# Consensus Meshing

Ryan Schmidt, Patricio Simari

[a]*Autodesk Research*

**Abstract**

Consider an algorithm for generating a triangle mesh interpolating a fixed set of 3D point samples, where the generated triangle set varies depending on some underlying parameters. In this paper we treat such an algorithm as a means of sampling the space of possible interpolant meshes, and then define a more robust algorithm based on drawing multiple such samples from this process and averaging them. As mesh connectivity graphs cannot be trivially averaged, we compute triangle statistics and then attempt to find a set of compatible triangles which maximize agreement between the sample meshes while also forming a manifold mesh. Essentially, each sample mesh "votes" for triangles, and hence we call our result a *consensus mesh*. Finding the optimal consensus mesh is combinatorially intractable, so we present an efficient greedy algorithm. We apply this strategy to two mesh generation processes - ball pivoting and localized tangent-space Delaunay triangulations. We then demonstrate that consensus meshing enables a generic decomposition of the meshing problem which supports trivial parallelization.

## 1. Introduction

Assume we are given a set of points, possibly with normals, sampled from some 3D surface, which may be unknown. We consider the problem of covering this set of points with a triangle mesh. This mesh can either pass through the given points, or only geometrically approximate them under some metric. In many applications it is important to preserve the existing points. For example, we may wish to retile or repair an existing mesh while at the same time preserving important per-vertex attributes. The sampling may have important global properties - for example a Poisson sampling - that should not be modified. Or we may be working with point-cloud data acquired from a 3D scanner, and desire the maximum geometric fidelity without any artificial smoothing. These are instances where the mesh generation algorithm must be *interpolatory*.

Given the prevalence of this problem, a variety of interpolatory mesh generation algorithms have previously been developed [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. However, as opposed to, say, a Delaunay triangulation, many of these algorithms involve tuning parameters such as search radii or initial conditions such as seed triangles. As a result, by varying the parameters and initial conditions we can generate a range of potential meshes.

It is usually the case that, given outputs from a meshing algorithm over a variety of input parameters, a more desirable mesh could be created by combining portions of several outputs. Rather than trying to improve the meshing algorithm, we explore the approach of attempting to automatically extract a higher-quality composite from the set of given meshes.

Motivated by recent work in randomized mesh analysis [11, 12], we hypothesize that the statistics of meshes themselves can directly inform us about which triangles are ideal. We treat the result of a mesh generation algorithm as a sample from the space of possible meshes for a given point set. Given a population of meshes, we can compute statistics over the union set of candidate triangles. In particular, the more frequently the output meshes agree about the presence of a particular triangle, the more *stable* that triangle is under the given meshing strategy. A manifold mesh formed out of the set of triangles which maximizes total stability - the mesh that the samples agree on the most - is the *consensus mesh*.

Although the consensus mesh is readily defined, finding it is complicated by the requirement that the set of triangles be *compatible* in some sense; for example that they form a manifold, orientable surface. This is a complex global constraint that is difficult to satisfy if we are to also maximize other factors such as coverage (*i.e.* not leaving gaps). In fact, we show that finding the optimal consensus mesh in the connected, closed case is NP-complete (Appendix A).

In the following sections we introduce the consensus mesh concept and define an efficient greedy algorithm for finding an approximate solution which is man-

ifold and orientable. We then present several experiments using two example meshing strategies. First is the well-known Ball Pivoting Algorithm (BPA) [1], for which our consensus meshes automatically compensate for the fixed-ball-radius limitation of BPA, producing meshes with much more accurate preservation of features at varying scales. We then present a novel local mesh generation strategy based on the Discrete Exponential Map [13]. Our approach essentially computes a local Delaunay triangulation "in the manifold", rather than in the planar projections used by other local-meshing works. The consensus mesh automatically extracts a compatible global connectivity graph from these overlapping incompatible local meshes. This local-to-global capability of consensus meshing, combined with a simple constraint mechanism, enables trivially-parallelizable meshing of large datasets. We also show that our use of simple triangle frequency counts is justified, as attempts to replace or augment it with geometric measures only served to reduce the output quality.

As any objective judgement of mesh quality is necessarily application-dependent, we cannot claim that our consensus meshes are in some way "better" than those produced by state-of-the-art meshing schemes (and hence do not perform such comparisons). Consensus meshing also lacks many of the theoretical guarantees of recent algorithms. The purpose of consensus meshing is instead a novel capability for interpolatory meshing: *automatic compensation for parameter-dependent deficiencies in mesh generation algorithms*. With consensus meshing it is not necessary to manually explore a parameter range to find the "best" output. As we demonstrate, a high-quality result can instead be generated automatically based on a straightforward sampling of the parameter space. We posit that this is a highly useful capability in practice, particularly for real-world point sets which often lack the sampling criteria needed to ensure robustness in existing algorithms.

### *1.1. Related Work*

The problem of inferring a surface from a set of point samples is a heavily studied area. Broadly speaking, there are many general approaches to the problem, including computational geometry [14, 15], implicit surfaces [16], and moving least squares [17]. We refer the interested reader to a survey of the area [18], since an in-depth taxonomy of all approaches is beyond the scope of this work.

Of particular interest to our contribution are those methods which retain the input points, as given, to appear as vertices of the output triangle mesh. Various such *interpolatory* algorithms have been proposed. The PowerCrust [2] is a well-known example, although as it inserts additional vertices it is not strictly interpolating. Related algorithms based on 3D Voronoi/Delaunay structures, such as Tight Cocone [4], or those based on the Flow Complex [5], are generally considered parameter-free, although in practice some threshold parameters are needed to deal with numerical issues. However, these algorithms are highly dependent on sampling quality (see [19]). Improved algorithms such as Robust Cocone [6] tend to involve some sampling-dependent radius parameter. The well-known $\alpha$-shapes [20] again has a radius parameter that significantly affects the output mesh, as do algorithms built on $\alpha$-shapes [9] and those inspired by it, such as the Ball Pivoting Algorithm [1]. Region-growing approaches [21] tend to have similar parameters, as well as dependencies on initial conditions.

Another class of algorithms focuses on producing a globally-consistent mesh while using only local information. If sampling is locally uniform, various algorithms can be applied to directly generate a mesh [22, 23]. The work of Kil and Amenta incorporates an element of consensus-finding similar in spirit to our approach. Mehra *et al.* [24] apply a novel point cloud visibility algorithm to generate local reconstructions based on convex hulls in a dual space, which are then integrated into a global mesh. Algorithms based on local information are also easy to parallelize; for example a technique based on local Delaunay triangulation in planar projections [25] has been implemented on the GPU [26]. Even when such algorithms do not produce an optimal result, local operators can be used to improve the mesh quality [27].

We do not propose a new algorithm as an alternative to any of these, rather our contribution is to view these algorithms as processes that enable the *sampling* of the space of all possible interpolating meshes by varying their input parameter values. Hence, our contribution is largely *orthogonal* to the actual mesh generation algorithm.

The idea of finding a consensus surface from a set of sampled candidate solutions has been applied to meshing problems. Binary voxelizations of a mesh can be combined via majority vote to create a voxelization robust to spurious topological changes [28]. Alternately implicit surfaces can be fit to the generated surfaces, and then functionally combined via robust averaging [29]. Though effective, these approaches are not directly applicable to interpolatory meshing.

