Pillowing doublets: refining a mesh to ensure that faces share at most one edge

Scott A. Mitchell

samitch@sandia.gov

Timothy J. Tautges

tjtautg@sandia.gov

Comp. Mechanics and Visualization Dept., Sandia National Laboratories, Albuquerque, NM 87185

Abstract

Occasionally one may be confronted by a hexahedral or quadrilateral mesh containing *doublets*, two faces sharing two edges. In this case, no amount of smoothing will produce a mesh with agreeable element quality: In the planar case, one of these two faces will always have an angle of at least 180 degrees between the two edges.

We describe a robust scheme for refining a hexahedral or quadrilateral mesh to separate such faces, so that any two faces share at most one edge. Note that this also ensures that two hexahedra share at most one face in the three dimensional case. We have implemented this algorithm and incorporated it into the CUBIT mesh generation environment developed at Sandia National Laboratories.

1. Introduction

In a quadrilateral or hexahedral mesh, a *doublet* is defined as two quadrilateral faces that share two edges. When this occurs in a mesh, there is something fundamentally wrong with the local connectivity: In any geometric embedding of these faces, at least one face will have poor quality. In order to remove the doublet and obtain good element quality, it is necessary to change the local connectivity of the mesh through refinement.

Doublets occasionally arise during the weaving stages of Whisker Weaving[1], and when wedges are collapsed[2][3]. Plastering[4] and Whisker Weaving both collapse wedges in their final stages, and collapsing wedges can be used to locally coarsen a mesh. Figure 1 shows that when a mesh is already fairly coarse, collapsing a face (e.g. via a wedge) can create a doublet[3]. Another type of poor connectivity that arises is two hexahedra sharing

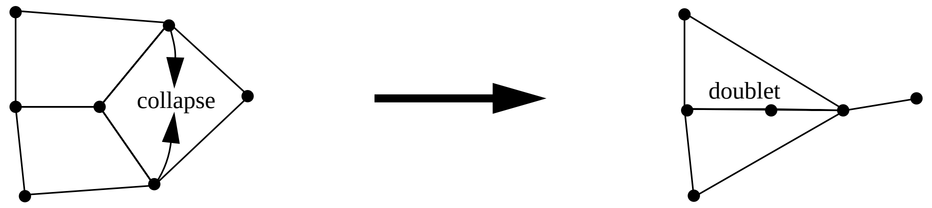

Figure 1. Collapsing a face in an already coarse mesh sometimes creates a doublet.

two faces. However, this implies that there are two doublets in the mesh, as seen in Figure 2. Thus ensuring that no doublets exist also ensures that two hexahedra share at most one face.

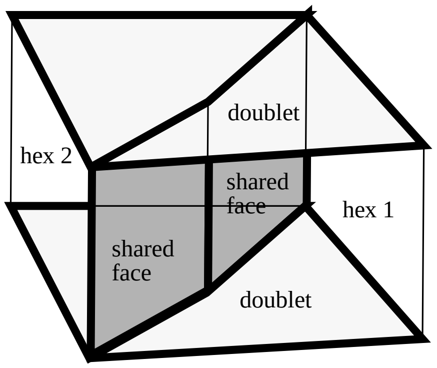

Figure 2. Two hexahedra sharing two faces implies two doublets.

In a quadrilateral mesh, there is a simpler alternative to what we propose: Doublets may be removed by simply deleting the two shared edges, forming one big quadrilateral out of the two doublet faces. However, this simple scheme is not possible in a hexahedral mesh in general, as there is no guarantee that the hexahedra containing the doublet faces are connected in such a way that they can be combined. This is illustrated in Figure 3.

