COMMUNICATIONS IN NUMERICAL METHODS IN ENGINEERING
Commun. Numer. Meth. Engng 2008; **24**:1641–1652
Published online 3 October 2007 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cnm.1056

On the 'most normal' normal

Romain Aubry*, † and Rainald Löhner

School of Computational Sciences M.S. 4C7, George Mason University, Fairfax, VA 22030-4444, U.S.A.

SUMMARY

Given a set of normals in $\mathscr{R}^3$, two algorithms are presented to compute the 'most normal' normal. The 'most normal' normal is the normal that minimizes the maximal angle with the given set of normals. A direct application is provided supposing a surface triangulation is available. The set of normals may represent either the face normals of the faces surrounding a point or the point normals of the points surrounding a point. The first algorithm is iterative and straightforward, and is inspired by the one proposed by Pirzadeh (*AIAA Paper 94-0417*, 1994). The second gives more insight into the complete problem as it provides the unique solution explicitly. It would correspond to the general extension of the algorithm presented by Kallinderis (*AIAA-92-2721*, 1992). Copyright © 2007 John Wiley & Sons, Ltd.

Received 6 April 2007; Revised 17 August 2007; Accepted 29 August 2007

KEY WORDS: surface triangulation; point normal; normal smoothing; smallest circumscribed circle

1. INTRODUCTION

Surface triangulations are used in many computer applications. Point normals are usually required in visualization, medical applications, boundary conditions in finite volume and elements solvers, free surface problems, tangent plane computations, curvature estimations, etc. Various ways to obtain such point normals are well known. In the context of a RANS mesh generator, where prisms are extruded in one sweep from the surface triangulation along the point normals [1, 2], a fast and accurate computation of the normal is required, particularly along corners and ridges, where the visibility cone could be almost non-existent. In this kind of applications, the 'best' normal is the normal that minimizes the maximal angle with the face normals surrounding a given point. It must be noted, however, that this 'best' normal is the best one if considered independently of all the other normals. A smoothing on the whole surface triangulation may be necessary to obtain the best global normals, as opposed to considering only a subset of normals.

*Correspondence to: Romain Aubry, School of Computational Sciences M.S. 4C7, George Mason University, Fairfax, VA 22030-4444, U.S.A.
†E-mail: raubry@gmu.edu

Although some research on this subject has been done in the past [3, 4], these algorithms do not rely on any theoretical background to prove that the 'most normal' normal is obtained. It is well known that the mesh generation of very anisotropic elements with a ratio of order $O(10^4–10^6)$ gives rise to many problems, as it stresses the difficulty to take numerical decisions with contexts of such different orders of magnitude. Therefore, an accurate algorithm to compute the 'most normal' normal seems to be missing, at least in this field. This is what this paper tries to remedy.

The organization of this paper is as follows. In the first part, a short review of previously proposed normal computations is given. Then, a straightforward iterative algorithm is proposed to give a first idea of the problem at hand. In the third part, a direct algorithm is designed to find the 'most normal' normal. Finally, some examples and timings are given.

2. NORMAL SMOOTHING

The simplest and most commonly used normal smoothing consists in taking the average of all the face normals connected to a given point and normalizing it. This is equivalent to take a least-square approach of the normal computation. Various straightforward variations consist in taking a different weighted average of the face normals according to Frey and George [5]:

- weighted with the surface area of each triangle;
- weighted with the inverse of the surface area;
- weighted with the angle made by the two edges connected in the point under consideration.

However, all these possibilities may fail to produce the 'most normal' normal because they are not associated with any theoretical optimum. A simple example is given by Pirzadeh [3], where the trailing edge of a wing is meshed with four elements on the upper part and three on the lower part of the wing, producing an unbalanced weighting.

A different approach was followed by Engelman *et al.* [6], where the normals were computed to satisfy the discrete incompressibility condition in a finite element context. The normals are then fully imposed on the mass conservation criterion, not considering the geometrical aspect of the boundary.

