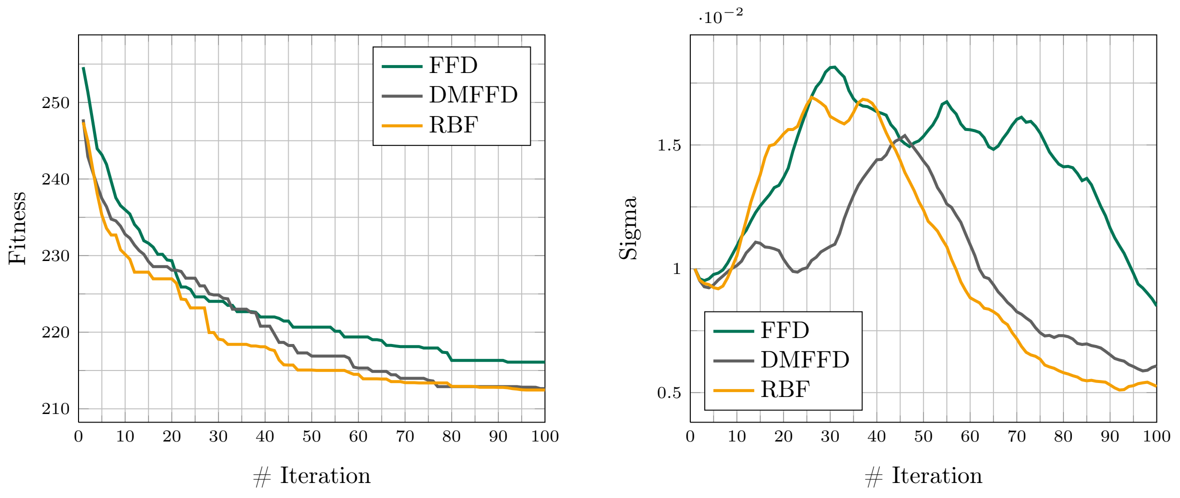

Figure 9: Progression of best solution fitness (left) and step size (right) during optimization.

4.3. Fitness Function Evaluation

We evaluate the individual designs based on a fitness function $f\colon \mathbb{R}^n \to \mathbb{R}$, combining two different performance values: $f(\mathbf{x}) = w_1 v_1 + w_2 v_2$, where v_1 is the aerodynamic drag as computed by the CFD simulation and v_2 is an additional volume weight to prevent the optimization from producing overly flat shapes. The aerodynamic drag v_1 is computed by solving the incompressible laminar Navier-Stokes equations using the SIMPLE algorithm as provided by the OpenFOAM C++ libraries [22]. The simulation mesh contained 28033 points, 24576 cells, and a total of 77056 faces. For each of the 15 offspring individuals the CFD simulation was run in parallel on 4 processor cores using OpenMPI in a cluster environment with systems containing two 2.4GHz Quad Core Xeon processors with 24GB RAM. The volume weight was computed by $v_2 = 1/\left\|\mathbf{p}_{\max} - \mathbf{p}_{\min}\right\|$, where $\mathbf{p}_{\max}$ and $\mathbf{p}_{\min}$ are the maximum and minimum points of the bounding box of deformable object points. To increase the impact of the volume weight in a reasonable manner we chose $w_1 = 1$ and $w_2 = 50$ as weights.

4.4. Results

The progression of the best solution fitness and of the step size is shown in Figure 9. For each of the deformation methods the optimization has been run for 100 generations. Examples of the designs with the best fitness values are shown for each deformation method in Figure 10. We note that the resulting deformations of the simplified car are purely artificial and have no practically relevant background. The overall run time was approximately two weeks for each method. As can be seen from Figure 9 (left), both DMFFD and RBF yield a better solution fitness than FFD. However, as becomes clear from Figure 9 (right), the step size of FFD is not as close to convergence as for the other methods, which might be an explanation for the higher fitness value. However, the step sizes of all optimizations did not fully converge within the number of iterations performed, which is mainly due to the initial step size being too small—a common problem when dealing with an unknown objective function.

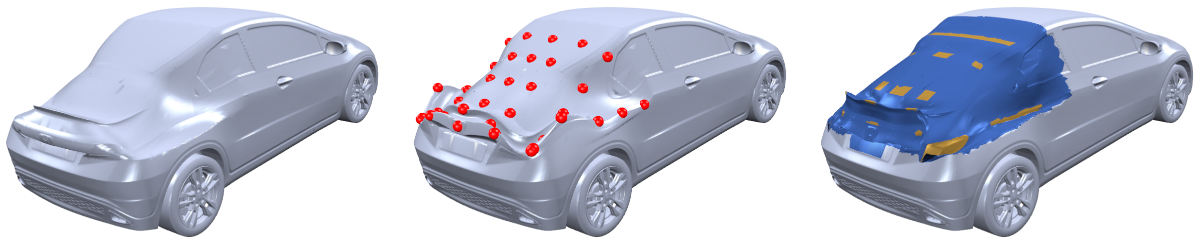

Figure 10: Best designs obtained with the different deformation methods: FFD, DMFFD, RBFs.

