ELSEVIER

Finite Elements in Analysis and Design 28 (1997) 137–149

FINITE ELEMENTS IN ANALYSIS AND DESIGN

The spatial twist continuum: A connectivity based method for representing all-hexahedral finite element meshes[1]

Peter Murdoch[a], Steven Benzley[a,*], Ted Blacker[b], Scott A. Mitchell[c]

[a]*College of Engineering, Brigham Young University, 368 CB, Provo, UT 89604, USA*
[b]*Fluid Dynamics International, 500 Davis St., Suite 600, Evanston, IL 60201, USA*
[c]*Computer Mechanics and Visualization Department, Sandia National Laboratories, Albuquerque, NM 87185, USA*

Abstract

This paper introduces the *spatial twist continuum* (STC), a powerful extension of the dual of a hexahedral mesh. The STC captures the global connectivity constraints inherent in hexahedral meshing. We begin by describing the two-dimensional analog of the representation for quadrilateral meshes: The STC of a quadrilateral mesh is an arrangement of curves called *chords*. Chords pass through opposite quadrilateral edges and intersect at quadrilateral centroids. The power of the STC is displayed in the three-dimensional representation, where continuous surfaces called *twist planes* pass through layers of hexahedra. Pairs of twist planes intersect to form chords that pass through opposite faces of hexahedra. Triples of twist planes orthogonally intersect at the centroids of hexahedra. The continuity of the twist planes and chords, and how twist planes and chords twist through space, are the basis of the spatial twist continuum. © 1997 Elsevier Science B.V.

1. Introduction

The demand for fully automated meshing algorithms for general three-dimensional volumes has increased dramatically in recent years. Increased computer hardware speeds and improved finite element analysis and modeling software now make it possible to analyze larger problems than in the past. Meshing done using traditional techniques takes a large fraction of the total solution time,

* Corresponding author.

[1] This work was supported by the US Department of Energy under contract DE-AC04-76DO00789, by the Applied Mathematical Sciences program, US Department of Energy Research, and by Sandia National Laboratories Cooperative Research and Development Agreement SC93/01247 (USCAR Scientific Computing for Automotive Applications Partnership).

and can be a serious bottleneck in the overall analysis process. Thus, the need for automated techniques.

Automated meshing algorithms for general three-dimensional volumes usually yield either tetrahedral or hexahedral elements. Generation of a tetrahedral mesh is a less constrained problem than for a hexahedral mesh, and therefore easier to solve. Hence, tetrahedral meshing algorithms have historically received the most attention [1–5]. However, the demand for all-hexahedral meshing algorithms remains strong due to various analysis and design benefits associated with hexahedral meshes.

A number of techniques have been developed that are successful in meshing simple geometries [6–10]. These work by interpreting the input geometry in some global way. These methods break down when the input geometry is complicated, in part because interpreting the global geometry is difficult. A robust method based only on the local geometry is to overlay a regular grid and trim to the surface [11], but poorly shaped elements result near the geometric boundary. Plastering [12] is an advancing-front type of algorithm that takes a local approach to the geometry and shows promise for complicated inputs. However, the major difficulty with advancing-front type of algorithms is ensuring that hexahedra share faces, edges, and nodes in the correct way when fronts contact or are merged. This problem is more difficult for hexahedral meshes than for tetrahedral meshes: the global connectivity of the hexahedral mesh behind the front constrains how hexahedra on merging fronts may share faces, edges and nodes. Although some of these problems can be resolved using local corrections [13], it seems that some global connectivity information is necessary.

The *spatial twist continuum* (*STC*) is a powerful extension of the dual representation of a hexahedral mesh. The STC quantifies global connectivity constraints inherent in hexahedral meshes. This provides numerous insights into how to generate meshes. The STC is represented by a geometric arrangement of two-dimensional surfaces called *twist planes*. The intersection of three twist planes defines a hexahedral element of the mesh. These planes "twist" through the space of the three-dimensional continuum being meshed, thus, the name spatial twist continuum.

Whisker weaving [14] is an advancing-front type of algorithm that is based primarily on the STC. With the global connectivity information of the STC, advancing and merging fronts is a straightforward operation that results in little perturbation of the mesh behind the fronts. This whisker weaving technique shows a practical application of the concepts presented in this paper.

The remainder of this paper describes the basic structures of the STC. Section 2 introduces the STC using the dual of a two-dimensional quadrilateral mesh. Section 3 demonstrates the power of the STC in the full three-dimensional case. Section 3 also notes the important role that the two-dimensional STC of a surface mesh plays in determining the possible meshes of the interior volume. Section 4 provides some brief remarks about the possible application of the STC to hexahedral meshing algorithms. Conclusions follow in Section 5.

