INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
*Int. J. Numer. Meth. Engng* 2004; **60**:2441–2466 (DOI: 10.1002/nme.1050)

# A freeform shape optimization of complex structures represented by arbitrary polygonal or polyhedral meshes

Jie Shen*, † and David Yoon

*Department of Computer and Information Science, University of Michigan, Dearborn, MI 48128, U.S.A.*

SUMMARY

In this paper we propose a new scheme for freeform shape optimization on arbitrary polygonal or polyhedral meshes. The approach consists of three main steps: (1) surface partitioning of polygonal meshes into different patches; (2) a new freeform perturbation scheme of using the Cox–de Boor basis function over arbitrary polygonal meshes, which supports multi-resolution shape optimization and does not require CAD information; (3) freeform shape optimization of arbitrary polygonal or polyhedral meshes. Numerical experiments indicate the effectiveness of the proposed approach. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: shape optimization; freeform shape; B-spline surface; finite element mesh

## 1. INTRODUCTION

A considerable amount of work in the area of shape optimization of structures has been conducted in the past. From the perspective of geometric representation, these approaches can be generally categorized as either CAD-based [1–7] or mesh-based [8–13]. For a more detailed classification, readers may refer to Reference [14], which contains an excellent review on different shape parameterization techniques. From the perspective of analyses, the applications of boundary element method (BEM) [15–17] and meshless or meshfree method [18–20] signify recent advances in the field of shape optimization. The advantages of using the BEM include the simplification in mesh preparation and higher accuracy within the solution domain, while a main benefit with the meshless method is to provide a treatable solution in the cases of large deformation and non-linear problems.

Even though many finite element and boundary element meshes are generated from their corresponding CAD models, in practice analysts may face situations in which only polygonal or polyhedral meshes are available without knowing any information about their underlying

*Correspondence to: Jie Shen, Department of Computer and Information Science, University of Michigan, Dearborn, MI 48128, U.S.A.
†E-mail: shen@umich.edu

Contract/grant sponsor: University of Michigan
Contract/grant sponsor: University of Michigan-Dearborn

*Received 8 July 2003*
Published online 14 June 2004 *Revised 30 September 2003*
 *Accepted 2 December 2003*

geometry in CAD formats. These situations [21] include (1) meshes are generated by digital sensing systems such as laser scanning equipment used in reverse engineering; (2) meshes are resulted from the simulation results of computational mechanics such as topology optimization [22], etc. If the meshes contain some surface noises, we assume that a certain surface smoothing algorithm has been applied to these meshes [23].

If polygonal or polyhedral mesh representation is used, we will face a problem of maintaining the smoothness of the original finite element mesh. It is a well-known fact that a shape optimization tends to cause zigzag surfaces if each finite element node is used as a design variable [24]. Even though multi-point constraints and dynamic adjustment of design variable bounds can be used to enforce the smoothness, the total number of design variables are directly linked to the number of mesh nodes, which could be very large in some circumstances, leading to a high computational cost and a difficult optimization problem [14, 25]. In addition, although B-spline representation can help in maintaining the smoothness of surfaces during a shape optimization, automatic conversion from complex polygon or polyhedral meshes with sharp edges to B-spline patches is a challenging problem to solve. We will face the following challenges along the line of using the B-spline representation. First of all, complex topology of the shape causes more troubles to the B-spline representation. If we use regular spline patches (i.e. each control vertex has valence 4), the Euler characteristic for a planar graph indicates that such meshes can only be constructed if the overall topology is of the type of infinite plane, infinite cylinder, or the torus. Any other shape like a sphere cannot be built from a quadrilateral control mesh with the valence of all vertices being 4. Furthermore, enforcing higher order continuity at extraordinary vertices is difficult and significantly increases the complexity of the representation. If trimmed NURBS representation is used, how to find and maintain all trimmed boundary curves is not an easy task, especially when some non-closed feature edges exist. Another weakness of the B-spline and trimmed NURBS representation is the need for a mapping between the control mesh and finite element mesh, which causes extra bookkeeping effort and computation.

Consequently, the authors raise a question about if we can design a perturbation scheme for polygonal or polyhedral meshes, which preserves the smoothness of original surfaces without using a B-spline surface representation. If we do not make any assumption on an input finite element mesh, another harder question is whether we can handle an arbitrary polygonal or polyhedral mesh with a uniformly distributed perturbation that is similar to B-spline deformation with a uniform parameterization. Parameterization means a process to establish a 2D parametric domain for a 3D polygonal surface mesh.

In this paper, we present a new scheme for freeform shape optimization on polygonal or polyhedral meshes. In the case of polyhedral meshes, a polygonal surface mesh needs to be extracted as a preprocessing step. The basic idea of our approach is to partition the entire surface of a structure into a finite number of surface patches w.r.t. the geometric characteristics of surface discontinuity, leading to geometric homogeneity and easy shape manipulation of each partitioned patch. Then we design a new freeform perturbation scheme that uses the Cox–de Boor basis function to achieve the smoothness constraints on polygonal surface meshes during the freeform shape optimization process. According to the best of our knowledge, we make the following claims about main contributions of this paper:

(1) A new freeform perturbation scheme that allows a direct freeform shape optimization over arbitrary polygonal or polyhedral meshes with a guaranteed $C^2$ continuity, i.e.

 

the continuity of second derivatives of a surface function, within each surface patch during the freeform shape optimization. It frees the burden of constructing a B-spline or NURBS representation for a complex polygonal or polyhedral mesh, and therefore avoids the corresponding limitations mentioned in the third paragraph of this section. It also removes the bottleneck in shape optimization processes of complex mesh models, i.e. data preparation for the B-spline representation, meshing and their coordination, which normally take hours or even weeks for a complex model. With a complex system such as an airplane, the meshing process alone may take several months to complete [26, 27].

(2) A new approach to the freeform shape optimization, which can handle the existence of surface discontinuity elegantly by using our surface partitioning scheme. It contains an automated scheme for generating a control mesh of knot image, in which the density of knot image is completely independent upon the density of the original polygonal or polyhedral mesh, and can be easily controlled in a multi-resolution manner to achieve a broad range of deformation patterns from local to global perturbations within the scope of each surface patch. Note that the knot image refers to the image of knot in a mapping from a parametric space to an object space.

The rest of this paper is organized as follows. In Section 2, our surface partitioning scheme is shortly introduced, and in Section 3 a new freeform perturbation scheme for surface patches in an arbitrary polygonal or polyhedral mesh is described. Next in Section 4, a shape optimization problem is defined. Numerical experiments are presented in Section 5, followed by concluding remarks in Section 6.

## 2. SURFACE PARTITIONING OF POLYGONAL MESHES

Surface partitioning of unstructured meshes is important to many problems in engineering and science. In the area of parallel computing, researchers have developed various approaches to partition a surface mesh into a number of subregions each of which is assigned to different processors [28–36]. The main objective in this area is to generate equally sized subregions with minimized boundaries. In the area of computational geometry and computer gaming, people focused on convex solid decomposition of polyhedra [37–41] or convex surface decomposition of polygonal meshes [42, 43]. The key idea of the approaches is to break an entire model into a number of convex entities such that collision detection, tolerance verification, motion planning, etc. can be easily conducted. Furthermore, in the area of computer vision, some studies have been carried out on the ridge detection [44, 45] and surface classification [46–48].

We have proposed a new surface partitioning scheme that is specifically designed for shape optimization [49]. The main idea of our approach is to partition polygonal meshes w.r.t. geometric characteristics: geometric discontinuity. Here, the geometric continuity of a surface means the continuity of its tangent vector and curvature for $G^1$ and $G^2$, respectively. Higher-order geometric continuity $G^i$ $(i > 2)$ of a surface means that there exists a reparameterization that can transform the surface to $C^i$ continuity. The reparameterization refers to a parameter transformation by using a strictly monotonic, differential function. Among all these geometric discontinuities such as $G^1, G^2, \ldots, G^n$, only the first two are practically useful, i.e. any $G^i$ $(i > 2)$ discontinuity has an insignificant influence on the shape of the surface if $G^1$ and $G^2$ continuities are guaranteed. Since $G^2$ discontinuity $\supset$ $G^1$ discontinuity in a sense that

 

some $G^2$ discontinuities do not imply $G^1$ discontinuity, the entire task can be divided into two main steps in a sequential manner: (1) surface partitioning by $G^1$ discontinuity and (2) surface partitioning by $G^2$ discontinuity.

The $G^1$ partitioning can be easily implemented by calculating the change in surface normal between adjacent elements, while $G^2$ partitioning is based an accurate computation method for discrete nodal curvatures [49]. $G^2$ partitioning is mainly used for separating fillets from other geometric entities. If an input mesh consists of polyhedra, we need first to extract a surface mesh from this domain mesh, and then to conduct the partitioning introduced in this section on the surface mesh.

## 3. A NEW FREEFORM PERTURBATION SCHEME FOR SURFACE PATCHES

There was a considerable amount of research on the freeform deformation modelling in the area of computer graphics, which inspired the corresponding applications in area of shape optimization. The first group of studies was carried out around Bezier spline, B-spline, NURBS, and their variants [50–52]. With B-spline and NURBS representations, deformation of a surface due to the movement of its control points is guaranteed to be continuous. The applications of these geometric representations in shape optimization have been reported in References [24, 53–58]. The second group of research was conducted on the basis of the concept of freeform deformation (FFD) [59–61], which is essentially an implementation of Bezier solid or B-spline solid as a bounding space. The smooth deformation of this spline bounding space leads to a smooth deformation of the embedded surface or domain meshes by following the interpolation rule of the spline solid. Some researchers applied the FFD in shape optimization [27, 62–64]. Furthermore, physics-based modelling was incorporated into the above two groups of approaches lead to a so-called constraint-based deformation [65–67].

Since we focus on the direct shape optimization on polygonal or polyhedral meshes without any underlying geometric information, explicit B-spline representation is not suited. Even though the FFD technique can be used as a perturbation scheme for shape optimization, it was usually used as a means for nonlinear global deformations such as twisting and shearing [27]. With the FFD, it is difficult, if not impossible, to specify a local perturbation exactly over a certain surface patch with an irregular boundary in terms of both direction and scope. Various approaches in constraint-based deformation focused merely on simulating the deformation as an interactive elastic deformation process with or without a volume preservation constraint, while the objective of our approach is to determine an optimal shape configuration in terms of structure stiffness or other meaningful engineering criteria without attention on a realistic deformation process that is useful for interactive modelling or animation.

