Seventh International Conference on Computational Fluid Dynamics (ICCFD7)
Big Island, Hawaii, July 9-13, 2012

ICCFD7-2001

Constraint-based Shape Parameterization for Aerodynamic Design

G. R. Anderson[*], M. J. Aftosmis[†] and M. Nemec[‡]
Corresponding author: george.anderson@stanford.edu

[*] Ph.D. Candidate, Aeronautics and Astronautics, Stanford University, Stanford, CA, USA
[†] Aerospace Engineer, Advanced Supercomputing Division, NASA Ames, Moffett Field, CA, USA
[‡] Research Scientist, Science and Technology Corp., Moffett Field, CA, USA

Abstract: We present a method for constructing design-appropriate shape parameterizations of discrete surfaces for optimization of flight vehicles. The method allows the designer to choose a set of points located on the aerodynamic surface to serve as design variables. Arbitrary geometric constraints can be prescribed and are intrinsically incorporated into the parameterization. The technique supports both fine-scale control and more rigid, large-scale deformation modes, enabling the recovery of standard wing design parameters. We show that local shape features, such as airfoil sections, can be preserved even under large-scale deformation. Deformations of high-resolution surfaces typically require less than a second to compute. Parameterizations can be modified and refined at design time, allowing for incorporation of new design variables and constraints throughout the design process. We demonstrate the technique on demanding 3D aerospace geometry manipulation problems, including the creation of a parametric wingtip generator. We conclude with an airfoil optimization problem conducted in two stages, where the shape parameterization is enhanced between stages. This work is an important step towards our broader goal of offering aerospace designers more relevant, customized control over shape deformation.

Keywords: Aerodynamic shape design, optimization, discrete geometry, deformation, parameterization, constraints, constructive modeling, parametrization

1 Introduction

The ability to drive a baseline aerodynamic shape to one with superior performance hinges on the quality of the shape parameterization. Every set of design parameters inherently restricts the range of reachable shapes, which can be helpful or hurtful. On the one hand it is often useful to delimit the design space, both to simplify the optimization problem and to avoid exploring regions known to be off-limits, often for non-aerodynamic reasons. On the other hand, pre-set deformation mechanisms can restrict the design space in irrelevant ways, needlessly hindering the discovery of valid designs. Deliberate, informed tailoring of the design space to individual problems is crucial for effective design.

In constructive modeling approaches to design (e.g. CAD or custom in-house modelers)[1–6], such customization is difficult. While shape parameters in constructive modelers are usually physically motivated, they are not always particularly relevant for aerodynamic design. It is often desirable to re-parameterize the shape before or during optimization, but this process is generally time-consuming and costly.

An alternate approach is to work directly with discrete geometry (e.g. triangulations or other surface meshes). With discrete geometry, there is no fixed set of shape parameters. The shape is free from the constraints of a particular constructive modeler. The discrete geometry approach thus has the potential to support custom shape parameterizations that are appropriate for the design problem at hand. Aerospace industry research on discrete geometry methods has resulted in several high-quality techniques for shape *deformation* [7–15]. However, these techniques have had limited success in providing intuitive parameters for shape *design*, as they typically require time-consuming manipulation and grouping of large numbers of control points with little relevance to design.

In this work we present a technique for creating custom, design-appropriate shape parameterizations of discrete surfaces. Our approach is to leverage the smooth deformation techniques listed above, but to automate them and remove them from the designer's direct control. Instead, the designer directly selects and manipulates any number of points on the surface and prescribes geometric constraints, while an automated system solves for a smooth deformation of the remainder of the surface that satisfies the constraints. The method, called "constraint-based deformation", is in fact already widely used in the computer graphics (CG) community, and was only recently introduced into the aerospace community [16]. Constraint-based deformation gives the designer direct control over the things that matter (i.e. the surface itself and geometric constraints), while eliminating the need to orchestrate the clouds of control points involved in discrete surface deformation. By supporting the construction of relevant design parameters and by exactly enforcing user-specified constraints, this technique can greatly improve the expression of design intent.

In the following section, we provide an overview of the open geometry manipulation platform that was developed in previous work. Thereafter we present a general constraint-based technique for designing custom shape parameterizations. We evaluate the technique on two aerodynamic shape design examples, and conclude with a brief discussion of preliminary work on returning discrete surface optimization results to constructive surface models.

2 Geometry Platform

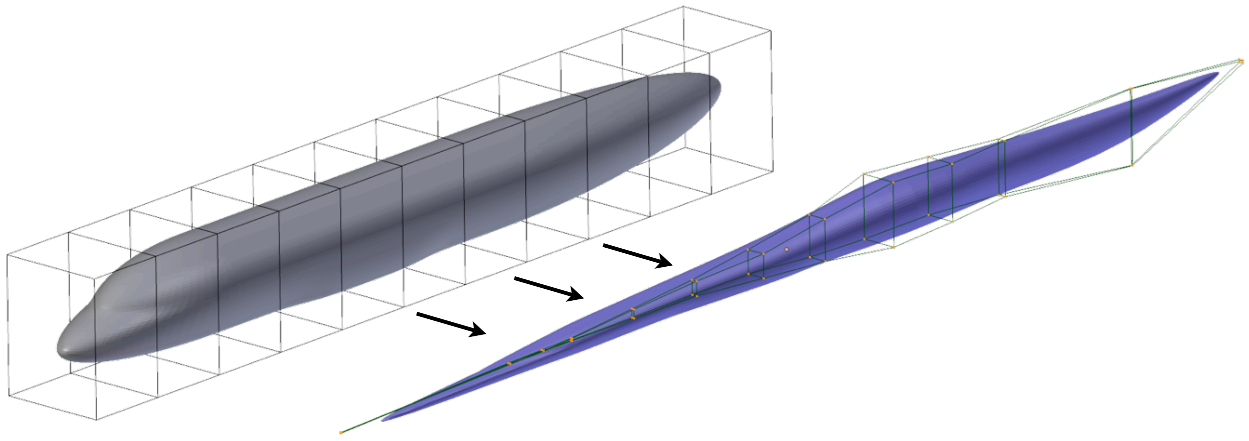
For decades, the CG industry has invested heavily in the development of sophisticated geometry manipulation tools. Although these tools were not originally developed with an engineering focus, they hold great potential for aerospace design. In addition to offering well-established deformation techniques, CG tools are of particular relevance to the aerospace industry because of their high degree of automation and extensibility through back-end scripting interfaces. They are designed for composing and rendering hundreds of thousands of visual scenes without user interaction in production environments. This focus on automation is crucial for aerospace design environments. Optimization frameworks likewise require reliable and flexible geometry engines. Furthermore, new deformation techniques can be rapidly prototyped and incorporated as extensions to these standard, unified geometry manipulation platforms. This enables design groups to tailor a robust discrete geometry platform to meet their needs by writing relatively simple plugins instead of coding a full geometry manipulation tool from scratch.

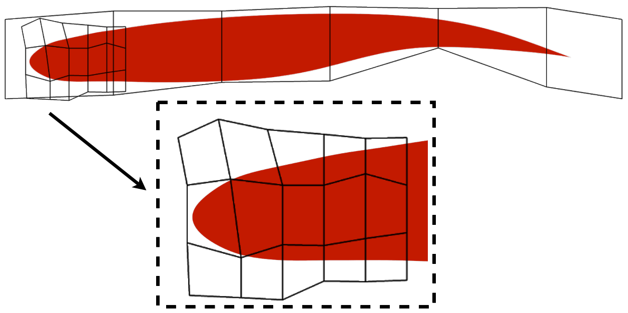
In previous work, we introduced an open discrete geometry tool for aerospace design [17], which we use as the platform for implementing the technique discussed in this paper. The platform is based on a powerful, open-source 3D modeling suite called Blender[a] and serves two roles. First, it allows rapid interactive prototyping of shape parameterizations, using custom panels in Blender's mature GUI. Second, it serves as a fully-automated geometry engine, in a role analogous to a CAD server or scriptable custom modeler. Optimization frameworks can request deformations through a scripting interface. Surface sensitivities are also provided to support gradient-based optimization. The platform can be used alone or pipelined with constructive modelers and other geometry tools to enhance existing design environments.

