Pacific Graphics 2014
J. Keyser, Y. J. Kim, and P. Wonka
(Guest Editors)

Volume 33 (2014), Number 7

Fast and Scalable Mesh Superfacets

Patricio Simari[1], Giulia Picciau[2], and Leila De Floriani[2]

[1]Department of Electrical Engineering and Computer Science, The Catholic University of America
[2]Department of Computer Science, Bioengineering, Robotics and Systems Engineering, University of Genova

Abstract

In the field of computer vision, the introduction of a low-level preprocessing step to oversegment images into superpixels *– relatively small regions whose boundaries agree with those of the semantic entities in the scene – has enabled advances in segmentation by reducing the number of elements to be labeled from hundreds of thousands, or millions, to a just few hundred. While some recent works in mesh processing have used an analogous oversegmentation, they were not intended to be general and have relied on graph cut techniques that do not scale to current mesh sizes. Here, we present an iterative superfacet algorithm and introduce adaptations of undersegmentation error and compactness, which are well-motivated and principled metrics from the vision community. We demonstrate that our approach produces results comparable to those of the* normalized cuts *algorithm when evaluated on the Princeton Segmentation Benchmark, while requiring orders of magnitude less time and memory and easily scaling to, and enabling the processing of, much larger meshes.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

1. Introduction

With the advent of higher-resolution geometry acquisition technologies, mesh sizes are growing larger and larger, creating a need for techniques that can enable their processing using available computing resources effectively and efficiently. Digital imaging has previously experienced such a growth and continues to do so. In order to enable the semantic processing of large images by computer vision applications, the concept of *superpixels* was proposed well over a decade ago [RM03]. These comparatively small regions are the result of an oversegmentation whose boundaries are aimed to coincide with those of the semantic elements in the scene. By relying on superpixels rather than directly on pixels, applications can reduce the number of elements under consideration from hundreds of thousands, or even millions, to just a few hundred [RM03, LSD10, FGYZ13].

Recent geometry processing works in the area of cosegmentation have relied on a similar oversegmention concept, which we may call *superfacets*. In particular, the de facto standard has become normalized cuts [SM97, GF08]. While this technique is straightforward to use and produces reliable results, its complexity in time and memory implies that it does not scale to meshes larger than a few tens of thousands of triangles.

Contributions: Inspired by recent advances in computer vision, we present a novel technique for computing superfacets based on an iterative k-means-style approach, computed over the face graph of the mesh. We introduce an especially-tailored edge weighting for the face graph that is scale-invariant and robust to varying mesh resolution, while incorporating terms to favor regularly-shaped superfacets that also adhere to mesh concavities. Our approach is straightforward to implement, yet requires orders of magnitude less memory and running time compared to the normalized cuts approach, enabling the processing of meshes with several million triangles. For an objective, numerical evaluation of our approach, we adapt standard metrics used for the evaluation of superpixels and evaluate our approach on the Princenton Mesh Segmentation Benchmark [CGF09], showing it to produce results which are comparable and often better than those resulting from the normalized cuts approach. In order to demonstrate the advantages of our proposed edge weighting, we compare our results to those obtained using an existing weighting [STK02] and show the error to be substantially lower. Finally, we compare our results with those of Variational Shape Approximation [CSAD04], a very well established variational approach, and show our concavity-favoring heuristic to be well warranted.

Figure 1: *Our superfacet method scales to very large, high-resolution meshes. To process such meshes, normalized cuts would require between 9GB (in the smallest case) and 90TB (in the largest case) of storage for all-pairs distances alone (see Section 5) and a running time which would make it inapplicable in practice (see Figure 5). In contrast, our approach has a modest memory footprint, which is linear on the input mesh size. These results were obtained using 150 superfacets and* $\alpha = 200$*, which controls the tradeoff between compactness and boundary adherence (see Figure 2). For details on mesh sizes and running times of our approach, please refer to Table 1.*

2. Related work

Image segmentation Within computer vision and image processing, the use of *superpixels* has gained wide adoption over the last decade. The technique consists of an oversegmetation of the image into relatively small regions whose boundaries agree with those of the semantic entities in the scene. This is motivated by two main reasons: first, as pointed out by Ren et al. [RM03], pixels are not natural image entities. Moreover, low-level cues may not be appropriate for feature tracking since they are subject to occlusion and clutter. By reducing the number of elements to be labeled from hundreds of thousands, or millions, to a just few hundred the semantic segmentation problem becomes much more tractable.

Superpixels have been used to improve the speed and quality of the results of image segmentation algorithms [RM03, HZR06, MPW*09, KNKY11, ZSL*13], and have been applied to mode searching [Mor05], contour closure [LSD10], image labeling [VFB11], object localization [FVS09, LFWZ13], and saliency detection [FGYZ13]. Superpixels have also provided benefits in the field of video segmentation [RAPM10] and feature tracking [RM07, WLYY11].

Since their introduction, myriad techniques have been investigated for the generation of superpixels, including watershed [VS91], normalized cuts [SM97], mean shift [CMM02], minimum spanning trees [FH04], medoid shift [VS08], level sets [LSK*09], lattice constraints [MPW10], gradient variations [VBM10, ZHMB11], random walks [LTRC11], hill climbing [dBBR*12], geodesic k-means [WZG*12], and reciprocal nearest neighbors [LV13].

Of particular importance to our method is the Simple Linear Iterative Clustering (SLIC) approach introduced by Achanta *et al.* [ASS*12]. This fast and memory-efficient superpixel algorithm is based on k-means. However, for each centroid, the radius within which distances are evaluated is limited by a constant proportional to the desired superpixel size. This results in the complexity of the algorithm being linear on the image size and not dependent on the number of desired superpixels. Achanta *et al.* show that the SLIC approach compares favorably to the state of the art while being the fastest and most memory efficient of the methods compared. Our contribution is to adapt the SLIC technique

for use on polygonal meshes, thus maintaining the speed and memory efficiency which allow our technique to scale to much larger mesh sizes compared to current graph-based methods whose complexity and storage costs limit them to mesh sizes of a few tens of thousands of triangles.

