Conversion of spatial-enumeration scheme into constructive solid geometry

Mehran Chirehdast and Panos Y Papalambros

An algorithm is presented that converts a spatial enumeration scheme into a CSG representation of an object. The scheme represents solid objects in terms of a discrete set of binary elements of the same (or comparable) size. An automatic conversion of such a representation into CSG trees would help the designer in reasoning with the represented artifact. The algorithm has been developed to aid the interpretation of structural design topologies generated by homogenization. The work presented poses a challenging problem that does not have a unique solution, proposes a solution that solves the problem under a set of assumptions, and acts as a motivation for future work.

Keywords: solid modelling, solid representation conversion, constructive solid geometry, spatial enumeration schemes, computer vision

Different ways of representing solids have gained popularity in the CAE community for a variety of applications. For example, a finite-element model for computational mechanics requires a representation scheme that is different from that of a CAM model that is used for manufacturing purposes, even though they both represent the same object (a brief classification of solid representations is provided further below). Automatic conversions of each of these representation schemes into others have presented a continuing challenge. In this paper, the conversion of a spatial-enumeration scheme into constructive solid geometry (CSG) is discussed.

Such a conversion is an element of an integrated structural optimization system (ISOS)[1] which is used to create complete structural designs, from topology generation to detailed dimensional sizing. The first phase of this system (topology optimization) generates a specific distribution of a given amount of material that represents the stiffest possible structure within a prescribed design domain. The program uses a homogenization method[2] that has gained popularity in the structural optimization community since the late 1980s. This optimal distribution of material resembles an initial design concept, and it requires further refinements. To automate the requisite manipulations in ISOS, computer-vision techniques have been used to convert the output of the first phase into other representations. The prime requirement for these representations is that they be at a higher level of abstraction, such that the designer, a human or an automaton, can reason with them more easily.

2D topologies in ISOS are discussed elsewhere[1,3]. This paper emphasizes some 3D activities in ISOS.

The initial design is generated in the form of a finite-element model referred to as a spatial-enumeration scheme. To manipulate the design effectively, its representation must be changed, a need that is illustrated further by several examples given in this paper.

The introductory section of the paper provides a short review of work in 3D computer vision and solid representations. This is followed by a discussion on the objective of the research presented in the paper.

3D computer vision

The primary emphasis in 3D computer vision research has been on reconstructing the 3D world from one or more 2D images. Some well known problems posed and partially solved are stereo vision, motion in images, and shape from shading. Davies[4] gives an introduction to such 3D problems.

Design Laboratory, Department of Mechanical Engineering and Applied Mechanics, University of Michigan, Ann Arbor, MI 48109, USA

Paper received: 31 May 1993. Revised: 15 October 1993

Some research has been conducted on interpretation and information extraction for 3D images, i.e. intensity arrays. Most of this work is concentrated on interpreting computer-tomography images. Intensive research is currently under way to solve problems in this field; for a review, see Reference 5 by Udupa and Herman. The basic goal of these research efforts is first to segment the 3D images given in the form of *slices*, i.e. 2D sections of the images. Next, some representation of the boundary data of the regions or their approximation is extracted. In the structural topology problem, the segmentation of the images generated is not a critical issue (see Reference 3 by Papalambros and Chirehdast for further information on image segmentation for such problems). In the investigation of the second issue, i.e. boundary extraction and representation, a brief introduction to solid representations from a CAD point of view is useful.

CAD schemes for solid representations

There are six basic schemes for representing solid objects for CAD purposes[6,7], as follows:

- *Boundary representation (B-rep):* The object is represented by its boundaries, which consist of a set of elements: vertices, edges and faces. Explicit information is needed about how these elements are connected.
- *Sweep methods:* The object is represented as a volume that is generated by sweeping a planar shape along a curve.
- *Primitive instancing:* A parametric description of all possible objects is available. Objects are represented by varying the scale and dimensions of these generic descriptions.
- *Constructive solid geometry:* A set of primitives, e.g. cube, sphere, cylinder, is available. The solid is stored as a binary tree with primitives at the leaf nodes and regularized Boolean operators at intermediate nodes.
- *Spatial enumeration:* This is a way of representing the solid by binary volume elements of uniform size referred to as voxels.
- *Cell decomposition:* The object is represented as the union of a set of cells of different sizes. For example, region octree is a regular version of this representation.

The most commonly used solid representations in CAD are CSG and B-rep[8]. The representation resulting from topology optimization (and almost every finite-element analysis) is spatial enumeration[6]. Thus, the goal of the 3D activities in the interpretation phase of ISOS becomes the conversion of the spatial-enumeration scheme into one of the commonly used representations. Each of the schemes mentioned (CSG and B-rep) has its own advantages and disadvantages (see References 6–8 for comparisons).

Briefly, CSG is an unambiguous, always valid, conceptually easy to comprehend, representation of solids. The internal representation (i.e. data structure) is simple, but not unique, i.e. more than one CSG tree can represent the same object, and certain geometric manipulations (queries) are not easily performed on CSG representations. A boundary representation is more easily manipulated. However, ensuring topological validity for a B-rep model is not trivial. A boundary representation is less transparent to the human designer than the CSG one. Conversion from CSG to B-rep is unique (under a set of assumptions), and algorithms exist to perform this task automatically.

The problem of the automated conversion of B-rep to a so-called minimal CSG has been treated by Vossler and Shapiro[9,10]. The application of this technique to the present problem, i.e. the conversion of spatial enumeration into CSG representation, requires the conversion of the spatial-enumeration scheme into an approximate B-rep of the object. Even though this route is promising, there are limitations on the approximate B-rep that can be handled by the B-rep-to-CSG algorithm discussed in these references. Specifically, the B-rep must be in the form of analytical half spaces. The conversion of a spatial-enumeration scheme into approximate B-rep using half spaces is a research problem in its own right. It is beyond the scope of this paper, and requires further investigation.

Objective

The main objective of the research described herein is to convert a spatial-enumeration scheme into a CSG representation. The algorithm that is presented has two essential constituents, namely a matching unit and a segmentation unit. The matching unit matches the regions against available primitives in the database or *library of primitives*. The segmentation unit segments the region in case no match is found for that region.

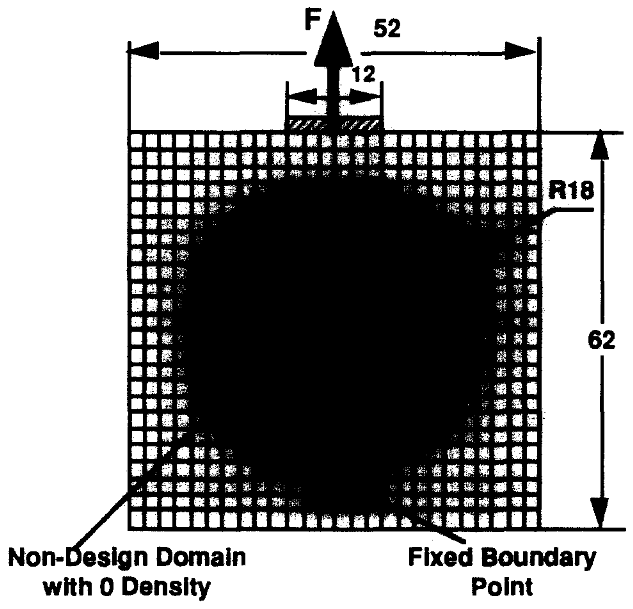
The remainder of this paper is organized as follows. First, an overview of the algorithm for 2D images is given. Template matching is described for 2D primitives, and this is followed by 2D examples. 3D extensions of the conversion algorithm are discussed next. Some 3D examples are provided, and these are followed by concluding remarks.

