AIAA SciTech Forum
7-11 January 2019, San Diego, California
AIAA Scitech 2019 Forum

10.2514/6.2019-1988

Towards Fully Regular Quad Mesh Generation

Julia Docampo-Sánchez* and Robert Haimes†
Department of Aeronautics & Astronautics
Massachusetts Institute of Technology
Cambridge, MA, 02138

Most structural analysis solvers provide much more accurate results when given a pure quadrilateral (as apposed to triangle or mixed) mesh as input. A triangulation of a complex trimmed surface can be always generated, so a complete quadrilateral mesh is possible by subdividing each triangle into 3 quadrilaterals. This is accomplished by spitting each triangle side and including the centroid of the triangle in the splitting. Though this produces a purely quadrilateral watertight mesh, the quad shapes suffer from inheriting the shape of the original triangle and the vertex irregularity is quite high. This paper describes a technique that recovers regularity by performing iterative topological changes on the mesh and attempts to produce a quad alignment that follows the geometrical features. The end result is a watertight surface mesh of a BRep that is completely quadrilateral and almost fully regular which is suitable for structural analysis and possibly other PDE solver schemes.

I. Quad Meshing

Quad meshes have a natural advantage over triangular meshes when representing the local geometry of a surface; since there are two logically orthogonal directions (u and v) for the surface, quads can be designed to follow these isoclines, producing meshes that reflect the underlying geometry. Generally, automatic quad meshing is achieved through *direct* methods such as advancing front techniques [1–3] which generate quads from the beginning or *indirect* methods which start with a triangular tessellation [4–6] that is subsequently converted into quads. This can be done by appropriate triangle pairing based on a version of the Blossom Algorithm [7] which solves the perfect matching problem (in this context, every triangle is paired to another) in minimal time. However, not every graph (triangular mesh) has a perfect match so the resulting quadrangulation might leave several triangles which have to be subsequently eliminated [4] in order to obtain a fully quadded mesh. Furthermore, there is still a cost associated with this algorithm [8]. Alternatively, quads can be produced directly from a triangle split using its medians and inserting an extra vertex at the centroid [9] (see Figure 1). Although this is a fast and effective technique, it produces a very irregular mesh containing a high number of irregular vertices (vertices with valence $\neq 4$) but as it will be shown later, vertex regularity can be recovered by performing local topological changes.

Vertex regularization consists of performing local changes around the quad configuration that result in an increase of vertices with valence four (regular). Regular vertices are desirable since it allows for aligning quads, for example, along the principal curvature directions of the surface or the isoclines. Examples of regularization techniques are appropriate repositioning of irregular vertices (singularities) in order to remove helix-like structures [10]. Or the Quad Mesh Simplification technique [11], based on poly-chords and the fundamental operation of quad collapsing or [12–16] which combine several fundamental operations, namely: swapping, splitting and collapsing.

Automatic mesh generation usually involves a post-processing stage since sharp and flat elements with undesirable aspect ratios may develop during the proccess. Usually, mesh quality is improved using an optimization scheme, for example solving for the quad internal angles [17], orientation or sizing [18]. Alternatively, the problem can be formulated as a Partial Differential Equation (PDE) problem using variations of the Laplacian [17, 19, 20] and the elliptic operator [21].

The methodology presented here for generating suitable quad meshes starts with a triangular tessellation followed by a splitting procedure [9] that produces a valid fully quadded mesh. The resulting mesh undergoes through a series of topological modifications that enable a significant reduction in the total number of irregular vertices while maintaining mesh validity. This is accomplished by detecting and readjusting distorted elements throughout the regularization process. In general, validity checking is performed in physical space while insertions, collapses and vertex movement is performed in the $[u, v]$ space of the underlying surface.

*Postdoctoral Associate, Aerospace Computational Design Labratory, 77 Massachusetts Avenue, 37-467, AIAA Member:Young Professional
†Principal Research Engineer, Aerospace Computational Design Labratory, 77 Massachusetts Avenue, 37-447, AIAA Member

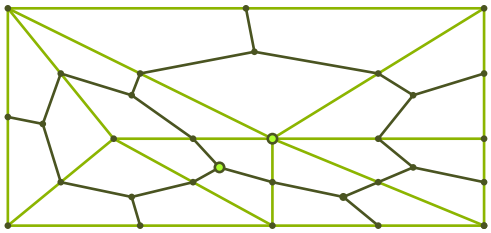
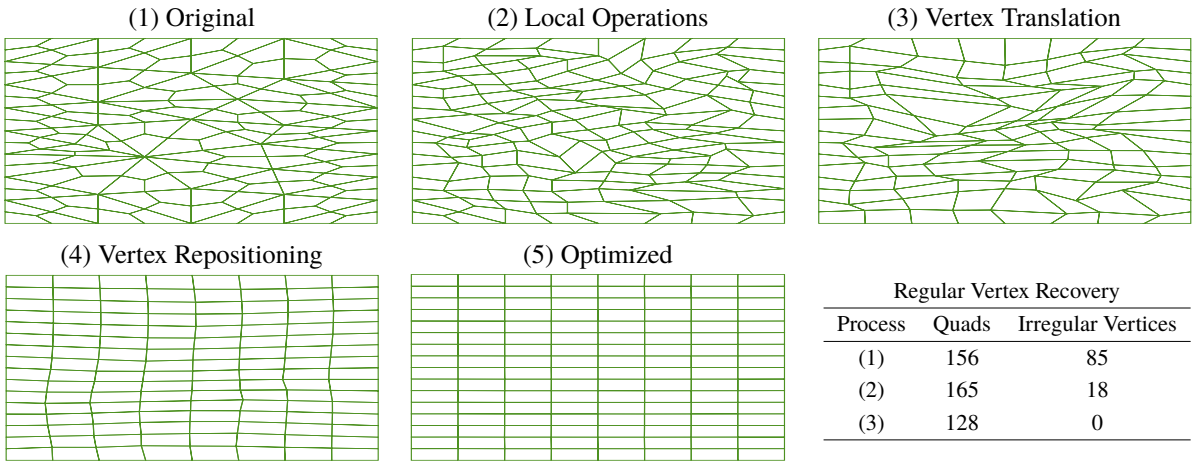


Fig. 1 Qudrangulation process using the Catmull and Clark [9] techinique. Each triangle is split into three quads by inserting a new vertex (barycenter) and three new sides (medians). The thicker lines (light green) show the original triangles and the highlighted vertices indicate high/low valences.



Regular Vertex Recovery

Process	Quads	Irregular Vertices
(1)	156	85
(2)	165	18
(3)	128	0

Fig. 2 Algorithm pipeline from mesh generation to optimization.