Mesh segmentation A full discussion of the vast area of mesh segmentation algorithms is out of the scope of this paper. We refer the interested reader to available in-depth surveys [Sha08, Sim09, Liu09]. Instead, we focus on the mesh segmentation approaches most relevant to the method presented in this paper.

Of the mesh segmentation algorithms, perhaps the most related to our approach is that of Shlafman *et al.* [STK02], who also use a k-means style iterative approach. However, in contrast to our approach, their metric is not scale invariant, and they compute distances to all vertices from each centroid at every iteration.

Cohen-Steiner *et al.* also present a related k-means-style algorithm called *Variational Shape Approximation* (VSA) [CSAD04] designed to approximate a given mesh using a number of planar proxies. Errors are measured using one of two metrics. The $\mathcal{L}^2$ measures the orthogonal distances from faces in the region to their best-fit plane, while the $\mathcal{L}^{2,1}$ only factors normal deviations from the region's mean normal. In order to avoid computing errors from every proxy to every face, this method employs a single priority queue. However, once elements are taken from the queue in the classification stage, they are not revisited by other proxies. This is in contrast to our approach which makes use of overlapping regions. It should be noted that in the context of VSA-based remeshing, the segmentation regions are heuristically subdivided using anchor vertices, edge extraction, and triangulation stages. However, this post-process is for remeshing purposes only, and is decoupled from the segmentation stage.

In geometry processing [XZT*], and particularly within the sub-area of co-segmentation, several methods have recently arisen that take the approach of using an oversegmentation preprocessing step, motivated by the success of superpixels for image segmentation. Sidi *et al.* [SvKK*11] use an application-specific mean-shift clustering of shape descriptors defined on faces. However, all other methods we have encountered [HKG11, HFL12, WWS*13, MXLH13, WSWL14] use normalized cuts [SM97, GF08]. This produces good results on benchmark meshes where the number of vertices is relatively low. However, this type of approach requires quadratic memory and log-quadratic time (to compute all-pairs graph distances) and does not scale to larger mesh sizes, thus motivating an approach that can produce results of similar quality with much more modest time and memory requirements.

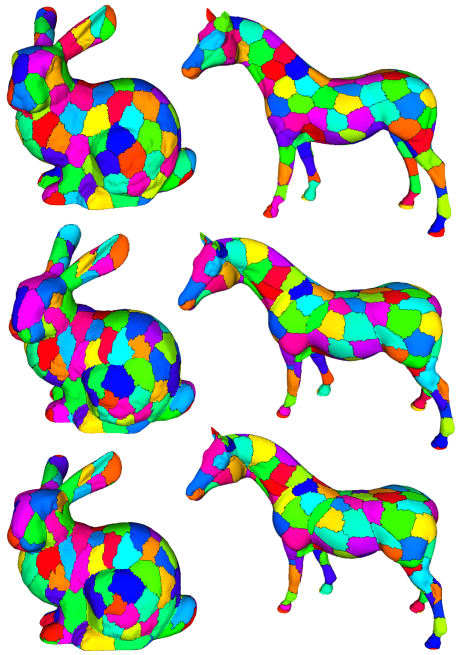

Figure 2: *Visual illustration of the effect of the* α *parameter.* ***Top:*** *A value of* $\alpha = 0$ *produces highly regular and compact superfacets, but they do not adhere well to concavities.* ***Center:*** *A value* $\alpha = 200$ *strikes a good balance between compactness and boundary adherence.* ***Bottom:*** *A value of* $\alpha = 1000$ *produces superfacets with very good adherence to concavities but which are highly irregular. A setting of* $k = 150$ *superfacets was used for all cases. For mesh sizes and running times, please refer to Table 1.*

3. Algorithm description

3.1. Overview

As is standard with k-means style algorithms, our algorithm can be subdivided into three high-level steps: 1) initialization, 2) update of segment centers, and 3) classification of triangles, where steps 2) and 3) are alternately repeated until convergence.

Initialization For the initialization, we use an *iterative farthest point* strategy. We place our first region center at the triangle whose centroid is closest to the centroid of the whole mesh; then, each subsequent center is added at the triangle with maximum Euclidean distance to the nearest already-placed center. This is the initialization which we use in the evaluations presented in this paper and is suited to the case where the number of desired superfacets is known.

If, instead, a desired approximate radius r of the superfacets is known, we allow for two alternative initializations, both of which are supported in our implementation. It should be noted that said radius is specified as a percentage of the bounding box diagonal in order to produce scale-invariant initialization results.

- *Flood initialization*: we start from the triangle closest to the mesh centroid and perform a uniform outward flooding (see the classification step). When the triangle being processed is at a distance from the initial seed greater than $2r$, we use it to begin a new region.
- *Regular grid*: We subdivide the 3D space in which the mesh is embedded into a regular 3D grid based on the desired radius. This can be efficiently done by using integer division by $2r$ on the mesh coordinates to form a key which can then be used with a hash map to store and lookup the corresponding segment ID.

Updating segment centers When all the triangles have been assigned to some superfacet, the position of each superfacet center is computed as follows. For each superfacet, we compute the Euclidean area-weighted mean of all triangle centroids belonging to that superfacet and then designate the new center to be the triangle closest to said mean. Then, we check if this new center is different from the previous one. If no center has changed, we terminate; otherwise the classification step is performed.

Classification For each triangle, we wish to compute its shortest-path distance along the face graph of the mesh to the nearest superfacet center as well as the superfacet label corresponding to said center. Initially, we set these distances to $+\infty$ and the corresponding labels to be undefined. Then, we take the superfacet centers resulting from the the previous update step and, using each as a source, run Dijkstra's algorithm in order to find the shortest paths to the nearby faces. Every time a triangle is reached from a given center, if the shortest-path distance from said center is lower than the currently-stored value (obtained from an expansion from a different centroid or from initialization), the face's distance and label are updated to that of the current center.