2. Dual of a quadrilateral mesh

A quadrilateral mesh can be uniquely described by its *dual* [14]. Fig. 1 (left) shows the dual for a simple quadrilateral mesh. The dual entities of dimension 0, 1 and 2 are called *STC centroids*, *STC edges*, and *STC 2-cells*. Only the connectivity of the dual is important here: the geometric

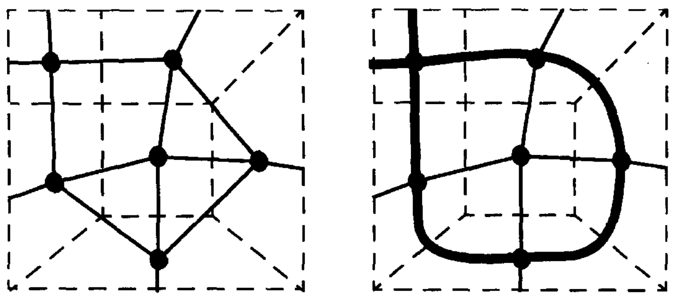
Fig. 1. Dual of a quadrilateral mesh (left). A chord of the STC (right).

Table 1
Correspondence of mesh entities to STC dual entities

Mesh entity	Dimension	STC entity	Dimension
Face	2	Centroid	0
Edge	1	Edge	1
Node	0	2-Cell	2

position of the centroids is taken to be somewhat arbitrary. What is important is which centroids are connected to each other by STC edges, the clockwise order of the STC edges around a centroid, and that STC edges do not cross each other. STC edges may be embedded as curves.

The connectivity of the dual can be constructed by first placing a centroid in each quadrilateral. Next, wherever two quadrilaterals share a mesh edge, an STC edge is added to join the corresponding centroids. To simplify the description of the *2D chords* below, the usual dual centroid at infinity and the dual centroids for input region holes are not included in the definition of the *mesh dual* used in this paper. However, STC edges that are dual to quadrilateral edges on the boundary of the input are constructed. One vertex of such an STC edge is a centroid, the other is a point on the input boundary. This construction is shown in Fig. 1 left.

Each centroid has four STC edges, since each quadrilateral has four sides. A 2-cell is the dual of a mesh node. The boundary of the 2-cell is the "polygon" of STC edges which are dual to the mesh edges containing the given mesh node, possibly together with a portion of the input boundary. Since the embedding is unimportant, the "polygon" may have curved sides, etc. The number of STC edges bounding a 2-cell is equal to the number of mesh edges attached to the corresponding node.

Note that a mesh entity of dimension d corresponds to a dual STC entity of dimension $2 - d$; see Table 1. For example, a mesh face (dimension 2) corresponds to a STC centroid (dimension 0).

2.1. 2D chords

Any type of mesh admits a dual [9], but an all-quadrilateral mesh dual has a unique property. Namely, portions of the dual can be logically grouped, allowing a "higher level" interpretation of the mesh and the global connectivity of its elements. Dual edges that correspond to opposite sides of a quadrilateral are grouped into a continuous curve called a chord. Fig. 1 (right) highlights one chord from the mesh in Fig. 1 (left). A chord actually represents a one-dimensional stack of elements. In effect, the quadrilateral mesh can be viewed as an intertwining of chords (or rows of elements) rather than as a collection of individual elements. The crossings formed by these intertwined chords are the elements, and how the chords intertwine dictates the validity and gross quality of the actual mesh. With this perspective in mind, the following properties of chords in a valid mesh can be identified:

(1) A chord that begins on the boundary of the input region must terminate on the boundary of the input region. This is equivalent to the fact that a quadrilateral mesh must have an even number of edges around its boundary [7].
(2) A chord that does not begin on the boundary must be a closed curve.
(3) Chords may cross each other multiple times, but such crossings may not be consecutive (i.e. if two chords cross at centroids 1 and 2, then on at least one of the chords centroids 1 and 2 must be non-consecutive). This forbids two quadrilaterals that share two edges.
(4) A chord is allowed to cross itself, but not in a local sense (i.e. between the first and second time a chord passes through a given centroid, there must be at least four other centroids).
(5) Each centroid is passed through exactly twice, either by two distinct chords or by one chord twice (e.g. three chords being coincident is forbidden). Also, chords are nowhere tangent.
(6) Each chord has at least one centroid.

