Engineering With Computers, October 2012, Volume 28, Issue 4, pp 331-344
DOI: 10.1007/s00366-012-0266-x

Particle Systems for Adaptive, Isotropic Meshing of CAD Models

Jonathan R. Bronson · Joshua A. Levine · Ross T. Whitaker

Abstract We present a particle-based approach for generating adaptive triangular surface and tetrahedral volume meshes from CAD models. Input shapes are treated as a collection of smooth, parametric surface patches that can meet non-smoothly on boundaries. Our approach uses a hierarchical sampling scheme that places particles on features in order of increasing dimensionality. These particles reach a good distribution by minimizing an energy computed in 3D world space, with movements occurring in the parametric space of each surface patch.

Rather than using a pre-computed measure of feature size, our system automatically adapts to both curvature as well as a notion of topological separation. It also enforces a measure of smoothness on these constraints to construct a sizing field that acts as a proxy to piecewise-smooth feature size. We evaluate our technique with comparisons against other popular triangular meshing techniques for this domain.

Keywords Adaptive meshing, particle systems, tetrahedral meshing, CAD

1 Introduction

Tetrahedral mesh generation is a key tool in the computer-aided design (CAD) pipeline. In particular, the conversion of shapes presented by the output CAD systems and solid modeling geometry kernels is necessary to provide input meshes for structural analysis, CFD and other CAE applications. The data from these systems is usually in the form of a boundary representation (B-Rep) made up of hierarchical connectivity (topology) and associated geometric entities. When the B-Rep is manifold and topologically closed, shapes of arbitrary geometric complexity can be produced. Care must be taken to provide accurate representations of these inherently piecewise-smooth solids while robustly preserving the curved features defined by the input topological description [18].

While many types of meshes are suitable for visualization and simulation, simplicial meshes have emerged as one of the dominant forms. Their popularity can be attributed to both the ease at which simplicial meshing algorithms can be implemented as well as the guarantees of termination and quality that can often be shown. Technologies to construct simplicial meshes vary greatly. Some of the most important paradigms include advancing-front [19,20,26], Delaunay refinement [10,25,29], and particle systems [22,33]. However, to build meshes that are adaptive, many variants of these techniques require an input oracle that evaluates a sizing field over the domain [4,8,23,27,32]. An early exception is the approach of Dey et al. [12] that uses Delaunay refinement for meshing smooth domains. Using the dual Voronoi diagram and the concept of *poles* [2], this algorithm automatically refines based on a simultaneously computed approximation of the *local feature size* (distance to the medial axis) of the shape whose accuracy increases as mesh density increases.

Local feature size of smooth shapes is a natural choice to use as a field to adapt to; however, most CAD models are inherently non-smooth. A notion of local feature size for piecewise-smooth shapes has been defined [8] by coupling local feature size for the smooth regions with a topological condition called *gap size* [9]. Computing this measure robustly is a significant challenge. The approach in this work aims to automatically infer a global sizing field of equivalent expressivity to [8] while using only locally available information as done by [12]. Such a technique must force a compromise, ours is to construct a proxy for feature size that is Lipschitz continuous by coupling curvature adaptivity with a topological separation condition.

Particle systems are an ideal framework for sampling parametric domains since they only require local calcula-

Scientific Computing and Imaging Institute, Salt Lake City, UT, U.S.A.
{bronson,jlevine,whitaker}@sci.utah.edu

tions. We can minimize energy by movements solely within the parameter space of each surface patch while knowing each particle stays within the confines of the surface in world space. Computing good quality triangulations from these samples can leverage the same benefits. We directly build triangulations on the particles in parameter space using 2D Delaunay triangulations (implemented by Triangle [28]). As 3D Delaunay triangulations can be expensive to compute, this provides a significant savings when only surface meshes are required. While this approach cannot immediately build triangles that are 3D Delaunay, we can improve their quality significantly by local modifications (e.g. edge flipping) that consider the world space positions of vertices. The resulting surfaces meshes make ideal inputs for volumetric meshing approaches, such as TetGen [30].

1.1 Contributions

This work focuses on automatic techniques for building triangular meshes of the boundary surface, and ultimately tetrahedral representations of the interior solid. We also improve the state-of-the-art for particle system-based techniques; our contributions can be summarized as the following:

- An automatic technique for constructing isotropic surface meshes by minimizing a world-space energy through parameter-space movements.
- Hierarchical sampling of features in increasing dimension, inspired by weighted Delaunay-based approaches [8, 11].
- Algorithmic control for both uniform and adaptive sampling, *without* requiring a pre-computation of global feature size needed by similar particle-based approaches [23].
- Fast mesh generation of these samples through the use of the 2D Delaunay triangulation in parameter space and 3D Delaunay edge flipping [6].
- Experimental evaluation that compares our approach to existing techniques [7, 15] for mesh generation of CAD domains.

2 Related Work and Background

While the history of tetrahedral mesh generation began much earlier, a shift in the emphasis of techniques has become popular within the past decade. In particular, variational approaches, i.e. based on energy minimization, have become one of the most important tools for mesh generation. Alliez et al. [1] describe a variational technique for mesh generation that couples Delaunay refinement with a relaxation process for vertex locations. This algorithm and later variants [31, 32, 34, 37] base their energy minimization on a sizing field for particle density coupled with an energy minimization grounded in the notion of a *centroidal Voronoi diagram* [14] and its dual, the *optimal Delaunay triangulation* [5]. Consequently, these meshing algorithms can generate nearly isotropic elements, as a byproduct of the centroidal Voronoi condition, as well as leveraging many of the benefits of Delaunay refinement techniques.

However, one deficiency is the need for knowledge of an element sizing field *a priori*. Computing a sizing field is considered expensive. Often, approaches for computing sizing fields are based on the medial axis [13] or quadratures of mesh elements [3], and thus can require $O(n^2)$ computations of dense point clouds to build accurate results. One recent solution of Tournois et al. [32] solves this problem by alternating a variational phase with a refinement phase. After each level of refinement, the sizing function is updated before switching back to variational phase. This interleaving allows the available information to drive the computation of a sizing field instead of necessitating a preprocessed computation. We aim to improve upon this theme by allowing an energy minimization based on particle systems to automatically improve its approximation of the sizing field.

A second thrust of recent algorithms is to provide provably good algorithms for meshing piecewise-smooth shapes. This general class describes shapes with a topological description in the form of a piecewise-smooth complex of k-cells that are compact subsets of k-manifolds. We use the same definition as Cheng et al. [8]. In summary, surface patches (2-cells) can meet non-smoothly at curves (1-cells) bounded by points (0-cells). Two k-cells are *adjacent* if one is on the boundary of the other.

Similar to the B-Rep definition, each k-cell has an associated geometric description. Recent Delaunay-based approaches [8, 24] for meshing this domain have been able to provide topological correctness guarantees using either weighted Delaunay triangulations [8] or bounding the angle deviations between smooth patches [24]. A missing piece to the implementations of these algorithms is the ability to adapt to a sizing field, primarily because there is no consensus on what is the correct sizing field for non-smooth shapes and how best to compute it. However, they do show that a careful sampling of points on sharp creases can preserve the features of a shape. Our approach is a natural extension of this work, but instead of requiring an accurate sizing field to guarantee topological correctness, our scheme will build watertight meshes provided a few easily satisfied conditions are met by the particle system (described in Section 4.3).

2.1 Particle Systems

At the core of our meshing scheme is a paradigm for sampling shapes using particles. The idea of using repulsive point clouds to (re-)sample a mesh was first introduced by Turk in the context of polygonal remeshing [33]. The first full particle system for meshing was later developed by Witkin

and Heckbert [35]. Their technique was primarily used as a mechanism to sample and control implicit surfaces, which was notoriously difficult under other schemes at the time. The key idea behind their work was the introduction of a Gaussian energy function to control the interaction between particles. Improvements to their scheme were made by Hart et al. [17]. Yamakawa and Shimada proposed a meshing scheme similar to particles by using packings of ellipsoidal bubbles [36].

