Block Cartesian Abstraction of a Geometric Model Using Fuzzy Logic

Y. Su[1] and A. Senthil Kumar[2]

[1]Institute of High Performance Computing, suyi@ihpc.a-star.edu.sg
[2]National University of Singapore, mpeask@nus.edu.sg

ABSTRACT

In this work, a fuzzy logic approach is proposed to transform a geometric model of arbitrary shape to its block Cartesian abstraction. This abstraction is topologically similar to the original model and it contains geometric sub-entities which are all aligned in the Cartesian directions. This is achieved by calculating the modifications made to the face normals as a result of the influences of the adjacent faces. A fuzzy logic inference engine is developed by combining heuristics to emulate the local changes in face normals with respect to the changes in the global space. A three-dimensional field morphing algorithm is used to position the features of this Cartesian abstraction so that a congruent geometric model can be reconstructed. Such a model is useful for the generation of structured quadrilateral boundary element meshes or structured hexahedral meshes based on grid-based meshing method, mesh mapping or sweeping. This approach is also able to overcome the traditional problem of having poorly shaped elements at the boundary using the grid-based method of mesh generation. As the topology of the Cartesian abstraction is congruent to the original model, the mesh can be mapped back to the original model by employing an inverse operation of the transformation.

Keywords: Fuzzy logic; Field morphing; Hexahedral mesh generation.

1. INTRODUCTION

In the preparation of a simulation model for numerical analysis, it is often required to pre-process the geometric model so that it can be meshed effectively. Processes like feature recognition and suppression, and domain decomposition are commonly employed with mesh generation algorithms to automatically create the finite element mesh. While automatic tetrahedral mesh generation techniques have matured, robust automatic hexahedral mesh generation remains a challenge. Given the rigid nature of the hexahedral grid [18], it becomes even more difficult if a structured mesh is required.

The research in automatic hexahedral meshing algorithms can be classified under three main categories: the block decomposition method, the superposition method and the advancing front method. The block decomposition approach involves subdividing the domain into meshable sub-entities and then using appropriate algorithms to discretize these sub-parts. Examples of such algorithms are the swept volume decomposition and recomposition method [6], the medial axis transformation [1],[12-13] and the midpoint subdivision and integer programming method [7], and the basic logical bulk shape (BLOBS) method [8-10]. The advancing front approach generates the mesh by starting at the boundary of the model and progressively building elements into the interior of the model. Some examples of algorithms employing this approach are the whisker weaving method [3],[17] and the plastering method [2],[11]. In the superposition approach, a sufficiently large mesh is superimposed on the model and it is then adapted to the boundary of the model. Examples of such a class of algorithms are the modified grid-based method which can use the isomorphic transformation approach [14] or projective approach [16]. Other variants involve using the octree scheme [15], or a sculpting algorithm [19] to generate the initial mesh.

The advantages of the different meshing techniques are balanced between two important issues: the quality of the boundary mesh and the quality of the core mesh. Achieving one usually compromises the other. A comparison of the different mesh generation algorithms was made by Su *et al*. [16].

In this paper, a fuzzy logic inference engine is proposed to map the geometric domain to a block Cartesian space. By meshing this block Cartesian abstraction, both the boundary and core mesh can be of high quality. Moreover, the mesh is essentially structured in nature. Apart from the immediate application in automatic hexahedral mesh generation, the block Cartesian abstraction is also useful in the area of feature recognition and domain decomposition. The paper is organized as follows. Section 2 describes the methodology of the algorithm, section 3 presents the usefulness of this method with respect to the new grid-based hexahedral mesh generation algorithm, and section 5 concludes the paper.

