AUTOMATIC 2D ABSTRACTION AND HEXAHEDRAL MESHING BY SORTING A DELAUNAY MESH

Reza Taghavi

SIW Corp., Saint Paul, MN 55114 USA. taghavi@siw.com

ABSTRACT

A framework is presented for the automatic 2D abstraction, decomposition and block-structured hexahedral meshing of a volume defined by a closed triangular mesh. No proofs, necessary or sufficient conditions are provided. Instead, constructive definitions and instructions describe a procedure for automatically building a 2D abstraction and an all-hexahedral mesh. First, a constrained Delaunay tetrahedralization of the interior of the object is built and its tetras are partitioned into groups based on their internal edge and face counts. Through labeling, organized structures are observed where 64-tetras (6 internal edges and 4 internal faces) bookend stacks of 54-tetras (5 internal edges and 4 internal faces). Stacks of 54-tetras, connected four-to-each 64-tetra, form a network of primary prism that populate the entire object near and along its edges. The exposed edges of the primary prism form rails that partition the input mesh into two categories of patches. Source and target patch pairs that define extrudable sub-volumes, and edge patches that define secondary prism. Degeneracies are remedied through affine transformations and an efficient local mesh manipulation process. The mid-mesh of the extrusion pairs is then computed and extended in order to obtain a 2D manifold that is a far simpler, albeit incomplete, 2D abstraction of the object than a customary mid-surface/medial axis representation. The result is a partition of the volume into extrudable, prismatic and tetrahedral blocks that trivially leads to a single block-structured hexahedral mesh.

Keywords: Block-decomposition, 2D abstraction, mesh generation, tetrahedra, hexahedra, Delaunay, extrusion

1. INTRODUCTION

Block-structured hexahedral meshes are of interest for both their geometrical and computational qualities. If available, a block-structured hexahedral mesh is often the preferred choice for non-linear plasticity and CFD analysis due to its ability, among other things, to ensure near-90º element corner angles while maintaining excellent orientation with respect to boundaries. Furthermore, in thin parts, commonplace in aerostructures, electronics and generative design, the density of hexahedral meshes are, in general, less affected by element aspect ratio than in the case of tetrahedral meshes. As a result, block-structured hex meshes of such parts may feature up to two orders of magnitude fewer elements than equivalent tetrahedral meshes, thus resulting in substantial computational gains.

2. PREVIOUS WORK

Attention to the special role played by 3-triangles (Delaunay triangles featuring 3 internal edges) and 64-tetras (Delaunay tetras featuring 6 internal edge and 4 internal faces) in identifying special morphologies in closed 2D and 3D regions was first reported in the context of approximating medial axes and mid-surfaces using meshes [1] [2] [3]. But while only in 2D may one approximate the "inside" mid-object of a closed loop as the locus of the circumcenters of its 3- and 2-triangles, in 3D, the circumcenters of 64- and 54-tetras may not be used as an approximation of the mid-surface [4]. In his LayTracks3D approach to meshing general solid, [5] identifies entire prismatic blocks along the sharp edges of objects and calls them tracks and rails. He then goes on to create an all-hexahedral mesh. We will use a similar concept in this work.

3. NOMENCLATURE AND CONSTRUCTION

We present a method for automatically building a 2D abstraction of an object that is simpler than a mid-surface/medial axis, with the final goal of generating a block-structured hexahedral mesh. Our approach is based on the observation that a constrained Delaunay tetrahedralization of a closed and finely meshed triangular surface mesh, without the addition of any internal nodes, yields only a handful of a certain type of tetras that we call 64-tetras (meaning, 6 internal edges and 4 internal faces). We speculate that the number and placement of the 64-tetras is robust, invariant and characterizes the morphology of an object. Without providing any proof nor any necessary or sufficient conditions, we suggest that the exposed edges of the 54-tetras (5 internal edges and 4 internal faces) partition the triangular surface of the mesh into patch pairs delineating extrudable volumes. We then use the patch pairs and their bordering primary prisms to compute a 2D abstraction of the object that leads us to a final hexahedral block-structured mesh.

3.1. Definition of 64- and 54-tetras

Figure 1 shows a finely triangulated surface mesh representing a test object that we will use throughout this presentation. This object is obtained by boring two partially offset and partially intersecting cylinders through a brick.

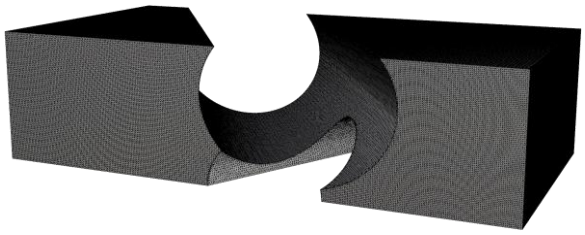

Figure 1: Surface mesh of test object

A fine triangulation of the test object followed by Delaunay tetrahedralization produces 236,242 tetras, only 15 of which have 6 internal edges and 4 internal faces and thus qualify as 64-tetras. For the sake of definition, in the context of a tetrahedral mesh, an internal face is defined as one that is shared by 2 tetras, and an internal edge is defined as one that borders only internal faces. We call such tetrahedra 64-tetras. 64-tetras play an important role in the present method. Figure 2 represents all the 64-tetras generated in our test. We speculate, without proof, that for a "fine enough" initial triangular mesh, the number and general placement of 64-tetras remain unaffected by the level of refinement.

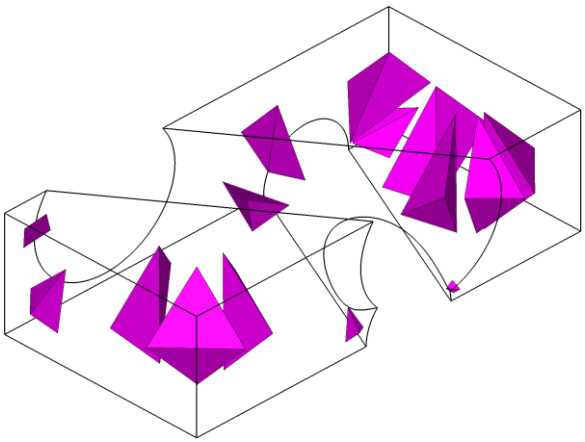

Figure 2: 64-tetras

A practical way of differentiating tetras is by labeling them with a decimal number where the tens represent the number of internal edges and the units represent the number of internal faces of that tetra. In this fashion, a 64 tetra will be labelled as 64 for its 6 internal edges and 4 internal faces. Let's now consider 54-tetras. In Figure 3 they appear organized as prismatic structures (stacks of tetrahedra) throughout the mesh. 54-tetras have 4 internal faces but only 5 internal edges, thus leaving one exposed edge on the surface. 54-tetras appear as stacks of tetras bookended by 64-tetras. In Figure 3, 64-tetras are rendered as solids while 54-tetras are rendered as gray wire meshes.

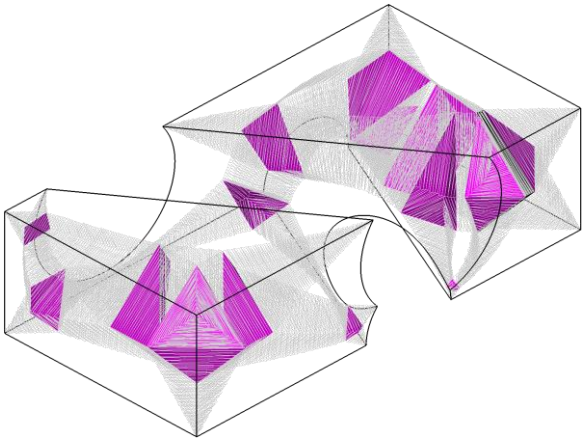

Figure 3: 64- and 54-tetras

3.2. Primary prisms

By sorting 54-tetras into individual stacks starting and ending at 64-tetras, in the present example, 38 individual sets are found, organized as jagged sets of connected tetras linking the triangular faces of pairs of opposite 64-tetras. We refer to these sets as primary prisms because they represent a volume homeomorphic to a prism. In Figure 4 different colors are assigned to each primary prism to help distinguish them from one another. Note that 64-tetras are not visible in this solid rendering as all 4 faces of 64-tetras are covered by primary prisms.

