1 Unfolding Convex Polyhedra

1.1 The Problem

From Demaine and O'Rourke's "A Survey of Folding and Unfolding in Computational Geometry" (2005):

> A classic open problem is whether (the surface of) every convex polyhedron can be cut along some of its edges and unfolded into one piece without overlap. [...] It seems folklore that the answer to this question should be yes, but the evidence for a positive answer is actually slim. [DO05]

My goal is to determine whether every discrete closed convex surface is developable.

1.2 What goes wrong

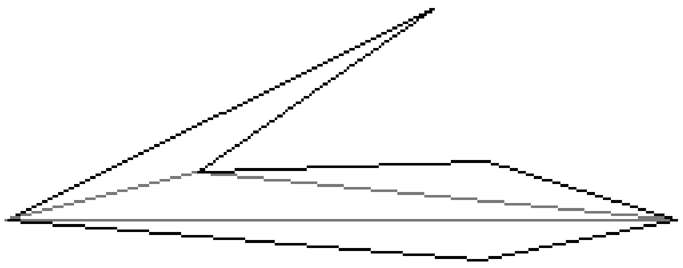


Fig. 1. An unfolding of a slim tetrahedron

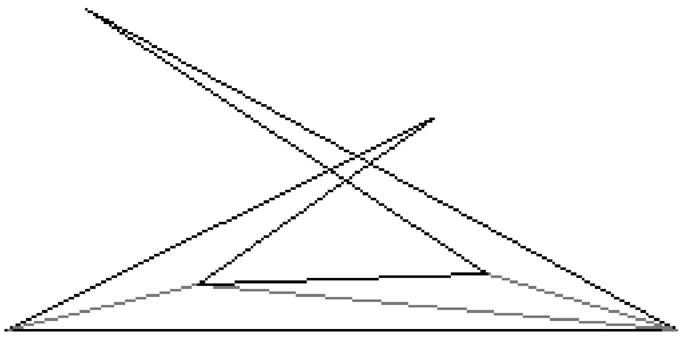


Fig. 2. Namiki's self-intersecting unfolding of the same tetrahedron

In Figure 2, a skinny tetrahedron discovered by Namiki [NF94] shows an overlapping unfolding. A valid unfolding of the same model is demonstrated in Figure 1. This demonstrates that even a very simple model can have an invalid development.

As the number of vertices in a convex mesh grows, the probability of a random unfolding of the surface being illegally self-intersecting goes to one[O98]. However, no example has yet been found of a convex surface which has *no* legal unfolding.

1.3 Local Convexity

1.3.1 Building the cut tree

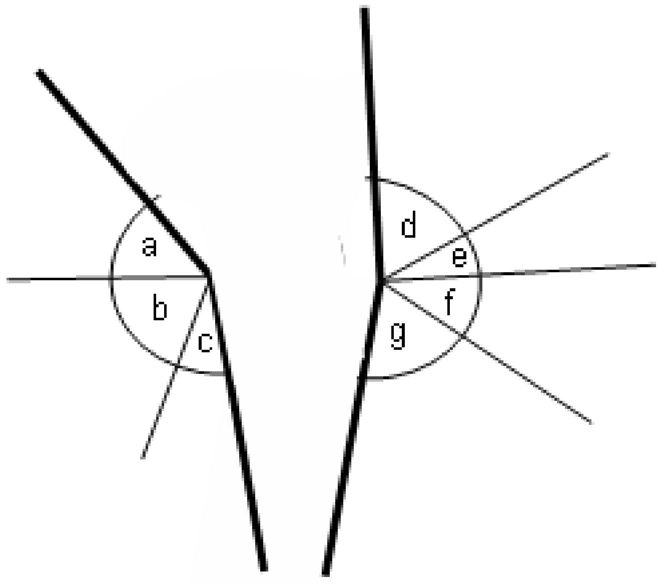


Fig. 3. A vertex with seven incident faces, with their incident angle values a–g. Two edges are cut in the cut-graph, giving two fans; their incident angles are (a+b+c) and (d+e+f+g). The vertex has a total incident angle of max((a+b+c), (d+e+f+g)), which in this example is greater than π; the vertex is locally concave.

Each face incident to a vertex meets that vertex at the endpoints of two edges. The angle between these two edges is the face's *incident angle* at that vertex. Where multiple faces sharing the vertex remain connected in the unfolded net, these faces form a *fan* around the vertex (although a fan may consist of only one face); the incident angle of the fan is the sum of the incident angles of the faces. I define the *total incident angle* of a vertex in a cut-graph as the maximum of the incident angles of all the fans at that vertex (Figure 3.)

I've adopted the terms *local convexity* and *local concavity* to describe how much the unfolding of a particular vertex bent inwards or outwards in the plane. A locally convex unfolded vertex would have an incident angle of less than or equal to π radians. A locally concave vertex would have more.

It occurred to me that:

Local concavity is essential for conflict.
So minimize local concavity.

I reasoned that the spanning tree which minimizes the local concavity of the unfolded geometry must be that of a valid unfolding. My reasoning was simple: Any two curves in the plane which proceed from a common point and only bend away from each other can never intersect.

I hypothesized that the unfolding from a cut-graph which minimizes local incident angles to values uniformly below π was guaranteed to have no conflicts. If it were provably possible to build such a cut-tree for every mesh then the existence of such a construction would prove that all closed convex polyhedra were unfoldable.

1.3.2 Defining the convex tree

A *convex tree* is the shorthand term used informally for the next few pages to describe any cut-graph which

- is complete (spanning)
- has no loops (no valid cut-graph on a genus zero surface can have a loop)
- ensures that no vertex has a total incident angle greater than π

I use the term *twin* to describe the unfolded image of a face, edge or vertex from the original source mesh. When a polygon is unfolded, its twin is a congruent polygon with a new position and a subset of the original's connectivity. The twin lies in the plane of the unfolding.

It can be slightly unclear to refer to unfolding an edge or vertex, as they are in a sense not discrete entities but only the borders of the sets of points which define the unfoldable planar subsets we call faces. When an edge $\overline{PQ}$ is 'unfolded', it creates two twins[1] ach twin is twin to the original source edge, altered only in orientation–they could rightly be called triplets. which can be described as the 'left' ($\overline{P_L Q_L}$) and 'right' ($\overline{P_R Q_R}$) twins. The left and right twins will unfold to different places in the plane; the placement of $\overline{P_L Q_L}$ will be defined by the unfolding of the source edge's left-hand face, while $\overline{P_R Q_R}$ will follow the right-hand face.

When a vertex V of degree n is 'unfolded' (more precisely, when a vertex is attached to faces which are unfolded) the vertex may split into up to n twins, each associated with the now-separated twins of the original's incident faces. These can often be identified as V_i. In some discussions, when referring to an edge which has been cut, it is can be useful for discussion to refer to V_L and

[1] E

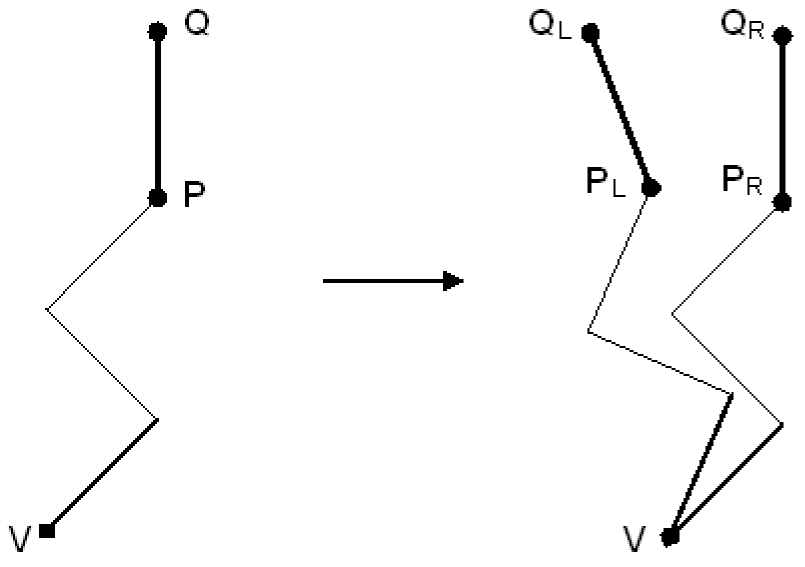


Fig. 4. An edge being cut 'unfolds' to left and right twins which lie akimbo in the plane

V$_R$, the endpoints of the left-hand and right-hand twins of the split edge.