2. METHODOLOGY

The objective of this paper is to develop a robust algorithm to obtain a block Cartesian abstraction of a solid model with arbitrary shape. The task is to modify the original geometric model such that its sub-entities conform to the Cartesian directions, that is, its faces lie along the *xy*-, the *yz*- or the *zx*-plane, and its edges are parallel to the *x*, *y* or *z*-direction. Chiba *et al*. [4] has proposed a method to generate such a recognition model. However, the algorithm faces stability problems and it fails to converge when certain features are encountered, like a 45° chamfer. In this paper, a new fuzzy logic engine is proposed. The major differences are as follows:

i) The use of surface normals to calculate the new orientations of the sub-entities of the model rather than using edge directions.
ii) The application of a different fuzzy logic inference engine for the computation of the new orientations of geometric entities.
iii) The employment of a feature placement algorithm for positioning the features of the model.

2.1 Creation of a Tessellated Model

To generate a Cartesian abstraction, it is first required to obtain a tessellated model so that every curved edge is approximated by straight line segments and every curved surface is approximated by triangular facets while planar faces are approximated by polygonal boundaries. The degree of tessellation must be such that the number of line segments and facets is minimal yet adequately represents the original model. An estimated length of the arc segment l used in the tessellation is given by

$$l \approx \frac{\pi}{4K_{\max}} \qquad (1)$$

where $K_{\max}$ is the maximum curvature of the edge. If the curve is a straight line ($K = 0$), it is not tessellated. Next, a set of triangular facets are used to approximate all non-planar faces using the tessellated edges as contraints to the triangulation, which is achieved by standard Delaunay's algorithm, as illustrated in Fig. 1.

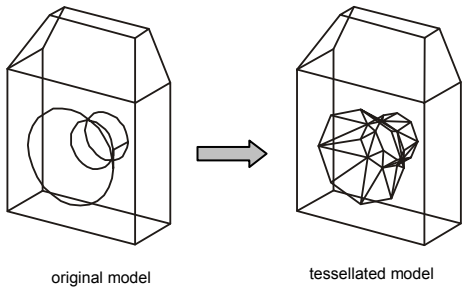

Fig. 1. Tessellation of a geometric model

2.2 Face Normal Reassignment

In order to create a Cartesian abstraction, all the face normals of the tessellated model must be reoriented in the *x*, *y*, or *z*-direction. There is, however, no unique way of determining the directions of the face normals and the problem is made much more complicated since changes made in local regions have an impact in the global sense. To solve this problem, a fuzzy logic system with three inputs (antecedent) and one output (consequent) is implemented. Consider two adjacent faces *A* and *B* as shown in Fig. 2., the probabilities ($P_{\eta,A}$ and $P_{\eta,B}$) that their face normals are assigned to the η-Cartesian directions are determined based on relation shown in Fig. 3(a)., where θ_η is the angle between the face normal and the η-direction. The probability P_α that these two faces are assigned to the same direction is also determined based on relation shown in Fig. 3(b)., where θ is the angle between the two adjacent faces *A* and *B*.

