MOSS:
Multiple Orthogonal Strand System

Bob Haimes
haimes@mit.edu

Aerospace Computational Design Labratory
Department of Aeronautics & Astronautics
Massachusetts Institute of Technology

12^{th} Symposium on
Overset Composite Grids and Solution Technology
Georgia Institute of Technology

Overview

- Introduction
- Design Goals
- Implementation
  - Geometry Import & Tessellation
  - Classification of Vertices
  - Merging of Normals
  - Fanned Fills
  - Radial Fills
  - Sharp Tailing-Edge Treatment
  - Laplacian Smoothing
  - Strand Direction Optimization
  - Setting Cutoff Indices
- Discussion & Status

Introduction

DoD HPCMP CREATE^{TM}-AV Mission

- Analyze aircraft designs before new concepts are purchased
  - Find flaws before manufacturing and expensive work-arounds
  - Use of *high* fidelity simulations – HPC
  - Reynolds-Averaged Navier-Stokes (RANS): HPCMP CREATE^{TM}-AV *Kestral* & *Helios*
  - Requires the use of anisotropic meshes
- Traditional RANS Meshing
  - Use of Structured blocks (abutting or OverSet)
  - Anisotropy naturally handled by adjusting the spacing close to the walls to resolve boundary layers
  - Intensive interaction and a meshing expert is required
- Too much of a burden for acquisition engineers

Need an automatic RANS meshing scheme!

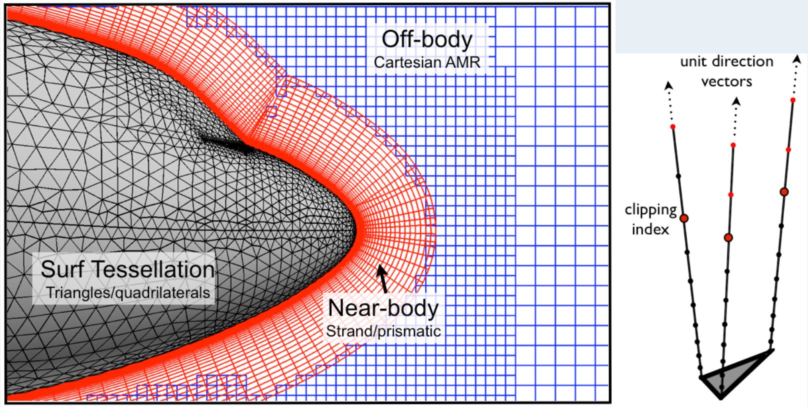
Introduction

Some Automatic RANS Meshing Schemes

- Kallinderis (1993)
  - Inflate the body's surface triangulation a number of times with the desired spacing
  - Builds prismatic layers
  - Fill the voids with an isotropic tetrahedral mesher
- Delanaye *et al* (1999)
  - Off-body mesh is AMR and over-set
- Meakin *et al* (2007)
  - Introduction of *Strands*
  - Prismatic layers are inferred by line segments
  - Off-body mesh is AMR and over-set

Introduction

Strands

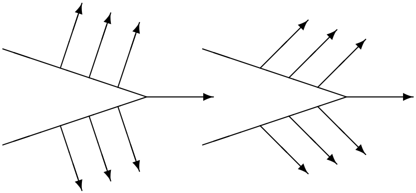


- Minimal memory footprint
  - Vector for *direction*
  - Integer for *clipping index*

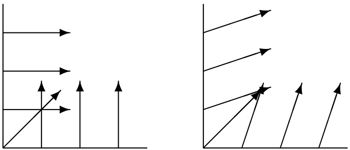
Introduction

Original Single-Strand Implementation (2007)

- Input: a mainfold surface tessellation
- Direction vector computed from triangle normals
- A single *strand* constructed from each surface vertex
- Laplacian smoothing applied to get consistent spatial coverage and reduce collisions in *corner* regions