The basic idea of our new freeform perturbation scheme is to use a cubic Cox–de Boor basis function [51] as a perturbation pattern for each shape design variable. Theoretically, there are many smooth functions such as Gaussian distribution function, which can be used as a perturbation pattern to ensure the surface smoothness during a shape optimization process. However, one unique benefit of using the Cox–de Boor basis function is its partition of unity (i.e. the sum of all non-zero basis functions is 1 at any point in the parametric domain), which leads to several salient properties such as strong convex hull property. The strong convex hull property means that a surface is contained in the convex hull of its control polygonal mesh, i.e. a mesh that consists of all the de Boor control points [68]. Compared to the Bezier basis

 

function, i.e. Bernstein polynomials, the advantage of using the Cox–de Boor basis function lies in the fact that the Cox–de Boor basis function supports local control of shape, which is a desirable property for shape optimization. In addition, an arbitrary movement of control points of a Bezier spline cannot even guarantee its $G^1$ continuity.

The $i$th Cox–de Boor basis function of degree $p$, $N_{i,p}(u)$, can be defined recursively by [51]

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leqslant u < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{1a}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{1b}$$

where $u_i$'s are called knot values in a parameter space. In order to facilitate a freeform perturbation scheme that uses the above function, $p$ is set to be 3 since this guarantees $C^2$ continuity and the degree is not too high such that the corresponding computation is not costly. The multiplicity of all knots is also confined to be 1 and the boundary type as open-ended, because the geometric discontinuity has been handled by the surface partitioning in Section 1. One implicit assumption of this paper is that we follow the essentials in the parametric model representation of surfaces. By the notations used in [68], a parametric model can be represented by a function $F: P \to \varepsilon$, where $P$ and $\varepsilon$ refer to a parameter space and an object space, respectively. The object space means an affine space in which the input polygonal or polyhedral mesh is described.

Equation (1) is for one-dimensional cases. To extend it to two-dimensional surface cases, we need to calculate another similar basis function, $N_{j,p}(v)$, for the second coordinate in the parameter space. The product of $N_{i,p}(u)N_{j,p}(v)$ is the overall basis function for two-dimensional cases, and it still maintains the partition of unity in the parametric domain.

In order to perform a freeform perturbation on a surface patch of an arbitrary input mesh, we propose two following principles, which can be considered as our contributions to the freeform perturbation, even though we cannot prove that they are the only effective guidelines for this issue.

*Principle 1*: we should try to achieve a uniform distribution of images of knots over an arbitrary mesh, when the knots in a parameter space are mapped to this surface mesh in an object space.

*Principle 2*: assume that the degree of a B-spline surface, $m \times m$, is $3 \times 3$. Given two simple knot sequences $\langle u_{k+1}, u_{k+2}, \ldots, u_{k+2m-1} \rangle$, $\langle v_{l+1}, v_{l+2}, \ldots, v_{l+2m-1} \rangle$ and its corresponding control point $d_{i,j}$, one of the best way to perturb the control point is to move in the surface normal direction at the image point of knot, $(u_{k+m}, v_{l+m})$, on the surface mesh.

Even though the concepts of knots and control points of B-spline surfaces are used in these two principles, as you will see in the later part of this paper, our approach does not really rely upon the B-spline representation. In Principle 1, we consider the image of knot $(u_k, v_l)$ as the range of the function $F(u_k, v_l)$. The basic rational for Principle 1 is that if we can achieve a uniform distribution of images of all knots $(u_k, v_l)$ in the object space, i.e. over the polygonal surface patch, each control point will have an influence on the shape of an approximate equally

 

sized local region. This is a desirable property for shape optimization of surface patches with homogeneity of curvatures. A critical reader may argue about the need for adaptively sized areas of different local control regions over a general surface. Our answer to this criticism is that since our surface partitioning algorithm decomposes a general surface into different patches with homogeneity of curvatures, the adaptively sized areas becomes more or less unnecessary on each partitioned patch in our overall approach to shape optimization. The benefit of using equally sized areas is that we can easily reduce the total number of shape design variables by adjusting the size of each area. This is a crucial property for an efficient shape optimization in engineering practices. Such kind of pseudo 'multi-resolution' effect will be elaborated in example 3 of Section 5.

The Principle 2 is based upon a fact that a surface normal flow is a perturbation that causes the least amount of local distortion on a smooth surface patch. If we consider a special case, a planar surface patch, a perturbation along the direction of surface normal is the only way that does not produce any possible interference between perturbations of two adjacent nodes located at a very short distance away from each other.

On the basis of the above two principles, we need to design a scheme that generates a uniform distribution of images of knots in the object space. This scheme leads to a situation in which each control point $d_{i,j}$, which corresponds to two knot sequences $\langle u_{k+1}, u_{k+2}, \ldots, u_{k+2m-1}\rangle$ and $\langle v_{l+1}, v_{l+2}, \ldots, v_{l+2m-1}\rangle$, will control a certain portion of vertices on the input mesh with an arbitrary distribution of its vertices. In our shape optimization approach, each control point corresponds to a shape design variable. With each control point $d_{i,j}$, a local coordinate system can be established in the parameter space with $(u_{k+m}, v_{l+m})$ as its origin. We normalize it in such a way that the distance between two adjacent knots is always 1. In this local coordinate system, the mathematical formula for the basis function, $N_{i,3}(u)N_{j,3}(v)$, is given by

$$N_{i,3}(u)N_{j,3}(v) = \begin{cases} \frac{1}{36}(3u^3 - 6u^2 + 4)(3v^3 - 6v^2 + 4), & 0 \leqslant u < 1,\ 0 \leqslant v < 1 \\ \frac{1}{36}(u^2 - 4u + 4)(v^2 - 4v + 4), & 1 \leqslant u \leqslant 2,\ 1 \leqslant v \leqslant 2 \\ 0, & u > 2,\ v > 2 \end{cases} \tag{2}$$

where $u$ and $v$ refer to the absolute values of co-ordinates of vertices on the arbitrary input mesh in the local parametric co-ordinate system.

One main advantage of our approach over the conventional B-spline approach [24] is that there is no need to construct B-spline or NURBS surface patches and to co-ordinate them with the polygonal mesh. Only the calculation for locating knot images is required as described in the following subsection.

### *3.1. Calculation for locating all knots in the parameter space*

Since the input mesh can be any arbitrary one, it is not an easy task to generate a set of uniformly distributed images of knots over the input mesh, even though it is still relatively easier than the task of constructing a real B-spline surface. Our overall procedure is as follows:

*Algorithm 1*: *location of all knots*

(1) Project a surface patch onto a planar surface
(2) Overlay the projected surface patch with a regular parametric grid mesh

 

(3) Smooth the overlaid grid mesh on the basis of length distortion of the projected surface patch
(4) Group vertices of the smoothed parametric grid mesh to form a set of knots

The purpose of step 1 in Algorithm 1 is to establish a mapping from the 3D object space to a 2D parameter space. Many approaches have been proposed for the projection of polygonal meshes from 3D to 2D. Eck *et al.* [69] used harmonic maps to generate a 2D projection of a 3D polygonal mesh. The method produces a reasonably good quality of mapping in terms of minimizing a geometric distortion metric. However, one limitation is that the boundary of the 2D mesh needs to be convex. Floater [70] provided a provable solution of generating a 2D projection by using convex combination. It guaranteed the validity of the approach when the boundary of the 2D mesh is predefined and convex. Marcum and Gaiter [71] used a novel finite element approach to conduct a global mapping and calculate a 2D projected mesh without a proof of its validity. In the community of computer graphics, related studies in texture mapping [72, 73] suffer similar limitations as mentioned above.

In this paper we use the angle-based flattening (ABF) method proposed by Sheffer and Sturler [74]. The advantage of using this method include (1) it can handle arbitrary manifold surfaces with large curvature gradients and (2) the boundary of the 2D mesh is not required to be predefined or convex. Even though the ABF was originally designed only for manifold surfaces, it can be still used for non-manifold input meshes if it is used in conjunction with our surface partitioning scheme that breaks a non-manifold surface into a number of manifold surface patches.

The basic concept of the ABF method is to convert a projection problem into a constrained optimization that is in turn transformed to an unconstrained optimization by using Lagrangian multipliers as follows:

$$\text{Minimize} \quad F(\alpha) + \sum_{i=1}^{P} \lambda_i^{(2)} g_i^{(2)}(\alpha) + \sum_{k=1}^{M_{\text{int}}} \lambda_k^{(3)} g_k^{(3)}(\alpha) + \sum_{k=1}^{M_{\text{int}}} \lambda_k^{(4)} g_k^{(4)}(\alpha) \tag{3}$$

where

$$F(\alpha) = \sum_{i=1}^{P} \sum_{j=1}^{3} \left( \frac{\alpha_i^j}{\phi_i^j} - 1 \right)^2$$

$$g_i^{(2)}(\alpha) = \sum_{j=1}^{3} \alpha_i^j - \pi = 0$$

$$g_k^{(3)}(\alpha) = \sum_{i} \alpha_i^{j(k)} - 2\pi = 0$$

$$g_k^{(4)}(\alpha) = \frac{\prod_i \sin\left(\alpha_i^{j(k)+1}\right)}{\prod_i \sin\left(\alpha_i^{j(k)-1}\right)} - 1 = 0$$

in which, $\alpha_i^j$, $i = 1, \ldots, P$, $j = 1, 2, 3$, are angles of the 2D projected mesh. $\phi_i^j$ is the optimal angle for $\alpha_i^j$ in the 2D projected mesh, and is calculated by scaling the original mesh

 

Table I. Algorithm 1.1 for parameterization of a surface patch.

(1) Create a bounding box for the 2D projected mesh
(2) Create a uniform overlaid grid mesh inside the bounding box
(3) Laplacian smoothing of the overlaid grid mesh

(3.1) For each edge, $e = (n_a, n_b)$, of the grid mesh, compute its desired length:

$$l(e) = \frac{S(n_a) + S(n_b)}{2}$$

(3.2) Adjust the edge lengths by using Laplacian smoothing on the basis of the sizing function

(3.2.1) For each interior node, $N$,

$$\hat{N} = \frac{\sum_{e=(N,N')}(1/l(e))N'}{\sum_{e=(N,N')}(1/l(e))}$$

$$N \leftarrow \hat{N}$$

(3.2.2) repeat (3.2.1) while $\max(\|\hat{N} - N\|_2) >$ tolerance

(3.3) Go back to (3.1) until the length, $l(e)$, or the vertex locations no longer change

angles in 3D proportionally such that the sum of $\phi_i^j$ at each vertex in the 2D mesh equals $2\pi$. $g_i^{(2)}$, $g_k^{(3)}$ and $g_k^{(4)}$ are constraints on angles in each triangle, angles at each vertex, and length consistency of edges contingent to each vertex, respectively. $\lambda_i^{(2)}$, $\lambda_k^{(3)}$ and $\lambda_k^{(4)}$ are the corresponding Lagrangian multipliers. Equation (3) can be solved by the Newtonian method in conjunction with a trust region search strategy. For further details, refer to Reference [74].