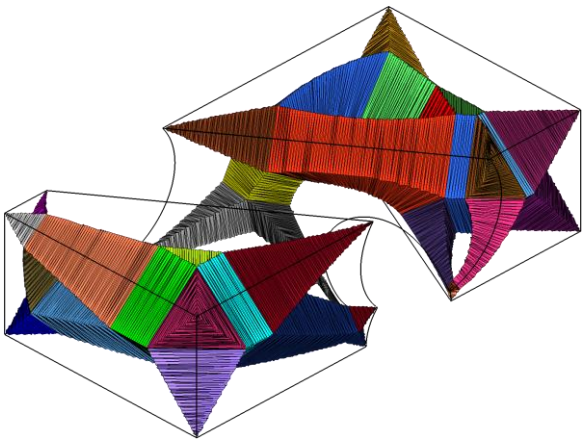

Figure 4: Colored primary prisms composed of connected 54-tetras linking opposite 64-tetras (not visible)

3.3. Regular, loop and degenerate primary prisms

Primary prisms are made of 54-tetras. In general, primary prisms originate and end at the face of a 64-tetra but there are exceptions: loop primary prisms and degenerate primary prisms.

1. Loop primary prisms which occur around holes are composed solely of 54-tetras. Figure 5 shows a closeup view of two loop primary prisms that occur in the Landing Gear example depicted in Figure 39.
2. Degenerate primary prisms originate at a 64-tetras but terminate at 34-tetras and are generally located near the corners of the object. Figure 6 shows a degenerate primary prism in the present example.

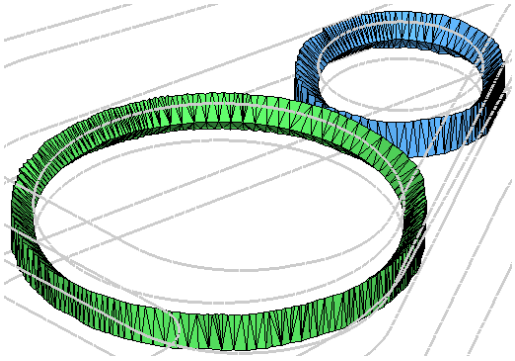

Figure 5: Two loop primary prisms produced in the Landing Gear example. Features of the model appear in gray

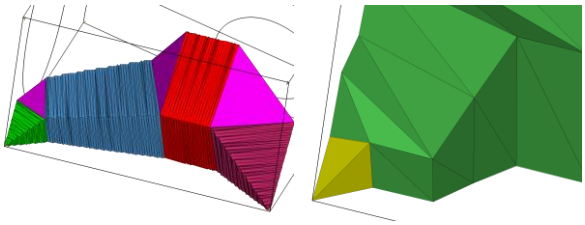

Figure 6: A degenerate primary prism (left) and its closeup view (right).

By definition, a 54-tetra possesses one exposed edge on the surface of the mesh. The exposed edges of the 54-tetras in a primary prism form 3 jagged curves on the surface mesh. We call these curves rails. Figure 7 shows one such prism and its 3 rails, as well as the pair of 64-tetras that bookend it.

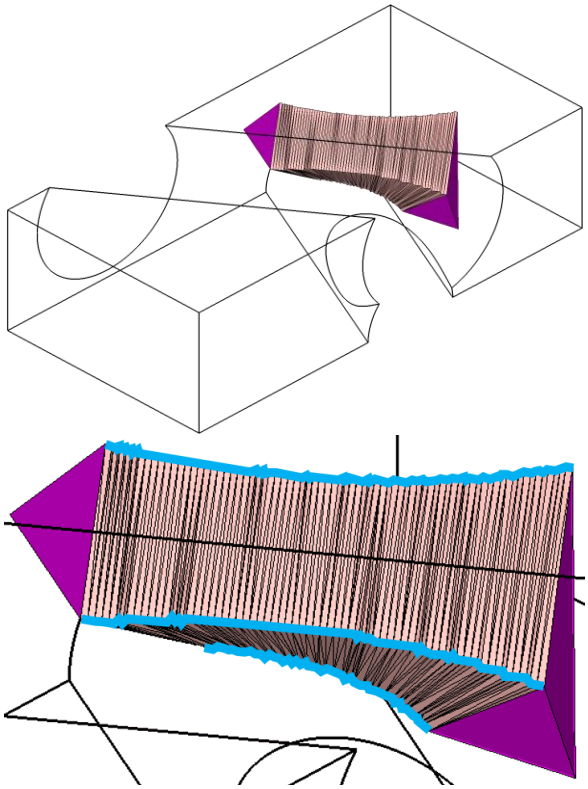

Figure 7: A single primary prism bookended by two 64-tetras (top) and its 3 rails highlighted in light blue (bottom)

3.4. Partition of the surface into extrusion and edge patches

Rails are connected sets of exposed 54-tetra edges. Again, without proof, we suggest that rails for closed loops on the surface mesh and as such partition the surface mesh into a number surface patches. In the present example, 40 patches are created as shown in Figure 8.

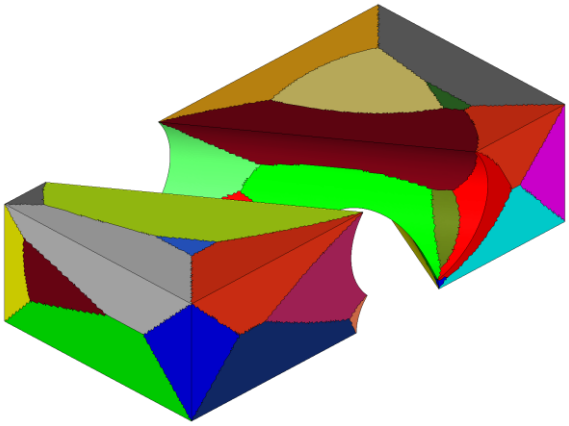

Figure 8: Surface patches resulting from the partition of the surface faces by rails

Surface patches may be further classified into two categories: extrusion patches and edge patches. Extrusion patches are formed when loops of primary prisms carve out two opposing patches from the surface mesh. Two such loops are shown in Figure 9: Two primary prism loops (top) and their associated extrusion patch pairs are shown in the bottom image.

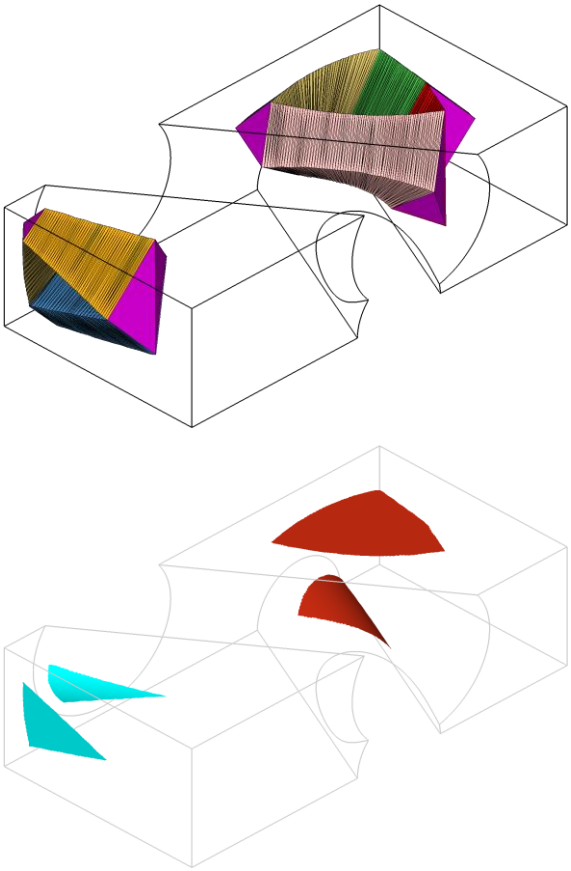

Figure 9: Two primary prism loops (top) and their associated extrusion patch pairs (bottom)

Extrusion patches always appear in pairs. Figure 10 shows the 7 pairs of extrusion patches produced in the present example. For the sake of clarity, patch pairs are colored identically.