Meyer et al. [21] formulated a more robust and stable solution for evolving particle systems. The new energy kernel was a modified cotangent function, with finite support. By falling off to a finite range, the resulting particle systems were more stable and more quickly lead to ideal packings. Additionally, this kernel was nearly scale invariant. Meyer et al. [23] later introduced a hierarchical scheme for particle-based sampling multi-material surfaces. For such datasets, the boundaries between the different materials can be represented as a piecewise-smooth complex. While without the formal guarantees of [8], they use a similar strategy of hierarchically sampling topological features in increasing dimension to build consistent, watertight meshes.

3 Particle System Formulation

In this section we provide the mathematical formulation behind our particle system. We define the total energy in the system as the sum of each energy E_i calculated with respect to particle p_i. Each particle p_i has a corresponding σ_i value representing the radius of its ball of influence B_i centered at p_i. It is the varying of σ_i that provides adaptivity. Each energy E_i is the sum of the energies between particle p_i and all neighboring particles p_j. Particles p_i and p_j are considered neighbors if either p_j falls within B_i or if p_i falls within B_j. We use a variation of the modified cotangent for the energy (1) between any two particles, E_{ij}. By varying σ_i, the potential function must be scaled to account for this new, lopsided interaction between particles. Thus, we scale both the modified cotangent function and its derivative (2) by σ_i.

$$E_{ij} = \sigma_{ij}\left[\cot\left(\frac{|r_{ij}|}{\sigma_{ij}}\frac{\pi}{2}\right) + \frac{|r_{ij}|}{\sigma_{ij}}\frac{\pi}{2} - \frac{\pi}{2}\right] \tag{1}$$

$$\frac{dE_{ij}}{r_{ij}} = \frac{\pi}{2}\left[1 - \sin^{-2}\left(\frac{|r_{ij}|}{\sigma_{ij}}\frac{\pi}{2}\right)\right] \tag{2}$$

In this form, $|r_{ij}|$ is the distance between particles p_i and p_j and the value σ_{ij} is taken to be the max of σ_i and σ_j. The hexagonal packings that result from this and related particle systems requires the particles to reach a critical density on the surface being sampled. For any surface and any set of σ values, there will always be an ideal number of particles, but calculating this number is not tractable. Like previous systems, we use splitting and deleting to control energy densities. Particles follow the rules:

$$E_i^* = E_i(1+\varepsilon) \tag{3}$$

$$\text{if } E_i^* < 0.35E_i^{\text{ideal}} \quad \text{Split} \tag{4}$$

$$\text{if } E_i^* > 1.75E_i^{\text{ideal}} \quad \text{Delete} \tag{5}$$

Using a hexagonal lattice packing as our notion of an ideal distribution, the ideal energy E_i^{ideal} for a particle p_i is six times the energy felt between p_i and p_j at the characteristic distance of approximately 0.58 [21]. This reflects the energies felt from the six particles in the one-ring neighborhood (connected directly by one edge in a hexagonal lattice). The two-ring particles (connected through two edges in a hexagonal lattice) are at distance 1.0 from particle p_i, allowing Equation (6) to compactly describes this relationship. Additionally, we scale this value by σ_i to match the scaling of actual energies.

$$E_i^{\text{ideal}} = \sigma_i 6E(\beta), \text{ with } \frac{|r_{ij}|}{\sigma_{ij}} = \beta = \frac{0.5}{\cos(\pi/6)} \approx 0.58 \tag{6}$$

Since one cannot predict what an ideal neighborhood will look like in the adaptive case, the ideal energy is less precise than in the constant case. This leads to more frequent splits and deletes for higher local variation, but ultimately provides much better packings than if the original energy was not scaled proportional to σ. An alternative to this approach would be to use a notion of scaled distance $d' = \frac{d}{\sigma}$, and forego the σ_i scaling. Then, to still achieve the high quality packings, a different scheme for deletion of poorly configured particles would need to be devised.

To allow the system to adapt to splits and deletes, E_i is biased by a small random number, $0 \leq \varepsilon \leq 1$, in Equation (3). This makes the discrete energy jumps have less of an impact on the speed at which the system stabilizes, by allowing time for the system to adapt between jumps. Additionally, this can help resolve any regions which are stuck in bad configurations. As the solution to the system converges, this bias can be adjusted to stop splits and deletes all together, ensuring termination.

To find the ideal packing of particles, we use a Newton-Raphson scheme, updating particle information after each movement (Equations (8), (9), and (10)). Each particle maintains its position in both world space (x_i^{xyz}) and parameter space ($x_i^{\mu\nu}$). Particles move with a velocity v_i generated by inter-particle forces between neighbors. Though energies between particles are computed in 3D world space, particles move strictly in parametric space (10), avoiding the error-prone projection onto the surface that results from 3D movements. Taking these steps in parameter space only requires a change of coordinates (7)(9). From each surface we can

obtain parametric derivatives **du** and **dv**, which we use to construct the metric tensor **M** and Jacobian **J** needed for the inverse transformation.

$$\mathbf{M} = \begin{bmatrix} E & F \\ F & G \end{bmatrix} \qquad \begin{aligned} E &= \mathbf{du} \cdot \mathbf{du} \\ F &= \mathbf{du} \cdot \mathbf{dv} \\ G &= \mathbf{dv} \cdot \mathbf{dv} \end{aligned} \qquad \mathbf{J} = \begin{bmatrix} \mathbf{du}^T \\ \mathbf{dv}^T \end{bmatrix} \tag{7}$$

$$\mathbf{v}_i^{xyz} = \sum_{j \in \mathcal{N}} dE_{ij} \cdot \frac{\mathbf{r}_{ij}}{|\mathbf{r}_{ij}|} \tag{8}$$

$$\mathbf{v}_i^{uv} = \left(\mathbf{M}^{-1}\mathbf{J}\right) \mathbf{v}_i^{xyz} \tag{9}$$

$$\mathbf{x}_i^{uv} = \mathbf{x}_i^{uv} + \mathbf{v}_i^{uv} \tag{10}$$

As mentioned earlier, we use a hierarchical sampling scheme, which works well for parametric models. First, we place particles on the 0-cells, the intersection of edges on the models. Next, we place particles on the 1-cells and allow them to be optimized. Finally, we place particles on the surface patch interiors and the final optimization proceeds. At each phase, the new optimization uses the fixed positions from the previous phase, ensuring consistency across surface patch boundaries.

3.1 Inferred Sizing Field

We recognize that there are several factors that often determine good sizing fields: local curvature, some notion of feature size, and a desired level of adaptivity. Additionally, users may have desires for mesh resolution limits, both minimum and maximum triangle or edge size. Other domain-specific factors also often come into play. In this section, we illustrate the constraints we would like to place on a sizing field. We show that these constraints can be inferred in a reliable way and used to form a smooth sizing field during energy minimization.

We aim for meshes that provide arbitrary levels of geometric accuracy and adaptivity, using high quality isotropic elements. In order to provide high quality elements, particle systems require enough spatial freedom to be able to move to lower energy states. Thus, the distance between nearby k-cells imposes its own sizing constraint on the particles. Thus, in order to determine the sizing field value σ_i at a particular point p_i on a model, we must consider the constraints placed on this location by curvature, topological distance, and desired level of adaptive continuity. We refer to these constraints as σ_κ, σ_τ, and $\sigma_{\mathscr{L}}$, respectively. The actual sizing field value at a particle location is resolved by finding the σ_i that respects all constraints. This can be expressed compactly as:

$$\sigma_i = \max\left\{\sigma_{\min},\ \min\left\{\sigma_{\max}, \sigma_\kappa, \sigma_\tau, \sigma_{\mathscr{L}}\right\}\right\} \tag{11}$$

Curvature

Since the curvature at a point is defined as the inverse of the radius of the osculating circle at that point, a reasonable default sizing field value is the radius of that circle itself. Thus, we use $\sigma_\kappa = \frac{1}{\kappa}$, which can be easily computable for parametric surfaces, or queried by middleware packages.

