INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2003; **58**:2201–2223 (DOI: 10.1002/nme.859)

A new scheme for efficient and direct shape optimization of complex structures represented by polygonal meshes

Jie Shen*,† and David Yoon

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

SUMMARY

In this paper, a new shape optimization approach is proposed to provide an efficient optimization solution of complex structures represented by polygonal meshes. Our approach consists of three main steps: (1) surface partitioning of polygonal meshes; (2) generation of shape design variables on the basis of partitioned surface patches; and (3) gradient-based shape optimization of the structures by reducing a weighted compliance among all load cases. The main contributions of this paper include (i) that our approach can be directly applied on polygonal meshes with the reduction of design variables or decision variables by 10 to 1000 times, compared to the conventional design variable scheme of using each mesh node; (ii) our perturbation scheme is mathematically proven with respect to maintaining the smoothness of each surface patch, except on its boundary; and (iii) overall, our approach can be used to automate time-consuming shape optimization of polygonal meshes to a greater extent. Numerical experiments have been conducted and the results indicate the effectiveness of the approach. Copyright © 2003 John Wiley & Sons, Ltd.

KEY WORDS: shape optimization; polygonal mesh; structure; surface partitioning

1. INTRODUCTION

Shape optimization is a process to seek an optimal geometric configuration under given constraints without changing the topology of structures. A considerable amount of work in this area has been conducted in the past and can be categorized as follows:

(1) Objective functions: minimum stress [1], minimum compliance [2], uniform stress [3], etc.
(2) Optimization methods: gradient-based method [4, 5], genetic method [6], dynamic simulation [7], heuristic evolution [3], etc.
(3) Geometry: mesh-based [8], CAD-based [9, 10].

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

Received 13 September 2002
Revised 20 March 2003
Accepted 17 April 2003

(4) Design variables: basis vector [11–15], parametric representation [16–19], natural design variable [20–22].

Finite element mesh is one of the most commonly used data formats in engineering analysis and optimization. Even though many finite element meshes are generated from their corresponding CAD models, in practice analysts may face situations in which only polygonal meshes are available without knowing any information about their underlying geometry in CAD formats. These situations include (1) meshes are generated from a mesher that does not need a CAD input model; (2) meshes are generated by digital sensing systems such as laser scanning equipment used in reverse engineering. If the meshes contain some surface noises, we assume that a certain surface smoothing algorithm can be applied to these meshes; and (3) analysts get meshes from a third party or from a set of legacy meshes without a corresponding CAD model, etc. Lack of underlying geometric information imposes the following difficulties on shape optimization of these types of meshes:

(1) Since we have very little information about the geometric configuration of a structure, if each finite element node is used as a design variable [23], a large-scale shape optimization of complex structures becomes computationally too expansive due to a great number of design variables.
(2) If each finite element node is used as a design variable, it is very difficult to maintain a smooth shape boundary during a shape optimization process [24]. Automatic conversion from polygonal meshes with sharp edges to B-spline patches is a very hard problem to solve.
(3) Even though commercial software (Optistruct$^{\text{TM}}$ and Genesis$^{\text{TM}}$) provide interactive tools to assist the process of creating design variables, the functionality is quite limited, because these tools rely entirely on users to manually select a number of elements for each design variable. With complex structures, this process is time-consuming and error-prone. Optistruct is the trademark of Altair Engineering Inc. and Genesis is the trademark of Vanderplaats Research & Development Inc.

To address the above problems, in this paper we propose a new scheme to facilitate a shape optimization directly on finite element meshes of complex structures. Our scheme consists of three main steps. First, surface partitioning is automatically carried out on finite element mesh models. Secondly, each partition can be assigned as a design variable. But users have an option to choose which partition will be a design variable in the current shape optimization setting. A special perturbation scheme is applied to each design variable such that the smoothness of the original surface patches is maintained. Lastly, a shape optimization is conducted.

In this study, we used only a gradient-based commercial optimizer in Optistruct$^{\text{TM}}$ and restricted the objective function of the optimization to the weighted compliance (i.e. strain energy) of the structure with the total volume of the structure being unchanged. However, our approach can also be applied to other objective functions and optimization algorithms.

The main contribution of this paper is to propose a new approach to efficient and direct shape optimization on finite element meshes of complex structures. This paper is organized as follows. In Section 2, some notations and definitions are introduced. Our surface partitioning scheme is given in Section 3. A new approach to automatic and direct shape optimization on finite element meshes is presented in Sections 4 and 5, and the results of numerical experiments are given in Section 6. Finally, some conclusions are drawn in Section 7.

2. NOTATIONS AND DEFINITIONS

Definition 1
A polygonal mesh $M(\{\mathbf{v}_i\}, \{\mathbf{t}_j\})$ is defined by a set of vertices $\{\mathbf{v}_i\}$ and a set of elements $\{\mathbf{t}_j\}$, i.e. polygons. Each vertex has three co-ordinates in $\mathbf{R}^3$, $[x_i \; y_i \; z_i]^{\mathrm{T}}$. Each element represents a flat face bounded by straight line segments, i.e. edges, and is specified by an m-tuple of vertices, $(\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_m)$.

Definition 2
The normal angle change between two adjacent elements means the angle formed by surface normals of two faces that share a common edge. Directional normal curvature is the normal curvature in the direction perpendicular to an element edge. Nodal curvature is a curvature at a vertex in M. Element curvature is an average of nodal curvatures of all vertices in an element. k_{12} is a new curvature index proposed in this paper to assist a surface partitioning process.

Definition 3
Propagation refers to a traverse process of a breath-first search over a surface mesh. Current vertex means the vertex at which the propagation front is located. Current element denotes the element in which the propagation front is located. Similarly, next vertex and next element refer to the location where the propagation front will visit next.

Definition 4
Compliance refers to the strain energy within the domain of a structure. Weighted compliance means a weighted summation of compliance among a number of load cases. Design element is a finite element whose shape can be changed in an optimization process, while the shape of non-design elements remain fixed. Design variables are the variables used in a shape optimization, and are also called decision variables in operation research. Design domain is a domain that contains all design elements of the structure.

3. AN ALGORITHM FOR SURFACE PARTITIONING OF POLYGONAL MESHES

Even though there are many surface partitioning algorithms in computational geometry [25–31] and in parallel computation [32–40], the focus of these algorithms is to obtain a set of convex entities or to generate equally sized subregions with minimized boundaries, which are not suited for the shape optimization. In the proposed surface partitioning, we divide the entire task into three main steps in a hierarchical manner as follows:

Algorithm 1: surface partitioning

(1) surface partitioning by G^1 discontinuity[‡]
(2) identification of flat regions
(3) surface partitioning by discontinuity of curvatures

[‡] G^1 discontinuity means that the first-order derivative is geometrically discontinuous.

3.1. Surface partitioning by G^1 discontinuity and identification of flat regions

We first partition the area on the basis of G^1 discontinuity lines. The algorithm proceeds in the following steps:

Algorithm 1.1: surface partitioning by G^1 discontinuity

```
(1) set up an element neighbour list for each surface element
(2) calculate the normal angle change between adjacent elements
(3) perform a breath-first search
    (3.1) initiate from an arbitrary surface element
    (3.2) propagate over the surface until a $G^1$ discontinuity line is encountered, which is
          identified by a condition: the normal angle change is greater than a user-specified
          angular threshold.
    (3.3) go back to step (3.1) and repeat this breath-first search over unprocessed regions
          until all surface elements are covered by a partition.
```

In our numerical experiments, 50° was found to be a reasonable value for the threshold of normal angle change in most cases. However, this threshold should be adjusted in some special situations. The magnitude of normal angle change reflects the discontinuity of surface normal of two adjacent elements. If an input surface is a non-manifold, special treatment is required for the above propagation. The rule of such treatment is that the propagation proceeds to the neighbouring element that has the smallest magnitude of normal angle change with the current element, if there are two or more neighbouring elements that share the same edge with the current element.