II. Pre-Processing: Mesh Generation and Regularization

We begin by briefly describing how to obtain a fully quadded mesh from a triangular tessellation using the Catmull and Clark [9] algorithm: given a triangular tessellation, for each triangle, we split the sides using the medians and insert another vertex at the barycenter (centroid). Then, as shown in Figure 1, three new quads are produced by linking the barycenter to the location where the median intersects each side. Although implementing this technique is straightforward and effective, the resulting mesh contains many irregular vertices (valence $\neq 4$): those that were originally there plus the new barycentric vertices which all have valence three (see Figure 1). This usually leads to undesirable meshes displaying very sharp or flat elements so it is necessary to introduce a regularization process that recovers as many regular vertices as possible.

In order to motivate the discussion, in Figure 2 we have produced an idealized mesh (an untrimmed surface with equal numbers of vertices on opposing patch sides) and show the whole mesh manipulation pipeline. The first step (local operations) which is discussed in Section II.A consists of applying systematically the basic element operations of swapping, collapsing and splitting and combinations of the same such as swap-collapse or double splitting. This process is followed by a vertex translation, which moves irregular vertices along the mesh allowing quads from far regions to interact with each other. Once there are no more possible operations (or the mesh is fully regular), we apply an iterative scheme for redistributing the vertices in order to provide a suitable mesh. This is discussed in Section II.B and finally, in III.A we present the challenges associated with a curvature driven optimization process.

A. Mesh Regularization

Here we present the element operations in detail and show examples of compositions of basic operations that allow for recovering regular vertices within a neighborhood (not all of them necessarily connected). We will use v_i to denote quad vertices, $val(v_i)$ their valences and a $(a,\ b)$ pair means a vertex pair of valences a and b respectively. Finally, the group (a, b, c, d) denote the valences of an ordered quad.

- **Swapping**: this process allows for exchanging high and low valences by changing the vertex pair forming the common side of any two adjacent quads. If (v_1, v_2) was the pair forming the common side which has now become

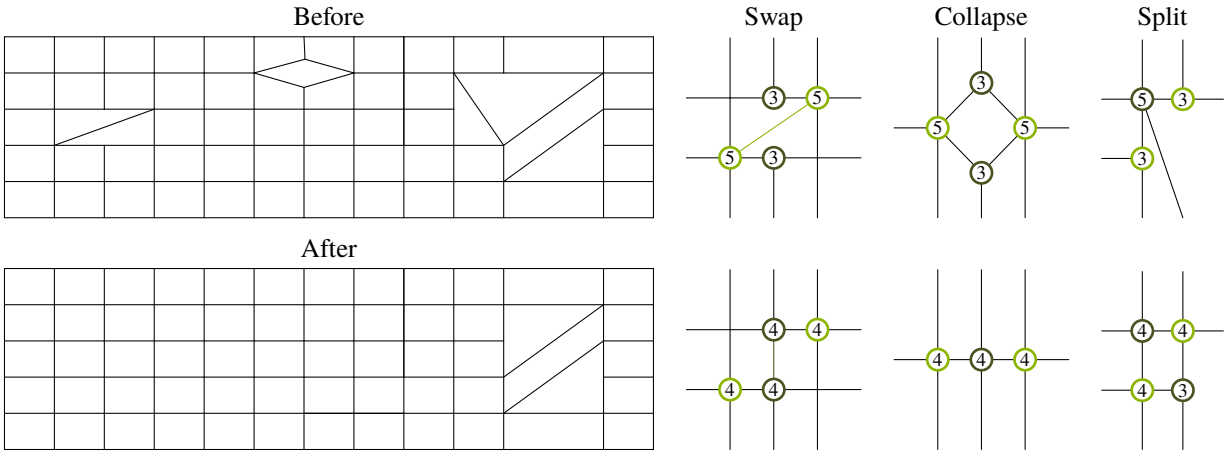


Fig. 3 The three basic vertex operations performed during regularization showing how valences exchange between quads. The numbers on the vertices denote the valence (regular vertices have valence four).

(v_3, v_4), then:

$$val(v_i) = val(v_i) - 1,\ i = 1, 2 \quad \text{and} \quad val(v_j) = val(v_j) + 1,\ j = 3, 4. \tag{1}$$

Hence, as shown in Figure 3, the perfect swap occurs for a $(5, 5)$ and $(3, 3)$ pair, producing two regular quads (all vertices of valence four). In practice, we don't require them to be perfect since, for example, swapping a $(5-5)$ edge for a $(3-4)$ edge results in a $(4-4)$ and $(4-5)$ pairs, thus increasing the topological regularity.

- **Collapsing:** eliminating a quad by merging two of its opposite vertices (see Figure 3). Given a quad with valences (v_1, v_2, v_3, v_4), collapsing (v_1) to (v_3) results in:

$$val(v_i) = val(v_i) - 1,\ i = 2, 4 \quad \text{and} \quad val(v_{13}) = val(v_1) + val(v_3) - 2. \tag{2}$$

Therefore, the ideal collapse occurs for a quad with valences $(3, 5, 3, 5)$ respectively $(val(v_2) = val(v_4) = 5 - 1$ and $val(v_{13}) = 3 + 3 - 2 = 4)$. Like for the previous case, this operation is suitable as long as there are three irregular vertices.

- **Splitting**: inverse operation to collapsing and it is applied when high valence vertices are linked to low valence vertices. Figure 3 shows a split which goes from three irregular vertices to one. In this case, the distance (in quads) between the vertices dictate the final valence distribution as illustrated in Figure 4.
- **Translation**: unlike splitting or collapsing, swapping edges preserves the number of quads. Hence, we can use this operation to move vertices along the mesh. For example, a $(4, 5)$ pair swapped with a $(3, 4)$ pair results in $(3, 4)$ and $(4, 5)$ valences. The total irregular vertices has not changed but the position has. In Figure 5 we show how this operation brings irregular vertices that were not connected closer to each other resulting in a regular region after two consecutive swaps.

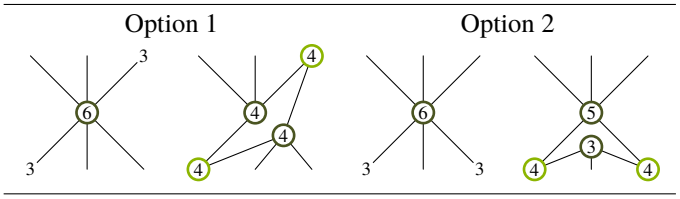


Fig. 4 Two splitting examples showing the role played by the relative position of the vertices for the final valence distribution. Option 1 results in a perfect split since the vertices are three quads away whereas in option 2, the split produced a pair $3, 5$ (vertices are two (four) quads away).

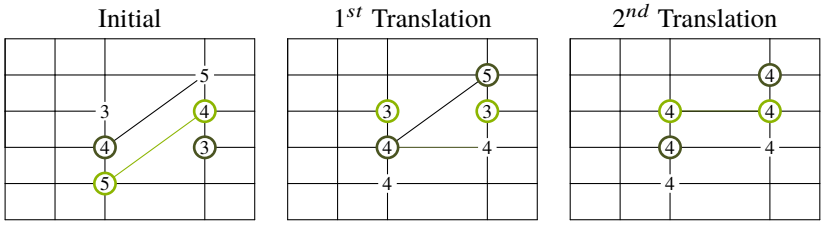


Fig. 5 Translation of an irregular vertex pair $(3,5)$ upwards along the mesh showing how after two movements, a suitable swap was found producing a regular region.