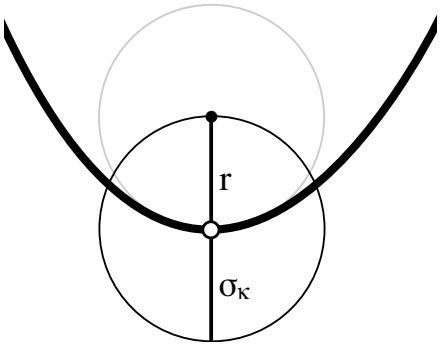

Fig. 1 Default curvature constraint on sizing field.

To increase or decrease the field relative to this radius, a scaling factor s_κ is exposed as a user parameter. Given a unit system, this value can be used to provide constraints to respect geometry to arbitrary levels of accuracy. Finally, $\kappa_{\min}$ and $\kappa_{\max}$ values are user parameters used to handle straight edges and arbitrarily high curvature, respectively. These form the total bounds for the sizing field as:

$$\sigma_{\min} = 1/\left(s_\kappa \kappa_{max}\right) \tag{12}$$

$$\sigma_{\max} = 1/\left(s_\kappa \kappa_{min}\right) \tag{13}$$

For 2-cells, we use the maximum principal curvature, since this size will dominate an isotropic sampling. For 1-cells, using the curvature of the edge itself is insufficient. The maximum principal curvature on both intersecting surfaces must also be considered, since the curve may either be a trim or a boundary curve, and there is no way of knowing which curvature will dominate. Last, 0-cells use the maximum curvature of all 1-cells terminating at its point.

Figure 2 illustrates the intuitive effect of modifying the curvature scaling parameter s_κ to achieve a better geometric fit. For practical purposes, a good scaling parameter will usually be based on the unit of measurement of the model.

Gap Size

If available, using the distance to the model's medial axis would provide a sizing field constraint that generates good samplings in a particle system. However, computing the medial axis on parametric models is a difficult task and still an active area of research. Instead, we use the notion of *gap size*, introduced by Chang & Poon [9] in the context

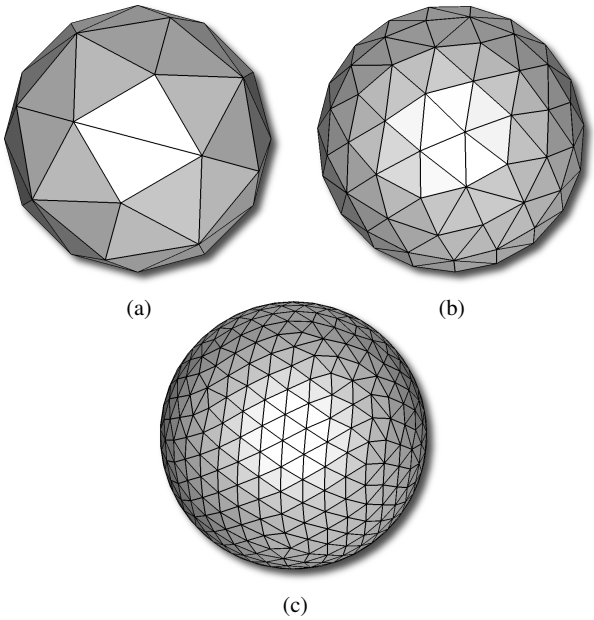

Fig. 2 Effects of changing curvature scaling parameter s_κ

of piecewise linear mesh generation. For a point p on a k-cell c, its gap size is the distance to the nearest non-adjacent (i.e. not on the boundary of c) cell. This measure also preserves topological features inherent to the model's decomposition into parametric patches. Depending on the model and the way it was created, this measure may sometimes be equivalent to definitions of local feature size. Figure 3 shows an example where the two are equivalent by a factor of one half.

We make the assumption that the topological representation provided as input for the CAD model should be respected in an output mesh. A byproduct of this approach is that some models have adaptivity in regions that are of little benefit to representing the geometry of the model. One could remove adaptivity in regions that do not actually need it by taking a pass over the model and detecting topological junctions that are G^1 continuous, and flagging them to be ignored. The remaining geometrically discontinuous junctions could then be preserved using our sampling scheme.

Gap size is approximated directly from inter-particle relationships. Particles store which k-cell they lie on, and each k-cell stores which particles lie on it. We define the topological constraint σ_τ to be the shortest distance from particle p_i to another particle p_j lying on a non-adjacent feature. That is, a 0-cell particle interacting with another 0-cell particle, a 1-cell particle interacting with another 1-cell particle, or a 0-cell particle interacting with a 1-cell particle that does not terminate at that 0-cell. This notion can be extended to 2-cells as well. We further provide a scaling factor s_τ as a user parameter to allow for higher densities of particles within these regions. This proves useful when sampling highly elongated surfaces, with parallel k-cells. Scaling the distance σ_τ allows more rows of particles, allowing for better energy minimization, and ultimately better triangulations.

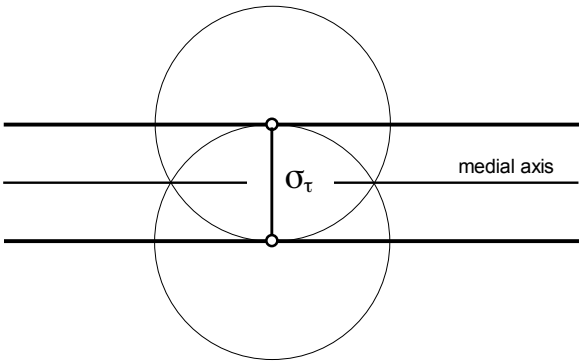

Fig. 3 Gap size constraint on sizing field. In this case, the gap size is equivalent to the distance to the medial axis by a factor of two.

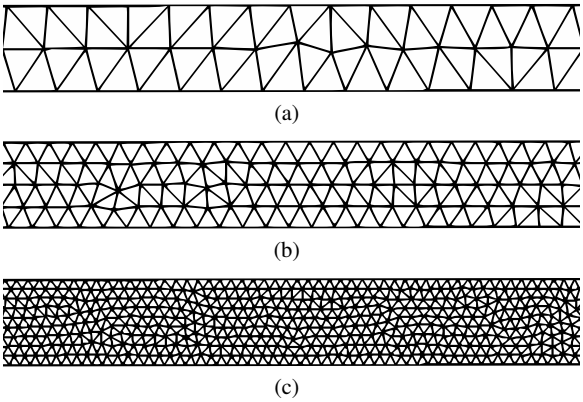

Fig. 4 Effects of changing topological scaling parameter s_τ

Figure 4 illustrates how changing the topological scaling parameter s_τ effects the distribution of particles. As the parameter decreases, more particles can fit in the same space enclosed by neighboring 1-cells. Depending on the minimum allowable curvature, and scaling, this parameter may or may not come into play.

Lipschitz Continuity

In order to provide finer control over the adaptivity of the particle samples, the system adheres to a Lipschitz constraint $\sigma_{\mathscr{L}}$ that enforces the Lipschitz continuity $\mathscr{L}$ on the sizing field. The Lipschitz condition can be expressed in terms of our formulation as:

$$\left|\mathbf{x}_i - \mathbf{x}_j\right| \leq \mathscr{L}\left|\sigma_i - \sigma_j\right| \tag{14}$$

The $\sigma_{\mathscr{L}}$ induced by this constraint is simply the minimum allowable value that satisfies this condition:

$$\sigma_{\mathscr{L}} = \min_{j \in N}\left\{\left|r_{ij}\right|\mathscr{L} + \sigma_j\right\} \tag{15}$$

Respecting this Lipschitz continuity provides more gradual adaptivity between areas of high and low particle densities. Lower values of $\mathscr{L}$ produce samplings that result in more isotropic triangles, while large values provide increased levels of adaptivity, at the cost of isotropy. Figure 5 illustrates the effects of changing the Lipschitz parameter, $\mathscr{L}$, for a conic surface patch. The parameter is set values ranging from 0.0 (a) to 0.3 (d). When $\mathscr{L}$ goes to zero, a uniform sizing field is produced, fitting the smallest constraint on the model. In this case, the sharp tip of the cone is the limiting feature. We found a default value of 0.3 provides a good trade-off between triangle quality and adaptivity.