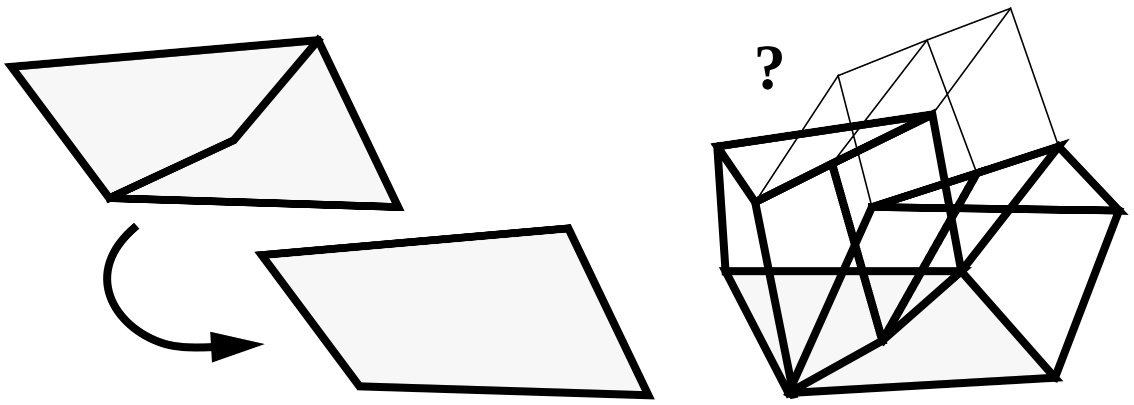

Figure 3. Left: In a quadrilateral mesh, a doublet may be removed by removing the shared edges and merging the doublet faces into one face. Right: In a hexahedral mesh, this is not always possible because the hexahedra containing the doublet faces cannot be combined in general.

We describe a robust scheme for eliminating doublets, so that any two faces share at most one edge. This idea works in both two and three dimensions, and we speculate that it could be extended to higher dimensional meshes as well. The basic idea is to refine the mesh in such a way that the connectivity of the doublet node is increased. The algorithm has three main steps, which are repeated for every face that shares two edges with another face. First, we find a *shrink set*, a group of elements containing one, but not both, of the faces that share two edges. Second, we disconnect the shrink set elements from the elements on its boundary and geometrically reduce the size of the shrink set. Third, we connect the shrink set to its old boundary with a layer of elements. These steps are outlined in Figure 4. In three dimensions the connecting layer often resembles a spherical pillow, hence the name *pillowing doublets.* Performing these three steps is called *pillowing a doublet face*. Performing these three steps for both faces of the doublet is called *pillowing a doublet.* If the input mesh has doublets that are very close together, then the output of the algorithm is dependent on the order in which the doublet faces are pillowed. At the end of these operations, the mesh is smoothed.

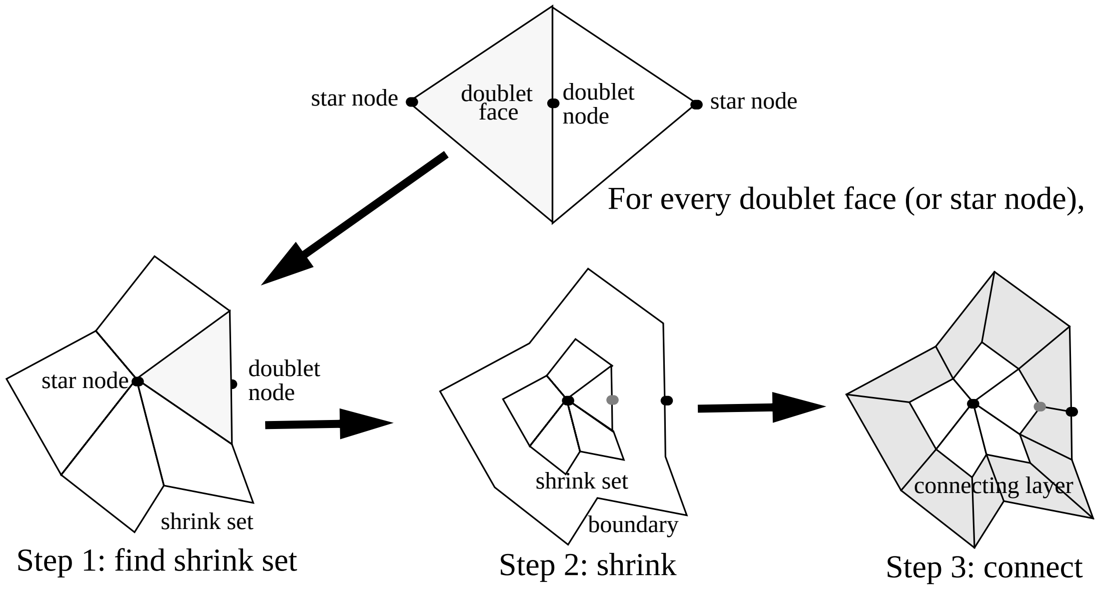

Figure 4. Pillowing a doublet in a quadrilateral mesh.