There are two key aspects to be noted. The first is the edge-weights used between adjacent triangles, which will be covered in detail in Subsection 3.2. The second is the termination criterion for the breadth-first expansion of Dijkstra's algorithm. Rather than continue until the shortest-path distances to all faces from the current source are computed, we halt the expansion when all faces within a distance threshold from said source have been visited. This threshold is expressed as a multiple of the desired approximate superfacet radius r. Our default is $2r$, which we use for all results presented in this paper. In the case where the number of desired superfacets k is given, we set $r := \sqrt{A/(k\pi)}$, and then use $2r$ as our default multiple as well. As we will see in Subsection 3.3, this early halting strategy (also used by the SLIC superpixel algorithm) is key in keeping the algorithm's complexity sub quadratic, allowing it to scale to meshes with several million triangles. It is possible that, after all expansions are complete, some triangles remain which were not reached from any centroid. In such a case, we seed a new region on the first orphaned triangle found and expand it. This is repeated until no such triangles remain. Typically, after the first iteration, this no longer occurs as the regions cover the entire mesh.

3.2. Face graph weights

As mentioned above, our approach is based on using a shortest-path formulation on the face graph of the mesh. As is standard, we consider two triangles to be adjacent if they share a common mesh edge. The weight between two adjacent triangles is determined by an approximate geodesic term and an angular term, thus combining the notion of surface distance with information about concavities. The first term will favor well-shaped, compact superfacets while the second will favor boundaries aligning with the shape's concavities in accordance with the *minima rule* [Hof84].

Approximate geodesic weight The centroids of the adjacent triangles are taken as points in the embedding 3D Euclidean space and we use a discrete approximation of *geodesic distance* between them. Given two adjacent triangles f_i and f_j with centroids c_i and c_j and a shared edge e_{ij} with mid-point m_{ij}, we calculate the approximate geodesic weight as :

$$\mathrm{geo}\left(f_i, f_j\right) = \|c_i - m_{ij}\| + \|m_{ij} - c_j\| \tag{1}$$

Angular weight For the angular component, we consider the unsigned dihedral angle at the shared edge $\widehat{e_{ij}}$ normalized by π so as to lie in the $[0,1]$ interval and multiplied by the edge length $\|e_{ij}\|$ so that it is an integral measure, making it tolerant to changes in mesh resolution. Then, we check if $\widehat{e_{ij}}$ is convex or concave in order to determine the value of a multiplicative factor $\eta(e_{ij})$. If $\widehat{e_{ij}}$ is convex, then $\eta(e_{ij})$ is set to a small positive value (in our case, 0.2 for all results); otherwise $\eta(e_{ij}) = 1$. This makes it such that crossing a concavity represents a greater distance than crossing a convexity, as suggested by Katz and Tal [KT03]. Thus, the final formulation of the angular weight is:

$$\mathrm{ang}\left(f_i, f_j\right) = \eta(e_{ij})\|e_{ij}\|\frac{\widehat{e_{ij}}}{\pi} \tag{2}$$

Combined weight Finally, we combine the approximate geodesic and angular terms using a weighted sum in order to obtain the final weight w between triangles f_i and f_j.

$$w(f_i, f_j) = \frac{\mathrm{geo}(f_i, f_j) + \alpha\,\mathrm{ang}(f_i, f_j)}{d} \tag{3}$$

where d is the bounding box diagonal of the mesh. Since both the geo and ang terms are in distance units, the latter being an integral measure of a unitless angular ratio along the length of the edge, this normalization makes the weight invariant to global uniform scaling. The parameter α will determine the importance of the angular term: the larger its value, the more closely superfacet boundaries will follow surface concavities, but also the less compact and uniform in size the superfacets themselves will be.

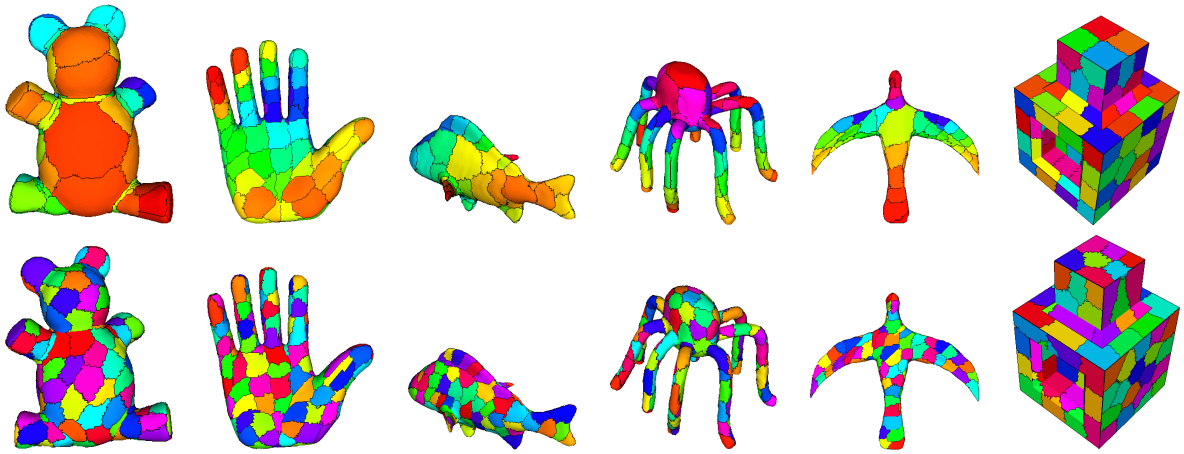

Figure 3: *Illustration of typical results obtained on models from the Princeton Segmentation Benchmark. All cases shown here were obtained with a setting of* $\alpha = 200$ *and* $k = 150$ *superfacets.* ***Top:*** *Results obtained using normalized cuts.* ***Bottom:*** *Results obtained using our method. We observe comparable boundary adherence, while our approach produces more regularly-sized regions and requires orders of magnitude less time and memory (see discussion in Section 5). For a numerical evaluation of results, please see Figure 4. For a comparison of running times, please see Figure 5.*

3.3. Complexity and running times

As mentioned in Subsection 3.1, a key factor in keeping the complexity of our method sub-quadratic is the fact that the Dijkstra expansion step is bounded by a constant multiple of the expected superfacet radius. Similarly to the case of SLIC superpixels [ASS*12], each triangle will fall under the expansion limit of a relatively small number of centroids, a number which does not depend on the mesh scale or resolution. In our experiments this number is ≈ 3.5 on average.