The basic objective of Steps 2 and 3 in Algorithm 1 is to construct a 2D parameterization that minimizes both angular and linear distortions in a projection from a 3D input mesh to a 2D mesh. This in turn guarantees a uniform distribution of images of knots in the object space. One algorithm proposed by Sheffer and Sturler [75] is used in this paper. The main idea of this algorithm is to overlay a uniform grid mesh on top of the projected 2D mesh as an initial parameterization, and then to apply a Laplacian smoothing on this uniform grid mesh to reduce the linear distortion caused by the projection to the 2D mesh in using the ABF algorithm. Consequently, in Step 4 the vertices on the smoothed grid mesh can be grouped to form a set of knots that have a uniform distribution of their images over the surface patch of the input mesh. Note that this set of knots represent a parameterization with which Equation (2) is ready to be applied.

A modified version of the Laplacian smoothing algorithm in Reference [75] is given in Table I.

In Step 1 of Algorithm 1.1, the size of the bounding box should be larger than the 2D projected mesh such that at least two extra layers of knots, i.e. vertices of the overlaid grid mesh in Step 2, are created beyond the boundary of the 2D projected mesh. In this paper, we let the size of the bounding box be 2.8 and 4.0 times as large as the 2D projected mesh when $n \geqslant 4$ and $n \leqslant 3$, respectively. Here, $n$ is related to the mesh density defined in Step 2.

 

Table II. Algorithm 1.2 for generation of a set of knots.

(1) Initialize the knot set to a null set
(2) Loop over each vertex of the smoothed grid mesh obtained from Algorithm 1.1
  (2.1) Get the co-ordinate of the current grid vertex
  (2.2) Test if the current grid vertex in one element of the 2D projected surface patch obtained from Step 1 in Algorithm 1
  (2.3) if yes, add the current grid vertex to the knot set
(3) Expand the boundary of the knot set one layer outward by using the node–node connectivity information of the smoothed grid mesh. All the new grid vertices encountered in this expansion are added into the knot set.
(4) Repeat Step 3 once to add another layer of knots to the knot set

The default mesh density in the direction $y$ is $2^n = 2^5 = 32$ in Step 2, while the density in direction $x$ is the multiplication of $2^n$ and a scaling factor that is dependent upon the ratio of dimension $x$ to dimension $y$ in the 2D projected mesh. Users have a control over the value of $n$ to achieve a kind of multiresolution shape optimization. For the details, refer to example 3 in Section 5.

In Step 3.1, $n_a$ and $n_b$ refer to two end vertices of the edge. $S(\ )$ is a sizing function. For each edge, $e = (n_a, n_b)$, $S(e) = \|e_{2D}\|_2 / \|e_{3D}\|_2$, in which $e_{2D}$ and $e_{3D}$ refer to the edge $e$ in the projected 2D mesh and the 3D input mesh, respectively. For each vertex, $n_a$, $S(n_a)$ is the average of $S(e)$ over all edges that are contingent to $n_a$.

Figure 1 illustrates the process of calculating the location of knots for freeform perturbation. The three-sphere data model in this figure was generously provided by Dr Sheffer at Technion. Note that the input mesh contains three surface patches that are projected together onto a plane to show the capability of the ABF algorithm [75]. The tiny dots over the input mesh in Figure 1(e) represents the images of knots, which have a one-to-one relationship with the knots, i.e. vertices of the smoothed grid mesh in the parameter space.

In Step 4 of Algorithm 1, the procedures to generate a set of knots in the parameter space are given in Table II.

Note that a bounding box technique is used to speed up the computation in Step 2.2, which is similar to the bounding volume technique used commonly in the area of collision detection. The reason for us to conduct Steps 3 and 4 is due to the fact that we use the cubic B-spline basis function as a perturbation function such that the radius of an influence zone of each control point is two knots away from the current reference knot in the parameter space, as indicated in Equation (2). Since Step 2 catches all the knots that are located within the boundary of the 2D projected surface patch obtained from Step 1 in Algorithm 1, a two-layer outward expansion will ensure that the knot set contains all the knots that cover a just-enough knot domain which supports all the control points that have an active influence on the shape of the current surface patch. For details, see the proof of Proposition 3 in Section 3.2.

### *3.2. Freeform perturbation scheme*

For each surface patch obtained after applying the surface partitioning in Section 2, if it is chosen as a patch for freeform shape optimization, procedures of our freeform perturbation scheme are applied as given in Table III.

 

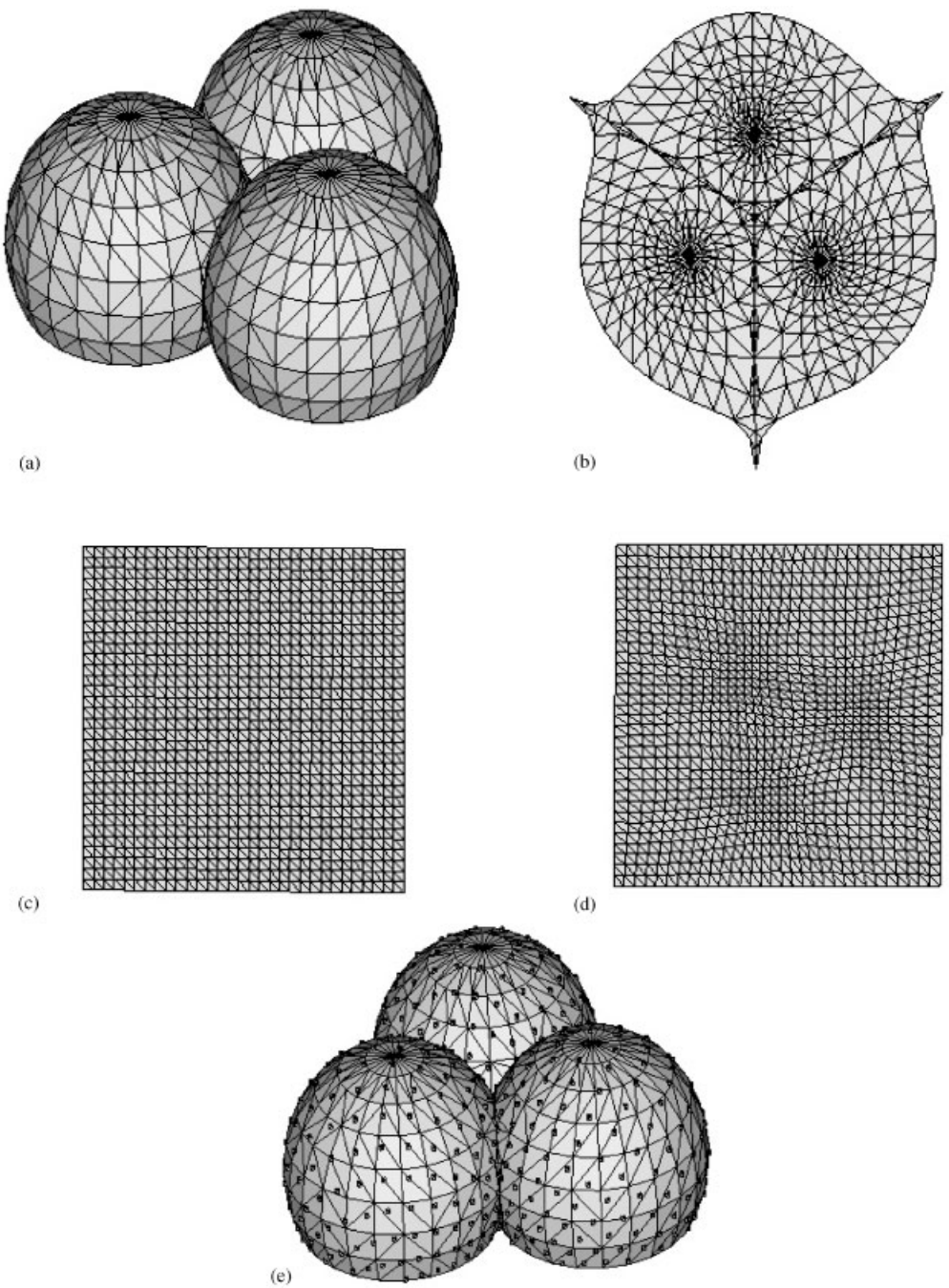


Figure 1. Generation of images of knots over a complex surface of three balls in the object space: (a) input mesh; (b) 2D projected mesh; (c) uniform grid mesh; (d) smoothed grid mesh; and (e) input mesh with the images of a set of knots represented by small dots.

 

Table III. Algorithm 2 for freeform perturbation.

(1) sort all elements of the 2D projected surface patch on the basis of the location of their bounding boxes
(2) calculate the nodal surface normal of all vertices on the current surface patch in the object space
(3) loop over each knot in the knot set, which is determined by Algorithm 1.2, over the current surface patch
  (3.1) find out which element of the current surface patch contains this knot in the parameter space
  (3.2) calculate the nodal surface normal at the image of the current knot in the object space
  (3.3) find out all nearby nodes on the patch w.r.t. the image of the current knot
  (3.4) calculate the perturbation of all nearby nodes on the patch w.r.t. the image of the current knot in the object space
  (3.5) all non-zero perturbations in a local neighbourhood of the current knot form a perturbation vector that corresponds to a shape design variable to be used in a shape optimization.

A typical reference for Step 1 in Table III is the position of the left lower corner in one coordinate direction. In Step 2, the nodal surface normal at a vertex on the current surface patch is an average of surface normal of all surrounding elements on the patch. Since all the elements of the current surface patch are already sorted in the parameter space in Step 1, it is relatively easy to complete Step 3.1. If the knot is outside the boundary of the current surface patch, then find out the nearest element on the current surface patch. In Step 3.2, the normal is calculated by a linear interpolation (for triangular elements) or bi-linear interpolation (for quadrilateral elements) of the nodal surface normal at all vertices of the corresponding element determined in Step 3.1.

We need to find a local neighbourhood for the current knot in Step 3.3, as shown in Figure 2. Since degree 3 and multiplicity 1 are used in this paper, the influence zone of a control point is a region centered at the current knot, and is four knots wide in both $u$ and $v$ directions in the parameter space. Any vertex of the 2D projected surface patch, which is located inside this influence zone, is considered to be in the neighbourhood of the current knot. Breath-first search is used to find out these nodes starting from the element determined in Step 3.1.

In Step 3.4, a bi-linear interpolation w.r.t. elements in the smoothed grid mesh is used to determine the $(u, v)$ values of each nearby vertex of the projected patch. These two values represent relative parametric co-ordinates with respect to the current knot in directions $u$ and $v$, respectively. The perturbation of each nearby vertex on the patch is calculated by multiplying the nodal surface normal of the image of the current knot in the object space with the B-spline basis function in Equation (2) in the parameter space.