Consensus-finding has been applied to various other problems in computer graphics. For example, Golovinskiy and Funkhouser [11] sample the space of possi-

ble segmentations and integrate the information by tallying the frequency with which an edge appears in the sampled segmentation. Zheng *et al.* [30] sample curve skeletons from different poses of a given shape and find the shape's consensus skeleton, which should remain topologically invariant. Mitra *et al.* [31] sample the space of transformations that map parts of a surface onto itself and find clusters of such transformations to identify local symmetries. Anguelov *et al.* [32] perform a similar consensus of transformations (which is later refined) in order to recover an articulated model from instances presented as independent scans of a model in different poses.

We note that our problem differs from these latter cases of consensus finding in that, while their selection space is continuous, and thus amenable to density-based clustering approaches, ours is discrete. In the case of consensus skeletons, the limited size, topological simplicity, and low dimensionality of curve skeletons allows for a correspondence-driven approach. Unfortunately, none of these conditions apply to our problem. In the case of consensus segmentations, the segmentation edges are discrete entities that can be tallied over a single reference mesh, thus enabling a graph cut approach to consensus finding. However, in our case, no such reference graph exists, making a graph-cut approach not directly applicable. Furthermore, even if such a reference could be found, a graph cut approach would likely not scale to the size of datasets we aim to handle.

Scalability of interpolating meshing algorithms is a challenge. An approach explored in various works is to apply a spatial decomposition such as an octree, mesh each cell independently, and then stitch the local meshes. An early work in the Cocone family [3] proposed such an approach, at the cost of theoretical guarantees. More recently these guarantees have been restored [10], and a similar strategy based on the medial scaffold has also been presented [8]. We also scale up via decomposition followed by stitching, however we do not depend on any geometric consistency to ensure that the disjoint meshes automatically "fit". Instead we simply apply the same consensus-finding strategy in the overlap regions. Assuming the underlying algorithm supports it, we can even use a different generation strategy for the stitching meshes than we did for the per-cell meshes. Hence, consensus meshing effectively acts as a "black-box" approach to parallelizing mesh generation.

## 2. Triangle Statistics

Consider a set of 3D point-samples $\mathcal{V} = \{p_1, p_2, \ldots, p_n\}$. We assume that no points are coincident up to floating-point error, *i.e.* for all $i, j$, $\|p_i - p_j\| > \delta$, where $\delta$ is some function of machine precision. A triangle can be defined by a triplet of indices $t_{abc} = (a, b, c)$ or equivalently the triplet of points $t_{abc} = (p_a, p_b, p_c)$. We then define a *mesh* as a list of triangles $\mathcal{M} = \{t_1, t_2, \ldots, t_m\}$. Note that in the context of this paper, $\mathcal{V}$ is fixed and hence when discussing different $\mathcal{M}$'s only the list of vertex triplets varies.

Although any set of triangles can be considered a mesh, for most applications it is necessary that the mesh be *manifold*, possibly with boundary. A mesh is manifold if each vertex has a neighbourhood isomorphic to a disc or half-disc, which implies that:

- each edge has one or two incident triangles
- each vertex has a simply-connected neighbourhood with either zero or two boundary edges

For most applications we also desire a mesh that is *orientable*, meaning that a consistent clockwise or counterclockwise orientation can be assigned to each triangle. In terms of triangle indices, orientability implies that across a shared edge $(a, b)$ the two connected triangles must have ordered indices $(a, b, c)$ and $(b, a, d)$.

Now consider some process $\mathbf{P}(\Theta)$ which generates a mesh based on input parameters $\Theta$. As we vary $\Theta$ a series of meshes $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_N$ will be output. We assume that there will be some "noise" in this process, in the form of spurious or undesirable triangles. Conceptually, we attempt to reduce the presence of these undesirable triangles by averaging over all $\mathcal{M}_i$. Unfortunately $\mathcal{M}_i$ is a set of vertex index tuples, so the mathematical notion of the average or *mean* is not well-defined. $\mathcal{M}_i$ can be interpreted as a connectivity matrix, and we can average these matrices, but the result would be fractional connectivities which have no clear interpretation.

It would seem that the only basic statistic that we can extract from a sequence of meshes is the presence or absence of a given triangle. Let

$$\delta(i, t_{abc}) = \begin{cases} 1 & \text{if } t_{abc} \in \mathcal{M}_i, \\ 0 & otherwise. \end{cases} \tag{1}$$

We consider any orientation-preserving permutation of $t_{abc}$ to be equivalent, so $t_{abc} = t_{cab} = t_{bca}$. We can now define the frequency of a triangle $t_{abc}$ as

$$f(t_{abc}) = \frac{\sum_{i=1}^{N} \delta(i, t_{abc})}{K} \tag{2}$$

where $K$ is an arbitrary normalizing constant. For example, $K = N$ gives us the probability that a triangle appears in a randomly selected mesh. For our pur-

poses we need only a relative ordering, so $K = 1$ suffices. In some cases we will also use the unit-range $\bar{f}(t_{abc}) = f(t_{abc}) / \max(f(t_{abc}))$.

## 3. Threshold Meshing

We posit that the more frequently a triangle occurs in sample meshes, the more *stable* the triangle is under the meshing process **P**. We would like to maximize overall stability, as in some sense, stable triangles most precisely represent the *intent* of **P**.

Given some frequency threshold $r$, we can define the *threshold mesh* $\mathcal{M}_r = \{t_{abc} | f(t_{abc}) > r\}$. Setting the threshold to some small $r$ will filter outlier triangles, however the resulting mesh will, in general, not be manifold (Figure 1a). Increasing $r$ will improve the situation, but in general setting $r$ high enough that all non-manifold triangles are filtered out will result in some desirable triangles also being filtered, creating holes (Figure 1b-c).

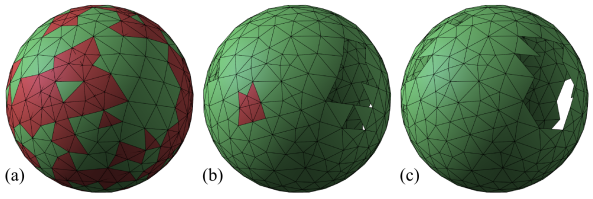



Figure 1: Triangles connected to a non-manifold edge (a) are drawn in red. Increasing the frequency cutoff threshold (to right) results in (b) holes in the mesh appearing before (c) all non-manifold edges are filtered out.

Conceptually it would be desirable if as $N \to \infty$, the threshold mesh $\mathcal{M}_r$ converged to a single manifold topology for some value of $r$. This would then clearly be the consensus mesh under **P**. However, in our experience most **P** have unstable vertex configurations in which edges can arbitrarily flip, resulting in conflicting faces with near-equal frequency.

## 4. Consensus Meshing

Our general problem is that we have a "triangle soup" over a fixed vertex set and we would like to extract a manifold, orientable mesh which covers the set of input vertices $\mathcal{V}$. To further constrain this extraction we would like to also maximize the sum of triangle frequencies. Hence, we want to find:

$$\mathcal{M}_C = \arg\max_{\mathcal{M} \in \Gamma} \sum_{t_{abc} \in \mathcal{M}} f(t_{abc}) \tag{3}$$

where $\Gamma$ is the space of manifold, orientable triangle meshes over $\mathcal{V}$. We call $\mathcal{M}_C$ the *consensus mesh*, as it maximizes the sum of "votes" for individual triangles under the constraint that the triangles agree on the mesh topology.

Finding $\mathcal{M}_C$ is challenging because whether or not a given triangle can be included in $\mathcal{M}_C$ depends on the triangles in its local neighbourhood, which in turn depend on their local neighbourhoods, and so on. This problem structure implies that a global combinatorial search would be necessary. In Appendix A we prove that the problem of finding a connected and closed $\mathcal{M}_C$ is, in fact, NP-completee. Hence, in this section we will explore greedy algorithms for finding an *approximate* consensus mesh. (Note that we will abuse our own terminology somewhat, and refer to any attempt to find the solution to Equation 3 as a consensus mesh $\mathcal{M}_C$).