2D CONVERSION ALGORITHM

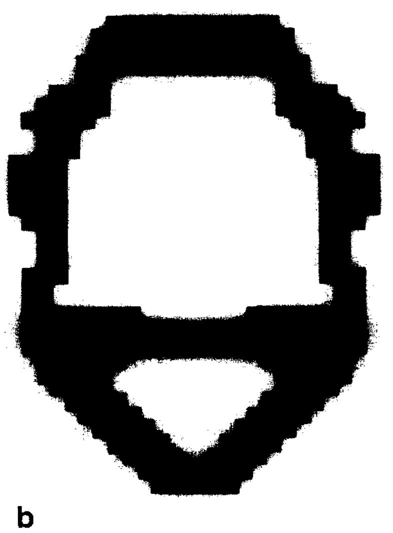
The concepts of CSG representation are primarily used for 3D purposes. The simplification of these concepts to the 2D representation of areas is possible. The 2D conversion algorithm discussed here takes a binary image as input and generates a CSG tree that approximates the structure. The following example gives an overview of what the conversion algorithm accomplishes.

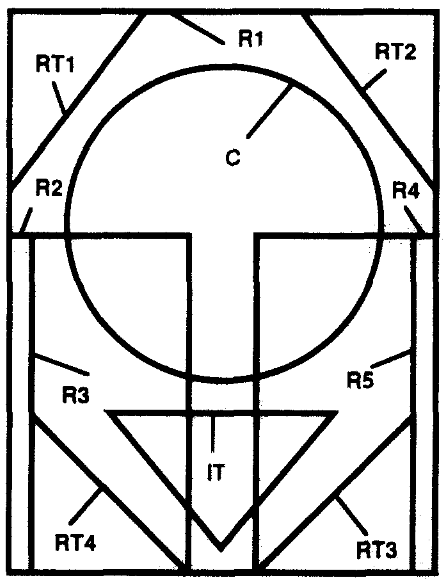
Example 1: Eye bolt

The starting point for ISOS is the model shown in *Figure 1a*. This example is the optimal-topology design of an eye bolt[1]. The output of the topology optimization is shown in *Figure 1b*, where only 32% of the initial domain is allowed to be used to construct the new image (this volume constraint corresponds to a solid-to-void ratio of 10:21). *Figure 1c* shows an approximation of the same image by 2D primitives, where the Rs are rectangles, the RTs are right-angled triangles, IT is the isosceles triangle, and C is the circle. All the extension lines point to a unique edge of each primitive. However, from *Figure 1c*, it is not clear what exactly *R*2, *R*3, *R*4 and *R*5 represent; *R*3 is inside *R*2, and, similarly, *R*5 is inside *R*4. *Figure 1d* shows the CSG tree that corresponds to this object. Interior nodes are denoted by circled letters representing regularized Boolean operators, where U and M denote regularized union and difference, respectively (conventionally, an asterisk is used as a superscript to denote regularized Boolean operators – throughout this paper, superscripts are dropped). Union and intersection are commutative operators, and the order of the leaves or nodes connected to these operations is irrelevant. The difference operator, however, is not commutative. The following convention, commonly used in the literature, is adopted here: the leaf or node to the right of a difference operator is *carved out* of the leaf or node to the left of the operator. The only missing information in the tree are specific dimensions and locations of primitives. This

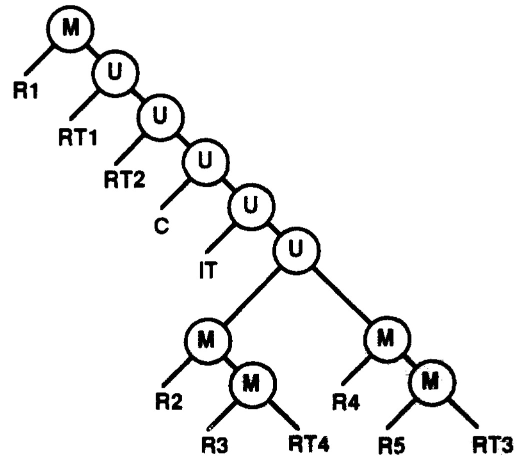


a


b



c



d

Figure 1 Example 1; (a) initial design model for ISOS, (b) image generated by topology optimization as optimum material distribution[1] (solid-to-void ratio is 10:21, or volume constraint is 32%), (c) primitives representing the regions of the object shown in *Figure 1b*, (d) CSG tree of object

information is readily available to the automaton, as it detects the primitives.

The automation of this conversion is the objective of the suggested algorithm. The algorithm is first described informally. Every image generated by topology optimization is basically the difference of an initial domain (which without loss of generality can be a rectangle) and the union of a set of holes. This fact is the premise of the algorithm. If any of the holes is matched with any of the primitives in the library of primitives, then that primitive represents the region occupied by the hole (Definition 4 below is a definition of a match). Otherwise, the hole itself becomes an initial design domain, and the matching continues recursively. The convergence of the algorithm is due to the discrete nature of the images: at some (hypothetical) point, the algorithm arrives at a pixel which is matched with a primitive available in the library of primitives, say a rectangle. The only complication arising in each iteration is the need to reckon what corresponds to a hole and what is an object, and how the regularized Boolean operators are affected by this distinction.

The following definitions are valid for binary images, and they follow closely those discussed by Rosenfeld and Kak[11]:

Definition 1: Every *region* that constitutes a hole has a finite area (volume), and is simply connected. For 2D images, 4-connectedness is assumed for the object (pixels of density 1), and 8-connectedness is assumed for the holes (pixels of density[1] 0).

Definition 2: The *boundary curve* of every region that constitutes a hole is a simply connected curve that consists of its boundary points $\{(x_i, y_i) \mid 1 \leqslant i \leqslant k\}$, where k is the number of boundary points.

Definition 3: If, for a boundary curve, $x_{\min} = \min\{x_i\}$, $x_{\max} = \max\{x_i\}$, $y_{\min} = \min\{y_i\}$ and $y_{\max} = \max\{y_i\}$ for $1 \leqslant i \leqslant k$, then the rectangle with the corner points $(x_{\min}, y_{\min})$, $(x_{\max}, y_{\min})$, $(x_{\max}, y_{\max})$ and $(x_{\min}, y_{\max})$ is the *bounding box* of that curve.

Proposition 1: Every simply connected region segments its bounding box into a finite number of subregions whose binary value is complementary to that of the region. In consequence, every subregion is simply connected.

Proposition 2: The bounding box of every subregion is strictly smaller than that of the original region.

Definition 4: There is a *match* between a subregion and a primitive if the primitive approximates the subregion meeting some approximation criteria.

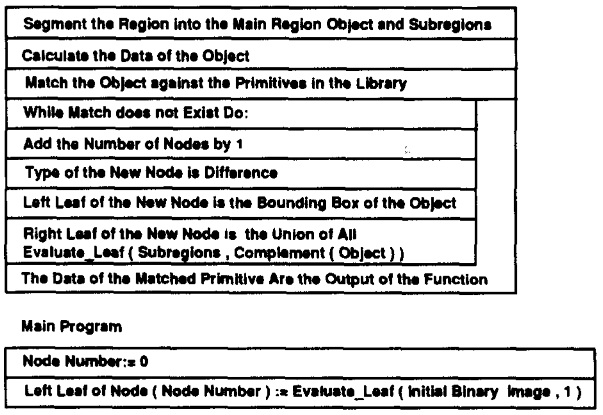
The details of the matching procedure are explained further below. The following algorithm is based on Definitions 1–4 and Propositions 1 and 2. The algorithm converts a 2D binary image (with the requirements mentioned) into a CSG tree of available primitives.

Algorithm 1

(1) Match the object with available primitives. If a match occurs, replace the object by its best-matching primitive in its proper CSG-tree location.

(2) If no match occurs, draw the bounding box of the object, treat each subregion as an object, and go to Step 1 recursively until convergence is reached.