The complexity of our algorithm is dominated by the classification (Dijsktra expansion) step. Let N be the number of triangles of the input mesh and k the number of superfacets, each of which will be approximately of size $M = \frac{N}{k}$. The complexity of the classification step is then $O(kM\log(M))$ due to the fact that we stop the Dijkstra expansion when it reaches the threshold distance. Substituting M and exploiting logarithm properties, we obtain

$$O\left(k\frac{N}{k}\log\frac{N}{k}\right) = O\left(N\left(\log(N) - \log(k)\right)\right) \subseteq O\left(N\log(N)\right).$$

The furthest point initialization step, on the other hand, must do k passes where, for each pass, it selects the triangle with maximum Euclidean distance to its nearest centroid. Our naive implementation is $O(Nk^2)$, which can be reduced to $O(Nk\log(k))$ using a kd tree. However, since this step is computed only once and $k \ll N$, the Dijkstra expansion step dominates the complexity.

When using flood initialization, the complexity of the initialization step becomes the same as that of an expansion step, while the grid initialization would require $O(N)$, since it can determine the initial label of each triangle in constant time using integer division and a hash table, for example.

The final complexity is also affected by the number of iterations. However, this number does not depend on the mesh size and in our experiments the number of iterations is typically less than 20 even for the largest mesh sizes. In a few instances, we observe non-convergence where a small number of triangle labels flip-flop between two regions. We address this by setting a limit number of iterations (we use a maximum of 50 in all experiments).

It bears emphasizing that our method's complexity is a considerable improvement over that of normalized cuts. Given the latter's quadratic memory requirement to store all-pairs shortest path distances, and the super-quadratic time needed to compute them, such an algorithm quickly becomes intractable. In practice, our approach proves to be two orders of magnitude faster than normalized cuts on the relatively small meshes of the Princeton Segmentation Benchmark (Fig. 5), while still being able to easily handle much larger meshes (Table 1), which, in the normalized cuts implementation would require between 9GB (in the smallest case) and 90TB (in the largest case) of storage for all-pairs distances alone. This is further discussed in Section 5.

4. Evaluation metrics

All of the most common evaluation metrics for mesh segmentation algorithms, such as those proposed in the Princeton Mesh Segmentation Benchmark [CGF09], are designed to evaluate the results of traditional segmentation algorithms, which produce few segments of relatively large size. To have an objective evaluation of the oversegmentations produced by a superfacets algorithm, we adapt two metrics from superpixel evaluation.

Mesh	Triangles	Time (sec)
Armadillo	345,944	153
Bunny	69,451	22
Dragon	7,219,045	3,125
Hand	105,860	39
Horse	96,966	34
Kitten	274,196	108
Lion Vase	400,000	137
Neptune	4,007,872	1,797
Ramesses	1,652,528	738

Table 1: *Mesh sizes and corresponding running times of our approach on several large meshes using $k = 150$ superfacets. As shown, our approach enables superfacet segmentations on large, high-resolution meshes, several with over a million triangles. To process such meshes, a normalized cuts implementation would theoretically require between 9GB (in the smallest case) and 90TB (in the largest case) of storage for all-pairs distances alone (see discussion in Section 5) and a running time which would make it inapplicable in practice (see Figure 5). For visual results on these meshes, please refer to Figures 1 and 2.*

Undersegmentation error This error is intended to measure the fraction of regions of an automatic segmentation which overlap with more than one ground-truth segment. In image segmentation, this metric measures the number of pixels that leak across the boundary of the ground-truth. Since mesh triangles are non-uniform in size, we calculate the area of the portion of segment that crosses the ground truth boundary rather than the number of triangles, making it robust to variations in sampling. The formula we use becomes:

$$\mathrm{UE}(s) = \frac{\sum_i \sum_{j:A(s_j \cap g_i)>\tau} A(s_j \setminus g_i)}{\sum_i A(g_i)} \qquad (4)$$

where g_i is the i-th ground truth segment, s_j is the j-th superfacet segment, $A(\cdot)$ denotes the area of a given segment, and τ is the tolerance threshold for the area overlap (in our experiments, 5% of the segment area). Note that $s_j \setminus g_i$ denotes the set-theoretic difference of the regions; *i.e.* the set of triangles that are in s_j but *not* in g_i.

Compactness This is a measure of how well-shaped the regions produced by an oversegmentation are. A straightforward measure is the average ratio between area and square of the perimeter of the region [KT96]. For coherence with the undersegmentation error, for which a lower value indicates a better segmentation, we use the inverse formulation [MB09] and take the square root:

$$\mathrm{compactness}(s) = \mathrm{avg}_k \left(\frac{\mathrm{perimeter}(s_k)}{\sqrt{\mathrm{area}(s_k)}} \right) \qquad (5)$$

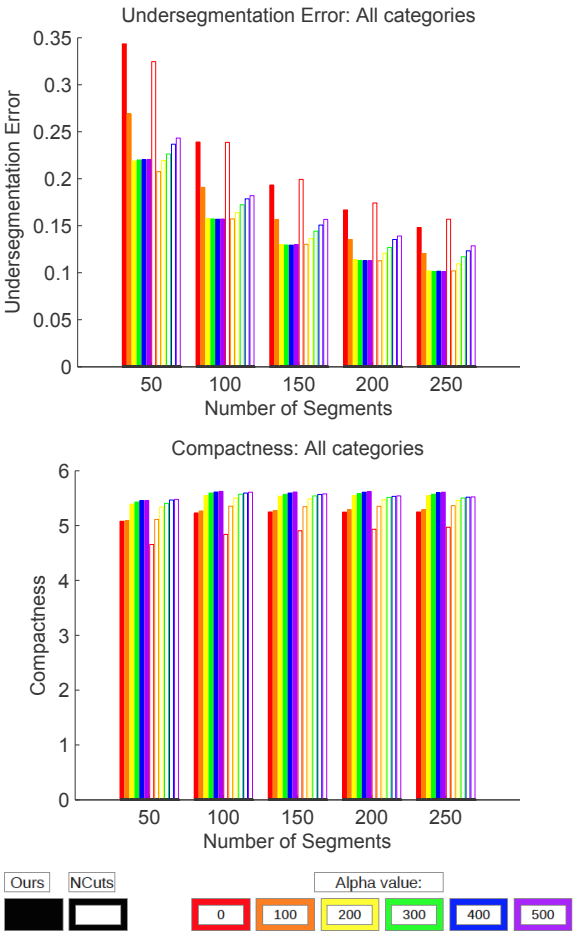

Figure 4: *Comparison of undersegmentation error and compactness values obtained with our approach and normalized cuts over all categories of the Princeton Segmentation Benchmark. Note that for non-zero values of α and higher number of segments our approach produces results which are comparable to those offered by normalized cuts.*