A Discrete Geometry

Like many modern geometry tools, our platform works directly with discrete surfaces. As compared to constructive modeling (e.g. CAD), discrete geometry allows highly flexible shape modifications. The emphasis here is not on having a completely open design space, which is rarely desirable. The focus is rather on the ability to *customize* the deformation modes and constraints to suit the problem at hand, freeing the designer from the constraints of the original constructive parameterization, which are often aerodynamically irrelevant. Another decided advantage of discrete geometry is the ability to refine the shape parameterization at design time, while preserving the results of optimization on the initial parameterization. Designers can begin optimization in the confidence that the initial shape parameterization is non-binding and can be readily modified as a clearer picture of the design space naturally emerges. For these reasons, pure discrete geometry approaches to aerodynamic design are becoming increasingly popular [4, 8–11, 13–20].

[a]www.blender.org


(a) A lattice reshapes a transport fuselage into a notional supersonic fuselage.


(b) Two lattices shape an airfoil.

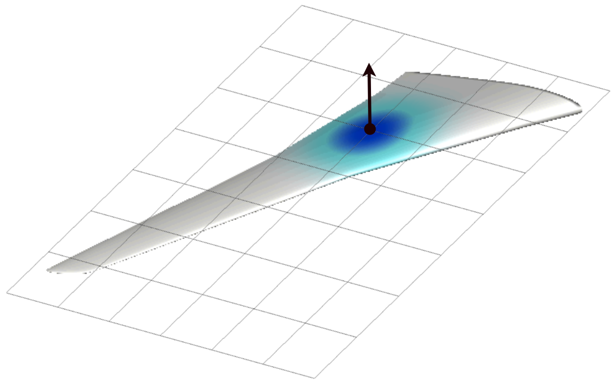

(c) Surface sensitivity to a lattice control point. Dark blue corresponds to the highest sensitivity, light gray indicates zero sensitivity.

Figure 1: Lattice deformation

B Lattice Deformation

In this work we rely heavily on a technique called *lattice deformation*, which is a type of free-form deformation (FFD). Figure 1 illustrates this technique. A regular, rectangular lattice is generated, and surface meshes are placed within this volume. As the lattice is deformed, the embedded surfaces are deformed along with the volume. Each lattice control point smoothly deforms its local neighborhood using a cubic B-spline basis. The total surface deformation is a smooth blending of the effects of all the control points. Even if the lattice control points are deformed non-smoothly, the surface deformation retains a degree of smoothness. If more precise control is needed over a particular region, a second localized lattice can be overlaid, as demonstrated in Figure 1(b), in which one lattice handles large-scale shape changes to an airfoil, while another exercises precise control over the leading edge.

Lattice deformations have a localized effect on the geometry, as depicted in Figure 1(c). Each control point has a region of influence of up to two control points away (the region of support for cubic B-splines). Vertices outside this zone are unperturbed. Lattice deformation has very low computational cost, essentially independent of the resolution of the embedded surface or the scale of its local features. The technique is also agnostic to the discrete geometry format: lattices can simultaneously deform surface triangulations, volume meshes, and even constructive models. Furthermore, deformations remain smooth even on low-quality surface discretizations.[b]

Lattice deformation is just one example of a broad class of "volumetric" deformation techniques. Many conceptually similar methods have been used directly for aerospace design [2, 8, 9, 13, 14, 16, 21]. These approaches are excellent at rapidly and smoothly *deforming* discrete surfaces. However, they are poor at *parameterizing* geometries for aerodynamic design. It is challenging and unintuitive to express engineer-

[b]By "low quality", we mean either coarse tessellations or features such as high aspect ratio faces or widely varying vertex degrees/valences. Such properties can cripple surface-based techniques but do not impact volumetric techniques.

ing design intent by orchestrating clouds of control points floating around the surface geometry, and it is time-consuming to properly enforce meaningful geometric constraints. Additionally, volumetric deformation techniques tend to produce illogical shapes due to their inherent lack of surface-awareness. Overcoming these challenges requires a high degree of user expertise and an intimate understanding of the underlying deformation mechanics.

In the following section, we discuss a higher-level constraint-based technique that leverages the inherent smooth properties of lattice deformation (or equivalently any of the techniques cited above), while giving the user a more direct means of expressing design intent and systematically enforcing constraints.

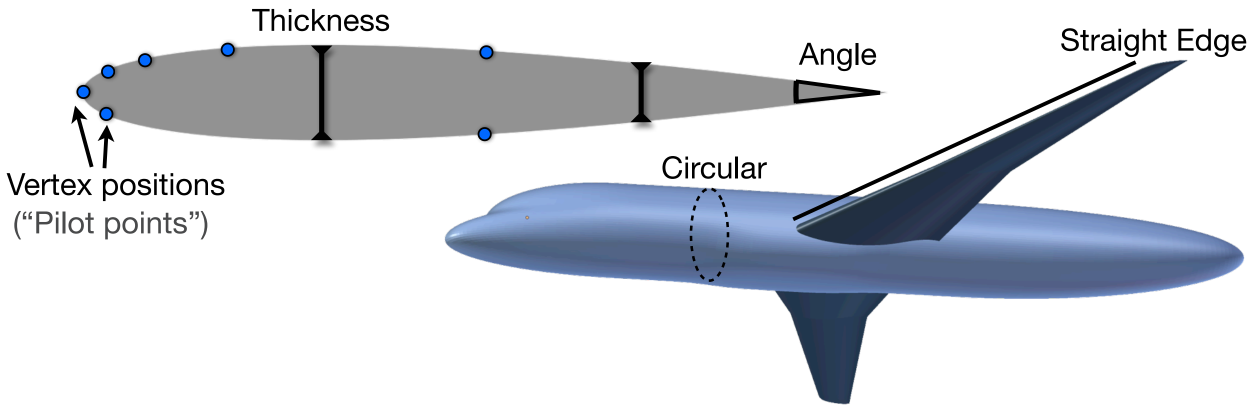


Figure 2: Some basic constraints in aircraft design.

3 Constraint-Based Deformation

In this section we discuss the constraint-based parameterization and deformation technique that is the essential element of our geometry platform. The technique is based on well-established ideas used widely in the CG industry [22–31]. Our particular approach is also inspired by the work by Yamazaki et al., who first used the idea in an aerospace setting [16].

In constraint-based deformation, the geometry is reshaped using a small set of handles, or "pilot points", located directly on the surface, which can serve as design variables for shape optimization. Surface manipulation is intuitive, because the user has direct control of the surface. There is complete freedom to choose the number and location of design variables. The user does not directly control the entire surface, only a sparse collection of pilot points. Manipulation of these points automatically drives a background lattice, which then smoothly deforms the remainder of the surface. The user never directly manipulates the lattice, allowing it to be hidden from view.

We use the same technique to exactly enforce geometric constraints frequently used in aerodynamic shape optimization, such as angles, volumes, surface areas, thicknesses, straightness (collinearity), etc. By construction, constraints are *intrinsically* included in the parameterization. We automatically restrict the design space to include only shapes satisfying the constraints, so that the parameterization inherently cannot produce an invalid shape. If no valid shape is possible, the system returns a best approximation to the constraints, in a least-squares sense.

To be more precise, both the design variables (pilot points) and the constraints are in fact simply "parameters" that drive the deformation. The distinction between design variables and constraints becomes relevant only in the context of optimization. This implies that any geometric "constraint" could alternately be used as a design variable. For example a thickness parameter could be used either as a true constraint (to *fix* the thickness), or as an active design variable with a lower bound (to set a *minimum* thickness). Each shape parameter is expressed as a function of the location of one or more surface vertices. For example, an airfoil thickness parameter can be expressed by relating the position of two vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ located on the upper and lower surfaces of the airfoil:

$$f = \mathbf{v}_i - \mathbf{v}_j \tag{1}$$

Alternately, taking $f = \mathbf{v}_i$ prescribes the absolute location of vertex $\mathbf{v_i}$, establishing it as a pilot point. Other constraints are derived in a similar manner. This focus on the specification of arbitrary constraints broadens the work of Yamazaki et al.[16], where the focus was primarily on pilot points.