Proposition 2 guarantees the convergence of Algorithm 1 if the rectangle is in the library of primitives. The proper location of every subregion in the CSG tree is determined by the binary value of the subregion and the level of recursion. *Figure 2* is a flowchart of the implemented version of Algorithm 1. The main program calls the function *Evaluate_Leaf* which is of type *Leaf* with the initial image as input. The function segments the image into regions, as described in Reference 1. The data for the object needed for matching are extracted next. Matching follows this calculation (see the next section). If a match occurs, the function *Evaluate_Leaf* is evaluated as the matched primitive. If not, a new node is generated whose regularized Boolean operator is of type difference. The left leaf of this node is the bounding box of the object, and the right node is the union of the *Evaluate_Leaf* functions of all the subregions. (A listing of the program written in PASCAL to implement the 3D algorithm is provided in Appendix G of Reference 1.)


```
Evaluate_Leaf ( Region , Object )
Segment the Region into the Main Region Object and Subregions
Calculate the Data of the Object
Match the Object against the Primitives in the Library
While Match does not Exist Do:
    Add the Number of Nodes by 1
    Type of the New Node is Difference
    Left Leaf of the New Node is the Bounding Box of the Object
    Right Leaf of the New Node is  the Union of All
    Evaluate_Leaf ( Subregions , Complement ( Object ) )
The Data of the Matched Primitive Are the Output of the Function

Main Program
Node Number:= 0
Left Leaf of Node ( Node Number ) := Evaluate_Leaf ( Initial Binary  Image , 1 )
```

Figure 2 Flowchart of implemented version of Algorithm 1

MATCHING

Background

Matching has been one of the most intensively studied subjects in computer vision and artificial intelligence (see, for example, References 4 and 12). The primary emphasis of these studies has been on matching 3D objects in the form of 2D images against known templates.

There are a number of differences between the matching problem studied in computer vision and the one of interest here. The following properties make the present problem easier than the traditional computer-vision one: no occlusion occurs in homogenization images, objects or areas are 2D, and image segmentation does not pose a major problem. However, the matching investigated in this paper must be independent of the size and orientation of the object, and must rely on area properties of the object in 2D images, such that the method can be extended to the 3D case.

A thorough survey of available object-recognition techniques is provided by Chin and Dyer[13]: three basic methods of object recognition are described in Reference 13: global, structural and relational-graph methods. The authors give a description of each method on the basis of the *models* used for the objects, the *features* or properties used to distinguish between objects, and the *matching* procedure. Note that 'feature' in the computer-vision literature means a property of an image or of a region in an image; this is a meaning that is different from the one used in feature-based design. Throughout this section, 'feature' is used in its computer-vision context.

Structural and relational-graph methods use local properties of objects in images, since global methods fail to treat the occlusion problem. Most matching procedures for structural and relational-graph methods are not invariant in terms of rotation, shift, and size. However, two of the methods categorized as structural feature methods are capable of handling size and rotation invariance, namely, generalized Hough transforms (GHT)[14] and Fourier descriptors[15]. The problem with GHT is that it is based on tangent information of the boundary curves of the object, and therefore its extension to 3D object recognition is not straightforward. Modelling and matching objects with Fourier descriptors poses difficulties. Extending Fourier descriptors to 3D object recognition is also questionable. Therefore, the focus of our studies shifts to the global methods[13,16].

These methods use global features to recognize objects. Global features are, for instance, areas, principal area moments of inertia, centroids, perimeters, and compactness[16]. Global feature methods usually treat problems of object recognition by robots and manipulators in industrial environments. These recognition procedures work by matching global features or properties of the model with those of the object(s) in the image. Similar techniques have been used in the current study. The main differences are that, here, matching must be performed at different recursion levels of the algorithm, no-match situations are possible, and matches are not unique.

Procedure

Figure 3 is a schematic overview of the matching algorithm. Note that the emphasis of the matching procedure is on its extendibility to 3D images. The first step is to compute the global features of the objects: area, centroid, principal area moments of inertia and their axes, perimeter, and compactness.

The next step is to use these features to extract matched shapes. Although these features are not alone sufficient to discriminate between shapes, they can prune the set of candidate shapes from a library of primitives. Additionally, features can provide an estimate for the dimensions of the matched primitives. By calculating the centroid, the problem of object recognition becomes shift-invariant. Calculating the principal axes of inertia of the shape provides rotation invariance. The size invariance is solved by checking if some nondimensionalized constraints between the area and the area moments of inertia are satisfied.

The primitives included here are rectangle (square), ellipse (circle), isosceles triangle (equilateral triangle), right-angled triangle, and diamond. Other simple shapes may be added as needed. The derivation of the constraints for the rectangle is provided here for illustration. *Figure 4* shows a rectangle and its principal axes of inertia (1-1 and 2-2). Once the principal axes of inertia and the centroid of the rectangle have been given, two parameters (the breadth b and height h) determine the exact area occupied by the rectangle. The principal area moments of inertia I_1 and I_2 of the rectangle are calculated as

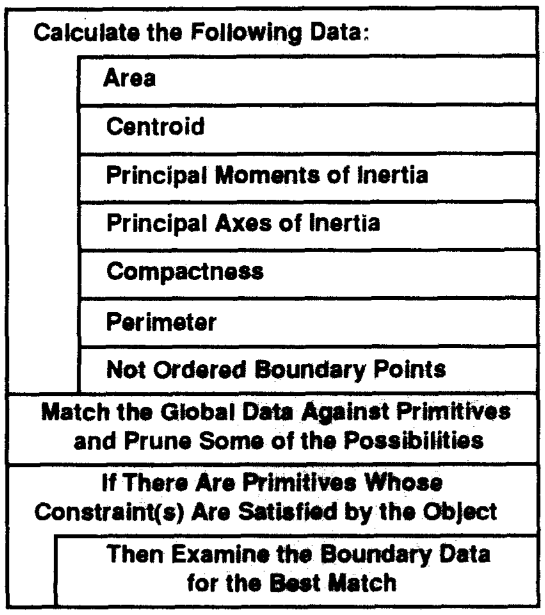

Calculate the Following Data:
Area
Centroid
Principal Moments of Inertia
Principal Axes of Inertia
Compactness
Perimeter
Not Ordered Boundary Points
Match the Global Data Against Primitives and Prune Some of the Possibilities
If There Are Primitives Whose Constraint(s) Are Satisfied by the Object
Then Examine the Boundary Data for the Best Match

Figure 3 Flowchart for matching procedure

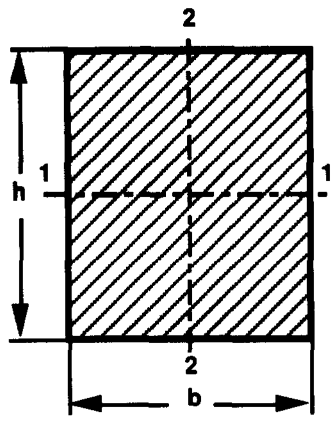


Figure 4 Rectangle, its dimensions, and its principal axes of inertia

follows[17]:

$$I_1 = \frac{bh^3}{12} \qquad (1)$$

$$I_2 = \frac{hb^3}{12} \qquad (2)$$

and its area A is calculated as

$$A = bh \qquad (3)$$

Given I_1, I_2 and A, Equations 1–3 represent an overconstrained system of three nonlinear equations in two unknowns (b and h). Therefore, if a pair b and h can satisfy all three equations, a match is likely. This system of equations is solved here by obtaining b and h from Equations 1 and 2. Their values, given by Equations 4 and 5, respectively, are substituted into Equation 3. If Equation 3 is satisfied with a specified accuracy, a match is considered to be likely. The constraint that must be satisfied can be expressed in terms of the known quantities, as shown in Equation 6.

$$b = \left(\frac{144 I_2^3}{I_1}\right)^{1/8} \qquad (4)$$

$$d = \left(\frac{144 I_1^3}{I_2}\right)^{1/8} \qquad (5)$$

$$A \approx 2(3^{1/2})(I_1 I_2)^{1/4} \qquad (6)$$

Another, more rigorous way to solve this overconstrained system of equations is to construct a weighted error function, and to minimize it with respect to the variables. An example of such an error function, denoted by E, is as follows:

$$E = \left(1 - \frac{bh^3}{12 I_1}\right)^2 + \left(1 - \frac{bh^3}{12 I_2}\right)^2 + \left(1 - \frac{bh}{A}\right)^2 \qquad (7)$$

Setting $\partial E/\partial b$ and $\partial E/\partial h$ equal to zero gives a system of two nonlinear equations in two unknowns (b and h). The values of b and h that minimize the error function do not necessarily guarantee a match. The value of the minimized error function $E(b, h)$ must lie below a certain threshold to make a match likely.