### 4.1. Incremental Extraction

A trivial greedy mesh extraction algorithm could be defined by sorting the potential triangles in order of decreasing frequency $f$ into a list $L_f$, and then iterating through the list and appending any compatible triangles to $\mathcal{M}$. Going forward we will define "compatible" as manifold and consistently orientable, which rules out a variety of triangle configurations. In particular, as non-manifold boundary vertices are disallowed, this simple linear pass over $L_f$ will result in many triangles being discarded which would be compatible if added in a different order.

To maximize the number of compatible triangles, we note that a potential triangle is incompatible with $\mathcal{M}$ if:

- any edge is connected to an interior edge of $\mathcal{M}$
- any vertex is connected to an interior vertex of $\mathcal{M}$
- any edge is connected to an edge of $\mathcal{M}$ with opposite orientation

These triangles are discarded from $L_f$ whenever they are encountered. Potential triangles with vertices connected to existing boundary vertices (Figure 2b) are simply skipped during the current iteration. Each time a compatible triangle is found, we remove it from $L_f$ and then *restart* the search from the beginning of the updated potential set $L_f$.

The result of running this algorithm is a mesh comprised of a set of small "islands" (Figure 2a). This occurs because we allow multiple isolated triangles to be added, but enforce the constraint that all vertices remain manifold. So, the algorithm is limited to adding isolated triangles or appending to existing boundary edges. The

"connector" triangles necessary to bridge gaps would temporarily create non-manifold vertices (Figure 2b), which are not allowed.

We can modify this algorithm by only permitting a single isolated triangle to be added, and requiring all further triangles to be connected to an existing edge. From this *seed triangle*, the mesh will grow outwards to cover the rest of the surface. Obviously only the component connected to the seed triangle will be meshed using this approach. More problematic is that it can leave cracks in the mesh, as shown in Figure 3b. These cracks occur when a compatible triangle on the boundary of the growing mesh is not in the potential set, because it was not generated in any sample mesh.

A somewhat more interesting issue with limiting incremental extraction to a single seed triangle is that the final result is biased by the choice of seed. Generally we use the triangle with the highest frequency as the seed. If we add another sample mesh, the slight change in statistics could result in a different seed triangle, which in turn could produce an entirely different mesh.

### 4.2. Pair Extraction

We have seen that sequentially appending compatible triangles may result in cracks forming where necessary triangles are missing from the potential set. In their greedy region-growing algorithm, Cohen-Steiner and Da [7] simultaneously add *pairs* of triangles to cross such gaps, as shown in Figure 2c. We take a similar approach, although we have many potential pair triangles and hence must have some selection criteria.

We augment our greedy algorithm as follows. If the next-best triangle $t_k = (a, b, c)$ would result in a non-manifold vertex $c$, we search the potential set for the next-best compatible triangle $t_j$ which is connected to either edge $(a, c)$ or edge $(b, c)$. Although the potential set will usually contain *some* triangle $t_j$ that can be

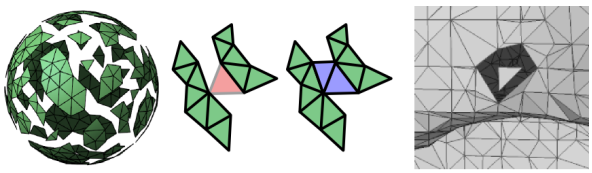



Figure 2: Incrementally growing from the boundary with multiple seed triangles results in (a) mesh "islands", as we disallow (b) the non-manifold vertices that would be created by gap-crossing triangles. A manifold gap-crossing can only be created by adding (c) pairs of triangles simultaneously. Even with gap-crossing, if multiple seed triangles are permitted then (d) isolated triangles can occur when the pair necessary to join the island is not in the potential set.

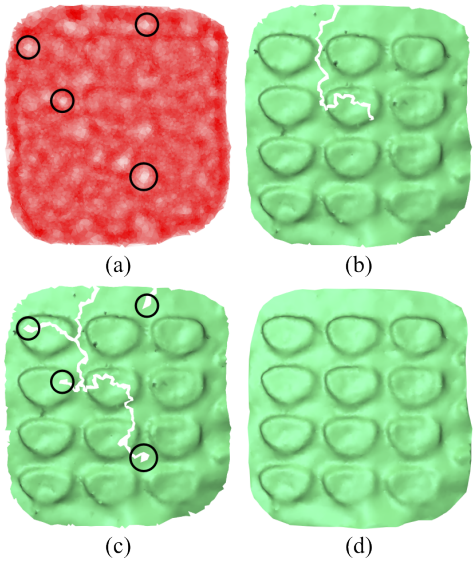



Figure 3: Dial-pad from scanned phone in Figure 4. Figure (a) shows an alpha-blended rendering of all potential triangles, with $\alpha$ set to the normalized frequency count of the triangle. Incremental extraction from a single seed triangle (b) covers the mesh but leaves cracks when a necessary triangle is missing from the potential set. Increasing the minimum $f$-threshold (c) results in additional cracks, which tend to start at locations where there is higher uncertainty (circles in a,c). Adding pair extraction (d) to cross the gaps results in full coverage.

paired with $t_k$, if the score for $t_j$ is very low then adding this triangle will often result in many desirable triangles becoming incompatible. To avoid this we also find the next compatible triangle $t_{k+1}$ after $t_k$. We then check if $(f(t_k) + f(t_j))/2 > f(t_{k+1})$. If so, we add $t_k$ and $t_j$, otherwise we add $t_{k+1}$.

To reduce the cost of this additional linear search we only consider potential $t_j$ whose score is within some threshold of $f(t_k)$. We use a threshold of 0.9, but note that (empirically) the output is not very sensitive to this parameter. Generally it serves only to reduce the search time - the score comparison is far more likely to be the deciding factor. Too large a threshold will simply increase computation time, while too small will mean that some pairs are skipped. However, such a high-scoring pair is necessarily composed of high-scoring triangles, which will eventually be added individually unless they become incompatible.

### 4.3. Constrained Extraction

One particularly useful capability of our approach is that it can easily incorporate constraints in the form of specific triangles which should exist in the consensus mesh. We simply remove these initial triangles from $L_f$ and append them directly to the output. Note that

while it is not strictly necessary that constraint triangles exist in $L_f$, it is unlikely that they will be compatible with the rest of the mesh if they do not. Constrained extraction will be useful in Section 5.4 where we will use constraint triangles to generate connective strips of mesh to join two existing meshes.

### *4.4. Efficiency Considerations*

The greedy algorithm described above is still computationally expensive. To append a triangle we must linearly search from the beginning of the current $L_f$. If we find a triangle which can potentially be paired, we must then linearly search from that triangle onwards to find a suitable pair triangle, and then perform a second search for the next-best non-pair triangle. While these two searches can be combined, and we can truncate the search at some score threshold, a significant number of triangles may need to be tested.

Ideally our searches would always terminate near the beginning of $L_f$. Discarding incompatible triangles from $L_f$ as they are encountered ensures that they are only tested once. However, in most cases a large number of triangles will remain in the list which must be tested and then skipped, as they are not yet incompatible, but also cannot be connected to the expanding mesh. These compatibility tests involve interrogating the current mesh topology and hence do have some cost. However, note that for most triangles in $L_f$, the addition of a single triangle $t_k$ will have no effect on the outcome of the compatibility tests.