The algorithm also has an easy explanation in terms of the spatial twist continuum[5], or STC. The rest of this paper makes no reference to the STC, so this explanation can be skipped if desired. In the STC, a doublet consists of two vertices connected by two edges, see Figure 5. This is also called a *degree-2 2-cell*[3]. In three dimensions, pillowing the doublet consists of inserting a twist plane that cuts across these two edges. Choosing the shrink set is equivalent to choosing the set of STC vertices that are on one side of the twist plane (i.e. inside the twist plane if it is a sphere).

The remainder of this paper is organized as follows. In section 2 we describe how to choose the shrink set so that the doublet face is always pillowed, and also a heuristic for choosing a large shrink set that leads to good element quality. In section 3 we describe the process of disconnecting the shrink set hexes from the surrounding mesh. In section 4 we describe how to insert the pillow layer of elements. In section 5 we analyze the number of hexes introduced by our algorithm, and its running time. In section 6 we discuss some further variants on the algorithm, and in section 7 we present conclusions.

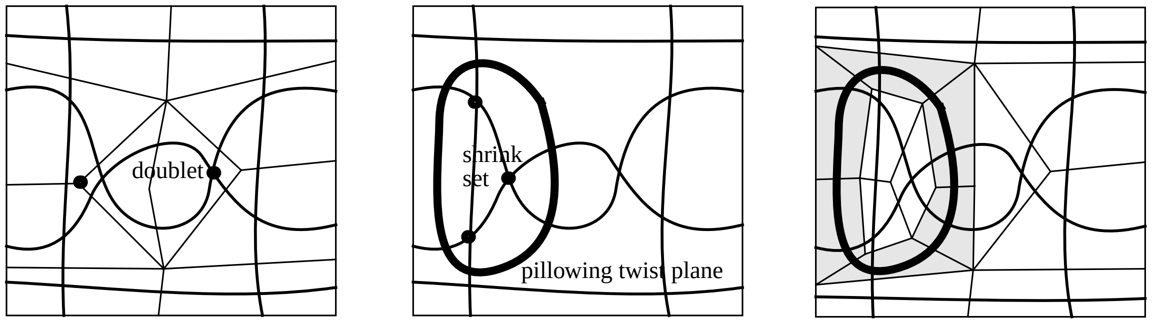

Figure 5. The STC interpretation of pillowing doublets. A twist plane that resembles a pillow is inserted so that it surrounds the STC 2- or 3-cell that contains the primal star node. Pillowing twist planes that pass through a common 2- or 3-cell are merged in an amoeba-like fashion into one big pillowing twist plane. In this figure, another pillowing twist plane will eventually be added around the other star node to pillow the other doublet face.

2. Building a shrink set

A *doublet* is two faces that share exactly two edges. The node common to those edges is called the *doublet node*. In each of the two faces, the node opposite the doublet node is a *star node*. In this section we describe the algorithm for constructing the shrink set for a given star node. The shrink set will contain the doublet face that contains the star node. This is the first step in pillowing a given doublet.

There are actually many options for constructing a shrink set that will lead to removal of the doublet. In two dimensions, *any set of elements that contains one doublet face but not the other is sufficient.* The reason this is sufficient will be evident in section 4. on page 9. In three dimensions, *any set of elements that contains all hexahedra containing one doublet face but not all hexahedra containing the other doublet face is sufficient.* In addition, in order to perform the second step of disconnecting and shrinking the shrink set, for every point in the shrink set, we require that there is a small neighborhood, open relative to the meshed object, whose intersection with the shrink set is homeomorphic to a ball or half-ball. Based on experience, large sets without sharp angles on their boundary tend to lead to better quality elements.

The basic scheme that we use is to add all of the elements containing the star node into the set. When star nodes for different doublet nodes are close together, the shrink set is made larger to contain multiple star nodes: if the shrink set contains another star node, then we recursively add all of the elements containing that star node into the set as well. This heuristic gives big, roundish shrink sets, which gives good mesh quality. However, to satisfy our sufficient condition, we take care that the two star nodes for a given doublet node are never in the same shrink set: If one is in the shrink set, the other is marked as forbidden. We only add the elements containing a star node if these elements do not contain a forbidden node.