1.3.3 Quest for the convex tree

I tried to think of as many ways as possible to prove that a convex tree exists for every closed convex mesh.

- Could I give a set of rules which, when applied repeatedly to an arbitrary cut-graph, would be guaranteed to iteratively produce a convex tree?
- Could I build a voronoi-style map of the convex hulls of small blossoms of unfolded geometry?
- Could I just say, "this is a minimum spanning tree problem" and find that some kind soul had already solved it? After all, it's just a matter of finding the spanning tree which minimises the local concavity of every vertex in the unfolding.
- Could I show that a convex tree can be built recursively?

1.3.4 Iterative evolution towards a convex tree

I began by thinking in terms of repairing an invalid cut-graph. The design would be to convert any cut-graph to a convex tree through the repeated application of discrete operations which would preserve the invariants of the graph. These operations would be used to convert vertices with local concavity into vertices which were locally convex.

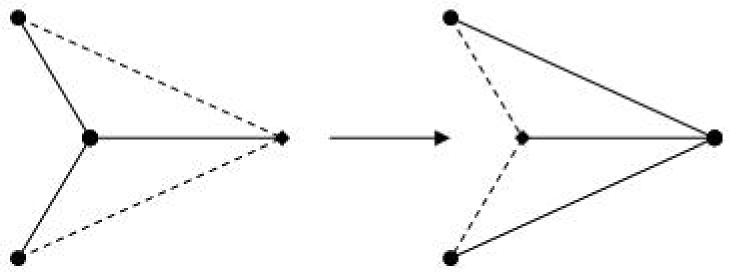


Fig. 5. A branch becomes a leaf; a leaf becomes a branch

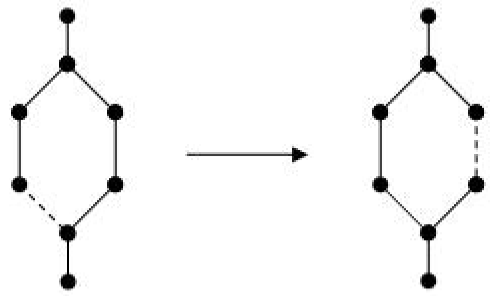


Fig. 6. A new edge is added to the graph, creating a loop which is then broken by removing another edge

I identified three editing operations:

- Make a leaf into a branch (Figure 5)
- Make a branch into a leaf (Figure 5)
- Add one edge and remove another (Figure 6)

For any locally concave vertex in the cut-tree, there are two options for resolution:

- *Cut another incident edge*
  Cut another edge of the source mesh, adding it to the graph to break up the vertex whose summed incident angles exceed π. The tree is a spanning tree, so adding another cut edge to the tree will induce a loop. The loop must be repaired by removing some other edge from the tree (which will have the effect of leaving that edge uncut on the original source mesh.) Can it be shown that any induced loop must be repairable?
  This operation adds an edge because of a local concavity, ie., a vertex with incident angle $>\pi$. A loop would be unresolvable if every edge on that loop were already necessary to split another vertex whose incident angle was too high.
- *Convert the vertex into a leaf of the cut-tree*
  Remove $n-1$ (all but one adjacent) edges from the node. This requires that all but one of the adjacent nodes weren't depending on those edges to keep

them legal. Another $n - 1$ edges will have to be added elsewhere, as their removal will have broken the tree into up to n separate pieces.

1.3.5 Where the convex tree approach breaks down - a counterexample

Unfortunately, a counterexample can be constructed which shows that the convex tree approach cannot always apply. If a vertex has three incident edges

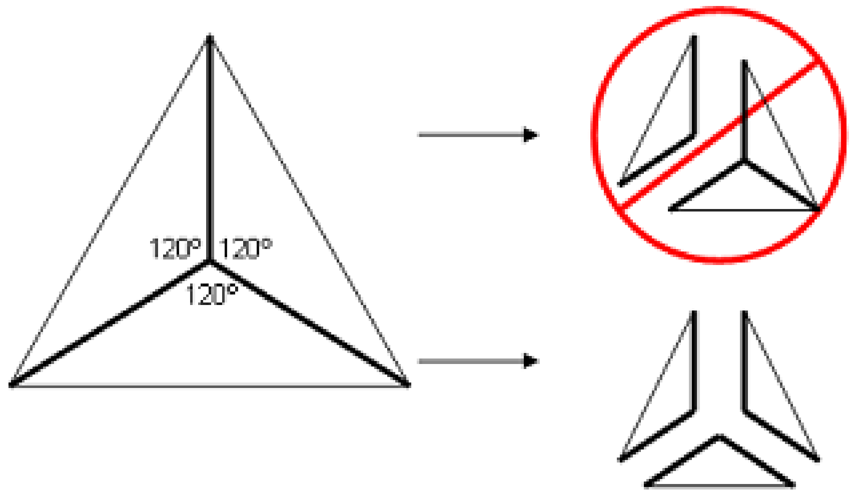


Fig. 7. A trivalent vertex which must be separated into triplets

meeting at $2\pi/3$ (Figure 7) then all three edges must be cut to generate a convex tree; leaving any edge uncut would give an incident angle of $4\pi/3$, which would exceed the convexity limit of π. But placing two such meshes side-by-side (Figure 8) would cause a loop to be formed in the cut-graph, separating two triangles from the rest of the unfolding illegally.

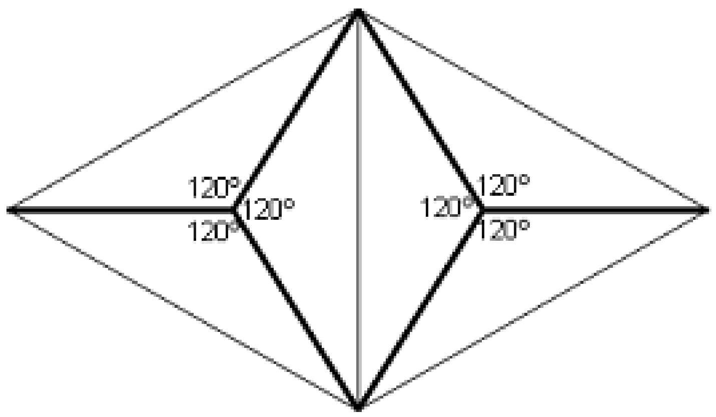


Fig. 8. Two triplet vertices create an illegal loop in the cut-graph

The convex tree approach is thus clearly too strong a restriction on the incident angles of the unfolded vertices.

1.4 The Anchorable Convex Hull

I next tried a new direction: a proof by recusion, in which I would show that I could evolve one stable state to another in a way that was guaranteed to terminate and which could grow from the seed of a single triangle. Could I show that a set of rules existed which could iteratively evolve a carefully-described singleton of convex unfolded geometry, by gluing faces to the singleton until it stabilises again into a new figure that obeys the same rules?