- **Composition**: The basic operations show that in order to improve the global mesh regularity, we need three or more irregular vertices to exchange valences effectively. However, notice that the three operations rely on the fact that these vertices are all contained within the neighborhood of a quad and in a precise configuration; suitable collapses need a vertex distribution where opposite vertices have low-low and high-high valences respectively. Swaps on the other hand, need the vertex pairs to be three vertex counts apart (clockwise or counter-clockwise). In Figure 6 we show a vertex star centered at S with all its surrounding quads and vertices. Star groups are employed during regularization to detect three or more irregular vertices within a region, not necessarily contained in a single quad or a quad pair. In Figures 7, 8, 9, 10 and 11 we illustrate several examples of effective composition of these operations that allow for improving the global mesh regularity.

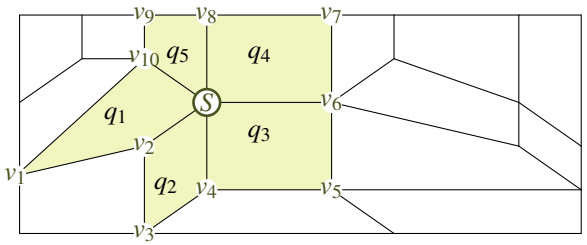


Fig. 6 Vertex star centered at S highlighting its surrounding quads and vertices. This data group allows us to study possible combinations of element operations within a region. For example, the pair (S, v_2) cannot see vertex v_6 from quad q_1 but star S sees the three irregular vertices so a double split (see Figure 10) can be performed.

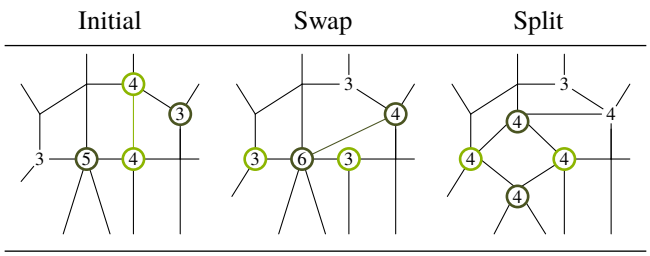


Fig. 7 Swap-split process using information from three different quads. The swap increases temporarily the number of irregular vertices but produces a scenario for optimal splitting, leaving one single irregular vertex (valence three). This operation is also suitable if rather than swapping a $(4,4)$ edge with a $(5,3)$ pair, we have a $(5,4)$, $(5,4)$ setup.

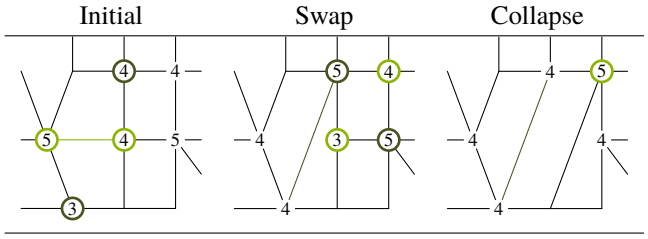


Fig. 8 Swap-collapse composition using two irregular quads. After swapping, the top right quad is almost fully irregular with opposite high valence vertices. Collapsing leaves a single irregular vertex of valence five. A similar result would follow if rather than having the quad $(4,5,4,4)$ for collapsing, we had a $(4,4,3,4)$ distribution.

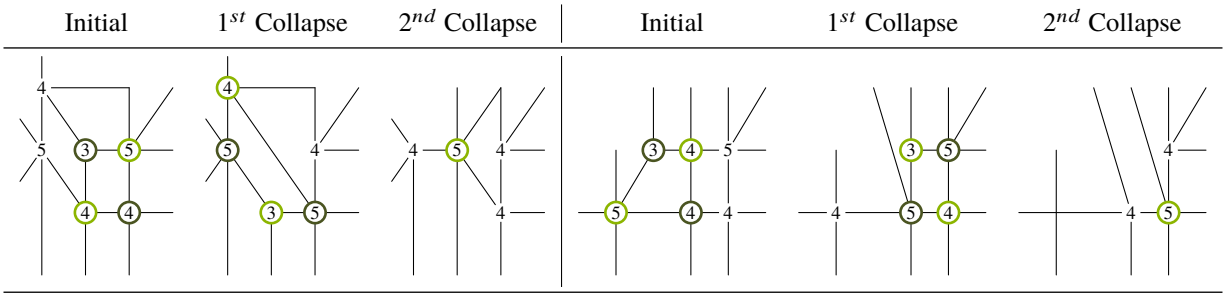


Fig. 9 Two double collapses: the first case starts collapsing the lower quad (valences $(4, 4, 5, 3)$) producing a (3,5) pair for the adjacent quad which now has valences $(3, 5, 4, 5)$. Collapsing leaves a single valence five vertex. The second case follows a similar methodology. Double collapses depend on the relative position of the irregular vertices and are applied when there are triples $(3, 5, 5)$ around two quads and has one $(3, 5)$ link.

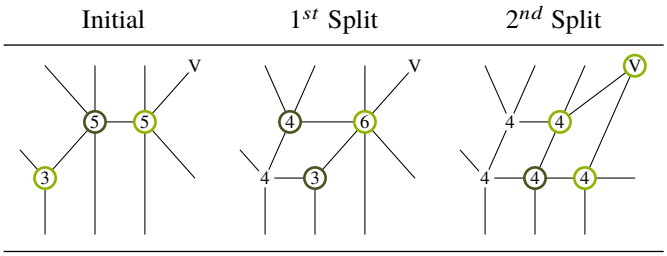


Fig. 10 Double split process: the first split produces a valence $(3, 6)$ link which can be further split through V (three quads distance), reducing the number of irregular vertices from three to one (or zero if V was valence 3). This operation is suitable as long as the $3, 5$ pair around the central vertex are at least one quad apart.

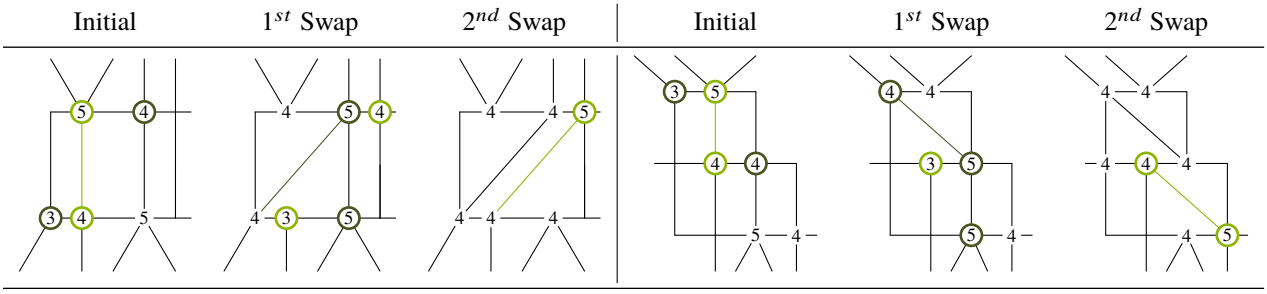


Fig. 11 Double swap operation using adjacent quads (first) and diagonal quads (second) leaving the local region with one (zero) irregular vertices. Double swaps between adjacent quads use triples $(3, 5, 5)$ which are two vertices counts apart. Diagonal swaps need a $(3, 5)$ link and the other valence five vertex opposite to the central vertex.