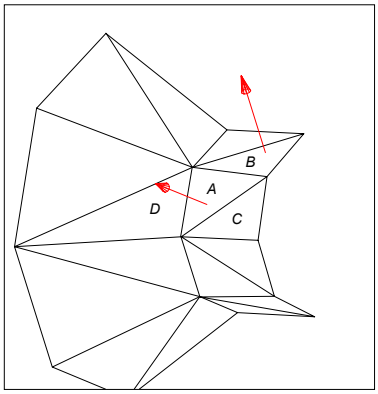

Fig. 2. A pair of adjacent faces *A* and *B*

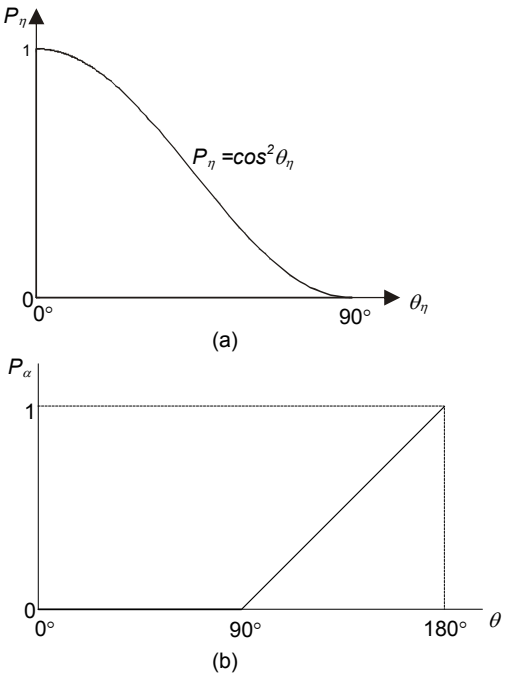

Fig. 3. Relations between (a) P_η and θ_η, and (b) P_α and θ

Given these values, the interest is to find the effect which one surface has on the other in terms of the assignment modification $\Delta P_{\eta,A}$ in each of the η-Cartesian direction. The logic of this system is described as follows:

> If P_α is high, then the assignment modification tends to change the normal direction of surface A to the direction of that of surface B. However, if P_α is low, then the assignment modification tends to change the normal direction of surface A away from that of surface B.

The rule-base with multiple antecedent and single consequent variables is

Rule 1: **IF** $P_{\eta,A}$ is high **AND** $P_{\eta,B}$ is high **AND** P_α is high **THEN** $\Delta P_{\eta,A}$ is positive
ALSO
Rule 2: **IF** $P_{\eta,A}$ is low **AND** $P_{\eta,B}$ is low **AND** P_α is high **THEN** $\Delta P_{\eta,A}$ is negative
ALSO
Rule 3: **IF** $P_{\eta,A}$ is high **AND** $P_{\eta,B}$ is low **AND** P_α is low **THEN** $\Delta P_{\eta,A}$ is positive
ALSO
Rule 4: **IF** $P_{\eta,A}$ is low **AND** $P_{\eta,B}$ is high **AND** P_α is low **THEN** $\Delta P_{\eta,A}$ is negative

The membership functions of the fuzzy sets are illustrated in Fig. 4. and the Multiple-Input, Single-Output (MISO) linguistic model is illustrated in Fig. 5.