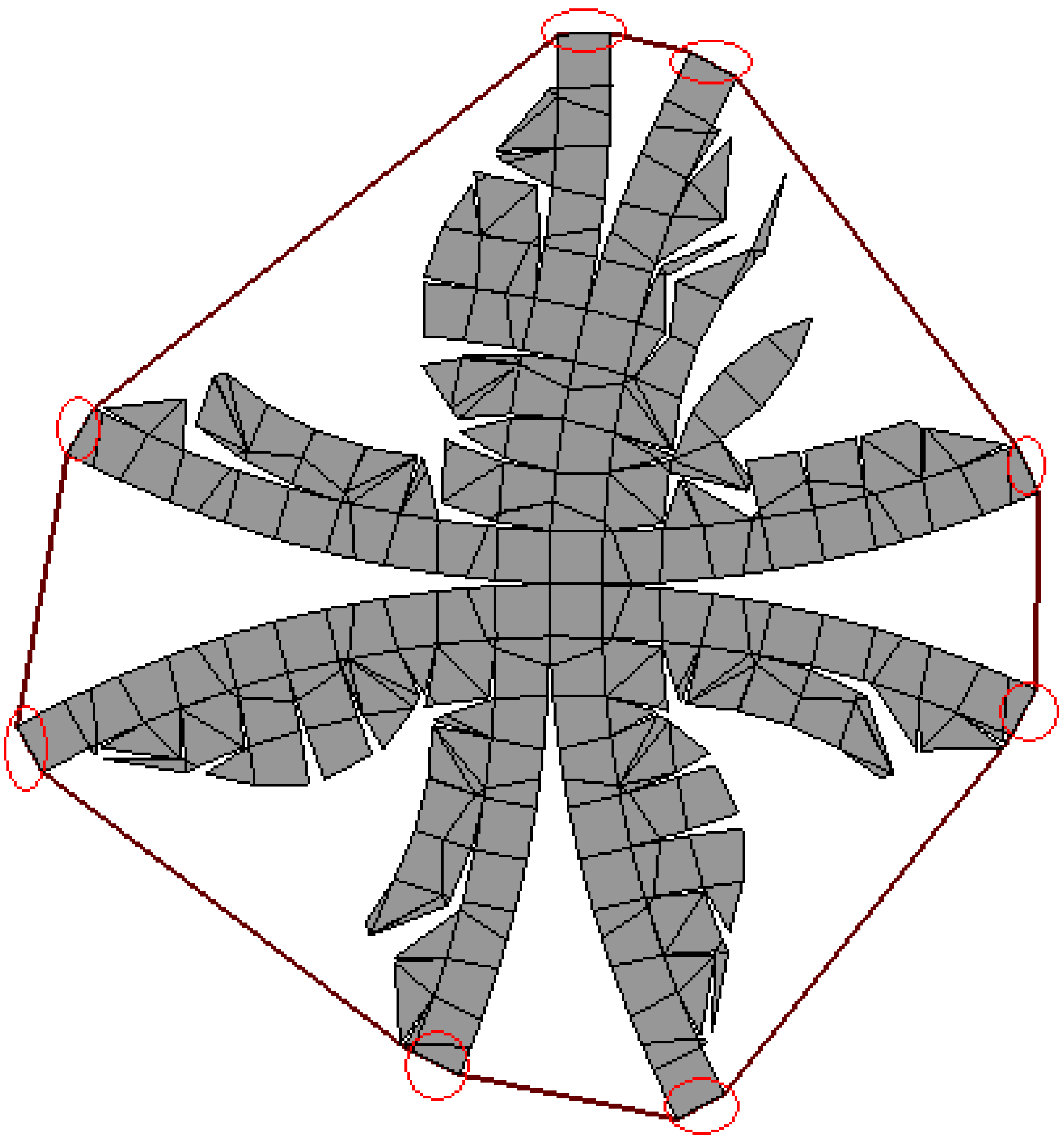


Fig. 9. An ACH (red) with anchorages (circled) for the unfolding a sphere

The singleton I had in mind was a convex subset of the unfolded geometry, a subset whose convex hull would have certain 'nice' attributes which would allow me to use it as a convenient stepping-stone to the next larger convex form. This approach depended on finding a nested series of convex hulls within the unfolding.

I define an *Anchorable Convex Hull* ("ACH") (Figure 9) as a convex planar polygon with the following traits:

- The ACH is the convex hull of a valid unfolding of a portion of the geometry of the source mesh.
- Certain edges of the ACH are *anchorages*. An anchorage is an edge of the ACH defined by two vertices of a source-mesh face within the ACH, lying along that edge. Informally, an anchorage is where another face could be 'glued' to the outside of the ACH. For every anchorage there is exactly one face in the source mesh which could be glue to that anchorage.
  Note that not all edges of the ACH are necessarily anchorages. As an unfolding blossoms outwards from a central seed face its geometry will crack and spread apart. The cracks will define some of the edges of the ACH; such edges will not be anchorages.
- There is no face in the source mesh which is not inside the ACH which shares an edge with any face within the ACH that is not an anchorage edge. Every face which can be glued to the ACH at any internal edge, an edge that isn't an anchorage, must be so glued.
- Neighboring anchorages will be share a vertex. If two anchorages A and B lie adjacent left-to-right on the ACH then the right-hand endpoint of A will be the right-hand unfolded twin V_A of some vertex V; the left-hand endpoint of B will be the left-hand unfolded twin V_B of V as well.

In the assembly of an ACH I will also use two construction operations:

- *Tasks* form the core of an advancing-front algorithm for ACH construction
- *Gaps* are inter-anchorage separations, complex tasks which will require special resolution

The final rule of the definition of the Anchorable Convex Hull has interesting implications. By requiring that neighboring anchorages share their endpoints, a strong upper bound on the angle between each pair of anchorages is found:

Consider any two neighboring anchorage edges A and B, whose source-mesh twins meet at the shared vertex V (Figure 10.) Edge A ends in vertex V_A; edge B begins with vertex V_B. Between them lies some third edge E that has been added to the cut-graph of the unfolding. Let αbe the sum of the incident face angles of the faces between A and E and the faces between E and B. α is thus the maximum possible inner angle between A and B in the unfolding plane. By construction, because A and B lie on edges of a convex hull it can

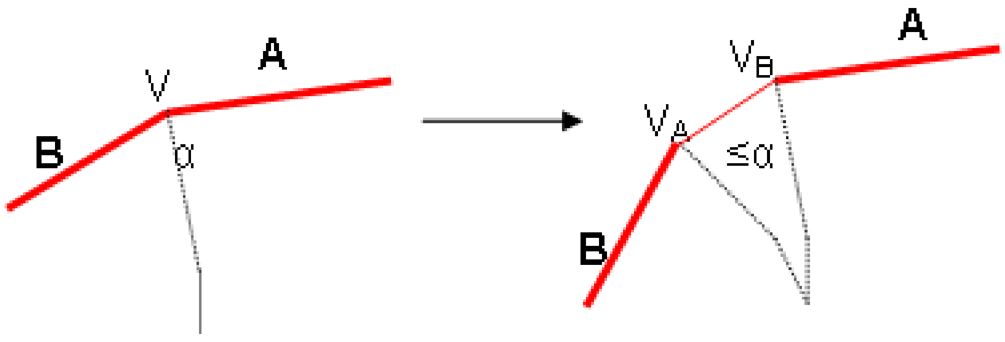


Fig. 10. As A and B are unfolded, the angle between them is reduced

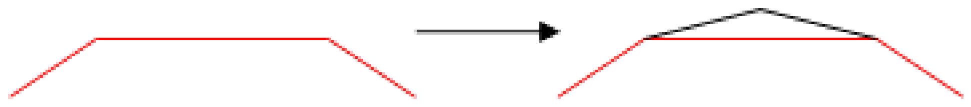
be known that $\alpha \leq \pi$.

The angle between the unfolded twins of A and B must actually be α - AD($\overline{\Phi_{VP}}$), where Φ_{VP} is the cut-tree branch which grows from V into the interior of the ACH and AD($\overline{\Phi_{VP}}$) is the sum of the angle deficits of every vertex on that path.

