1 Introduction

Computational design and analysis has become a fundamental part of industry and academia for use in research, development, and manufacturing. In general, the accuracy of computational analysis depends heavily on the fidelity of the computational representation of a real-world object or phenomenon. However, the task of creating high fidelity models of an actual geometry can be time-consuming—sometimes consuming up to seventy-five percent of the time required to produce a solution [1]. Most mesh generation strategies focus on element quality—with the justification being that downstream applications require high quality geometries in order to achieve a desired level of accuracy. However, element quality should be secondary to accurately representing the underlying physical object or phenomenon.

The justification for the development of these methods was to explore the concept of optimal mesh refinement, and subsequently optimal representation, through the development of a formal description of fundamental mesh operations. These mesh operations (triangle split, edge split, edge collapse, edge swap, and vertex smoothing) were developed with respect to optimal representation of a discrete element for a continuous, parameterized surface. The proposed algorithm is a general-use method that can be applied to any "digital surface" regardless of its representation. This is due to every step being developed without the use of derivatives. Most other methods operate on a specific type of surface geometry, e.g. NURBS surfaces, and use the specific information available for the type of surface in use. [TRANSITION SENTENCE] A result of not using derivatives is that each step in the algorithm is robust to large changes in derivatives or curves that are not "well behaved", e.g. they were highly oscillatory.

This process can only be automated if some way of judging "how well" a surface grid represents a surface is present. To this end, a method of generating surface grids through constrained, local optimization is detailed below. The concept of discrete element quality is also defined with respect to optimal representation. The above concepts are contrasted with traditional methods for grid generation through direct comparison of representation optimality and efficiency.

2 Related Work

In general, grid generation is a name for any process that creates a grid. For example, the advancing-front algorithm advances boundaries into space to generate a grid [CITE]. Other methods generate grids from iterative refinement or enrichment from initial, coarse configurations [CITE]. Usually the benchmark for separating the two methods, generation and refinement, are the prioritization of grid quality and grid accuracy (both of these issues will be addressed later). From a standard text [CITE]:

[PARAGRAPH FROM: HANDBOOK OF NUMERICAL GRID GENERATION]

Traditionally, surface grid generation processes produce good quality grids from the combination of geometric growth rates and smoothing. However, the process requires input and if the input, or starting place, is not appropriate, then the geometry can be under and/or over sampled for the intended use. That fact is not an indictment of the grid generation process, but instead implies that the final grid is heavily dependent on the inputs. In addition, if some way of controlling the point spacing in the middle of the surface (away from the boundaries) is not present, then more points could be wasted/omitted in an attempt to accurately represent the geometry.

Other efforts have gone into creating a locally or globally "optimal" surface grid. Many names have been assigned to the particular task, but the underlying goal is very similar — represent a surface as accurately as possible—whatever that means for each application. For example, [CITE, EXPLAIN, CITE, EXPLAIN, CITE, EXPLAIN]

3 Discretization Error, Discrete Curvature Approximation

3.1 Discretization Error

The accuracy, or discretization error, of a piecewise, linear representation of an analytical surface in $\mathbb{R}^3$ can be defined in many ways depending on the intended application. The error associated with with the discretization is discussed in terms of the "deviation" from the surface—most often quantified by calculating or approximating the distance from the surface for each linear entity (triangles and edges) in the discretization, or some [AREA RELATED ERROR]. Another way of quantifying the error associated with a discretization would be to consider how well it approximates the surface area of the surface it represents. In general the surface area is not known *a priori*, but depending on the underlying representation it can be calculated exactly ([EXAMPLE]) or can be estimated ([EXAMPLE]). One of the goals of the developed method was to be "general" in that it should be independent of the underlying geometric representation. Therefore, a method that requires the surface area of the underlying geometry violates the aforementioned concept of "generality" and restricts the applications for which the proposed method could be applied. Some other way of determining/generating a surface grid based on surface area is needed. This process will be detailed later.