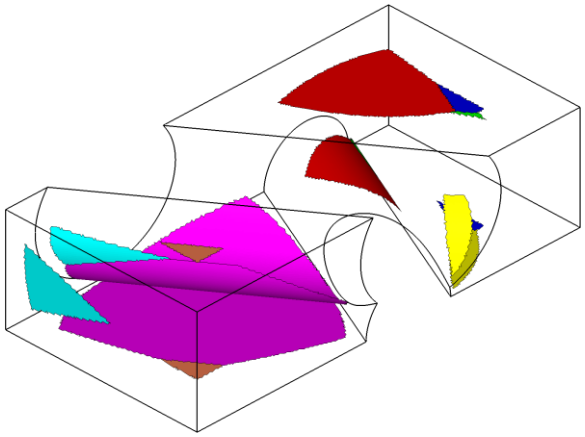

Figure 10: Extrusion patches

Extrusion patch pairs, along with the lateral walls of the primary prisms surrounding them carve out swaths of extrudable volumes from the object. Figure 11 shows the two earlier patch pairs and their associated sidewalls extracted from the primary prism loops that surround them.

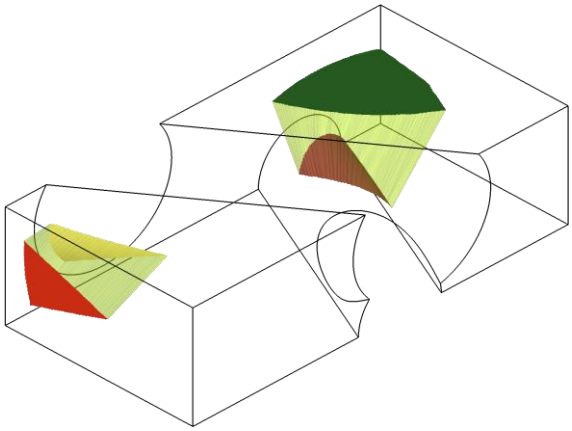

Figure 11: Two extrusion pairs and the sidewalls surrounding them

The remaining non-extrusion patches are called edge patches. Figure 12 shows the edge patches in the current example. Masked extrusion patches appear as holes.

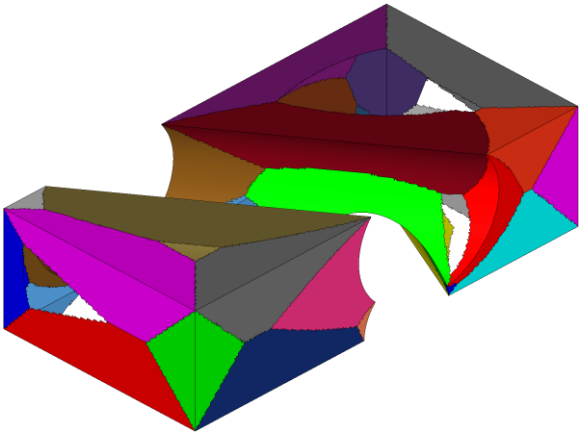

Figure 12: Edge patches

3.5. Edge patches and secondary prisms

Secondary prisms are a new type of prism enclosed by edge patches and the internal walls of one or more primary prisms. Remembering that an edge patch is one that is not an extrusion patch, the two edge patches (out of a total of 26) shown in Figure 13. The boundaries of these edge patches are composed of primary prism rails.

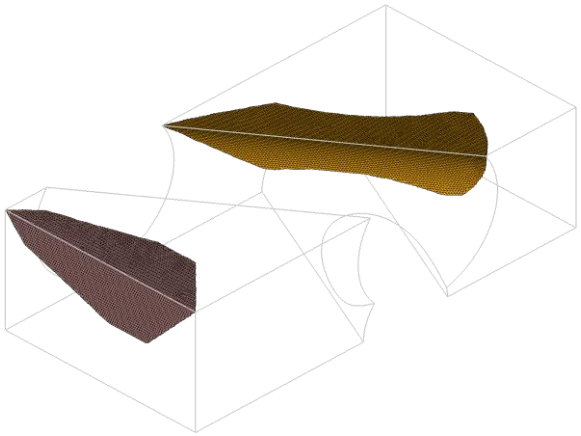

Figure 13: Two edge patches

Consider all primary prism that shares two rails with these two edge patches. These prisms are shown in Figure 14 where, for the sake of clarity, the edge patches are not shown .

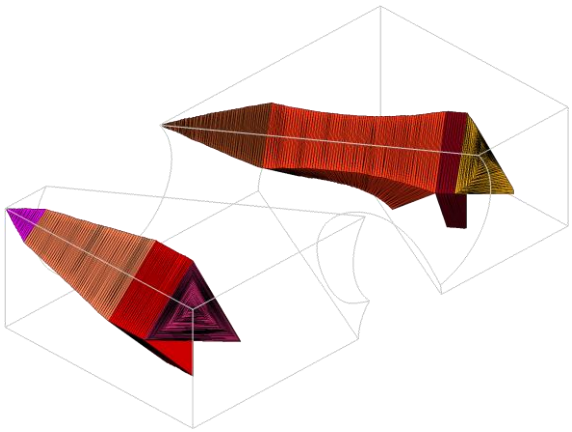

Figure 14: Primary prisms that share 2 rails with the two edge patches

Finally, consider the sidewalls of these primary prisms that face the edge patches which, along with the edge patches, form two closed volumes. these volumes are called secondary prisms and are shown in Figure 15 from a different angle for the sake of clarity. Like primary prisms, secondary prisms define precise volumes (blocks) within the object but unlike primary prisms, secondary prisms are not made of 54-tetras. Instead (without further development) they are mostly made of 33- and 44-tetras.

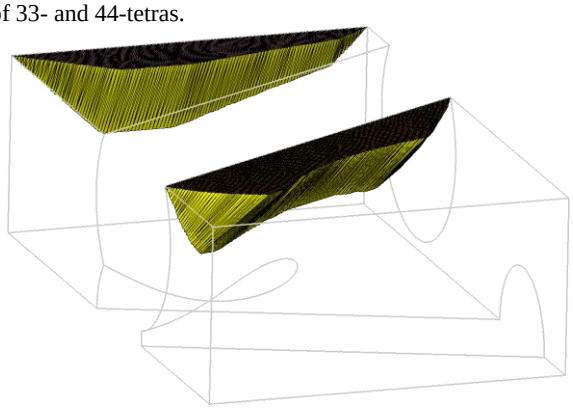

Figure 15: Edge patches and the primary prism sidewalls that define secondary prisms

3.6. Calculation of the mid-mesh of extrusion patch pairs

Without any proof, we assert that extrusion patch pairs are connected through 33- and 44-tetras only. 33-tetras have one exposed face and 3 internal edges. 44-tetras have no exposed faces but 2 exposed edges, in general opposed to one another in the tetra and belonging to opposite patches. There is no easy way to represent the 33- and 44-tetras that join each patch pair. Figure 16 shows the leftmost extrusion patch pair depicted earlier in Figure 10, with a section cut through the 33- and 44-tetras that connect the patch pair. It can be noted that 33- (in blue) and 44-tetras (in yellow) form, by far, the majority of the tetras in a Delaunay mesh with no internal nodes obtained from a “fine enough” surface mesh.

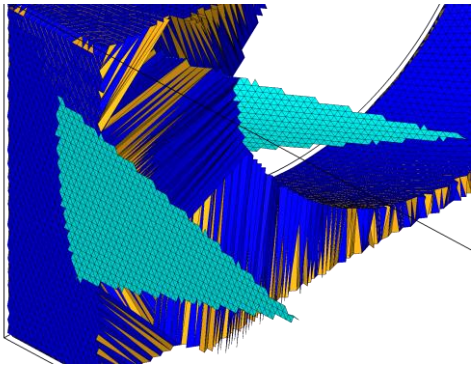

Figure 16: Cut through 33-and 44-tetras

The 33- and 44-tetras that connect the 2 patches can be easily identified as the set of 33- and 44-tetras that have either a surface face and its corresponding opposed node or, opposed edges, on each patch of the pair. Consequently, we can accurately construct the mid-mesh of such extrusion patch pairs. The process is described later in detail. Figure 17 represents a 2D abstraction of the example model composed of the 7 smoothed and "extended" (details below) mid-meshes of the 7 extrusion patch pairs depicted earlier in Figure 10.