We can take advantage of this coherence by storing a timestamp for each vertex and potential triangle. Our "time" is an iteration counter, and when we insert a triangle we set the timestamp on each of its vertices to the current time. Then during the iteration through $L_f$, we only consider a triangle for insertion if its timestamp is less than the maximum timestamp of its vertices. If this is the case, we test the triangle, and if it is not inserted or discarded, we set the timestamp to this maximum.

Exploiting coherence via timestamping significantly reduces runtimes for our algorithm by several orders of magnitude. As an illustrative example, consider the result in Figure 4, where we have 44,023 points and 487,348 triangles in the potential set. Without timestamping, extracting the 85,406 triangles in the consensus mesh takes over 3 hours. With timestamping the same mesh is extracted in 34 seconds. (We emphasize that these numbers are provided only to demonstrate the relative speed-up; no attempt has been made to optimize the efficiency of the manifold and orientation checks which dominate the computation.)

## 5. Experiments

In this section we compute consensus meshes for several different types of meshing problem, using two different meshing algorithms - Ball Pivoting [1] and a local Delaunay triangulation approach based on the Discrete Exponential Map [13]. We then demonstrate that constrained consensus meshing can be used to decompose the meshing problem in a manner amenable to parallel implementations.

### *5.1. Ball Pivoting (BPA)*

The Ball Pivoting Algorithm (BPA) [1] is a well-known technique for reconstructing a triangle mesh given a set of 3D points. The algorithm is straightforward: first a *seed triangle* is selected, then for each edge of the seed a ball of radius $r$ is pivoted around the edge until another point is hit. This generates a new triangle, and new edges, which are processed until no active edges remain.

The fixed ball radius $r$ is the main limitation of the BPA. If too large, the resulting mesh may differ significantly from the surface that generated the point set. If too small, holes will be introduced. Clearly this poses problems for point sets where the spatial density of points varies. Furthermore, in general only a subset of the input points are interpolated.

We take BPA to be our process $\mathbf{P}$ and $r$ to be our parameter set $\Theta$. Using the implementation available in the software MeshLab [33], we generate a set of meshes by sampling $r$ at regular intervals. An example on a surface with variable point density is shown in Figure 4. Note that at no value of $r$ is the entire surface covered.

Applying our extraction algorithm from the previous section, we see in Figure 4d that the consensus mesh is of significantly higher quality than any of the input meshes. First, save for some small holes the entire surface is covered in a way not exhibited in any of the input surfaces. The consensus mesh interpolates 99.6% of the 44,023 input vertices, versus 95.1% for the best individual BPA mesh. Incidentally, our BPA consensus mesh also has higher triangle quality than the original mesh generated by the photo-reconstruction software, which presumably uses an algorithm more robust than BPA.

In Figure 5 we explore convergence properties of the BPA consensus mesh under increasing sample count. We observe that with uniform parameter-space sampling, doubling the sample rate from 50 to 100 does result in a higher mesh quality (fewer holes). However, re-distributing the 100 sample points such that smaller ball radii are sampled more densely results in a significant improvement.

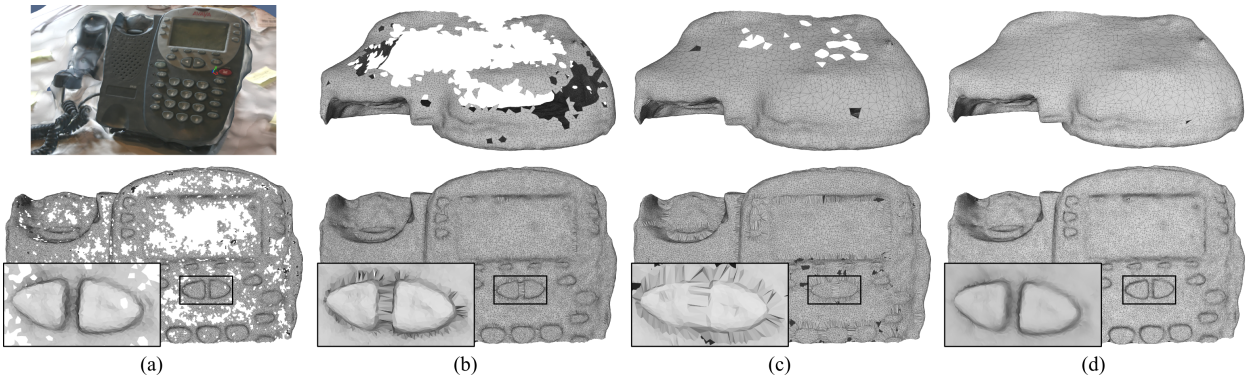



Figure 4: We used a 3D-model-from-photographs web service to scan a telephone (a,top), then discarded the mesh and generated a consensus meshing using the Ball Pivoting Algorithm (BPA). Due to variable point sampling density, the BPA meshes (a-c, increasing ball radius) exhibit both holes and (insets) undesirable topological changes. The consensus mesh (d) extracted from 100 sample BPA meshes still contains some small holes, but is of higher quality than that achievable with BPA alone.

The BPA paper [1] does propose a method to work around the fixed-ball-size limitation, namely running the algorithm with an initial ball size, and then iteratively increasing the ball size while appending to the existing triangulation. The robustness of this approach has not been evaluated in the literature. In basic experiments we found that it can be effective, but depends on the first iteration having a ball size large enough to result in coverage over most of the surface, which in turn means that small features will be lost. If the ball size begins at the smallest feature size, many small disconnected components will be produced, similar to Figure 4a. Later iterations will have difficulty merging these components, creating many non-manifold regions. Essentially, this approach cannot recover from any low-quality triangles chosen in previous iterations. By deferring decisions until all the information (*i.e.* meshes) is available, our consensus approach can avoid many of these errors.

### 5.2. Tangent-Space Delaunay Triangulation

A desirable property for a meshing algorithm is the ability to produce a globally-consistent mesh while only using local information. This problem has been previously explored [25, 22], and even if the result is not optimal, algorithms exist for efficiently improving quality via local operations [27]. In this section we consider reconstruction from local triangulations in the context of Consensus Meshing.

The Discrete Exponential Map (DEM) [13] algorithm projects points in a local neighbourhood around a *seed point* $\mathbf{p}$ of a point set into the *tangent space* $T_{\mathbf{p}}$. The tangent space is effectively a local planar parameterization, and can be put to many uses. Schmidt and Singh [34] demonstrated that local constrained Delaunay triangulations within these spaces can be used to implement mesh editing operations. However, we can consider a local Delaunay triangulation computed in a DEM tangent space as simply a meshing process $\mathbf{P}$ which does not cover the entire surface. Our parameter $\Theta$ is then the seed point $\mathbf{p}$, and by varing it we can generate sample meshes and accumulate triangle statistics.

Note that the DEM is not a planar projection - it wraps around the surface with distortion proportional to the surface curvature [13], and has zero distortion on developable surfaces. Hence, meshes generated in the DEM

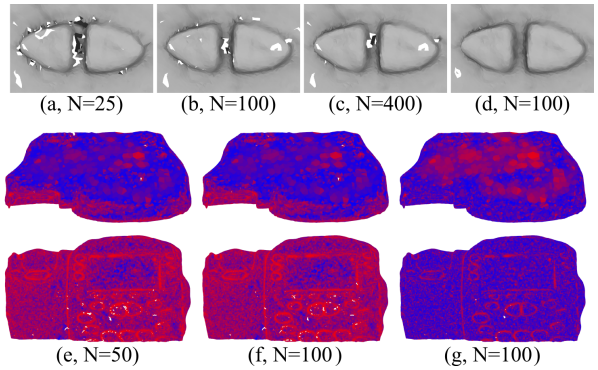



Figure 5: In (a-c) we sample a fixed BPA ball-radius parameter range at regular intervals, with an increasing number of samples $N$. As expected, the result improves, but slowly. In (d) we sample nonuniformly, with more samples concentrated at smaller ball radii, and get a better result with N=100 than uniform sampling with N=400. In (e-g) we map the consensus score $\bar{f}$ to a color range from red (0) to blue (1). We observe that with uniform sampling the smaller triangles tend to have lower $\bar{f}$, while (g) concentrating samples at smaller ball radii shifts the distribution. Note that while the $\bar{f}$ scores cannot be compared between the different cases, clearly higher $\bar{f}$ is preferable. The distribution of $\bar{f}$ relative to triangle size in the output could perhaps be used to drive importance sampling strategies.