B. Vertex Repositioning

During the regularization process, every step is controlled by a mesh validity check ensuring that all the quad internal angles are within an admissible range and that there are no self-intersecting elements (see Figure 12, bottom). For a given quad, we extract the surface normal plane at the centroid and project all its vertices onto the plane (this is accomplished by getting the surface first derivatives at the point and performing a cross product). At each vertex, we compute the angles and the quad area using information from the other three vertices. This is illustrated in Figure 12 (top). If (A, B, C, D) denote the quad ordered vertices, then angle at A is obtained by adding $\angle DAC$ and $\angle CAB$. The signed area is computed as the sum of the areas of the two triangles multiplied by their orientation (with respect the normal direction). Invalid elements are detected when the sign of the triangles pair is different. In such cases, the quad and its neighbors undergo a vertex repositioning process attempting to produce a valid region.

Vertex repositioning depends on the information around each vertex: the new coordinates are the weighted average of all its linking vertices. For invalid elements, we look for the greatest angle and start moving that vertex. In general, this process requires repositioning several vertices (in the neighborhood) in order to find a suitable valid region. Note that the position movement is done in $[u, v]$, so that no projections are required.

For example, a swap involves two quads and six vertices so the moving region in this case would be the sum of all the vertex stars (see Figure 6). Figure 13 shows an example of the number of vertices that are involved in the process of area validation during a side swap operation. Notice that after swapping the vertices $(30 - 43)$ for $(19 - 31)$, one of the quads has a degenerate angle ($> 180°$) and in order to produce a valid element, several vertices are required to move. The actual computation of the vertex coordinates is described in Algorithm 1. In the event that invalid elements remain, the topological operation is rejected and the mesh is restored from its previous valid configuration.

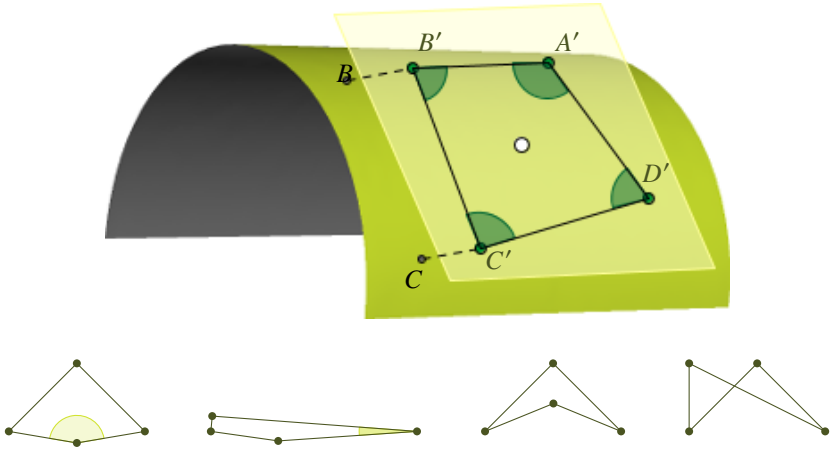


Fig. 12 Top: Quad internal angles at the surface normal plane through the centroid showing the vertex projection from the surface. Bottom: valid elements according to a user defined maximum and minimum angle tolerance and two examples of invalid elements: internal angles $> 180°$ and a self-intersecting polygon.

Algorithm 1 Weighted Average Vertex Repositioning

INPUT: Vertex ID (vID)
$star \leftarrow buildStar\ (vID)$ ▷ Surrounding vertices and quads (Figure 6)
$u = v = \Gamma = 0$
for $j = 1$ to $star \rightarrow links$ **do**
  $(s, t) \leftarrow surfaceUVs\ (links\ (j))$
  $(arc) \leftarrow arcLength\ (vID,\ links(j))$
  $u+ = arc \cdot s; v+ = arc \cdot t$
  $\Gamma+ = arc$
end for
$u = u/\Gamma; v = u/\Gamma$
$updateUV\ (vID,\ u,\ v)$
return

Since the same vertices may undergo several element manipulations during regularization, the maximum (minimum) angles should be relatively large (small). Otherwise, many steps are rejected leaving more irregular vertices than the "optimal" number. In practice, we allow angles within the interval range $(5°,\ 175°)$. However, this results in very distorted elements. Hence, once the regularization is complete, we apply an iterative technique that attempts to drive the quad internal angles as close to 90° as possible. This is described in Algorithm 2. In this case, since the input mesh is valid and we are just trying to adjust the angles, the process stops once it cannot reduce the angle interval any further.

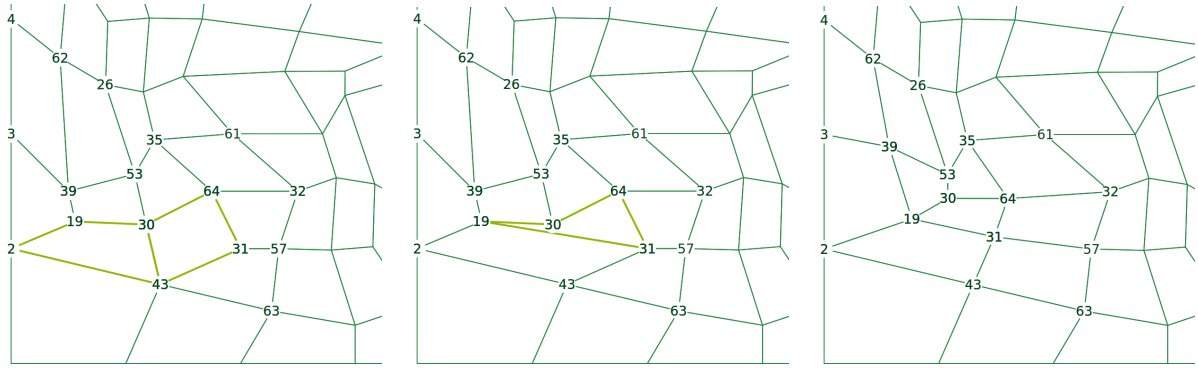


Fig. 13 Vertex repositioning during a side swap. The left image highlights the quads that will be swapped (side 30 – 43 will become 19 – 31). The middle image shows how the swap produces an invalid element (angle at 30 is > 180°) and in the last image we see how a valid mesh is recovered. All the labeled vertices where involved in the process and moved (if necessary).