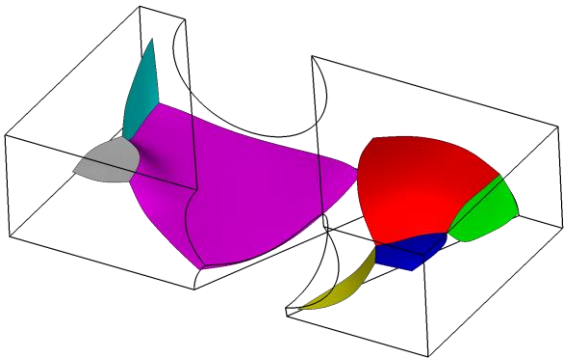

Figure 17: Smoothed 2D abstraction (extended mid-meshes) of extrusion pairs

Smoothed 2D abstractions may now be re-meshed as quad meshes using any quad meshing meshed such as advancing front or paving (Figure 18).

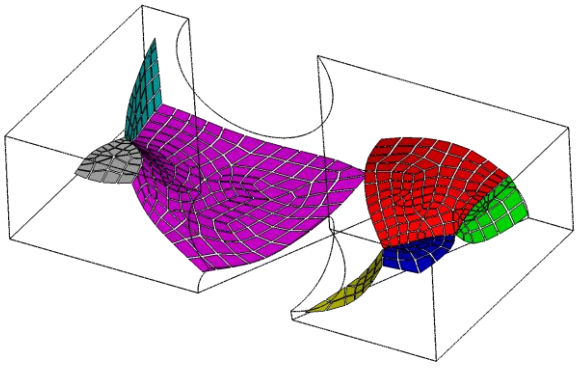

Figure 18: Quad-meshed mid-meshes

4. HEX MESHING

Without proof, we suggest that the volume of the object, and more specifically the tetras composing the Delaunay mesh, are partitioned into 4 sets:

1. 64-tetras
2. Primary prisms, composed of 54- and a few 34 tetras
3. Extrudable sub-volumes, composed of 33- and 44-tetras
4. Secondary prisms, composed 33- and 44-tetras almost exclusively

These sets are all primitives and as such can be treated as blocks and meshed and combined as a compatible all-hexahedral mesh.

4.1. Hex meshing of extrusions

The vertices of quad-meshed 2D abstractions may be extruded along the internal edges of the 33- and 44-tetras that connect each patch pair. It can be noted that wherever two extruded mesh blocks come into contact, their grids are compatible by construction (detailed later) because their abstracted patches are adjacent. Figure 19 shows the hex meshes resulting from all 7 extrusions and subdivided in 2 along the extrusion.

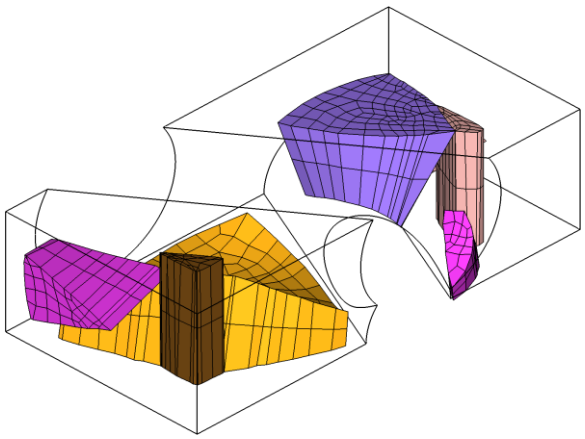

Figure 19: Extruded mid-mesh quads

4.2. Hex Meshing of primary prism blocks and blocks defined by 64-tetras (64-blocks)

The blocks defined by primary prisms, referred to hereafter as primary prism blocks are meshed as hexes by first subdividing them longitudinally, then azimuthally, by splitting their triangular cross-section into 3 quads. The boundary nodes of the quad meshed abstractions dictate the slicing locations of the primary prism blocks in their longitudinal direction thus ensuring that the extruded mesh and the hex-meshed prism blocks share compatible nodes. 64-tetras and degenerate primary prism blocks, which are also homeomorphic to tetras, are tetrahedral blocks and are split each into 4 hexas each (Figure 20).

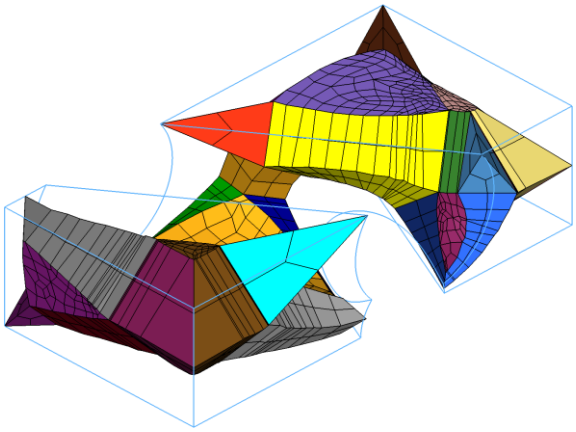

Figure 20: Hex-meshed extrusion, primary prisms and 64-blocks

4.3. Hex meshing of secondary prism blocks

Secondary prisms are defined by edge patches and the sidewalls of primary prisms. Therefore, the blocks they represented can be subdivided following the longitudinal subdivision of the primary prism blocks whose sidewall they share. Figure 21 shows the completed mesh composed of the extrusions, split in 2 in the direction of the extrusion, matching the primary and secondary prisms split into hexas and the 64-tetras also split into 4 hexas.

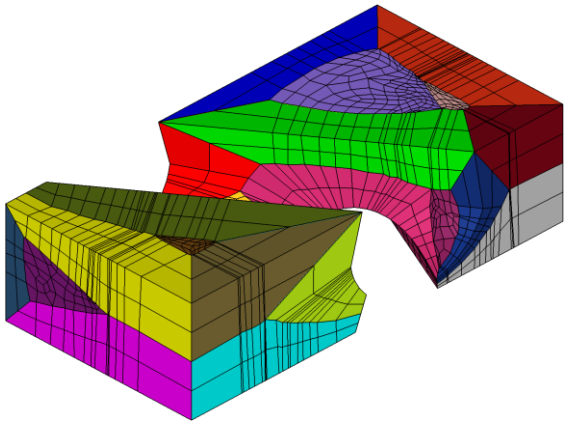

Figure 21: Final lock-structured hexahedral mesh

5. DETAILS

The details of the construction of the 2D abstraction which ensures the compatibility of the extruded meshes, the data structure needed to maintain a link between the tetra and hexa meshes and other fine points are addressed in this section.

5.1. Construction of a 2D abstraction that ensures the compatibility of the extruded hex meshes

Extrusion source and target patch pairs connect 33- and 44-tetras. For each pair, the mid-mesh can be constructed by assembling the individual 2D elements obtained when a 33-tetra is split with a plane across its 3 internal edges (producing a triangle) and a 44-tetra is split across its 4 internal edges (producing a quadrilateral). For the sake of example, assuming that the source patch is at the bottom and the target patch on top, Figure 22 illustrates how 33- and 44-tetras contribute to the mid-mesh. The contribution of a 33-tetra whose exposed face belongs to the target patch (blue triangle shown in A), the contribution of a 33-tetra whose exposed face belongs to the source patch (blue triangle shown in B), and the contribution of a 44-tetra with one exposed edge on the source and another on the target patch (blue quadrilateral shown in C) are shown.

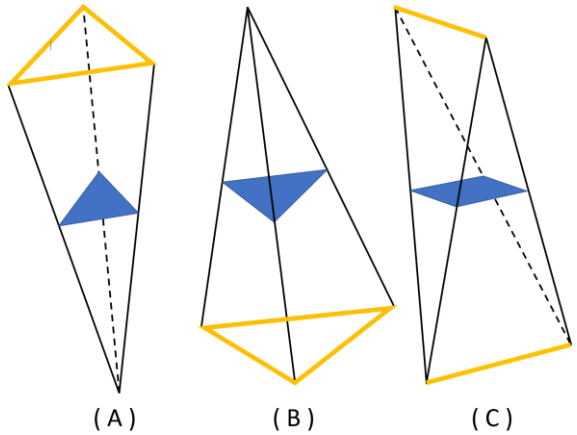

Figure 22: Contributions of 33-and 44-tetras to the mid-mesh. Surface edges are highlighted in thick gold color.