3.2. Identification of flat regions

The purpose of this step is to locate all flat regions within each surface patch generated by the algorithm in Section 3.1 such that the task of surface partitioning by curvature in Section 3.4 could be reduced. It proceeds in a similar way as the method given in Section 3.1:

Algorithm 1.2: identification of flat regions

```
(1) loop over each surface patch generated by Algorithm 1.1
    (1.1) perform a breath-first search
        (1.1.1) initiate from an surface element that has, w.r.t. each neighbouring ele-
                ment, a normal angle change that is less than a user-specified angular
                threshold for flat planes.
        (1.1.2) propagate over the surface until a boundary line of a flat plane is en-
                countered, which is identified by a condition: the normal angle change is
                greater than a user-specified angular threshold for flat planes.
        (1.1.2) go back to step (1.1.1) and repeat this breath-first search over unprocessed
                regions until all surface elements are processed.
    (1.2) group the remaining elements that do not belong to flat regions into one or more
          different surface patches by means of their connectivity.
```

The angular threshold for flat planes is set to a very small value, 1°. Similar to the threshold in Section 3.1, readers have an option to change it at their disposal.

3.3. Estimation of nodal curvatures

In order to conduct step 3 in Algorithm 1, estimation of surface curvatures is required. Since a polygonal mesh is an approximation to an underlying surface, surface curvatures can be estimated in either nodal or element formats, depending upon the requirement of how the approximated surfaces will be used in related computation. In this study, only nodal curvatures are used.

The procedures for determining surface curvatures at each vertex of M are as follows:

Algorithm 1.3: nodal curvatures

(1) Loop over each vertex **P** in M

(1.1) Determine a set of neighbouring vertices for vertex **P**.
The rule for selecting neighbouring vertices is to choose at least six different points. We first loop over all adjacent vertices that share an edge with **P**, and then look at the remaining vertices that are in a same element with **P**. If the number of vertices is still less than 6, as often in the case in which **P** is on a boundary, the centroid and the middle edge points of each neighbouring element are added into the set of neighbouring vertices for vertex **P**.

(1.2) Determine a tangent plane at vertex **P**.
The principal component method [41] is used to determine the tangent plane at vertex **P**. The covariance matrix of the set of neighbouring vertices is

$$\mathbf{CV} = \sum_{\mathbf{q}\in \mathrm{Nbhd}(\mathbf{P})} (\mathbf{q}-\mathbf{P})\otimes(\mathbf{q}-\mathbf{P}) \tag{1}$$

where Nbhd(**P**) is the set of neighbouring vertices at vertex **P** and $\otimes$ is outer product operator of vectors. A Jacobi transformation [42] can be used to determine eigenvectors $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ and eigenvalues $(\lambda_1 \geqslant \lambda_2 \geqslant \lambda_3)$ of the **CV**. $\mathbf{v}_3$ represents the normal direction of the tangent plane, $\mathbf{v}_1$ and $\mathbf{v}_2$ are the base vectors of orthogonal parameter co-ordinates in the tangent plane.

(1.3) Determine a local quadric co-ordinate patch.
In the local co-ordinate system $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$, a quadric co-ordinate patch:

$$z = f(x,y) = a_1x^2 + a_2xy + a_3y^2 + a_4x + a_5y + a_6 \tag{2}$$

is used to approximate the surface in the neighbourhood of vertex **P**. Here, co-ordinates (x, y) are measured along $(\mathbf{v}_1, \mathbf{v}_2)$ directions, and z co-ordinate is measured in the $\mathbf{v}_3$ direction. The least-squares estimation of six coefficients a_i is expressed as

$$\mathbf{BA} = \mathbf{Z} \tag{3}$$

in which

$$\mathbf{B}=\begin{bmatrix} x_1^2 & x_1y_1 & y_1^2 & x_1 & y_1 & 1.0 \\ x_2^2 & x_2y_2 & y_2^2 & x_2 & y_2 & 1.0 \\ \cdots & & & & & \\ x_n^2 & x_ny_n & y_n^2 & x_n & y_n & 1.0 \end{bmatrix}, \quad \mathbf{Z}=\begin{bmatrix} z_1 \\ z_2 \\ \cdots \\ z_n \end{bmatrix}, \quad \mathbf{A}=\begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_6 \end{bmatrix} \tag{4}$$

where n is the number vertices of Nbhd($\mathbf{P}$). If we solve this equation by using $\mathbf{A}=(\mathbf{B}^{\mathrm{T}}\mathbf{B})^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{Z}$, $\mathbf{B}^{\mathrm{T}}\mathbf{B}$ may be ill-conditioned. Thus, singular value decomposition [42] is used to solve $\mathbf{BA}=\mathbf{Z}$.

(1.4) Calculate curvatures at vertex $\mathbf{P}$.
By differential geometry [43], the curvatures at vertex $\mathbf{P}$ can be expressed as

$$K=4.0a_1a_3-a_2^2 \quad \text{and} \quad H=a_1+a_3 \tag{5a}$$

where K and H are Gaussian and mean curvatures, respectively. The maximum and minimum principle curvatures are then determined by

$$k_1=H+\sqrt{H^2-K} \quad \text{and} \quad k_2=H-\sqrt{H^2-K} \tag{5b}$$

The above procedures for calculating surface curvatures are supported by the following lemma and propositions.

Lemma 1
Suppose a surface with a point $\mathbf{P}$ on the surface. If x and y are orthogonal co-ordinates in the tangent plane at $\mathbf{P}$ and z is the co-ordinate in the normal direction to the surface at $\mathbf{P}$, the local quadric co-ordinate patch: $z=f(x,y)=a_1x^2+a_2xy+a_3y^2+a_4x+a_5y+a_6$ has the same principal curvature values (k_1,k_2) at $\mathbf{P}$ as the original surface.

Proof
If Equation (2) is used to approximate the surface neighbourhood at point $\mathbf{P}$, then the Gaussian and mean curvatures at point $\mathbf{P}$ can be expressed by $K=4.0a_1a_3-a_2^2$ and $H=a_1+a_3$. If we take $a_1=a_3=0.5H$ and $a_2=\sqrt{H^2-K}$, relationships in Equation (5a) will be satisfied. This means that we can find a surface of the form defined by Equation (2), which has given values of K and H. Similarly, since $k_1=H+\sqrt{H^2-K}$ and $k_2=H-\sqrt{H^2-K}$, then the relationships in Equation (5a) can be transformed to

$$\begin{aligned} k_1&=a_1+a_3+\sqrt{(a_1+a_3)^2-(4a_1a_3-a_2^2)} \\ k_2&=a_1+a_3-\sqrt{(a_1+a_3)^2-(4a_1a_3-a_2^2)} \end{aligned} \tag{6}$$

If we take $a_1=a_3=0.25(k_1+k_2)$ and $a_2=0.5(k_1-k_2)$, the above relationships will be satisfied, which means that we can find a surface of form (2) that has given values of k_1 and k_2. □

On the basis of Lemma 1, we can prove the following proposition.

Proposition 2
If the tangent plane at point **P** is determined by using the principal component method in Algorithm 1.3, the local quadric co-ordinate patch: $z = f(x, y) = a_1x^2 + a_2xy + a_3y^2 + a_4x + a_5y + a_6$ sufficiently represents all major types of surfaces: elliptic, hyperbolic, parabolic and planar, at the neighbourhood of point **P**.

Proof
According to differential geometry, major types of surfaces can be characterized by curvatures at each point **P** as follows:

- *elliptical* if $K(\mathbf{P}) > 0$,
- *hyperbolic* if $K(\mathbf{P}) < 0$,
- *parabolic* if $K(\mathbf{P}) = 0$ and $k_1(\mathbf{P}) \neq 0$ or $k_2(\mathbf{P}) \neq 0$,
- *planar* if $k_1(\mathbf{P}) = k_2(\mathbf{P}) = 0$,

where $K(\mathbf{P})$ refers to the Gaussian curvature at point **P**, etc. By Lemma 1, we can use a surface of form (2) to sufficiently represent a pair of arbitrarily valued principal curvatures (k_1, k_2) or a pair of arbitrarily valued Gaussian and mean curvatures (K, H). Thus, the local quadric co-ordinate patch of the form (2) can sufficiently represent all major types of surfaces listed above. □

Proposition 3
If the local quadric co-ordinate patch: $z = f(x, y) = a_1x^2 + a_2xy + a_3y^2 + a_4x + a_5y + a_6$ is used to approximate the surface S at the neighbourhood of point **P**, and if surface S is continuous at point **P**, the discrete curvatures in Equations (5a) and (6) approach or converge to the curvatures of surface S at point **P** in a limit.