5. Experimental Results

In this section, we present the empirical validation of our method. First, we offer a visual inspection of our results. Figure 1 visually illustrates the results of running our approach on very large meshes (some with several million triangles). The number of triangles and respective running times required by our implementation on these meshes is presented in Table 1.

The main parameter that influences the output of our superfacets algorithm is α, which is the weight that determines the influence of the angular term in the distance formula, allowing control over the trade-off between superfacet compactness and boundary adherence to concavities. Figure 2 visually illustrates the effect of this parameter on two sample meshes. A value of $\alpha = 0$ produces highly regular and

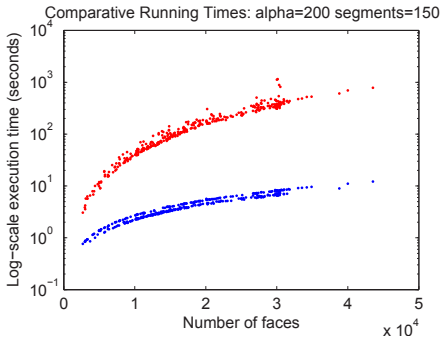

Figure 5: *Running times of our approach (**blue**) and normalized cuts (**red**). Our approach is already two orders of magnitude faster (note the log-scale y axis) than normalized cuts even on relatively low-resolution meshes.*

compact superfacets, but they do not adhere well to concavities. A value of $\alpha = 200$ (our suggested default value) strikes a good balance between compactness and boundary adherence. A value of $\alpha = 1000$ produces superfacets with very good adherence to concavities but which are highly irregular. The sizes and running times for these meshes are also in Table 1. The role of α is further explored in the numerical comparison discussed below.

Next, we numerically evaluate our approach using the metrics presented in Section 4. In particular, we compare the results obtained by our iterative *k*-means style approach and those obtained using normalized cuts [SM97] using the implementation provided by the authors of the approach. We precompute the *NxN* matrix of all-pairs shortest path distances using our same metric and Dijkstra implementation and then pass this matrix to the normalized cuts implementation, which computes a corresponding segmentation. It should be noted that, in this case, no bound can be placed on the expansion in Dijkstra's algorithm since all pairs distances are required by normalized cuts. It should also be noted here that it was necessary for us to skip the larger meshes in the benchmark for which the normalized cuts approach was not able to fit the necessary data in memory; 25 such meshes were skipped, all with approximately 50k faces. In order to have a fair comparison, the same meshes were skipped for both algorithms.

Figure 4 shows the result of this comparison on the Princeton Segmentation Benchmark for a varying number of superfacets and settings of α. In it, we can observe that the best tradeoff between undersegmentation error and compactness seems to lie approximately at $\alpha = 200$ for our approach and $\alpha = 100$ for normalized cuts. A visual comparison of sample segmentations obtained with our method and with normalized cuts is shown in Figure 3.

Figure 5 shows a comparison of running times between our approach and the normalized cuts implementation offered by Shi and Malik [SM97]. It should be noted that,

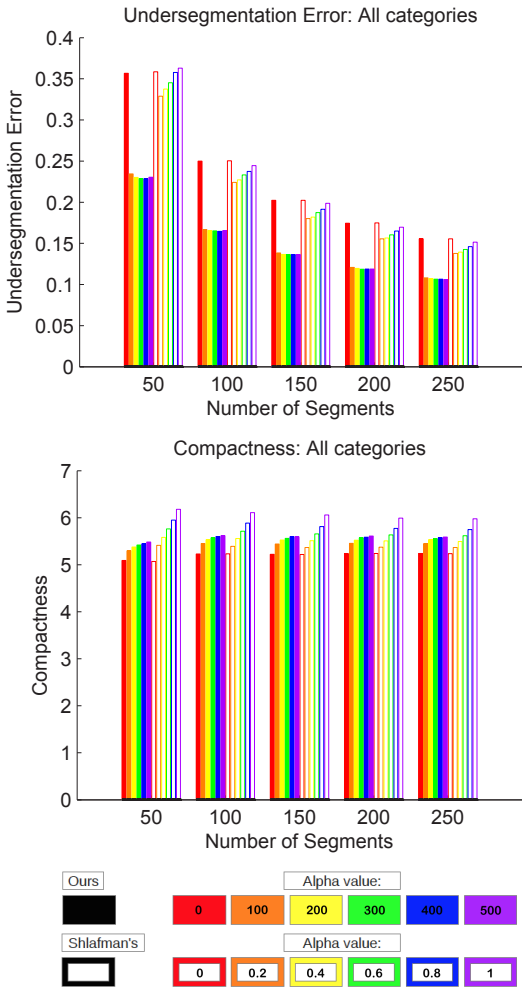

Figure 6: *Results obtained using our metric compared to those obtained using the weighting of Shlafman* et al. *[STK02] over all categories of the Princeton Segmentation Benchmark. Our formulation is scale-invariant, robust to varying resolution, and incorporates elements to favor concavities over convexities, while Shlafman* et al.*'s lacks these properties. As a result, our weights result in drastically lower undersegmentation error and more compact regions.*

while the interface of this code is in Matlab, the underlying implementation is compiled C++ using Mex, so Matlab is only providing the interface for the data. The running times do *not* include the writing/loading of files to disk for communication of data and include *only* the CPU time necessary to compute the distances and the segmentation. As the figure illustrates, our approach is two orders of magnitude faster (note the log-scale *y* axis) even on modest mesh sizes. All experiments were run on an Intel core i7-9390k at 3.20 GHz with 64 GB of RAM under OS Debian 3.13.7-1.