In addition, STC 2-cells have the following properties:

(7) The portion of the object's boundary touched by a STC 2-cell is a connected set. Furthermore, a STC 2-cell touches at most one boundary node. This ensures that the object's boundary is correctly captured by the mesh.
(8) Two STC 2-cells have at most one STC edge in common. This prevents two mesh nodes from sharing two edges.

Any nonempty arrangement of curves inside an input region that satisfy these conditions defines a valid quadrilateral mesh. Pathological connectivity is ruled out, but beyond that there is no guarantee that the mesh will have well-shaped elements, even after smoothing. The chord and centroid representation provides a simple, but powerful, dual method for defining the basic connectivity rules for all quadrilateral meshes.

3. Hexahedral meshes and the 3D STC

Analogous to the two-dimensional case there are base entities and associated constructs that make up the STC for a three-dimensional all-hexahedral mesh. A hexahedral element has six quadrilateral faces and eight three-edged corners. The connectivity of an all hexahedral mesh requires that neighboring hexahedra share common faces. Thus, by the very nature of the hexahedral element and based solely on its requirement to share faces with neighboring elements, there exists a well-defined connectivity construction. As in the quadrilateral mesh case, this construction, the STC, is defined using a logical grouping of portions of the dual of the hexahedral mesh. First we will examine the hexahedral mesh dual and then define the chords and twist plane constructs which define the spatial twist continuum.

A hexahedral mesh is uniquely described by the STC entities introduced earlier (i.e. STC centroids, edges, and 2-cells) and a three-dimensional entity called a STC 3-cell. By definition, in three dimensions a mesh entity of dimension d corresponds to a dual entity of dimension $3 - d$; see Table 2. This dual can be constructed similar to the 2D case. A centroid is placed in each hexahedron. A STC dual edge is constructed by joining the centroids of elements that share a face. Thus, 6 edges emanate from each centroid. A 2-cell is the "polygon" of STC edges that are dual to the mesh faces containing a given mesh edge, possibly together with part of the input boundary. As in 2D, the 2-cell could have any number of sides, equal to the number of faces sharing the edge. Likewise, a 3-cell is the "polyhedron" of STC 2-cells dual to the mesh edges containing a given mesh node, possibly together with part of the input boundary. The 3-cell could have any number of 2-cell facets, equal to the number of edges sharing the node. Fig. 2 shows a simple hexahedral mesh composed of four elements and its dual, with 2-cells shown as shaded surfaces. In this example, the 3-cells are all cubic. As with the 2D quad mesh, any 3D mesh admits a dual, but a hexahedral mesh dual has some unique properties and constructs defined below.

Table 2
Correspondence of hex mesh entities to 3D STC dual entities

Mesh entity	Dimension	STC entity	Dimension
Hex element	3	Centroid	0
Face	2	Edge	1
Edge	1	2-Cell	2
Node	0	3-Cell	3

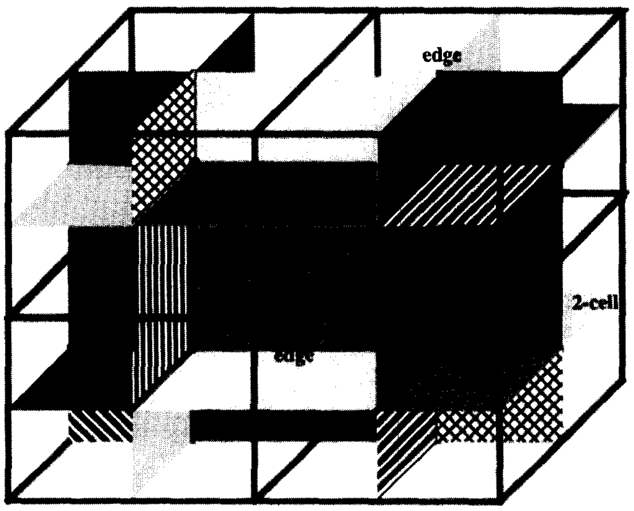

Fig. 2. A simple four hex mesh and its dual. The 2-cells devide volume inside the mesh into 3-cells, one for each exterior hex mode.