Given a set of custom shape parameters (including both design variables and constraints), written as functions $\mathbf{F} = [f_1, f_2 \dots f_n]^T$ with values $\mathbf{X} = [x_1, x_2 \dots x_n]^T$, and noting that the new surface $\mathbf{T}$ is a direct function of the lattice deformation[c] $\mathbf{D}$, we have a matrix system of equations:

$$\mathbf{F}(\mathbf{T}(\mathbf{D})) = \mathbf{X} \tag{2}$$

We then solve for a lattice deformation $\mathbf{D}$ to obtain a new surface satisfying the set of constraints.

In general, the solution is frequently non-unique. There may exist an infinite number of valid deformations that meet the prescribed constraints (e.g. when only a few constraints are applied). In other cases, it may be impossible to simultaneously meet all the specified constraints. How to select the "best" deformation in under- and over-constrained situations is an open question and usually depends on the context. However, some choices are more natural than others. In previous work [17], we presented a preliminary constraint-based technique that used a least-squares solution. We summarize that approach in the following section and then proceed to discuss an energy-based method for picking more intuitive deformations.

A Least-Squares Solution

Intuitively, the least-squares solution seeks to meet all the user-specified constraints while minimizing the lattice deformation. For under-constrained systems, it returns the lattice deformation vector $\Delta\mathbf{D}$ of smallest magnitude that satisfies the constraints. For over-constrained systems, it returns the $\Delta\mathbf{D}$ that results in the smallest overall constraint error in a least-squares measure. To compute this deformation, we solve Equation 2 iteratively using

$$\Delta\mathbf{D} = -\left(\frac{\partial\mathbf{F}}{\partial\mathbf{D}}\right)^{-1}\mathbf{P}\Delta\mathbf{X} \tag{3}$$

where $\Delta\mathbf{X}$ is the difference between the current and target constraint values (i.e. the deviation from a satisfied set of constraints), and $\mathbf{P}$ is an optional preconditioning matrix to improve stability and convergence. Iteration is only necessary when nonlinear constraints are present. If all constraints are linear, then the solution requires only a single step and $\mathbf{P}$ can be dropped. We defer a systematic treatment of nonlinear constraints and preconditioning as topics for future work and focus here on the essential solution technique.

The term $\frac{\partial\mathbf{F}}{\partial\mathbf{D}}$ is the sensitivity of the constraints to the lattice control points, which in general is a rectangular matrix. Equation 3 therefore implies a pseudo-inverse. To compute $\frac{\partial\mathbf{F}}{\partial\mathbf{D}}$, we first expand it into a product of independent terms

$$\frac{\partial\mathbf{F}}{\partial\mathbf{D}} = \frac{\partial\mathbf{F}}{\partial\mathbf{T}}\frac{\partial\mathbf{T}}{\partial\mathbf{D}} \tag{4}$$

The term $\frac{\partial\mathbf{T}}{\partial\mathbf{D}}$ is the sensitivity of the surface to the deformation mapping (i.e. the "surface sensitivities"). The term $\frac{\partial\mathbf{F}}{\partial\mathbf{T}}$ is the sensitivity of the parameter values to the vertices on the discrete surface. It may be either derived analytically for each type of constraint, or finite-differenced if a closed form derivative does not exist. For example, the analytic constraint sensitivities for the thickness parameter in Equation 1 are

$$\frac{\partial f}{\partial\mathbf{T}} = \frac{\partial\mathbf{v}_i}{\partial\mathbf{T}} - \frac{\partial\mathbf{v}_j}{\partial\mathbf{T}} \tag{5}$$

If the constraint system is fully linear, Equation 3 will reach the solution in one step, otherwise iteration is required. The constraint system is linear if both **(1)** the constraints are linear functions of the surface vertices ($\frac{\partial\mathbf{F}}{\partial\mathbf{T}}$ is constant) and **(2)** the surface deformation is a linear function of the deformation parameters ($\frac{\partial\mathbf{T}}{\partial\mathbf{D}}$ is constant). If only pilot points and relative position constraints (Equation 1) are used, the system is fully linear. Other constraints, such as volumes, straightness and angles, may be nonlinear.

[c]Properly, $\mathbf{D}$ is the vector of scalar (x,y,z) deformations of the lattice control points. Note that the use of lattice deformation is not required. Any other deformation technique could be used instead, in which case $\mathbf{D}$ would be the vector of parameters governing that particular deformation model.

While at first glance the matrix inversion in Equation 3 may raise concerns about expense, in fact the size of the matrix is manageable. The two dimensions of $\frac{\partial \mathbf{F}}{\partial \mathbf{D}}$ are the number of custom geometric parameters n_{param} (design variables plus constraints) and the number of lattice control points n_{latt}. Most importantly, the dimensions of $\frac{\partial \mathbf{F}}{\partial \mathbf{D}}$ are independent of the surface resolution, so the technique remains viable even for complex geometries. Deformations typically require less than a second. We discuss deformation speed more thoroughly in section D.

One of the drawbacks to using the least-squares approach is a strong dependence on the lattice resolution, as demonstrated in Figure 3. Using a coarse lattice results in a broader deformation, while using a fine lattice localizes the deformation, sometimes severely. This can be a major disadvantage, as it requires the user to be familiar with the underlying lattice mechanics, preventing automation and requiring user expertise. We would thus like to choose a more intuitive deformation satisfying the same constraints, using a method that is less dependent on the lattice resolution. Figure 3 also shows that deformations resulting from the least-squares solution have little physical basis. This fact motivated Yamazaki et al. [16] to turn to a more sophisticated energy-based technique, which we discuss in the following section.

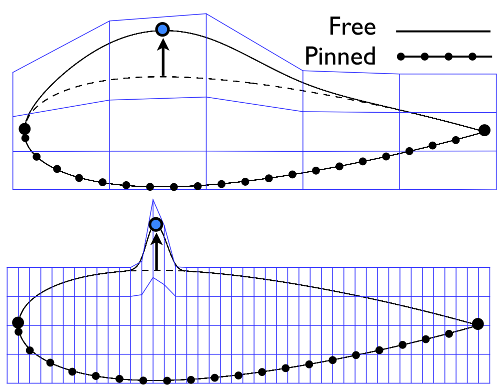


Figure 3: Least-squares lattice solution to constrained airfoil deformation. The top pilot point is moved vertically while the remainder are pinned.

B Energy-based Solution

The energy-based approach to deformation is motivated by the observation that physical deformations tend to follow an energy-minimizing principle. In shape design we are not usually concerned with realistically modeling real-time surface deformation. However, following some of the basic physical principles can tremendously improve the intuitiveness of the deformation.

We still require the geometric constraints in equation 2 to hold, but instead of choosing the least-squares solution, we now choose the solution minimizing some energy E. To keep the deformations rapid, we define this energy on the lattice, which has fixed complexity (i.e. number of control points), rather than on the arbitrarily high-resolution surface.[d] We also require the energy definition to be quadratic in the deformation, which, as before, allows us to obtain a solution in a single step if all constraints are linear.

Before choosing a particular energy definition, we can state the general formulation as follows. Let the discrete quadratic energy E be defined as

$$E = \mathbf{D}^{\mathbf{T}} \left(\sum_{i}^{n_{latt}} \mathbf{K}_i \right) \mathbf{D} = \mathbf{D}^{\mathbf{T}} \mathbf{K} \mathbf{D} \tag{6}$$

where $\mathbf{D}$ is the perturbation of the lattice control points (i.e. $\mathbf{D} = \mathbf{0}$ for the unperturbed lattice) and $\mathbf{K}_i$ are symmetric, positive definite matrices defined at each lattice control point in terms of a stencil over the neighboring control points. The definition of $\mathbf{K}_i$ has a large impact on the types of deformation obtained.