Trailing Edge

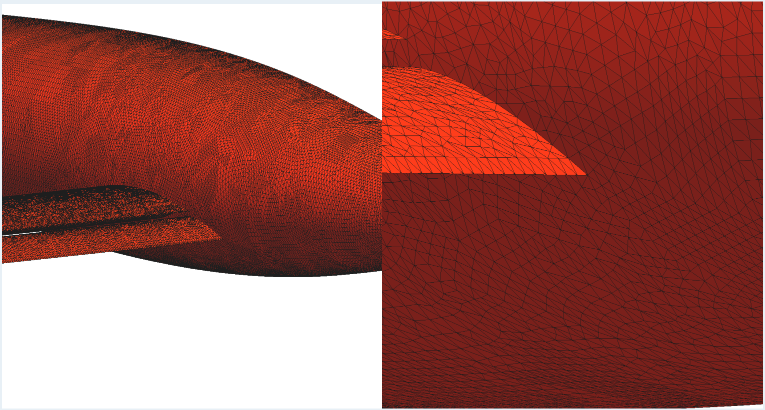


Corner Smoothing

Introduction

Issues with the Current Strand Implementation

- Spatial coverage
- Orthogonality
- Premature cutoff



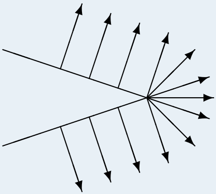
Design Goals

Take the good, fix the bad...

- Automatic
- Small memory footprint
- Improve spatial coverage
- Maintain orthogonality (as much as possible)
- Maximize cutoff index

Simple Idea

- Allow multiple strands to emanate from certain positions
- *Lifted Surface*
  - Surface topology is not the same as above
  - Base of a prismatic stack is degenerate at these positions



Implementation

MOSS Inputs

- A *Solid Model*
- The strand length (in the model's units)
- The number of strand positions and their relative spacings
- The number of smoothing iterations

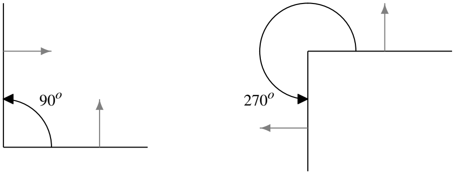
Geometry Import & Tessellation

- Reads and parses the BRep – Faces, Edges, Nodes
  can use various geometry kernels (CAPRI, EGADS, Capstone)
- Geometry kernel provides a *watertight* tessellation
  (can be a mix of quadrilaterals and triangles)
- Provide Face normals for each vertex in the tessellation
  Edges vertices will have 2 Face normals, Nodes have 2 or more

Implementation

Classification of Edges/Nodes Vertices

- Convex Edge Vertex – *Winding Angle* $> 180^o$
- Concave Edge Vertex – *Winding Angle* $< 180^o$
- Convex Node – each Face pair has a *Winding Angle* $> 180^o$
- Concave Node – any Face pair has a *Winding Angle* $< 180^o$
- Opposite-Normals – sharp Trailing-Edge/Fuselage intersections
- Same Normal – difference between normals for a Face pair $< 3^o$