Proof
If surface S is of class C^m, where $m \leqslant 2$, it is obvious that the above local quadric co-ordinate patch can accurately describe S and therefore has the same curvatures. When S is of class of C^n, where $n \geqslant 3$, let $\mathbf{x}(u, v)$ be a parameterization at point **P** of surface S, with $\mathbf{x}(0, 0) = \mathbf{P}$. Since $\mathbf{x}(u, v)$ is differentiable, the following Taylor's formula holds:

$$\mathbf{x}(u, v) = \mathbf{x}(0, 0) + \mathbf{x}_u u + \mathbf{x}_v v + 0.5(\mathbf{x}_{uu}u^2 + 2\mathbf{x}_{uv}uv + \mathbf{x}_{vv}v^2) + R \tag{7}$$

where the derivatives are taken at $(0, 0)$, i.e. point **P**, and the remainder R satisfies the following condition:

$$\lim_{(u,v)\to(0,0)} \frac{R}{u^2 + v^2} = 0$$

Thus, when (u, v) approaches $(0, 0)$, term R is negligible, compared to other terms on the right-hand side of Equation (7). It follows that Equation (7) without R has an exactly same quadric form as Equation (2) and the same formulas for all six coefficients as in Equation (2), because the coefficients in Equation (2) can be rewritten as

$$a_1 = 0.5z_{xx}(0, 0), \quad a_2 = z_{xy}(0, 0), \quad a_3 = 0.5z_{yy}(0, 0)$$

$$a_4 = z_x(0, 0), \quad a_5 = z_y(0, 0), \quad a_6 = z(0, 0)$$

Table I. Convergence of discrete curvature k_{12} to the theoretical value 1.0.

Model name	Sphere 5	Sphere 10	Sphere 15	Sphere 20	Sphere 30
Elements	28	152	324	600	1352
k_{12}	1.1	1.02	1.012	1.004	1.002

Thus, we can conclude that the discrete curvatures in Equations (5a) and (6) approach or converge to the curvatures of surface S at point $\mathbf{P}$ in a limit. □

A numerical experiment on a unit sphere with different degrees of discretization was conducted. The convergence of discrete curvature k_{12}, which is defined in Equation (8), is given in Table I. A similar convergence trend is observed with the Gaussian, mean and principal curvatures.

3.4. Surface partitioning by discontinuity of curvatures

On top of the above two passes in Sections 3.1 and 3.2, we can have a third pass of partitioning on the basis of curvature discontinuity. This pass is extremely important to curved surfaces. There are several important issues that need to be addressed before we introduce an algorithm for surface partitioning by discontinuity of curvatures.

3.4.1. Choices of curvatures. By differential geometry, we can use one of four well-known curvatures (Gaussian, mean, maximum and minimum principal curvatures) as an index to guide the surface partitioning process. However, through numerical experiments, we found that the following index is more suitable to the surface partitioning problem:

$$k_{12} = 0.5(|k_1| + |k_2|) \tag{8}$$

We cannot prove why the above index is better than the four well-known curvatures in a surface partitioning process. However, intuitively k_{12} is an average of curvatures at a point, similar to the total curvature $k_1^2 + k_2^2$ or the absolute curvature $|k_1| + |k_2|$. The problem with the Gaussian curvature is that it becomes zero whenever either k_1 or k_2 is zero. In such a situation, little amount of information is available to guide the surface partitioning process. Similarly, k_1 and k_2 may cancel each other if their signs are different in terms of the mean curvature. The maximum principal curvature or the minimum principal curvature alone does not provide enough information about curvatures along different directions at a point.

3.4.2. Directional nodal curvatures w.r.t. element edges. If nodal curvatures are used, normal curvature along the tangential directions that are perpendicular to different element edges is used directly to represent the directional curvature. By differential geometry, the calculation of normal curvature is given by

$$k_{\mathrm{n}}(\mathbf{t}) = \frac{\mathrm{II}(\mathbf{t})}{\mathrm{I}(\mathbf{t})} = \frac{e(x,y)a^2 + 2f(x,y)ab + g(x,y)b^2}{E(x,y)a^2 + 2F(x,y)ab + G(x,y)b^2} \tag{9}$$

where, I( ) and II( ) are the first and second fundamental forms in differential geometry. x and y are orthogonal co-ordinates in the tangent plane at a point $\mathbf{P}$ of M. $\mathbf{t}$ is a unit vector that is

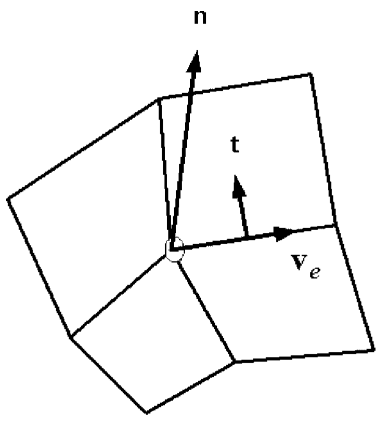
n
t
$\mathbf{v}_e$
$\mathbf{v}_e$: element edge vector
n: surface normal
t: unit tangential vector

Figure 1. Normal curvature w.r.t. an element edge.

in the tangential plane and perpendicular to an element edge $\mathbf{v}_e$ projected onto the tangential plane, as shown in Figure 1, with $\mathbf{t} = a\mathbf{x} + b\mathbf{y}$. The definitions of functions $e(\ )$, $f(\ )$, $g(\ )$, $E(\ )$, $F(\ )$, and $G(\ )$ can be found in a textbook of differential geometry [43]. $k_n(\mathbf{t})$ is the normal curvature at point $\mathbf{P}$ in the tangential direction $\mathbf{t}$, i.e. the normal curvature w.r.t. the element edge $\mathbf{v}_e$.

3.4.3. Curvature partitioning separator. With a surface patch obtained from executing steps 1 and 2 in Algorithm 1, curvature partitioning means that we intend to divide the surface patch into two or more subregions, and is essentially implemented by a breath-first search in which the normal curvature cross each element edge (i.e. in the direction perpendicular to that edge) is checked during the search propagation over the surface.

In order to conduct the curvature partitioning, a curvature threshold is required as a separator to assist the partitioning process. In this study, we use an average nodal curvature over the surface patch as a default value of the curvature threshold. One exceptional case to the average nodal curvature is that when all nodal curvatures over the surface patch are very close to each other, the following modification is used:

$$\mathrm{CR_t} = 3.0\mathrm{CR_{av}}, \quad \text{if diff\_ratio} < 1.8 \tag{10}$$

where

$$\text{diff\_ratio} = \frac{\mathrm{CR_{max}} - \mathrm{CR_{min}}}{\mathrm{CR_{av}}}$$

in which $\mathrm{CR_{max}}$, $\mathrm{CR_{min}}$, and $\mathrm{CR_{av}}$ are maximum, minimum, and average curvatures over the surface patch. $\mathrm{CR_t}$ refers to the curvature threshold. The formula in Equation (10) is a heuristic that was found to work well in most cases. The meaning of this heuristic is that if the magnitudes of all curvatures on a surface patch are almost the same, then the curvature threshold with an initial value, $\mathrm{CR}_{\mathrm{av}}$, is increased three times such that no division will occur on this surface patch.

The basic strategy in implementing the curvature partitioning is to let the breath-first search locate one subregion with low curvature and group the remaining elements into one or more subregions. The search is controlled to start from an element with all its nodal curvatures smaller than the nodal curvature threshold. In the search propagation, if normal curvature w.r.t. an element edge is smaller than the nodal curvature threshold, the propagation continues. Otherwise, it terminates at that element edge.