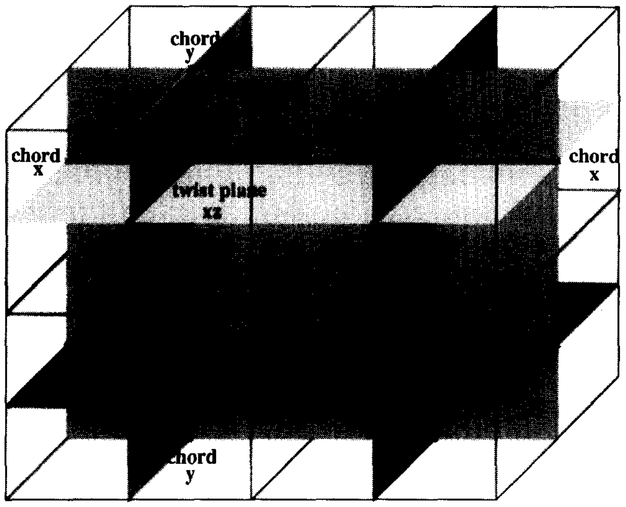

Fig. 3. The STC of a four hex mesh. Like colored surfaces are the same twist plane.

3.1. 3D chords

One such construct for the 3D STC is a *chord*. As in the 2D case, a 3D chord is a one-dimensional curve. A chord for a hexahedron is constructed by grouping the two STC edges that are dual to opposing faces. Fig. 3 labels the chords for the upper left hex. A hexahedron may be defined as the intersection of three chords, since the chords are continuous through the element. A chord also extends into the surrounding mesh as follows: if two hexahedra share a face, the chords through that face are grouped together. Thus, a chord represents a continuous stack of hexahedral elements. This stack either starts and stops on the input boundary, or closes on itself. In Fig. 3, chord x passes through the top two hexes and stops where the input boundary is met.

3.2. Twist planes

The real power of the STC can be seen through the two-dimensional construct termed a *twist plane*. A twist plane is formed by grouping the 2-cells which are logically perpendicular to a chord at a centroid, as shown in Fig. 3. In this figure, twist plane xy is logically perpendicular to chord z at centroid xyz. A twist plane within a single hexahedron will cut four faces, which form a cycle. Each hexahedron contains three such twist planes and the STC centroid is actually the intersection of these three as shown in Fig. 3 (centroid xyz can be viewed as the intersection of twist planes xy, yz, and xz). Note that a twist plane contains the two chords of the hexahedron that it is not perpendicular to. In Fig. 3, twist plane xy contains chord x and chord y. Chords remain on a twist plane, and thus a twist plane can alternatively be defined by the chords it contains.

As with the chord construct, twist planes extend continuously through the mesh as follows: If two hexahedra share a mesh edge, the twist planes through that edge are grouped together. Thus, a twist plane represents a continuous “layer” of hexahedral elements in the mesh. Fig. 3 shows how a twist plane (twist plane xy) passes through the four hexes of the example mesh.

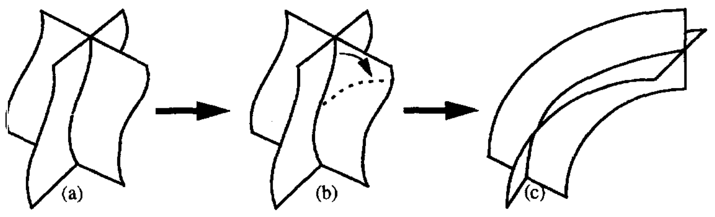

Fig. 4. How the planes twist as a chord changes position.

3.3. *Symbiosis of chords and twist planes*

From the above definitions of chords and twist planes one can see that the definitions are coupled. A chord may be defined as the intersection of two twist planes. Alternatively, a twist plane can be defined by the chords that it contains (the exact geometric embedding of the twist plane is not unique, but its combinatorial structure is completely specified). Hexahedra can be derived from either description, either as the intersection of three twist planes, or as the intersection of three chords. Chords, however, cannot be used independently of twist planes to define a hexahedral mesh. Although an arrangement of surfaces may be quite general and still define the twist planes of a valid STC, an arrangement of curves in 3D must be highly structured in order to be able to define them as chords and extract twist planes.

Taking the view that twist planes define chords, the position of the twist planes in space defines the position of the chords of intersection. In alternate view that chords define twist planes, the location of the twist planes is defined by some smooth interpolation of the location of the chords. Hence, if a chord changes position, but stays on one twist plane (see Fig. 4b), then the other twist plane containing it must change position ("twist") to match it (see Fig. 4c). These relationships are shown in Fig. 4. The key element is that a chord stays on both twist planes

3.4. *Surface loop chords*

The intersection of a twist plane with the boundary of the object being modeled creates a surface loop chord. This loop chord is continuous around the boundary. The loop chord differs from a three-dimensional chord since it lies on the faces of the surface and does not pass through a hexahedral centroid. A loop chord can also be thought of as a chord of the two-dimensional STC of the surface mesh. Fig. 5 shows loop chords as thick lines. An important feature of a surface loop is that it always closes upon itself (since the object's surface is closed). There are no limitations to the number of surface loops associated with a given twist plane: each connected component of the intersection of the twist plane with the object boundary creates a separate surface loop. A twist plane with no surface loops corresponds to a closed surface (e.g. a topological sphere) entirely inside the mesh. A cylindrical twist plane may form two surface loops, etc.