Momentarily we will compare two particular choices of $\mathbf{K}_i$. However, the following standard solution technique is independent of the particular energy definition. To solve for the lattice deformation minimizing a quadratic energy (Equation 6) subject to linear constraints (Equation 2), we introduce the Lagrangian

$$\mathcal{L} = E + \boldsymbol{\lambda} \cdot [\mathbf{F}(\mathbf{D}) - \mathbf{X}] \tag{7}$$

where $\boldsymbol{\lambda}$ is a vector of Lagrange multipliers. The minimal energy solution satisfying the constraints occurs where both $\frac{\partial \mathcal{L}}{\partial \mathbf{D}} = 0$ and $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}} = 0$. Because nonlinear constraints may be present, the formulation becomes slightly more complicated. While the energy is defined in terms of the total perturbation $\mathbf{D}$ of the lattice

[d]The energy can also be defined on the discrete surface itself. This approach is called "surface-based deformation" and is characterized by physically intuitive results but also by costly deformation times on high-resolution surfaces [12, 26, 32].

from the baseline state, the gradually converging nonlinear constraints must be written in terms of a local perturbation from the previous time step, $\Delta\mathbf{D} = \mathbf{D}^{k+1} - \mathbf{D}^{k}$. This results in the following set of equations:

$$\begin{bmatrix} \mathbf{K} & \frac{\partial\mathbf{F}}{\partial\mathbf{D}} \\ \frac{\partial\mathbf{F}}{\partial\mathbf{D}}^{\mathbf{T}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{D} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\mathbf{K}\mathbf{D}^{\mathbf{k}} \\ \mathbf{X} \end{bmatrix} \tag{8}$$

where as before, $\frac{\partial\mathbf{F}}{\partial\mathbf{D}}$ is computed from Equation 4. Roughly speaking, the top row of equations minimizes the lattice deformation energy, while the bottom row enforces the constraints.

1 Lattice Laplacian Energy

We now examine two particular definitions of the discrete lattice energy. The first attempts to minimize the distortion in the lattice as measured by the discrete Laplacian of the lattice control points. This energy is zero when each control point is located at the centroid of its immediate neighbors $\mathcal{N}$, as shown in Figure 4. Using the notation in Equation 6,

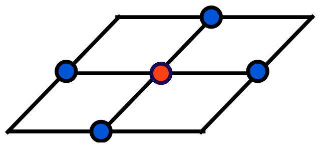


Figure 4: Locally minimized Laplacian energy.

$$\mathbf{K} = \mathbf{L}^{\mathbf{T}}\mathbf{L} \tag{9}$$

where $\mathbf{L}$ is populated using Laplacian stencils. For example, using a 5-point stencil on the interior of a 2-D lattice:

$$\mathbf{L}[i,j] = \begin{cases} -1 & : i = j \\ \frac{1}{4} & : i \in \mathcal{N}(j) \\ 0 & : otherwise \end{cases} \tag{10}$$

In practice we extend this approach to 3D using a 27-point stencil for interior points and reduced stencils for treatment of the boundaries.

Figure 5 shows deformations that minimize the lattice Laplacian energy. Unlike the least-squares solution, this energy-minimizing solution is nearly insensitive to the lattice resolution. This allows the lattice to be generated automatically and be hidden from the designer. Figure 5 also shows, however, large shear and expansion of the airfoil section. The largest deformation does not even occur at the displaced pilot point. In extreme cases this can result in lattices that flip over on themselves. While mathematically this is the expected behavior, stemming from the Laplacian's blindness to both shearing and dilation, it would rarely be the behavior intended by a designer.

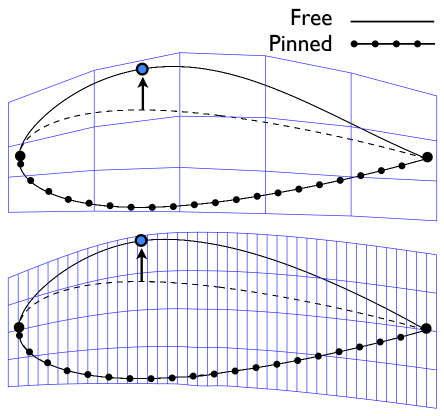


Figure 5: Laplacian-minimizing solutions to the constrained lattice deformation.

2 Elastic Lattice Deformation Analogy

One alternative approach to overcome the drawbacks of the Laplacian energy formulation is the approach developed by Yamazaki et al. [16], where the lattice is viewed as an elastic object that resists deformation. An elastic deformation energy is defined using

$$\mathbf{K} = \mathbf{S}^{\mathbf{T}}\mathbf{B}\mathbf{S} \tag{11}$$

By analogy, the matrix $\mathbf{S}$ is related to the "strain" in the lattice and $\mathbf{B}$ is a constitutive matrix:

$$\mathbf{B} = \begin{bmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & 4\mu & 0 & 0 \\ 0 & 0 & 0 & 0 & 4\mu & 0 \\ 0 & 0 & 0 & 0 & 0 & 4\mu \end{bmatrix} \tag{12}$$

where λ and μ are expressed in terms of Poisson's ratio and an arbitrary modulus of elasticity:

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \mu = \frac{E}{2(1+\nu)} \tag{13}$$

The reader is referred to the original paper for the full derivation [16]. Figure 6 shows deformations minimizing the elastic lattice deformation energy. As expected, the lattice now behaves somewhat like a stretching net, while the surface deformation is still smooth but more localized than under the Laplacian energy definition. Shearing and dilation are penalized, avoiding the problems seen in Figure 5 with the Laplacian energy. While the elastic technique is more sensitive to the lattice resolution than the Laplacian method, it avoids the sometimes pathological contortions of the least-squares method noted earlier.

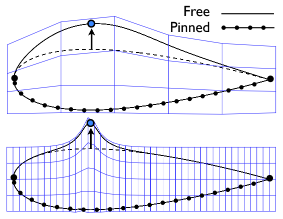


Figure 6: Elastic energy-minimizing solutions to the constrained lattice deformation.

C Ambiguity

Figure 7 shows the two energy-minimizing methods satisfying the same set of constraints to within numerical precision. The Laplacian-based energy excels at broad, smooth deformations with global impact, while the elastic energy results in more localized deformations. Which is more desirable depends on the problem at hand, and which is more "intuitive" depends on the user's perspective. Figure 7 illustrates the ambiguity of intent that is present in any set of constraints. In different situations, one or the other might be the desired deformation. Our aim is not to provide a one-size-fits-all solution, but rather to present the designer with a reasonable and clear set of decisions to make, such as desired locality of deformation, required smoothness, etc. Harnessing this freedom without overwhelming the designer with excessive details is a topic of ongoing research.

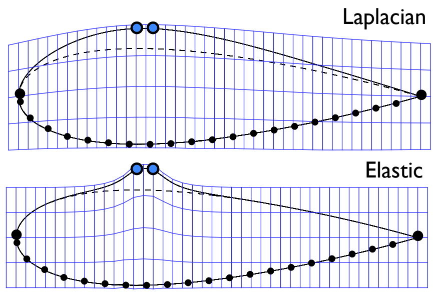


Figure 7: Two lattice deformations satisfying the same set of constraints.

D Cost

The computation is split into two parts. To obtain rapid deformations during optimization, we pre-compute and store as much as possible. This pre-computation phase can require several minutes but only takes place once. Subsequent deformations require seconds or less.

1 Pre-computation

The most expensive step is the pre-computation of the surface sensitivities to the lattice ($\frac{\partial \mathbf{T}}{\partial \mathbf{D}}$ in Equation 4). The energy matrix $\mathbf{K}$ is computed quite rapidly, but since it remains constant throughout the optimization, we pre-compute it as well. In our prototype implementation, the pre-computation phase can take up to several minutes, depending on the resolution of the surface and of the lattice. This should not pose a serious problem, as it is performed offline before optimization. However, several major speed improvements are possible. For simplicity, we currently compute the sensitivity of every surface vertex to the lattice. In reality, only the sensitivities of the small set of vertices that affect the constraints are required. In parallel settings, the computation of surface sensitivities can be split across multiple processes.

2 Deformation

Each deformation proceeds in three steps:

1. Populate matrix system.
2. Solve for lattice deformation $\mathbf{D}$ (Equation 3 or 8).
3. Apply lattice deformation $\mathbf{D}$ to obtain new surface $\mathbf{T}$.

If any constraints are nonlinear, the above sequence iterates to converge on the final deformation.