3.4.4. Algorithm. Now we are ready to introduce the algorithm by means of nodal curvatures.

Algorithm 1.4: surface partitioning by discontinuity of curvatures

```
(1) loop over each surface patch generated by Algorithms 1.1 and 1.2, which is not a flat
    region
    (1.1) calculate nodal curvatures
    (1.2) calculate an average nodal curvature
    (1.3) perform breath-first searches
        (1.3.1) initiate only from an element with a curvature that is less than the average
                curvature
        (1.3.2) propagate over the surface until a termination condition is satisfied: the
                curvature is greater than the curvature threshold.
        (1.3.3) If there are still some elements unprocessed, go back to (1.3.1) to initiate
                another breath-first search.
    (1.4) group the remaining elements, which do not belong to regions formed by breath-
          first searches, into one or more different surface patches by means of their con-
          nectivity.
```

4. DESIGN VARIABLES AND PERTURBATION PATTERNS

The shape optimization of structures can be written as a mathematical statement that requires the optimization of an objective function (design objective) subject to certain constraints on the types of designs that are admissible. Both the objective function and the constraint functions are represented by structural responses calculated in finite element analyses and are dependent upon a number of design variables as follows:

$$\text{minimize} \quad f(\mathbf{x}) \tag{11a}$$

$$\text{subject to} \quad \mathbf{g}_{\mathrm{e}}(\mathbf{x}) \leqslant 0 \tag{11b}$$

$$\mathbf{g}_{\mathrm{i}}(\mathbf{x}) \leqslant 0 \tag{11c}$$

where $\mathbf{x}$ is the vector of design variables. $f(\mathbf{x})$ is the objective function. $\mathbf{g}_e(\mathbf{x})$ and $\mathbf{g}_i(\mathbf{x})$ are explicit and implicit constraints, respectively. Even though the objective function can be defined in many ways, in this study we focus on one type of objective function:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{u}(\mathbf{x})^{\mathrm{T}}\mathbf{K}(\mathbf{x})\mathbf{u}(\mathbf{x}) = \frac{1}{2}\int_{\Omega} \boldsymbol{\varepsilon}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\sigma}(\mathbf{x})\,\mathrm{d}\Omega = \frac{1}{2}\mathbf{u}(\mathbf{x})^{\mathrm{T}}\mathbf{F}$$

where the right-hand side refers to the compliance, the strain energy of the structure, and can be considered as a reciprocal measure of the stiffness of the structure. $\mathbf{u}$ and $\mathbf{K}$ are displacement vector and stiffness matrix, respectively. $\boldsymbol{\varepsilon}$ and $\boldsymbol{\sigma}$ are strain and stress tensors, respectively. $\mathbf{F}$ refers to external loads that are considered as constants. Ω refers to the domain occupied by the structure.

The explicit constraints, $\mathbf{g}_e(\mathbf{x}) \leqslant 0$, can be expressed explicitly by the design variables as follows:

$$\mathbf{x}^{\mathrm{L}} \leqslant \mathbf{x} \leqslant \mathbf{x}^{\mathrm{u}} \tag{12}$$

where $\mathbf{x}^{\mathrm{L}}$ and $\mathbf{x}^{\mathrm{u}}$ are the lower bound and upper bound vectors of design variables, respectively. These bounds are inputs from users. On the other hand, the implicit constraints, $\mathbf{g}_i(\mathbf{x}) \leqslant 0$, cannot be expressed explicitly by the design variable. Typical implicit constraints include structural responses such as stress, displacement, and natural frequency. In this study, for the sake of simplicity no stress constraints are used in our numerical experiments, even though these constraints can be added into the shape optimization. Nevertheless, the following volume constraint is used:

$$\frac{V(\mathbf{x})}{V_0} = 1 \tag{13}$$

where $V(\mathbf{x})$ and V_0 are the volume of the structure at the current iteration and before the first iteration of the optimization process, respectively.

Commercial software such as Optistruct$^{\mathrm{TM}}$ and Genesis$^{\mathrm{TM}}$ can be used to solve a subset (see Equation (28) for details) of the optimization problems represented by Equations (11). However, with large-scale mesh models, if each mesh node is defined as a design variable, the optimization becomes computationally less feasible for numerical and economic reasons. A simple and effective way to circumvent the computational complexity is to reduce the space of original design variables to one of its subspaces such that the dimensionality of the optimization problem is decreased accordingly. Pickett *et al.* [11] first proposed this idea that evolved to the well-known basis vector approach as follows. Instead of using $\mathbf{x}$ that contains n design variables, we replace it by a linear combination of a smaller set of approximations:

$$\mathbf{x} = \mathbf{D}\mathbf{y} \tag{14}$$

where $\mathbf{y}$ is a vector that contains m design variables with $m < n$ or $m \ll n$. $\mathbf{D}$ is a $n \times m$ matrix that represents a smaller set of m approximations to the final design. Consequently, the optimization problem in Equations (11) is reduced to

$$\text{minimize} \quad f(\mathbf{y}) \tag{15a}$$

$$\text{subject to} \quad \mathbf{g}_e(\mathbf{y}) \leqslant 0 \tag{15b}$$

$$\mathbf{g}_i(\mathbf{y}) \leqslant 0 \tag{15c}$$

If $\mathbf{y}$ satisfies Equations (15b)–(15c), then the corresponding $\mathbf{x}$ satisfies Equations (11b)–(11c). However, the optimization of Equation (15a) is generally an approximation to that of Equation (11a).

One variation to Equation (14) is the perturbation vector approach:

$$\mathbf{x} = \mathbf{x}^0 + \sum_{i=1}^{m} \mathbf{P}^i y^i \tag{16}$$

where $\mathbf{x}^0$ refers to the initial geometric configure of the structure before the shape optimization. $\mathbf{P}^i$ and y^i are ith perturbation vector and ith design variable, respectively. m denotes the number of design variables in vector $\mathbf{y}$. Compared to the basis vector approach, the flexibility of the perturbation vector approach lies in the fact that the length of vector $\mathbf{P}^i$ can be the number of nodes on a small surface patch rather than n, the number of all nodes in the structure.

The art of the shape optimization lies in how to determine the matrix $\mathbf{D}$ in Equation (14) or $\mathbf{P}^i$ in Equation (15). Even though many studies related to the basis or perturbation vector methods [11–15] have been conducted in the past and some commercial software (Optistruct$^{\text{TM}}$ and Genesis$^{\text{TM}}$) are available in the market, creating $\mathbf{D}$ or $\mathbf{P}^i$ is, in general, still a time-consuming process with need for a considerable amount of human intervention. By following the spirit of the perturbation vector approach, we herein propose an efficient scheme to generate perturbation vectors and design variables as follows.

Algorithm 2.1: generation of perturbation vectors and design variables

(1) Perform a surface partitioning (Algorithm 1) of the input mesh model such that each resulting surface partition corresponds to a possible shape design variable
(2) Let users decide which surface patch will be involved in the shape optimization
(3) Generate perturbation vectors associated with all chosen surface patches and the corresponding design variables.

In the above algorithm, steps (1) and (3) can be fully automated, while step (2) needs a little amount of users' intervention. To carry out step (3) of Algorithm 2.1, a special perturbation method, *constant surface normal flow*, is proposed as follows. The perturbation of each partitioned surface patch is carried out by a constant magnitude of movement of all nodes along the direction of surface normal, as shown in Figure 2. This perturbation scheme is supported by the following lemma and proposition.

Lemma 4

Let $\mathbf{x} = \mathbf{x}(u, v)$ be a regular parameterized surface patch.§ If the constant surface normal flow is used to generate another surface patch

$$\mathbf{y}(u, v) = \mathbf{x}(u, v) + a\hat{N}(u, v) \tag{17}$$

§A regular surface refers to a surface $S \subset R^3$ that is locally diffeomorphic to R^2. In other words, for each point $P \in S$, there exists a neighbourhood V of P in S, an open set $U \subset R^3$, and a map $\mathbf{x} : U \to V$, which is a diffeomorphism [43].

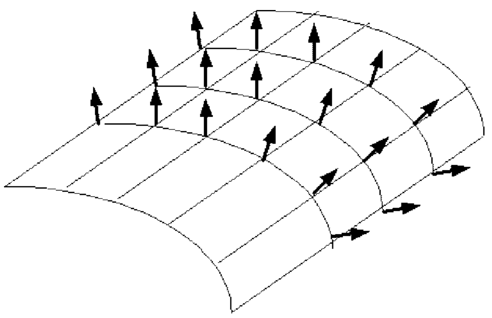

Figure 2. Constant surface normal flow.

where a is a constant that denotes the length of perturbation vectors. $\hat{N}(u,v)$ is a unit surface normal. Then at regular points, the Gaussian and mean curvatures of $\mathbf{y}$ are respectively

$$K^{(y)}(u,v) = \frac{K^{(x)}(u,v)}{1 - 2H^{(x)}(u,v)a + K^{(x)}a^2}$$
$$H^{(y)}(u,v) = \frac{H^{(x)}(u,v) - K^{(x)}(u,v)a}{1 - 2H^{(x)}(u,v)a + K^{(x)}a^2} \quad (18)$$

where $K^{(x)}(u,v)$ and $H^{(x)}(u,v)$ are Gaussian and mean curvatures of $\mathbf{x}$. $K^{(y)}(u,v)$ and $H^{(y)}(u,v)$ are Gaussian and mean curvatures of $\mathbf{y}$.