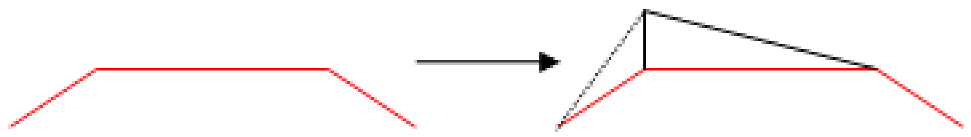
This also guarantees that the operation of gluing a face to an anchorage on the ACH will never introduce a conflict in the unfolding. In fact, the ACH construction ensures that faces could be glued to every anchorage on the ACH and none of them would conflict, with each other or with any face within the ACH. This is ensured because at every corner of each anchorage is a vertex shared with the next neighboring anchorage. The faces F_A and F_B to be glued to neighboring anchorages share a common vertex V and their anchor edges A and B meet at at most α radians; $\alpha + \angle(F_A \text{ at V}) + \angle(F_B \text{ at V}) \leq 2\pi$, because AD(V) $\leq 2\pi$. Thus the unfoldings of F_A and F_B can never overlap.

By construction any ACH contains a valid unfolding. It remains only to be shown that there exists a set of rules which can always evolve an ACH into another ACH; that will prove that an ACH can be constructed for the complete unfolding of any mesh, thereby showing that any suitable mesh can be unfolded.

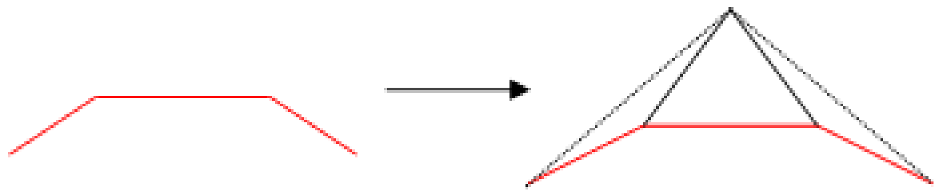
Consider the addition of a single face to an anchorage. This will create one of three scenarios (up to symmetry):



(1)
The anchorage edge is replaced by two new anchorages. No new tasks are created. The result is a new valid ACH, guaranteed to contain no conflicts.



(2)
One edge of the new triangle lies on the outline of the convex hull but the other edge lies within it. The anchorage edge is replaced by one new anchorage and one new task. The task's goal will be to fill in the gap created to one side of the anchorage; in this example, counter-clockwise.



(3)
The opposing vertex of the new triangle extends the convex hull but neither opposing edge of the triangle lies on the convex hull. The anchorage edge is replaced by two new tasks. The goal of each task will be to fill in the gaps to either side of the former anchorage.

This introduces the concept of a 'task'. A *task* is a job which must be completed through a series of gluing operations which will yield a new, valid ACH. A task is created when a face is glued to the unfolded mesh which extends the convex hull but adds an internal edge, within the bounds of the ACH and not on its border, to which another face which has not yet been glued on might be added which would fall within, or across, the border of the ACH. (Recall that such a gap must be repaired before the figure can once again be called an ACH.)

Tasks are oriented. A given task seeks to fill a gap in the convex hull in either the clockwise (CW) or counter-clockwise (CCW) directions.

Consider a pair of neighboring anchorages A and B in an Anchorable Convex Hull, where B lies counter-clockwise from A. Examine the case where gluing a face F_A to A has lifted the convex hull beyond the edge B, creating a CCW

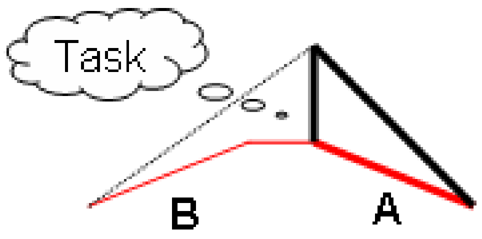


Fig. 11. Adding a face creates a task

task to be resolved.

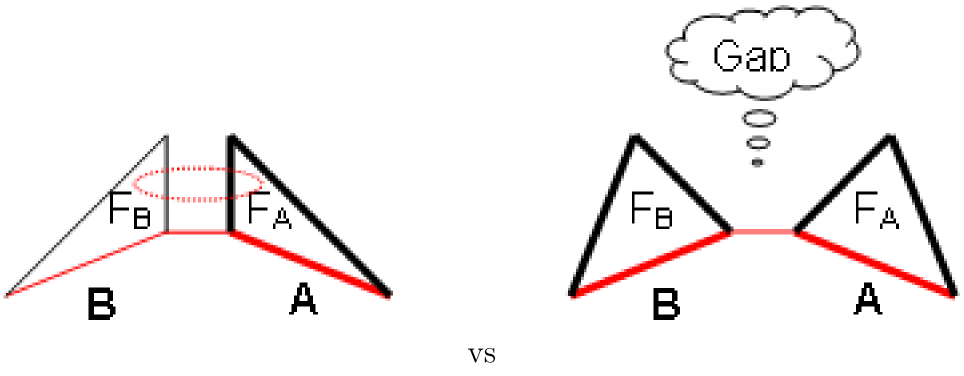


Fig. 12. If the two new faces do not share an edge then a gap is created

We know that the anchorage B is free; we know that we can safely glue F$_B$ to B without fear of introducing a conflict. The three possible scenarios hold once more and the new face could create a new CCW task. If F$_B$ creates no new task then this task is done; if F$_B$ creates a new CCW task, it can be resolved recursively.

It is possible that the CW edge of F$_B$ is not the CCW edge of F$_A$ (Figure 12.) If so then we have introduced a 'gap' in the border of the convex hull. A *gap* is a space along the ACH between two filled anchorages. Gaps are a new kind of task.

Recursive resolution of a task can be seen informally as a task 'travelling' around the outer edge of the convex hull. At each new anchorage, the task either terminates or carries on; perhaps leaving gaps behind in its wake. As a task is propagated around the hull, it will always be able to advance because we know that the hull's anchorages are waiting–until the task has wrapped around the convex hull and approaches itself from the other side. As the task reaches itself from the other side of the ring, it is transformed into a gap.

We must now fill in each gap across some vertex V with the fan of faces which lie between F$_B$ and F$_A$ and which have V as one corner. This operation is guaranteed to not induce conflicts, because V has positive angle deficit. We need only ensure that the new figure complies with the other rules of the ACH.

Unfortunately, it is difficult to prove that the face fan at V conforms to the rules of the Anchorable Convex Hull.

1.4.1 Insights from the Anchorable Convex Hull

The work above would seem to show (slightly reflectively) that any convex mesh whose unfolding contains multiple nested convex sets of edges is always unfoldable. Intuitively, one subset of such meshes would be meshes generated by marching-cubes or octree-based algorithms; such meshes often display latitudinal loops of edges ranged along a central axis.

This would constitute a new class of unfoldable polyhedra in the literature, although describing it with a definition that is more than implicit may be difficult.

1.4.2 Where the Anchorable Convex Hull approach breaks down

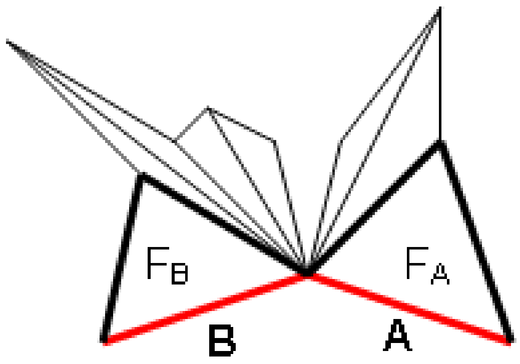


Fig. 13. A gap whose internal faces will be difficult to fit to the rules of the ACH

Gaps can develop into ‘complex gaps’, in which multiple faces fill in a gap, perhaps crossing the boundary of the ACH, perhaps embedded within it.

Unfortunately, the problem of resolving a gap is much more difficult than the problem of resolving a task. With tasks we knew that we were operating around the ACH and could derive useful preconditions that guaranteed resolution. But with gaps we have lost some of that supporting data, and I cannot see a way to ensure that the process of filling a gap can be shown to generate a figure which conforms to the rules of the ACH.