The concept of "deviation" as defined above is relatively straightforward and intuitive. However, another related way of describing "how well" a *discretization* represents a *surface* is the degree to which the discrete representation approximates curvature – where curvature is defined as the amount of "bend" in a *curve* or surface, or "how much" a *curve* or surface "differs" from a straight line or plane (words in quotes are subject to gradation). First, however, curvature must be defined in such a way that a discrete approximation is meaningful and appropriate. In relevant literature, there are many ways to estimate surface curvature [3]. Some of it bears repeating, because it is germane to what is being discussed here: Consider the following surface, S, embedded in $\mathbb{R}^3$, at point P. The concept of an osculating circle [REFERENCE] does not generalize to higher dimensions and therefore cannot be used here as an approximation of curvature. An "osculating ellipsoid" could be constructed by considering the principal curvature directions and surface normal at a given point on a surface. However, this requires the computation or approximation of derivatives—which we have ruled out here.

Instead of determining a three-dimensional analog to an osculating circle for S, an "osculating sphere" could be defined for a triangle, T, contained in a discretization, D, which is approximating S. A value of curvature can be calculated for each triangle in the D by considering the corresponding osculating sphere for T. The osculating sphere here (sphere,[FIGURE]) can be approximated by considering the circumscribed sphere (circumsphere) [2] defined by the three points of the triangle, P_0, P_1, and P_2, and a point, P on the surface located within the triangle in (u, v) space. — the radius of the circumsphere will be referred to as the discrete radius of curvature, R_D. This argument for curvature approximation is a two-dimensional analog to the one in [CITE SELF] for approximating curvature along a curve. In that work, the authors showed that for an edge on a curve, as the radius of curvature of the osculating circle approaches infinity the arc length of the circular segment approaches the arc length of the edge. However, here as R_D approaches infinity, the surface area of the spherical cap does not approach the area of T—it approaches that of the circumscribed circle defined by T.

Finally, what is most appropriate is to consider the surface area of the tetrahedron, $\mathbb{T}$ formed by P_0, P_1,P_2, and P. This is a meaningful comparison: as R_D approaches infinity (as the distance between P and the plane defined by T approaches 0) the surface area of $\mathbb{T}$ approaches two times the area of T. Stated another way, as D is refined the surface area of D approaches that of S.

Surface area convergence of a discretization is a sufficient condition for other schemes of surface grid generation/refinement. That is: if the difference between the surface area of the surface and the sum of the surface area of the discrete entities in the discretization approaches zero then that is sufficient to conclude that the distance between the discretization and surface is also approaching zero, also the dihedral angles between segments approaches 0 degrees [see PROOF once definition of REPRESENTATION DEFICIT is detailed in APPENDIX]. However, the converse of that statement is not true. The pathological case of a highly oscillatory, low amplitude surface approximated by two triangles [MORE DETAIL] can have a small "deviation" or angles between elements but be a poor estimate for surface area.

4 Nomenclature and Definitions

4.1 Parametric Surface

I need a parametric surface so that I can maintain a valid topology. Projection is not required since the topology gives me uv bounds. The mesh generation is going on in parametric space. There is an optimization function for creating the mesh in parametric space. Parametric space is also great for optimization since you can write the topology and optimization function constraints in uv space. Mapping a discretization to a surface in general involved the determination of "what" an edge means on the surface—whether it's a geodesic line, or what... If you can do this that's great but the development of a map is not the purpose of this work so, for ease of presentation, a parametric surface is used.

There's plenty of precedent for surface mesh generation in parameterized space.

The real justification for using a parameterization is that the formulation of constraints for representation deficit is straightforward. The search space for the local optimization problems is straightforward and intuitive to define–as well as efficient to search. However, if the surface mesh (in $\mathbb{R}^3$) were mapped to the surface...[SOMEHOW REMEMBER WHERE YOU WERE GOING AND FINISH THIS THOUGHT]

Consider an orientable, parameterized surface, $\vec{S}\,(u,v) : \mathbb{R}^2 \rightarrow \mathbb{R}^3$.