Proof
By differentiating $\mathbf{y}(u,v)$ in Equation (17) with respect to u and v, we get

$$\mathbf{y}_u = \mathbf{x}_u + aN_u, \quad \mathbf{y}_v = \mathbf{x}_v + aN_v \quad (19)$$

Then the cross product of the above two vectors is

$$\mathbf{y}_u \otimes \mathbf{y}_v = \mathbf{x}_u \otimes \mathbf{x}_v + a\mathbf{x}_u \otimes N_v + aN_u \otimes \mathbf{y}_v + a^2 N_u \otimes N_v \quad (20)$$

where $\otimes$ refers to a vector cross product. Since

$$S(\mathbf{x}_u) \otimes \mathbf{x}_v + \mathbf{x}_u \otimes S(\mathbf{x}_v) = 2H(u,v)(\mathbf{x}_u \otimes \mathbf{x}_v)$$

and

$$S(\mathbf{x}_u) \otimes S(\mathbf{x}_v) = K(u,v)(\mathbf{x}_u \otimes \mathbf{x}_v) \quad (21)$$

where S is shape operator or Weingarten map [44] such that

$$S(\mathbf{x}_u) = -N_u, \quad S(\mathbf{x}_v) = -N_v \quad (22)$$

hence

$$\mathbf{y}_u \otimes \mathbf{y}_v = (1 - 2H^{(x)}(u,v)a + K^{(x)}(u,v)a^2)(\mathbf{x}_u \otimes \mathbf{x}_v) \quad (23)$$

Now by using Equations (19) and (21)–(23), we can write the curvatures of $\mathbf{y}$ as follows:

$$\begin{aligned} K^{(y)}(u,v) &= \frac{N_u \otimes N_v}{\mathbf{y}_u \otimes \mathbf{y}_v} = \frac{K^{(x)}(u,v)}{1 - 2H^{(x)}(u,v)a + K^{(x)}(u,v)a^2} \\ H^{(y)}(u,v) &= \frac{-N_u \otimes y_v - y_u \otimes N_v}{2(\mathbf{y}_u \otimes \mathbf{y}_v)} \\ &= \frac{-N_u \otimes \mathbf{x}_v - aN_u \otimes N_v - \mathbf{x}_u \otimes N_v - aN_u \otimes N_v}{2(\mathbf{y}_u \otimes \mathbf{y}_v)} \\ &= \frac{H^{(x)}(u,v) - K^{(x)}(u,v)a}{1 - 2H^{(x)}(u,v)a + K^{(x)}(u,v)a^2} \end{aligned} \tag{24}$$

□

On the basis of Lemma 4, the following proposition is ready to be derived.

Proposition 5
If the curvatures of a surface patch are continuous, i.e. the surface patch is smooth except on its boundary, then the perturbation scheme of *constant surface normal flow* does not destroy the smoothness of the surface patch after the shape optimization.

Proof
Since the curvatures of the surface patch represented by $\mathbf{x}(u,v)$ in Equation (17) is initially continuous, at each regular point (u_0, v_0) of $\mathbf{x}$ we have

$$\lim_{\substack{u\to u_0\\ v\to v_0}} K^{(x)}(u,v) = K^{(x)}(u_0,v_0), \quad \lim_{\substack{u\to u_0\\ v\to v_0}} H^{(x)}(u,v) = H^{(x)}(u_0,v_0) \tag{25}$$

By Equations (24) and (25), we can write

$$\begin{aligned} \lim_{\substack{u\to u_0\\ v\to v_0}} K^{(y)}(u,v) &= \frac{\lim_{\substack{u\to u_0\\ v\to v_0}} K^{(x)}(u,v)}{1 - 2a \lim_{\substack{u\to u_0\\ v\to v_0}} H^{(x)}(u,v) + a^2 \lim_{\substack{u\to u_0\\ v\to v_0}} K^{(x)}(u,v)} = K^{(y)}(u_0,v_0) \\ \lim_{\substack{u\to u_0\\ v\to v_0}} H^{(y)}(u,v) &= \frac{\lim_{\substack{u\to u_0\\ v\to v_0}} H^{(x)}(u,v) - a \lim_{\substack{u\to u_0\\ v\to v_0}} K^{(x)}(u,v)}{1 - 2a \lim_{\substack{u\to u_0\\ v\to v_0}} H^{(x)}(u,v) + a^2 \lim_{\substack{u\to u_0\\ v\to v_0}} K^{(x)}(u,v)} = H^{(y)}(u_0,v_0) \end{aligned} \tag{26}$$

Thus, the curvatures of surface patch $\mathbf{y}$ are continuous as long as the denominators in Equations (26) are not zero, and the smoothness of the surface patch after the shape optimization is not changed. □

The conclusion in Proposition 5 means that the proposed *constant surface normal flow* scheme does not destroy the smoothness of an original surface patch, unlike the approach in [23]. In contrast to the B-spline approach [9], our approach avoids the troubles of converting polygonal meshes into B-spline patches in the cases without CAD information, and has no need for trim curves of NURBS patches or awkward constraint management between B-spline patches. In addition, our approach is well suited for handling arbitrary topology, compared to the limitation of the B-spline approaches.

Even though the proposed *constant surface normal flow* mathematically maintains the smoothness at interior points of each surface patch, heuristic treatment is needed to handle boundary points where the discontinuity of surface normal or curvature occurs. We categorize the boundary between adjacent surface patches into the following two groups.

(1) Boundary between non-design and design surface patches
Note that only design surface patches are allowed to be perturbed as design variables. Assume that **P** is a boundary point and shared by a non-design surface patch nd_1 and a design surface patch d_1, as shown in Figure 3. If there is a G^1 discontinuity between patches nd_1 and d_1 (see Algorithm 1.1), the perturbation at point **P** due to d_1 is the overall perturbation at **P**, as illustrated in Figure 3(a). Otherwise, the perturbation at point **P** due to d_1 is reduced by half to make a smooth transition, as in Figure 3(b).

(2) Boundary between design surface patches
Let **P** be a boundary point shared by two design surface patches d_1 and d_2, as shown in Figure 4. If there is a G^1 discontinuity between d_1 and d_2 as in Figure 4(a), vector **v** should be generated to maintain the sharp corner at point **P**. Given perturbation vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ in Figure 4(a), angle γ_1 in ΔPAC of Figure 5 can be determined by

$$\gamma_1 = \arcsin\left(\frac{|\mathbf{v}_2|\sin\alpha}{\sqrt{|\mathbf{v}_1|^2+|\mathbf{v}_2|^2-2|\mathbf{v}_1||\mathbf{v}_2|\cos\alpha}}\right) \tag{27}$$

where α is the angle between vectors $\mathbf{v}_1$ and $\mathbf{v}_2$. In ΔABC, length BC can be expressed as

$$c = \frac{\sin\gamma_2\sqrt{|\mathbf{v}_1|^2+|\mathbf{v}_2|^2-2|\mathbf{v}_1||\mathbf{v}_2|\cos\alpha}}{\sin\alpha} \tag{28}$$

where γ_2 is complementary to γ_1, i.e., $\gamma_2 = 90^\circ - \gamma_1$. Let $\mathbf{v}_3 = (\mathbf{v}_2 \otimes \mathbf{v}_1) \otimes \mathbf{v}_2$ and $\hat{\mathbf{v}}_3$ be a unit vector in the direction of $\mathbf{v}_3$. Then, the vector **v** in Figures 4 and 5 can be determined by

$$\mathbf{v} = \mathbf{v}_2 + c\hat{\mathbf{v}}_3 \tag{29}$$

After **v** is known, the following scaling factor can be computed:

$$s = \frac{|\mathbf{v}|}{|\mathbf{v}_1+\mathbf{v}_2|} \tag{30}$$

Perturbation vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ at point **P** can then be changed to $s\mathbf{v}_1$ and $s\mathbf{v}_2$, respectively, such that the sharp corner is approximately preserved. Note that if $\alpha = 90^\circ$, $s = 1.0$. The handling of three or more design patches that form a G^1 discontinuity between each other at point **P** is much more complex, as shown in Figure 6. In this paper, we do not provide special treatment to this kind of cases. It will be a future research topic.