Note that in Step 3.4 we use the nodal surface normal at the image of the current knot to determine the perturbation for all nearby vertices on the surface patch. This treatment exactly follows the spirit of B-spline deformation, as indicated in Equation (4). An alternative treatment is that nodal surface normal at each nearby vertex is multiplied by the B-spline basis function. In our opinion, the second treatment may work well with an interactive shape modeling system, but not in a shape optimization in which we do not want any mixture of perturbation directions

 

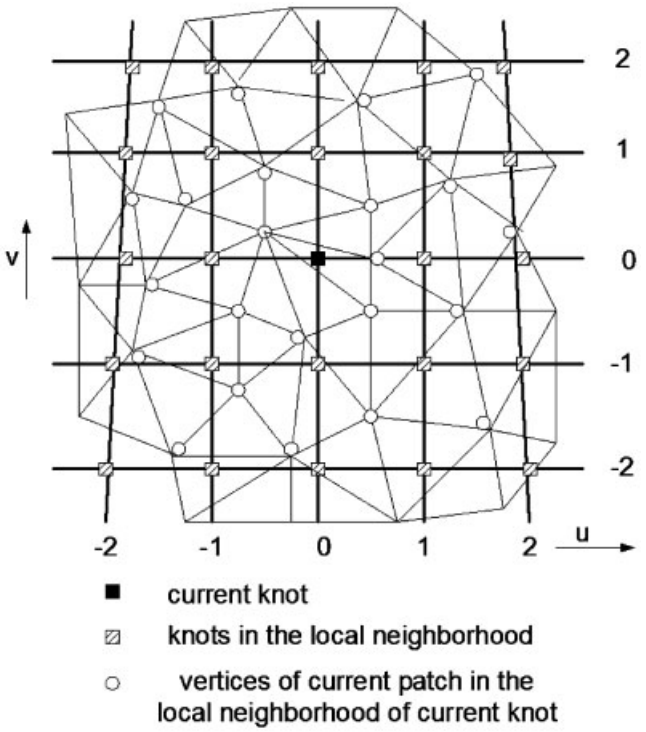


Figure 2. Local neighbourhood of the current knot in the parameter space.

within a single shape design variable so as to help an optimizer in searching an optimal solution in a design space.

Our freeform perturbation scheme is supported by the following propositions:

*Proposition 1*

With an open-ended B-spline surface that contains only simple knots, for any perturbation of a vertex of its control net (control point), we can find a corresponding perturbation of surface vertices of this B-spline surface. In addition, this corresponding perturbation follows the pattern defined by the basis function of the control point.

*Proof*

The details of this proof can be found in Reference [21]. The key point of this proof is that we can prove

$$D(u, v) = C(u, v) + N_{i,p} N_{j,q} \mathbf{w} \tag{4}$$

where $D(u, v)$ and $C(u, v)$ represent the perturbed and original surfaces, respectively. $u$ and $v$ are two parameters associated with this surface. $N_{i,p}$ and $N_{j,q}$ are B-spline basis functions in directions $u$ and $v$, respectively. Both $p$ and $q$ are set to 3 in this paper. $\mathbf{w}$ refers to a perturbation vector applied at control point $\mathbf{P}_{i,j}$. □

 

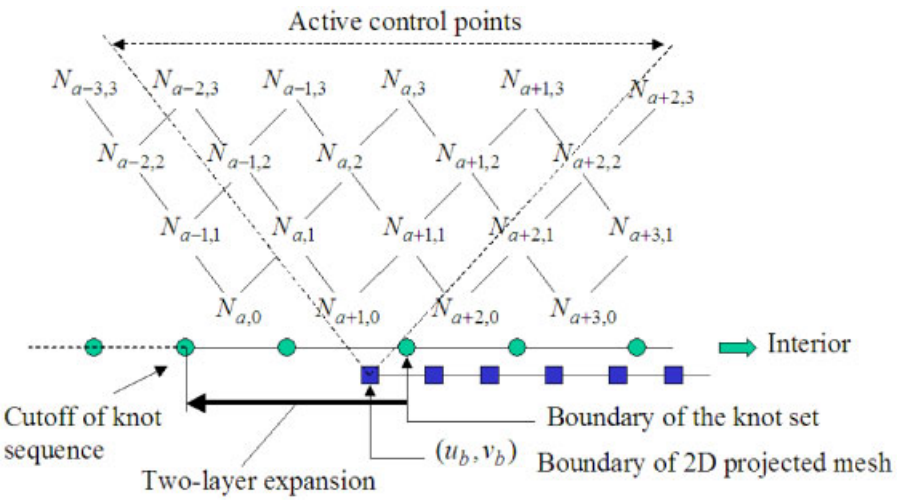


Figure 3. Two-layer expansion of knot sequence at the boundary of 2D projected mesh. $N_{i,j}$ represents a Cox–de Boor basis function.

*Proposition 2*
If vertices of a surface patch are perturbed on a pattern that follows the basis function of the corresponding control vertex, the smoothness of the original surface patch is well maintained.

*Proof*
First assume that the original surface patch is $C^2$ continuous. According to Equation (4), the surface patch obtained after applying our freeform perturbation scheme, $D(u, v)$, is still $C^2$ continuous, because both $p$ and $q$ are set to 3, and multiplicity of every knot is 1 in this paper. If the original surface patch is $C^m$ continuous ($m > 2$), $D(u, v)$ is $C^2$ continuous, which is normally sufficient for the smoothness requirement of most engineering applications. We therefore say the smoothness is well preserved in this case. In the cases of $m < 2$, the smoothness of the original surface patch is exactly maintained. Overall, our freeform perturbation scheme well maintains the smoothness of the original surface patch. □

*Proposition 3*
If vertices of a surface patch are perturbed on a pattern that follows the basis function of the corresponding control vertex, the partition of unity is valid with respect to the sum of all perturbations at the interior of the surface patch.

*Proof*
Note that in this paper we focus on cubic uniform B-spline surfaces with simple knots for each surface patch. Even though the dimension of the smoothed grid mesh (Algorithm 1.1) is limited, the knot sequence, which corresponds to this grid mesh, can be imagined as infinite and as being cut off at the boundary defined by the knot set determined by Algorithm 1.2, as shown in Figure 3. Under these restrictions, for an arbitrary quadrilateral region on the

 

parametric grid mesh, $\{(u, v)|u_s \leqslant u < u_{s+1}, v_t \leqslant v < v_{t+1}\}$, we can write

$$\sum_{k=i-3}^{i}\sum_{l=j-3}^{j} N_{k,3}(u)N_{l,3}(v) = \sum_{k=i-3}^{i}\sum_{l=j-3}^{j}\left[\frac{u-u_k}{u_{k+3}-u_k}N_{k,2}(u) + \frac{u_{k+3+1}-u}{u_{k+3+1}-u_{k+1}}N_{k+1,2}(u)\right] \times \left[\frac{v-v_l}{v_{l+3}-v_l}N_{l,2}(v) + \frac{v_{l+3+1}-v}{v_{l+3+1}-v_{l+1}}N_{l+1,2}(v)\right] \tag{5}$$

Since $N_{i-3,2}(u) = N_{i+1,2}(u) = N_{j-3,2}(v) = N_{j+1,2}(v) = 0$, Equation (5) can be rewritten as

$$\begin{aligned}&\sum_{k=i-3}^{i}\sum_{l=j-3}^{j} N_{k,3}(u)N_{l,3}(v)\\ &= \sum_{k=i-3+1}^{i}\sum_{l=j-3+1}^{j}\left[\frac{u-u_k}{u_{k+3}-u_k} + \frac{u_{k+3}-u}{u_{k+3}-u_k}\right]N_{k,2}(u)\left[\frac{v-v_l}{v_{l+3}-v_l} + \frac{v_{l+3}-v}{v_{l+3}-v_l}\right]N_{l,2}(v)\\ &= \sum_{k=i-2}^{i}\sum_{l=j-2}^{j} N_{k,2}(u)N_{l,2}(v)\end{aligned} \tag{6}$$

By repeating the above procedures, we can finally get

$$\sum_{k=i-3}^{i}\sum_{l=j-3}^{j} N_{k,3}(u)N_{l,3}(v) = \sum_{k=i-1}^{i}\sum_{l=j-1}^{j} N_{k,1}(u)N_{l,1}(v) = \sum_{k=i}^{i}\sum_{l=j}^{j} N_{k,0}(u)N_{l,0}(v) = 1$$

Similarly, even at $(u_b, v_b)$ that corresponds to the boundary of the 2D projected mesh in Figure 3, the partition of unity should hold by summing the contributions from all active control points. □

The partition of unity leads to several salient properties such as convex hull. With these three propositions, we can perturb the finite element surface nodes in a pattern of the Cox–de Boor basis function without a need for B-spline representation. This is the main advantage of our approach.

Since we introduce a two-layer outward expansion of knot sequence or corresponding control points, the partition of unity still holds at the boundary nodes of surface patches. Therefore, there is no need for a special treatment and the basis function in Equation (2) is still used. We rely on the shape optimization process to find a suitable perturbation magnitude for the perturbation at the boundary.

In order to avoid a possible mesh distortion due to the perturbation of surface nodes on each patch, the domain nodes close to these surface nodes can be proportionally perturbed. The image of each knot in the object space has a spherical influence zone with an adjustable radius that is an input from users, and inside the zone the perturbation becomes effective with a magnitude that linearly decreases with the distance between the domain node and the image of the knot. This is a simple scheme that alleviates but cannot eliminate the mesh distortion in the domain. To guarantee no distortion of the domain mesh, more sophisticated schemes are needed. The studies by Farhat *et al.* shed a light in this direction [76, 77]. From the perspective

 

of shape optimization, a major disadvantage of using the FEM is a possible severe distortion of the domain mesh, compared to the BEM or meshless method. A complete solution to the mesh distortion problem in shape optimizations and general finite element analyses is a future research topic.

If we need to conduct freeform shape optimization on two adjacent surface patches that share a sharp edge boundary, no special geometric constraint is imposed in our approach. The final shape at the boundary is totally dependent upon the result of an optimizer to be used. Whether or not any geometric constraint is needed is a topic of future work.