It may be verified that the constraint in Equation 6, derived for a rectangle, is identical to that for a diamond. This identity means that, if a shape satisfies the constraint for a rectangle, it may also be a diamond. The critical question then becomes one of how to discriminate between these two shapes. Compactness, which is a global property based on perimeter and area, is not a very strong discriminator. Some other property must be checked to find the final match. The property chosen here is the sum of the squared deviations of the boundary points of the object from the boundary of the likely primitive matching the object. Note that the boundary points need not be ordered for this operation, thus making a 3D extension relatively easy. *Figure 5* shows the distances of the discrete boundary points of an object from the edges of a matched rectangle.

A few remarks are necessary about the implementation of the matching procedure. The best match for the region among the possible candidate primitives is the primitive with the least sum of boundary deviations. This measure needs to be below a certain threshold to make a match valid. As usual, to maintain the integrity of the procedure regardless of the dimensions of the objects, the measure of the boundary deviations must be nondimensionalized. One way of doing this is to divide by the square of the perimeter. It can be shown that this nondimensionalized quantity is invariant to the proportions of the objects, i.e. two congruent objects have the same deviation measure, as long as their matching objects are also congruent.

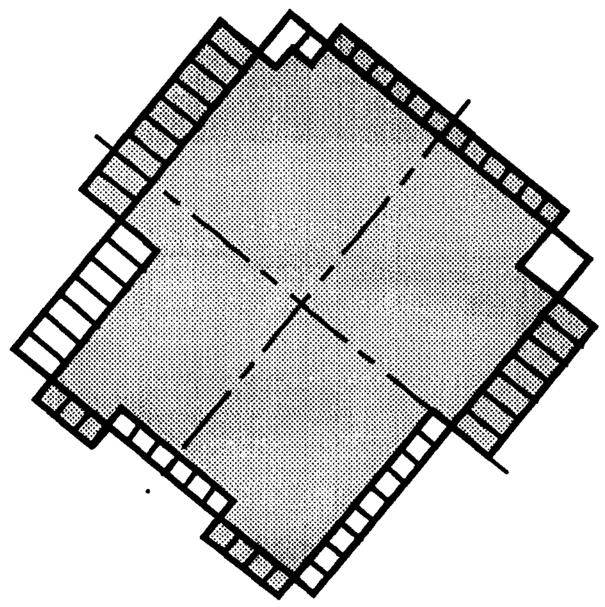


Figure 5 Object (tinted region) with possible matching rectangle [The small extension lines show the distances of the discrete boundary points of the object to the boundary of the rectangle.]

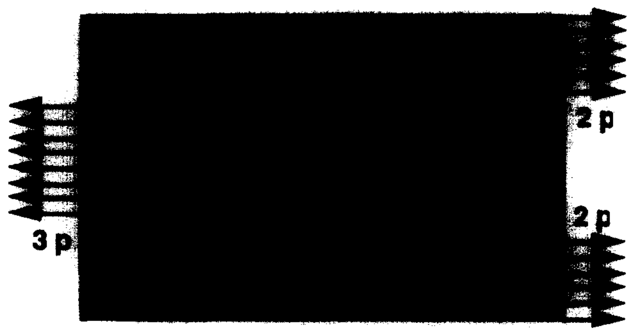
2D EXAMPLES

Example 1 at the beginning of this paper is an actual output of the algorithms discussed above. Two additional examples are provided here. Throughout this paper, the conventions for pointing to the primitives in the figures and for representing the CSG trees are the same as described for the first example of this paper.

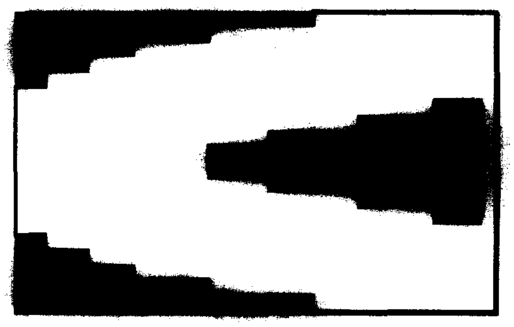
Example 2: Plate under uniform pressure

Figure 6b shows a thresholded image of the homogenization output for the initial design problem shown in *Figure 6a*, in which p is an arbitrary distributed load. The stiffness matrix of this problem is singular, since no kinematic (so-called first-type) boundary conditions are applied. A regularization method[18] is applied to overcome the problem.

Figure 7a shows the primitives extracted by the matching algorithm, and *Figure 7b* shows the CSG tree extracted for this object. The program has information on the exact location of the primitives. As mentioned earlier, the orientation of the object is extracted from their principal axes of inertia. Because of discretization and numerical inaccuracies, this information is noisy. If a primitive appears on the boundaries of the object, its orientation may not be exactly the one seen by the human eye. In other words, the primitive may be slightly inclined.

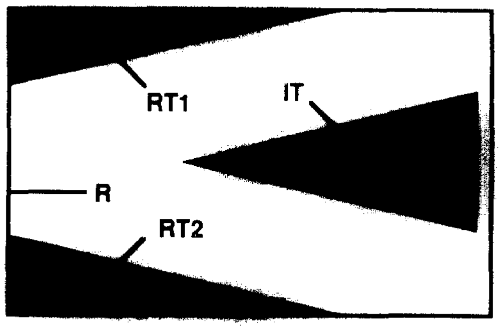


a

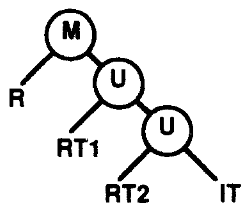


b

Figure 6 Example 2; (a) initial design domain and loading conditions, (b) homogenization output for solid-to-void ratio of 5:2 (i.e. the density constraint is 72%)



a



b

Figure 7 Example 2; (a) primitives matched against regions of object shown in *Figure 6b*, (b) CSG tree approximating object

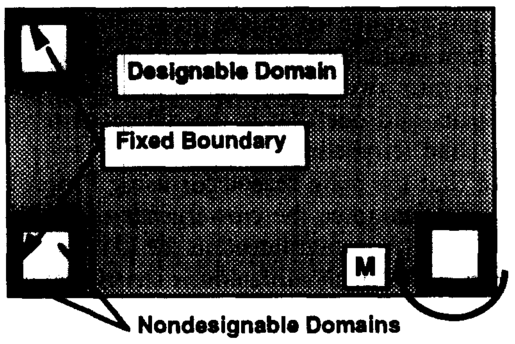
This problem can be easily overcome by a set of simple rules.

Example 3: Bracket subjected to bending moment

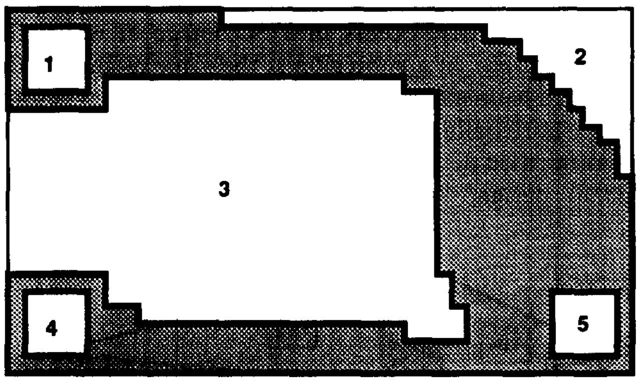
This bracket problem is extensively discussed in the literature[3]. The initial design domain is shown in *Figure 8a*. For this example, however, no force is applied, and only the moment is kept at the right end of the bracket. The binary image output of topology optimization for a volume constraint of 42% (corresponding to a solid-to-void ratio of 21:29) is shown in *Figure 8b*. The object is represented by the tinted region, and the numbers 1–5 denote regions representing holes.