If there is no G^1 discontinuity between d_1 and d_2 as in Figure 4(b), we simply change perturbation vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ at point **P** to $0.5\mathbf{v}_1$ and $0.5\mathbf{v}_2$, respectively. An additional way to handle the boundary between design surface patches is to use a feature-recognition technique to detect

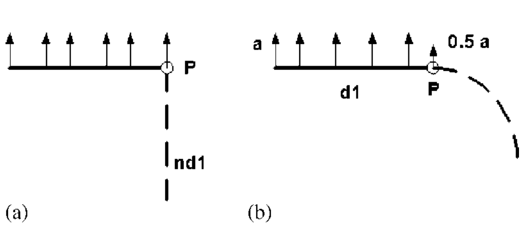

Figure 3. Perturbation at boundary point **P** between a design surface patch d_1 and a non-design surface patch nd_1: (a) G^1 discontinuity at point **P**; and (b) no G^1 discontinuity at point **P**.

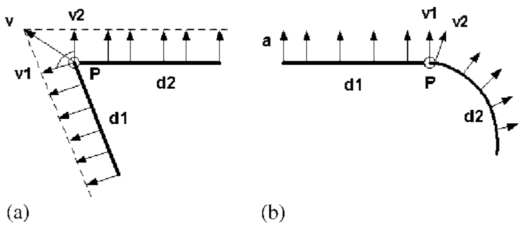

Figure 4. Perturbation at boundary point **P** between two design surface patches d_1 and d_2: (a) G^1 discontinuity at point **P**; and (b) no G^1 discontinuity at point **P**.

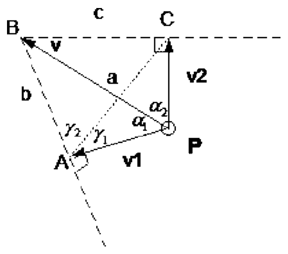

Figure 5. Calculation of vector **v** in Figure 4(a).

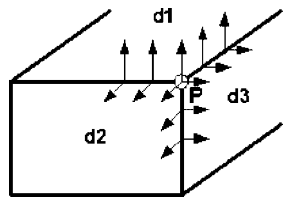

Figure 6. Perturbation at boundary point **P** shared by three design surface patches.

design features such as holes, fillets, chamfers, pockets, slots and beams on the basis of partitioned surface patches. Each design feature can be then perturbed as a meaningful design unit. This part of work will be a research topic in the near future.

In order to avoid a possible mesh distortion due to the perturbation of surface nodes on each patch, we proportionally perturb the domain nodes close to these surface nodes. Each surface node 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 surface node. The overall perturbation of a domain node is the accumulative contribution from all nearby surface nodes divided by the number of these surface nodes.

5. SHAPE OPTIMIZATION OF FINITE ELEMENT MESHES OF COMPLEX STRUCTURES

By combining Equations (11)–(13), (15) and (16), our shape optimization can be expressed by

$$\text{Minimize} \quad f(\mathbf{y}) = \frac{1}{2}\mathbf{u}(\mathbf{y})^{\mathrm{T}}\mathbf{K}(\mathbf{y})\mathbf{u}(\mathbf{y}) = \frac{1}{2}\int_{\Omega} \boldsymbol{\varepsilon}(\mathbf{y})^{\mathrm{T}}\boldsymbol{\sigma}(\mathbf{y})\,\mathrm{d}\Omega = \frac{1}{2}\mathbf{u}(\mathbf{y})^{\mathrm{T}}\mathbf{F} \tag{31a}$$

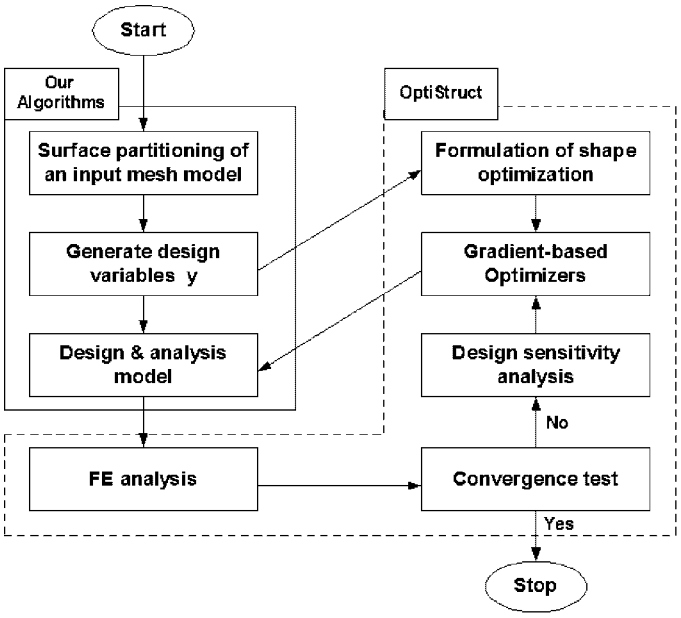
Start
Our Algorithms
OptiStruct
Surface partitioning of an input mesh model
Generate design variables y
Design & analysis model
Formulation of shape optimization
Gradient-based Optimizers
Design sensitivity analysis
No
FE analysis
Convergence test
Yes
Stop

Figure 7. Overall process of a shape optimization.

$$\text{subject to} \quad \mathbf{g}(\mathbf{y}) - \mathbf{g}(\mathbf{y})^{\mathrm{u}} \leqslant 0 \tag{31b}$$

$$\mathbf{y}^{\mathrm{L}} \leqslant \mathbf{y} \leqslant \mathbf{y}^{\mathrm{u}} \tag{31c}$$

where the design variables in $\mathbf{y}$ are defined in Equation (16). Functions $\mathbf{g}(\mathbf{y})$ are structural responses obtained from a finite element analysis, and include the volume of the entire structure, nodal displacements, etc.

In this study, the shape optimization was carried out by using the optimizers available in Optistruct$^{\mathrm{TM}}$, a dual method and a primal feasible directions method that are both based on the convex linearization of the design space. The overall process of the shape optimization is illustrated in Figure 7. Since we use only polygonal meshes, a design model is the same as the corresponding analysis model for a finite element analysis.

6. NUMERICAL EXPERIMENTS AND DISCUSSIONS

The algorithms introduced in this paper were implemented in VC++ and tested on a Pentium III HP PC. Table II shows experimental results of surface partitioning on different mesh models. Since the major part of the surface partitioning algorithm is three passes of breath-first searches over all elements of M, its time complexity is $O(n)$, where $n = \max(N_{\mathrm{e}}, N_{\mathrm{v}})$, N_{e} and N_{v} are the numbers of elements and vertices in M, respectively. However, element neighbour relationship needs to be set up as a precomputation, which takes $O(n \log n)$ time. Thus, overall time cost of the proposed surface partitioning is $O(n \log n)$.

Table II. Surface partition of different test mesh models.

Model name	Vertex	Element	Time (s)	Rate (Element/s)	Partition	Reduction ratio (Node/partition)
Bumper	473	432	0.17	2541	8	59.1
Bracket	236	186	0.07	2657	11	21.5
Deck lid	8807	8624	3.0	2871	9	978.6
Curver1	143	120	0.04	3000	3	47.7
Block	56	46	0.01	4600	5	11.2
Base	12640	25328	6.13	4132	43	294.0
Ellipsoid	156	308	0.06	5133	1	156
Torus	1521	3042	0.67	4533	1	1521
Hyperbolic Paraboloid	441	400	0.14	2857	1	441
Cone	902	1800	0.39	4603	3	300.7
Engine bracket	12465	46373	5.81	3098	173	72.1
Control arm	2643	1730	0.61	2854	38	69.6

The last column of Table II demonstrates the ratio of total number of nodes to total number of partitioned surface patches. It reflects the reduction of design variables from the conventional design variable scheme of using each mesh node to the new scheme of using each partitioned patch. Depending upon the geometric complexity and mesh density, the design variables were reduced by 10–1000 times. Even though using B-spline patches may achieve a similar amount of reduction, there is a need for trim curves of NURBS patches or awkward constraint management between B-spline patches.

Two practical examples are used to demonstrate the effectiveness of the proposed approach in the direct shape optimization of complex structures represented by polygonal meshes.

Example 1
Control arm.

Specification: The control arm is a typical structure encountered in the structure design of automobiles, as shown in Figure 8(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.

Results: Figure 8(b) illustrates the surface partition results after removing all non-design elements. Among all 38 partitioned patches, users have an option to choose certain number of patches as design variables for a 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, 11 design variables were chosen, each of which corresponded to a partitioned surface patch. The visible design surface patches are marked in Figure 8(b). Since there is a V-shaped discontinuity at the top surface of beam 3 in Figure 8(b), two surface patches are generated to cover that surface. The sets of perturbation vectors generated automatically by the proposed approach are shown in Figure 8(c). The weighted compliance of the entire structure is reduced by 12.95%, as shown in Figure 8(d). The comparison between the original and optimized geometric configurations is illustrated in Figure 9.