In [7], Connell and Braaten advocate the use of the normal of the analytic surface instead of relying on the discrete surface. As noted, it is certainly more reliable for coarse meshes. However, the geometrical model could not be available, substituted by sterereolithography data as noted in Ito and Nakahashi [8]. Moreover, as problems usually occur at patch intersections, averages of the geometrical surfaces are necessary, as discussed by Connell *et al*. Therefore, not much has been gained. Finally, due to mappings from 2D to 3D or small angles, it is possible that the geometrical surface itself be distorted. As the prisms will be extruded from the discrete surface exclusively, there are many other advantages to rely on it.

Wang *et al.* solve the Eikonal equation in [9] to offset the surface in the direction of the gradient of the solution. They also propose a classification of offset methods based on a direct or indirect approach if the offset technique relies on the resolution of a partial differential equation (PDE) or not. However, an extra grid is needed to solve this PDE, which may dramatically increase the computer resources.

A first improvement of the pure geometrical approach would consist in creating discrete patches around a point by evaluating the dihedral angle of the faces carried by all the edges connected to this point. Given a user-defined threshold, the discrete ridges of the geometry are then defined

DOI: 10.1002/cnm

(see Löhner [10] and Borouchaki *et al.* [11]). These ridges are the local boundaries of the discrete patches surrounding the point. The averaging process is then performed for each discrete patch, independently of the number of faces included in each patch. However, the averages may not as well be optimal.

A notable improvement was proposed in the context of the RANS gridding by Kallinderis and Ward [4], based on the visibility cone of the each point. The algorithm consists in choosing the pair of normals which produces the largest angle, to take a plane bisector of this faces, to limit the visibility on this plane by projecting the other faces on this plane, and finally to take the bisector of the angle created by the edges which limit at most the visibility in this plane. This normal is always valid if the faces allow it. However, no guarantee is given that the method produces the 'most normal' normal. It is actually not true in general.

Finally, Pirzadeh [3] proposed an iterative method relying on a predictor corrector basis to give the 'most normal' normal. No general convergence criterion is given. It is not obvious that all the computed coefficients will remain positive.

3. A STRAIGHTFORWARD ITERATIVE ALGORITHM

Inspired by the algorithm proposed by Pirzadeh [3], an iterative version was developed to find the 'most normal' normal. Weights are given to each face normal depending on the angle created with

```
!Compute the initial weights:
```

$$w_i = \frac{1}{n} \tag{1}$$

```
!Compute the initial guess normal:
```

$$\mathbf{N}_p = \frac{\sum w_i \mathbf{N}_i}{\| \sum w_i \mathbf{N}_i \|} \tag{2}$$

```
while(not converged):
    !Compute the angle α_i :
```

$$\alpha_i = arccos(\mathbf{N}_p \cdot \mathbf{N}_i) \tag{3}$$

```
    !Compute the new weights:
```

$$w_i = w_i \times \frac{\alpha_i}{\sum \alpha_i} \tag{4}$$

```
    !Normalize the weights:
```

$$w_i = \frac{w_i}{\sum w_i} \tag{5}$$

```
    !Compute a new normal approximation:
```

$$\mathbf{N}_{pnew} = \frac{\sum w_i \mathbf{N}_i}{\| \sum w_i \mathbf{N}_i \|} \tag{6}$$

```
    !Relax the new vector:
```

$$\mathbf{N}_p = \beta \mathbf{N}_{pnew} + (1 - \beta) \mathbf{N}_p \tag{7}$$

```
    !Check convergence
end while
```

Figure 1. An iterative algorithm for computing the 'most normal' normal.

DOI: 10.1002/cnm

the current normal. If the angle is high, more weight is given to the normal. Given the normals $\mathbf{N}_i$, $i=1,n$, the whole procedure consists in finding $\mathbf{N}_p$ and is described in Figure 1.

The weights are always positive as they are fractions of angles, and lesser or equal to one due to the normalization step. Apart from the convergence speed, which will be considered in Section 5, a drawback of the algorithm is the repeated necessary use of the arccos function, which is expensive. A shift of the scalar product values from $[-1:1]$ to $[0:1]$ could be a remedy but has not been followed here.