The surface loop chords play an important role in any advancing-front type of algorithm. If a surface mesh is given, then the intersections of twist planes with the geometric surface are already determined: However, elements are added inside the volume, the resulting twist planes must coincide with the predetermined surface loops on the geometric boundary, or the surface mesh

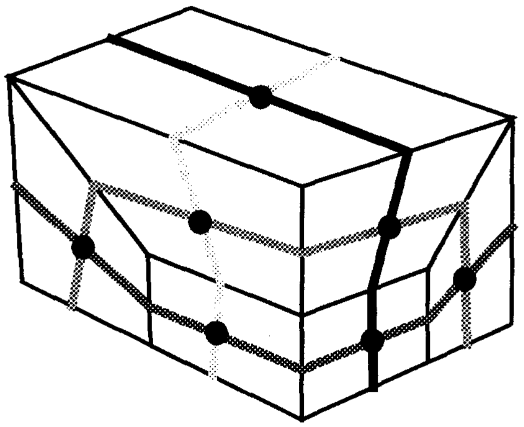
Fig. 5. Surface loops for an example mesh.

must be changed. The same applies to the two-dimensional surface loops that exist on any meshing front: as fronts merge the twist planes must respect the loops, or the loops must be changed by modifying the mesh behind the front.

3.5. Connectivity configurations

To gain some intuition about how twist planes can interact within a mesh, several examples of twist plane constructs (or states) have been identified. The basic constructs defined here include *parallelism*, the *orthogonal connectivity*, the *fold* and the *corner*.

Two twist planes are said to be (locally) *parallel* if they pass through hexes that share a face but do not intersect each other. The layers of hexahedra formed by parallel twist planes lie directly on top of each other. This parallelism may continue throughout the mesh, or be disrupted by the folds and corners that are defined below.

The *orthogonal connectivity* corresponds to a regular mesh: each internal node is connected to 8 elements. In this state, two nearby twist planes are either parallel or perpendicular to each other throughout the entire construct. For example, the orthogonality state and the three-dimensional hexagonal mesh it defines is shown in Fig. 6. A *fold* is defined when two locally parallel twist planes change to become perpendicular; see Fig. 7. A fold results in a triangular STC 2-cell on a twist plane orthogonal to the first two. This triangular structure extends through the mesh on other orthogonal twist planes, until another fold or corner is reached. A *corner* occurs when a folded twist plane folds again in a perpendicular direction. Fig. 8 and 9 show a STC corner and the related hexahedral elements.

3.6. Meshing algorithms derived from the STC

The STC is a general description of constraints associated with a hexahedral mesh. As such, it may prove the basis of numerous algorithms, and perhaps existing algorithms could be recast in the

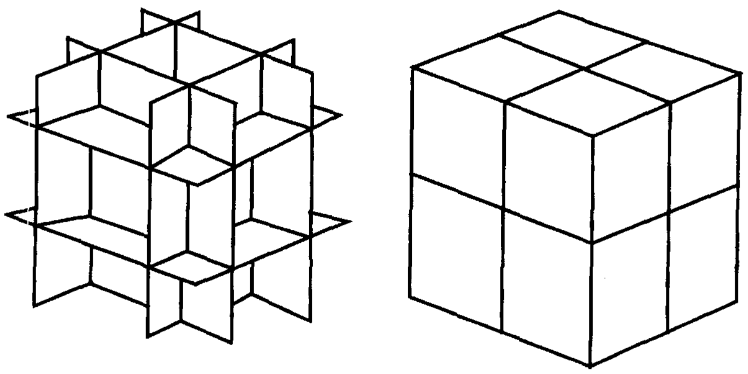
Fig. 6. The orthogonality state (left) and the mesh it defines (right).

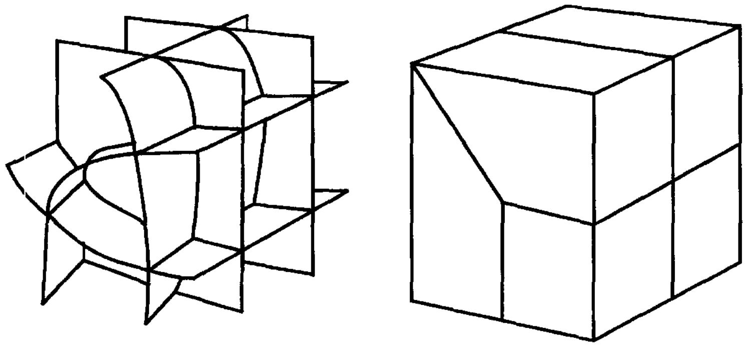
Fig. 7. The STC fold (left) and the mesh it defines (right).