The pseudocode for this is as follows:

```
find_shrink_set( given star_node, return shrink_set)
shrink_set := empty
star_node_list := given star node
do
   star_node = pop star_node_list
   elements = elements containing star_node
   if ( elements don't contain a forbidden star node )
      shrink_set += elements
      star_node_list += new star_nodes in elements
while star_node_list is not empty
```

Note that it is possible that this algorithm returns an empty shrink set. In two dimensions, this may occur when a third face shares an edge with both faces of a doublet, as in Figure 6.

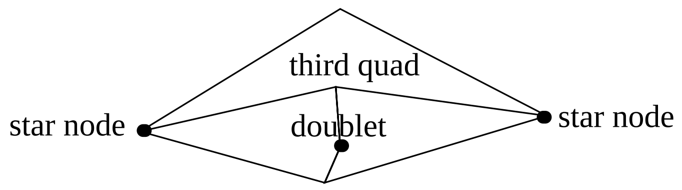

Figure 6. The shrink set building algorithm fails when an element contains both star nodes of a doublet. The algorithm is retried after progress is made elsewhere in the mesh. In any event, the algorithm can be made robust by merely taking a smaller shrink set.

A robust solution is to take the shrink set to be the single doublet face containing the star node. Certainly this meets the sufficient condition that the shrink set contains one doublet face but not the other: After pillowing, the doublet edges are contained in two different quadrilaterals of the pillow layer. This removes the doublet. Angles would be bounded away from 180 degrees and the local connectivity would appear acceptable, but in fact element quality would still not be very good, as seen in Figure 7 top.

In three dimensions, if the doublet face containing the star node is in the interior of the meshed object, it is always possible to take the shrink set to be the two hexahedra h_1 and h_2 containing that doublet face as in Figure 7 bottom. The sufficient condition is satisfied by noting that the shrink set can't contain all hexahedra containing the other doublet face: If it did, then at least one of h_1 and h_2 contains both doublet faces. That hexahedra has only two edges (the shared doublet edges) at the doublet node, but a hexahedron has three edges at every node, a contradiction. Similarly, if the doublet face is on the boundary of the meshed object, it is always possible to take the shrink set to be the single hexahedron containing it. As in two dimensions, while this solution is robust, it is not ideal from an element quality standpoint. Element quality would be particularly poor if this solution was used for many doublets which are near one another.

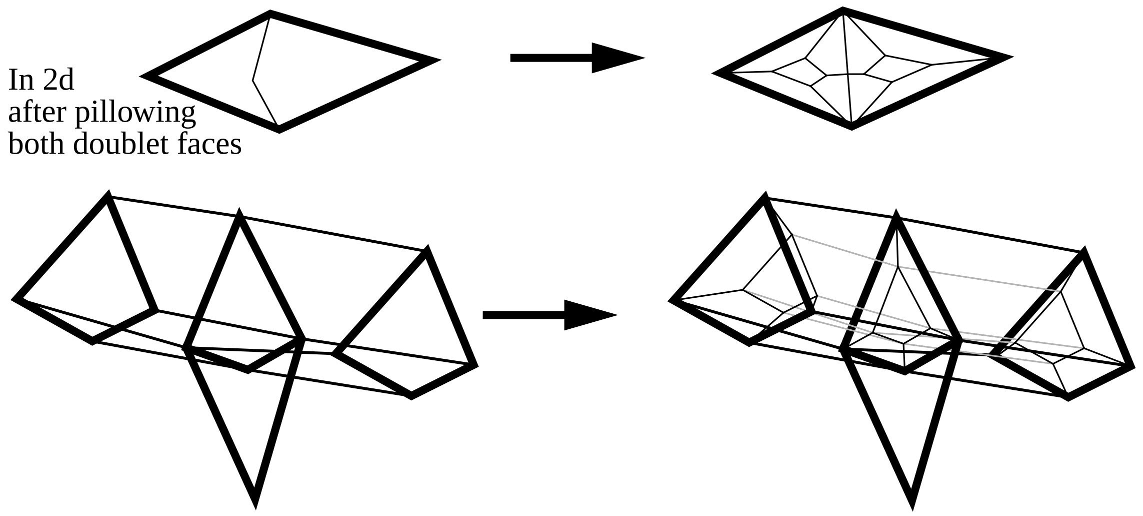

Figure 7. These shrink sets are always possible and always lead to the removal of the doublet, but element quality could be better by taking larger, rounder shrink sets as in Figure 4.