A more tractable flaw with the ACH approach is that it does not inherently enforce an ordering upon the faces of an unfolding, which can lead to unpre-

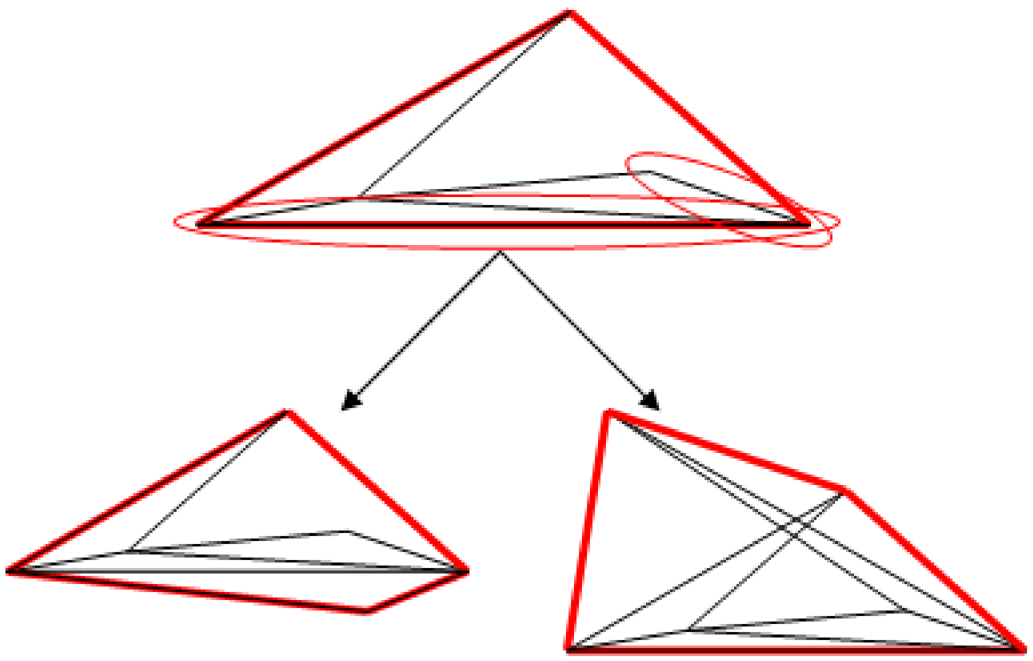


Fig. 14. Indeterminate order has created a job whose resolution can introduce a conflict

dictable behavior (Figure 14. For example, in Namiki's slim tetrahedron, the choice of the third face to add has created a clockwise task which introduces a conflict with the fourth. If the fourth face had been added before the third, no conflict would have resulted. This indicates that the rules of the ACH are not quite sufficiently strong or precise to guarantee that there will be no collisions in an unfolding.

1.5 Abandoning convexity in favor of angular restrictions

We will consider an edge $\overline{PQ}$ to be added to a cut path with V at its tip (Figure 15.)

The branch extends along a path $\overline{\Phi_{VP}}$, the ordered list of vertices from V to P. When the source mesh is unfolded, $\overline{\Phi_{VP}}$ splits into twinned sets of edges, left- and right-handed copies of the flattened source. The left-hand twins of vertices P and Q will be denoted P_L and Q_L; the right-hand twins, P_R and Q_R.

The angle deficit of vertex V is written AD(V); the total angle deficit of the path $\overline{\Phi_{VP}}$ is

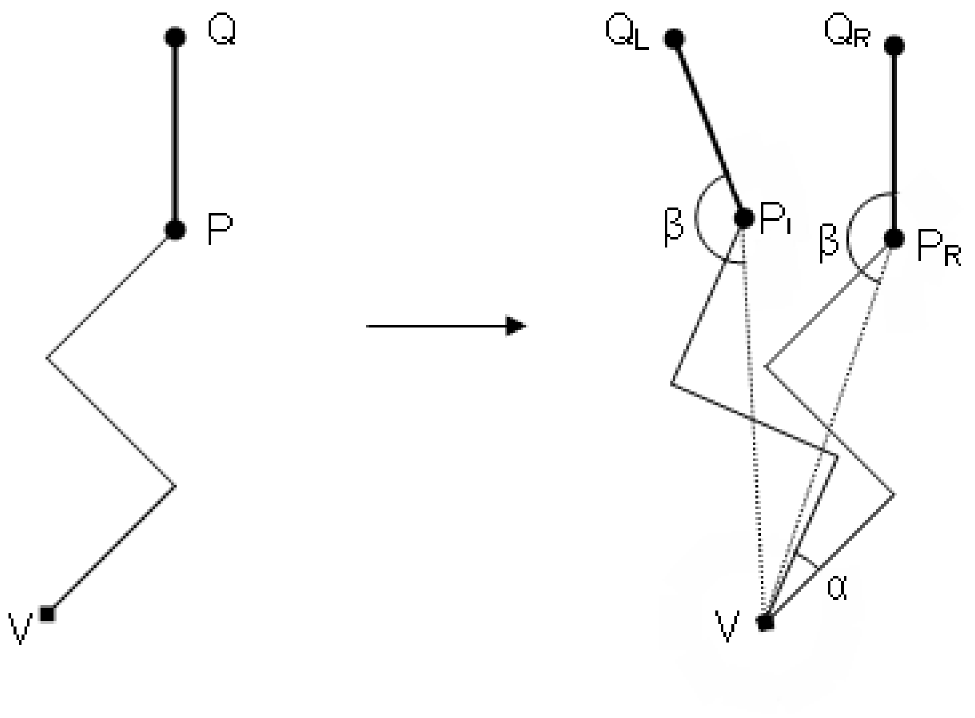


Fig. 15. Alpha is the summed angle deficit of the path; beta is the greatest variation of the edge from the path

$$AD(\overline{\Phi_{VP}}) = \sum_{v=V}^{v=P} AD(v) \tag{1}$$

Because the angle between $\overline{P_L Q_L}$ and $\overline{P_R Q_R}$ increases by AD(v) for each v in $\overline{\Phi_{VP}}$, we can state that

$$\angle(\overline{P_L Q_L}, \overline{P_R Q_R}) = AD(\overline{\Phi_{VP}}) \tag{2}$$

Note that this reduces trivally to

$$\angle(\overline{P_L Q_L}, \overline{P_R Q_R}) = AD(V) \tag{3}$$

if $\left\|\overline{\Phi_{VP}}\right\| = 1$.

The definition of β may seem to be slightly counterintuitive. One might at first think that $\angle$(VP$_L$Q$_L$) must equal $\angle$(VP$_R$Q$_R$), but this would be inaccurate.

The two edges do diverge by AD(Φ), but then they also diverge from each other by AD(P). How this angle deficit is distributed across the two edges depends on the local geometry of the faces to either side of the edge $\overline{PQ}$.

There exists a relationship between ∠(VPQ)[2] t can be misleading to write "∠(VPQ)", because it is unclear in which plane such an angle would be measured. I will therefore define any expression of the form *∠(ABC)*, where A, B and C are vertices on the surface of the source mesh, as the minimum of the angles ∠($A_LB_LC_L$) and ∠($A_RB_RC_R$), both of which are well-defined in the unfolding plane. These angles are not oriented and thus cannot exceed π. and the possibility of intersection. This relationship allows us to describe a state of a cut-graph which guarantees a valid unfolding.

This relationship can be formulated as

Lemma 1 *If ∠(VPQ)≥π/2 and* $\|VP_L\|$=$\|VP_R\|$ *then it is impossible for* $\overline{P_LQ_L}$ *to intersect* $\overline{P_RQ_R}$.

Proof: An affine transformation can be constructed (up to symmetry) which maps the unfolded twin of V to the origin, P_L to (0,1) and Q_L, P_R and Q_R to the positive X half of the XY plane. We shall call these point O, A, B, C and D, respectively.