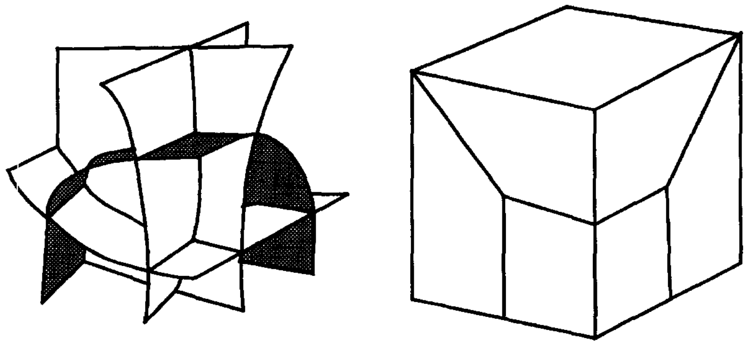
Fig. 8. The front view of a STC corner (left) and the mesh it defines (right).

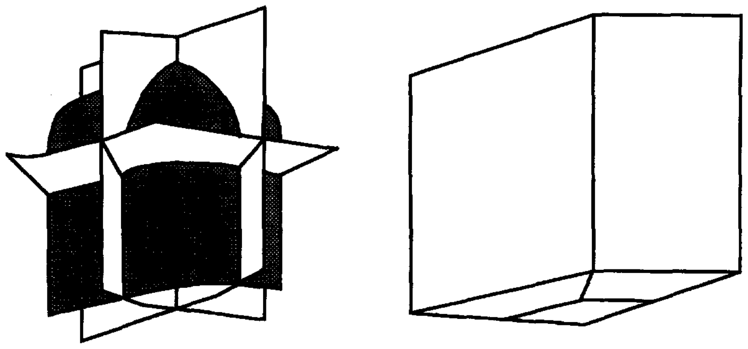
Fig. 9. The rear view of a STC corner (left) and the mesh it defines (right).

terminology of the STC. Two algorithms described here are a *Twist Plane Insertion* procedure, a process conceived directly from STC definitions and *whisker weaving*, a somewhat abstract approach which is explained in detail by Tautges et al. [14]. Twist plane insertion can be used to generate an all hexahedral mesh when no existing surface mesh exists on the solid being meshed. Whisker weaving can be used when generating an all hexahedral mesh using an existing all quadrilateral surface mesh.

The *Twist Plane Insertion* algorithm proceeds as follows. Twist planes are inserted one at a time into a 3D volume without the aid of a surface mesh. Correct connectivity is insured since each twist plane is inserted totally. Care needs to be taken to match the STC to the object's input faces and vertices. This process is easily visualized by considering Fig. 6 (left), 7 (left) and 8 (left). Six twist planes have been inserted into a box as depicted in Fig. 6, five twist planes have been inserted into a similar box as depicted in Fig. 7 and four twist planes have been inserted as depicted in Fig. 8. The geometric features of the twist planes uniquely define all faces and nodes of the all hexahedral mesh via the dual. The procedure could be enhanced by defining a local density function to dictate the distance between twist planes. This algorithm is geometrically complex because the insertion is, of necessity, performed in general 3-space.

The *whisker weaving* algorithm takes the following approach. First, given an arbitrary solid, an arbitrary quadrilateral mesh of its surface is created. This quadrilateral surface mesh is then sorted into 2D surface loops. Each loop contains information about which other loops it crosses on the surface. These crossings indicate the beginning/ending locations of chords on the twist plane containing the loop. These crossings, or "whiskers," are then extended and intersected locally (woven). Chords are also joined: a joined chord is complete in the sense that it starts and ends on the surface mesh. Eventually, every whisker of every loop is complete and the algorithm terminates with a valid 3D STC. The main power of the algorithm is that the 3D meshing problem is reduced to a series of local problems on interdependent 2D *sheets* (twist planes flattened/projected into a 2D space).

An example of this algorithm can be explained by beginning with a surface mesh as given in Fig. 8 (right). Surface loops are formulated as shown in Fig. 10 with each loop depicted with a different line type (i.e. solid, dashed, etc.). The surface loops, with an initial whisker labeled as a surface quadrilateral, are shown in Fig. 11. Note that each loop defines a separate twist plane.