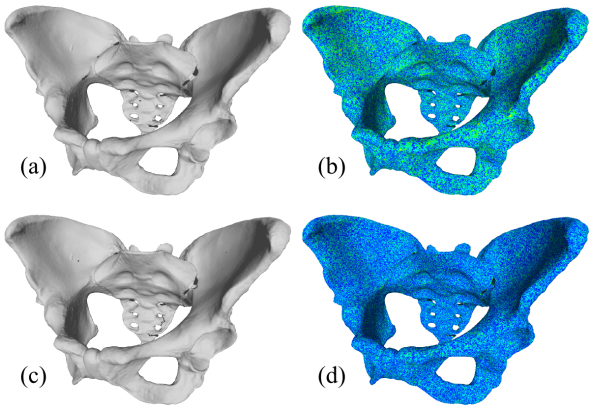



Figure 6: The vertices of (a,b) an initial mesh are re-tiled by (c,d) a consensus mesh, significantly improving triangle quality (blue=1 is highest quality, green is lower quality). The aggregate triangle quality statistics of (b) the initial mesh are avg=0.78 median=0.82 stddev=0.16, while for (d) the consensus mesh avg=0.84 median=0.87 stddev=0.12.

tangent space are much closer to the principle of a Delaunay triangulation "in the manifold" than prior works based on tangent-plane projection.

In the examples that follow, we vary $\mathbf{p}$ by iterating through the input point set, which ensures that every point in the input point set is covered by some generated mesh (a form of stratified sampling). Alternatives such as randomly generating points and projecting onto the surface via MLS [17] could also be used. We use the more recent robust-DEM formulation [35] and the TRIANGLE software [36] to generate the 2D Delaunay triangulations. Although the DEM usually involves a geodesic radius, which could be varied as well, we ensure reasonable coverage instead by simply running the DEM propagation until a fixed number of neighbours are parameterized. We generally set this value to 200, which is excessive for near-regular meshes but necessary on meshes with more variable sampling density.

One simple application of consensus meshing using local Delaunay triangulations is to improve the quality of an existing tessellation. An example is shown in Figure 6, where the initial mesh was generated by 3D scanning software. Although no vertex positions are modified, the consensus mesh has significantly improved triangle quality statistics, clearly inheriting some of the properties of the underlying local 2D Delaunay triangulations. Note that we measure triangle quality as the deviation from identity of the linear transformation matrix taking the triangle to an equilateral triangle [33].

Figure 7 demonstrates another application, in which a Poisson resampling is computed for an existing mesh (again using MeshLab [33]) and then a consensus mesh computed based on local tangent-space meshes. Such near-regular reduced resolution meshes are particularly useful in multiresolution algorithms.

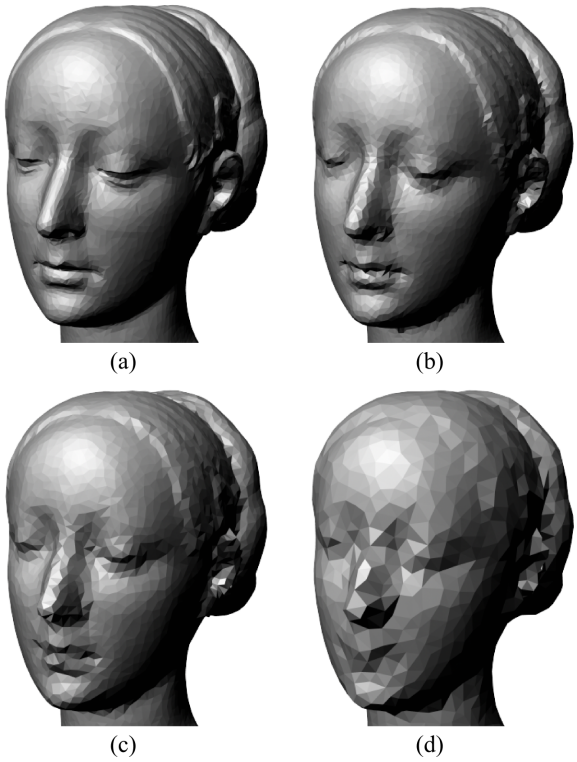



Figure 7: Consensus meshes for Poisson resamplings of (a) an initial mesh using approximately (b) 26k, (c) 15k, and (d) 5k sample points. The resamplings are highly regular, and hence so are the resulting meshes (rounded to two digits, the triangle quality statistics are the same in each case: avg=0.89 median=0.91 stddev=0.09).

### *5.3. Comparison with Geometric Measures*

In this paper we have proposed to use simple frequency counts as our metric for selecting triangles. If our goal is high-quality meshes, then this approach appears to ignore the clearly salient and easily computable per-triangle quality measures.

We first consider ignoring frequency counts altogether and using aspect ratio to score triangles. As can be seen in Figure 8, this significantly reduces the mesh quality. Many more holes are present. In addition, this strategy may even result in a mesh with poorer average aspect ratio, because the greedy selection of a single triangle with better aspect ratio can easily rule out a set of mutually compatible triangles with slightly lower average aspect ratios. The algorithm is then forced to select lower-quality triangles to fill in the gaps.

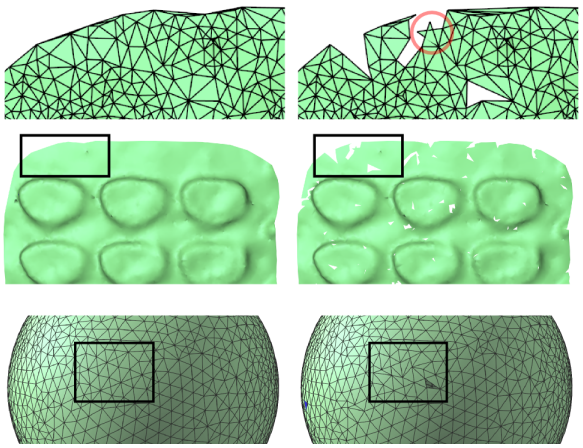


Figure 8: Extraction with triangles scored by (left) frequency count and (right) aspect ratio. Discarding frequency information results in many poor decisions by the greedy algorithm, including geometric clashes (circled). Despite favoring aspect ratio, in the top example the average aspect ratio is lower (0.863 vs 0.875 for the left) and deviation is higher (0.133 vs 0.109 for the left). This is clearly visible in the bottom example, where although coverage is similar, greedily selecting triangles with good aspect ratio also results in more triangles with poorer aspect ratio.

We also attempted to combine the aspect ratio measure with the consensus measure. This resulted in meshes with slightly higher quality statistics (a fraction of a percent) but introduced more local holes, reducing overall coverage of the point set. In addition, most improvements were in the form of isolated edge flips which could have been performed as a post-process.