By assembling the individual triangles and quads (in turn, split into triangles along their shortest diagonal) the mid-mesh of any patch pair is built. A closeup of the mid-mesh can be seen in Figure 23 as the fine jagged light pink mesh where, for the sake of visibility, one of the primary prisms bounding the extrusion is shown as a wire mesh and a 64-tetra ending it is rendered in solid. It can be noted that the boundary of the mid-surface stops short of reaching the center of the prism and stops at the mid-points of the edges of the 54-tetras that form the primary prism.

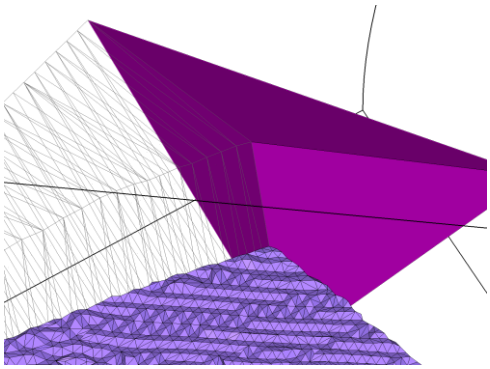

Figure 23: Closeup view of the partially constructed mid-mesh

At this stage, the set of mid-meshes would resemble what is depicted in Figure 24, i.e. a collection of isolated and unconnected surface meshes. Smoothing and quad-meshing these separate mid-meshes would result in a series of independent quad mesh patches which, once extruded, would result in a set of hex meshes that would not have any common nodes, unlike what was shown earlier in Figure 19, incompatible. Earlier, in Figure 17, a set of smoothed 2D abstractions were shown that did feature quad mesh compatibility (Figure 18) and compatibility of the resulting hex meshes

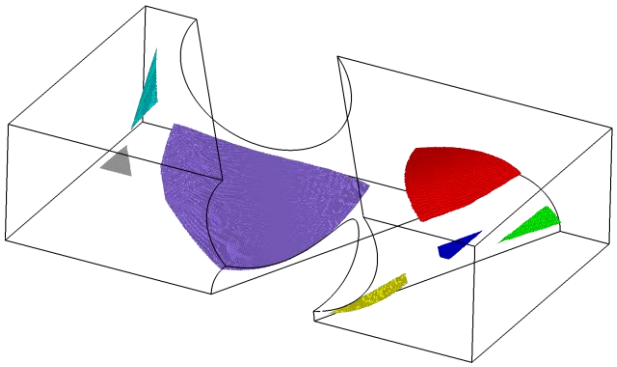

Figure 24: Incomplete mid-meshes

To ensure compatibility of the hex meshes resulting from extrusion, we introduce what we call a 2D abstraction by expanding each individual mid-mesh and augmenting them, peripherally, all the way to the center of the primary prisms and 64-tetras that surround them. By adding additional triangles, patches will meet at the center of primary prisms and 64-tetras. The added triangles are visible in Figure 25 where, for the sake of improved visibility, the corner 64-tetra is rendered transparent and one of the surrounding primary prisms is rendered as a wire mesh. The added triangles, visible as elongated light pink triangles, are constructed between the boundary vertices of the mid-patches and additional vertices placed at the center of faces (of 54-tetras) inside the primary prisms. In addition, two large triangles are also visible (resulting from two mid-face vertices and the center vertex of the 64-tetra) which complete the expansion of the mid-meshes into a 2D abstraction. After smoothing, the 2D abstraction, shown earlier in Figure 17, is ready for quad-meshing.

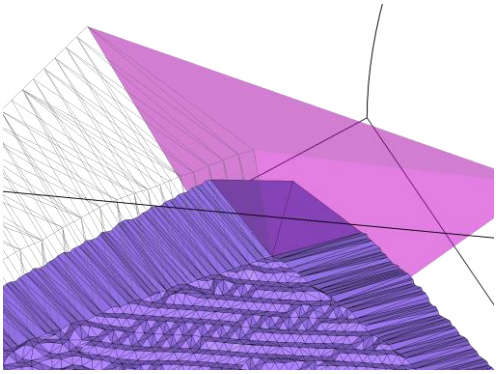

Figure 25: Completed unsmoothed extended mid-mesh (2D abstraction)

5.2. Data structure and sequence of operations

An internal data structure guarantees a two-way connection between the nodes of the initial Delaunay tetra mesh and the nodes of the final hex mesh as the Delaunay mesh is partitioned and the hex mesh is created through the following stages:

1. Initial closed triangular mesh

2. Constrained Delaunay tetrahedral mesh
3. Primary prisms and their rails
4. Secondary prisms
5. Patch pairs resulting from the partitioning of the surface mesh by the rails
6. Mid-patches computed from the patch pairs and extended to the centerline of prisms to form 2D abstractions
7. Smoothing and quadrilateral meshing of 2D abstractions and their extrusion into hexahedral mesh blocks
8. Meshing by slicing of primary prism blocks along marks dictated by the quad mesh boundary nodes
9. Meshing by slicing of secondary prism blocks along slices dictated by the slices of the primary prism blocks
10. Meshing of 64-tetras

5.3. Meshing of primary prism blocks

Each node of the quad-meshed 2D abstraction points to a node of the triangular 2D abstraction from which it stems. This is enforced by the quad meshing software component utilized in this work, namely, MeshGems-SurfOpt, developed by DISTENE [6]. In turn, each node of the triangular 2D abstraction is, by construction, the mid-point of a 33- or 44-tetra internal edge which connects one node of the source to one node of the target patch. Therefore, along the boundary of the quad mesh, each node also sits at the center of an internal face of a 54-tetra belonging to a primary prism bordering the extrusion.

Every boundary node of the quad mesh, shown in Figure 26, is represented by a black dot. By construction, each border node is at the center of an internal face of the primary prism that borders it. Therefore, each border node defines a slice of the primary prism block.

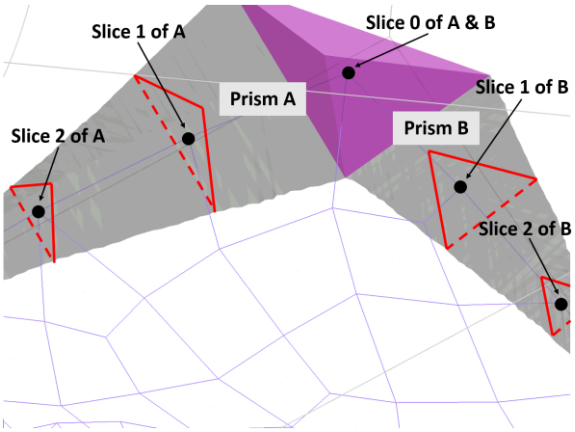

Figure 26: Quad border vertices define slices in bordering prism blocks

Primary prism blocks may now be sliced accordingly to create prism elements compatible with the extruded hexahedral blocks. In Figure 19, the extruded hex meshes have been split in 2 along the direction of the extrusion and the primary prism block slices have been split into 3 hexas each, and 64-tetra blocks and degenerate primary prisms blocks have been split into 4 hexas each.

5.4. Meshing of secondary prism blocks

Secondary prisms are bound by edge patches and flanked by primary prims. As with primary prism blocks, they can be split into prism elements and subsequently split into 3 hexas along the same slices as the primary prisms. Degenerate secondary prism blocks are treated as 64-tetras and meshed into 4 hexas accordingly.

5.5. Ducts and cables for handling non-manifold 2D abstractions

Sometimes, multiple primary prisms may be bunched together where multiple extrusions converge from different directions. Figure 27 shows such an example where two prisms, shown in light blue and gold, share a wall and act as one. In such cases, the data structure is further extended to include two new categories. Ducts, which are composed of multiple primary prisms sharing 2 rails, and cables which are a set of rails sharing all their edges.