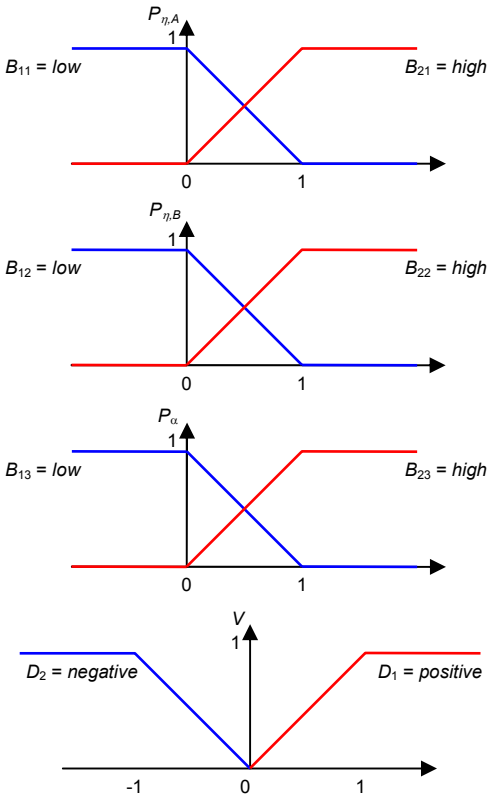

Fig. 4. Membership functions of fuzzy sets

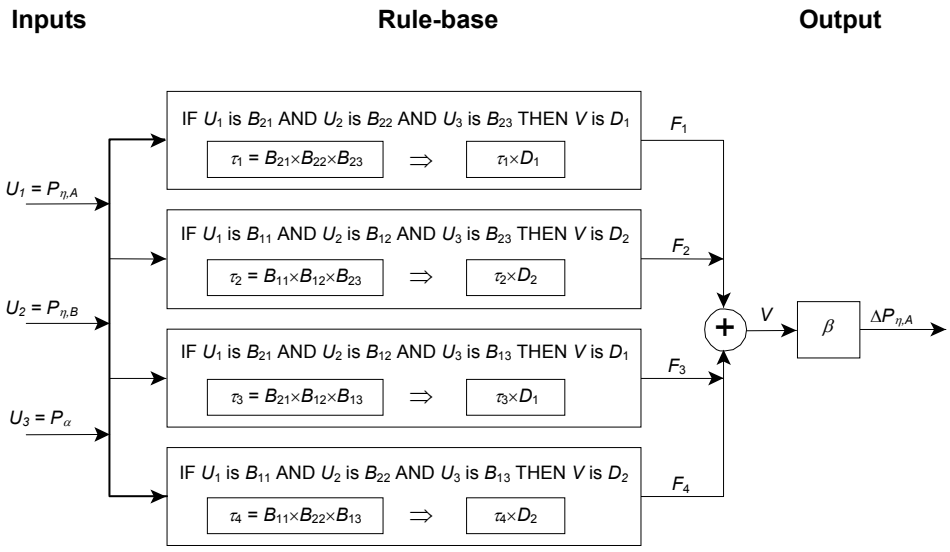

Fig. 5. Multiple-Input, Single-Output (MISO) linguistic model

The algorithm to obtain the crisp output is a two-step process:

i) For each rule of the linguistic model, calculate the degree of firing τ_i using Larsen's method (multiplicative product) [21]

$$\tau_i = B_{i1}(P_{\eta,A}) \times B_{i2}(P_{\eta,B}) \times B_{i3}(P_\alpha) \tag{2}$$

ii) Use the product-sum method to obtain the fuzzy set F_i inferred by the i^{th} rule and aggregate the inferred fuzzy sets to obtain the output

$$F_i = \tau_i D_i \tag{3}$$

$$V = \sum_{i=1}^{m} F_i = \sum_{i=1}^{m} \tau_i D_i \tag{4}$$

The product-sum method yields

$$\Delta P_{\eta,A} = \beta[\Delta P_{\eta,A} - \Delta P_{\eta,B} + (2\Delta P_{\eta,B} - 1)\, P_\alpha] \tag{5}$$

For face A which has m adjacent faces,

$$\Delta P_{\eta,A} = \sum_{i=1}^{m} \Delta P_{\eta,i} \tag{6}$$

A problem that this algorithm faces is the case when $P_{x,A} = 0.5$. In other words, face *A* makes an angle of 45° with the *x*-axis and is at 135° to face *B*. This configuration is commonly found at chamfered corners. To overcome this problem, an area sensitivity factor ΔP_α and a random factor is introduced to P_α such that

$$P_\alpha' = P_\alpha + \Delta P_\alpha + e\times randn \tag{7}$$

where *e* is a sufficiently small number and *randn* is a random number chosen from a normal-distribution with mean zero and variance one. The complete system is illustrated in Fig. 6.

For each iteration *k*, the modified assignment probability for P_i is then calculated from