This result is easily explained. The triangle frequency counts are not taken from a random triangle soup, but rather from meshes with largely consistent local neighbourhoods. Hence, the existence of a triangle in a mesh also implies the existence of compatible neighbours. Triangles that occur more frequently are by definition triangles that are likely to "fit" in a composite mesh. Furthermore, the underlying mesh generation algorithm presumably is already selecting high-quality triangles for the given point set. As a result, triangle frequency counts implicitly encode both neighbour information and triangle quality. We conjecture that given sufficient sampling, no quality measure will significantly improve on raw frequency counts. Empirically we have yet to observe any contradictions to this statement.

As some additional evidence of the power of raw frequency counts, we observe that with aspect ratio scoring, many overlapping triangles are produced (see Figure 8). These could be detected and avoided (though not robustly) via geometric clash detection. However, similar cases are rare when using the frequency count scoring. This makes sense as the underlying algorithms will prefer compatible local configurations.

### 5.4. Parallel Reconstruction

We have shown that consensus meshing can be used to integrate a set of disjoint mesh patches into a composite surface. We can further decompose the problem by computing consensus meshes for subsets of the input point set, and then using constrained extraction (Section 4.3) to stitch these sub-meshes together. We employ a basic strategy of splitting the point set into a grid of axis-aligned boxes, computing meshes for each grid cell, and then incrementally appending each grid cell mesh to the output mesh. In addition to improving runtime by reducing the overall computational complexity, the consensus mesh for each grid cell can be trivially computed in parallel. Hence, using this strategy we can parallelize any point-set meshing algorithm.

To stitch the mesh for a cell $c_i$ into the composite mesh $\mathcal{M}$, we first find the cell faces shared with neighbouring cells $c_j$ that are already appended to the output mesh. For each of these we compute an overlap bounding-box $B_{ij}$ centered at the shared face, with width set to $5-10\%$ of the cell dimensions. All triangles of $\mathcal{M}$ contained in $\bigcup B_{ij}$ are discarded. Next we define $B'_{ij}$ by expanding $B_{ij}$ by a factor of two outwards from the cell face. We define a new point set $\mathcal{V}' = \left\{v \in \mathcal{V} | v \in \bigcup B'_{ij}\right\}$, and compute the consensus mesh, adding all triangles from $\mathcal{M}$ contained within $\bigcup B'_{ij}$ as constraints (see Figure 9).

The result of this process is a thin band of mesh surrounding cell $c_i$ comprised of two types of triangles: the constraint triangles which already exist in $\mathcal{M}$, and the stitching triangles which do not. We can simply append these latter triangles to $\mathcal{M}$ to fill in the stitching region.

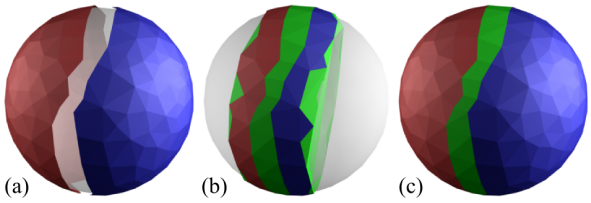



Figure 9: Given (a) disjoint regions of mesh covering the same point set, we (b) compute a consensus mesh of points in an overlap region, with constraint triangles included from the two initial meshes. We then (c) extract a transition mesh simply by discarding the constraint triangles.

A larger example is shown in Figure 10, where raw output from a handheld 3D scanner has been denoised

using Locally-Optimal Projection [37] and then meshed by slicing into 8 segments along the longest axis of the model bounding box. This scanning technique results in highly irregular point distribution, and even after applying LOP significant noise is still present near the sharper creases. Direct meshing of the point set does have the advantage of preserving fine-scale features such as the faintly visible creases on the rear face of the bone. These details may be lost by meshing strategies which only approximate the original point set.

Additional large-dataset examples are shown in Figure 11, with some statistics on meshing with different parameter settings in Table 1. Note that although small gaps and islands of the form seen in Figure 2d exist, coverage is generally high. For example, the Thai Statue model has 5M points and we cover all but 59 of them. Output quality slighly suffers as the grid cell density increases, although this can be counteracted by larger transition regions.

We note that the computation times in Table 1 are comparable to recent decomposition-based interpolatory meshing strategies [10]. Our current code is in no way optimized, and computation times are also limited by the 8-core, 12GB RAM machine used for testing. Again for the Thai Statue model, with additional compute power we could mesh all 221 cells in parallel, in the time of the single largest cell. Similarly, our currently-sequential reassembly algorithm can be parallelized by permuting the order in which we append grid cells.

## 6. Limitations

The main limitation of the approach we present is of course that it is entirely dependent on the underlying meshing algorithm. If a necessary triangle is not in the sample set, it will not be in the output mesh. As with any other meshing strategy, post-processing can be applied to repair defects.

*Sampling.* The consensus mesh is also dependent on the sampling of the mesh generation parameter space. Any under- or un-sampled features will be missed. We have not explored sampling strategies in any particular depth. For BPA we used regular parameter-space sampling strategies (linear and nonlinear) and for the tangent-space Delaunay approach we used opportunistic sampling (samples at vertices). We do emphasize that we did not need to carefully tune our sampling strategies. Save for Figure 5 we did not vary the BPA sampling strategy at all (regular sampling between the range of minimum to maximum point-set-neighbourhood edge lengths). For the local Delaunay approach, we simply meshed fixed-size neighbourhoods around each sample point. We note that neither of these strategies is random. Conceivably random sampling may result in faster convergence, this remains a question for future work.

*Topology.* Another limitation is that, at the moment, we can offer little in the way of theoretical guarantees about the consensus mesh. For example, given adequate surface sampling the Cocone family of algorithms [4, 3, 10] are certain to generate a mesh topologically equivalent to the surface the samples came from, and can even provide geometric accuracy guarantees. If this property were to hold for all sample meshes, our region-growing strategy should not introduce any new topological features. However, holes caused by incompatible triangles could cause existing topological features to remain uncovered. Even if some input meshes do have topological errors, it does seem plausible that with sufficient sampling the consensus mesh could avoid selecting such erroneous triangles. We do observe this in practice with the Ball-Pivoting results, however a formal validation remains future work.

*Watertightness.* Similarly, many meshing algorithms focus on watertightness as a desirable property [4]. Again, if the input meshes are watertight, presumably a watertight consensus mesh can be found, as in the worst case one could simply take all the triangles from a single watertight input mesh. However our current strategy makes no attempt to enforce watertightness.

*Boundaries.* Boundaries are another common issue [9]. Clearly if the input meshes have boundaries, they can be reproduced in the consensus mesh. However we make no guarantees about the topological consistency of these boundaries relative to the boundaries of the sampled surface.

We do note that although we have not explored mathematical guarantees, consensus meshes based on both Ball-Pivoting and Tangent-Space Delaunay meshing both do quite well at handling boundaries and topological similarity. Most of the point-sets we tested were collected from 3D scanners, and hence both contained boundaries and would not meet the sampling criteria necessary for the theoretical guarantees of known algorithms [10].