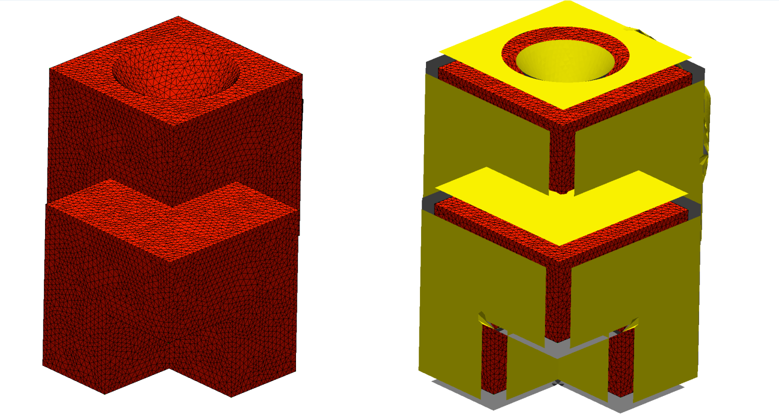


Winding Angle – Plane is generated by $\otimes$ of Face Normals

Implementation

Merging of normals for a *Concave Edge/Node* strand

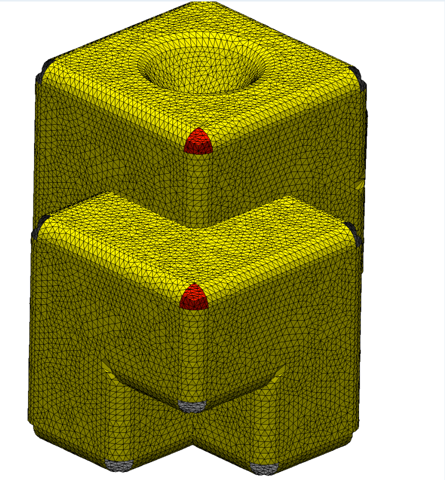
- Sum normals and renormalize
- Any pairs marked as *Same Normal* only summed once



Implementation

Fanned fills for *Convex Edge* strands

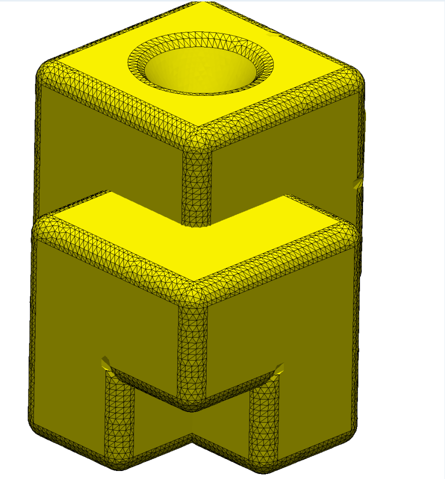
- Compute *Fanning* numbers for each Convex Edge vertex
  - Quad/triangle spacing orthogonal to Edge segment used with *Winding Angle* to figure subdivisons
- Traverse each Edge and remove abrupt changes
- Add triangles for the segment in strips
- Degenerate base element is a *wedge*



Implementation

Radial fills for *Convex Node* strands

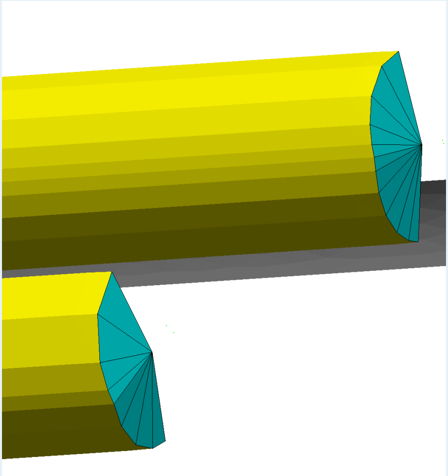
- Creation of center strand
- Close up the exposed Edge segments by triangles that connect to the center
- Insert a strand where the spacing is too large by splitting an interior tri side
- Use a MINMAX angle criteria to drive swapping of interior triangles
- Iterate on the 2 items above
- Degenerate base element is a *tetrahedron*



Implementation

Opposite-Normals Node – sharp Trailing-Edge/Fuselage

- Reopen Edge segments touching ONN
- Create a *frozen* strand aligned with the Trailing-Edge
- Create a *frozen* center strand
- Close up the exposed Edge segments by creating tris that have 2 positions on the exposed Edge openings & connect to the center strand



Implementation

Smoothing (strand pivoting)