Populating the matrices (Step 1) and applying the final deformation (Step 3) both involve computations that scale linearly with the number of vertices on the discrete surface, the number of points in the lattice, and the number of constraints. For the energy-based techniques, the square matrix to be solved (Step 2) has dimensions[e] $(3n_{latt} + n_{param})$, which is notably independent of the complexity of the surface discretization. As the matrix solution generally requires the bulk of the time, the total deformation cost is strongly dependent on the lattice deformation but only weakly dependent on the surface resolution.

Table 1 shows typical deformation timings confirming this assessment. The most immediate observation is that deformations all required less than a second, even for fine surfaces and lattices. In the top portion of the table, the number of constraints and the lattice resolution are held fixed. Thus the matrix solution time (Step 2) is constant, and the time variation is due solely to the number of vertices. In the bottom table, the number of vertices is held fixed, and the times are dominated by the matrix solution, which is most sensitive to the lattice resolution. The solution time does scale at least quadratically with the number of lattice control points, however all deformations examined here required less than a second. Furthermore, these results are based on a preliminary implementation that uses a Python-based linear matrix system solver. Naturally, replacing this with a more efficient solver will greatly reduce deformation times across the board, but we re-emphasize that the relative insensitivity to the resolution of the discrete surface means that the technique remains viable even for very high resolution surfaces.

Table 1: Deformation timings (in seconds) on a mid-2009 laptop.[†]

Surface vertices	**Time**
6 K	0.07 *s*
48 K	0.27 *s*
127 K	0.62 *s*
185 K	0.87 *s*

(64 lattice points, 20 constraints)

Lattice points	**Time**
8	0.001 *s*
27	0.04 *s*
64	0.06 *s*
125	0.17 *s*
216	0.53 *s*

(6K vertices, 2 constraints)

[†] Intel 3.06-GHz Core2Duo wall clock times, averaged over 1000 random deformations. All constraints were linear — no iteration was required.

4 Standard Wing Parameters

In addition to the highly flexible deformation modes demonstrated in Figures 3-7, we can also obtain more rigid modes of deformation that are important in engineering settings. In this section we briefly show how a small set of constraints can be used to express standard wing design parameters, while preserving important aerodynamic and structural features such as airfoil sections, straight leading and trailing edges, or wing-fuselage intersection lines.

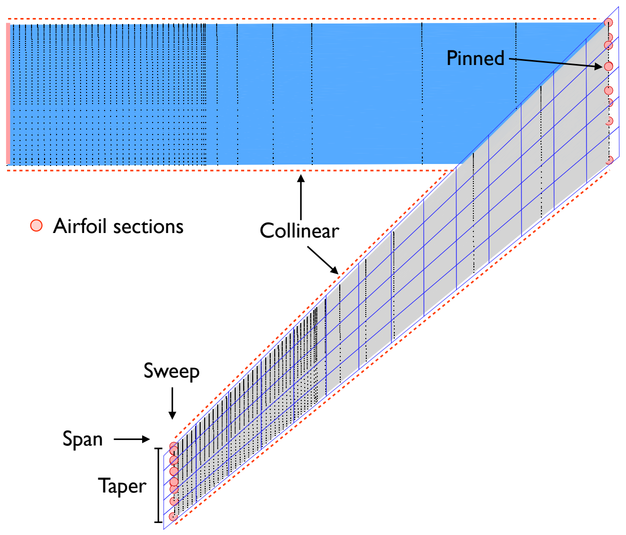
We focus on two basic elements in wing design: airfoil preservation and straightness enforcement (e.g. on leading and trailing edges). Straightness is maintained by constraining a line of surface vertices to remain collinear. Preservation of airfoil sections is accomplished by grouping the sets of vertices defining each section and moving the groups as rigid bodies (i.e. translating, rotating, and uniform scaling, but not warping). While our constraint-based solver sees each airfoil section as a collection of pilot point constraints, optimization can be performed using higher-level parameters derived from these groups.

Using combinations of these two basic design elements, we can recover standard wing design parameters. Figure 8(a) shows how to obtain global wing planform parameters, using two airfoil sections located at the wing root and tip. The root section is pinned, while collinearity constraints preserve the leading and trailing edges. Scaling the tip section sets the taper ratio, while translating it sets span and sweep. Figure 8(b) shows how sectional wing parameters are obtained. Twist is performed by rigidly rotating a section about the quarter-chord line. Dihedral is set by applying a combination of vertical translation and rotation about the stream-wise axis. Finally, thickness and chord are recovered by scaling the section.

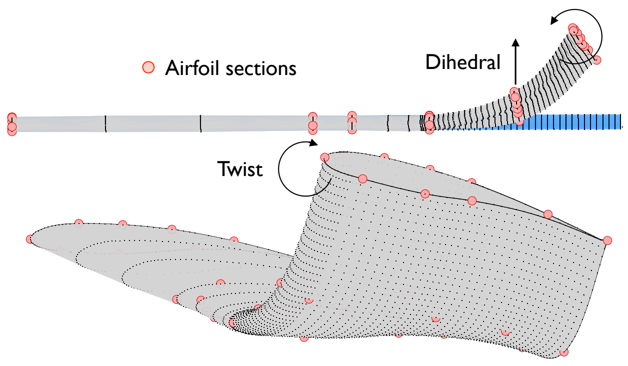
Standard freeform deformation techniques, while excelling at smooth deformations, are typically clumsy at such feature preservation. By adopting a constraint-based approach, we can preserve important features even under radical deformations, as we show in the next section. Following a similar approach, we can also

[e]The factor of 3 indicates that each lattice control point has (x,y,z) degrees of freedom. Similarly, each pilot point is actually 3 constraints (x,y,z).

express standard design parameters and constraints for other components of an aircraft configuration, such as fuselage cross-sections, nacelle geometries, etc.



(a) Global parameters



(b) Sectional parameters

Figure 8: Use of constraint-based deformation to recover standard wing design parameters.

5 Examples and Discussion

We test the new components of our geometry platform on two examples. Before introducing aerodynamics, we evaluate some purely geometric aspects of our approach by using it to parametrically generate a class of wingtips. In the second example, we perform a full aerodynamic shape optimization problem.

A Wingtip Design

In this first example, we examine a challenging deformation problem: deforming a straight wing to add a winglet or generate a C-wing, while preserving airfoil sections. We start with the straight baseline wing shown in blue in Figure 9. The wing is a discrete surface containing about 8500 surface vertices, which we parameterize following the approach outlined in section 4. Eight airfoil sections (loops of surface vertices) are selected and each is linked into a group of pilot points. As the sections are treated as rigid groups, the number of design parameters is about 7 per section: 3 translation, 3 rotation, and 1 scale.

Figure 9 shows an example of a deformation possible using purely rigid transformations of the airfoil sections. In order to clearly depict the designer's view of constraint-based deformation, we have hidden the lattice. The lattice is an underlying deformation mechanism that never has to be directly manipulated, or even viewed, except as a diagnostic measure. In this case we used the Laplacian-minimizing solution, although the elastic energy solution provided very similar results due to the low ambiguity present in our constraints. In general, the energy definition is more important in under-constrained situations. The results of the two definitions tend to converge as more constraints are applied and ambiguities are naturally resolved.

Figure 10 shows a range of other "instances" of wingtips generated using this approach. The parameterization can generate standard wingtips and C-wings, offering wing planform parameters as well as parameters governing the tip shape, such as twist, taper, span, wingtip height, wingtip juncture radius, etc. This parameterization was designed to provide rigid control of large-scale features, but the individual airfoil sections can also be modified, allowing the specification of custom airfoil sections on the wingtip, for example.