From our experiments we observe that, for non-zero values of α, our approach produces results which are compara-

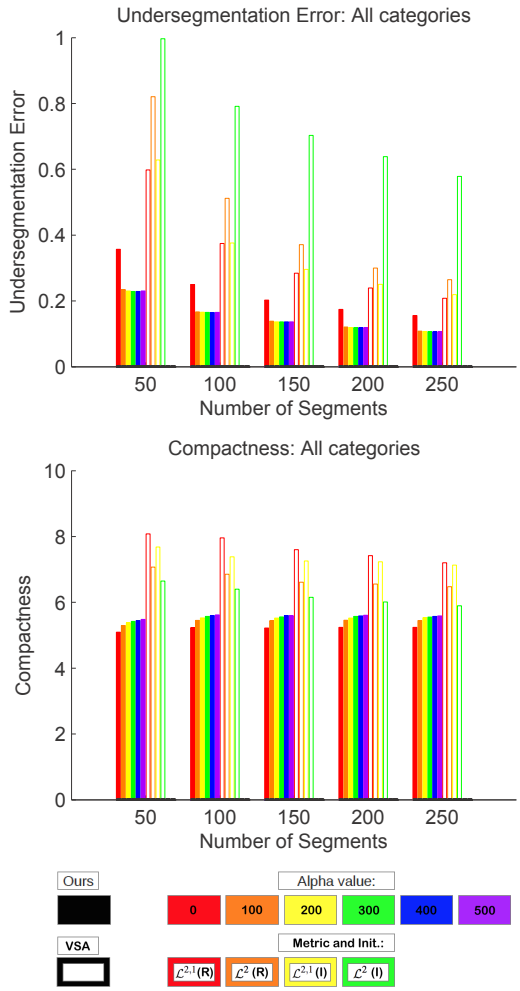

Figure 7: *Results obtained using our algorithm compared to those obtained using the segmentation result of VSA [CSAD04] with the $\mathcal{L}^{2,1}$ and $\mathcal{L}^{2}$ metrics over all categories of the Princeton Segmentation Benchmark. Having not been designed for semantic analysis, VSA produces results which contain large, non-compact segments that fully straddle semantic boundaries. For visual examples of the segmentations produced, see Figure 8.*

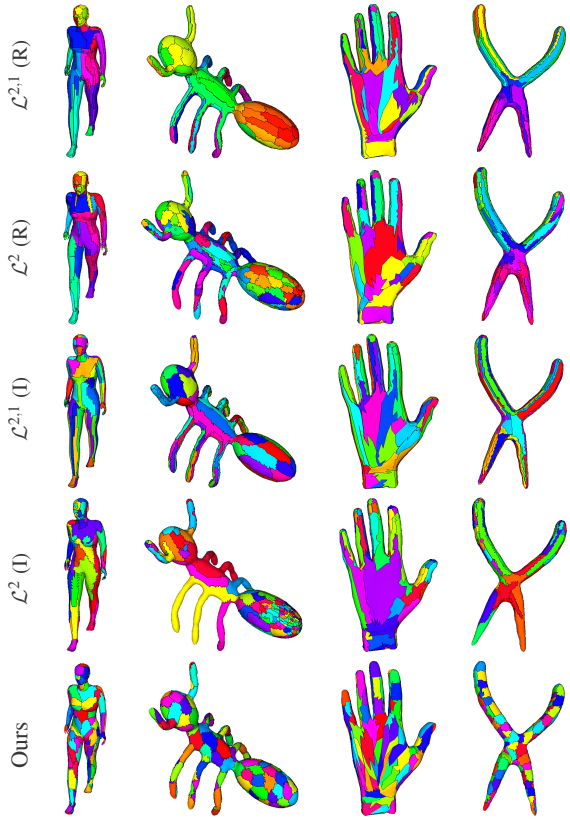

Figure 8: *Results obtained using VSA [CSAD04] using the $\mathcal{L}^{2,1}$ and $\mathcal{L}^{2}$ metrics and the random (R) and incremental proxy addition (I) initialization strategies compared to those obtained using our algorithm (**bottom**) on example meshes from the Princeton Segmentation Benchmark. In all cases we set $k = 150$ superfacets and, in our case, $\alpha = 200$. Having not been designed for semantic analysis, VSA produces results which contain large, non-compact segments that fully straddle semantic boundaries. In contrast, we produce compact segments well aligned with semantic boundaries. For a numerical evaluation of results, please see Figure 7.*

ble to those offered by normalized cuts. Our key contribution, however, is that we can offer such results while requiring orders of magnitude less time and memory, thus enabling superfacet segmentation on *much* larger meshes. Given the quadratic memory required to store all-pairs shortest path distances for the normalized cuts approach, and the super-quadratic time needed to compute these, obtaining results such as those illustrated in Figures 1 and 2 and listed in Table 1 is simply not tractable using such an algorithm. Consider the all-pairs distances storage cost assuming we use 32-bit single-precision floating point and only store the upper triangle of the symmetric distance matrix (something which the implementation of Shi and Malik does not currently allow for). Such cost would range from approximately $\frac{1}{2}(70\text{K})^2 \times 4\,\text{bytes} \approx 9\,\text{GB}$ for the smallest mesh, to approximately $\frac{1}{2}(7\text{M})^2 \times 4\,\text{bytes} \approx 90\,\text{TB}$ for the largest.

Next, we also validate our proposed graph weighting by comparing our results to those obtained using the weighting proposed by Shlafman *et al.* [STK02] whose formula differs from ours as follows:

$$w'(f_i, f_j) = (1-\alpha)\,\text{geo}(f_i, f_j) + \alpha\,(1-(\vec{n}_i \cdot \vec{n}_j)^2) \qquad (6)$$

where $\vec{n}_i$ and $\vec{n}_j$ represent the outward normals of triangles i and j respectively. Here, we have reversed the α and $(1-\alpha)$ terms so a higher α translates to higher importance placed on concavity adherence, thus matching our own.