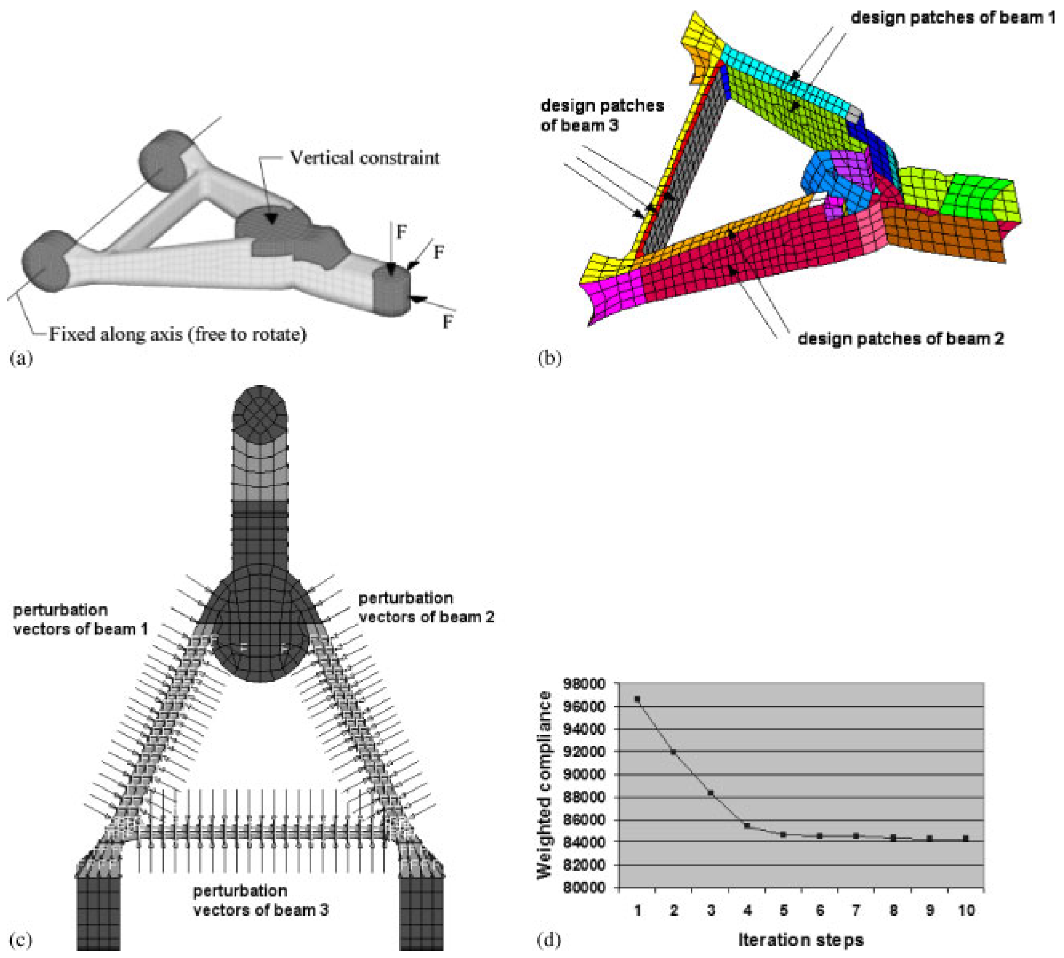

Figure 8. A control arm defined by a finite element model: (a) original model; (b) partitioned patches; (c) perturbation vectors; and (d) optimization history.

Example 2
Engine mount bracket.

Specification: An aluminium engine bracket of an automobile is used here. The finite element model in the original problem specification is shown in Figure 10(a) in which the dark grey colour refers to design domain consisting of 9046 elements. Six load cases are considered to reflect different driving and service conditions: (1) start; (2) backup; (3) into a pothole; (4) out of a pothole; (5) loads from an attached part and (6) loads during engine transport [45].

Results: Among 173 partitioned surface patches, we chose 19 patches as design variables. The chosen surface patches after removing non-design elements are shown in Figure 10(b). The sets of perturbation vectors that are automatically generated by the new approach are

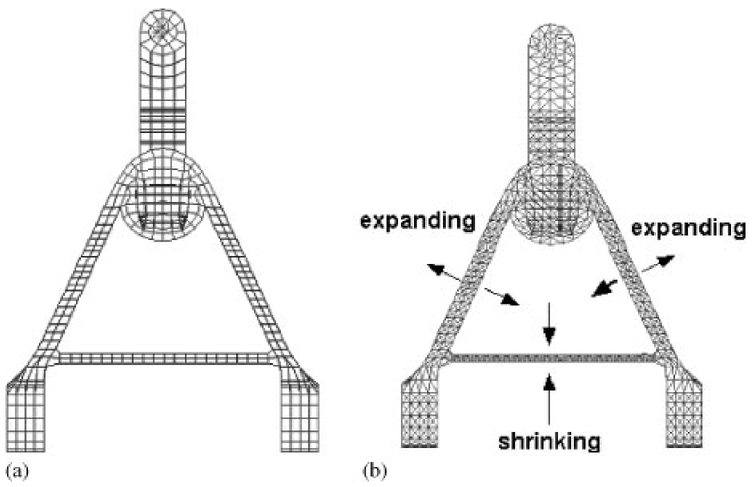

Figure 9. Original versus optimized geometric shape of the control arm defined in Figure 8: (a) original; and (b) optimized.

Table III. Comparison of sizes of beam components before and after the shape optimization on the engine bracket model in Figure 10.

Beam set no.	Width before opt.	Width after opt.		Height before opt.	Height after opt.	
		Isotropic	Anisotropic		Isotropic	Anisotropic
1	9.041	5.39	7.26	13.101	8.37	8.60
2	11.023	6.59	7.03	11.051	7.05	7.25
3	6.337	9.15	6.80	12.226	15.10	15.39
4	5.004	7.41	1.36	2.000	4.21	4.26
5	5.017	6.67	4.53	9.701	11.36	11.70
6	10.091	12.52	10.10	41.618	42.99	40.60

illustrated by Figure 10(c). Since there were four patches related to each beam in Figure 10(c), we devised two slightly different ways, isotropic and anisotropic perturbations, to create design variables. In the isotropic perturbation, four patches related to each beam are considered as a single shape design variable such that the beam is allowed to expand and shrink isotropically. In contrast, with the anisotropic perturbation, four patches related to each beam are allowed to change independently and considered as four different shape design variables. The comparison between the optimized shape and original shape is listed in Table III. The weighted compliance of the entire structure is reduced by 9.1% (anisotropic) and 5% (isotropic), respectively, as shown in Figure 10(d).

7. CONCLUSIONS

In this paper, we present a new algorithm to conduct shape optimizations directly on polygonal meshes. Surface partitioning is used to dramatically reduce the total number of shape design