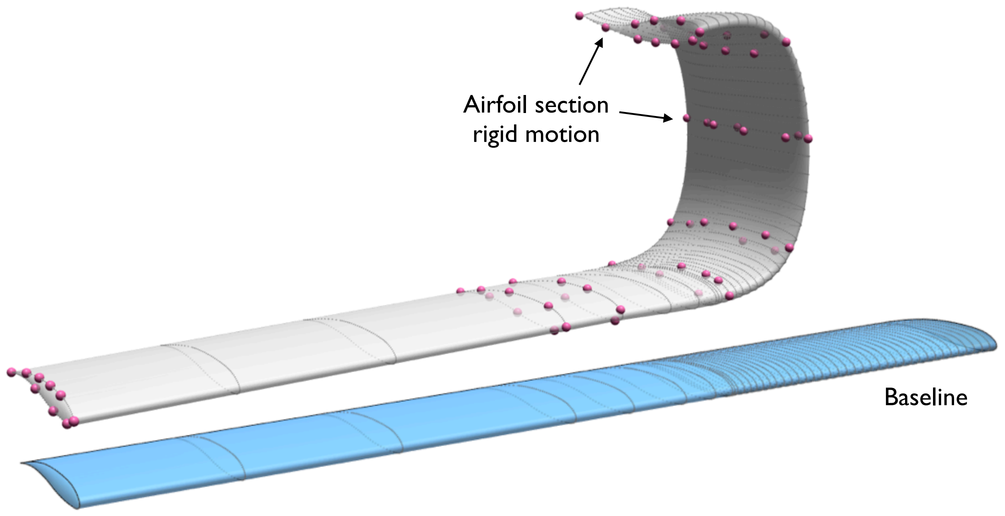


Figure 9: Initial wing (blue) is turned into a C-wing by rigidly translating and rotating groups of pilot points. Airfoil sections are preserved at every station.

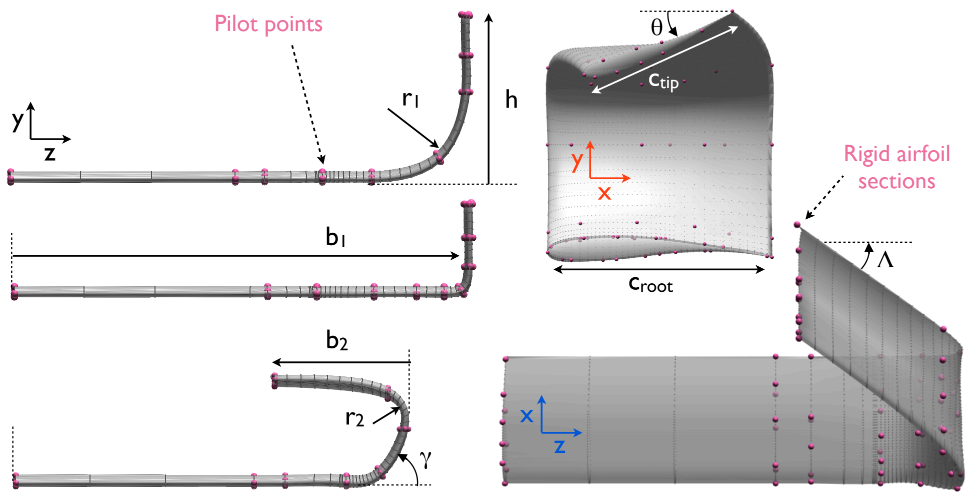


Figure 10: Parametric wingtip definition, showing various wingtips and some key parameters.

Each airfoil section on the baseline discrete surface contained approximately 100 vertices. Instead of constraining all of them, which would have resulted in a much larger number of constraints (and thus a larger constraint system to solve), we constrained representative sets of about 8-10 vertices per section. Exact matching is therefore expected only at these locations, with approximate lofting of the shape between them. Figure 11 shows how well a sparse set of constrained vertices can preserve the airfoil section. The partially constrained section is located at the tip. The unconstrained section is located midway between the tip and the next inboard constrained section. As shown in Figure 11, the explicitly selected pilot points are matched exactly, as required. Between the pilot points, excellent agreement is maintained, especially considering the low

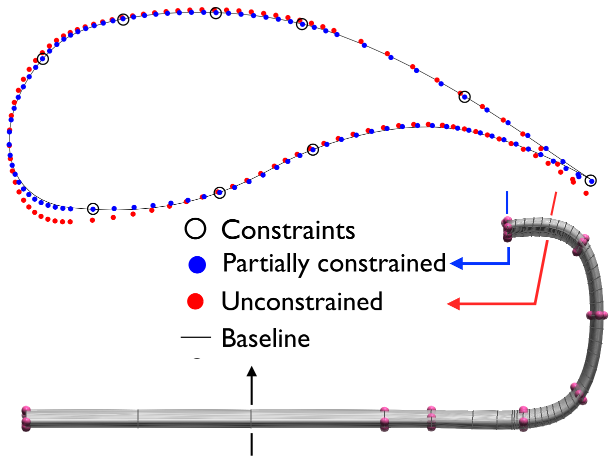


Figure 11: Airfoil preservation.

number of constraints. As expected, more deviation occurs at the unconstrained section, but there is still quite good airfoil preservation, despite the sparsity of the constraints. Selecting more vertices per airfoil or more sections results in more accurate airfoil preservation at a modest increase in computational cost due to the larger matrix system of constraints.

Although the lattice is hidden from the user, its effects are not. For this problem we used a lattice with 240 control points, which was the coarsest lattice supporting the required deformation resolution. As stated in section 3.D, the matrix system dimensions are proportional to the number of lattice control points and the number of constraints. For these cases, the linear system had dimensions around 780×780, resulting in deformation times of 1 - 2 seconds using the preliminary implementation with a Python-based solver. Deformation times should substantially decrease with the incorporation of a more efficient linear algebra package.

Surface sensitivities can be computed to any geometric parameter, for use in gradient-based optimization. Figure 12 shows the surface sensitivity to a point on the inboard vertical surface of the C-wing. We use finite differencing, which provides sufficiently accurate gradients for design. Two deformations are required per design variable at every design iteration. However, we can compute these sensitivities in parallel, processing each design variable on its own CPU.

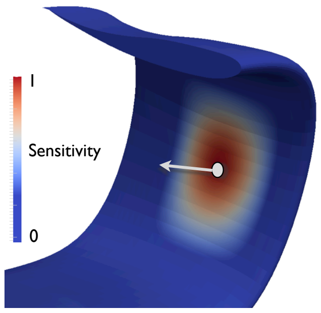


Figure 12: Surface sensitivities to motion of a pilot point.

B Constrained Airfoil Design

In this example we design an airfoil for transonic conditions, subject to wing box constraints and a leading edge bluntness constraint. We start by optimizing on a relatively sparse set of geometric parameters and constraints governing the large-scale features of the airfoil. We then enrich the design space by strategically adding new shape parameters and continue the optimization on the larger design space.

1 Initial Parameterization

The initial geometry is the NACA 0012 airfoil, which we parameterize using the constraint-based technique developed in section 3. Figure 13 shows the parameterization. Thickness parameters are added 30% and 60% chord to approximate typical wing box structural constraints. The thickness at these stations can increase but not decrease. The stations themselves are free to move horizontally and vertically. An additional minimum thickness constraint is added at 8% chord to enforce bluntness at the leading edge. Four additional pilot points on the upper and lower surfaces are used as design variables. The leading and trailing edges are pinned using fixed pilot points. For the initial optimization phase, there are a total of 15 geometric design variables ((x, y) for each pilot point plus the thickness parameters), in addition to the angle of attack at each design point.

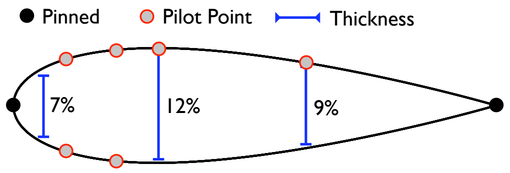


Figure 13: Initial parameterization.

A background 18×7 lattice performs the surface deformation. As we specified only linear parameters, each deformation required only a single iteration. For this problem we used the elastic deformation energy-minimizing technique presented in section 3.B.2 to choose a particular deformation satisfying the constraints. Our constraint-based solver computes a new lattice configuration that generates a new surface satisfying the design variables and constraints, while minimizing the elastic lattice energy. At every design iteration, all geometric constraints are intrinsically and exactly satisfied, because the constraints are built into the parameterization itself.