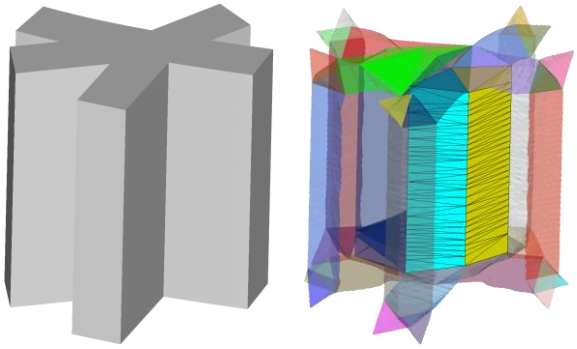

Figure 27: Bunching of two prisms into a duct. Solid model (left) and highlighted duct composed of 2 prisms colored light blue and gold

In the presence of a duct, during the process of building 2D abstractions, the mid-meshes are extended all the way to the centerline of ducts instead of the centerline of individual prisms. This ensure node compatibility among all the extrusions converging at the duct.

5.6. Saddle Points and 144-tetras

At geometry saddle-points, a different type of 44-tetras may be created where the exposed segments of the tetra are adjacent instead of being opposed, as encountered earlier. We call these 144-tetras. These define a new category of primary prism that start at 64-tetras and terminate at 144-tetras. We refer to these primary prism as knife primary prism.

5.7. Computation of geodesics on the Mesh

In many instances, including the meshing of secondary prism blocks it may be necessary to compute distances and geodesics along the edges at the surface of the initial surface mesh. To this end Dijkstra's algorithm is used as implemented by [7].

5.8. Smoothing prior to quad meshing

By construction, a 2D abstraction is generally very noisy as seen in Figure 25. Therefore, smoothing is necessary prior to quad-meshing of the 2D abstraction in order to improve signal-to-noise ratio and avoid the creation of tiny quads. A shrink-proof smoothing of the 2D abstraction and its boundary is used [8]. After smoothing, MeshGems-Surfopt [6], which incorporates its own mesh smoothing, is applied to obtain a quad mesh suitable for extrusion.

6. DEGENERACIES

Several types of degeneracies occur in this procedure. They are addressed by a series of local mesh manipulations, including edge collapse, edge split, edge flips [9], fast data structure updates and topology recalculation. The iterative resolution of these degeneracies uses a considerable portion of the computational work before the network of 64-tetra and primary prisms is finalized and 2D abstraction can proceed. These degeneracies are:

1. Symmetry degeneracies. These are Delaunay degeneracies which occur when a Delaunay mesh is created from an input triangular mesh that features local symmetries
2. Degeneracies resulting in 1- and 12-tetra proliferation. These degeneracies are related to symmetry degeneracies and occurs in bulky parts along convex curved surfaces.
3. Overlapping and folding rails which may also result in captive isolated faces between rails.
4. Convex curvature degeneracies which are also related to symmetry degeneracies.
5. Occurrence of convex sharp edges adjacent to more than one tetra or adjacent to a tetra other than a 12-tetra.
6. Formation of very small prisms made up of only a handful of 54-tetras or prisms with one or two zero-length rails.
7. Formation of entire captive patches. These occur when either te source or target patch in an extrusion pair is absent.
8. Occurrence of fused segments between otherwise distinct rails.

6.1. Symmetry degeneracies

The most common type of degeneracy is, referred to hereafter symmetry degeneracy, is one in which the surface mesh of an object featuring local symmetries is tetrahedralized using Delaunay. Due to the occurrence of multiple equidistant options for the formation of a Delaunay tetra, neighboring tetras may form in a seemingly chaotic fashion. For instance, the 4 vertical connector columns in the electronic component shown in Figure 28 degenerate into fragmented and chaotically oriented primary prisms.

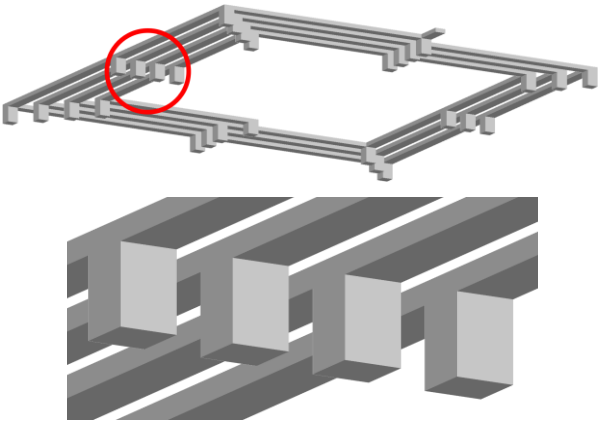

Figure 28: Circuit board (top) and detail (bottom) where Delaunay degeneracy occurs

Without special treatment, degeneracy leads to fragmented primary prisms in the vertical connectors (shown as stacks of small multicolor primary prisms in the columns, in Figure 29, top). An ad-hoc symmetry-breaking affine transformation Φ is applied to all the vertices of the surface mesh prior to Delaunay meshing. After the hex mesh is built, the reverse transform, Φ^{-1}, is applied to the vertices of the hex mesh in order to recover the true geometry. Here, we use a simple transform that stretches the model by a factor 1.1 in the x-, 1.2 in the y- and 1.3 in the z-direction. The degeneracy is removed and single primary prisms are ensured in the vertical columns of the model as shown in Figure 29, bottom.

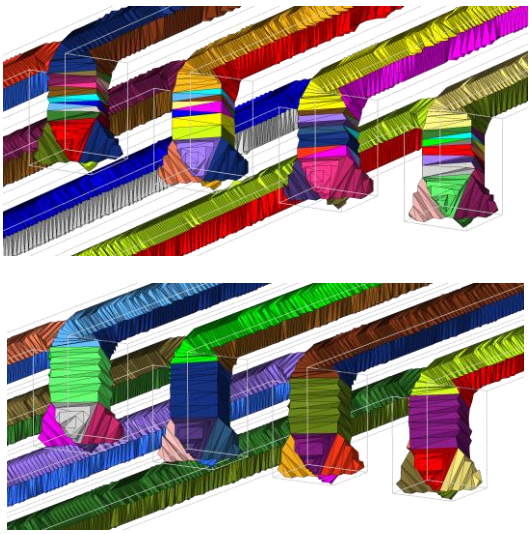

Figure 29: Fragmentation of prisms without affine transform (top). After removal of degeneracy through transform (bottom)

The Delaunay degeneracy may occur in any object. Therefore, the affine transformation is systematically applied as part of the general algorithm as shown in the pseudocode further below.

6.2. Degeneracies resulting in 1- and 12-tetra proliferation

The proliferation of 1- and 12-tetras are resolved by the systematic removal of such tetras. Figure 30, top, shows a cut constant-velocity boot (automotive, CV boot) which features large extents of slightly convex curved surfaces. Delaunay tetrahedralization results in flat 12-tetras, appearing in red, which, if not removed, would result in the creation of ghost primary prisms and the subsequent failure of the procedure.

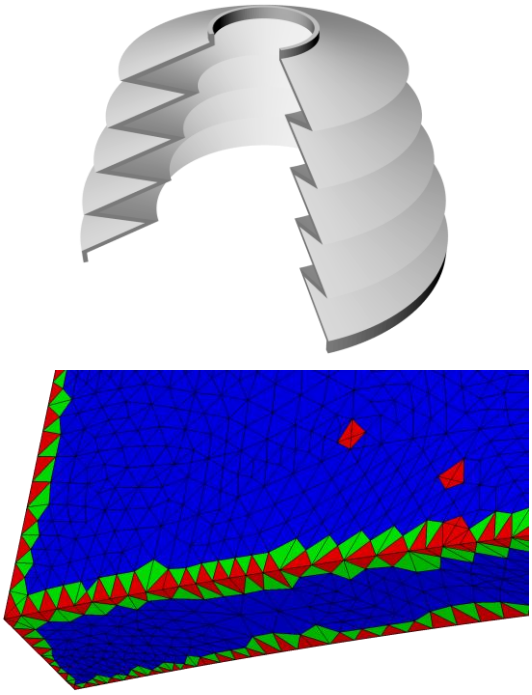

Figure 30: A cut CV-boot (top) results in spurious flat 1- and 12-tetras (in red, bottom) which are removed. Non-flat 12-tetras remain along edges

6.3. Overlapping and folding rail degeneracies

The coarser the input triangular mesh, the more likely it is that rails may overlap, especially in the vicinity of their starting nodes which are 64-tetra corner nodes. Figure 31 shows two such overlapping rails found in the CV boot example shown earlier. The 3 overlapping edges of the rails, shown with the 4 node marks, are collapsed in order to remove this degeneracy.