4. THE DIRECT APPROACH

In this section, the approach followed is to find explicitly the normal that minimizes the maximal angle given a set of normals. Some questions arise from the previous algorithm. Is it necessary that all the weights be positive? And does the 'most normal' normal belong to the visibility cone? Some characteristic sets of the problem must be firstly introduced.

Given $\mathscr{F}_p$ the set of faces around point P with their respective normal $\mathbf{N}_j$, $j \in \mathscr{F}_p$, the visibility cone at point P is defined by Kim *et al.* [12] as

$$\{M \in \mathscr{R}^3 \,|\, \overrightarrow{\mathbf{PM}} \cdot \mathbf{N}_j > 0,\ \forall j \in \mathscr{F}_p\} \tag{8}$$

Another cone playing an important part is the normal cone defined as

$$\{M \in \mathscr{R}^3 \,|\, \overrightarrow{\mathbf{PM}} = \alpha_j \mathbf{N}_j,\ \alpha_j \geqslant 0\ \forall j \in \mathscr{F}_p\} \tag{9}$$

Both cones are duals in the sense that, according to Yamaguchi [13], the faces of the visibility cone are normals to the edges of the normal cone and the faces of the normal cone are normals to the edges of the visibility cones. Denoting by $\mathscr{E}_p$ the set of edges $\mathbf{U}_i$ of the visibility cone, the dual definition of the visibility cone becomes:

$$\{M \in \mathscr{R}^3 \,|\, \overrightarrow{\mathbf{PM}} = \alpha_i \mathbf{U}_i,\ \alpha_i \geqslant 0\ \forall i \in \mathscr{E}_p\} \tag{10}$$

and the one of the normal cone becomes:

$$\{M \in \mathscr{R}^3 \,|\, \overrightarrow{\mathbf{PM}} \cdot \mathbf{U}_i > 0,\ \forall i \in \mathscr{E}_p\} \tag{11}$$

so that the duality is highlighted. Figure 2 illustrates in two dimensions the visibility and normal cone around point P in a convex situation. In this particular case, the visibility cone is contained in the normal cone. The normal cone is convex and infinite, but as only the directions matter, the important features are located at the intersection between the normal cone and the sphere of radius one due to the normalization of the vectors. This intersection represents a curved convex polygon. As the curvature is constant on the sphere, the length of a curved side between two normals is equal to the angle created by these normals, the radius being equal to one. Hence, our initial problem of finding the normal that minimizes the maximal angle given a set of normals is shifted to the problem of finding a point inside (not necessarily strictly) the convex-curved polygon on the surface of the sphere that minimizes the maximal distance from this point to the end point of the normal vectors different from P.

In the plane, the problem of finding the point that minimizes the maximal distance from a set of points is classical and has a long history (see [14, p. 248]). It is sometimes called the

DOI: 10.1002/cnm

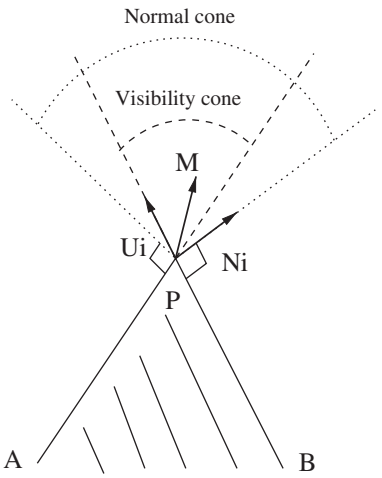

Figure 2. The visibility cone carried by vector $\mathbf{U}_i$ in particular and the normal cone carried by vector $\mathbf{N}_i$ in particular around point P, surrounded by faces BP and PA in two dimensions with point M inside the visibility cone.