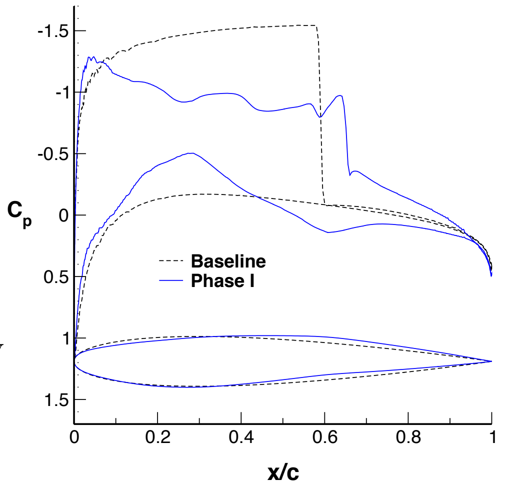
2 Objective and Design Framework

Our objective is to minimize drag at Mach 0.75 at a fixed lift coefficient of 0.63 (based on unit span), corresponding to the lift generated by the initial airfoil at 3° angle of attack. We also specify two secondary design points, also at Mach 0.75 but at lift coefficients of 0.42 and 0.81, to avoid obtaining an alpha-sensitive single-point design. The objective function is a weighted sum of lift and drag functionals at each design point:

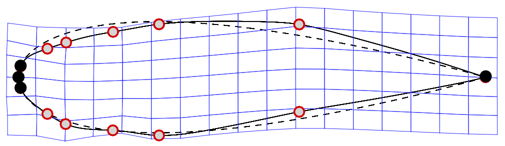
$$\mathcal{J} = \sum_{j=1}^{3} \left[C_{D_j} + \left(1 - \frac{C_{L_j}}{C^*_{L_j}} \right)^2 \right] \tag{14}$$

where $C^*_{L_j}$ are the target lift coefficients. Our constraint-based system only presents the optimizer with designs exactly satisfying the constraints. There is no need to use geometric penalty terms, which only approximately enforce the constraints.

To solve the optimization problem, we used a recently-developed design framework [33] that uses an embedded-boundary inviscid Cartesian mesh method for flow solutions. Objective gradients are computed using an adjoint formulation, and the optimization is handled with SNOPT [34].



(a) Pressure distribution for primary design point. Baseline $\alpha = 3.0°$, final $\alpha = 1.9°$.



(b) Optimized airfoil, pilot points, and deformed lattice minimizing elastic deformation. (2× thickness scale)

Figure 14: Airfoil optimization phase I results.

3 Optimization Phase I

Figure 14 shows the results after about 60 design iterations on the initial parameterization. The minimum thickness constraints were automatically met at every design iteration, because they were intrinsic to the parameterization. In fact the thickness slightly increased at the 60% chord station. As shown in Table 2, wave drag was vastly reduced. Lift was maintained at all design points, except for a slight deficit at the high-lift design point, which could be remedied by a stronger lift penalty term in the objective function. Figure 14(a) shows the greatly weakened transonic shock at the primary design point, where the angle of attack was reduced from 3.0° to 1.9°.

4 Design Space Enrichment

At this point we have substantially improved the design but have reached the limits of our initial parameterization. To find an airfoil yielding yet higher performance, we now modify and refine our parameterization using insight gained during the preliminary optimization. As we are using discrete geometry, the initial choice of shape parameters is non-binding. We can discard less useful parameters and add new ones without losing the results of the initial optimization.

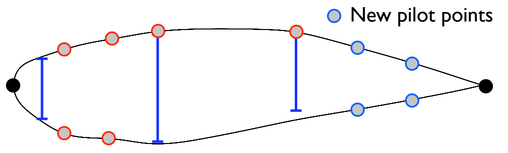


Figure 15: Refined airfoil parameterization with four new pilot points near the trailing edge.

Figure 15 shows the new parameterization. The final discrete surface that resulted from the previous optimization serves as our new starting point. While keeping the initial wing box thickness constraints and leading edge bluntness constraint, we enrich the parameterization in the aft section by adding four new pilot points, for a total of 23 geometric design variables. We also reset the lattice to its initial regular shape again. This is not strictly necessary but makes it easier to see the improvements that are due solely to the second phase of optimization.

5 Optimization Phase II

Figure 16 shows the results after a further 25 design iterations on the refined parameterization. Referring again to Table 2, the drag is substantially reduced at the primary design point, and the lift deficit at the high-lift design point is mostly recovered. As shown in Figure 16(b), the additional degrees of freedom near the trailing edge have allowed the optimizer to introduce some camber that was not possible under the initial parameterization.

Table 2: Airfoil drag minimization results.

	Primary Design Point			Secondary	
	C_L	C_D	L/D (inviscid)	C_L	C_L
Initial	0.63	0.0313	20	0.42	0.81
Phase I	0.63	0.0030	209	0.42	0.78
Phase II	0.63	0.0027	231	0.42	0.80

This problem illustrates the capability of constraint-based deformation to give the designer the ability to create a custom parameterization that is suitable to the problem at hand (in this case transonic airfoil design subject to structural constraints). It also demonstrates the ability to exactly enforce geometric constraints without resorting to awkward penalty functions. Finally, it shows that design time re-parameterization is straightforward using this technique.

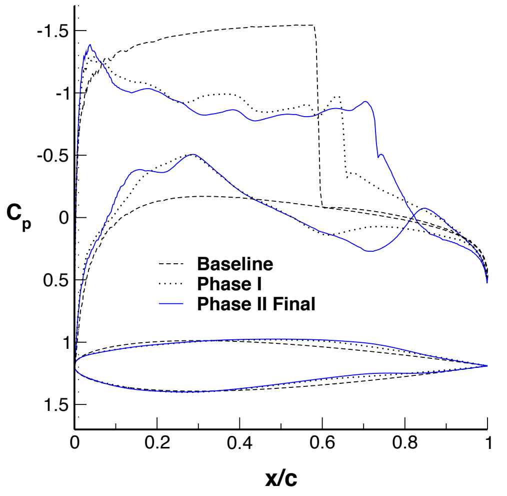
6 Summary and Conclusions

We have presented a constraint-based deformation technique for custom parameterization of discrete geometry. The technique enforces constraints by including them intrinsically in the parameterization, enabling the designer to cleanly and exactly restrict the design space without resorting to penalty functions. The technique automates the lattice manipulation and allows the designer to more directly express of design intent. In addition to free-form deformation modes, standard wing design parameters can be recovered, and local shape features can be preserved. Computational time is modest, with deformation often requiring less than a second. The technique can be used for generating custom parametric deformations for use in automated aerodynamic design frameworks.

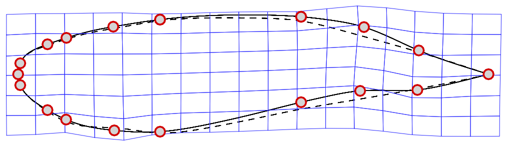
7 Continuing Research

Looking to the future, we plan to explore how to harness the flexibility present in the inherent constraint ambiguity, in a way that provides the user with a reasonable degree of control over the deformation behavior, while not requiring excessive expert knowledge. We also intend to produce a systematic treatment of non-linear constraints (e.g. surface area, collinearity, etc.) using a more efficient matrix solution implementation.

In many design environments, CAD is the standard geometry format, often for manufacturing reasons. In such situations it is important to be able to capture the results of a discrete surface optimization and transfer them back into the originating analytic surface definition, though this transfer can never be exact. One possible approach is to simply apply the final lattice deformation to the control nodes that generate the analytic curve or surface, such as Bézier curves and NURBS surfaces[f]. This functionality is native to Blender, which can



(a) Pressure distribution for primary design point. Baseline $\alpha = 3.0°$, final $\alpha = 1.9°$.



(b) Optimized airfoil, pilot points, and deformed lattice minimizing elastic deformation. (2× thickness scale)

Figure 16: Airfoil optimization phase II results.

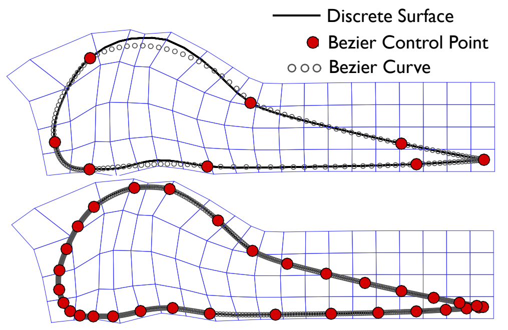


Figure 17: Applying lattice deformation to control nodes of Bézier curves of different resolutions.