While we have taken care to make our formulation scale

invariant, robust to varying resolution, and incorporated elements to favor concavities over convexities, the formulation of Shlafman *et al.* lacks all of these properties. As a result, as can be observed in Figure 6, our error is drastically lower.

Lastly, we compare our approach to Variational Shape Approximation (VSA) [CSAD04], the results of which can be observed in Figures 7 and 8. For this comparison, we use the implementation of Lavoué *et al.* [LTD12], which we modify to allow batch processing of the benchmark, exporting of the resulting segmentations over the original mesh, and we incorporate the $\mathcal{L}^2$ metric (see Section 2). We then evaluated VSA on the benchmark using both the $\mathcal{L}^{2,1}$ and $\mathcal{L}^2$ metrics as well as the random (R) and incremental (I) proxy addition initialization strategies.

Unsurprisingly, VSA performs very well with respect to undersegmentation error in the mechanical part category, whose members are often composed of planar regions. In all other categories, however, their error is dramatically higher than ours with respect to both undersegmentation and compactness. This is accounted for by two factors respectively. First, by finding a segmentation composed of approximately planar regions, this method does not emphasize concavities above any other deviations from planarity, resulting in regions which often fully straddle semantic boundaries. Second, in contrast to our approach, neither VSA error explicitly accounts for each face's distance to the region's center, resulting in highly non-compact regions. Note also that the incremental proxy addition strategy results in higher undersegmentation error, since it focuses most of the proxies in curved regions, even when they are concave.

As a final note, while the results of Figures 4, 6, and 7 show the average results across all mesh categories of the Mesh Segmentation Benchmark, and those of Figure 5 show results for one setting of α and k, we observe similar results across all mesh categories and for all other parameter settings. Plots for all cases can be found in the supplemental materials accompanying this paper.

6. Concluding remarks

We have presented a method for computing superfacet segmentations of meshes based on a k-means style approach using shortest-path distances over the face graph of the mesh. By using a bounded expansion strategy in the reclassification step, our approach obtains a log-linear complexity, enabling the segmentation of large meshes (with several million triangles) where applying normalized cuts [SM97] or other such cut-based approaches would be intractable. Furthermore, when compared with normalized cuts using undersegmentation error and compactness metrics, our approach yields comparable, and in some cases favorable results.

It should be noted that we have not incorporated a post processing technique in order to "clean up" the superfacet boundaries. This does not hinder the conclusions yielded through comparing our approach to normalized cuts, and the weighting of Shlafman *et al* [STK02], and VSA segmentation [CSAD04]. However, it should be a priority next step in order to make our approach a complete solution for an end-user wishing to adopt it. That being said, a user with ready access to such a clean-up method can readily employ it as a post-process to our result.

Our formulation readily applies to quad and general polygonal meshes, though we have not implemented this aspect. Our metric, along with the overall algorithm, should also be adaptable to point clouds and tetrahedral meshes with a careful reformulation of weights for adjacent elements.

Finally, our implementation is single-threaded. A parallel implementation would further improve the ability to handle very large meshes.

Acknowledgements: This work has been partially supported by the National Science Foundation under grant number IIS-1116747. We wish to thank Pierre Alliez for his contribution of source code, which we used to supplement the VSA implementation of Lavoué *et al.* [LTD12].

References

[ASS*12] ACHANTA R., SHAJI A., SMITH K., LUCCHI A., FUA P., SUSSTRUNK S.: Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence 34*, 11 (2012), 2274–2282.

[CGF09] CHEN X., GOLOVINSKIY A., FUNKHOUSER T.: A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH) 28*, 3 (Aug. 2009), 73:1–73:12.

[CMM02] COMANICIU D., MEER P., MEMBER S.: Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24* (2002), 603–619.

[CSAD04] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. *ACM Trans. Graph. 23* (2004), 905–914.

[dBBR*12] DEN BERGH M. V., BOIX X., ROIG G., DE CAPITANI B., GOOL L. J. V.: SEEDS: Superpixels extracted via energy-driven sampling. In *European Conference on Computer Vision, ECCV (7)* (2012), vol. 7578 of *Lecture Notes in Computer Science*, Springer, pp. 13–26.

[FGYZ13] FU K., GONG C., YANG J., ZHOU Y.: Salient object detection via color contrast and color distribution. In *Asian conference on Computer Vision* (2013), ACCV, pp. 111–122.

[FH04] FELZENSZWALB P. F., HUTTENLOCHER D. P.: Efficient graph-based image segmentation. *Int. J. Comput. Vision 59*, 2 (Sept. 2004), 167–181.

[FVS09] FULKERSON B., VEDALDI A., SOATTO S.: Class segmentation and object localization with superpixel neighborhoods. In *International Conference on Computer Vision* (2009), IEEE, pp. 670–677.

[GF08] GOLOVINSKIY A., FUNKHOUSER T. A.: Randomized cuts for 3d mesh analysis. *ACM Trans. Graph. 27*, 5 (2008), 145.

[HFL12] HU R., FAN L., LIU L.: Co-segmentation of 3d shapes via subspace clustering. *Comput. Graph. Forum 31*, 5 (2012), 1703–1713.

[HKG11] HUANG Q.-X., KOLTUN V., GUIBAS L. J.: Joint shape segmentation with linear programming. *ACM Transactions on Graphics 30*, 6 (2011), 125.

[Hof84] HOFFMAN D. D.: Parts of recognition. *Cognition 18* (December 1984), 65–96.

[HZR06] HE X., ZEMEL R. S., RAY D.: Learning and incorporating top-down cues in image segmentation. In *European Conf. on Comp. Vision 2006 (ECCV)* (2006), Springer, pp. 338–351.

[KNKY11] KIM S., NOWOZIN S., KOHLI P., YOO C. D. D.: Higher-order correlation clustering for image segmentation. In *Advances in Neural Information Processing Systems 24*. 2011, pp. 1530–1538.

[KT96] KALVIN A. D., TAYLOR R. H.: Superfaces: Polygonal mesh simplification with bounded error. *j-IEEE-CGA 16*, 3 (May 1996), 64–77.

[KT03] KATZ S., TAL A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics 22*, 3 (2003), 954–961.