5. Conclusion & Future Work

From our application scenario we can draw two major conclusions: First, we can affirm that in contrast to lattice-based free-form deformation methods radial basis functions are significantly easier and flexible to set up while providing equivalent or better results. The manual adaptation of the control lattice to the shape of the model took a significant amount of time, while selecting the handle regions for the RBF method was rather straightforward. Our second conclusion is that the slightly better results of both methods directly manipulating object points seems to confirm the importance of a strong coupling between the optimization parameter and resulting phenotype within evolutionary optimization.

In a future work we intend to extend our comparison in several ways. First, additional methods such as cage-based deformations should be included. Second, a unified handle-based direct manipulation interface should be used for all methods in order to increase comparability. Third, additional synthetic benchmarks regarding computational performance, numerical robustness, quality and smoothness of the deformation, precision of constraint satisfaction, and adaptive refinement should be performed and analyzed. Fourth, depending on the desired behavior for the RBF deformation different choices of basis functions should be investigated. Finally we note that even though the range of parameters in the optimization was heavily constrained and the change of volume was taken into account for fitness evaluation the resulting optimized shapes do not directly provide design alternatives ready for production. We therefore feel that additional constraints yielding more meaningful results should be directly integrated into the deformation.

6. Acknowledgments

Daniel Sieger gratefully acknowledges the financial support from the Honda Research Institute Europe. Mario Botsch is supported by the German Research Foundation (Center of Excellence in “Cognitive Interaction Technology”, CITEC).

References

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

[2] H. G. Beyer and H. P. Schwefel. Evolution strategies—a comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[3] P. Borrel and A. Rappoport. Simple constrained deformations for geometric modeling and interactive design. *ACM Trans. Graph.*, 13(2):137–155, 1994.

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

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

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

[7] S. Coquillart. Extended free-form deformation: A sculpturing tool for 3D geometric modeling. In *Proc. of ACM SIGGRAPH*, pages 187–96, New York, 1990. ACM.

[8] J. Gain and D. Bechmann. A survey of spatial deformation from a user-centered perspective. *ACM Trans. Graph.*, 27(4):107.1–107.21, 2008.

[9] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, 1989.

[10] J. Griessmair and W. Purgathofer. Deformation of Solids with Trivariate B-Splines. In *Proceedings of Eurographics*, 1989.

[11] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[12] W. M. Hsu, J. F. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. In *Proc. of ACM SIGGRAPH*, pages 177–84, New York, 1992. ACM.

[13] C. Igel, V. Heidrich-Meisner, and T. Glasmachers. Shark. *Journal of Machine Learning Research*, 9:993–996, 2008.

[14] A. Jacobson, I. Baran, J. Popović, and O. Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.*, 30(4):78.1–78.8, 2011.

[15] H. J. Lamousin and N. N. Waggenspack. NURBS-based free-form deformations. *IEEE Computer Graphics and Applications*, 14(6):59–65, 1994.

[16] R. MacCracken and K. I. Joy. Free-form deformations with lattices of arbitrary topology. In *Proc. of ACM SIGGRAPH*, pages 181–88, New York, 1996. ACM.

[17] A. Manzoni, A. Quarteroni, and G. Rozza. Shape optimization for viscous flows by reduced basis methods and free-form deformation. *International Journal for Numerical Methods in Fluids*, 2011.

[18] S. Menzel, M. Olhofer, and B. Sendhoff. Application of free form deformation techniques in evolutionary design optimisation. In *6th World Congress on Structural and Multidisciplinary Optimization (WCSMO6)*, 2005.

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

[20] S. Menzel and B. Sendhoff. Representing the change – free form deformation for evolutionary design optimisation. In *Evolutionary Computation in Practice*, Studies in Computational Intelligence, pages 63–86. Springer, 2008.

[21] L. Moccozet and N. M. Thalmann. Dirichlet free-form deformations and their application to hand simulation. In *Computer Animation '97*, pages 93–102, 1997.

[22] OpenFOAM. Open Source Field Operation and Manipulation C++ libraries. `http://www.openfoam.org`, 2012.

[23] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, Cambridge, UK, 2nd edition, 1992.

[24] I. Rechenberg. *Evolutionsstrategie '94*, volume 1 of *Werkstatt Bionik und Evolutionstechnik.* Frommann-Holzboog, 1994.

[25] J. A. Samareh. A survey of shape parameterization techniques. Technical Report NASA/CP-1999-209136/PT1, NASA Langley Research Center, 1999.

[26] J. A. Samareh. Aerodynamic shape optimization based on free-form deformation. In *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2004.

[27] T. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-NURCCs. *ACM Trans. Graph.*, 22(3):477–84, 2003.

[28] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *Proc. of ACM SIGGRAPH*, pages 151–59, New York, 1986. ACM.

[29] R. Sibson. A vector identity for the Dirichlet tessellation. *Mathematical Proceedings of the Cambridge Philosophical Society*, 87(1):151–155, 1980.

[30] W. Song and X. Yang. Free-form deformation with weighted T-spline. *The Visual Computer*, 21(3):139–151, April 2005.

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