[f]Non-Uniform Rational B-Splines.

serve as a platform for prototyping a general procedure for capturing discrete deformations back into CAD or other NURBS-based modelers.

Figure 17 shows a notional demonstration of this technique. In each frame, a lattice has deformed a discrete surface that was generated by sampling a Bézier curve. We apply this same lattice deformation to the control nodes of the Bézier curve, which in turn generate the full analytic curve. In the top frame, the lattice resolution is much higher than the spacing between Bézier control nodes. The coarse analytic definition cannot capture the higher frequency lattice deformations that occur between successive control nodes. In the bottom frame, there is better parity of resolution, so the Bézier curve can closely match the deformed discrete surface. These results illustrate the unavoidable issue of resolution compatibility: If the resolution of the analytic generating surface is lower than the resolution of the discrete deformations, then the finely-optimized results obtained on a discrete surface optimization may be lost when returning to the constructive model. In the future we plan to examine this problem more closely, using this simple method as a baseline for comparing more sophisticated transfer techniques.

Acknowledgments

The authors gratefully acknowledge support from NASA grant NNX09SE60G and the NASA Education Associates Program.

References

[1] Nemec, M. and Aftosmis, M. J., "Aerodynamic Shape Optimization Using a Cartesian Adjoint Method and CAD Geometry," *24th Applied Aerodynamics Conference*, No. 2006-3456, San Francisco, CA, June 2006.

[2] Hicken, J. E. and Zingg, D. W., "Aerodynamic Optimization Algorithm with Integrated Geometry Parameterization and Mesh Movement," *AIAA Journal*, Vol. 48, No. 2, February 2010.

[3] Désidéri, J.-A., Majd, B. A. E., and Janka, A., "Nested and Self-Adaptive Bezier Parameterizations for Shape Optimization," *Journal of Computational Physics*, Vol. 224, No. 1, May 2007, pp. 117–131.

[4] Duvigneau, R., "Adaptive Parameterization using Free-Form Deformation for Aerodynamic Shape Optimization," Tech. Rep. 5949, INRIA, 2006.

[5] Fudge, D. M., Zingg, D. W., and Haimes, R., "A CAD-Free and a CAD-Based Geometry Control System for Aerodynamic Shape Optimization," *43rd AIAA Aerospace Sciences Meeting and Exhibit*, No. 2005-0451, Reno, NV, January 2005.

[6] Kulfan, B. M., "Universal Parametric Geometry Representation Method," *J. Aircraft*, Vol. 45, No. 1, January 2008, pp. 142–158.

[7] Samareh, J. A., "A Survey of Shape Parameterization Techniques," *CEAS/AIAA/ICASE/NASA Langley International Forum on Aeroelasticity and Structural Dynamics*, Williamsburg, VA, June 1999.

[8] Samareh, J. A., "Multidisciplinary Aerodynamic-Structural Shape Optimization using Deformation (MASSOUD)," *8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, No. 2000-4911, Long Beach, CA, September 2000.

[9] Samareh, J. A., "Aerodynamic Shape Optimization Based on Free-Form Deformation," *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, No. 2004-4630, Albany, NY, July 2004.

[10] Dubé, J.-F., Guibault, F., Vallet, M.-G., and Trépanier, J.-Y., "Turbine Blade Reconstruction and Optimization Using Subdivision Surfaces," *44th AIAA Aerospace Sciences Meeting and Exhibit*, No. 2006-1327, Reno, NV, January 2006.

[11] Jakobsson, S. and Amoignon, O., "Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization," *Computers and Fluids*, Vol. 36, No. 6, July 2007, pp. 1119–1136.

[12] Berkenstock, D. C. and Aftosmis, M. J., "Structure-Preserving Parametric Deformation of Legacy Geometry," *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, No. 2008-6026, Victoria, BC, September 2008.

[13] Anderson, W. K., Karman, S. L., and Burdyshaw, C., "Geometry Parameterization Method for Multidisciplinary Applications," *AIAA Journal*, Vol. 47, No. 6, June 2009, pp. 1568–1578.

[14] Morris, A. M., Allen, C. B., and Rendall, T. C. S., "Domain-Element Method for Aerodynamic Shape Optimization Applied to a Modern Transport Wing," *AIAA Journal*, Vol. 47, No. 7, 2009.

[15] Allen, C. B. and Rendall, T. C. S., "CFD-Based Shape Optimization of Hovering Rotors Using Global and Local Parameters," *28th AIAA Applied Aerodynamics Conference*, No. 2010-4236, Chicago, IL, June 2010.

[16] Yamazaki, W., Mouton, S., and Carrier, G., "Geometry Parameterization and Computational Mesh Deformation by Physics-Based Direct Manipulation Approaches," *AIAA Journal*, Vol. 48, No. 8, August 2010, pp. 1817–1832.

[17] Anderson, G. R., Aftosmis, M. J., and Nemec, M., "Parametric Deformation of Discrete Geometry for Aerodynamic Shape Design," Vol. AIAA Paper 2012-0965, Nashville, TN, January 2012.

[18] Désidéri, J.-A., Majd, B. A. E., and Janka, A., "Nested and Self-Adaptive Bezier Parameterizations for Shape Optimization," *Journal of Computational Physics*, Vol. 224, 2007, pp. 117–131.

[19] Persson, P.-O., Aftosmis, M. J., and Haimes, R., "On the Use of Loop Subdivision Surfaces for Surrogate Geometry," *15th Annual Meshing Roundtable*, October 2007.

[20] Majd, B. A. E., Desideri, J.-A., and Duvigneau, R., "Multilevel Strategies for Parametric Shape Optimization in Aerodynamics," *REMN*, 2008.

[21] Sculptor, *<www.gosculptor.com>*.

[22] Hsu, W. M., Hughes, J. F., and Kaufman, H., "Direct manipulation of free-form deformations," *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '92, ACM, New York, NY, USA, 1992, pp. 177–184.

[23] Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., and Seidel, H.-P., "Laplacian surface editing," *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, SGP '04, ACM, New York, NY, USA, 2004, pp. 175–184.

[24] Botsch, M. and Kobbelt, L., "Real-Time Shape Editing using Radial Basis Functions," *Eurographics*, Vol. 24, No. 3, 2005.

[25] von Funck, W., Theisel, H., and Seidel, H.-P., "Vector field based shape deformations," *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, ACM, New York, NY, USA, 2006, pp. 1118–1125.

[26] Botsch, M., Pauly, M., Gross, M., and Kobbelt, L., "PriMo: Coupled Prisms for Intuitive Surface Modeling," *Eurographics Symposium on Geometry Processing*, 2006.

[27] Sumner, R. W., Schmid, J., and Pauly, M., "Embedded deformation for shape manipulation," *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, ACM, New York, NY, USA, 2007.

[28] Sorkine, O. and Alexa, M., "As-rigid-as-possible surface modeling," *Proceedings of the fifth Eurographics symposium on Geometry processing*, SGP '07, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2007, pp. 109–116.

[29] Gain, J. and Bechmann, D., "A Survey of Spatial Deformation from a User-Centered Perspective," *ACM Transactions on Graphics*, Vol. 27, No. 4, October 2008.

[30] Ben-Chen, M., Weber, O., and Gotsman, C., "Variational Harmonic Maps for Space Deformation," *ACM*, Vol. 28, No. 3, August 2009.

[31] Gal, R., Sorkine, O., Mitra, N. J., and Cohen-Or, D., "iWIRES: An Analyze-and-Edit Approach to Shape Manipulation," *ACM Transactions on Graphics (Siggraph)*, Vol. 28, No. 3, 2009, pp. #33, 1–10.

[32] Botsch, M. and Sorkine, O., "On Linear Variational Surface Deformation Methods," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 14, No. 1, January 2008.

[33] Nemec, M. and Aftosmis, M. J., "Parallel Adjoint Framework for Aerodynamic Shape Optimization of Component-Based Geometry," Vol. AIAA Paper 2011-1249, Orlando, FL, January 2011.

[34] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Journal on Optimization*, Vol. 12, 1997, pp. 979–1006.