$$P_{\eta,i}(k+1) = P_{\eta,i}(k) + \Delta P_{\eta,i}(k) \tag{8}$$

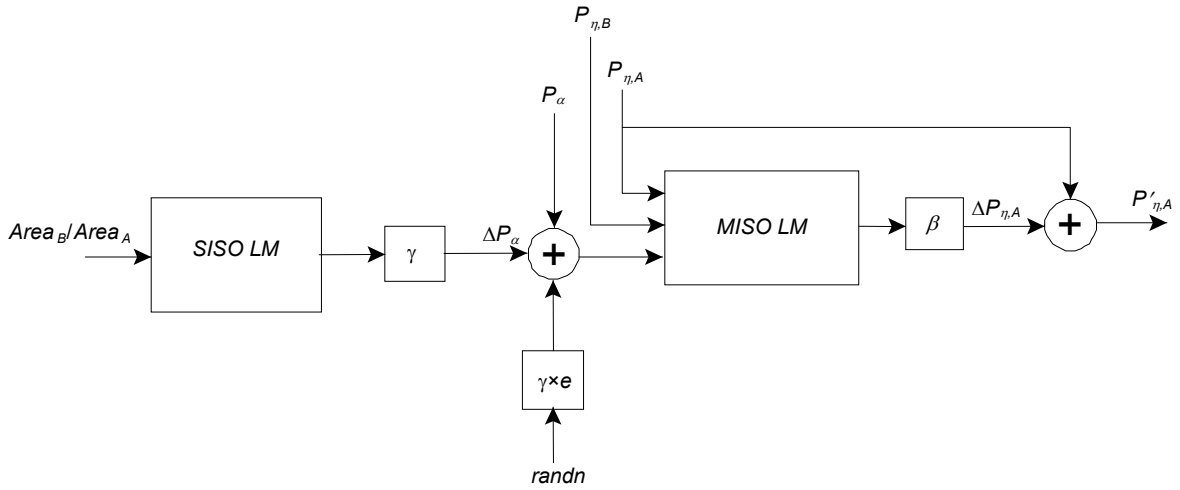

Fig. 6. Complete fuzzy logic inference engine

2.3 Edge Length and Direction Assignment

After the directions of the face normals have been reassigned, the directions of the edges for each surface are determined. For a surface *i*, the direction of its j^{th} edge is calculated as follows:

$$\bar{e}_{i,j} = \bar{n}_i \times \bar{n}_j \tag{9}$$

where $\bar{e}_{i,j}$ is the direction of the j^{th} edge of surface *i*, $\bar{n}_i$ is the surface normal direction of surface *i*, and $\bar{n}_j$ is the normal direction of its adjacent surface at edge *j*.

The calculation of the new length of each edge after the edge direction assignment is based on simple proportion. Since every edge is already in the *x*, *y* or *z*-direction, then the sum of the edge lengths in the positive orientation must equal the sum of the edge lengths in the negative orientation. Thus, for an edge *l* in consideration,

$$l' = \frac{\sum l_\omega}{2\sum l_\delta} \tag{10}$$

where *l'* is the new edge length, l_ω is the length of an edge in the same orientation ($\omega = x, y$ or z), and l_δ is the length of an edge in the same direction ($\delta = \pm x, \pm y$ or $\pm z$). Since each edge is shared by two surfaces, the new length is the average of the two lengths calculated for the surfaces.

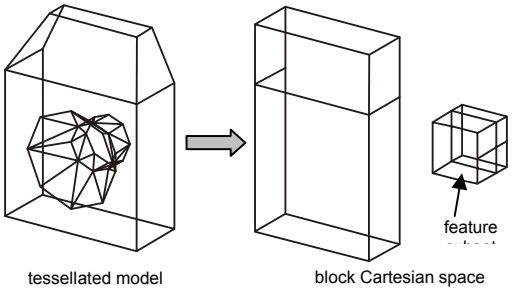

Fig. 7. Feature subset in the block Cartesian space

2.4 Feature Placement Using a Modified Field Morphing Technique

After determining the new face normal directions and edge lengths, the task remains to construct the block Cartesian model. As such, information is inferred from the original set of faces to determine how the positions of features are affected after the transformation. A feature is identified by a group of interconnected edges whereby some of the edges form the inner boundary of some faces of the original model, as shown in Fig. 7. To approximate the position of a feature with respect to the main body, the new positions of every vertex on the feature is calculated based upon the influences of the surrounding control primitives.

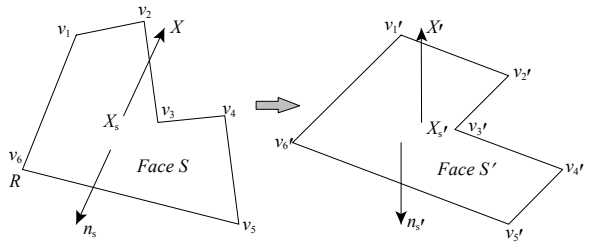

Fig. 8. Transformation of a face S to S′

A vertex X of the feature undergoes a coordinate mapping to a new location X' based upon the transformation between a pair of faces as illustrated in Fig. 8. First, a point X_s is defined such that X_s is the perpendicular projection of X onto face S. Given a reference point R of the surface S and its unit normal vector $\bar{n}_s$, X_s can be found by using the following equation:

$$X_s = X - \left[(X - R) \cdot \bar{n}_s\right]\bar{n}_s \tag{11}$$

After X_s is determined, the corresponding point X_s' of the surface S' is calculated. To find the new position X', the following equation is used:

$$X' = X_s' + \left[\sqrt{\frac{A'}{A}}(X - X_s) \cdot \bar{n}_s\right]\bar{n}_s \tag{12}$$

where A' and A are the surface areas of S' and S, respectively. Also, the coordinate mapping of each vertex must be weighted with respect to all the faces of the object. The weight of the i^{th} pair of faces is computed as follows:

$$w_i = \left(\frac{A_i^p}{a + d_i}\right)^b \tag{13}$$

where A is the area of the face S, and d is the distance between X and S. If X_s lies within the outer loop of S, then d is given by $\left\|X - X_s\right\|$. Otherwise, d is the closest distance to any outer edge of S. The values of a, b and p used here are 0.005, 2 and 1, respectively. For m pairs of faces, the final position of the vertex is computed using the following equation:

$$X' = X + \frac{\sum_i^m w_i (X_i' - X_i)}{\sum_i^m w_i} \tag{14}$$

The complete process of constructing a recognition model from the original model is illustrated in Fig 9.

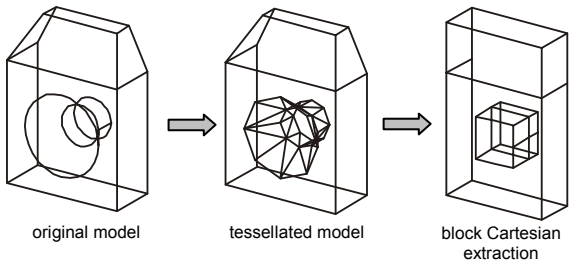

Fig. 9. Construction of block Cartesian abstraction

3. DISCUSSIONS

In this section, a new grid-based method [16] is employed to mesh the block Cartesian abstraction of a model using hexahedral elements, as shown in Fig. 10. A Laplacian-Isoparametric transformation [5] is used to map the mesh back to the original model. The mapping is performed progressively by transforming the nodes on the vertices, then for those on the edges, and followed by those on the faces and finally for the nodes inside the solid. In general, the transformation of nodal locations on an n-dimensional entity is done by fixing the nodes on its (n - 1)-dimensional sub-entities. It is observed that the final mesh is boundary sensitive. Thus, by using the block Cartesian abstraction as a mapping space, the inherent disadvantage of having poorly shaped elements at the boundary in the grid-based type of hexahedral mesh generation algorithm is avoided. In general, if a block Cartesian model can be abstracted, then a hexahedral mesh can be derived. Moreover, by employing certain types of mesh generation algorithm, like the sub-mapping [20] and grid-based algorithm [16], a structured hexahedral mesh can be obtained, as shown in Fig. 11.

One inherent problem in the method presented in this paper is that certain geometrical configurations are very difficult to mesh with elements of high quality. This occurs when the local geometry tapers significantly, with the worst cases being corners which are less than 45° or greater than 135°. Using a simple grid overlay to generate the mesh does not suffice in such cases and

modification to the mesh is required. Research work is currently being undertaken in this area.

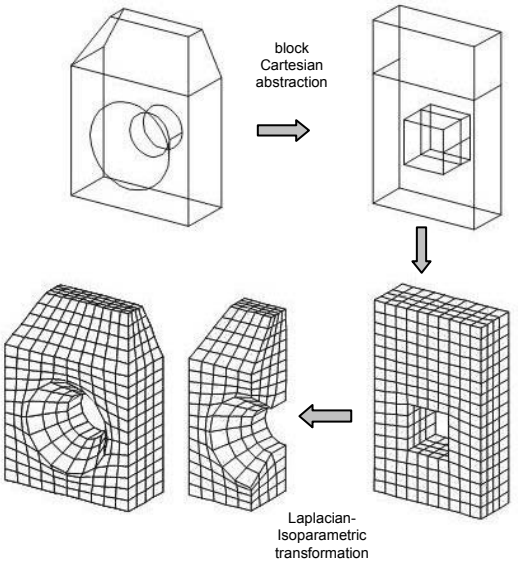

Fig. 10. Hexahedral mesh generation using new grid-based algorithm and Laplacian-Isoparametric transformation