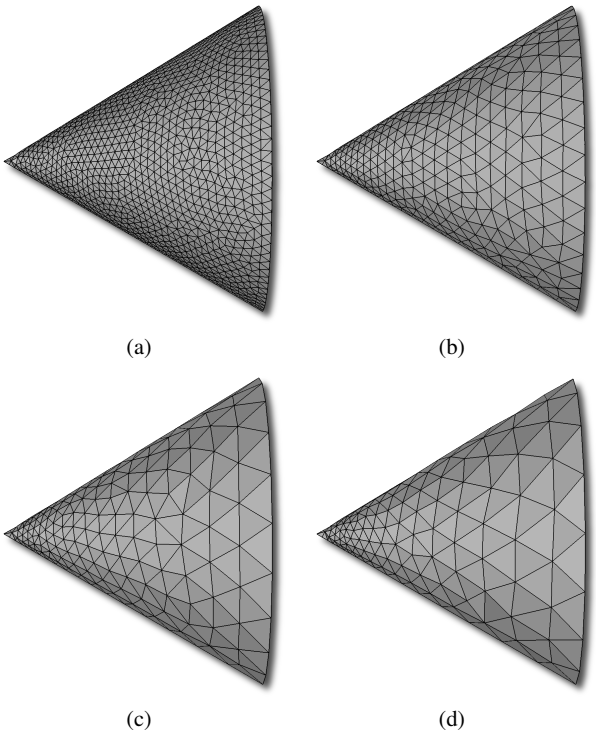

Fig. 5 Effects of changing Lipschitz parameter $\mathscr{L} = 0.0$ to $\mathscr{L} = 0.3$, (a) to (d) respectively.

It is worth noting that the Lipschitz continuity is not satisfiable for arbitrary surfaces. Since we place samples hierarchically, it is possible the sizing field may need to adapt more quickly on the interior of the surface than it does on the edges. In these situations, the Lipschitz constraint needs to be relaxed to allow the sizing field to adjust.

4 Algorithm

Our implementation takes as input a parametric model and outputs a triangular mesh. We use the middleware package CAPRI [16] to provide us direct geometry access to shapes generated by CAD software. It also gives access to the topology of the model, including access to the 0-cells, 1-cells, and 2-cells, and their boundary adjacencies. In this section, we elaborate only on the parts of the update algorithm that are independent from the middleware.

4.1 Particle Optimization

The sampling algorithm consists of three phases: Phase 1 optimizes 0-cell and 1-cell samples based strictly on the curvature and the Lipschitz constraints, σ_κ and $\sigma_{\mathscr{L}}$. Phase 2 continues the 0-cell/1-cell optimization, but includes the topological constraint σ_τ. Finally, Phase 3 optimizes samples with surface patches. A phase is considered complete when the change from one iteration to the next drops below some threshold.

We initialize Phase 1 by placing one particle on each 0-cell, and one particle on the midpoint of each 1-cell. Along the 1-cells, splitting happens to increase particle density as the sizing field is inferred. Similarly, if user parameters make any 1-cell particle unnecessary, it will be deleted. Phase 3 is initialized by placing k random samples in the parameter domain of the surface. Each iteration of the optimization, a particle updates both its position as well as its sizing field value σ_i. A scaling factor λ_i is used to increase stability. Pseudocode for the updates of particle positions is shown in Algorithm 1.

Algorithm 1 Position Update

for all particles **do**
  Compute energies E_i, dE_i (Equations 1,2)
  Compute velocity $\mathbf{v}_i^{xyz}$ (Equation 8)
  Transform to parameter space, obtain v_i^* (Equation 9)
  Compute scaling $v_i^{*\text{new}} = \lambda_i v_i^*$
  Compute new particle position u_i^{new} (Equation 10)
  Transform to world space x_i^{new}
  Compute the new energy value, E_i^{new}
  if $E_i^{\text{new}} >= E_i$ **then**
    if $\lambda_i <= \lambda_{\min}$ **then**
      Skip to next particle on list
    end if
    Decrease λ_i by a factor of 10 and go back to Step 3.
  end if
end for

4.2 Corner Cases

The motivation for splitting the optimization of 0-cells and 1-cells into two phases is illustrated in Figure 6. When it comes to enforcing the topological condition, just as feature size goes to zero in discontinuous corners, so does our notion of topological feature size. Left unchecked, particles in corners would continually shrink their σ_τ, split and move in closer to the corner.

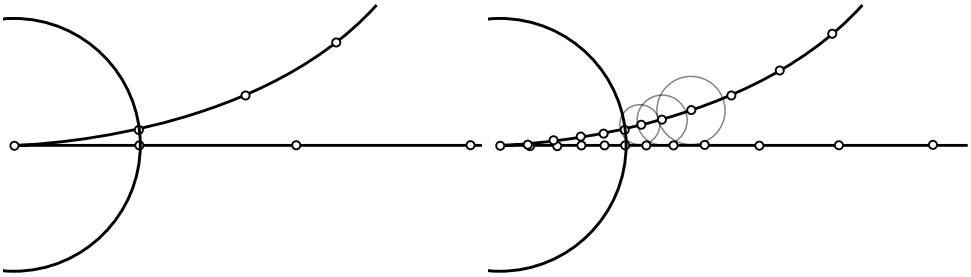

Fig. 6 (a) Phase 1, respecting only curvature and Lipschitz constraints. (b) Phase 2, additionally respecting topology constraints outside the corner ball.

To curtail this response, we detect and label corners in the first phase. Figure 6(a) shows what one corner might look like after Phase 1 has completed. Notice only the curvature and Lipschitz constraints have been met. The σ_i value of the particle on the 0-cell is saved as the size of the 0-cell's corner ball. This is similar to the protecting ball idea in Delaunay meshing [9]. Figure 6(b) shows the same corner after Phase 2 has completed. The topological constraint is satisfied for all particles that lie outside of the corner ball. The particles inside adapt smoothly and guarantee the sampling terminates. An alternative approach would be to fix the position of particles laying on this corner ball boundary. The downside to such an approach is that it could easily violate the Lipschitz constraint. With corner cases considered as part of the σ_i constraint, the pseudocode for the sigma update is shown in Algorithm 2.

Algorithm 2 Sigma Update

for all particles **do**
  if $p_i \in$ 1-cell **then**
    for all $p_j \in N$ **do**
      if edge(p_i) $\neq$ edge(p_j) and not in corner **then**
        Set topology constraint $\sigma_\tau = \min\left\{\sigma_\tau, \left|x_i - x_j\right|\right\}$
      end if
      Set Lipschitz constraint $\sigma_{\mathscr{L}} = \min\left\{\sigma_{\mathscr{L}}, \left|r_{ij}\right| \mathscr{L} + \sigma_j\right\}$
    end for
  end if
  Satisfy Eq. 11: $\sigma_i = \max\left\{\sigma_{\min}, \min\left\{\sigma_{\max}, s_\kappa \sigma_\kappa, s_\tau \sigma_\tau, \sigma_{\mathscr{L}}\right\}\right\}$
end for

4.3 Triangulation in Parameter Space

Our formulation builds a distribution of samples in 3D. To construct a mesh from these samples, one alternative would be to directly build a 3D Delaunay triangulation of the point cloud. Through pruning and filtration one could construct the surface triangulation and interior tetrahedralization. However, because of the parametric nature of the system, we can instead construct triangulations for each 2-cell and its boundary in the parameter space. This dimensional reduction gains us speed in terms of building the triangulation. However, particles distributed well in 3D may be in poor configurations in their corresponding parameter space, we account for this using local modifications after constructing an initial triangulation.