Let

$$\begin{aligned} \alpha = && AD(\overline{\Phi_{VP}}) We are interested in the range 0 \leq \alpha \leq \pi. \\ \beta = && min(\angle(VP_LQ_L), \angle(VP_RQ_R)) \\ k = && \left\|\overline{AB}\right\| \end{aligned} \tag{4}$$

C and D can be expressed in terms of A and B, respectively, rotated about the origin by α radians. Let:

$$\begin{aligned} C = && (sin(\alpha), cos(\alpha)) \\ D = && (cos(\alpha) * B_x + sin(\alpha) * B_y, -sin(\alpha) * B_x + cos(\alpha) * B_y) \end{aligned} \tag{5}$$

Solving the system of linear equations

$$\begin{aligned} \overline{AB}(t) = && A + t(B - A) \\ \overline{CD}(u) = && C + u(D - C) \end{aligned} \tag{6}$$

for t, u, yields:

[2] I

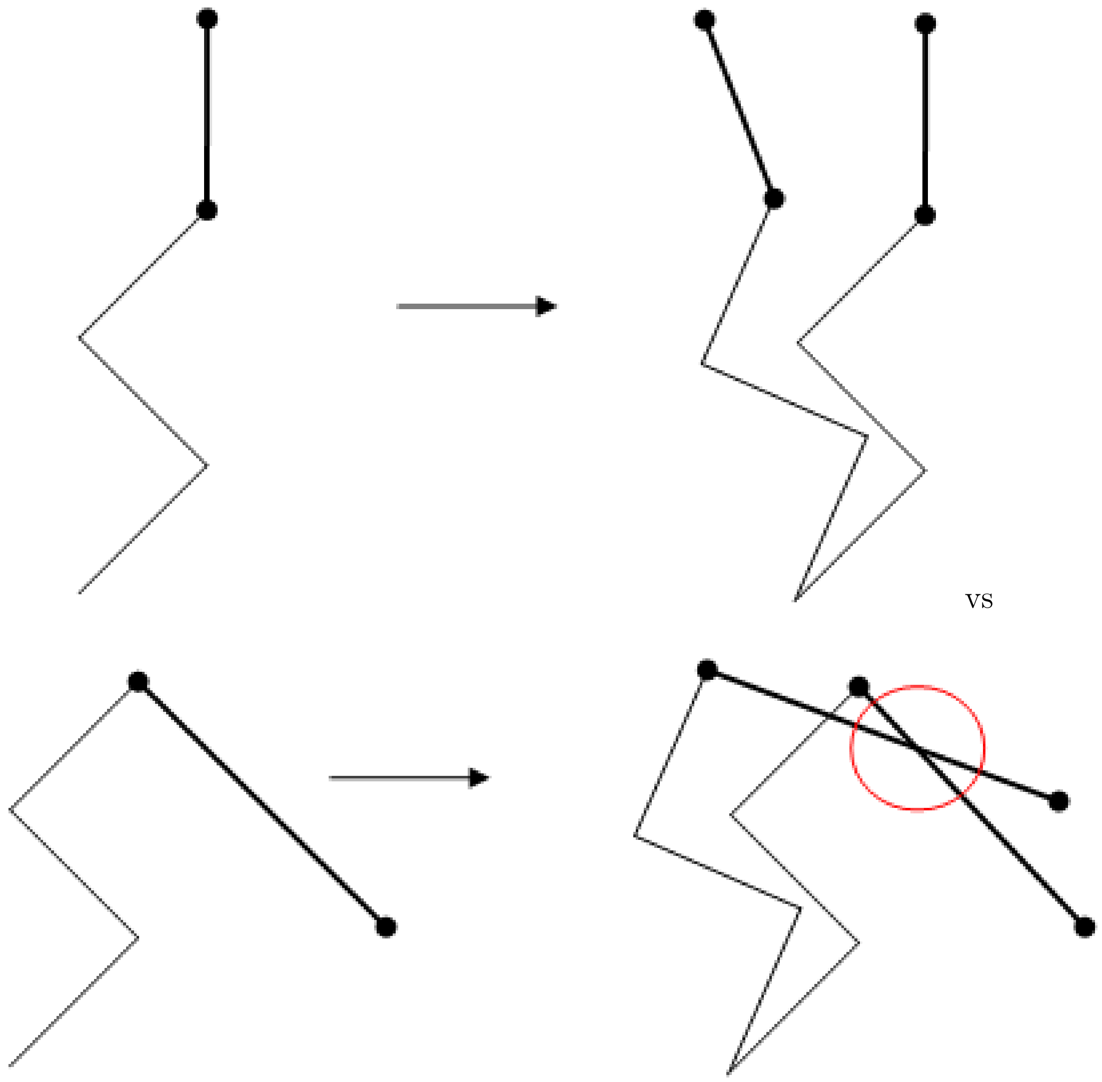


Fig. 16. The angle of the edge being cut partially determines the possibility of intersection

$$
\begin{aligned}
t &= \frac{1 - B_y + B_x tan(\alpha/2)}{B_x^2 + (B_y - 1)^2} \\
u &= \frac{1 - B_y + B_x(cot(\alpha) - csc(\alpha))}{B_x^2 + (B_y - 1)^2}
\end{aligned}
\tag{7}
$$

Rewriting B as A+k*(cos(β), sin(β)) = (k*cos(β), 1+k*sin(β)) yields

$$
\begin{aligned}
t &= \frac{cos(\beta)tan(\alpha/2) - sin(\beta)}{k} \\
u &= \frac{-cos(\beta)tan(\alpha/2) - sin(\beta)}{k}
\end{aligned}
\tag{8}
$$

If 0 $\leq\alpha\leq\pi$ then $\tan(\alpha/2)$ will always be positive. If $\pi/2 \leq\beta\leq\pi$ then $\cos(\beta)$ will always be negative and $\sin(\beta)$ will always be positive. If 0 $\leq\alpha\leq\pi$ and $\pi/2 \leq\beta\leq\pi$ then t $\leq$ 0; the two line segments cannot intersect.
□

Lemma 2 *Given a valid cut-branch $\overline{\Phi_{VP}}$ with no other leaf nodes on $\overline{\Phi_{VP}}$, if $\angle$(VPQ)$\geq\pi/2$ and $\|VP_L\|=\|VP_R\|$ then the unfolding of $\overline{PQ}$ cannot intersect the unfolding of any edge in $\overline{\Phi_{VP}}$.*

Proof:

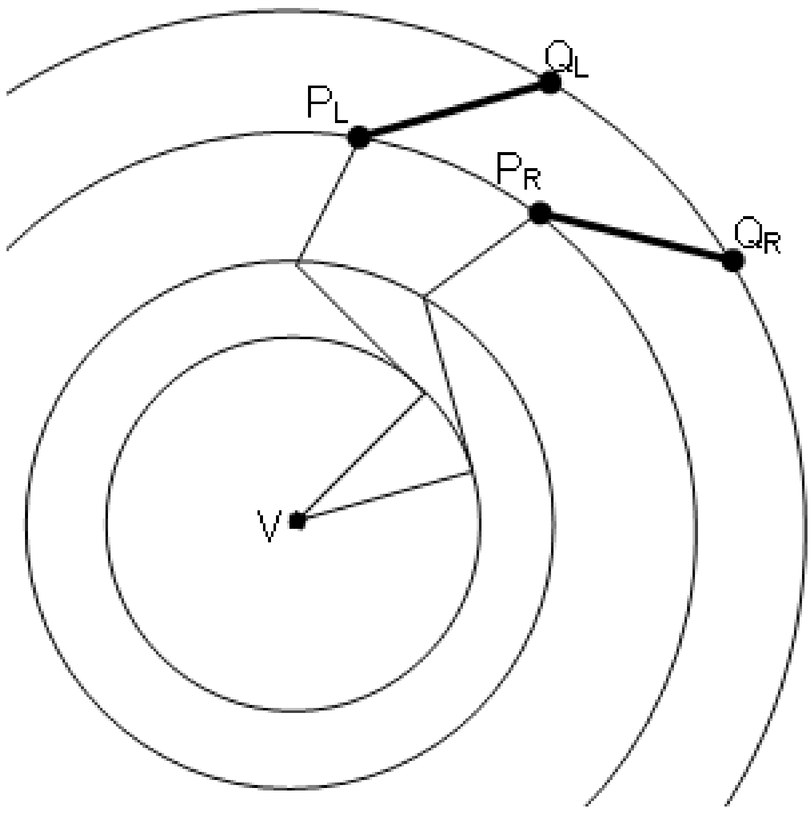


Fig. 17. Nested circles bound regions of intersection

Referring to Figure 17, we see that $\beta\geq\pi/2$ and $\left\|\overline{VP_L}\right\| = \left\|\overline{VP_R}\right\|$ implies that $\overline{P_LQ_L}$ and $\overline{P_RQ_R}$ will lie outside the line tangent to the circle of radius $\left\|\overline{VP_L}\right\|$ centered at V.