4. CONCLUSION

Using a fuzzy logic inference engine to derive the block Cartesian abstraction of a geometric model is a viable and effective approach. Such an abstraction is useful in applications like domain decomposition, feature extraction and mesh generation. This is especially so in the area of hexahedral mesh generation where boundary sensitivity is an important issue. This approach also facilitates the construction of a structured mesh. An issue which requires further research effort is the problem of degeneracies which occur due to sharply varying geometry and shape.

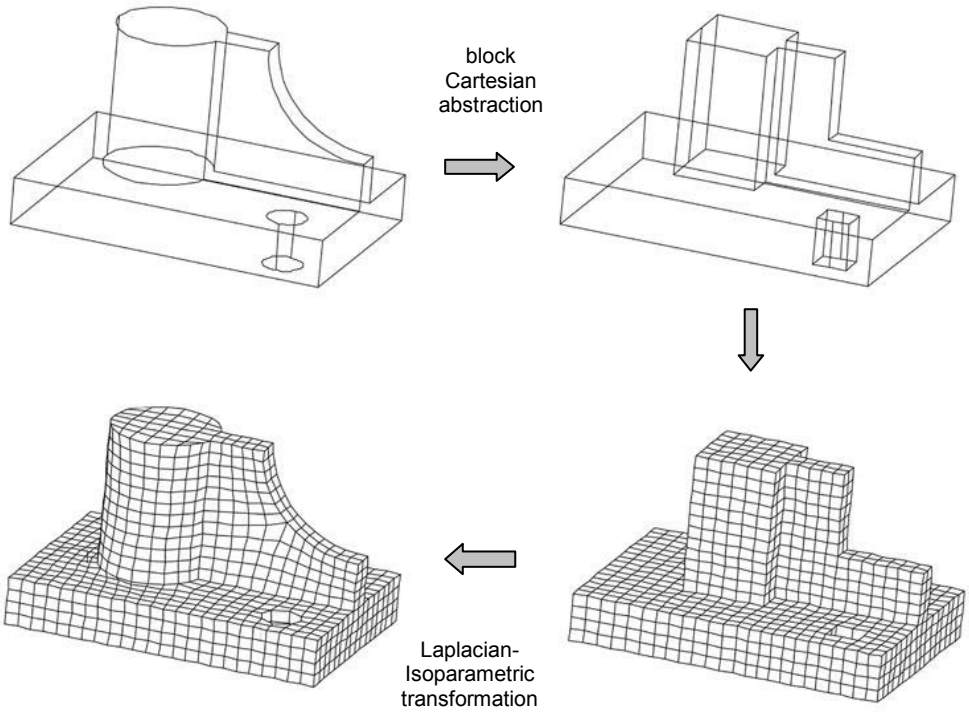

Fig. 11. Generation of structured mesh

5. REFERENCES

[1] Ang, P. Y. and Armstrong, C. G., Adaptive curvature-sensitive meshing of the medial axis, *Proceedings 10th International Meshing Roundtable,* Sandia National Laboratories, 2001, pp 155-165.

[2] Blacker, T. D. and Meyers, R. J., Seams and wedges in plastering: A 3-D hexahedral mesh generation algorithm, *Engng Comput*, Vol. 9, 1993, pp 83–93.

[3] Calvo, N. A. and Sergio, R. I., All-hexahedral element meshing: Generation of the dual mesh by recurrent subdivision, *Comput Methods Appl Mech Engng*, 2000, pp 371-378.