The CSG tree of the object shown in *Figure 8b* consists of the difference of the surrounding rectangle (initial design domain) and the union of holes 1–5. Holes 1, 4 and 5 are simply represented by squares. The primitives and CSG-tree representations of holes 2 and 3 are shown in *Figures 9* and *10*, respectively. *Figure 11* shows the overall result of the algorithm for the object shown in *Figure 8* in the form of a CSG tree. In *Figure 11*, RI is the rectangular initial design domain for this design problem, and *R*1, *R*4 and *R*5 are, respectively, the rectangular holes 1, 4 and 5 shown in *Figure 8*. The remaining primitives have been identified in *Figures 9* and *10*.

As emphasized above, the CSG representation of objects is not unique, and hence the suggested algorithm is just one of many possible ways to convert the spatial-enumeration scheme into a CSG representation.

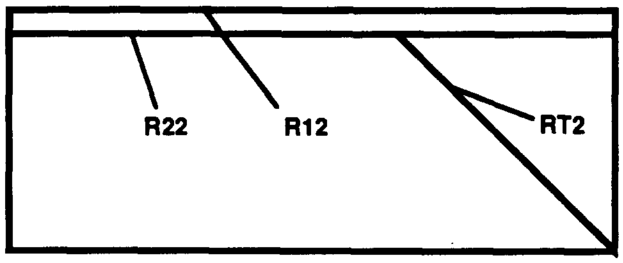


a

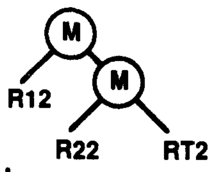


b

Figure 8 Example 3; (a) input to ISOS for bracket problem, (b) output of topology optimization for this problem
[The numbers refer to the holes (white regions), and the main structure (object) is tinted.]



a



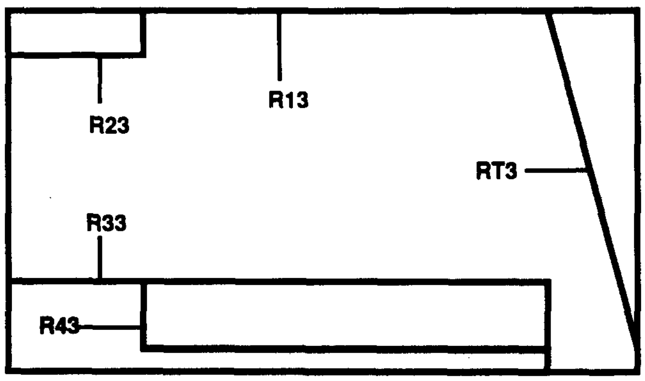
b

Figure 9 Hole 2 in *Figure 8b*; (a) extracted primitives, (b) CSG tree representation

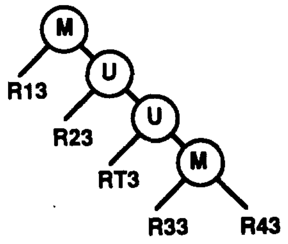
3D EXTENSIONS

One natural way of extending the 2D concepts and methods introduced in the previous sections is to proceed in a manner similar to the so-called *slice* approach taken in computer tomography, that is, to extract the CSG trees of a series of 2D images along one particular axis. Comparing the CSG tree of each layer with the CSG trees of its neighbouring layers reveals information about the CSG tree of the 3D dimensional object as a whole. This approach is suitable for 2.5D structures. An example of this type of structure follows.

Figure 12 gives the 3D initial design domain and boundary and loading conditions for this example (the actual finite-element mesh is $20 \times 40 \times 3$). The points whose x coordinates vanish are clamped. Four equal



a



b

Figure 10 Hole 3 in *Figure 8b*; (a) extracted primitives, (b) CSG tree representation

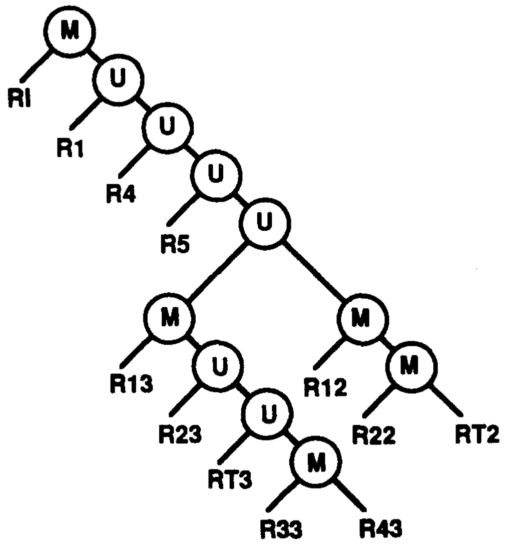


Figure 11 Final output of CSG conversion algorithm for object in *Figure 8b*
[*R*1, *R*4 and *R*5 are, respectively, the rectangular holes 1, 4 and 5, and RI is the rectangular primitive for the initial design domain in *Figure 8b*; the remaining primitives are shown in *Figures 9* and *10*.]

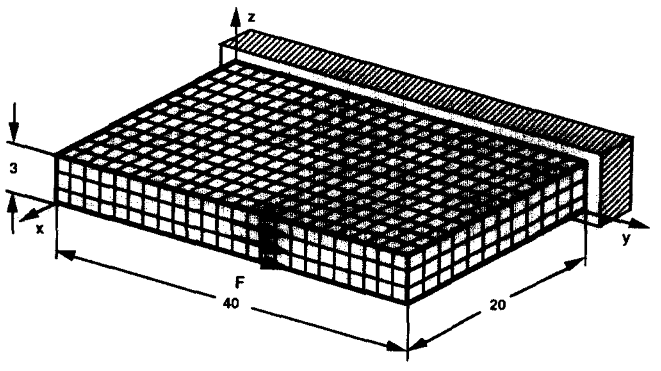


Figure 12 Initial design domain and boundary and loading conditions for 3D example

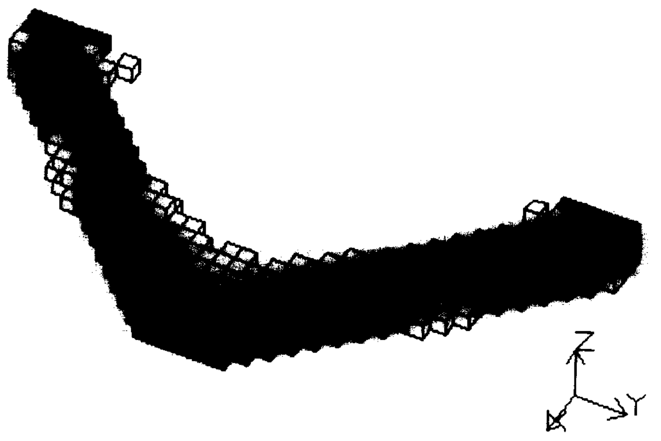


Figure 13 Homogenization output for model of *Figure 12* (solid-to-void ratio is 1:3)

point forces F are applied at points at which $y = 20$, $x = 20$ and $z = 0$, 1, 2 and 3. From studies performed on 2D structures[19], the optimal design is known to be a 2-bar truss. The homogenization output is shown in *Figure 13*. CSG trees for the 2D images in planes parallel to the xy plane consist of the difference of a rectangle and the union of three right-angled triangles. Therefore, the primitives shown in *Figure 14a* and the CSG tree of the solid object shown in *Figure 14b* can be extracted by comparing the CSG trees of the three layers.

To extend the *slice* approach to more complex 3D structures, a set of rules is needed to reconcile dissimilar CSG trees in two, and possibly more, neighbouring layers. There are at least three main axes (say x, y and z) along which the layer-by-layer CSG-tree extraction can be performed. The search is by no means limited to these three axes. Finding the most suitable axis is an issue that requires further investigation.

In the remainder of this section, the 3D extensions of concepts and methods introduced above are discussed. Only slight modifications are necessary to extend the 2D algorithms for labelling the regions to 3D images[12,20]. Face connectivity is used for objects (binary value 1), and edge connectivity for holes (binary value 0).