facility location problem or the minimum spanning circle problem. Its solution consists in finding the smallest circumscribed circle to the plane convex polygon created by the boundary of the convex hull of the set of points. A naive but straightforward version of the algorithm is given by Papadopoulos [15] in the linear fatigue context running in $O(n^4)$, where n is the number of given points. An algorithm is given by Megiddo [16] that computes in linear time the smallest circle. However, for the application at hand, the faces surrounding a point or the points surrounding a point are expected to be between 4 and 8 so that it is difficult to beat the naive algorithm for such small inputs. The main result relies on the fact that the smallest circumscribed circle to a plane polygon must touch either two or three points [17]. It is then straightforward to build all the circles created by all the combinations of two and three points and pick the one which contains all the other points with the smallest radius. An important property of the smallest circle is recalled:

Lemma 4.1
The smallest enclosing ball of a set of points S has its center lying in the convex hull of at most three points of S and is unique.

The proof is given in Elzinga and Hearn [18], and relies on the Kuhn–Tucker conditions [19] of the associated convex programming problem.

Coming back to the 3D problem on the surface of the sphere, it was seen before that the distances are angles due to the constant curvature on the sphere. Therefore, the point that minimizes the maximal distance on the sphere will trivially minimize the maximal angle, which is exactly what is sought. The algorithm in the plane is straightforward to extend on the sphere surface. In particular, Lemma 4.1 is directly inherited. Figure 3 displays the minimum circumcircle for point P with its center C, **PC** being the 'most normal' normal, and its radius R with some normal $\mathbf{N}_j$. Only two extreme normals **N**1 and **N**2 have been depicted for the sake of clarity. The center of an edge is the extremity of the normed bisector of the angle made by an edge, and the center of the circle

DOI: 10.1002/cnm

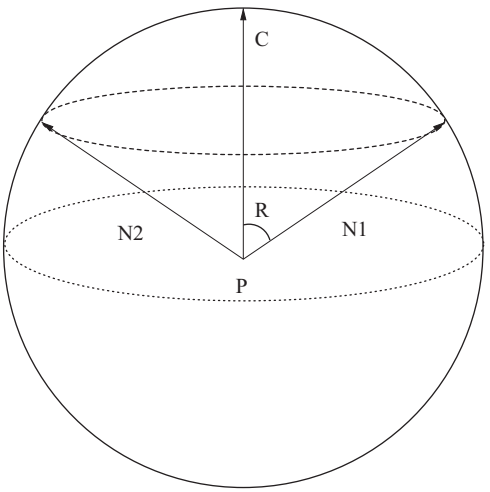

Figure 3. The smallest circumscribed circle around point C for point P.

```
!Initialize scalmin
scalmin=-1
do i=1,n
    do j=i+1,n
        !Compute the bisector vector (center):
        N_b = N_i + N_j
        N_b = N_b / ||N_b||
        !Compute the radius (angle)
        scal=N_b · N_i = N_b · N_j
        !Compare the current length against current
        smallest length
        if(scal.lt.scalmin) goto 100
        !Are all the points inside this circle?
        do l=1,n
           scalt=N_l · N_b
           if(scalt.lt.scal) goto 100
        enddo
        !Store the minimal radius and the center
        c = N_b
        scalmin=scal
        100:continue
        enddo
    enddo
enddo
```

Figure 4. Computing the circles created by the diameter of all pairs of points.

made by three vectors is the extremity of the vector which gives the same scalar product with the three vectors forming the three points. For the sake of completeness, full details of the naive algorithm are given, reordering efficiently the sequence of computation. As seen before, it mainly

DOI: 10.1002/cnm

```
do i=1,n
    do j=i+1,n
        do k=j+1,n
        !Compute the center of the circle
```

$$denom = rx_{ij} \times ry_{ik} - rx_{ik} \times ry_{ij} \tag{12}$$

$$Ncx = (rz_{ik} \times ry_{ij} - rz_{ij} \times ry_{ik})/denom \tag{13}$$

$$Ncy = (rx_{ik} \times rz_{ij} - rx_{ij} \times rz_{ik})/denom \tag{14}$$

$$Ncz = 1/\sqrt{1 + Ncx \times Ncx + Ncy \times Ncy} \tag{15}$$

$$Ncx = Ncx \times Ncz \tag{16}$$

$$Ncy = Ncy \times Ncz \tag{17}$$

```
        !Check the orientation:
```

$$\texttt{scal} = \mathbf{N}_c \cdot \mathbf{N}_i \tag{18}$$

```
        if(scal.lt.0) then
```