Algorithm 2 Mesh Vertex Redistribution

$(\theta_{min}, \theta_{max}) \leftarrow (\ 5,\ 175)$
$(\beta_{min}, \beta_{max}) \leftarrow (\ 85,\ 95)$ ▷ Target angles
$incr = 1°$ ▷ Angle increment
while $validMesh = true$ **do**
  $count = 0$
  for $q = 1$ to $totalQuads$ **do**
    if $validArea\ (q, \theta_{min}, \theta_{max}) = true$ **then** $count++$
    end if
  end for
  $\theta_{min} += incr;\ \theta_{max} -= incr;$
  if $count = 0 \parallel \theta_{min} = \beta_{min} \parallel \theta_{max} = \beta_{max}$ **then**
    $validMesh = 0$
  end if
end while

In the middle set of images seen in Figure 14 we show two examples of applying the full vertex repositioning from Algorithm 2 after mesh regularization. Both cases resulted in a fully regular mesh which seems unstructured. Besides the negative visual effect, it also results in an unsuitable initial condition for any optimization process. On the other hand, the rightmost images in Figure 14 show how the mesh quality (angle-wise) has improved as a result of the iterative process described previously.

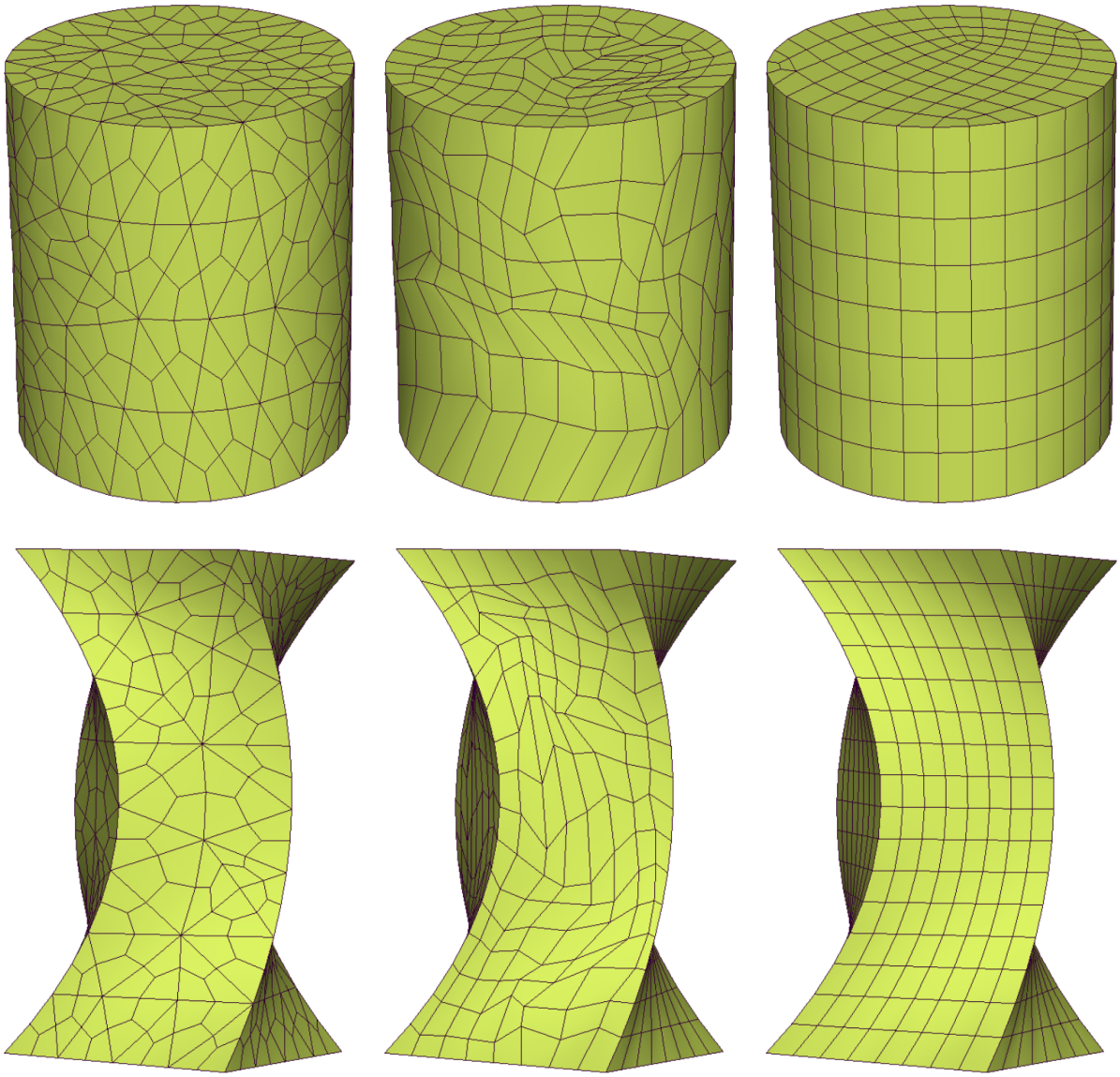


Fig. 14 A cylinder and a twisted body showing the original mesh (left), the regularization (middle) and after reallocating the vertices (right) using the methodology described in Algorithm 1.

III. Validation

We applied the mesh regularization technique over an aircraft body and for two different types of wings: with and without flaps. In Table 1 we show the results in terms of regular vertices for several tessellation spacings and surfaces. Notice the number of irregular vertices relative to the mesh doesn't grow with refinement. Also, the algorithm performance is not effected as the geometry becomes more difficult. In every case, the initial number of irregular vertices accounts for around 48% of the total vertices and after regularization we are left with around a 4%.

	Cylinder		Wing		Fuselage	
	Before	After	Before	After	Before	After
Quads	528	498	492	435	687	659
	1648	1398	1213	1123	1203	1156
	2943	2654	2976	2995	2553	2527
Irregular Vertices (%)	276 (47.8)	12 (2.2)	253 (48.6)	20 (4.3)	342 (48.3)	29 (4.3)
	751 (49.8)	42 (2.9)	555 (47.7)	50 (4.3)	591 (48.3)	49 (4.2)
	1325 (49.2)	68 (2.5)	1437 (47.8)	122 (4.0)	1244 (48.3)	121 (4.7)

Table 1 Mesh regularity before and after applying our regularization process (Section II.A) over three different surfaces and for three mesh refinements showing the number of quads and irregular vertices. The latter are displayed as totals and as a proportion of the total number of vertices.

It needs to be reiterated; the target for these quadrilateral meshes is structural analysis (specifically *Built-up Element Models*), so these results should not be viewed through a CFD lens. In a real sense, the task at hand is more difficult; it is harder to produce valid meshes for curved geometry when its applications require the element size to be coarser.