Definitions 1–4 can easily be extended, but Propositions 1 and 2 are no longer true. *Figure 15* shows a 3D counterexample to those propositions. The solid tetrahedron inside the cube (its bounding box) segments the cube into two subregions, one inside the solid tetrahedron, and one that is the difference of the cube and the solid tetrahedron.

Recall that the main reason for using Propositions 1 and 2 is to guarantee the convergence of Algorithm 1. The convergence is guaranteed in the 2D case, since the bounding box of each subregion is strictly smaller than the bounding box of the object, and the problem has a discrete nature. By definition, the bounding box of subregions cannot be larger than the bounding box of the object.

Now, if a case occurs in a 3D image in which the bounding box of a subregion is the same as that of the object, a heuristic segmentation of the object may resolve this shortcoming. The nature of the heuristic rules implemented depends mostly on the requirements of the detail design, including manufacturing. One simple

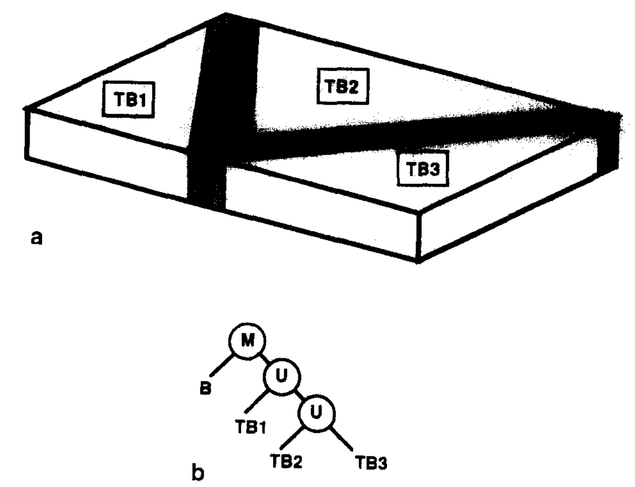


Figure 14 Object; (a) primitives detected from *Figure 13*, where the object is heavily tinted, (b) CSG-tree representation of object
[*TB1*, *TB2* and *TB3* are triangular blocks, and *B* is the block representing the initial design domain.]

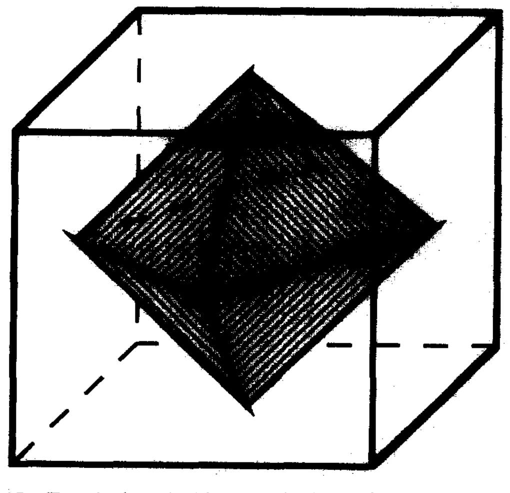


Figure 15 Tetrahedron inside cube (its bounding box) segments box into only two subregions
[The vertices of the tetrahedron are on the faces of the box.]

segmentation may be to divide the image perpendicularly to an arbitrary axis. Such a segmentation will lead to the convergence of the algorithm. Depending on the application, the output may or may not be satisfactory.

The matching part of the algorithm can be extended easily. Matching is shift-invariant after the centroid of the object is located, and it is rotation-invariant by the determination of the three principal axes of inertia[21]. The dimensions of the primitive can be estimated by using the values of the volume and the volume moments of inertia of the object. As in the 2D case, the example of a box (block) is as shown here. The primitives considered for the 3D case are the ellipsoid, cylinder, triangular block, and cone[1]; the cube and sphere are special cases of a block and ellipsoid, respectively. This list is by no means complete.

Figure 16 shows a block and its dimensions, i.e. its width w, breadth b and height h, and its principal axes of inertia. The principal moments of inertia I_1, I_2 and I_3 of the block are calculated as follows (see Reference 21 for these and other formulae for the volume moments of inertia of 3D objects):

$$I_1=\frac{bwh(b^2+w^2)}{12} \tag{8}$$

$$I_2=\frac{bwh(b^2+h^2)}{12} \tag{9}$$

$$I_3=\frac{bwh(w^2+h^2)}{12} \tag{10}$$

and its volume V is trivially calculated as follows:

$$V=bhw \tag{11}$$

Given I_1, I_2, I_3 and V, Equations 8–11 represent an overconstrained system of four nonlinear equations in three unknowns (b, w and h). Therefore, if a triplet (b, w, h) can satisfy all four equations, a match is likely. The way in which this system of equations is solved for the matching procedure is that b, w and h are obtained from Equations 8–10. Their values, shown in Equations 12–14, respectively, are substituted into Equation 11. If Equation 11 is satisfied with a certain accuracy, a match is considered to be likely. The constraint that must be satisfied can be expressed in terms of the known quantities, as shown in Equation 15. An error function similar to the one introduced above for 2D matching can be used for the 3D case:

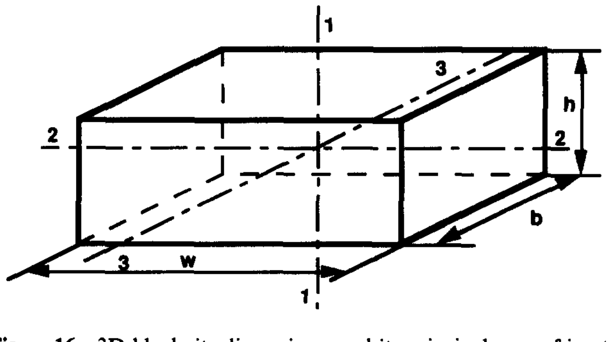


Figure 16 3D block, its dimensions, and its principal axes of inertia

$$b=\left(\frac{6(I_1+I_2-I_3)}{V}\right)^{1/2} \tag{12}$$

$$h=\left(\frac{6(I_2+I_3-I_1)}{V}\right)^{1/2} \tag{13}$$

$$w=\left(\frac{6(I_3+I_1-I_2)}{V}\right)^{1/2} \tag{14}$$

$$V\approx(216(I_1+I_2-I_3)(I_2+I_3-I_1)(I_2+I_3-I_1))^{1/5} \tag{15}$$

The distance of a boundary point of the object from the boundary of a primitive can be calculated by appropriate linear transformations in space. Since the surfaces of the primitives are of second order at most, the distance can be calculated analytically.

3D EXAMPLES

The examples given in this section illustrate the 3D capabilities of this algorithm.

Example 4: Sandwich structure

This example has been suggested and solved previously by Suzuki[22] with different density constraints. The initial design model is shown in *Figure 17*. The load F is acting in the negative direction of the z axis.

The homogenization output for a solid-to-void ratio of 1:1 is shown in *Figure 18*, and it is a sandwich structure. The threshold value to generate the binary image is 0.5, and its choice is not critical for the outcome of the algorithm. The upper and lower layers are identical, and are visible in *Figure 18*. A cut through the middle of the structure parallel to the xy plane is shown in *Figure 19*, which shows the two identical middle layers.

The output of the algorithm is shown in *Figure 20*. *Figure 20a* shows the primitives in an axonometric view. *Figure 20c* shows the extracted CSG-tree representation for the object. It is basically the difference of the block $B1$ representing the initial design domain and the union of the holes 1, 2, 3, 4 and object OM (whose top view is shown in *Figure 19*). The hole OM cannot be approximated by a single primitive, and it becomes the difference of $B2$, as shown in *Figures 19* and *20b*, and the union of regions 5–8, which are approximated by corresponding triangular blocks.