4.2 Discretization

Consider a discretization, D, defined on S, comprised of n_p points, $P \in \{p_1, ..., p_{n_p}\}$, and a non-overlapping, non-degenerate, consistently oriented, triangular connectivity $T \in \{t_1, ..., t_{n_t}\}$. Each triangle in T, t_i, is defined by points (p_j, p_k, p_m), and edges (e_n, e_o, e_p)—where the edges are defined (for t_i) as $e_n\,(p_j, p_k)$, $e_o\,(p_k, p_m)$, $e_p\,(p_m, p_j)$. For the purposes of defining an edge in this work, the ordering of the nodes does not matter. However, the ordering of the triangle connectivity (sometimes referred to as winding) is used to maintain consistent orientation during topological operations and to define constraints in the developed optimization strategy.

Each point, p_i, in D is defined at a $(u,v)_i$ coordinate, and an edge is a straight line in both parametric space and in $\mathbb{R}^3$. This is not the case for a discretization, $D : \mathbb{R}^3$, which is mapped onto a surface, $S : \mathbb{R}^3$. In the absence of a parameterization the mapping of an edge in $\mathbb{R}^3$ to a curve (possibly a straight line, but not guaranteed) on a surface is an ambiguous task which is outside the scope of this work. Additionally, the development of topological constraints for optimization, which will be detailed later, would not be as straightforward as is possible when using a parameterized, planar space.

4.3 Representation Deficit

The concept of representation deficit was discussed in [5] in the context of scale-independent measure of curve discretizations. Here, the concept will be extended to two dimensions. It is important that any criteria for refinement maintain scale indepdence. This is so that the values for representation deficit can be compared between different candidate operations. Below, representation deficit is defined and discussed for four fundamental mesh operations: triangle split, edge split, edge swap, and node smoothing.

4.3.1 Triangle Split

A triangle is a planar object, and is representing a possibly non-planar portion of a surface. Any triangle, with area A_T, in the discretization can have at most the same area as the portion of the surface which it represents, A_S. Therefore, the representation deficit for a triangle, RD_T, can be defined as $RD_T = \frac{A_S - A_T}{A_T}$. Note that the areas are calculated in $\mathbb{R}^3$. The difference in surface area, $(A_S - A_T)$, is normalized by A_T so that the result is scale independent. Also, this is a representation *deficit* since $A_S \geq A_T$ is always true.

In order to apply the above definition of representation deficit to a mesh generation, a replacement for A_S must be determined since the area of the surface represented by A_S is not always able to be determined — or, most often, the area calculation is impractical. Generally, in order to reduce the representation deficit for a triangle the triangle is split by inserting an interior point. Any point that is inserted into the interior of the triangle would decrease the representation deficit — or at worst it will remain the same. However, the determination of where to split the triangle should be done in such a way that the representation deficit is minimized. This strategy of refinement, refining each triangle in such a way that the representation deficit is locally minimized, would take advantage of the optimal substructure the discrete topology.

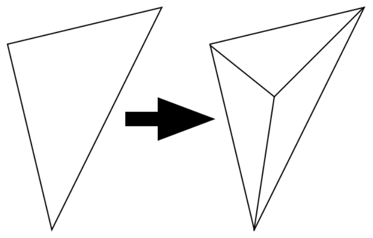

Figure 1: Triangle Split

The process of determining where to split a triangle is defined here by a locally optimizing an objective function defined for a triangle: Let $S(u,v)$ be a parameterized surface, D be be a discretization of the surface, and T be a triangle (included in D) defined by an ordered tuple of nodes, (n_i, n_j, n_k). These nodes are ordered such that the triangle normal is positive. Specifically, if $\vec{V_0} = \{n_j - n_i\}$ and $\vec{V_1} = \{n_k - n_i\}$ then the triangle normal, $N_T = \vec{V_0} \times \vec{V_1}$, is positive. Note that a two-dimensional cross-product (or two-dimensional curl) is a scalar quantity. Additionally, let a node on the interior of T be defined as n_O. The four nodes, $n_i, n_j, n_k, andn_O$ define three triangles, $T_i\left(n_i, n_j, n_0\right), T_j\left(n_j, n_k, n_0\right), T_k\left(n_k, n_i, n_0\right)$. If $A(T)$ is a function that calculates the area of a triangle in (x,y,z) space, then the optimization problem for finding the optimal position for n_0 in T can be stated as:

$$\begin{aligned}\underset{n_O}{\text{minimize}}\ O(T) &= -\tfrac{A(T_i)+A(T_j)+A(T_k)}{A(T)} \\ \text{subject to } N_{T_i} &> 0 \\ N_{T_j} &> 0 \\ N_{T_k} &> 0\end{aligned}$$