The initial triangulations used in Figures 14, 15, 17, 19 and 21 were generated by the tessellator in EGADS [22]. It generates watertight triangulations of BReps by first discretizing the BRep Edges, then performing the trimmed surface (BRep Face) triangulations. There is no notion of grading spacings because the tessellator is driven by being able to best represent the geometry with the fewest number of vertices/triangles. The technique used simply bifurcates regions that don't meet the user input criteria, which can obviously display abrupt linear spacing changes on the order of 2 or more. Other points include:

- The quadrilateral templating scheme described in [23] is disabled.
- Large interior angle deviations between neighboring triangles are allowed.
- Large triangle side spacing are also allowed generating triangulations that are far from equilateral.

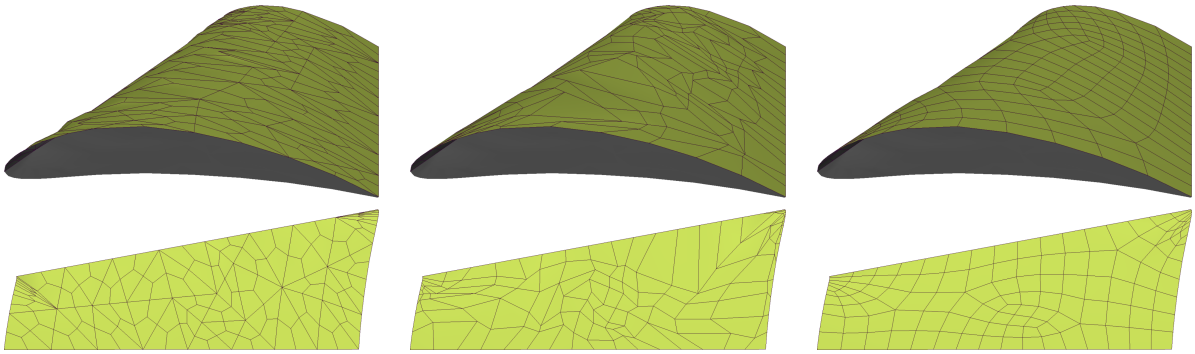


Fig. 15 A wing profile seen from above (top) and below (bottom) showing the original mesh, after regularization and the final mesh after repositioning the vertices.

In Figure 15 we show a wing profile at its three stages: mesh generation, regularization and redistribution of vertices. Remember that the vertices at the bounds of the BRep Face are fixed.

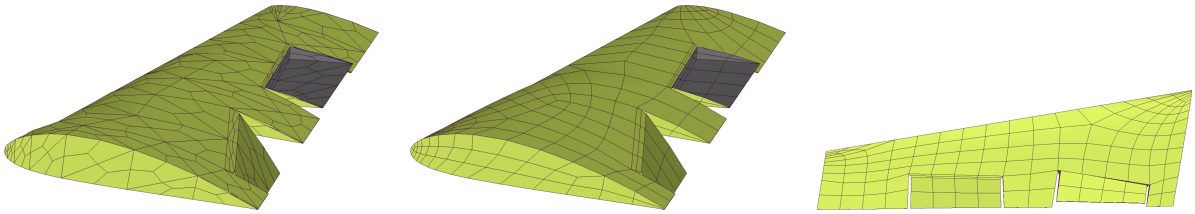


Fig. 16 A wing profile with two flaps seen from above (first two) and below (right) before and after applying the mesh regularization technique.

Figure 16 shows similar result but in this case, the wing has two flaps. The final meshes contain few irregular vertices becoming suitable for structural analysis applications. The aircraft in Figure 17 shows the meshes at the initial and final stage. Notice that the original mesh is highly irregular, especially on the fuselage. This is because the initial EGADS triangulation is having difficulties resolving the large amount of curvature with the requested coarse spacing. The end result is patches of denser triangles and anisotropic large triangles. Once the initial triangles are split into quadrilaterals those from the anisotropic triangles can display rather acute angles.

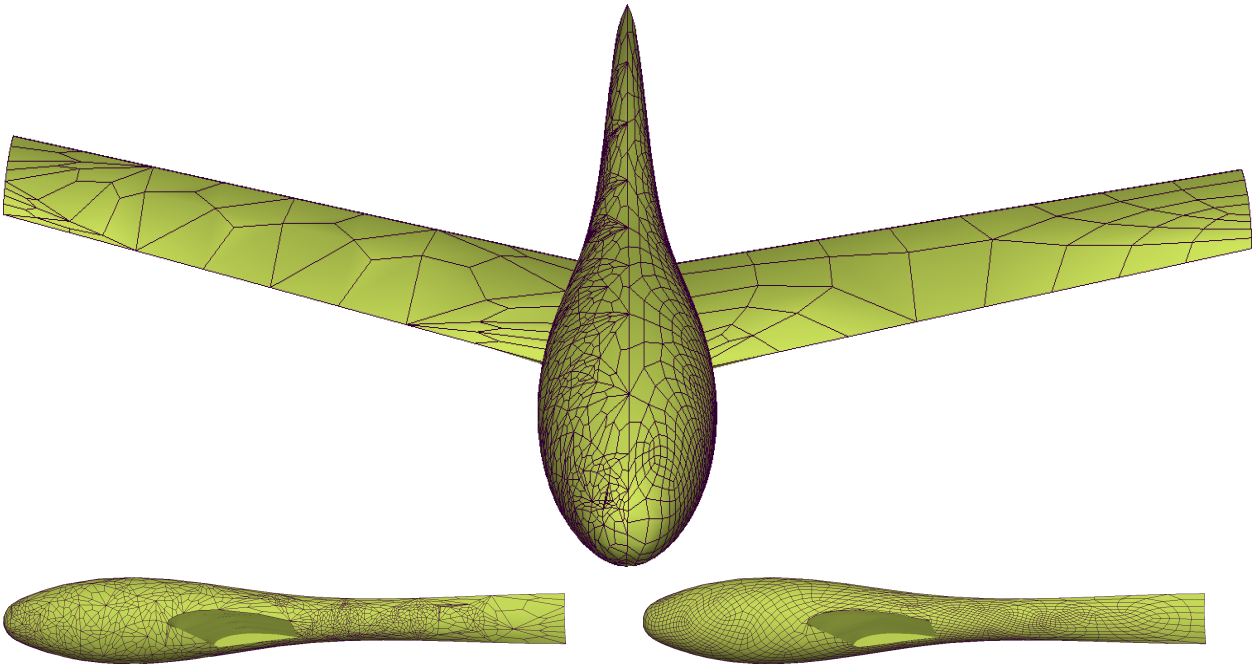


Fig. 17 An aircraft (top) split in half showing both meshes: the original obtained with the triangle splitting process and after applying the mesh regularization technique. The bottom images show the fuselage seen from the side.