[4] Chiba, N., Nishigaki, I., Yamashita, Y., Takizawa, C. and Fujishiro, K., A flexible automatic hexahedral mesh generation by boundary-fit method, *Comput Methods Appl Mech Engng,* Vol. 161, 1998, pp 145-154.

[5] Hermann, L. R., Laplacian-isoparametric grid generation scheme, *J Engng Mech Div Proc Am Soc Civil,* Vol. 20, 1976, EM5.

[6] Jankovich, S. R., Benzley, S. E., Shepherd, J. F. and Mitchell, S. A., The Graft Tool: An all-hexahedral transition algorithm for creating a multi-directional swept volume mesh, *Proceedings 8th International Meshing Roundtable*, Sandia National Laboratories, 1999, pp 387-392.

[7] Li, T. S., McKeag, R. M. and Armstrong, C. G., Hexahedral meshing using mid-point subdivision and integer programming, *Comput Methods Appl Mech Engng,* Vol. 124, 1995, pp 171–193.

[8] Liu, S.-S. and Gadh, R., Automatic hexahedral mesh generation by recursive convex and swept volume decomposition, *Proceedings 6th International Meshing Roundtable*, Sandia National Laboratories, 1997, pp 217–231.

[9] Liu, S.-S. and Gadh, R., Basic LOgical Bulk shapes (BLOBs) for finite element hexahedral mesh generation, *Proceedings 5th International Meshing Roundtable*, Sandia National Laboratories, 1996, pp 291–306.

[10] Lu, Y., Gadh, R. and Tautges, T. J., Feature based hex meshing methodology: Feature recognition and volume decomposition, *Comp-Aid Des,* Vol. 33, No. 3, 2001, pp 221-232.

[11] Owen, S. J. and Saigal, S., H-Morph: An indirect approach to advancing front hex meshing, *Int J Numer Meth Engng,* Vol. 49, No. 1, 2000, pp 289-312.

[12] Price, M. A., Armstrong, C. G. and Sabin, M. S., Hexahedral mesh generation by medial axis subdivision: I. Solids with convex edges, *Int J Numer Meth Engng*, Vol. 38, 1995, pp 3335–3359.

[13] Price, M. A., Armstrong, C. G. and Sabin, M. S., Hexahedral mesh generation by medial axis subdivision: II. Solids with flat and concave edges, *Int J Numer Meth Engng*, Vol. 40, 1997, pp 111–136.

[14] Schneiders, R., A grid-based algorithm for the generation of hexahedral element meshes, *Engng Comput*, Vol. 12, 1996, pp 168–177.

[15] Schneiders, R., An algorithm for the generation of hexahedral element meshes based on an octree technique, *Proceedings 6th International Meshing Roundtable*, Sandia National Laboratories, 1997, pp 183–194.

[16] Su, Y., Lee, K.-H. and Senthil Kumar, A., Automatic hexahedral mesh generation for multi-domain composite models using a hybrid projective grid-based method, *Comp-Aid Des,* 2003, pp 203-215.

[17] Tautges, T. J., Timothy, J., Blacker, T. D. and Mitchell, S. A., The whisker weaving algorithm: A connectivity-based method for constructing all-hexahedral finite element meshes, *Int J Numer Meth Engng,* Vol. 39, 1996, pp 3327–3349.

[18] Thompson, J. F., Soni, B. K. and Weatheril, N. P., *Handbook of grid generation*, Boca Raton, FL: CRC Press, 1998.

[19] Walton, K. S., Benzley, S. E. and Shepherd, J., Sculpting: An improved inside-out scheme for all-hexahedral meshing, *Proceedings 11th International Meshing Roundtable*, Sandia National Laboratories, 2002, pp 153-160.

[20] Whitley, M., White, D., Benzley, S. and Blacker, T., Two and three-quarter dimensional meshing facilitators, *Engng Comput*, Vol. 12, 1996, pp 155-167.

[21] Yager, R. R. and Filev, D. P., *Essentials of fuzzy modeling and control*, John Wiley and Sons, Inc., 1994.