It should be noted that this definition of representation deficit would be difficult to derive or express for a topological entity other than a triangle. This is due to the inherent ambiguity in the definition of not only the surface area, but also the surface representation of (possibly) non-planar elements, e.g. non-planar, or skew, quadrilateral. [MORE POSSIBLY]

4.3.2 Edge Split

An edge in parametric space represents a (possible) curve on the surface. Any edge, with length L_E, can have at most the same length as the portion of the surface which it represents, L_S. Therefore, the representation deficit for an edge, RD_E, can be defined as $RD_E = \frac{L_S - L_E}{L_E}$. Note that the lengths are calculated in $\mathbb{R}^3$. The difference in length, $(L_S - L_E)$, is normalized by L_E so that the result is scale independent. Also, this is a representation *deficit* since $L_S \geq L_E$ is always true;

In order to apply the above definition of representation deficit to mesh generation, a replacement for L_S must be determined since the area of the surface represented by L_S is not always able to be determined – or, most often, the arc-length calculation is impractical. Generally, in order to reduce the representation deficit for an edge the edge is split by inserting an interior point. Any point that is inserted into the interior of the edge would decrease the representation deficit — or at worst it will remain the same. However, the determination of where to split the edge should be done in such a way that the representation deficit is minimized. This strategy of refinement, refining each edge in such a way that the representation deficit is locally minimized, would take advantage of the optimal substructure of the discrete topology. A method for generating a locally optimal edge split is detailed in [4, 5][OTHERS].

In addition, the fact that an edge split will change the surface area of the discretization should be considered. Since the overall justification of this work is reduce the *area* representation deficit of a discretization, the representation deficit will not be defined for an edge but for an edge-split. Therefore, for a given edge, only the edge-split that minimizes *area* representation deficit is considered. . The process of determining where to split a triangle is defined here by locally optimizing an objective function defined for an edge-split. Let $E(n_i, n_j)$ be an edge in D that is shared topologically by two triangles, $T_i(n_i, n_j, n_k)$, and $T_j(n_i, n_l, n_j)$. Additionally, let a node on the interior of the edge be defined as n_O. The five nodes, $\{n_i, n_j, n_k, n_l, n_O\}$ define four triangles, $T_a(n_i, n_O, n_k)$,

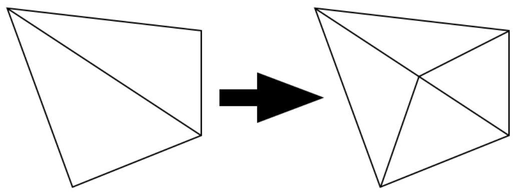

Figure 2: Edge Split

$T_b\,(n_O, n_j, n_k)$, $T_c\,(n_i, n_l, n_O)$, $T_d\,(n_O, n_l, n_j)$. The optimization problem for finding the optimal position for n_O on E can be stated as:

$$\begin{aligned}
\underset{n_O}{\text{minimize}}\; O(E) \;&=\; -\tfrac{A(T_a)+A(T_b)+A(T_c)+A(T_d)}{A(T_i)+A(T_j)} \\
\text{subject to } N_{T_a} \;&>\; 0 \\
N_{T_b} \;&>\; 0 \\
N_{T_c} \;&>\; 0 \\
N_{T_d} \;&>\; 0
\end{aligned}$$

In considering the edge swap operation it became apparent that the constraint of finding a candidate point, P_c, that topologically lies on the edge, E, is an "implicit" constraint. Is it required that P_c lie on E or should P_c be allowed to be inserted where most appropriate in the hull formed by the two triangles topologically adjacent to E?. [DO I EVEN WANT TO MENTION THIS? THEN I'LL HAVE TO GET INTO, "WELL WHY CONSTRAIN THE SEARCH TO ONLY TWO TRIANGLES? WHY NOT LET THE NODE TRAVEL UNTIL IT FINDS THE BEST PLACE?" AND THEN I'D HAVE TO COMPARE METHODS AND MORE CODING... IT'S INTERESTING, BUT I DON'T THINK THERE'S ROOM HERE. PRACTICALLY, EDGE SPLITS ARE USUALLY NEEDED SINCE THE TRIA SPLITS AND RECONNECTIONS TAKE CARE OF MOSTLY EVERYTHING.]