$$\mathbf{N}_c = -\mathbf{N}_c \tag{19}$$

```
        endif
        !Compute the radius (angle)
```

$$\texttt{scal} = \mathbf{N}_b \cdot \mathbf{N}_i = \mathbf{N}_b \cdot \mathbf{N}_j = \mathbf{N}_b \cdot \mathbf{N}_k \tag{20}$$

```
        !Compare the current against smallest length
        if(scal.lt.scalmin) goto 200
        !Are all the points inside this circle?
        do l=1,n
           scalt=N_l · N_c
           if(scalt.lt.scal) goto 200
        enddo
        !Store the minimal radius and the center
        c = N_c
        scalmin=scal
        200:continue
        enddo
    enddo
enddo
```

Figure 5. Computing the circles created by all triples of points.

consists in two parts, the first part evaluating the circles created with two points and the second part computing the circles created with three points. Given the normals $\mathbf{N}_i$, $i = 1, n$, the first part consisting in finding all the circles with two points is displayed in Figure 4. The second part consisting in finding all the circles with three points is described in Figure 5. It may be noticed that if a circle with two points is found to enclose all the points after the first part, those two points forming the diameter of the cloud of points, then this is the smallest circumscribed circle as, in the plane, all the triangles formed by a third point inside this circle and this diameter will then be obtuse with their circumscribed circle having a larger radius, supposing that it also encloses all the points.

When computing the center of the circle, which is the vector with the same scalar product with vectors $\mathbf{N}_i$, $\mathbf{N}_j$, and $\mathbf{N}_k$, an expression like rx_{ij} means

$$rx_{ij} = (\mathbf{N}_i)x - (\mathbf{N}_j)x \tag{21}$$

DOI: 10.1002/cnm

Equations (12)–(17) give the vector producing the same scalar product for three given vectors. As two vectors only differing by the orientation are solutions of this problem, orientation is checked to consider only the one which has a positive scalar product with the three other vectors. The monotony of the arccos function has been used to compare scalar products instead of angles. As all the algorithms dealing with circle computations, some tolerances must be introduced to handle multiple cospherical point configurations. The input data can be checked during the first part by verifying that no two normals are opposed. It should also be noticed that the *denom* term chosen is just one of nine possible permutations, three on the coordinates and three on the triplet of points. A robust implementation should take these permutations into account.

Finally, the relevant lemma for the application at hand is given:

Lemma 4.2
The 'most normal' normal belongs to the visibility cone.

Supposing that the input is valid, all the pairs of faces surrounding the point make an angle strictly lesser than π. In case of two extreme faces, the 'most normal' normal procedure creates a vector that divides this angle by two. As the largest angle between these extreme faces should be less than π, the largest angle between the 'most normal' normal and each of the extreme faces is less than $\pi/2$ so that it belongs to the visibility cone. The same idea applies for the case of three extreme faces. It was then valid to find a representation of the 'most normal' normal as a combination of the set of normal with positive coefficients. It is now easy to see that the algorithm proposed by Kallinderis and Ward [4] does not in general produce the 'most normal' normal.