## 7. Conclusions and Future Work

We have presented consensus meshing, an approach to finding the conceptual "average" of a set of meshes

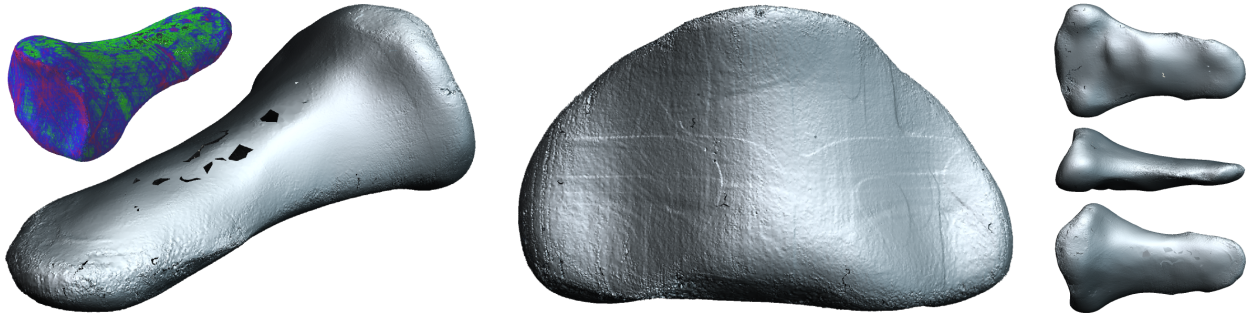


Figure 10: Direct meshing of point cloud data acquired from a handheld scanner and then denoised as a post-process. This bone scan has 656k points, the mesh has 1.3m triangles. Note preservation of fine-scale features. This point set is far from uniformly sampled, the leftmost color rendering indicates average neighbour distance, with the values clamped to $(\mu - 2\sigma, \mu + 2\sigma)$ and then linearly mapped to color range $(red = \mu - 2\sigma, blue = \mu, green = \mu + 2\sigma)$, and in this case $\sigma/\mu = 0.28$.

generated by sampling a mesh generation algorithm at different parameter values. We proved that extracting an optimal closed and connected consensus mesh is NP-complete, and then provided a greedy algorithm that can efficiently extract a manifold, orientable mesh. In addition to generating high quality meshes, consensus meshing can be used to transparently parallelize existing meshing algorithms.

We emphasize that this is not a new meshing algorithm, but rather a strategy for improving the output of existing meshing algorithms. We do not claim that the consensus meshes built from Ball-Pivoting or our Tangent-Space Delaunay approach are "better" than a state-of-the-art interpolatory mesh generation algorithm. Objectively defining mesh quality is too difficult. The purpose of consensus meshing is rather to *compensate for parameter-dependent deficiencies in the generation algorithm*. In this respect we do claim some success–rather than have to manually tune parameters, we have demonstrated that we can simply sample a parameter range and automatically combine the results.

The consensus mesh can inherit some local properties of the original algorithm, at the cost of some global guaratees. More useful is that by combining triangles from multiple parameter samples we can observe properties not achievable with the input mesh, such as we did with the composite Ball-Pivoting results in Section 5.1. Consensus meshing also improves practical robustness issues, such as algorithms failing or crashing for individual sample meshes. We regularly encountered both these issues, however in most cases nearby sample meshes can "fill in the gaps" and hence the failures have minimal effect on the final result.

We have in some sense presented the simplest possible approach to consensus meshing - each triangle is given one unit vote, and we greedily extract triangles in-order. We showed in Section 5.3 that integrating geometric measures into the triangle score may be problematic. We did limit ourselves to approaches that were fixed during the extraction stage - perhaps better results could be found if the scores of potential triangles adapted to those already selected (for example using Delaunay-like criteria). Similarly, more sophisticated schemes for extraction are likely to further improve results. For example, many of the remaining holes are due to specific triangles conflicting with others which could have filled the hole. Discarding a region around the hole and then applying the constrained re-computation of Section 5.4 could repair these holes.

The timestamping scheme presented in Section 4.4 essentially provides a way to track the active edges on the expanding boundary of the mesh. Spatial queries could be incorporated to further restrict the search to triangles close to this advancing front.

Finally, we have focused on statistics of triangles but another possibility would be to consider statistics of edges. We have had some success applying our approach to statistics of edges, although preserving manifold and orientable output is more difficult. A more intriguing possibility, though, is that by focusing on edges it may be possible to extract consensus quad-meshes, even if the input meshes are triangles.

## References


[1] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, G. Taubin, The ball-pivoting algorithm for surface reconstruction, IEEE Trans. Vis. Comp. Graph. 5 (1999) 349–359.

[2] N. Amenta, S. Choi, R. K. Kolluri, The power crust, in: Proc. Symp. Solid Modeling and App. (SMA), 2001, pp. 249–266.

[3] T. K. Dey, J. Giesen, J. Hudson, Delaunay based shape reconstruction from large data, in: Proc. IEEE Symp. Parallel Vis. and Graph. (PVG), 2001, pp. 19–27.

| model | points | grid | cells | max cell/time | threads | cell time | merge time | total time | triangles |
|---|---|---|---|---|---|---|---|---|---|
| dragon | 437k | 6x1x1 | 6 | 73k pts / 9 min | 6 | 9 min | 4 min | 13 min | 868k |
| buddha | 543k | 4x1x1 | 4 | 136k pts / 25 min | 4 | 25 min | 3 min | 28 min | 1.08m |
| buddha | 543k | 8x1x1 | 8 | 68k pts / 9 min | 8 | 9 min | 2 min | 11 min | 1.08m |
| buddha | 543k | 16x1x1 | 16 | 34k pts / 3 min | 8 | 6 min | 3 min | 9 min | 1.08m |
| thai statue | 5m | 8x12x17 | 221 | 81k / 10 min | 8 | 125 min | 68 min | 193 min | 10.01m |

Table 1: Statistics for some of the examples in Figure 11. Column *cells* is the number of nonempty cells in the grid, each of which is meshed individually (summing to *cell time*). The column *merge time* is dominated by the computation of the inter-cell stitching meshes. Note that for Nx1x1 grids, the cell size is non-uniform, ensuring that a similar number of points is in each cell. We emphasize that our code is in no way optimized, however relative comparisons between the different rows give a sense of how consensus meshing scales with different point set sizes and grid configurations. In particular, we see a nonlinear relationship between the number of points in a cell and the time to mesh them (for the buddha rows). This is to be expected, as we discussed in Section 4.4.

Figure 11: Stress-testing our approach with several standard models. See Table 1 for statistics.


[4] T. K. Dey, S. Goswami, Tight cocone: a water-tight surface reconstructor, in: Proc. ACM Symp. Solid Modeling and App. (SMA), 2003, pp. 127–134.

[5] J. Giesen, M. John, The flow complex: a data structure for geometric modeling, in: Proc ACM-SIAM Symp. Disc. Alg. (SODA), 2003, pp. 285–294.

[6] T. K. Dey, S. Goswami, Provable surface reconstruction from noisy samples, in: Proc. Symp. Comp. Geom. (SCG), 2004, pp. 330–339.

[7] D. Cohen-Steiner, F. Da, A greedy delaunay-based surface reconstruction algorithm, The Visual Computer 20 (1) (2004) 4–16.

[8] M.-C. Chang, F. F. Leymarie, B. B. Kimia, Surface reconstruction from point clouds by transforming the medial scaffold, Comp. Vis. and Image. Under. 113 (2009) 1130–1146.

[9] T. K. Dey, K. Li, E. A. Ramos, R. Wenger, Isotopic reconstruction of surfaces with boundaries, in: Proc. Symp. Geom. Proc. (SGP), 2009, pp. 1371–1382.

[10] T. Dey, R. Dyer, L. Wang, Localized cocone surface reconstruction, Comp. & Graph. 35 (2011) 483–491.

[11] A. Golovinskiy, T. Funkhouser, Randomized cuts for 3d mesh analysis, ACM Trans. Graph. 27 (2008) 145:1–145:12.

[12] X. Sun, P. L. Rosin, R. R. Martin, F. C. Langbein, Random walks for feature-preserving mesh denoising, Comp.-Aided Geom. Des. 25 (7) (2008) 437–456.