Adjust the direction for those strands that locally collide

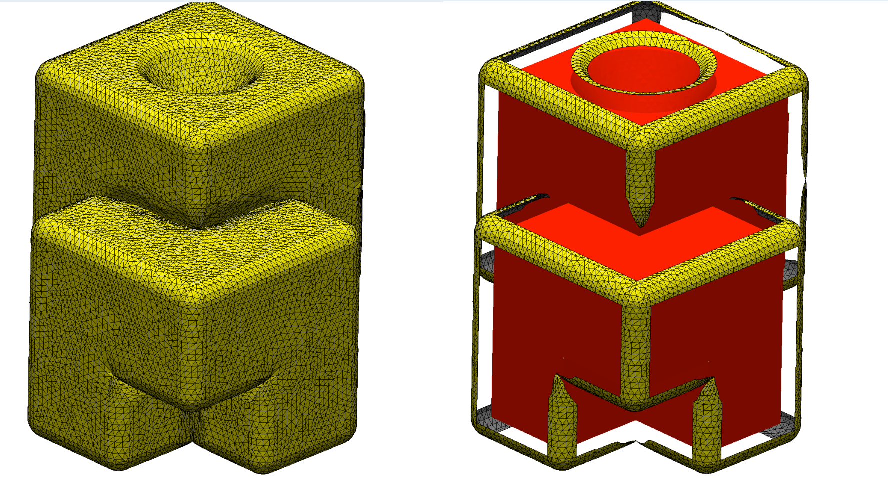
- Mark all strand stacks that are *problematic* at the *lifted surface*
- Flood the *lifted surface* neighbors up to a specified depth
- Update the touched strands by performing the Laplacian smoother (averaging neighboring strand directions and renormalizing) unless the strand is marked as *frozen*.
- Iterate until no movement

This is iteratively done in 2 phases:

- Edge/Node phase. This only adjusts strands emanating from either Edge or Node vertices.
- Interior phase. Only smoothes strands that can be found interior to Faces.

Implementation

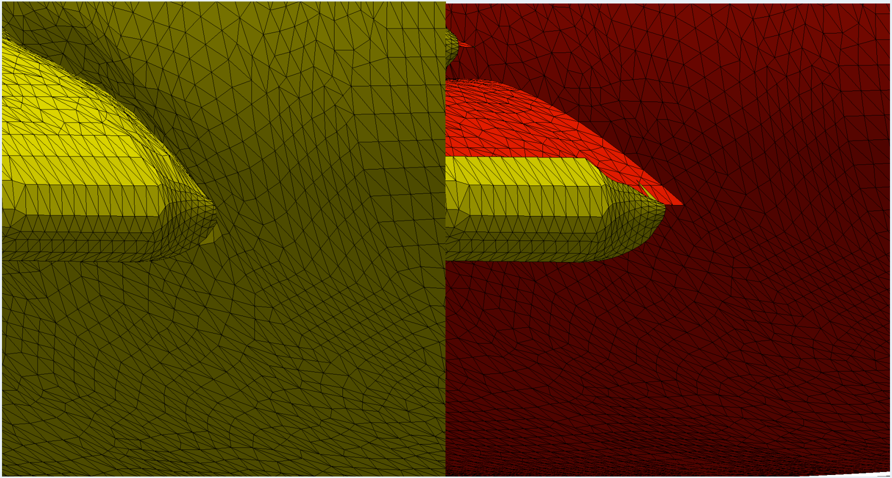
Smoothing (strand pivoting)



Smoothed *lifted surface* and examination of smoothed fans

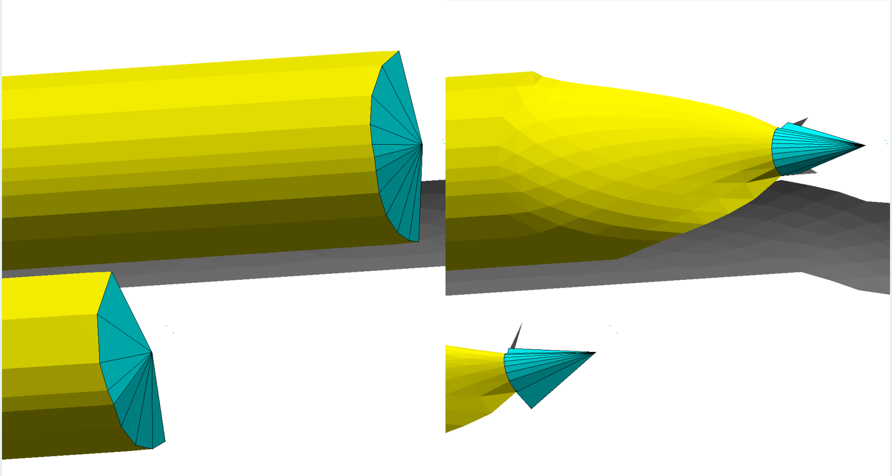
Implementation

Smoothing (strand pivoting)