5. NUMERICAL EXAMPLES

Two examples have been chosen to compare the timing and robustness of the proposed algorithms. Figure 6 illustrates a detail of an hypersonic flyer where two very convex parts converge with two very concave parts on the bottom of the flyer. Some faces are almost coplanar. The face normals are displayed with the 'most normal' normal. Figure 7 shows a detail of the normals. The most vertical normals are the one of the three upper faces. Then the normals of the three lower faces appear, almost superimposed. Then the 'most normal' normal is displayed. Finally, the most horizontal normal corresponds to the face which presents a cusp. Both algorithms were tested on the supersonic flyer presenting many convex and concave ridges and corners. The surface mesh contains 121×10^3 points and five iterations were performed. Computations were made on an Intel Pentium 4 HT with Suse 9.0. For the standard smoothing, the whole process requires 1 s. The algorithm of Section 3 is rather slow. Almost 300 iterations are necessary most of the time to converge to a precision of 10^{-4} on the angle difference. The value of the relaxation seems to play no influence on the convergence. Relying on the above discussion, it is to be expected that points on the same circle will slow down convergence. Furthermore, beyond a convergence criterion of 10^{-4}, convergence is not reached. The whole process requires 14 s for this tolerance. The algorithm of Section 4 gives the 'most normal' normal explicitly. Even though the naive algorithm is in $O(n^4)$ for each point, so $O(n^5)$ globally, the number of points surrounding points or faces surrounding points is bounded, giving a global linear complexity in the application considered. As the coding is straightforward, it is the algorithm of choice, giving always as good or better results that the iterative version. The whole process requires 8 s.

DOI: 10.1002/cnm

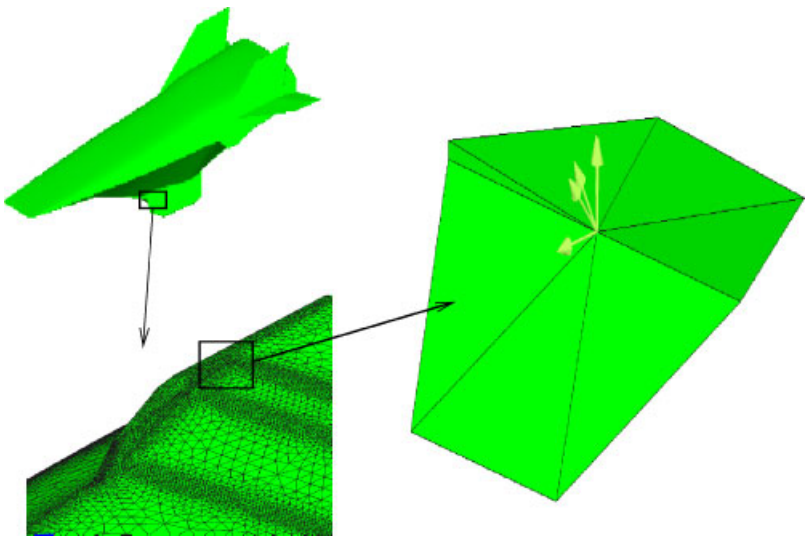

Figure 6. A 'most normal' normal at a cusp point on an hypersonic flyer.

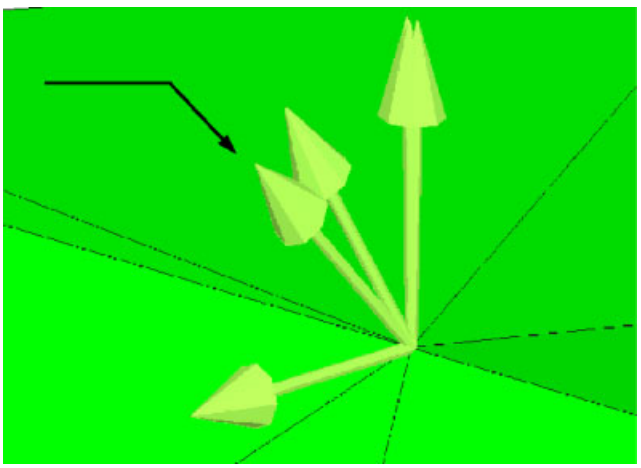

Figure 7. Detail of the normals with the 'most normal' shown by the arrow.