$\beta\geq\pi/2$ implies that $\left\|\overline{VQ_L}\right\|>\left\|\overline{VP_L}\right\|$ and $\left\|\overline{VQ_R}\right\|>\left\|\overline{VP_R}\right\|$.

Therefore each pair of unfolded edges in the cut-branch $\overline{\Phi_{VQ}}$ lies between circles of progressively larger radius centered at V.

Thus the unfoldings of $\overline{PQ}$ can never intersect any edge in $\overline{\Phi_{VP}}$.
□

Conjecture 1 *If a branch $\overline{\Phi}$ of the cut-graph has multiple leaf nodes and we wish to extend $\overline{\Phi}$ by the addition of a new root edge $\overline{PQ}$, it is sufficient that* *there exist a single $V \varepsilon \overline{\Phi}$ such that $\angle(VPQ) \geq \pi/2$ to ensure that there is no possibility of a conflict.*

The conjecture is false:

Counter: Lemmas 1 and 2 both depend on $\|VP_L\| = \|VP_R\|$. If this condition is not held then the lemmas do not apply.

Unfortunately, multiple leaf nodes makes the length constraint virtually

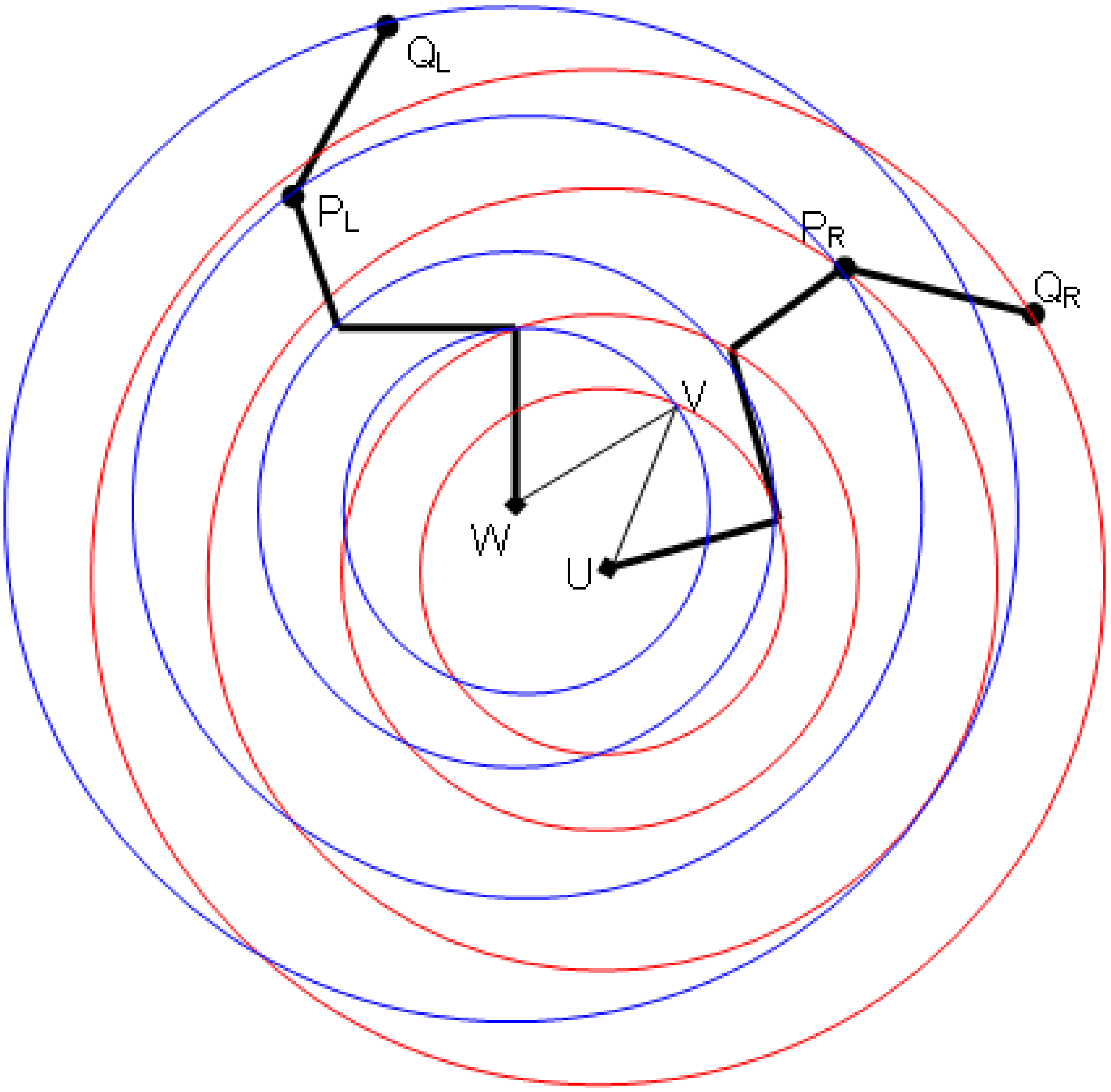


Fig. 18. Multiple leaves break the length constraint

impossible to meet (Figure 18.) Because P$_L$ and P$_R$ are rotating around two separate centers (here labelled W and U), $\left\|\overline{WP_R}\right\|$ cannot equal $\left\|\overline{WP_L}\right\|$ at the same time that $\left\|\overline{WP_R}\right\|$ equals $\left\|\overline{UP_L}\right\|$.

□

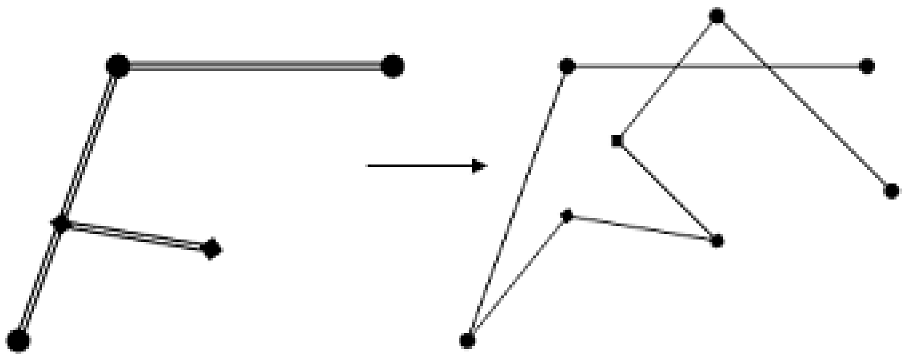


Fig. 19. The problem with multileaf paths

This failure is nicely demonstrated by the counterexample shown in Figure 19. Here the upper edge is completely compliant with the rules governing β for the lowest leaf node but the second leaf introduces a conflict.