[13] R. Schmidt, C. Grimm, B. Wyvill, Interactive decal compositing with discrete exponential maps, ACM Trans. Graph. 25 (3) (2006) 605–613.

[14] N. Amenta, M. Bern, Surface reconstruction by voronoi filtering, in: Proc. Symp. Comp. Geom. (SCG), 1998, pp. 39–48.

[15] N. Amenta, S. Choi, T. K. Dey, N. Leekha, A simple algorithm for homeomorphic surface reconstruction, in: Proc. Symp. Comp. Geom. (SCG), 2000, pp. 213–222.

[16] M. Kazhdan, M. Bolitho, H. Hoppe, Poisson surface reconstruction, in: Proc. Symp. Geom. Proc. (SGP), 2006, pp. 61–70.

[17] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, C. T. Silva, Computing and rendering point set surfaces, IEEE Trans. Vis. Comp. Graph. 9 (2003) 3–15.

[18] O. Schall, M. Samozino, Surface from scattered points: A brief survey of recent developments, in: Intl. Workshop on Semantic Virtual Environments, 2005, pp. 138–147.

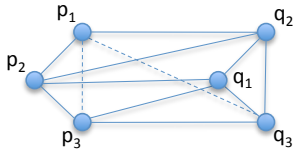
[19] C. E. Scheidegger, S. Fleishman, C. T. Silva, Triangulating point

set surfaces with bounded error, in: Proc. Symp. Geom. Proc (SGP), 2005.
[20] H. Edelsbrunner, E. P. Mucke, Three-dimensional alpha shapes, ACM Trans. Graph. 13 (1994) 43–732.
[21] C.-C. Kuo, H.-T. Yau, A delaunay-based region-growing approach to surface reconstruction from unorganized points, Comput. Aided Des. 37 (2005) 825–835.
[22] D. Dumitriu, S. Funke, M. Kutz, N. Milosavljević, On the locality of extracting a 2-manifold in r3, in: Proc. SWAT '08, 2008, pp. 270–281.
[23] Y. J. Kil, N. Amenta, Gpu-assisted surface reconstruction on locally-uniform samples, in: Proc. Intl. Meshing Roundtable, 2008, pp. 369–385.
[24] R. Mehra, P. Tripathi, A. Sheffer, N. Mitra, Visibility of noisy point cloud data, Comput. Graph. 34 (3) (2010) 219–230.
[25] M. Gopi, S. Krishnan, C. T. Silva, Surface reconstruction based on lower dimensional localized delaunay triangulation, Comp. Graph. Forum 19 (3) (2000) 467–478.
[26] C. Buchart, D. Borro, A. Amundarain, Gpu local triangulation: an interpolating surface reconstruction algorithm., Comput. Graph. Forum 27 (3) (2008) 807–814.
[27] J. R. Shewchuk, Star splaying: an algorithm for repairing delaunay triangulations and convex hulls, in: Proc. SoCG '05, 2005, pp. 237–246.
[28] I. Ivrissimtzis, Y. Lee, S. Lee, W.-K. Jeong, H.-P. Seidel, Neural mesh ensembles, in: Proc. 3DPVT, 2004, pp. 308–315.
[29] M. Yoon, M. Lee, S. Lee, I. Ivrissimtzis, H.-P. Seidel, Surface and normal ensembles for surface reconstruction, Comp. Aid. Des. 39 (2007) 408–420.
[30] Q. Zheng, A. Sharf, A. Tagliasacchi, B. Chen, H. Zhang, A. Sheffer, D. Cohen-Or, Consensus skeleton for non-rigid space-time registration, Comp. Graph. Forum 29 (2) (2010) 635–644.
[31] N. J. Mitra, L. J. Guibas, M. Pauly, Partial and approximate symmetry detection for 3d geometry, ACM Trans. Graph. 25 (2006) 560–568.
[32] D. Anguelov, D. Koller, H.-C. Pang, P. Srinivasan, S. Thrun, Recovering articulated object models from 3d range data, in: Proc. Uncertainty in Art. Intel. (UAI), 2004, pp. 18–26.
[33] P. Cignoni, G. Ranzuglia, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, N. Pietroni, M. Tarini, Meshlab v1.3.0a, http://meshlab.sourceforge.net (September 2011).
[34] R. Schmidt, K. Singh, Sketch-based procedural surface modeling and compositing using Surface Trees, Comp. Graph. Forum 27 (2) (2008) 321–330.
[35] R. Schmidt, Part-based representation and editing of 3d surface models, Ph.D. thesis, University of Toronto, Canada (2010).
[36] J. R. Shewchuk, Applied Computational Geometry: Towards Geometric Engineering, Vol. 1148 of Lecture Notes in Comp. Sci., Springer-Verlag, 1996, Ch. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator.
[37] Y. Lipman, D. Cohen-Or, D. Levin, H. Tal-Ezer, Parameterization-free projection for geometry reconstruction, ACM Trans. Graph. 26.

## Appendix A. Extracting an optimal connected and closed consensus mesh is NP-complete

The associated decision problem of finding an optimal connected and closed census mesh is that of, given a value $K$, determining if there exists a consensus mesh with sum triangle weight $\geq K$. We will show that this problem is NP-complete by performing a reduction from the Traveling Salesman Problem (TSP).

Let a graph $G = (V, E)$ and a distance function $d$ over $E$ describe a TSP instance where we wish to determine if there exists a tour of length $\leq K$. We map this into a consensus mesh instance as follows. For every edge $(p, q) \in E$, let us create six points $p_1, p_2, p_3$ and $q_1, q_2, q_3$. Further, let us create six triangles, $(p_1, p_2, q_2)$, $(p_2, q_1, q_2)$, $(p_2, p_3, q_1)$, $(p_3, q_3, q_1)$, $(p_3, p_1, q_3)$, and $(p_1, q_2, q_3)$.




Let us associate a weight of $-d(p, q)$ to the first triangle and 0 to the rest; finally, let this instance's $K$ be the same as that given for the TSP instance.

Now, given this instance, suppose we can determine in polytime if there exists a consensus mesh with sum weight $\geq K$. If a triangle $(p_1, p_2, q_2)$ is chosen to be in the mesh, then $(p_2, q_1, q_2)$ must also be chosen, since it is the only other triangle incident on edge $(p_2, q_2)$ and it must be selected to guarantee the closed manifold property. This in turn, guarantees $(p_2, p_3, q_1)$ is selected, and so on. In other words, if any of the triangles associated with $(p, q)$ are chosen, then they must all be chosen. Furthermore, if the triangles associated with $(p, q)$ are chosen, then there must exist edges $(o, p)$ and $(q, r)$ all of whose associated triangles must also be chosen since edges $(p_1, p_2)$, $(p_2, p_3)$, $(p_3, p_1)$, $(q_1, q_2)$, $(q_2, q_3)$, and $(q_3, q_1)$ must each also have two incident triangles due to the closed, manifold property. Thus, all vertices of the chosen subgraph have degree 2, and since the consensus mesh is one connected component, the subset of edges chosen from $E$ form a cycle. Since all points must be included in the optimal consensus mesh, all vertices are included in the cycle. And finally, the sum triangle weights of the consensus mesh, which are set using the negated distance of the associated vertices, is $\geq K$ iff the sum distance of the associated cycle is $\leq K$. Since for each edge in $E$ we create a constant number of entities, this reduction is polynomial in time. Lastly, given a $K$, we can check that the sum of chosen triangle weights is $\geq K$ in polytime, proving membership in NP. Thus, extracting an optimal connected and closed consensus mesh is NP-complete.