4.3.3 Edge Swap

Is it possible to have an optimal topology? If so, then it should be reachable through repeated edge flipping. Therefore, we should be able to write some sort of formulation for a pair of triangles and go from there? [SUZANNE] Comment: for the sake of not doing any more coding, if your ethics can let you come to the conclusion that this is either not possible or not practical then this would fit into the logic/justification in the next section.

4.3.4 Nodal Smoothing

For a node, N_i, the representation deficit can be defined only by comparing it to the node when located at another point in space. Here the comparison is bound by limiting the range of comparison within the edge-hull topologically adjacent to N_i. Formally, let N_i be a node in D that is shared topologically by n triangles, where n is the face-valence of N_i. The optimization for finding the optimal position for N_i can be stated as:

$$\begin{aligned}
\underset{N_O}{\text{minimize}}\ O(N) &= -\textstyle\sum_{j=1}^{n_t-1} A\left(T_j\right) \\
\text{subject to } N_{T_1} &> 0 \\
N_{T_2} &> 0 \\
N_{T_3} &> 0 \\
&\vdots \\
N_{T_{n-1}} &> 0 \\
N_{T_n} &> 0
\end{aligned}$$

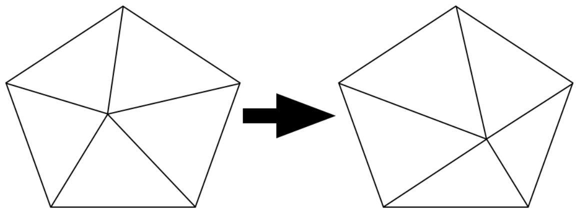

Figure 3: Nodal Smoothing

4.4 Boundary Refinement

[Is there a section needed on this? It'd be straightforward to talk about but would be pretty repetitive and I don't have it coded up yet. If I do talk about it then due diligence says I should show some results and there are already tons of variables to consider. This might get left out due to space concerns.]

5 Optimal Substructure vs Global Optimization

[NEED TO COME TO THE CONCLUSION HERE THAT GLOBAL OPTIMIZATION IS NOT PRACTICAL] [COMMENT ON HOW ITERATIVE REFINEMENT IS USED HERE, NOT SETTING UP SOME GLOBAL EQUATIONS]

5.1 Local Optimization via Pattern Matching

[Suzanne] I used a simple ”cross” pattern (up down left right center) and sized it based on the local geometry so that it could be moved inside of the triangles/hulls many times before encountering the boundary.

6 Element Quality

[SHOULD THIS EVEN BE A SECTION. IT NEEDS TO BE DISCUSSED BUT I'M NOT SURE THAT IT NEEDS IT'S OWN SECTION

7 Results

8 Conclusions

9 Acknowledgements

References

[1] S Bischoff and L Kobbelt, *Structure preserving CAD model repair*, Eurographics **24** (2005), no. 3.

[2] J. Casey (ed.), *A sequel to the first six books of the elements of euclid, containing and easy introduction to modern geometry with numerous examples*, 5th ed., Hodges, Figgis, & Co., Dublin, 1888.

[3] S Hermann and R Klette, *A comparative study on 2D curvature estimators*, Proc. of the International Conference on Computing: Theory and Applications (ICCTA), 2007, pp. 584–589.

[4] D. McLaurin, *Automated, curvature based edge-grid generation*, Proc. of AlaSim 2012 (Huntsville, AL), Alabama Modeling and Simulation Council, 2012.

[5] D. McLaurin and S. Shontz, *Automated edge grid generation based on arc-length optimization*, Proc. of 22nd International Meshing Roundtable (Orlando, FL), Sandia National Laboratories, 2013.