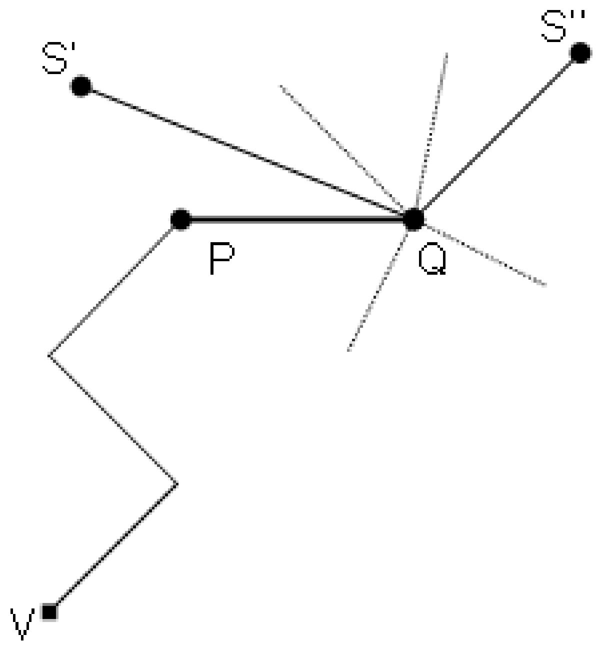


Fig. 20. Extending a valid cut-path without the chance of conflict.

Lemma 3 *On a mesh of convex faces, for any vertex Q added to the root of a cut-branch* $\overline{\Phi_{VP}}$ *by an edge* $\overline{PQ}$*, within the length constraints, there exists a vertex* $S \neq P$ *such that that* $\overline{QS}$ *can be added to* $\overline{\Phi_{VQ}}$ *without conflict.*

Proof: Figure 20 shows a sample labelling. For any vertex S' adjacent to Q, on a simplicial mesh there must exist at least one other vertex S" such that $\angle$(S'QS")>$\pi/2$ and $\angle$(S'QS")< $3\pi/2$. Thus if it were the case that $\angle$(VQS')<$\pi/2$ then $\pi/2 <\angle$(VQS') + $\angle$(S'QS")< $3\pi/2$, giving $\angle$(VQS")>$\pi/2$. □

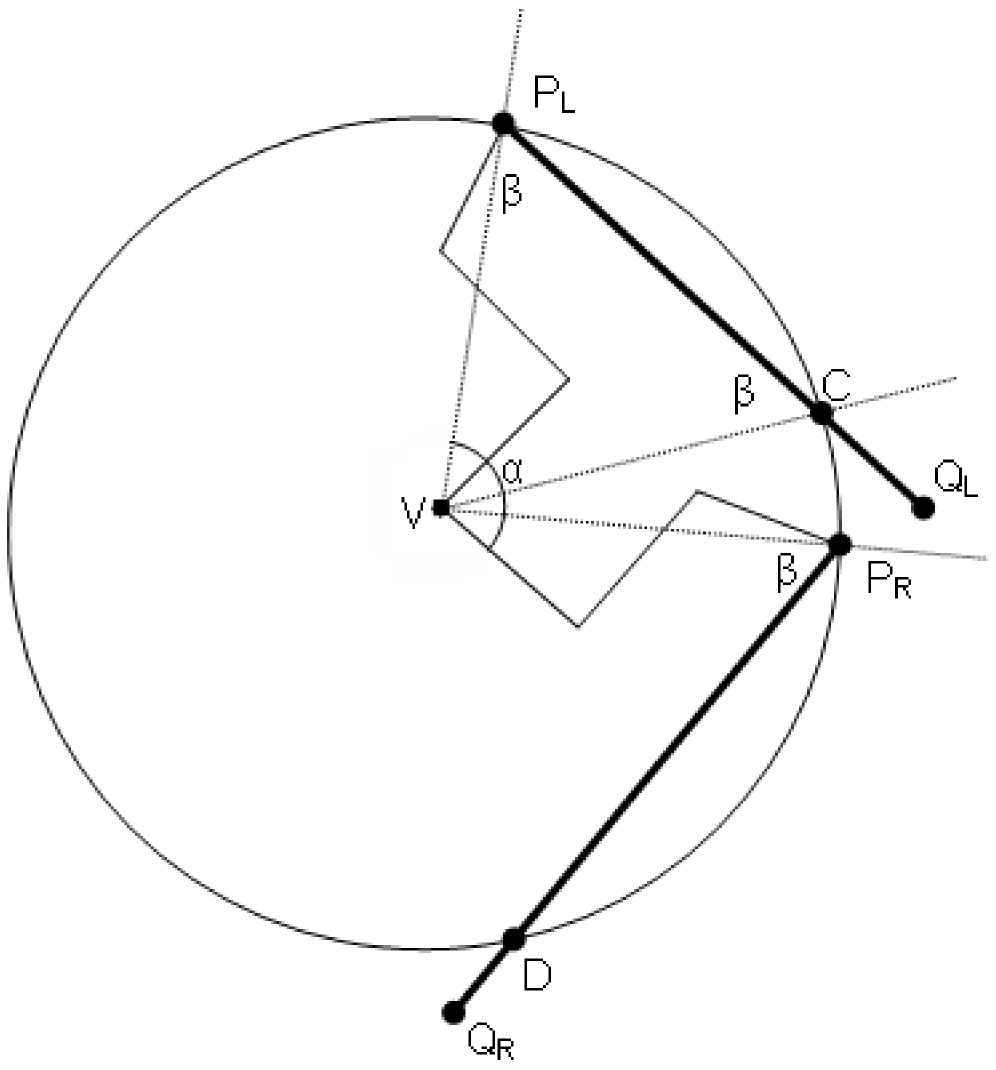


Fig. 21. Sufficiently large angle deficit eliminates the possibility of conflict

Lemma 4 *If $\angle$(VPQ)> (π-AD(V))/2 and $\|VP_L\|$=$\|VP_R\|$ then there can be no conflict between $\overline{P_LQ_L}$ and $\overline{P_RQ_R}$.*

Proof: If $\beta \geq \pi/2$ then there is no intersection (Lemma 1.)

If $\beta < \pi/2$ then the edge segment P$_L$Q$_L$ intersects the circle of radius $\|VP_L\|$ at some point C.

The triangle P$_L$CV is an isoceles triangle with angles β, β, π-2β.

α>π-2β implies that P$_R$ must lie outside the triangle P$_L$CV, indicating that the triangle P$_R$DV does not overlap P$_L$CV at any point other than V.

Therefore if β> (π α) / 2 then P$_R$Q$_R$ and P$_L$Q$_L$ cannot intersect.

□

Lemma 5 *While the summed angle deficit for any vertex on a cut-graph branch on a closed convex polyhedron may exceed 2π, there will always ex-*

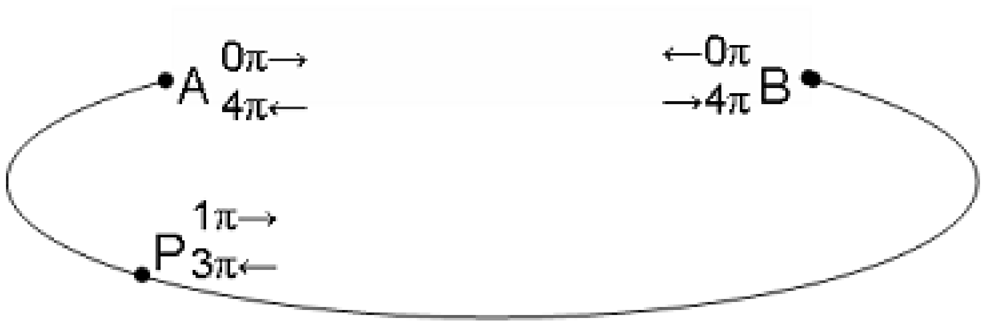


Fig. 22. Alpha need never exceed two pi

ist at least one cut-graph branch from which it does not.

Proof: Referring to Figure 22:

The angle deficit α of a branch $\overline{\Phi_{VP}}$ at a vertex P is the sum of the angle deficits of each vertex $\overline{\Phi_{VP}}$.

On a closed genus-zero polyhedron the total angle deficit is 4π.

The subset of the cut-graph which is the set of all points not in $\overline{\Phi_{VP}}$ is the remainder of the cut-graph. Summing the nodes in this set gives a total angle deficit of 4π-α.

Thus for any vertex P there will always be a value of $\alpha \leq 2\pi$.

□