After regularization, we are left with only valences five or less, allowing for a better distribution along the surface. This can be appreciated in the bottom images of Figure 17 where we show a side-view of the fuselage. Generally, a single irregular vertex affects the quad alignment over a large portion of the mesh [16, 23], presenting extra difficulties for any angle based optimization process. In the following, we discuss a curvature driven optimization approach and the challenges associated with this problem.

A. Surface Curvature and Optimization Challenges

A high quality mesh produced from trimmed surfaces must respect both the trimming and the surface's underlying curvature. For a regular mesh, the natural quad structure suggests that sides should be aligned with the local curvature lines. However, as it was shown in Figure 17, irregular vertices affect the alignment not only within its surrounding

quads but within a larger region. This presents an important challenge when deciding a suitable objective function for a curvature driven optimization process. In addition, as it will be shown later, the surface bounds play also a major role in the final mesh geometry.

The EGADS [22] geometry kernel has the full geometry representation of each surface so we can directly evaluate the curvatures at any surface point. For every regular vertex we construct the surface normal plane and project all its linking vertices onto this plane. The quad alignment is driven by the side that is closest to either curvature direction and it is used as pivot for assigning a target direction for each of the other three sides. This is illustrated in Figure 18. On the other hand, since irregular vertices can't align with the curvature lines (two directions = four sides) these vertices are *floating* degrees of freedom and we only quantify the internal angles in order to prevent sharp or flat elements.

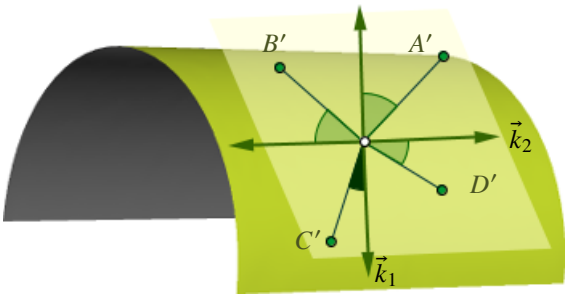


Fig. 18 A regular vertex linked to the vertices (A, B, C, D) showing its tangent plane and the projected vertices (A', B', C', D'). The darker angle denotes the side which is closest to one of the principal curvature directions and from which we decide how to compute the error with respect the other sides.

Element sizing on the other hand, should be set according to the curvature magnitude: zero curvature (flat) regions should allow relatively large sizing whereas high curvature zones should employ smaller elements. The authors in [2, 24] proposed an estimation of the side size based on the local curvature:

$$s = \frac{1}{\kappa}\sqrt{40\left(1-\sqrt{1-1.2\epsilon}\right)} = \frac{1}{\kappa}g(\epsilon), \quad \kappa = \text{curvature} \tag{3}$$

and the size of ϵ is decided based upon the allowed error for replacing the arc of the osculating circle centered at a vertex by the chord length.

After applying this curvature driven optimization approach over fairly simple surfaces we have noticed that both the surface boundary and prevailing irregular vertices limit the performance of the optimizer. In Figure 19 we show a simple flat surface which was cut producing a tilted shape. The mesh was optimized based on a uniform sizing approach (since it is flat there is no curvature to follow) and although the visual effect is very positive, the mesh alignment doesn't follow an orthogonal system. Hence, if it had been driven towards, for example the Cartesian axis, it would have destroyed its uniformity. Consider now the twisted surface shown in Figure 14. In this case, even though the final mesh is uniform, curvature lines are completely ignored. This is reflected in Figure 20 where we show the curvature lines at every vertex overlapped with the uniform mesh. Finally, the performance of our scheme is strongly affected by the mesh regularity. In Figure 21 we show a slice of a toroidal surface with a relatively fine mesh spacing in several stages: the initial mesh, after regularization and vertex repositioning and after optimization. The final mesh still contains several irregular vertices and after optimization, the quads orientation have not improved. This is highlighted in the rigthmost image where we show an overlap of both meshes.

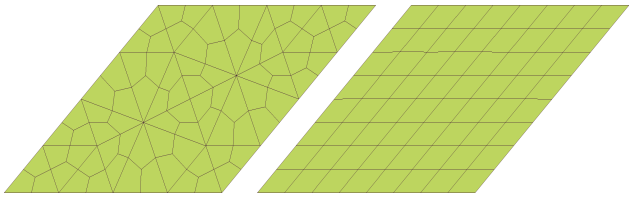


Fig. 19 A flat surface with a non orthogonal boundary structure before and after mesh smoothing. The final mesh is structured and has uniform sizing but its internal angles are not in a 90° configuration.

IV. Conclusions and Ongoing Work

In this paper we have presented a mesh generation technique that produces almost regular quadrilateral meshes. Using the triangle splitting approach from [9], we obtain a fully quadded mesh which then undergoes topological changes in order to recover regularity. Our approach shows that in general, it is possible to reduce the number of

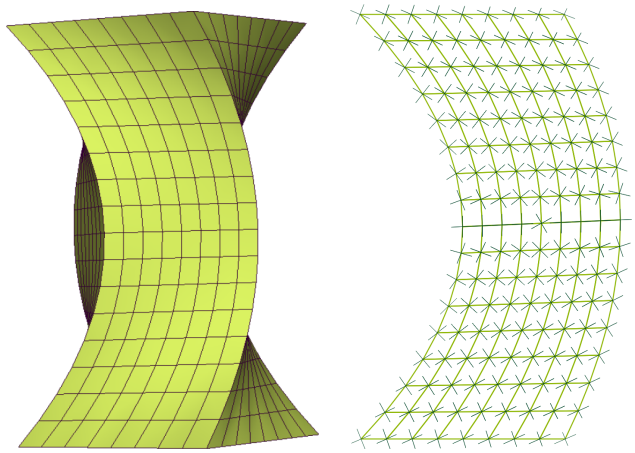


Fig. 20 A twisted surface where the curvature lines cross at the center and the natural alignment is in contradiction with the trimmed surface geometry. The dark arrows denote the curvature lines at each vertex and they are superimposed with the original mesh (ligher lines) in order to highlight the angle deviation.

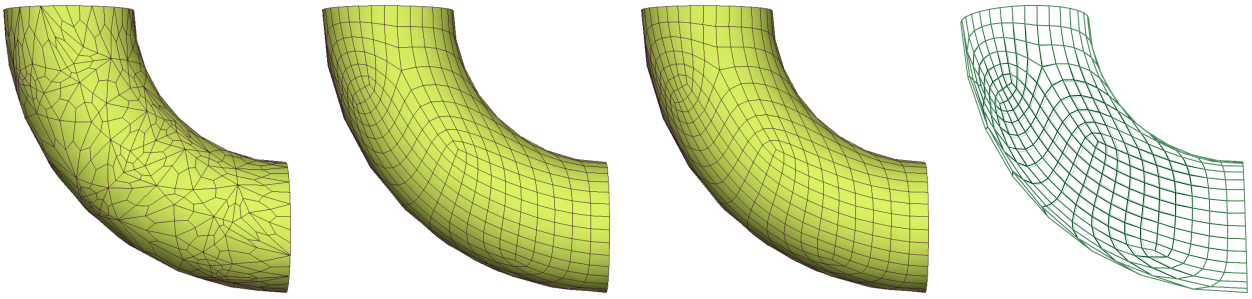


Fig. 21 A slice of a toroidal surface shoing the following meshes: initial (1), after regularization and vertex repositioning (2), optimized mesh (3) and an overlap of (2) and (3).