In practice, if the algorithm returns an empty shrink set for a star node, we defer the resolution of that star node until all other star nodes are resolved. This heuristic works well in practice: The algorithm initially fails when the mesh is very coarse near the star node. This usually means that other doublets are close by. After some of these close doublets are pillowed the local mesh is often changed enough so that re-running the algorithm returns a good shrink set for the first star node.

3. Disconnect and shrink

In this section we describe how to disconnect the shrink set from the remaining mesh and reduce the geometric size of the shrink set. This is the second step of pillowing a doublet face.

Given the hexahedra of the shrink set, it is straightforward to determine if a facet is on the boundary of the shrink set (the exception is when a facet lies on the boundary of the meshed object; see below). We create a copy of each facet on the shrink set boundary: one copy stays with the shrink set, and the other stays with the remainder of the mesh. The condition that every point in the shrink set has a small neighborhood whose intersection with the shrink set is homeomorphic to a ball or half-ball ensures that a facet has only one copy that needs to stay with the shrink set. If a star node is on the boundary of the shrink set, but has not yet been pillowed, then the copy that stays with the remainder of the mesh is considered to be a star node, but the copy that stays with the shrink set is not.

A complication arises when, in two dimensions, the shrink set contains edges on the boundary of the meshed object. Is such an edge on the boundary of the shrink set, or in its interior? Figure 8 shows the difference between the two answers. A similar question arises in three dimensions.

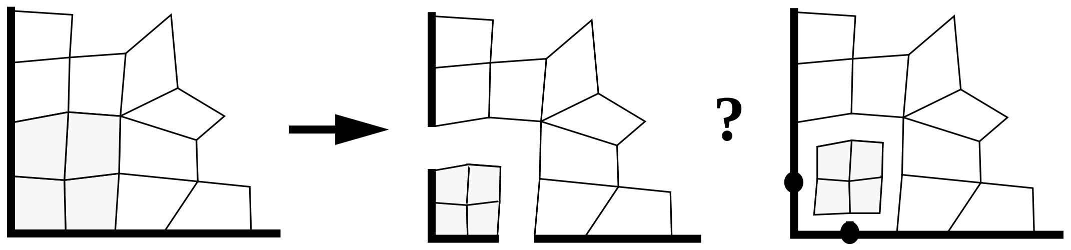

Figure 8. Should edges on the boundary of the object be considered to be in the interior of the shrink set (center), or on its boundary (right)? In three dimensions, if a doublet face lies on the boundary of the meshed object, then we must consider it to be in the interior of the shrink set.

In order for the algorithm to remove a doublet face, the doublet face must be considered to be in the interior of the shrink set. Hence, in three dimensions, when a doublet face lies on the object boundary it must be considered to be in the interior of the set. For example, there are four doublet faces on the surface of the meshed object in Figure 9 Additionally, some

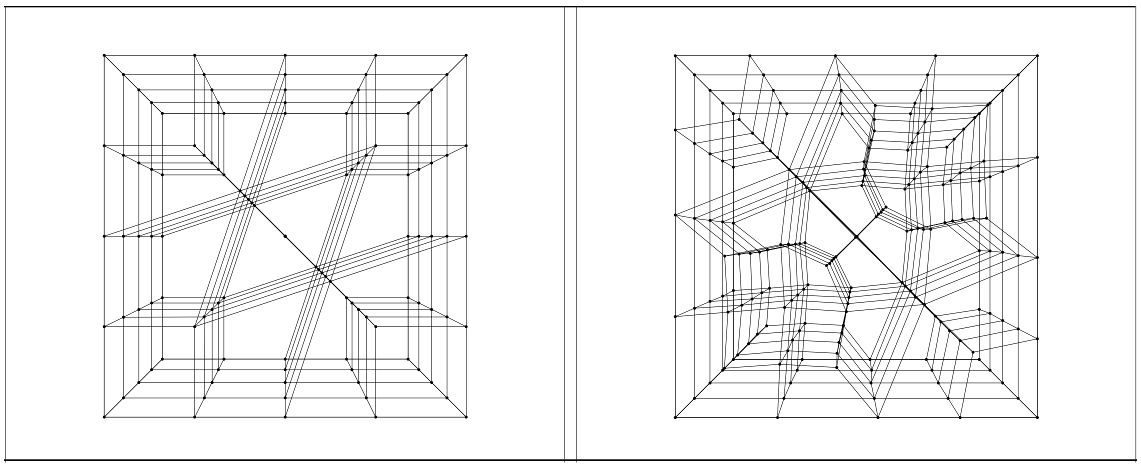

Figure 9. A wire-frame view of the “block-doublet” test problem before (left) and after (right) pillowing the column of doublets in the center of the block. Two large, cylindrical pillow sheets are added with axis into the paper, one around the upper right and one around the lower left column of star nodes.