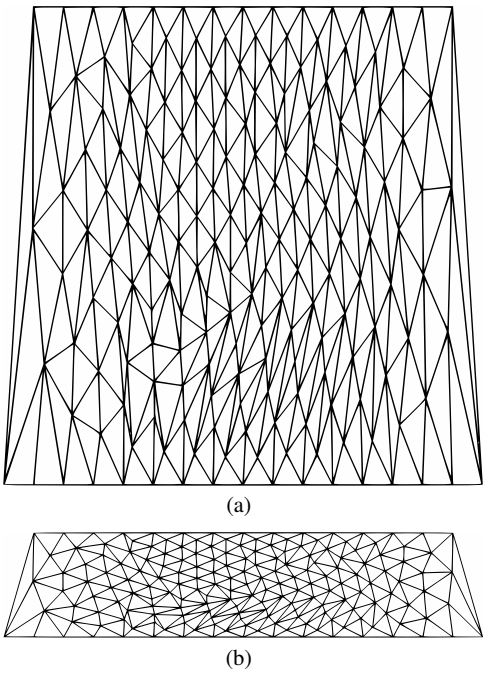

Fig. 7 Example parameter space before (a) and after (b) affine scaling.

Since the parameter space set of samples may be highly distorted, we first perform an affine scaling to regularize the 2-cell as much as possible. Figure 7 shows an example of one such scaling, transforming the parameter space 7(a) into parameter space 7(b). We obtain this transform by solving the least squares solution to the transform that best preserves Euclidean distances. This constraint can be expressed as:

$$\min_{\mathbf{A}} \left\| \mathbf{A} \left|\mathbf{u}_i - \mathbf{u}_j\right| - \left|\mathbf{r}_{ij}\right| \right\|^2 \tag{16}$$

This constraint urges pairwise distances between particles' 2D parametric coordinates to be as similar as possible to pairwise distances in 3D euclidean space. While we found this transform to be sufficient for the models we tested, clearly more sophisticated transformations could be utilized to remove distortion more uniformly in parameter space. It is conceivable that a sufficiently distorted surface patch might not benefit from a transformation as simple as an affine scaling. Given that such a system already requires the ability to query the surface patch, an optimal transformation would probably need to take surface information into account as much as the world space position of particle samples.

Next, for each 2-cell we construct the 2D Delaunay triangulation of its particle samples as well as the samples on its boundary curves using Triangle [28]. This triangulation has two problems which we address. (1) This triangulation includes extra triangles (within the convex hull) that may in fact be trimmed portions of the parameter space. (2) The quality of the triangles lifted back in 3D may be poor.

Our hierarchical sampling scheme is devised in part to correct for the first concern. The samples of the 1-cells create a dense sample of each curve in both spaces. Moreover, the particle energies on these samples repel particles within neighboring 2-cells away. As a result, these samples act in a role similar to a weighted sample used in recent Delaunay refinement schemes [8]. If each curve is sampled dense enough so that in the 2D triangulation each pair of adjacent 1-cell particles has a Delaunay edge, then we can recover the 1-cell. Will a formal proof of this fact remains future work, our experimentation indicates we can handle arbitrarily sharp edges, without the need for a weighted Delaunay. If we were using a full 3D Delaunay triangulation without weights, we would suffer from angle limitations, as noted by [24].

Having the 1-cells recovered as a sequence of edges in the 2D Delaunay triangulation is sufficient to prune away triangles that are exterior to trims. Once we have pruned these triangles, the remaining triangles are lifted to form the surface triangles of the 2-cell in 3D. However, because of the distortion of the 2-cells parameterization, they may be of poor surface quality. A recent result of Cheng and Dey [6] discusses a scheme to use edge flipping to recover Delaunay surface triangulations. In their technique, a *Gabriel* property is enforced for each triangle, requiring that each triangle's diametric ball is empty (a stronger version of the Delaunay property). We use a similar scheme, for each edge, we check if two triangles that share that edge have diametric balls that do not contain the opposite, unshared vertex. If they do not, we flip. The recent theoretical result of Cheng and Dey showed this property would only work for ε-dense surface triangulations; however, we found our point sets to be flippable in nearly all cases. A few rare edges could flip indefinitely. To break these ties, we selected the triangle pair that maximize the minimum circumradius (similar to the Delaunay property). Figure 8 shows an example of a two-patch sphere model that undergoes this process. Figure 8(a) shows the mesh resulting from the affine scaled 2D Delaunay triangulation, while Figure 8(b) shows the mesh after edge flipping.

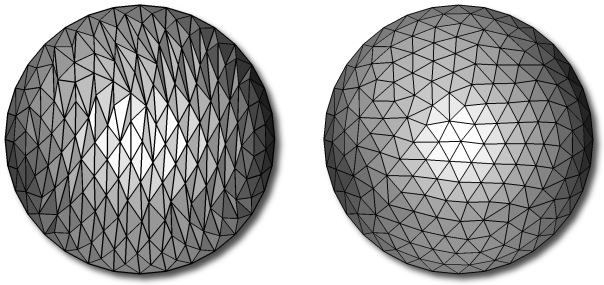

Fig. 8 Parametric Delaunay before (a) and after (b) edge flips.

4.4 Tetrahedralization

The resultant triangulations are not true 3D Delaunay as we do not ensure that each triangle has a circumball empty of all other points. However, we found they still had two desirable properties. First, nearly all triangles had excellent aspect ratio (shown in the experimental results). Second, these meshes were quite suitable for a constrained Delaunay triangulation that preserves each triangle. We use TetGen [30] to generate high quality tetrahedralizations of these surface meshes.

5 Evaluation

We break the evaluation of this meshing technique into two parts. First, we compare it with two other popular triangular meshing techniques for this domain. Then, we evaluate the technique for its own sake, including: strengths, weaknesses, and convergence properties.

5.1 Method Comparison

We compare our particle system technique (PSYS) to DelPSC [7] and CAPRI's meshing algorithm [15]. We chose these methods because they were both readily available, and actively used in practice. We evaluate the three methods using surface triangle aspect ratio and volume tetrahedra aspect ratio. To provide a fair test environment, we hand tuned the parameters of each algorithm to generate surface meshes of