## 4. FREEFORM SHAPE OPTIMIZATION

The freeform shape optimization in this paper is expressed by

$$\text{minimize} \quad f(\mathbf{y}) = \frac{1}{2}\,\mathbf{u}(\mathbf{y})^{\mathrm{T}}\mathbf{K}(\mathbf{y})\mathbf{u}(\mathbf{y}) = \frac{1}{2}\int_{\Omega}\boldsymbol{\varepsilon}(\mathbf{y})^{\mathrm{T}}\boldsymbol{\sigma}(\mathbf{y})\,\mathrm{d}\Omega = \frac{1}{2}\,\mathbf{u}(\mathbf{y})^{\mathrm{T}}\mathbf{F} \tag{7a}$$

$$\text{subject to} \quad g(\mathbf{y}) - g^{\mathrm{u}} \leqslant 0 \tag{7b}$$

$$\mathbf{y}^{\mathrm{L}} \leqslant \mathbf{y} \leqslant \mathbf{y}^{\mathrm{u}} \tag{7c}$$

where the objective function $f(\mathbf{y})$ is a compliance, i.e. strain energy over a design domain $\Omega$. If there are multiple load cases, a weighted compliance is used on the basis of linear combination of compliance in each load case with a specific weighting factor. $\mathbf{u}$ and $\mathbf{K}$ are displacement vector and stiffness matrix, respectively. $\boldsymbol{\varepsilon}$ and $\boldsymbol{\sigma}$ are strain and stress tensors, respectively. $\Omega$ refers to the domain occupied by the structure. $\mathbf{g}(\mathbf{y})$ represents constraint functions that are structural responses such as nodal displacements and volume of the structure. Right superscripts L and u in Equations (7b) and (7c) refer to lower and upper bounds, respectively. The perturbation vector approach is used in this paper such that the components of design variable vector $\mathbf{y}$ are related to structure shape by the following formula:

$$\mathbf{x} = \mathbf{x}^0 + \sum_{i=1}^{m} \mathbf{P}^i y^i \tag{8}$$

where $\mathbf{x}^0$ and $\mathbf{x}$ refers to the geometric configure of the structure before and after the shape optimization, respectively, $\mathbf{P}^i$ and $y^i$ are $i$th perturbation vector and $i$th design variable, respectively. $\mathbf{P}^i$ is defined by the basis function in Equation (2) and the nodal surface normal at the image of each knot. $m$ denotes the number of design variables in vector $\mathbf{y}$.

Note that Equation (7) defines only one category of shape optimization problems. Our freeform perturbation scheme is not tightly tied to this category. More accurately speaking, design parameterization and selection of design criteria are independent. Therefore, the technique introduced in this paper should be also suited in some other problems such as minimization of frequency response, uniform stress or minimization of stress. Since our approach is surface-oriented, it is naturally compatible with the BEM and can be easily implemented with the meshfree or meshless analysis method.

 

In this paper, the freeform shape optimization is carried out by using the optimizers available in OptiStruct$^{TM}$ [1], a dual method and a primal feasible directions method that are both based on the convex linearization of the design space.

## 5. NUMERICAL EXPERIMENTS

The proposed approach was implemented in VC++ and tested on a Pentium III HP PC. The surface partitioning algorithm in Section 2 has a time complexity $O(n \log n)$, where $n = \max(N_e, N_v)$, $N_e$ and $N_v$ are the numbers of elements and vertices in a mesh $M$, respectively [78]. The time complexity for Algorithm 1.1 is $O(s^2 t)$, where $s$ is the width of the overlaid grid mesh and $t$ is the iteration number of Laplacian smoothing that is dependent upon the geometric complexity of each surface patch. Algorithm 1.2 is basically a traversal over the smoothed grid mesh such that it takes $O(s^2)$ time. However, in order to test if each grid vertex is in one element of the 2D projected surface patch, we need to use a quick sort routine to sort the 2D projected surface patch on the basis of the bounding box position of each element, which takes $O(N_e \log N_e)$ time. In Algorithm 2, Step 1 takes $O(N_e \log N_e)$ time and in the worst case Step 3 takes $O(s^2 N_e)$, while the time cost of Step 2 is negligible.

Four practical examples are used to demonstrate the effectiveness of the proposed approach in the freeform shape optimization of different structures represented by arbitrary polygonal meshes. Our first example is a control arm that is a typical structure encountered in the structure design of automobiles, as shown in Figure 4(a). It contains one constant-cross-section beam (at the rear) and two tapered beams (along the sides). The non-design elements are shown in dark grey colour and the design elements are in light grey with the total number of hexa elements = 1730.

Figure 4(b) illustrates the surface partitioning results after removing all non-design elements. Among all 38 partitioned patches, users have an option to choose certain number of patches for a freeform shape optimization. We do not want to automate this step, because it will provide tremendous flexibility to try out different combinations of shape design variables.

In our test, four partitioned surface patches were chosen, each of which corresponded to a side of the two tapered beams. Figures 4(c) and 4(d) demonstrate a chosen patch and the image of its corresponding knot set, respectively. The perturbation vectors of all shape design variables are generated automatically by the proposed approach. The optimized shape as two smooth I-shaped beams is shown in Figure 4(e). Note that only surface elements are displayed in Figure 4(e) with the volume elements being omitted. With the external loading and material volume fixed, the weighted compliance of the entire structure is reduced from 9.7e4 to 8.0e4, which is 17.5% reduction. If we do not use our freeform perturbation scheme and use the conventional scheme of perturbing each FE node as independent shape design variable, only 12% reduction in the weighted compliance can be achieved with the same magnitude of perturbation and external loading. However, this performance improvement is priced at a higher computation time (4.45 min), compared to 1.35 min in the conventional scheme. Note that both time costs are measured only in the shape optimization without including the CPU time spent in surface partitioning (0.12 min) and parameterization (0.075 min) by using our approach.

Even though the conventional approach of perturbing each finite element node has been abandoned for over 20 years [24, 79], we use it as a reference to indicate how far the optimization result can be improved by using our approach, and to show the difference in the smoothness of

 

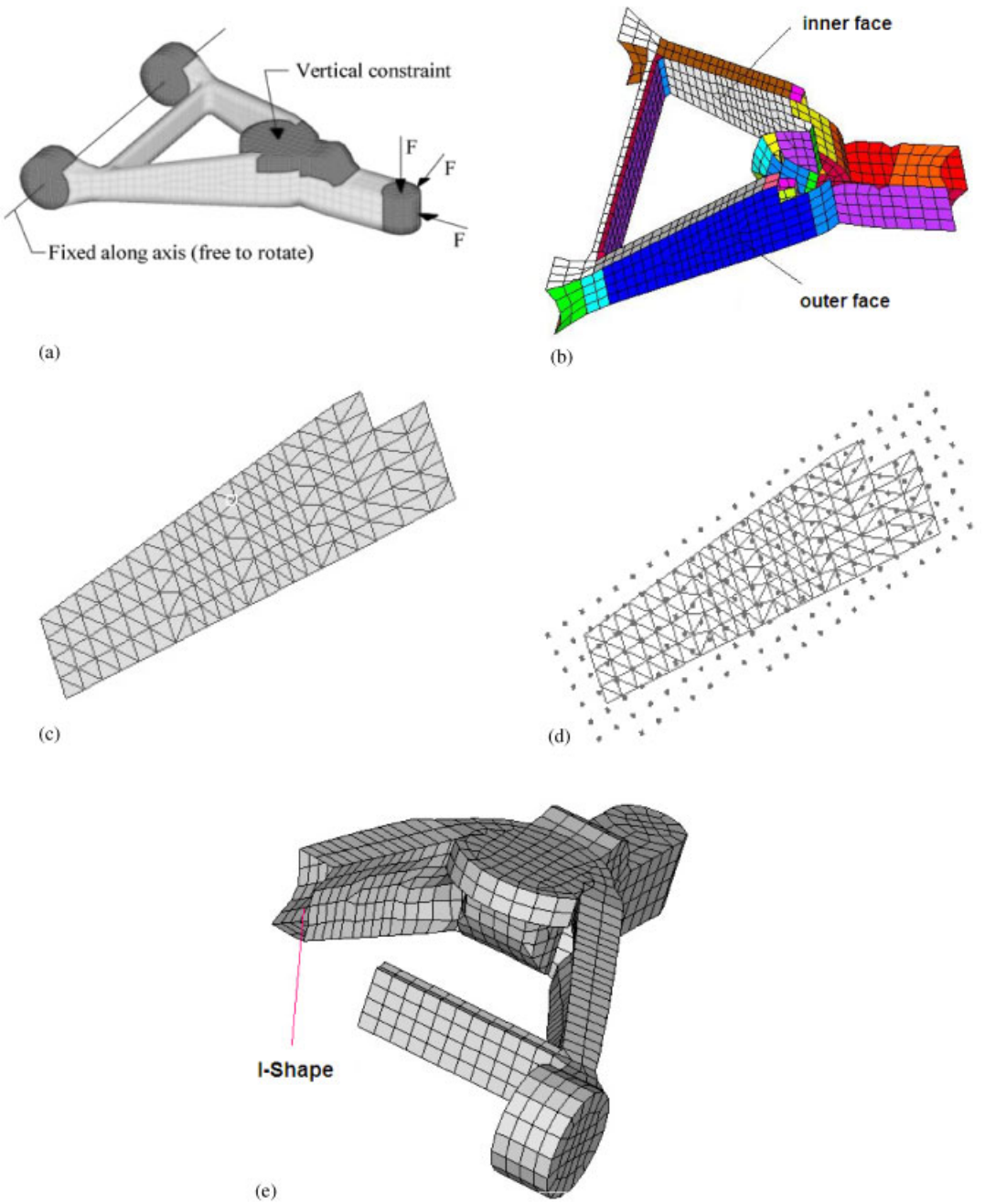


Figure 4. Freeform shape optimization of a control arm (w.c. = 9.7e4 → 8.0e4): (a) original FE model; (b) surface partitioning; (c) one patch for freeform optimization; (d) images of the knot set; and (e) optimized shape (I-beam).