[LFWZ13] LI L., FENG W., WAN L., ZHANG J.: Maximum cohesive grid of superpixels for fast object localization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2013).

[Liu09] LIU R.: *Spectral Mesh Segmentation*. PhD thesis, 2009.

[LSD10] LEVINSHTEIN A., SMINCHISESCU C., DICKINSON S.: Optimal contour closure by superpixel grouping. In *European Conference on Computer Vision 2010 (ECCV 2010)* (2010).

[LSK*09] LEVINSHTEIN A., STERE A., KUTULAKOS K. N., FLEET D. J., DICKINSON S. J., SIDDIQI K.: Turbopixels: Fast superpixels using geometric flows. *IEEE Trans. on Pattern Analysis and Machine Intelligence 31*, 12 (Dec. 2009), 2290–2297.

[LTD12] LAVOUÉ G., TOLA M., DUPONT F.: MEPP-3D Mesh Processing Platform. *International Conference on Computer Graphics Theory and Applications* (2012), 206–210.

[LTRC11] LIU M.-Y., TUZEL O., RAMALINGAM S., CHELLAPPA R.: Entropy rate superpixel segmentation. In *IEEE Conf. on Comp. Vision and Pattern Recognition (CVPR)* (2011), pp. 2097–2104.

[LV13] LV J.: An approach for superpixels using uniform segmentation and reciprocal nearest neighbors clustering. *J. of Theoretical and Applied Information Technology 47.No.3* (2013).

[MB09] MONTERO R. S., BRIBIESCA E.: State of the art of compactness and circularity measures. *International Mathematical Forum 4*, 25-28 (2009), 1305–1335.

[Mor05] MORI G.: Guiding model search using segmentation. In *Tenth IEEE International Conference on Computer Vision, 2005. ICCV 2005.* (2005), vol. 2, pp. 1417–1423 Vol. 2.

[MPW*09] MOORE A. P., PRINCE S. J. D., WARRELL J., MOHAMMED U., JONES G.: Scene shape priors for superpixel segmentation. In *International Conference on Computer Vision 2009 (ICCV 2009)* (2009), IEEE, pp. 771–778.

[MPW10] MOORE A. P., PRINCE S. J. D., WARRELL J.: Lattice cut - constructing superpixels using layer constraints. In *IEEE Conf. on Comp. Vision and Pattern Recognition (CVPR)* (2010).

[MXLH13] MENG M., XIA J., LUO J., HE Y.: Unsupervised co-segmentation for 3D shapes using iterative multi-label optimization. *Computer-Aided Design 45*, 2 (Feb. 2013), 312–320.

[RAPM10] REINA A. V., AVIDAN S., PFISTER H., MILLER E. L.: Multiple hypothesis video segmentation from superpixel flows. In *European Conference on Computer Vision* (2010), vol. 6315 of *Lecture Notes in Computer Science*, pp. 268–281.

[RM03] REN X., MALIK J.: Learning a classification model for segmentation. In *IEEE International Conference on Computer Vision* (2003), pp. 10–17 vol.1.

[RM07] REN X., MALIK J.: Tracking as repeated Figure/Ground segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2007), IEEE Computer Society.

[Sha08] SHAMIR A.: A survey on mesh segmentation techniques. *Computer Graphics Forum 27* (2008), 1539–1556.

[Sim09] SIMARI P. D.: *Algorithms in 3D Shape Segmentation*. PhD thesis, Toronto, Ont., Canada, Canada, 2009. AAINR61094.

[SM97] SHI J., MALIK J.: Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22* (1997), 888–905.

[STK02] SHLAFMAN S., TAL A., KATZ S.: Metamorphosis of polyhedral surfaces using decomposition. In *Computer Graphics Forum* (2002), pp. 219–228.

[SvKK*11] SIDI O., VAN KAICK O., KLEIMAN Y., ZHANG H., COHEN-OR D.: Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM Transactions on Graphics 30*, 6 (2011), 126:1–126:9.

[VBM10] VEKSLER O., BOYKOV Y., MEHRANI P.: Superpixels and supervoxels in an energy optimization framework. In *European conference on Computer vision* (2010), ECCV'10, Springer-Verlag, pp. 211–224.

[VFB11] VEZHNEVETS A., FERRARI V., BUHMANN J. M.: Weakly supervised semantic segmentation with a Multi-Image model. In *IEEE Int. Conference on Computer Vision* (2011).

[VS91] VINCENT L., SOILLE P.: Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 6 (June 1991), 583–598.

[VS08] VEDALDI A., SOATTO S.: Quick shift and kernel methods for mode seeking. In *European Conference on Computer Vision (ECCV)* (2008).

[WLYY11] WANG S., LU H., YANG F., YANG M.-H.: Superpixel tracking. In *International Conference on Computer Vision (ICCV)* (2011), IEEE, pp. 1323–1330.

[WSWL14] WU Z., SHOU R., WANG Y., LIU X.: Interactive shape co-segmentation via label propagation. *Computers and Graphics 38*, C (Feb. 2014), 248–254.

[WWS*13] WU Z., WANG Y., SHOU R., CHEN B., LIU X.: Unsupervised co-segmentation of 3D shapes via affinity aggregation spectral clustering. *Computers and Graphics 37*, 6 (2013), 628–637.

[WZG*12] WANG P., ZENG G., GAN R., WANG J., ZHA H.: Structure-Sensitive superpixels via geodesic distance. *International Journal of Computer Vision 103* (Nov. 2012), 1–21.

[XZT*] XU K., ZHANG H., TAGLIASACCHI A., LIU L., LI G., MENG M., XIONG Y.: Partial intrinsic reflectional symmetry of 3d shapes.

[ZHMB11] ZHANG Y., HARTLEY R. I., MASHFORD J., BURN S.: Superpixels via pseudo-boolean optimization. In *Int. Conf. on Computer Vision (ICCV)* (2011), IEEE, pp. 1387–1394.

[ZSL*13] ZHANG L., SONG M., LIU Z., LIU X., BU J., CHEN C.: Probabilistic Graphlet Cut: Exploiting Spatial Structure Cue for Weakly Supervised Image Segmentation. In *Conf. on Comp. Vision and Pattern Recognition* (June 2013), IEEE.