The second example is a Micro Tactical Expandable Flyer (MITE) [20] displayed in Figure 8. The surface mesh contains 109×10^3 points. Various intersections of concave and convex ridges make the 'most normal' normal computation difficult. The example displayed in Figure 8 shows the intersection of a concave ridge with a convex ridge. On the convex ridge, multiple normal faces are used, as explained in [2]. However, as the ridge ends, a cusp point is created, which is also connected to a concave ridge. The cusp point is pointed by an arrow in Figure 8. The multiple normals are displayed at the neighbor of the cusp point along the convex ridge and the curved concave ridge is also depicted. The 'most normal' normal clearly minimizes the maximal

DOI: 10.1002/cnm

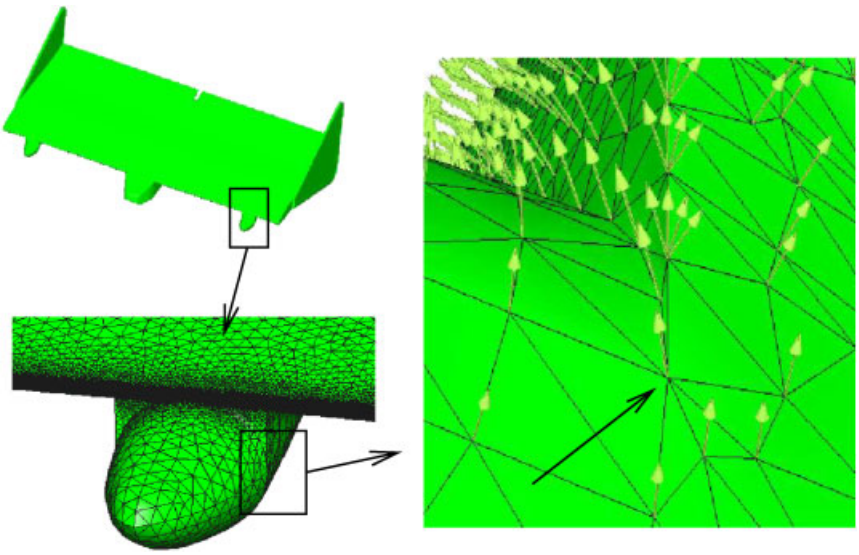

Figure 8. A 'most normal' normal at a concave/convex point on a MITE flyer.

angle with all the first-order neighboring face normals. The timings are similar to the supersonic flyer. For the whole process, 7 s are required for the 'most normal' normal computation for the same iteration number as before, while 13 s are needed for the iterative version. As before, no convergence is observed for some points beyond a convergence criterion of 10^{-4}.

Even if the previous algorithm is expected to behave linearly for the application in mind, much more operations are involved than in a standard normal smoothing as shown by the difference in CPU time. It must be noted, however, that all the operations are performed on the neighboring normals plus some temporary variables. It is advisable to first copy all the neighboring normals in a small array, and then to work on this array, minimizing the cache misses. An immediate improvement of the previous algorithm consists in first computing the standard smoothed normal, evaluate the deviation with the other normals, and only if the maximal deviation is higher than a given threshold, activate the 'most normal' normal procedure. This idea is motivated by the fact that it is expected that cusp, ridge, and corner points represent a small fraction of the total point surface, the rest of the surface being smooth. With this improvement, only 3 s are needed for the hypersonic flyer for a conservative threshold of 0.95 on the scalar product for the whole procedure, which confirms that the 'most normal' normal procedure is triggered only for few configurations. Similar results are obtained with the MITE.

6. CONCLUSION

Two algorithms have been given to compute the normal that minimizes the maximal angle given a set of normals. The algorithm of Section 3 is original, iterative and was found slow in practice. The algorithm of Section 4 is due to the shifting of the problem to the smallest circumscribed circle of the concave polygone formed by the extreme points of the normals in the plane, made

DOI: 10.1002/cnm

possible by the constant curvature of the sphere, and for which optimal algorithms are available. This is the algorithm of choice when applied to a surface triangulation.

Both algorithms are given in full details, particularly the second one, and some direct improvements have been commented. It could be interesting to compare timings with the implementation proposed by Megiddo [16], although it is expected that the improvement could be limited due to the few numbers of surfaces surrounding points or points surrounding points, in practice less than 10 in average. The algorithm proposed by Elzinga and Hearn [21] may be a better option due to its greatest simplicity.