 

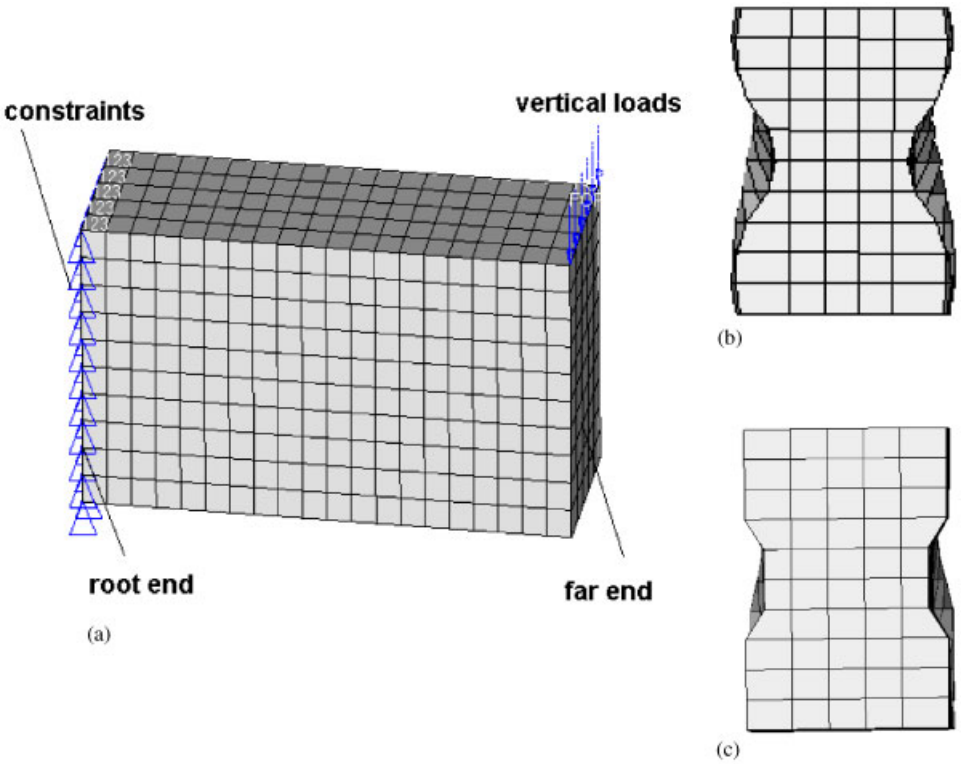


Figure 5. Freeform shape optimization of a simple beam: (a) FE model; (b) optimized cross-section by our approach; and (c) optimized cross-section by the conventional approach.

resulting surfaces. With complex mesh models, the bottleneck of a shape optimization process is frequently due to a huge amount of preparation time for B-spline representation, meshing, remeshing, and coordinating between B-spline and mesh. It is not uncommon for the preprocessing time to take days or weeks to complete, sometime even a year with a complex system such as an airplane [26, 27]. One main benefit of our approach over the B-spline approach is that the preprocessing time for a single mesh model can be reduced to the magnitude of minutes instead of hours or days.

Figure 5(a) shows our second test example, a finite element model of a beam with a total number of hexa elements = 1000 [21]. Even though this is a very simple example, it does provide an excellent illustration about the optimized shape as a smooth cross-section. We choose its two side faces as surface patches for freeform shape optimization. Our approach is compared with the conventional approach of perturbing each single finite element node, as illustrated in Figures 5(b) and 5(c). With the same magnitude of perturbation, the objective function is optimized 7.6% $(= (119 - 110)/119)$ more by using our approach, compared to the conventional one. However, the conventional scheme takes only 0.65 min to finish, while our approach needs 3.1 min in the freeform shape optimization alone without counting the CPU time used in surface partitioning (0.024 min) and parameterization (0.042 min). The

 

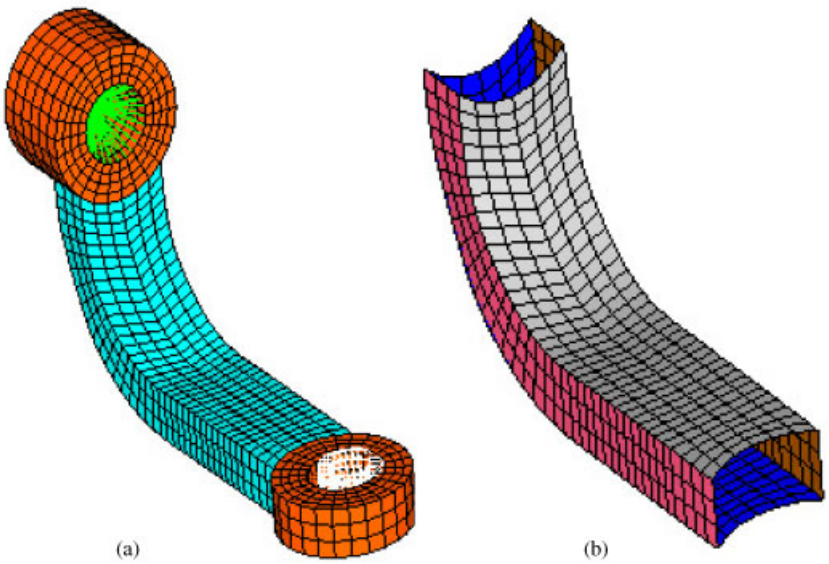


Figure 6. A curved beam: (a) original FE model; and (b) surface partitioning.

optimized shape produced by our approach tends to be smoother than that generated by the conventional one, which supports the statement that our approach well maintains the smoothness of the original surface patches. Unlike the B-spline approach, our approach does not need the B-spline surface representation as a precondition that may not easily be satisfied in some engineering applications.

The third test example is a curved beam in Figure 6(a) as a base model with the total number of elements = 1603. The left end of this beam is constrained, while the right end is subject to some pressure loads. The surface partitioning of this mesh is shown in Figure 6(b). Note that two ends of the original model are not shown in Figure 6(b), because they are non-design elements as a connecting part.

Figures 7 illustrates a side face of this beam and the images of knots associated with this face with different knot densities, which can be easily controlled by a user input integer $n$ in Step 2 of Algorithm 1.1, because the knot density is proportional to $2^n$. The optimized shape at different knot densities is shown in Figure 8, which indicates that we can achieve a kind of multiresolution shape optimization by controlling the users' input integer $n$. Note that the 'multiresolution' herein does not mean a multigrid method for partial differential equations, and instead stands for a multiresolution shape perturbation that ranges from global to local deformation.

Table IV shows the results of the multiresolution freeform shape optimization of the curved beam. Column 2 represents the number of shape design variables associated to a side face of the beam, while column 3 refers to the CPU time for the freeform shape optimization. The runtimes for surface partitioning and parameterization are 0.034 min and 0.064 min, respectively. From this table, it is interesting to note that with this beam model, low resolution shape optimization ($n = 3$) is less computationally expensive and provides a poorly optimized value

 

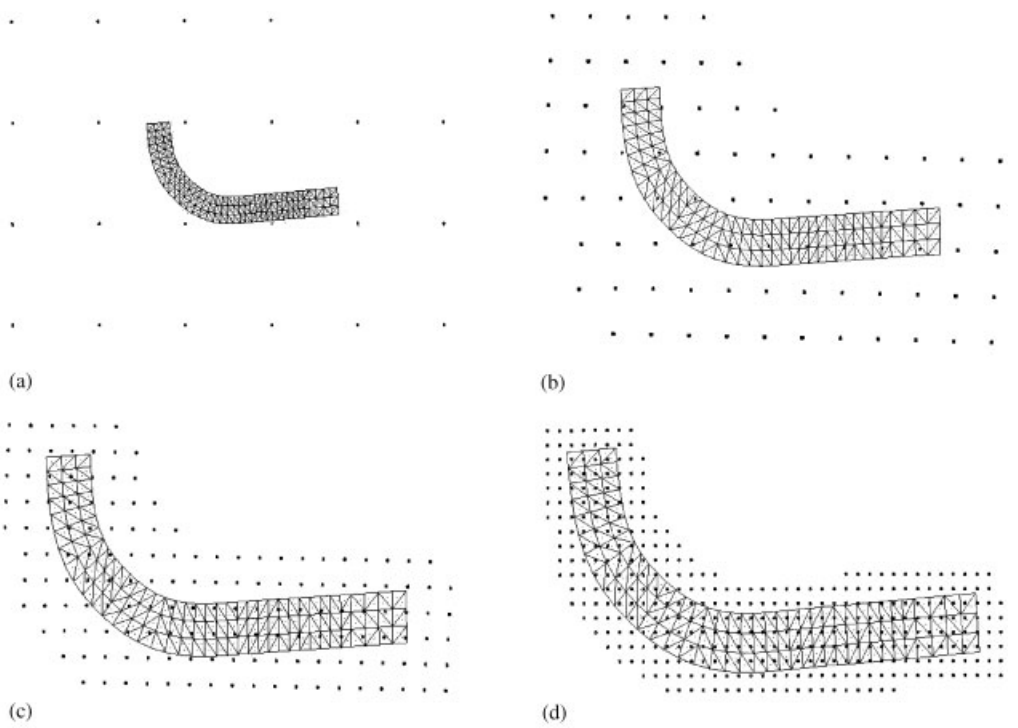


Figure 7. Multiresolution knot distribution of a side face of the beam in Figure 6 with the density of knots being proportional to $2^n$ in the parameter space.

Table IV. Multiresolution freeform shape optimization of a curved beam.

| $n$ | No. of shape design variables | CPU time (min) | Initial weighted compliance | Optimized weighted compliance |
|---|---|---|---|---|
| 3 | 22 | 0.78 | 241 | 218 |
| 4 | 79 | 0.88 | 241 | 214 |
| 5 | 156 | 0.95 | 241 | 214 |
| 6 | 384 | 1.2 | 241 | 214 |

of the objective function (i.e. weighted compliance), while high resolution shape optimization ($n = 4, 5, 6$) takes longer computational time but produces a better optimization in terms of the objective function. However, if we make the distribution of knots over dense, our freeform shape optimization scheme will be degenerated to the conventional scheme of perturbing each individual finite element node independently, because the influence zone of each knot in the parameter space is shrunk to contain only one finite element node or none. As indicated in Example 2, the conventional scheme provides a poor result of shape optimization with undesirable zigzag surfaces. Furthermore, there is no difference among the resolutions ($n = 4, 5, 6$)

 

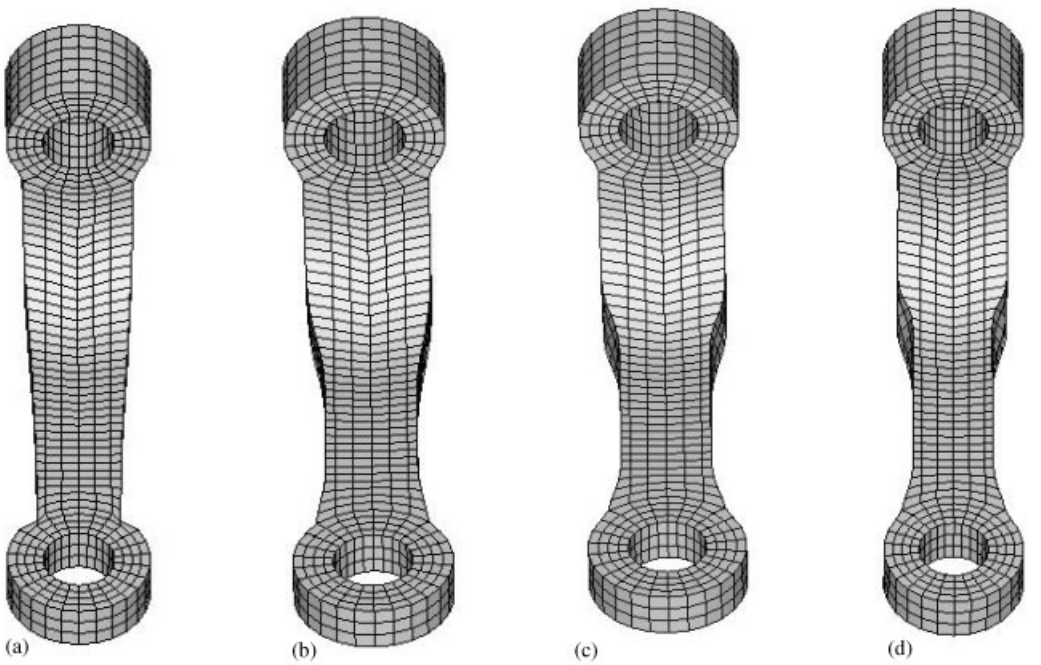


Figure 8. Multiresolution optimized shape of the beam in Figure 6.

in terms of the optimized weighted compliance, which suggests that there may exist an optimal resolution ($n = 4$ in this example) with which we can use a relatively low resolution to achieve a reasonably good optimization of the objective function. How to determine this optimal resolution for a freeform shape optimization is a subject of future work.