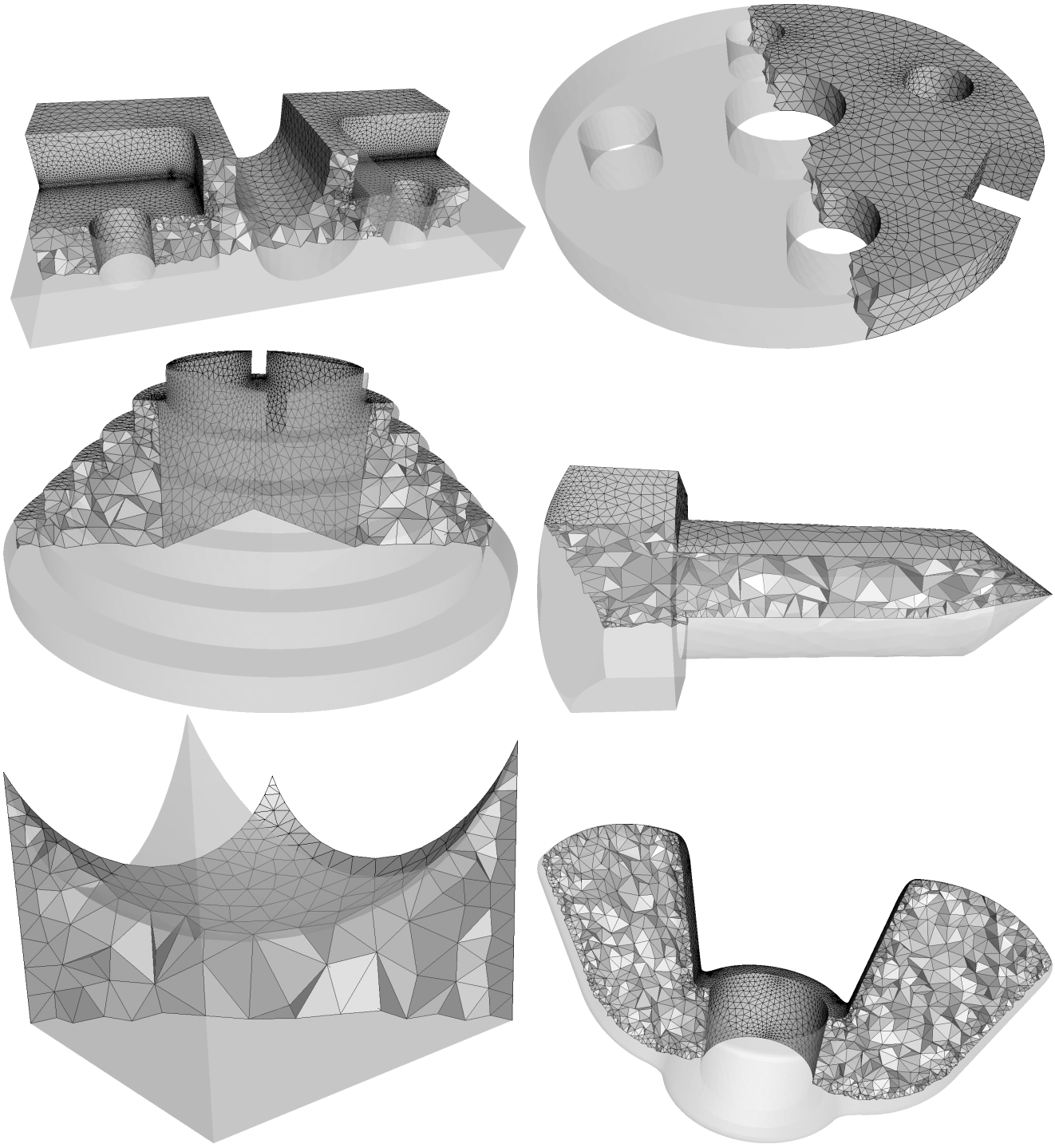

Fig. 9 From left to right, top to bottom: the output volume meshes of PSYS for the Block, Disk, Hanoi, Screw, Table, and Wingnut models.

approximately the same number of vertices. PSYS uses default settings of $s_\kappa = 2$, $s_\tau = 0.5$, and $\mathscr{L} = 0.3$ for all input models.

Figure 9 shows the output volume meshes of PSYS while Figure 10 shows a comparison of the surface meshes for each of the three algorithms. In the insets of Figure 10 we show close up views of mesh to highlight how PSYS's adaptivity can build superior geometric approximations using the same number vertices. While the shape of elements is good for all meshes, PSYS can produce especially isostropic triangles. Even in areas of high variability for curvature, PSYS was able to adapt especially well without sacrificing element quality.

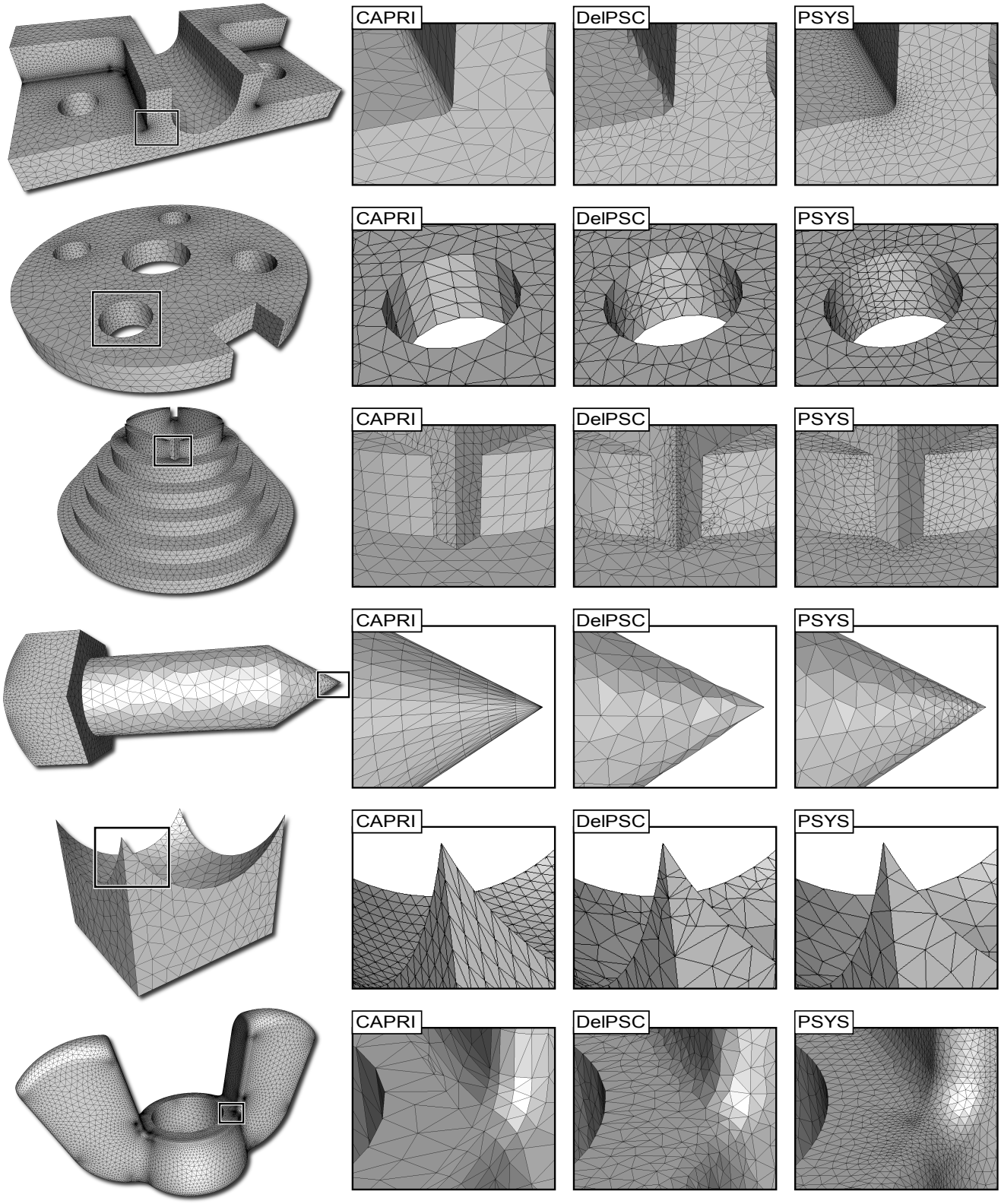

Fig. 10 Various models meshed. From top to bottom we show the Block, Disk, Hanoi, Screw, Table, and WingNut meshes for PSYS. Insets show comparison between CAPRI, DelPSC, and PSYS. For some inputs (such as the Block and WingNut) DelPSC approximated smooth regions without sampling topological curves.

To investigate this aspect further, we report the geometric quality of elements on both the surface triangulation as well as the volume tetrahedra. We use the aspect ratio (circumradius to shortest edge length ratio) as a criteria for mesh element quality. Figure 11 shows plots of both mesh quality statistics for the mesh of each model using each algorithm. For triangle quality, in Figure 11(a), it is interesting to note that PSYS did exceptionally well in the median case. DelPSC has a user parameter to bound triangle quality, the conservative theoretical bounds to guarantee termination require it to be set near 1.0. In addition, DelPSC does not improve element quality near sharp features. As a result, it outperforms CAPRI's surface meshing scheme (which has no refinement parameters for triangle quality), but its median behavior is slightly worse than PSYS.

For volume meshing, the algorithms all behave quite similarly in the median case as shown by Figure 11(b). Since TetGen is used for two of the algorithms, this is not an unexpected result. The full 3D Delaunay refinement used by DelPSC also achieves results on par with the other algorithms. We remark that setting naive parameters to CAPRI's meshing algorithm would build surfaces meshes not suitable to TetGen. Since CAPRI provides no direct control over the quality of surface triangles, if their angles are too sharp TetGen's refinement could require an impossible number of insertions. We found that PSYS's good quality triangles always lead to suitable inputs for TetGen.