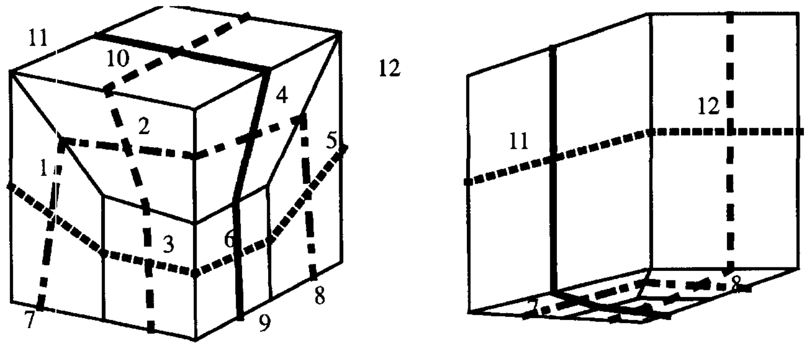

Fig. 10. Surface mesh showing STC loops – front view (left), rear view (right).

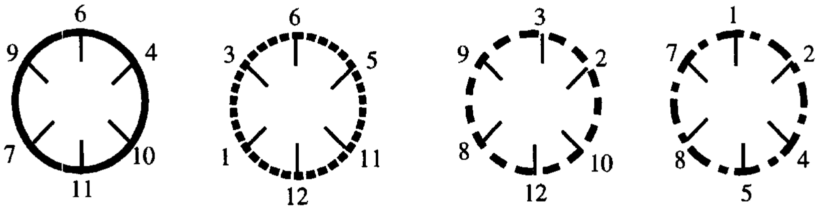

Fig. 11. Loops depicted as 2D circles with whiskers labeled as surface quad.

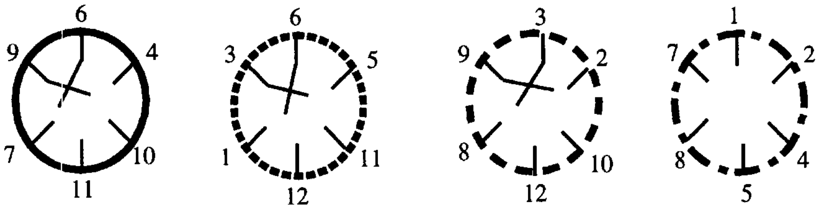

Fig. 12. Whiskers 9, 6 and 3 woven to form element 1.

The weaving process proceeds following the criteria that three chords, A, B, and C, are found such that A is adjacent to B, B is adjacent to C, and C is adjacent to A. Chords A and B lie on one twist plane, chords B and C lie on another twist plane, and chords C and A lie on yet another twist plane. The chords meeting this criteria are crossed forming an element. These steps are shown for a single element in Fig. 12 and 13. The process continues as shown in Fig. 14 and 15 until all the elements are formed. The whiskers are ultimately joined to complete the process, thus making chords continuous on any given twist plane.

As shown by the interdependency of surface loops and twist planes, a surface mesh imposes certain global constraints on how the volume may be filled with hexahedra. Any three-dimensional hexahedral meshing algorithm that starts with a surface mesh must (implicitly) address these

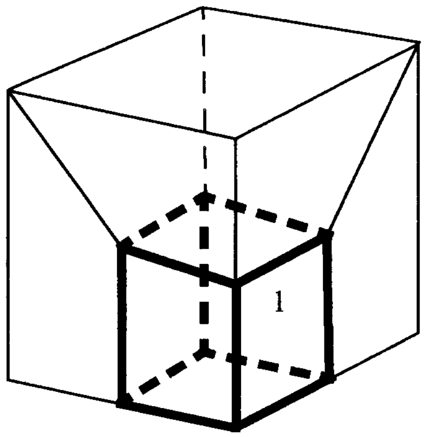

Fig. 13. Relative position of element 1.

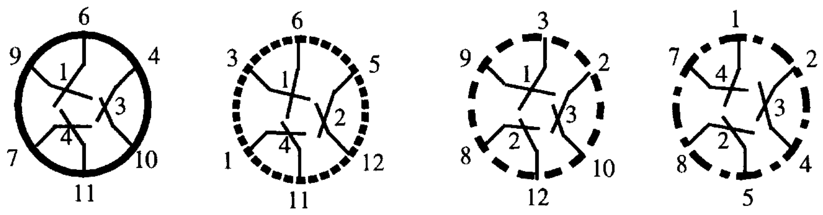

Fig. 14. Whiskers 5, 8 and 12 woven to form element 2, Whiskers 2, 4 and 10 woven to form element 3, and Whiskers 1, 7 and 11 woven to form element 4.