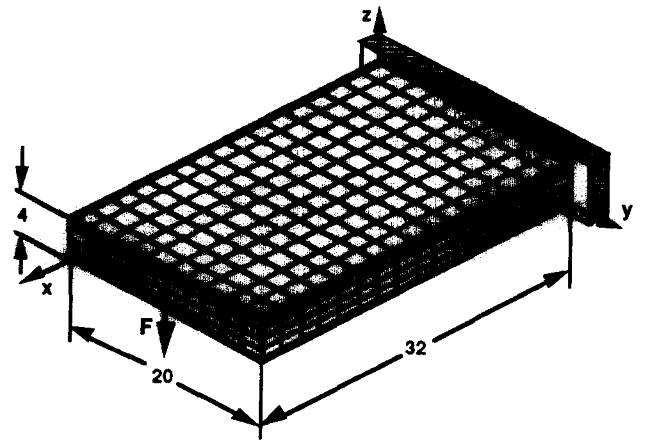


Figure 17 Example 4: initial design domain and boundary and loading conditions

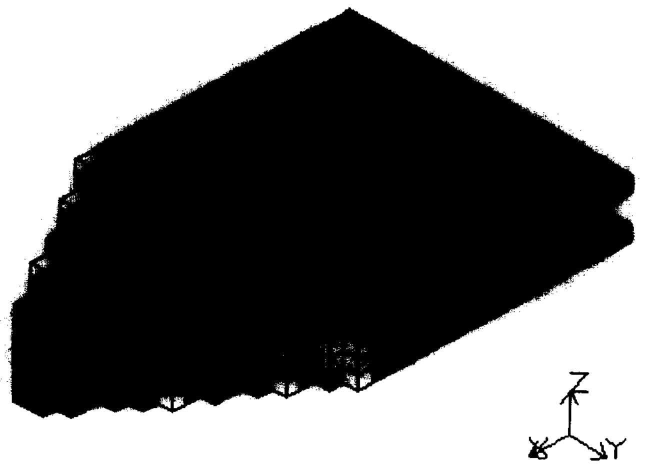


Figure 18 Example 4: homogenization output for model of *Figure 17* (solid-to-void ratio is 1:1)

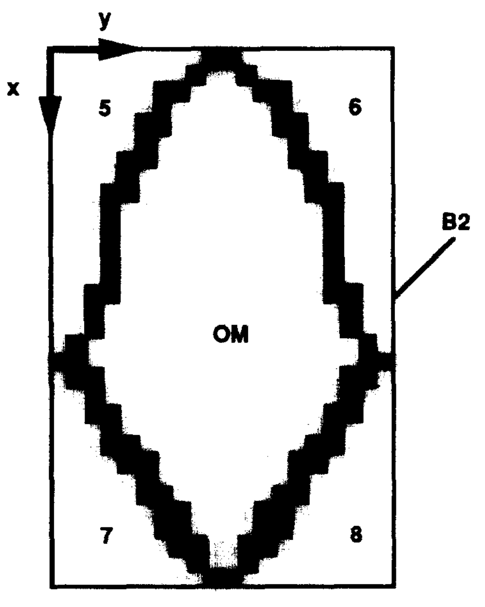


Figure 19 Section of structure shown in *Figure 18*, where cutting plane is parallel to xy plane and at $z=2$

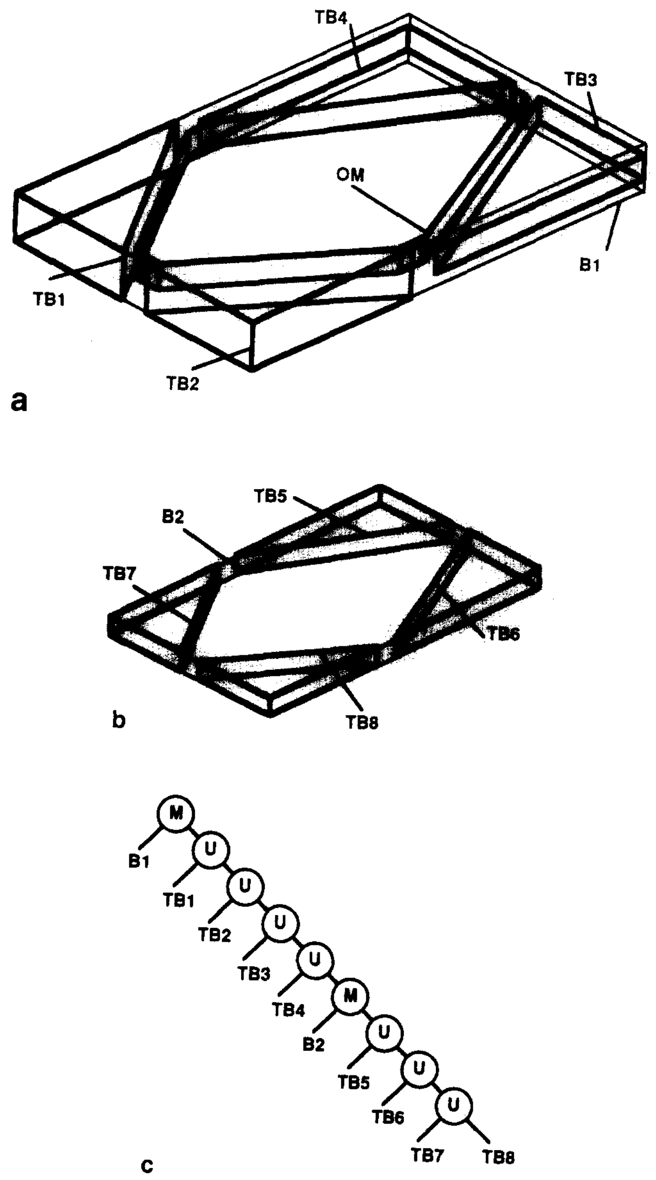


Figure 20 Example 4; (a) primitives approximating structure, (b) primitives approximating hole OM in middle of structure, (c) CSG tree of structure

Example 5: Solid cube under torsion

The initial design domain (a cube) and the boundary and loading conditions for this example are shown in *Figure 21*. The finite element mesh is $12 \times 12 \times 12$, and each element side is of unit length. As shown in *Figure 21*, the torque is applied by exerting two equal force couples on the sides of the end square. The last row of the design domain (between $x=11$ and $x=12$) consists of undesignable elements with a density of 1.

The output of the topology optimization for a solid-to-void ratio of 1:1 has predominantly internal features. Therefore, the output is shown in *Figures 22a* and *b* in terms of sections where the cutting planes are perpendicular to the z and x axes, respectively. The threshold value is 0.4. The white and tinted regions indicate elements whose density values are 1 and 0, respectively.

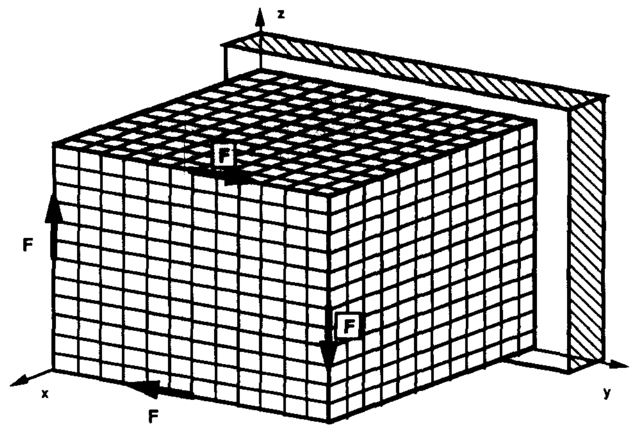


Figure 21 Example 5: initial design domain and boundary and loading conditions

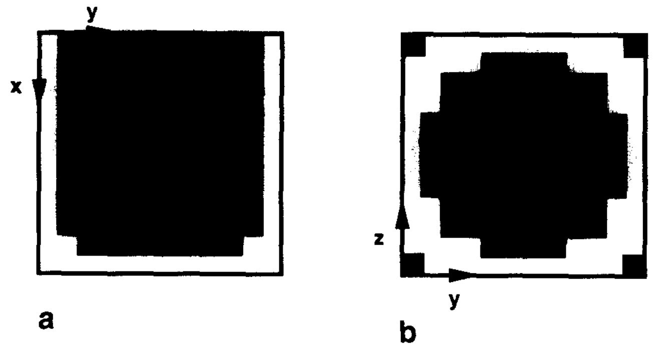


Figure 22 Example 5: sections through output of topology optimization for solid-to-void ratio of 1:1; (a) cutting plane perpendicular to z axis at $z=6$, (b) cutting plane perpendicular to x axis at $x=6$
[The white and tinted regions show elements of densities 1 and 0, respectively.]