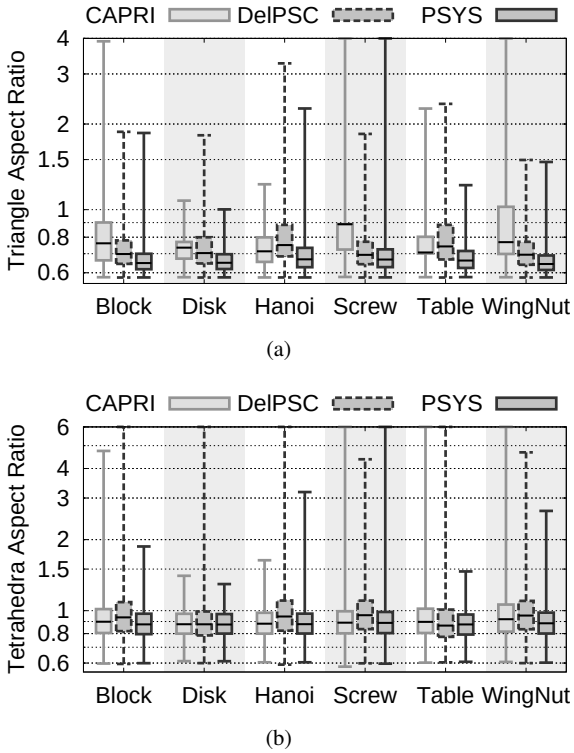

Fig. 11 Box plots of the aspect ratios (circumradius/shortest edge length) on a log scale. We show for triangles (a) and tetrahedra (b) of each output mesh of each algorithm. These plots show the minimum, 25th percentile, median, 75th percentile, and maximum aspect ratio over all elements in each mesh.

5.2 Evaluating PSYS

For most models, we are able to obtain good distributions in only a few hundred iterations total. The convergence rates for the particle system to find optimal distributions are based primarily on the number of particles needed and the level of adaptivity. Thus, most iterations take place in Phase 3 of the algorithm. It should be clear why more samples require more iterations, as information has a longer path to travel before reaching equilibrium. How adaptivity comes into play is more subtle. We enforce the Lipschitz condition at every iteration, which means boundary values pull down local σ values very quickly. This change propagates outward to areas that can otherwise tolerate a larger σ. This means surfaces may become oversampled prior to fitting the Lipschitz continuous field. As the field values increase, so do energies, and particles begin to delete to make room for particles of larger σ values. Relaxing the Lipschitz condition towards the beginning of the energy minimization could provide improved converge rates. Additionally, relaxing the energy requirements for insertion and deletion can improve convergence rates, but at the cost of less ideal packings.

5.2.1 Performance

The execution time of an iterative particle meshing algorithm is based on two factors: The cost of running a single iteration, and the number of iterations required for convergence. A naive particle system can behave like an n-body system, requiring an $O(n^2)$ update cost per iteration. While the work in this paper was motivated by quality, we utilized a gridded binning structure to reduce the number of particle-particle lookups for neighborhood computations. More efficient spatial binning structures (e.g. octrees, kd-trees) can be used, however, our system adds a unique challenge to these structures. Since we compute a proxy to feature-size during optimization, the structures need to be optimized for self-updates as the particle system refines and coarsens. Care must be taken to ensure that this overhead does not cancel out the gains of reduced lookup times.

An important avenue unexplored in this work is using parallel computations to improved of these systems. Particle systems have a natural parallelism associated to them, as only local information is required to make an informed update. The specific formulations could extend well to parallel computing environments such as GPUs. Additionally, any particle system that uses a k-cell hierarchy may poten-

tially achieve further parallelism across edges and surface patches, with minimal k-cell interaction.

While we do not have theoretical bounds on convergence rates, there are two observations that we have made about these systems. First, like many optimization problems, the initial state of the system plays a large role in the distance to convergence. Our seeding strategy is naive in that it attempts to use as few samples as possible early on, causing many particles to be generated during the early stages of each phase. A heuristic to bootstrap initial samplings could provide large speedups for models that require fine samplings. Second, we observe that the Lipschitz constraint tend to propagate outwards from corners. This means that particularly long edges with sharp corners tend to take longer to converge then both long edges without sharp features, as well as short edges with very sharp features.

5.2.2 Discussion and Limitations

For our test examples, we used default settings that were all derived in intuitive ways. The curvature parameter was in unit proportion to surface curvature. The Lipschitz parameter provided a fair compromise between triangle isotropy and adaptivity. Even the topological parameter was set to be half unit distance, to ensure sampling room on surfaces with restrictive boundaries. While a typical user might find themselves modifying these values for particular applications, care should be taken to ensure that these values are compatible with the geometry and topology of the model. This is especially true for the topological parameter. For example, the effect of using too large a value is shown in Figure 12.

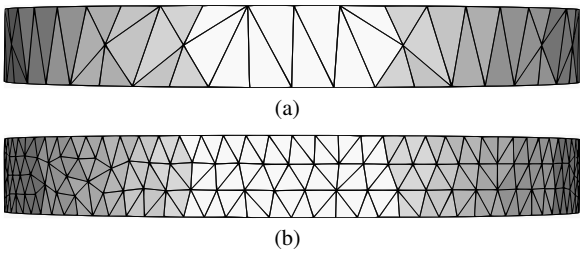

Fig. 12 Example of bad and good topological scaling parameters: (a) $s_\tau = 1.0$ (b) $s_\tau = 0.5$

The mesh in Figure 12(a), while still valid, has low quality elements on this edge region, where particles did not have enough freedom to distribute evenly. The mesh in Figure 12(b) instead uses the default topological scaling value, and allows sufficient freedom for particles to distribute and provide a more isotropic triangulation.

Due to the nature of discrete samplings in real spaces, there will always be situations in which a uniform packing is simply not possible. Most often this error will be distributed over much of the samples and will not be visually noticeable. However, occasionally, pockets of low energy show up that cannot be rectified using our rules, since no single particle has an energy that is too low or too high.

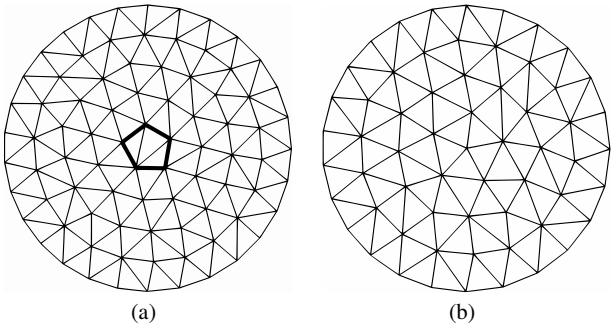

Fig. 13 (a) Configuration where pentagonal region remains open. (b) Configuration where pentagonal region is completed.

Figure 13 illustrates this problem on the endcap of a cylinder. The triangulation generated in Figure 13(a) has formed an energy pocket in the center, due to the geometry of the cap. The second triangulation in Figure 13(b) has no such issue, simply due to a differing particle count. This problem is not unique to PSYS, but rather a shortcoming of formulating a discrete sampling scheme as a continuous energy problem. The likelihood of this artifact occurring could be reduced by increasing the sensitivity of energy measures for splits and deletes, but the gains would be balanced by a decrease in stability of the system.

As described, our implementation makes two assumptions that may not apply to all models. The first is that curves and surfaces vary smoothly, with eccentric surface regions modeled as separate surface patches, rather than higher order surfaces. For a user who wishes to model with higher order surfaces, our seeding strategy of a single particle per edge is likely not sufficient. Since particles utilize only local information, it is conceivable that a sufficiently sharp spike in curvature could be missed. Remedies for this range from increasing the initial seed counts, to running a preprocess for finding local peaks and valleys in the domain. Our affine scaling step would also be effected by isolated peaks of high curvature.