irregular vertices from around 50% to 5%. However, the technique employed for vertex valence exchange is not yet optimal. The element operations use information within vertex stars (Figure 6) and although the scheme successfully eliminates any irregular star made of three or more irregular vertices, it still leaves pairs of high-low valences which could balance out but are not detected within this data group. In addition, the remaining irregular vertices should be driven towards surface regions with high curvature changes where they won't affect the mesh alignment and in some cases, even improve it. The regularization process is followed by an iterative vertex repositioning method that attempts to redistribute the mesh points. This process eliminates distorted elements arising when performing mesh manipulations. However, this approach does not use orientation information from the surface geometry and therefore should be followed by an optimization process. Our first attempt on driving the mesh towards the curvature lines showed that the surface trim has a strong impact on the overall mesh quality (see Figures 19 and 20). In addition, as shown in Figure 21, the remaining irregular vertices also limit the performance of the optimizer. In general, we have not yet found a suitable objective function that produces satisfactory results over more complex geometries accounting for several irregular vertices like those shown in Figures 15, 16 and 17. Future work will focus on optimization objective functions and appropriate balancing of angle and sizing control while respecting the trimming (bounds that do not align with isoclines or principal directions of curvature).

Acknowledgements

This work was funded by the CAPS project, AFRL Contract FA8050-14-C-2472: "CAPS: Computational Aircraft Prototype Syntheses"; Dean Bryson is the Technical Monitor.

References

[1] Talbert, J. A., and Parkinson, A. R., "Development of an automatic, two-dimensional finite element mesh generator using quadrilateral elements and Bezier curve boundary definition," *International Journal for Numerical Methods in Engineering*, Vol. 29, No. 7, 1990, pp. 1551–1567.

[2] Blacker, T. D., and Stephenson, M. B., "Paving: A new approach to automated quadrilateral mesh generation," *International Journal for Numerical Methods in Engineering*, Vol. 32, No. 4, 1991, pp. 811–847.

[3] White, D. R., and Kinney, P., "Redesign of the paving algorithm: Robustness enhancements through element by element meshing," *6th International Meshing Roundtable*, Citeseer, 1997, pp. 323–335.

[4] Owen, S. J., Staten, M. L., Canann, S. A., and Saigal, S., "Q-Morph: an indirect approach to advancing front quad meshing," *International Journal for Numerical Methods in Engineering*, Vol. 44, No. 9, 1999, pp. 1317–1340.

[5] Lee, C. K., and Lo, S., "A new scheme for the generation of a graded quadrilateral mesh," *Computers & structures*, Vol. 52, No. 5, 1994, pp. 847–857.

[6] Borouchaki, H., and Frey, P. J., "Adaptive triangular–quadrilateral mesh generation," *International Journal for Numerical Methods in Engineering*, Vol. 41, No. 5, 1998, pp. 915–934.

[7] Edmonds, J., "Paths, trees, and flowers," *Classic Papers in Combinatorics*, Springer, 2009, pp. 361–379.

[8] Remacle, J.-F., Lambrechts, J., Seny, B., Marchandise, E., Johnen, A., and Geuzainet, C., "Blossom-Quad: A non-uniform quadrilateral mesh generator using a minimum-cost perfect-matching algorithm," *International journal for numerical methods in engineering*, Vol. 89, No. 9, 2012, pp. 1102–1119.

[9] Catmull, E., and Clark, J., "Recursively generated B-spline surfaces on arbitrary topological meshes," *Computer-aided design*, Vol. 10, No. 6, 1978, pp. 350–355.

[10] Bommes, D., Lempfer, T., and Kobbelt, L., "Global structure optimization of quadrilateral meshes," *Computer Graphics Forum*, Vol. 30, Wiley Online Library, 2011, pp. 375–384.

[11] Daniels, J., Silva, C. T., Shepherd, J., and Cohen, E., "Quadrilateral mesh simplification," *ACM transactions on graphics (TOG)*, Vol. 27, No. 5, 2008, p. 148.

[12] Kinney, P., "Cleanup: Improving quadrilateral finite element meshes," *6th International Meshing Roundtable*, 1997, pp. 437–447.

[13] Tarini, M., Pietroni, N., Cignoni, P., Panozzo, D., and Puppo, E., "Practical quad mesh simplification," *Computer Graphics Forum*, Vol. 29, Wiley Online Library, 2010, pp. 407–418.

[14] Peng, C.-H., Zhang, E., Kobayashi, Y., and Wonka, P., "Connectivity editing for quadrilateral meshes," *ACM Transactions on Graphics (TOG)*, Vol. 30, ACM, 2011, p. 141.

[15] Bozzo, A., Panozzo, D., Puppo, E., Pietroni, N., and Rocca, L., "Adaptive Quad Mesh Simplification." *Eurographics Italian Chapter Conference*, Citeseer, 2010, pp. 95–102.

[16] Verma, C. S., and Suresh, K., "A robust combinatorial approach to reduce singularities in quadrilateral meshes," *Procedia Engineering*, Vol. 124, 2015, pp. 252–264.

[17] Zhou, T., and Shimada, K., "An Angle-Based Approach to Two-Dimensional Mesh Smoothing." *9th International Meshing Roundtable*, 2000, pp. 373–384.

[18] Freitag, L. A., "On combining Laplacian and optimization-based mesh smoothing techniques," *ASME applied mechanics division-publications-amd*, Vol. 220, 1997, pp. 37–44.

[19] Shontz, S. M., and Vavasis, S. A., "A Mesh Warping Algorithm Based on Weighted Laplacian Smoothing." *IMR*, 2003, pp. 147–158.

[20] Zhang, Y., Bajaj, C., and Xu, G., "Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow," *Communications in Numerical Methods in Engineering*, Vol. 25, No. 1, 2009, pp. 1–18.

[21] Knupp, P. M., "Winslow smoothing on two-dimensional unstructured meshes," *Engineering with Computers*, Vol. 15, No. 3, 1999, pp. 263–268.

[22] Haimes, R., and Drela, M., "On the construction of aircraft conceptual geometry for high-fidelity analysis and design," *50th AIAA Aerospace sciences meeting including the new horizons forum and aerospace exposition*, 2012, p. 683.

[23] Haimes, R., and Aftosmis, M. J., "Watertight Anisotropic Surface Meshing Using Quadrilateral Patches," *13th International Meshing Roundtable*, 2004, pp. 311–322.

[24] Lee, C. K., "On curvature element-size control in metric surface mesh generation," *International Journal for Numerical Methods in Engineering*, Vol. 50, No. 4, 2001, pp. 787–807.