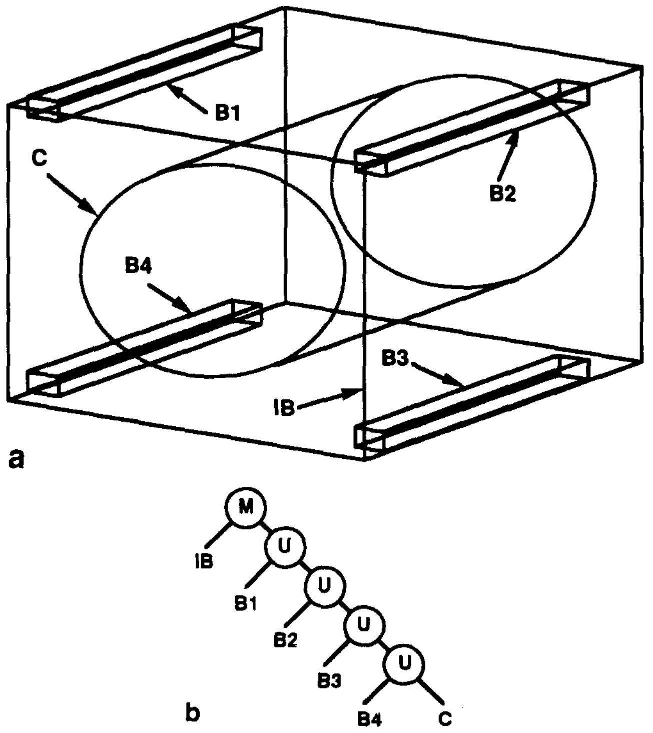


Figure 23 Example 5: output of Algorithm 1; (a) axonometric view of extracted features in wireframe representation, (b) CSG tree of object

The output of the extended Algorithm 1 is shown in *Figure 23*. *Figure 23a* shows an axonometric view of the extracted features in a wireframe representation. The CSG tree of the object is shown in *Figure 23b*, which is basically the difference of the initial design domain IB and the union of the holes represented by a cylinder C and four rectangular blocks *B*1, *B*2, *B*3 and *B*4. The radius and height of the cylinder are found to be 4.9 and 10.8 units, respectively.

CONCLUDING REMARKS

The approach suggested and implemented in this paper is an initial step towards the full automation of ISOS for 3D structures. A natural next step is to generate a 3D finite-element mesh on the basis of the information provided by the algorithm. (Automatic mesh generation for 3D objects is an active area of research.) The finite-element mesh will be used for detailed analysis and optimization activities in a subsequent detailed design optimization. The 3D algorithm may be augmented by a heuristic module to guarantee its convergence.

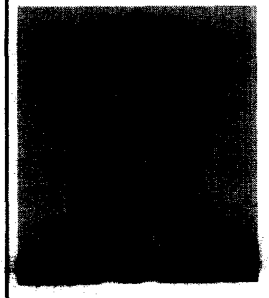
ACKNOWLEDGEMENTS

The research presented in this paper has been partially funded by the US National Science Foundation and Ford Motor Company. This support is gratefully acknowledged. The authors wish to express their thanks to Professor D Dutta of the University of Michigan and the other referees of the paper for their comments and suggestions.

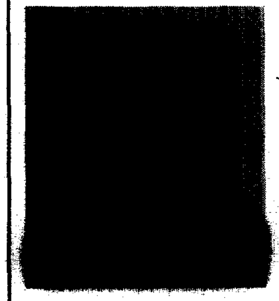
REFERENCES

1 Chirehdast, M 'An integrated structural optimization environment for structural configuration design' *PhD Thesis* Dep. Mechanical Engineering & Applied Mechanics, University of Michigan, USA (1992)
2 Bendsøe, M and Kikuchi, N 'Generating optimal topologies in structural design using a homogenization method' *Comput. Methods Appl. Mech. & Eng.* Vol 71 (1988) pp 197–224
3 Papalambros, P and Chirehdast, M 'An integrated environment for structural configuration design' *J. Eng. Des.* Vol 1 No 1 (1990) pp 73–96
4 Davies, E R *Machine Vision: Theory, Algorithms, Practicalities* Academic Press, UK (1990)
5 Udupa, J K and Herman, G T *3D Imaging in Medicine* CRC Press (1991)
6 Samet, H *The Design and Analysis of Spatial Data Structures* Addison-Wesley, USA (1990)
7 Requicha, A A G 'Representation for rigid solids: theory, methods and systems' *ACM Comput. Surv.* Vol 12 (1980) pp 437–464
8 Hoffman, C M *Geometric and Solid Modeling: An Introduction* Morgan Kaufmann, USA (1989)
9 Shapiro, V and Vossler, D 'Efficient CSG representation of planar solids' *ASME J. Mech. Des.* Vol 113 (1991) pp 292–305

10 Shapiro, V and Vossler, D 'Construction and optimization of CSG representations' *Comput.-Aided Des.* Vol 23 No 1 (1991) pp 4-20
11 Rosenfeld, A and Kak, A *Digital Picture Processing – Vols 1 & 2* Academic Press, USA (1982)
12 Ballard, D and Brown, C *Computer Vision* Prentice-Hall, USA (1989)
13 Chin, R T and Dyer, C R 'Model-based recognition in robot vision' *Comput. Surv.* Vol 18 No 1 (1986) pp 67-108
14 Ballard, D H 'Generalizing the Hough transform to detect arbitrary shapes' *Pattern Recognition* Vol 13 No 2 (1981) pp 111-122
15 Zahn, C T and Roskiew, R Z 'Fourier descriptors for plane closed curves' *IEEE Trans. Comput.* Vol 26 No 9 (1972) pp 882-894
16 Jain, A K *Fundamentals of Digital Image Processing* Prentice-Hall, USA (1989)
17 Young, W C *Roark's Formulas for Stress and Strain* (6th Ed.) McGraw-Hill, USA (1989)
18 Kikuchi, N *Finite Element Methods in Mechanics* Cambridge University Press, UK (1986)
19 Suzuki, K and Kikuchi, N 'Homogenization method for shape and topology optimization' *Comput. Methods Appl. Mech. & Eng.* Vol 93 (1991) pp 291-318
20 Liu, H K 'Two- and three-dimensional boundary detection' *Comput. Graph. & Image Proc.* Vol 6 (1977) pp 123-134
21 Higdon, A and Stiles, W B *Engineering Mechanics, Vector Edition: Statics and Dynamics* Prentice-Hall, USA (1962)
22 Suzuki, K 'Shape and layout optimization using homogenization method' *PhD Thesis* Dep. Mechanical Engineering & Applied Mechanics, University of Michigan, USA (1991)



Mehran Chirehdast gained a Diplom-Ingenieur degree in mechanical engineering from the Technical University of Vienna, Austria, in 1987, and an MSc in engineering and a PhD in mechanical engineering from the University of Michigan, USA, in 1988 and 1992, respectively. He is a senior research engineer in the Alpha Simultaneous Engineering Division of the Ford Motor Company, USA. His research and development interests are in design automation and optimization, the durability of structures, and CAD/CAE in industrial environments.



Panos Y Papalambros received a diploma in mechanical and electrical engineering from the National Technical University of Athens, Greece (1974), and an MS (1976) and a PhD (1979) from Stanford University, USA. Having been on the University of Michigan, USA, faculty since 1979, he is currently serving as Chair of the Department of Mechanical Engineering and Applied Mechanics. He is a fellow of the ASME.