From Figure 8, it can be seen that low-resolution freeform shape optimization provides a smooth global shape change, while high-resolution shape optimization leads to a smooth local shape change. Since the density of knot images is entirely independent upon the vertex density of the polygonal mesh, a wide range of deformation patterns from global to local deformation can be easily achieved over each surface patch. This is another benefit of our approach, compared to the B-spline approach.

Figure 9 gives the results of a complex structure with over 90 thousand elements. In Figure 9(a), the images of knots associated to a 3D curved surface patch is displayed in the object space, while Figure 9(b) shows the optimized shape due to the freeform shape optimization on this surface patch with a reduction in the weighted compliance from 7.5e4 to 6.9e4. The CPU time for this freeform shape optimization alone is 40.6 min, while the runtimes for surface partitioning and parameterization are 1.97 and 0.8 min, respectively.

In this paper, we mathematically prove that our freeform perturbation scheme preserves the smoothness of original surface patches, but fail to prove why our scheme can achieve better performance than the conventional one in terms of the optimized value of the objective function. A possible explanation is due to the incapability of the optimization algorithm to find the optimum or the non-convexity of the optimization problem.

 

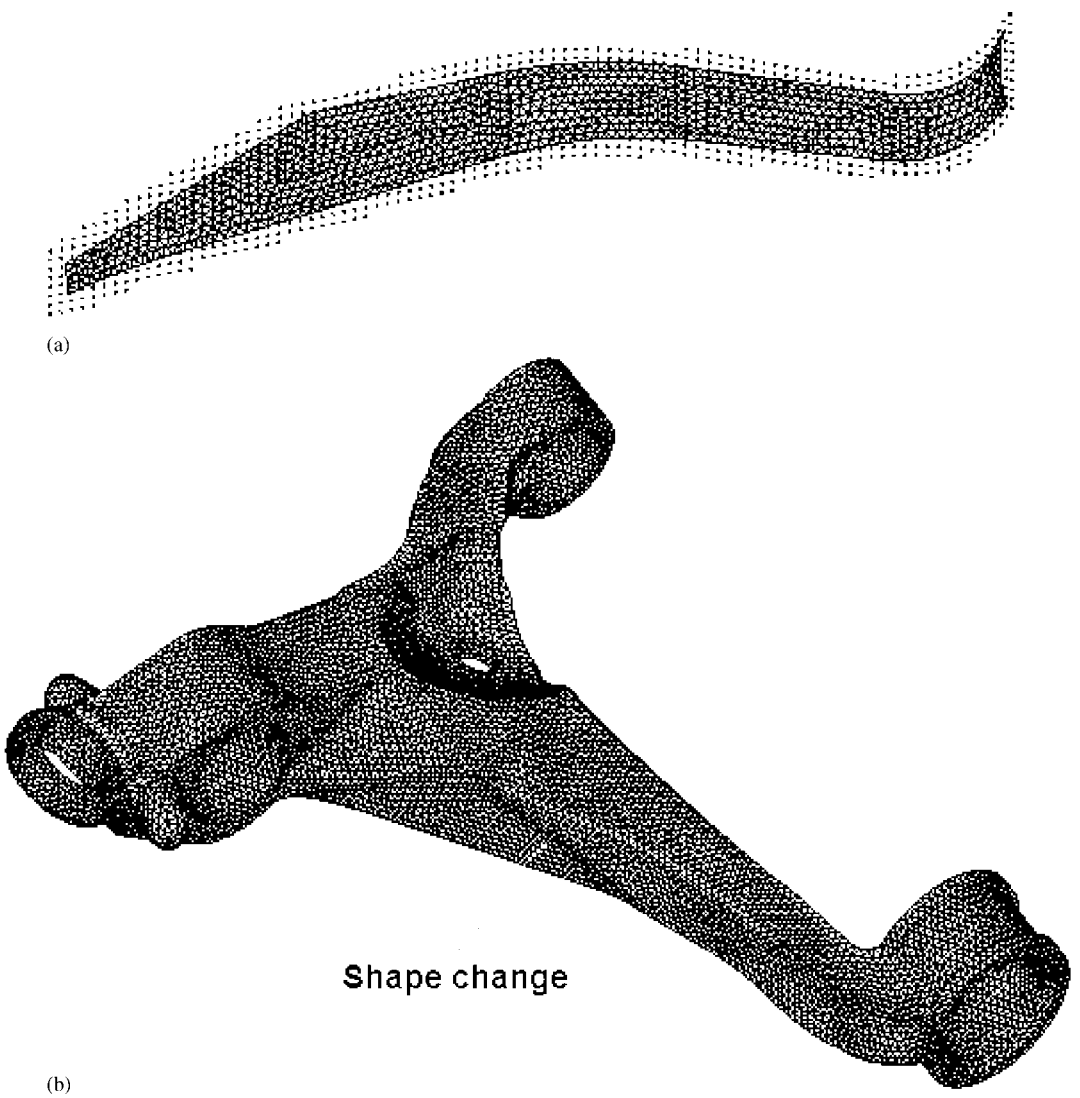


Figure 9. Freeform shape optimization of a curved connector (weighted compliance = 7.5e4 → 6.9e4): (a) images of the knot set in the object space; and (b) optimized shape.

The topography method in Optistruct$^{TM}$ [8] can be used as a way to perform freeform shape optimization. However, the basic perturbation function is a piecewise $C^1$ continuous line segments, as shown in Figure 10. Overall, that method can guarantee only $C^0$ continuity over each surface patch, which is definitely not sufficient for some engineering problems. Without the preservation of smoothness over each surface patch, a freeform shape optimization becomes less meaningful in some cases.

We implement our algorithms in such a way that users have flexibility to decide which surface patch, after the surface partitioning, is subject to freeform optimization.

 

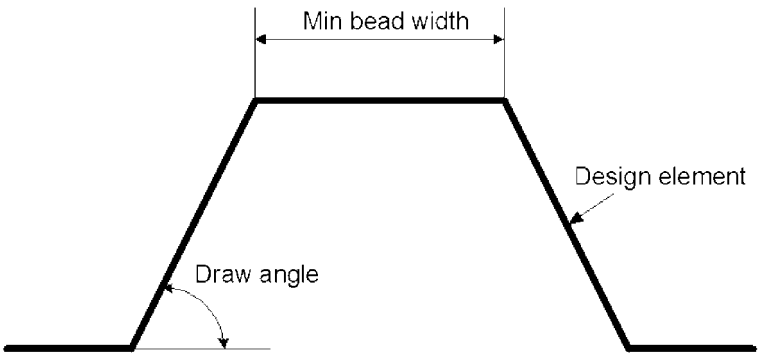


Figure 10. A basic perturbation function of topography method in OptiStruct$^{TM}$.

Constraints on the shape modifications are an important issue in engineering designs. In this paper, only the surface smoothness is imposed in our freeform shape optimization scheme, which may not be enough in many cases. In order to impose a regular set of design and manufacturing constraints in a shape optimization process, the authors have completed a study on feature recognition on finite element meshes [78], and the design and manufacture features will be considered as meaningful engineering constraints in a so-called feature-based optimization. The combination of freeform optimization with feature-based and parametric optimizations is under the investigation.

## 6. CONCLUDING REMARKS

In this paper, we propose a new scheme for freeform shape optimization on arbitrary polygonal meshes. Compared to the B-spline approach, one main advantage of our approach is to reduce the preparation time for shape optimization to the magnitude of minutes instead of hours or days. Another benefit is a wide range of perturbation patterns from global to local deformation in a multi-resolution manner without a need for remeshing. In comparison with the conventional method of perturbing each individual finite element node, our approach produces a better optimization result at a price of higher computation cost.

ACKNOWLEDGEMENTS

This work is supported in part by University of Michigan OVPR and Rackham research grants as well as University of Michigan-Dearborn RDEEF grant. The author would like to express his sincere gratitude to Dr. Alla Sheffer at Technion for her generosity in providing the source code of the ABF algorithm and related data models. Thanks also go to anonymous reviewers who provided many valuable comments for improving the quality of this paper.

## REFERENCES

1. ANSYS Inc. *ANSYS/Structural Product Features*. 2002.

 