care is necessary to determine if a boundary facet or its copy stays on an input surface, curve or vertex.

However, occasionally we are presented with a mesh that contains *surface doublets*. In two dimensions, a surface doublet is a single face, two of whose edges are constrained to lie on a straight or near-straight curve bounding the meshed object; see Figure 10. In three dimensions, the edges may also lie on a flat or near-flat bounding surface as in Figure 11. In order to pillow such doublets, it is necessary to consider the facets of the shrink set lying on the meshed object's boundary to be on the boundary of the shrink set, as in Figure 10 and Figure 11.

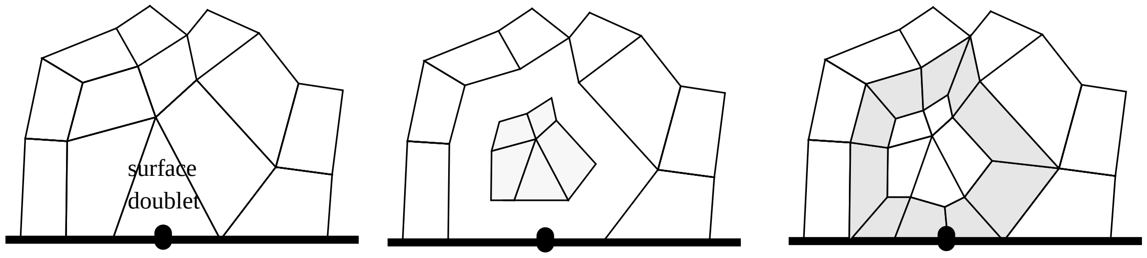

Figure 10. Pillowing a *surface doublet.* Here the edges on the boundary of the meshed object must be considered to be on the boundary of the shrink set.

Our solution to this dilemma in three dimensions is to consider an object surface face to be on the interior of the shrink set if and only if it contains a star node of the shrink set that is on the object surface. In addition to handling the above two cases, this has the additional advantage that the surface mesh is only modified when necessary. Unfortunately, this can create surface doublets in rare cases. Our solution is to find and pillow surface doublets in a separate pass, after all non-surface doublets have been pillowed.

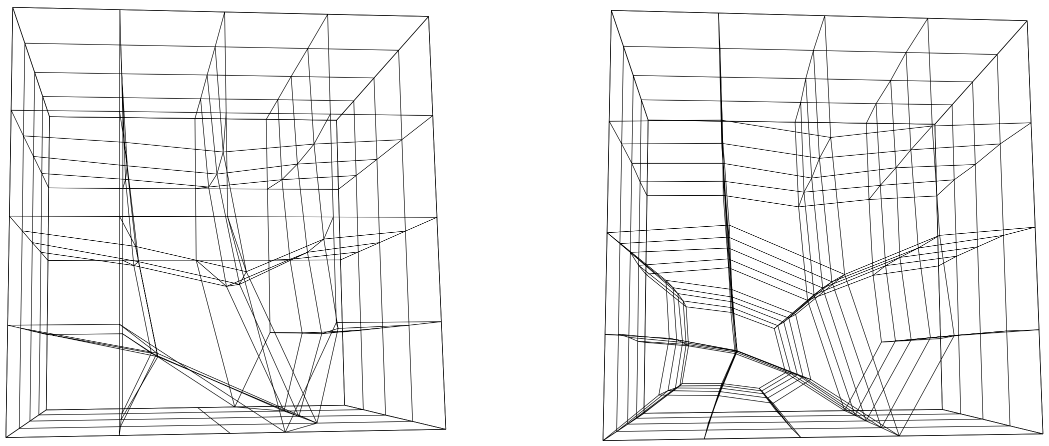

Figure 11. Pillowing to remove surface doublets in a three dimensional example. Left, a column of surface doublets exists, with each doublet face containing two edges on the bottom face of the cube. Right, a cylindrical pillow sheet with axis into the paper was inserted, removing the surface doublets.