It is straightforward now to generalize the classical normal smoothing on the whole surface by the most sophisticated smoothing by the 'most normal' normal. It is sufficient to replace the face normals surrounding a given point by the point normals or the first-order neighborhood of the point.

Finally, all the smoothing procedures are very sensitive to the boundary conditions created by the imposed normals. There are cases where it is difficult to determine satisfactorily appropriate boundary conditions, and a first smoothing on these imposed boundary conditions may cause unacceptable shrinkage. The fair surface design of Taubin [22] eliminates shrinkage by producing a low-pass filter. Therefore, a coupling of the 'most normal' normal procedure with the one of Taubin for difficult imposed boundary conditions appears to be a good solution. This algorithm is currently tested with success and will be reported in further work.

REFERENCES

1. Löhner R. Matching semi-structured and unstructured grids for Navier–Stokes calculations. *AIAA-93-3348-CP*, 1993.
2. Aubry R, Löhner R. Generation of viscous grid with ridges and corners. *AIAA Paper 07-3832*, 2007.
3. Pirzadeh S. Viscous unstructured three dimensional grids by the advancing-layers method. *AIAA Paper 94-0417*, 1994.
4. Kallinderis Y, Ward Y. Prismatic grid generation with an efficient algebraic method for aircraft configurations. *AIAA-92-2721*, 1992.
5. Frey PJ, George PL. *Maillages*, *Applications aux Elements Finis*. Hermes: Paris, 1999.
6. Engelman M, Sani S, Gresho PM. The implementation of normals and/or tangential boundary conditions in finite elements codes for incompressible fluid flows. *International Journal for Numerical Methods in Fluids* 1982; **2**:238–255.
7. Connell SD, Braaten ME. Semistructured mesh generation for three dimensional Navier–Stokes calculations. *AIAA Journal* 1995; **33**(6):1017–1024.
8. Ito Y, Nakahashi K. Direct surface triangulation using stereolithography data. *AIAA Journal* 2002; **40**(3):490–496.
9. Wang Y, Guibault F, Camarero R, Tchon K. A parameterization transporting surface offset construction method based on the Eikonal equation. *AIAA Paper 05-5241*, 2005.
10. Löhner R. Regridding surface triangulations. *Journal of Computational Physics* 1996; **126**:1–10.
11. Borouchaki H, Villard J, Laug P, George PL. Surface mesh enhancement with geometric singularities identification. *Computer Methods in Applied Mechanics and Engineering* 2005; **194**:4884–4894.
12. Kim DS, Papalambros PY, Woo TC. Tangent, normal and visibility cones on Bézier surfaces. *Computer Aided Geometric Design* 1995; **12**:305–320.
13. Yamaguchi Y. Bézier normal vector surface and its application. *Proceedings of the Shape Modeling International*, Aizu-Wakamatsu, Japan, 1997.
14. Preparata FP, Shamos MI. *Computational Geometry*: *An Introduction*. Springer: New York, 1985.
15. Papadopoulos IV. Critical plane approaches in high-cycle fatigue: on the definition of the amplitude and mean value of the shear stress acting on the critical plane and mean value of the shear stress acting on the critical plane. *Fatigue and Fracture Engineering Materials and Structures* 1998; **21**(3):269–285.
16. Megiddo N. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM Journal on Computing* 1983; **12**(4):759–776.

DOI: 10.1002/cnm

17. O'Rourke J. *Private Communication* 2007.
18. Elzinga J, Hearn D. The minimum covering sphere problem. *Management Science* 1972; **19**(1):96–104.
19. Peressini FE, Sullivan AL, Uhl JJ. *The Mathematics of Nonlinear Programming*. Undergraduate Texts in Mathematics. Springer: Berlin, 1988.
20. Kellogg J. *The NRL MITE Air Vehicle* 2001.
21. Elzinga J, Hearn D. Geometrical solutions for some minimax location problems. *Transportation Science* 1972; **6**:379–394.
22. Taubin G. A signal processing approach to fair surface design. *SIGGRAPH '95*: *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, Los Angeles, CA, U.S.A., 1995; 351–358.

DOI: 10.1002/cnm