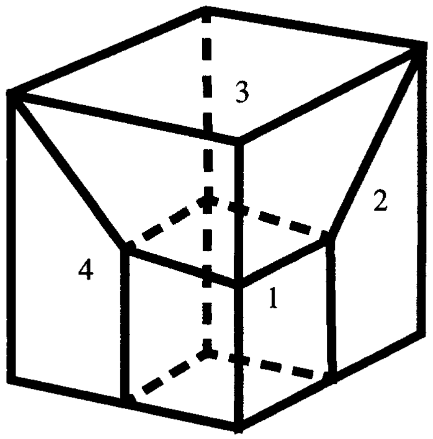

Fig. 15. Relative positions of elements 1, 2, 3, and 4.

constraints, whether or not the STC is used explicitly. The STC gives a natural representation of these constraints. Note that the input geometry imposes other constraints (due to desired element shape quality) that are not addressed by the STC itself.

4. Conclusions

The spatial twist continuum is a new way of representing a hexahedral mesh that captures inherent global connectivity structures. The STC identifies continuous twist planes, which are layers of hexahedra in the mesh. Twist planes intersect pairwise to form chords, which are stacks of elements within a layer. By definition, a chord cannot leave the twist planes that define it. These structures can guide mesh generation: given a partial mesh, it is easy to identify constraints on what the rest of the mesh must be like. This forms the foundation of the Whisker weaving algorithm [14]. The STC provides a rich base from which to derive other meshing algorithms as well.

5. References

[1] J.C. Cavindish, D.A. Field, W.H. Frey, an approach to automatic three-dimensional finite element mesh generation, Int. J. Numer. Methods Eng. 21 (1985) 329–347.

[2] T.J. Baker, Automatic mesh generation for complex three-dimensional regions, using a constrained delaunay triangulation, Eng. Comput. 5 (1989) 1161–175.

[3] P.L. George, F. Hecht, M.G. Vallet, Creation of internal points in Voronois type method. control adaptation, Adv. Eng. Soft. 13 (5/6) (1991) 303–312.

[4] Y.J. Jung, K. Lee, Tetrahedral-based octree encoding for automatic mesh generation, Comput. Aided Des. 25 (3) (1993) 141–153.

[5] N.P. Weatherill, O. Hassan, efficient three-dimensional delaunay triangulation with automatic point creation and imposed boundary constraints, Int. J. Numer. Methods Eng. 37 (1994) 2005–2039.

[6] W.A. Cook, W.R. Oakes, Mapping methods for generating three-dimensional meshes, Comput. Mech. Eng. (1982) 67–72.

[7] M.B. Stephenson, T.D. Blacker, Using conjoint meshing primitives to generate quadrilateral and hexahedral elements in irregular regions, Proc. ASME Computations in Engineering Conf. 1989.

[8] T.K.H. Tam, C.G. Armstrong, Finite element mesh control by integer programming, Int. J. Numer. Methods Eng. 36 (1993) 2581–2605.

[9] T.D. Blacker, M.B. Stephenson, J.L. Mitchner, L.R. Phillips, Y.T. Lin Automated quadrilateral mesh generation: a knowledge system approach, ASME Paper NO. 88-WA/CIE-4, 1988.

[10] A.P. Gilkey, G.D. Sjaardema, GEN3D: A GENESIS database 2D to 3D transformation program, SAND89-0485, Sandia National Laboratories, Albuquerque, New Mexico, March 1989.

[11] M.S. Shephard, M.K. Georges, Automatic three-dimensional mesh generation by the finite octree technique, Int. J. Numer. Methods Eng. 32 (1991) 709–749.

[12] M.B. Stephenson, S.A. Canann, T.D. Blacker, R.J. Meyers, Plastering progress report I, SAND89-2192, Sandia National Laboratories, Albuquerque, New Mexico, 1992.

[13] T.D. Blacker, M.B. Stephenson, Seams and wedges in plastering: a 3-D hexahedral mesh generation algorithm, Eng. Comput. 9 (1993) 83–93.

[14] T.J. Tautges, T.D. Blacker, S.A. Mitchell, Whisker weaving: a connectivity based method for representing and constructing all-hexahedral finite element meshes, Int. J. Numer. Methods Eng. 39(1995) 3327–3349.

[15] F.P. Preparata, M.I. Shamos, Computational Geometry an Introduction, Springer, New York, 1985, pp. 24–26.