Geometrically shrinking the set is straightforward. The exact positions are not important, since the entire mesh will be smoothed later. This step is really just included here to make the description of the algorithm more intuitive, and the code easier to develop.

4. Connecting with a layer of elements

We place an edge between each node on the boundary of the shrink set B and its copy on the remainder of the mesh C. Similarly, we place a face between each edge on B and its copy on C. In three dimensions, we place a hexahedron between each face on B and its copy on C. The layer of elements often resembles a ring in the two-dimensional case, and a spherical pillow in the three dimensional case. However, this is not necessary. Because large shrink sets without sharp angles give the best element quality, in three dimensions we build layers that can resemble cylinders, tori, or any other type of orientable manifold.

Algorithm correctness

If doublet face f is in the shrink set, and its shared edges are on the boundary of the shrink set, then each shared edge will be split into two and a face inserted between the two copies. By construction, all other doublet faces sharing those edges with f are outside the shrink set. Hence f now shares only one edge with any other face, and the doublet has been removed.

5. Output size and running time

Let n be the number of nodes, edges, faces, and hexahedra in the input mesh, and d the number of doublet nodes. In this section we show that pillowing doublets produces $\gamma = O(d)$ new elements, and can be performed in time proportional to γ after an $O(n)$ preprocessing step to find the doublets.

We assume that given a facet f of some dimension, we can obtain the list of m facets containing f, or contained by f, in time $O(m)$. We further assume that, on average, m can be considered to be a small constant compared to the size of the mesh. By stepping through the mesh faces, and visiting faces that share an edge with the current face, all doublets can thus be found in time $O(n)$.

Similarly, by starting with a star node and visiting containing elements, each shrink set can be found in time proportional to the size of the shrink set, times an average connectivity constant. Each star node keeps a list of the other star nodes for the same doublet node, in order to be able to mark forbidden nodes quickly. The size of this set is also a small connectivity constant.

Each shrink set, regardless of whether nearby shrink sets have been previously inserted, is proportional to the number of star nodes it contains times a connectivity constant. The number of original star nodes is at most $2d$. The number of surface doublets created by pillowing the original star nodes is smaller than the number of original star nodes.

Disconnecting, copying, and inserting the layer of elements can all be done in time proportional to the size of the shrink set.

6. More alternatives

The astute reader may have noticed that both star nodes for a doublet are pillowed, while in actuality pillowing one of the star nodes of a doublet is sufficient to ensure that the doublet is removed. We pillow both doublets, since this gives better element quality. Pillowing a star node increases the edge degree of every node on its boundary by one. In particular, in the two dimensional case, the doublet node had edge degree increased from two to four after pillowing both star nodes.

7. Conclusion

We have given a robust algorithm for refining a hexahedral or quadrilateral mesh so that no *doublets* are present, i.e. so that any two faces share at most one edge. In three dimensions, the surface mesh of the object is only modified if doublet faces are present on the surface mesh. The algorithm is very general, and may be modified by several heuristics for making larger shrink sets in order to get better element quality. The number of new elements created is on the order of the number of doublets in the original mesh, times a connectivity constant. The running time is proportional to the number of new elements created, plus time proportional to the size of the original mesh to initially find the doublets.

References

1. T. J. Tautges, T. D. Blacker, S. A. Mitchell, “Whisker Weaving: a connectivity based method for constructing all–hexahedral finite element meshes”, submitted to Int. J. Numer. Methods Engrg. (1995).

2. T. D. Blacker, R. J. Meyers, “Seams and wedges in Plastering: a 3-D hexahedral mesh generation algorithm”. Engineering with Computers, 9:83-93 (1993).

3. T. D. Blacker, S. A. Mitchell, T. J. Tautges, P. Murdoch, S. Benzley, “Forming and resolving wedges in the spatial twist continuum”. Engineering with Computers, to appear (1995).

4. 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.

5. P. Murdoch, S. Benzley, T. D. Blacker, S. A. Mitchell, “The spatial twist continuum: a connectivity based method for representing all–hexahedral finite element meshes”, submitted to Int. J. Numer. Methods Engrg. (1995).