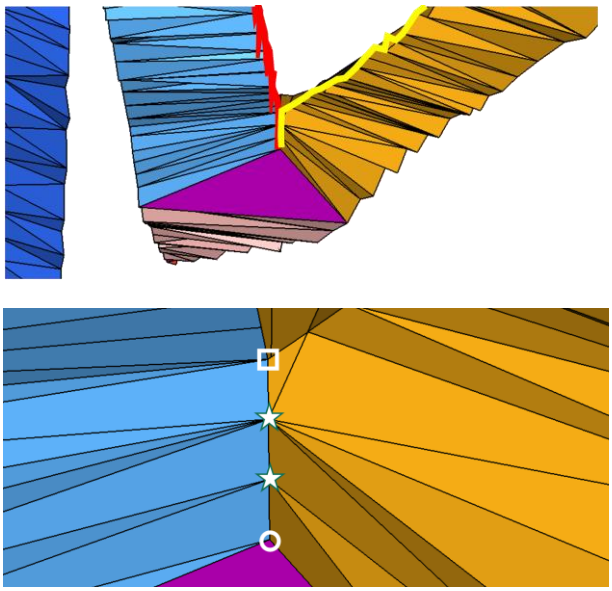

Figure 31: Overlapping rails shown in red and yellow in the top figure. The bottom figure shows how the rails must separate at the circle mark instead of the square.

6.4. "Peel rails or discard captive faces" degeneracies

Figure 32 shows how two neighboring primary prisms may share nodes (tagged as white circles) resulting in overlapping rails and/or the creation of captive faces. If a captive face (shown as a transparent yellow triangle) is removed through edge collapse, two separate and valid extrusions are created on the left and right side of the overlapping nodes. But if the overlapping rails are eliminated through a combination of edge split and edge collapse one single extrusion is created that surrounds the two loop primary prisms surrounding the 2 holes of the model. The latter is the solution that is automatically retained based on aspect ratio considerations, and can be seen in the Examples section.

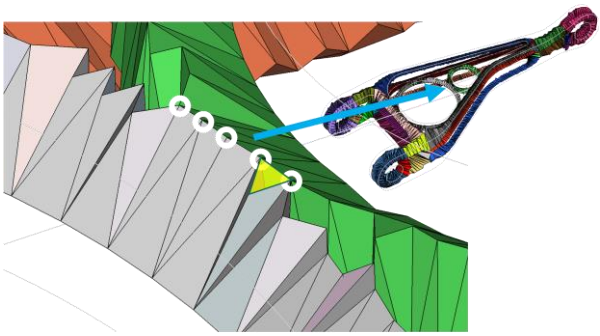

Figure 32: Rail peel and captive face degeneracy

6.5. *Convex curvature degeneracies*

Figure 33 shows a type of degeneracy caused by noisy convex and featureless surfaces. They also result in a proliferation and concentration of 1- and 12-tetras in the curved areas. Their systematic removal would leave behind an intractable primary prism structure that would result in the failure of the procedure.

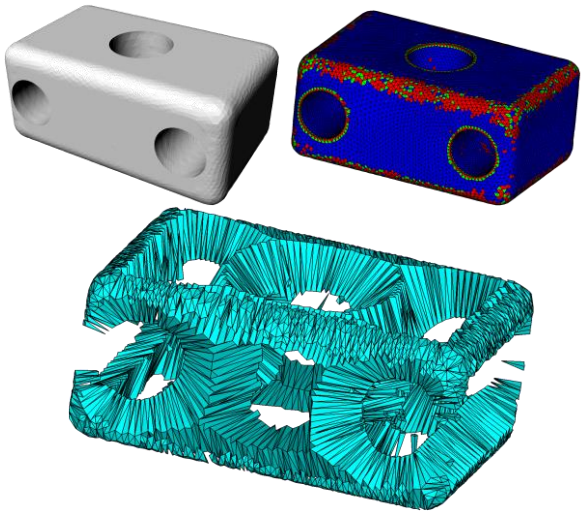

Figure 33: Convex curvature degeneracy

7. AUTOMATION

The following pseudocode (algorithm) is performed on a closed triangulated surface mesh.

1. Read Input surface mesh

2. Process surface mesh
- 2.1. Improve mesh sizing
- 2.2. Identify feature angles

3. Affine transform Φ of the input surface

4. Build 2D Abstraction of input as a quad mesh
- 4.1. Build Delaunay tetrahedral mesh
- 4.2. Iterative removal of other (than symmetry) degeneracies
  - 4.2.1. Discard Isolated 01- and 12-tetras that have "flat" dihedral angles
  - 4.2.2. Ensure 12-tetras at convex features (flip)
  - 4.2.3. Identify and label tetras as 64, 54, 33, 44, etc.
  - 4.2.4. Identify primary prisms and their rails
  - 4.2.5. Correct Folding rails (collapse)
  - 4.2.6. Discard captive faces between rails (collapse)
  - 4.2.7. Discard "tiny" prisms (collapse)
  - 4.2.8. Move 64-tetra corners to closest concave node (collapse)
  - 4.2.9. Partition surface mesh into patches
  - 4.2.10. Identify extrusion patch pairs
  - 4.2.11. Remove captive patches when one of the 2 extrusion patches is missing (collapse)
  - 4.2.12. Peel rails apart (edge splitting)
  - 4.2.13. Identify ducts
  - 4.2.14. Ensure that all prisms in a duct have the same number of 54-tetras (collapse)
- 4.3. Build mid-meshes of extrusion patch pair
- 4.4. Extend mid-meshes and obtain 2D abstraction
- 4.5. Smooth and quad mesh 2D abstraction

5. Build mixed hexa, prism and tetra mesh
- 5.1. Build extrusion blocks (hex elements)
- 5.2. Build 64-blocks (single tetra element)
- 5.3. Build degenerate primary prism blocks (single tetra element)
- 5.4. Build non-degenerate primary prism blocks (stacks of prism elements)
- 5.5. Build degenerate secondary prism blocks (single tetra element)
- 5.6. Build non-degenerate secondary prism blocks (stacks of prism elements)

6. Split mixed mesh to get all-hex mesh
- 6.1. Split tetras into 4 hexas
- 6.2. Split prisms into 3 hexas
- 6.3. Split extrusion hexas into 2 along the extrusion

7. Inverse transform Φ^{-1} of hex mesh

8. EXAMPLES

The following figures represent 2D abstractions and coarse hex meshes obtained for 5 examples. Once again, it can be noted that these 2D abstractions are simpler and contain fewer branches than mid-surface/medial axis abstractions of similar objects.