The second assumption that our system makes is that 1-cell corners (sharp edges generated by two surfaces meeting) will also be protected by guarding 0-cells since these edges must be bounded. Again, it is conceivable that a complex patch might have safe 0-cell corners, but a sharp interior 1-cell corner due to some atypical curvature. In such a case, our implementation would need to apply the same principal of corner protection to the 1-cells, as it currently does to the 0-cells.

6 Conclusions & Future Work

The high quality results generated from our algorithm illustrate how well-suited such particle systems are for sampling parametric domains. Compared to the other methods we evaluated, our system was able to generate better quality triangle meshes with the added benefit of adaptively sized elements. Moreover, the sizing field we adapt to can be inferred directly from the point samples, removing the need for model preprocessing.

The success of this technique indicates that there are many unexplored avenues to take with respect to particle meshing. The approach in this paper is centered around generating quality isotropic surface meshes, which happen to be good inputs to a constrained 3D Delaunay solution. However, optimizing a particle system directly in 3D space from the start may allow for high-quality, isotropic tetrahedral meshes similar to other variational techniques [1]. An interesting direction would be to infer the tetrahedralization without requiring computing a 3D Delaunay triangulation.

Another avenue we believe could prove fruitful is the introduction of anisotropic kernels to the energy formulation. Doing so could provide an easy and powerful method for generating anisotropic surface and volume meshes. Coupled with adaptivity, these meshes could provide ideal inputs to simulations across many application areas.

Acknowledgements

This research was supported by NIH/NIGMS Center for Integrative Biomedical Computing, 2P41-RR0112553-12

References

1. Alliez, P., Cohen-Steiner, D., Yvinec, M., Desbrun, M.: Variational tetrahedral meshing. ACM Trans. Graph. **24**(3), 617–625 (2005)
2. Amenta, N., Bern, M.W.: Surface reconstruction by Voronoi filtering. Discrete & Computational Geometry **22**(4), 481–504 (1999)
3. Antani, L., Delage, C., Alliez, P.: Mesh sizing with additively weighted Voronoi diagrams. In: 16th IMR, pp. 335–346 (2007)
4. Boissonnat, J.D., Oudot, S.: Provably good sampling and meshing of surfaces. Graphical Models **67**(5), 405–451 (2005)
5. Chen, L.: Mesh smoothing schemes based on optimal Delaunay triangulations. In: 13th IMR, pp. 109–120 (2004)
6. Cheng, S.W., Dey, T.K.: Maintaining deforming surface meshes. In: 19th Symp. on Discrete Algorithms, pp. 112–121 (2008)
7. Cheng, S.W., Dey, T.K., Levine, J.A.: A practical Delaunay meshing algorithm for a large class of domains. In: 16th IMR, pp. 477–494 (2007)
8. Cheng, S.W., Dey, T.K., Ramos, E.A.: Delaunay refinement for piecewise smooth complexes. Discrete & Computational Geometry **43**(1), 121–166 (2010)
9. Cheng, S.W., Poon, S.H.: Three-dimensional Delaunay mesh generation. Discrete & Computational Geometry **36**(3), 419–456 (2006)
10. Chew, L.P.: Guaranteed-quality triangular meshes. Tech. Rep. TR-89-983, Comp. Science Dept., Cornell University (1989)
11. Dey, T.K., Levine, J.A.: Delaunay meshing of piecewise smooth complexes without expensive predicates. Algorithms **2**(4), 1327–1349 (2009)
12. Dey, T.K., Li, G., Ray, T.: Polygonal surface remeshing with Delaunay refinement. In: 14th IMR, pp. 343–361 (2005)
13. Dey, T.K., Sun, J.: Normal and feature approximations from noisy point clouds. In: Found. of Soft. Tech. and Theor. Comp. Sci., pp. 21–32 (2006)
14. Du, Q., Wang, D.: Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations. Intl. J. Numerical Methods in Engr. **56**(9), 1355–1373 (2003)
15. Haimes, R., Aftosmis, M.J.: On generating high quality watertight triangulations directly from CAD. In: Intl. Soc. for Grid Gen. (2002)
16. Haimes, R., Follen, G.J.: Computational analysis programming interface. In: 6th Intl. Conf. Num. Grid Gen. in Comp. Field Sim., pp. 663–672 (1998)
17. Hart, J.C., Bachta, E., Jarosz, W., Fleury, T.: Using particles to sample and control more complex implicit surfaces. In: SIGGRAPH '05 Courses, p. 269. New York, NY, USA (2005)
18. Jones, W.T.: Toward a global parameterization for quilted CAD entities. In: 42nd AIAA Aerospace Sciences Meeting and Exhibit (2004). AIAA Paper 2004-0611
19. Karkanis, T., Stewart, A.J.: Curvature-dependent triangulation of implicit surfaces. IEEE Computer Graphics and Applications **21**(2), 60–69 (2001)
20. Lohner, R.: Surface gridding from discrete data. In: 4th IMR, pp. 29–44 (1995)
21. Meyer, M.D.: Dynamic particles for adaptive sampling of implicit surfaces. Ph.D. thesis, Salt Lake City, UT, USA (2008)
22. Meyer, M.D., Kirby, R.M., Whitaker, R.T.: Topology, accuracy, and quality of isosurface meshes using dynamic particles. IEEE Trans. Vis. Comput. Graph. **13**(6), 1704–1711 (2007)
23. Meyer, M.D., Whitaker, R.T., Kirby, R.M., Ledergerber, C., Pfister, H.: Particle-based sampling and meshing of surfaces in multimaterial volumes. IEEE Trans. Vis. Comput. Graph. **14**(6), 1539–1546 (2008)
24. Rineau, L., Yvinec, M.: Meshing 3d domains bounded by piecewise smooth surfaces. In: 16th IMR, pp. 443–460 (2007)
25. Ruppert, J.: A Delaunay refinement algorithm for quality 2-dimensional mesh generation. J. Algorithms **18**(3), 548–585 (1995)
26. Scheidegger, C.E., Fleishman, S., Silva, C.T.: Triangulating point set surfaces with bounded error. In: Symp. on Geom. Proc., pp. 63–72 (2005)
27. Schreiner, J.M., Scheidegger, C.E., Fleishman, S., Silva, C.T.: Direct (re)meshing for efficient surface processing. Comput. Graph. Forum **25**(3), 527–536 (2006)
28. Shewchuk, J.R.: Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In: M.C. Lin, D. Manocha (eds.) Applied Computational Geometry: Towards Geometric Engineering, vol. 1148, pp. 203–222. Springer-Verlag (1996)
29. Shewchuk, J.R.: Tetrahedral mesh generation by Delaunay refinement. In: 14th Symp. on Comp. Geom., pp. 86–95 (1998)
30. Si, H., Gärtner, K.: Meshing piecewise linear complexes by constrained Delaunay tetrahedralizations. In: 14th IMR, pp. 147–163 (2005)
31. Tournois, J., Alliez, P., Devillers, O.: Interleaving Delaunay refinement and optimization for 2d triangle mesh generation. In: 16th IMR, pp. 83–101 (2007)
32. Tournois, J., Wormser, C., Alliez, P., Desbrun, M.: Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. ACM Trans. Graph. **28**(3) (2009)
33. Turk, G.: Re-tiling polygonal surfaces. In: SIGGRAPH '92, pp. 55–64 (1992)

34. Valette, S., Chassery, J.M., Prost, R.: Generic remeshing of 3d triangular meshes with metric-dependent discrete Voronoi diagrams. IEEE Trans. Vis. Comput. Graph. **14**(2), 369–381 (2008)
35. Witkin, A.P., Heckbert, P.S.: Using particles to sample and control implicit surfaces. In: SIGGRAPH '94, pp. 269–277 (1994)
36. Yamakawa, S., Shimada, K.: High quality anisotropic tetrahedral mesh generation via ellipsoidal bubble packing. In: 9th IMR, pp. 263–274 (2000)
37. Yan, D.M., Lévy, B., Liu, Y., Sun, F., Wang, W.: Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. Comput. Graph. Forum **28**(5), 1445–1454 (2009)