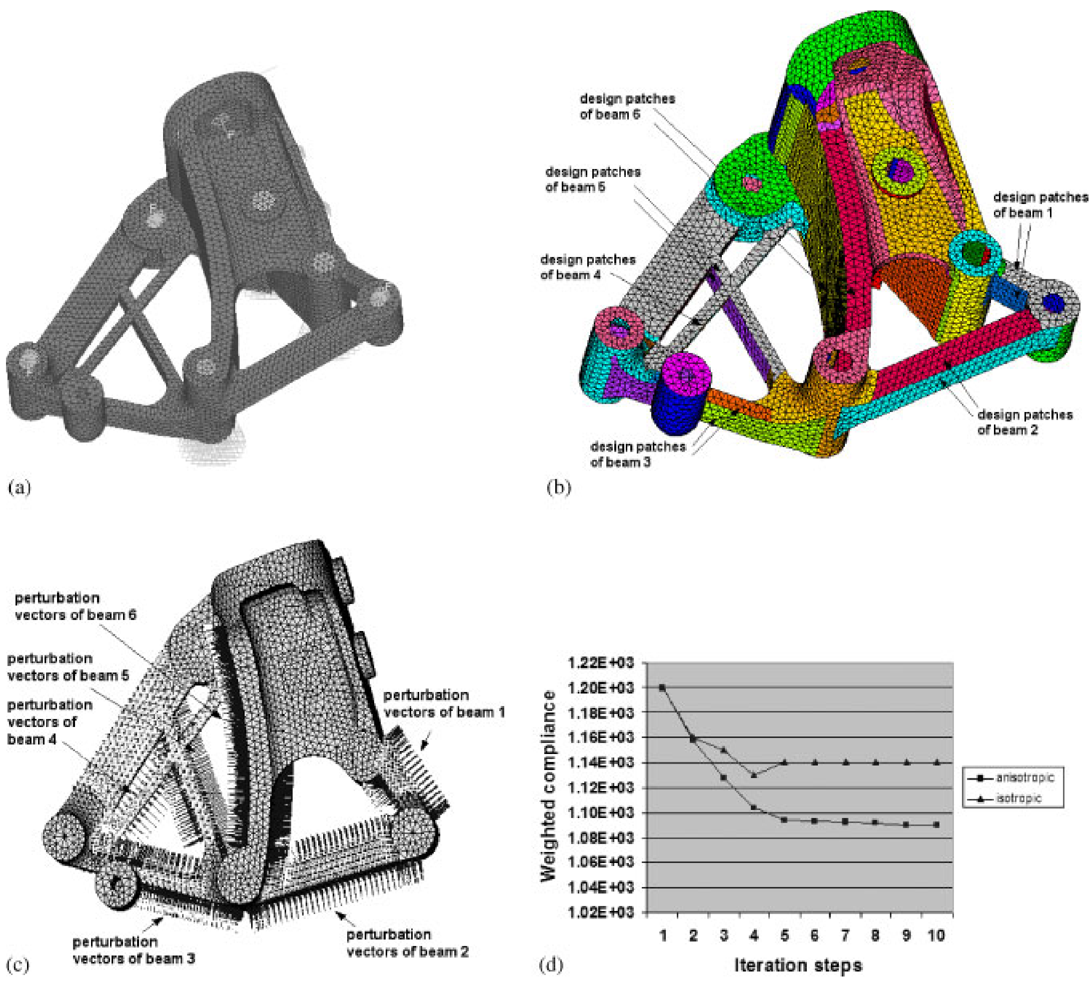

Figure 10. An engine block defined by a finite element model: (a) original model; (b) partitioned patches; (c) perturbation vectors; and (d) optimization history.

variables by about 10 through 1000 times. A perturbation scheme of *constant surface normal flow* is proposed to maintain the smoothness of original surfaces. Each partitioned surface patch can be assigned as a shape design variable, depending upon users' preference. Numerical experiments indicate that the proposed approach simplifies the conventional shape optimization processes and works well with large-scale polygonal mesh models.

REFERENCES

1. Ding Y. Shape optimization of structures: a literature survey. *Computers and Structures* 1986; **24**:985–1004.
2. Vigdergauz S. Two-dimensional grained composites of minimum stress concentration. *Journal of Structural Mechanics* 1997; **34**:661–672.
3. Xie YM, Steven GP. *Evolutionary Structural Optimization*. Springer: Berlin, Heidelberg, New York, 1997.

4. Fleury C. An efficient dual optimizer based on convex approximation concepts. *Structural Optimization* 1989; **1**:81–89.
5. Vanderplaats GN. CONMIN-A Fortran Program for Constrained Function Minimization. *NASA TM X-62282* 1973.
6. Woon SY, Querin OM, Steven GP. Structural application of a shape optimization method based on a genetic algorithm. *Structural and Multidisciplinary Optimization* 2001; **22**:57–64.
7. Fourie PC, Groenwold AA. The particle swarm optimization algorithm in size and shape optimization. *Structural and Multidisciplinary Optimization* 2002; **23**:259–267.
8. Schleupen A, Maute K, Ramm E. Adaptive FE-procedures in shape optimization. *Structural and Multidisciplinary Optimization* 2000; **19**:282–302.
9. 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.
10. 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.
11. Pickett RM, Rubinstein MF, Nelson RB. Automated structural synthesis using a reduced number of design coordinates. *AIAA Journal* 1973; **11**(4):489–494.
12. Dybbro JD, Holm NC. On minimization of stress concentration for three-dimensional models. *Computers and Structures* 1986; **24**:637–643.
13. Tvergaard V. On the optimum shape of a fillet in a flat bar with the restrictions. In *Optimization in Structural Design*, Sawczuk A, Mroz Z (eds). Springer: New York, 1975; 181–195.
14. Pedersen P, Laursen CL. Design for minimum stress concentration by finite elements and linear programming. *Journal of Structural Mechanics* 1982; **10**:375–391.
15. Yang RJ, McGeen DT. Application of basis function concept to practical shape optimization problems. *Structural Optimization* 1992; **5**:55–63.
16. Botkin ME. Shape optimization of plate and shell structures. *AIAA Journal* 1982; **20**:268–273.
17. Yang RJ, Botkin ME. A modular approach for three dimensional shape optimization. *AIAA Journal* 1987; **25**:492–497.
18. Botkin ME. Structural optimization of automotive body components based upon parametric solid modeling. *AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, U.S.A. 2000 (Paper no. 2000-4707).
19. Schumacher A, Hierold R. Parametrized CAD-models for multidisciplinary optimization process. *AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, U.S.A. 2000 (Paper no. 2000-4912).
20. Tortorelli DA. A geometric representation scheme suitable for shape optimization. *Mechanics of Structures and Machines* 1993; **21**(1):95–121.
21. Belegundu AD, Rajan SD. A shape optimization approach based on natural design variables and shape function. *Computer Methods in Applied Mechanics and Engineering* 1988; **66**:87–106.
22. Rajan SD, Belegundu AD. Shape optimal design using fictitious loads. *AIAA Journal* 1989; **27**:102–107.
23. Zienkiewicz OC, Campbell JS. Shape optimization and sequential linear programming. *Optimal Structural Design.* Wiley: New York, 1973; 109–126.
24. Braibant V, Fleury C. Shape optimal design using B-spline. *Computer Methods in Applied Mechanics and Engineering* 1984; **44**:247–267.
25. Bajaj C, Dey TK. Convex decomposition of polyhedra and robustness. *SIAM Journal on Computing* 1992; **21**:339–364.
26. Chazelle B. Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm. *SIAM Journal on Computing* 1984; **13**:488–507.
27. Chazelle B, Palios L. Triangulating a non-convex polytope. *Discrete and Computational Geometry* 1990; **5**: 505–526.
28. 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.
29. Chazelle B, Palios L. Decomposing the boundary of a nonconvex polyhedron. *Algorithmica* 1997; **17**:245–265.
30. Rappoport A. The extended convex differences tree (ECDT) representation for N-dimensional polyhedra. *International Journal of Computational Geometry and Applications* 1991; **1**(3):227–241.
31. Ruppert J, Seidel R. On the difficulty of triangulating three-dimensional non-convex polyhedra. *Discrete and Computational Geometry* 1992; **7**:227–253.
32. Kiwi M, Spielman D, Teng SH. Min-max-boundary domain decomposition. *Theoretical Computer Science* 2001; **261**(2):253–266.
33. Teng SH. Points, spheres, and separators, a unified geometric approach to graph partitioning. *Ph.D. Dissertation*, School of Computer Science, Carnegie Mellon University, 1991.
34. Barnard ST, Simon HD. Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency*: *Practice and Experience* 1994; **6**:101–117.

35. Simon HD. Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering* 1991; **2**(2/3):135–148.
36. Pothen A, Simon HD, Lion KP. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Mathematical Analysis* 1990; **11**:430–452.
37. 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.
38. Williams RD. Performance of dynamic load balancing algorithms for unstructured mesh calculations. *Concurrency: Practice and Experience* 1991; **3**:457–481.
39. 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.
40. Vaughan C. Structural analysis on massively parallel computers. *Computing Systems in Engineering* 1991; **2**(2/3):261–267.
41. Hoppe H, De Rose T, Duchamp T, MaDonald J, Stuetzle W. Mesh optimization. *ACM SIGGRAPH Computer Graphics Proceedings* 1993; **27**:19–26.
42. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press: Cambridge, 1992.
43. Do Carmo MP. *Differential Geometry of Curves and Surfaces*. Prentice-Hall: Englewood Cliffs, NJ, 1976.
44. Gray A. *Modern Differential Geometry of Curves and Surfaces with Mathematica*. CRC Press: Boca Raton, 1997.
45. Meyer-Pruessner R. Prozesskette topologieoptimierung. *Beitrage zum NAFEMS Seminar zur Topologieoptimierung*, Aalen, Germany, 1997; 12.1–12.5 (in German).