2. Choi KK, Chang KW. A study of design velocity field computation for shape optimal design. *Finite Elements in Analysis and Design* 1994; **15**:317–341.
3. EDS. *IDEAS Version 9*, 2002.
4. Falk A, Barthold FJ, Stein E. A hierarchical design concept for shape optimization based on the interaction of CAGD and FEM. *Structural Optimization* 1999; **18**:12–23.
5. Hardee E, Chang KH, Tu J, Choi KK, Grindeanu I, Yu X. CAD-based shape design sensitivity analysis and optimization. *Advances in Engineering Software* 1999; **30**:153–175.
6. Parametric Technology Corp. *Pro/MECHANICA Design Study Reference*. 1996.
7. Structural Research & Analysis Corp. *COSMOS/M Features Introduction*. 2002.
8. Altair Engineering Inc. *OptiStruct User's Guide*. 2001.
9. Hinton E, Ozakca M, Rao NVR. An integrated approach to structural shape optimization of linearly elastic structures. Part II: shape definition and adaptivity. *Computing Systems in Engineering* 1991; **2**:41–56.
10. Moor GJ. Design sensitivity and optimization. *MSC/NASTRAN User's Guide*, vol. 68. 1994.
11. Rietz A, Petersson J. Simultaneous shape and thickness optimization. *Structural and Multidisciplinary Optimization* 2001; **23**:14–23.
12. Schleupen A, Maute K, Ramm E. Adaptive FE-procedures in shape optimization. *Structural and Multidisciplinary Optimization* 2000; **19**:282–302.
13. Vanderplaats Research & Development Inc. *Genesis User's Manual*. 2002.
14. Samareh JA. Survey of shape parameterization techniques for high-fidelity multidisciplinary shape optimization. *AIAA Journal* 2001; **39**(5):877–884.
15. Tafreshi A. Shape design sensitivity analysis of 2D anisotropic structures using the boundary element method. *Engineering Analysis with Boundary Elements* 2002; **26**:237–251.
16. Tafreshi A. Shape optimization of anisotropic structures using the boundary element method. *Journal of Strain Analysis for Engineering Design* 2003; **38**(3):219–232.
17. Zhang Q, Mukherjee S, Chandra A. Shape design sensitivity analysis for geometrically and materially nonlinear problems by the boundary element method. *International Journal of Solids and Structures* 1992; **29**:2503–2525.
18. Bobaru F, Mukherjee S. Meshless approach to shape optimization of linear thermoelastic solids. *International Journal for Numerical Methods in Engineering* 2002; **53**(4):765–796.
19. Grindeanu I, Choi KK, Chen JS, Chang KH. Shape design optimization of hyerelastic structures using a meshless method. *AIAA Journal* 1999; **37**(8):990–997.
20. Kim NH, Choi KK, Chen JS, Park YH. Meshless shape design sensitivity analysis and optimization for contact problem with friction. *Computational Mechanics* 2000; **25**(2–3):157–168.
21. Shen J, Yoon D. Freeform shape optimization of regular quad meshes. *Proceedings of 8th International Conference on Computer Aided Optimum Design of Structures*, 2003; 221–230.
22. Bendsoe MP. Optimal shape design as a material distribution problem. *Structural Optimization* 1989; **1**: 193–202.
23. Shen J, Maxim B, Akingbehin K. An accurate feature-preserving algorithm for surface denoising of polygonal meshes. *ACM Transactions on Graphics* 2003; accepted.
24. Braibant V, Fleury C. Shape optimal design using B-spline. *Computer Methods in Applied Mechanics and Engineering* 1984; **44**:247–267.
25. Jameson A. Re-engineering the design process through computation. *35th Aerospace Sciences Meeting and Exhibit*, 1997, *AIAA Paper 97-0641*.
26. Samareh JA. Multidisciplinary aerodynamic-structural shape optimization using deformation. *AIAA Paper No. 2000-4911 LTRS*, 2000.
27. Samareh JA. Novel multidisciplinary shape parameterization approach. *Journal of Aircraft* 2001; **38**(6): 1015–1024.
28. Barnard ST, Simon HD. Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency*: *Practice and Experience* 1994; **6**:101–117.
29. Farhat C, Lesoinne M. Automatic partitioning of unstructured meshes for parallel solution of problems in computational mechanics. *International Journal for Numerical Methods in Engineering* 1993; **36**:745–764.
30. Kiwi M, Spielman D, Teng SH. Min-max-boundary domain decomposition. *Theoretical Computer Science* 2001; **261**(2):253–266.
31. Nour-Omid B, Raefsky A, Lyzenga G. Solving finite element equations on concurrent computers. *Parallel Computations and their Impact on Mechanics*. ASME: New York, 1986; 209–227.

 

32. Pothen A, Simon HD, Lion KP. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Mathematical Analysis* 1990; **11**:430–452.
33. Simon HD. Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering* 1991; **2**(2/3):135–148.
34. Teng SH. Points, spheres, and separators, a unified geometric approach to graph partitioning. *Ph.D. Dissertation*, School of Computer Science, Carnegie Mellon University, 1991.
35. Vaughan C. Structural analysis on massively parallel computers. *Computing Systems in Engineering* 1991; **2**(2/3):261–267.
36. Williams RD. Performance of dynamic load balancing algorithms for unstructured mesh calculations. *Concurrency: Practice and Experience* 1991; **3**:457–481.
37. Bajaj C, Dey TK. Convex decomposition of polyhedra and robustness. *SIAM Journal on Computing* 1992; **21**:339–364.
38. Chazelle B. Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm. *SIAM Journal on Computing* 1984; **13**:488–507.
39. Chazelle B, Palios L. Triangulating a non-convex polytope. *Discrete and Computational Geometry* 1990; **5**:505–526.
40. Rappoport A. The extended convex differences tree (ECDT) representation for N-dimensional polyhedra. *International Journal of Computational Geometry and its Applications* 1991; **1**(3):227–241.
41. Ruppert J, Seidel R. On the difficulty of triangulating three-dimensional non-convex polyhedra. *Discrete and Computational Geometry* 1992; **7**:227–253.
42. Chazelle B, Dobkin D, Shouraboura N, Tal A. Strategies for polyhedral surface decomposition: an experimental study. *Computational Geometry—Theory and Applications* 1997; **7**:327–342.
43. Chazelle B, Palios L. Decompoing the boundary of a nonconvex polyhedron. *Algorithmica* 1997; **17**:245–265.
44. Eberly D. *Ridges in Image and Data Analysis*. Kluwer Academic: Dordrecht, 1996.
45. Koenderink J, van Doorn A. The structure of two-dimensional scalar fields with applications to vision. *Biological Cybernetics* 1979; **33**:151–158.
46. Besl P. *Surfaces in Range Image Understanding*. Springer: Berlin, 1988.
47. Faugeras D, Hebert M. A 3-D recognition and positioning algorithm using geometric matching between primitive surfaces. *Proceedings of the International Joint Conference on Artificial Intelligence*, 1983; 996–1002.
48. Fisher RB, Fitzgibbon AW, Eggert D. Extracting surface patches from complete range descriptions. *Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, 1997; 148–154.
49. Shen J, Yoon D. A new scheme for efficient and direct shape optimization of complex structures represented by polygonal meshes. *International Journal for Numerical Methods in Engineering* 2004; **58**(4):2201–2223.
50. Barsky BA. The beta-spline. *A local representation based on shape parameters and fundamental geometric measures*. University of Utah, 1981.
51. Piegl L, Tiller W. *The NURBS Book*. Springer: Berlin, 1997.
52. Riesenfeld RF. Application of B-spline approximation to geometric problems of computer-aided design. *Ph.D.*, Syracuse University, 1972.
53. Anderson WK, Venkantakrishnan V. Aerodynamics design optimization on unstructured grids with a continuous adjoint formulation. *AIAA Paper 97-0643*, 1997.
54. Baysal O, Ghayour K. Continuous adjoint sensitivities for general cost functions on unstructured meshes in aerodynamic shape optimization. *Proceedings of 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 1998; 1483–1491.
55. Cosentino GB, Holst TL. Numerical optimization design of advanced transonic wing configurations. *Journal of Aircraft* 1986; **23**(3):193–199.
56. Jameson A, Reuther J. A comparison of design variables for control theory based airfoil optimization. *Proceedings of 6th International Symposium on Computational Fluid Dynamics*, vol. 4, 1995; 101–107.
57. Schramm U, Pilkey WD. Structural shape optimization for the torsion problem using direct integration and B-spline. *Computer Methods in Applied Mechanics and Engineering* 1993; **107**:251–268.
58. Schramm U, Pilkey WD, DeVries RI, Zebrowski MP. Shape design for thin-walled beam cross sections using rational B-splines. *AIAA Journal* 1995; **33**(11):2205–2211.
59. Coquillart S. Extended free-form deformation: a sculpting tool for 3D geometric modeling. *ACM SIGGRAPH Computer Graphics Proceedings* 1990; **24**(4):187–196.
60. Hsu WM, Hughes JF, Kaufman H. Direct manipulation of free-form deformations. *ACM SIGGRAPH Computer Graphics Proceedings* 1992; **26**(2):177–184.


 

61. Sederberg TW, Parry SR. Free-form deformation of solid geometric models. *ACM SIGGRAPH Computer Graphics Proceedings* 1986; **20**(4):151–160.
62. Perry E, Balling R. A new morphing method for shape optimization. *AIAA Paper 98-2896*, 1998.
63. Perry E, Balling R, Landon M. A new morphing method for shape optimization. *AIAA Paper 98-4907*, 1998.
64. Yeh TP, Vance JM. Applying virtual reality techniques to sensitivity-based structural shape design. *Proceedings of 1997 ASME Design Engineering Technical Conferences, No. DAC*-3765, 1997; 1–9.
65. Hirota G, Maheshwari R, Lin M. Fast volume-preserving freeform deformation using multi-level optimization. *Computer Aided Design* 2000; **32**(8–9):499–512.
66. Qin H, Terzopoulos D. Dynamic NURBS swung surfaces for physics-based shape design. *Computer Aided Design* 1995; **27**(2):111–127.
67. Welch W, Witkin A. Free-form shape design using triangulated surfaces. *ACM SIGGRAPH Computer Graphics Proceedings* 1994; **28**:247–256.
68. Gallier J. *Curves and Surfaces in Geometric Modeling, Theory and Algorithms*. Morgan Kaufmann Publishers: San Francisco, CA, 2000.
69. Eck M, De Rose T, Duchamp T, Hoppe H, Lounsbery M, Stuetzle W. Multiresolution analysis of arbitrary meshes. *ACM SIGGRAPH Computer Graphics Proceedings* 1995; **29**:173–181.
70. Floater MS. Parameterization and smooth approximation of surface triangulations. *Computer Aided Geometric Design* 1997; **14**:231–250.
71. Marcum DL, Gaiter JA. Unstructured surface grid generation using global mapping and physical space approximation. *8th International Meshing Roundtable*, 1999; 397–406.
72. Levi B, Mallet JL. Non-distorted texture mapping for sheared triangulated meshes. *ACM SIGGRAPH Computer Graphics Proceedings* 1998; **32**:343–352.
73. Zigelman G, Kimmel R, Kiryati N. Texture mapping using surface flattening via multi-dimensional scaling. *IEEE Computer Graphics and Applications* 2002; **8**(2):198–207.
74. Sheffer A, Sturler E. Parameterization of faceted surfaces for meshing using angle based flattening. *Engineering with Computers* 2001; **17**(3):366–375.
75. Sheffer A, Sturler E. Smoothing an overlay grid to minimize linear distortion in texture mapping. *ACM Transactions on Graphics* 2002; **21**(4):874–890.
76. Degand C, Farhat C. A three-dimensional torsional spring analogy method for unstructured dynamic meshes. *Computers and Structures* 2002; **80**:305–316.
77. Farhat C, Degand C, Koobus B, Lesoinne M. Torsional springs for two-dimensional dynamic unstructured fluid meshes. *Computer Methods in Applied Mechanics and Engineering* 1998; **163**:231–245.
78. Shen J, Yoon D, Song Y, Zhao D. A Surface partitioning algorithm of finite element meshes for shape optimization. *Finite Elements in Analysis and Design* 2004, submitted.
79. Haftka RT, Grandhi RV. Structural shape optimization—a survey. *Journal of Computer Methods in Applied Mechanics and Engineering* 1986; **57**(1):91–106.

 