Smoothed *lifted surface* for Trailing-Edge/Fuselage junction and fans

Smoothing (strand pivoting)



ONN treatment (on wing & flap) before and after smoothing

Implementation

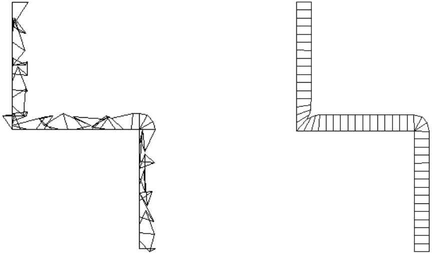
Strand Pivoting (Smoothing)

- Seductively simple
- At times requires user intervention
- Has no constraints or bounds
- Usually cannot straighten out a situation that is tangled at the base
- Can just go wrong!

Implementation

Strand Pivoting as an Optimization Problem

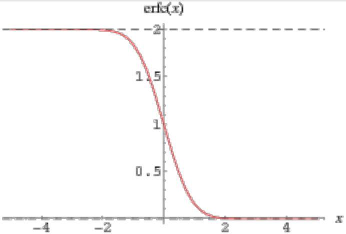
- Minimize the *lifted surface* area in problematic regions
  Will drive individual areas to *zero* – use longer *strand length*
- Solved with conjugate gradients (Polak-Ribiere method)
  Line searches with modified golden section procedure
- Gradients computed via *reverse* differentiation of code assisted by **TAPENADE** (Inria)
- Can straighten out a situation that is tangled at the base



Implementation

Strand Pivoting as an Optimization Problem

- Optimization requires that you get everything correct!
- Gradient-based optimization needs a *smooth* (and hopefully monotonic) objective function:
  $erfc(\pi * MIN(facet_{norm} \bullet strand_{norm})/smooth_{factor}) * facet_{area}$
- Separate regions mapped by *problematic* elements and strand length — can be handled independently
  - Smaller problems (n = 2*$nStrands$ — 2 angle dofs)
  - Threaded/parallel implementation



Implementation

Setting Cutoff Indices

For each Strand the cutoff is set as the smallest index that provides a suite of valid elements (touching the strand)

- Loop from the top (the *lifted surface*) down
- Is the dot product of the normal (at the facet at this level) positive against all strand directions?
- If so, set the index and stop

It should be noted that this only takes care of local collisions:

- Parts of a concave body come close together
- Interference from other bodies are not covered

This is not done as part of MOSS but is performed by PICASSO.

Discussion & Status

Fast & Automatic

- Building the *lifted surface* is straight forward construction
- Most CPU time is consumed in the smoothing & optimization phases

Solver Requirements

- Must be able to deal with the elements types
- Should construct elements *on the fly*
- Needs to be fairly immune to abrupt changes in element sizes
- Could take advantage of the strands as lines for solving
- The use of traditional *finite volume* solvers can be a problem! This is an unusual situation where the meshing scheme is driving choices in solver technologies and development.

Discussion & Status

Adaptation of the *lifted surface*

- Not currently implemented
- Provide element barycentric coordinates for insertion
- Care must be taken if removal of strands effects the surface
- Insertion on Concave Edges requires locally rebuilding *Fans*

Optimization/Smoothing

- Use NLOPT – Low-storage BFGS
- Completely remove Laplacian smoothing (?)

Acknowledgements

Work funded through NASA Grant #NNX13AK16G
Michael J. Aftosmis, *technical monitor*

Bob Meakin (HPCMP CREATE^{TM}-AV) provided the inspiration and support.

William Chan (Nasa Ames) and Andrew Wissink (Army Rotorcraft) provided guidance.

Romain Aubry (HPCMP CREATE^{TM}-MG) assisted in improving the presentation of this work.