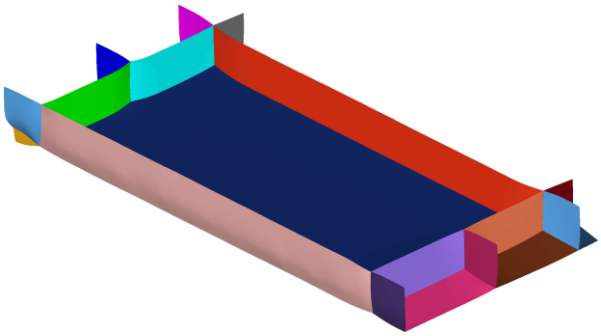

Figure 34: Heatsink 2D abstraction

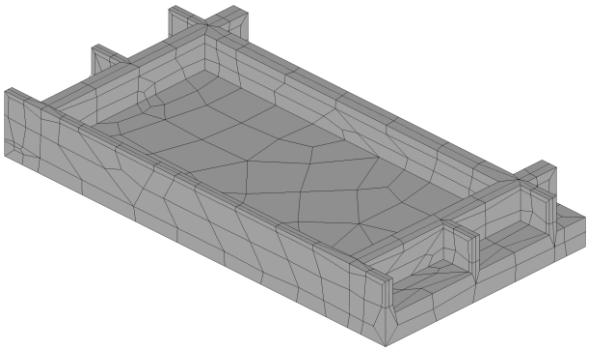

Figure 35: Heatsink hexa mesh

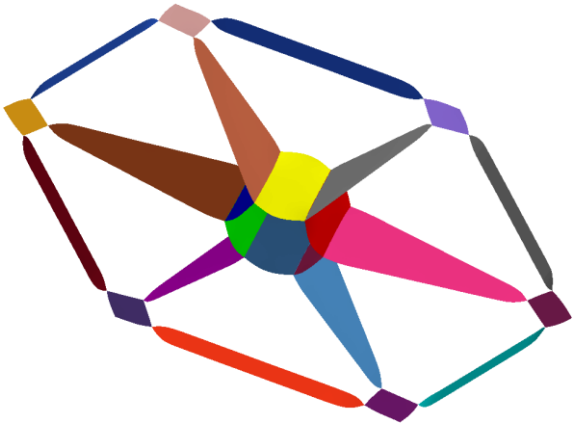

Figure 36: Flywheel 2D abstraction

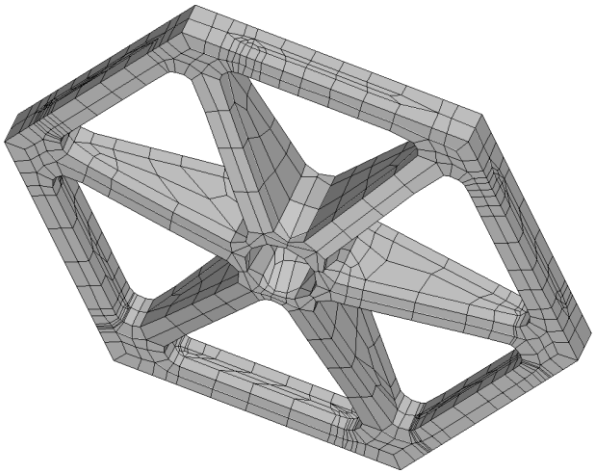

Figure 37: Flywheel hexa mesh

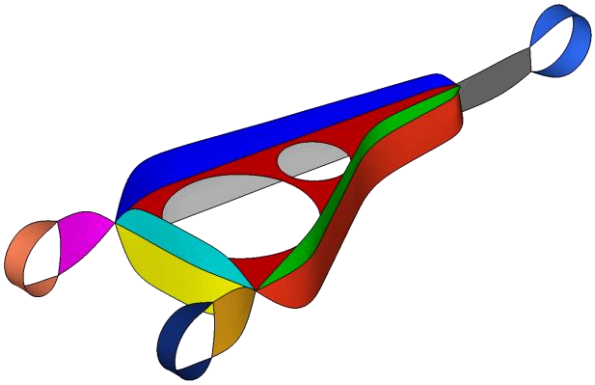

Figure 38: Landing gear 2D abstraction

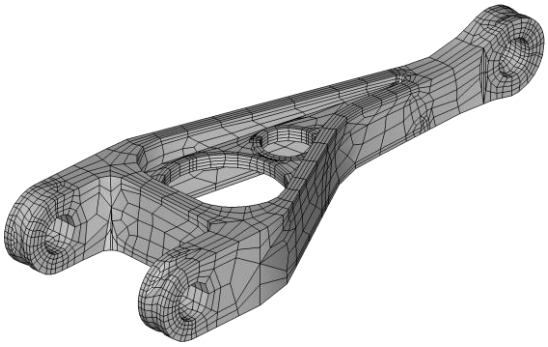

Figure 39: Landing gear hexa mesh

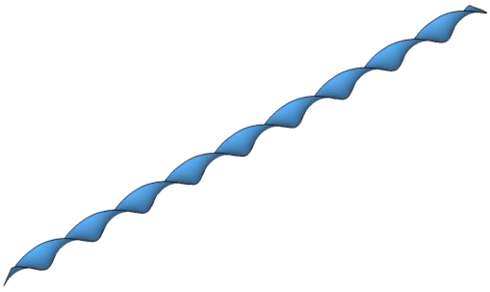

Figure 40: Drill bit 2D abstraction

Figure 41: Drill bit hexa mesh

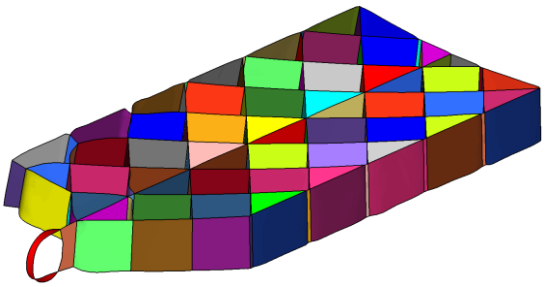

Figure 42: SpaceX Falcon-Heavy center booster half-gridfin 2D abstraction

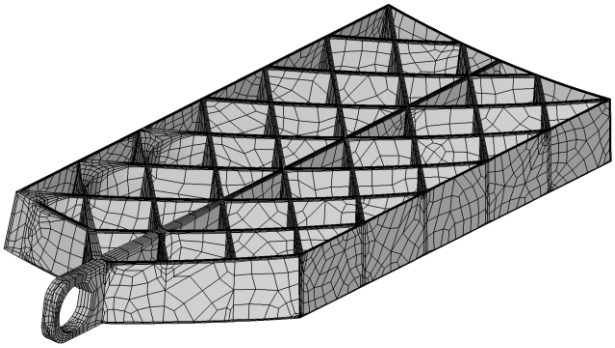

Figure 43: SpaceX Falcon-Heavy center booster half-gridfin hexa mesh

9. CONCLUSION AND REMARKS

An automatic hexahedral meshing framework is presented that uses the optimal properties of a Delaunay tetrahedralization to create a 2D abstraction and a block-structured hexahedral mesh. Several open questions remain including how to deal more effectively with various types of degeneracy. A few useful outcomes of this work are:

1. The proposed 2D abstraction is simpler, has fewer branches than a mid-surface/medial axis and, a-priori, requires no pruning. It essentially lets the branching be handled by the network of tetra and prism blocks. It could possibly replace semi-automatic abstraction tools used in today's CAD systems.
2. The block-structured hex mesh has the special property that the grid layering along extrusions and that along the prism blocks do not interfere and thus allow for the independent assignment of extrusion mesh density and mesh resolution across the extrusions and azimuthally within the prism and tetra blocks. This also allows for the implementation of local and compact hex mesh refinement schemes for local mesh refinement.
3. The hexahedral mesh layering near holes is adequately oriented along hole features; which is a desirable quality of FEA meshes.
4. A potential application of this type of combined abstraction/hex meshing is in meshing ultra-thin objects where vast swaths of extrusions coexists with a few highly 3D zones. This would enable the creation of mixed shell and solid meshes from the same input object.

10. ACKNOWLEDGMENTS

The author wishes to thank Altair Engineering, Inc. (USA), InTechSolutions Co., Ltd (Korea) and NEC Solution Innovators, Ltd. (Japan) for their support.

11. REFERENCES

[1] K. Y. Lee, *Analysis Model Derivation from Design Geometry*, School of Mechanical & Manufacturing Engineering Faculty of Engineering The Queen's University of Belfast, 2004.

[2] R. Taghavi, "Automatic clump generation based on mid-surface," in *Continuum and Distinct Element Numerical Modeling in Geomechanics*, Melbourne, 2011.

[3] C. Armstrong, *Personnal Communication*, 2018.

[4] N. Amenta, S. Choi and R. K. Kolluri, "The power crust,unions of balls, and the medial axis transform," *Comput. Geom. Theory and Applications*, vol. 19, 2001.

[5] W. R. Quadros, "LayTracks3D: A new approach for meshing general solids using medial axis transform," *Computer-Aided Design*, vol. 72, pp. 102-117, March 2016.

[6] DISTENE S.A.S., "MeshGems-SurfOpt Version 2.7 API Manual," Bruyères-le-Chatel, 2019.

[7] M. Yan, "Dijkstra's Algorithm," 2015. [Online]. Available: http://math.mit.edu/~rothvoss/18.304.3PM/Presentations/1-Melissa.pdf. [Accessed 3 August 2015].

[8] J. Vollmer, R. Mencl and H. Müller, "Improved Laplacian Smoothing of Noisy Surface Meshes," *EUROGRAPHICS*, vol. 18, no. 3, 1999.

[9] J. R. Shewchuk, *Delaunay Refinement Mesh Generation*, Pittsburgh, PA: School of Computer Science Computer Science Department Carnegie Mellon University , 1997, pp. 22-24.