Supertetras: A Superpixel Analog for Tetrahedral Mesh Oversegmentation

Giulia Picciau[1(✉)], Patricio Simari[2], Federico Iuricich[3], and Leila De Floriani[1]

[1] Università Degli Studi di Genova, Genova, Italy
giulia.picciau@dibris.unige.it
[2] The Catholic University of America, Washington, DC, USA
[3] University of Maryland College Park, College Park, USA

Abstract. Over the past decade, computer vision algorithms have transitioned from relying on the direct, pixel-based representation of images to the use of *superpixels*, small regions whose boundaries agree with image contours. This intermediate representation improves the tractability of image understanding because it reduces the number of primitives to be taken under consideration from several million to a few hundred. Despite the improvements yielded in the area of image segmentation, the concept of an oversegmentation as an intermediate representation has not been adopted in volumetric mesh processing. We take a first step in this direction, adapting a fast and efficient superpixel algorithm to the tetrahedral mesh case, present results which demonstrate the quality of the output oversegmentation, and illustrate its use in a semantic segmentation application.

1 Introduction

Over the past decade, computer vision algorithms have gained many benefits from the introduction of the concept of *superpixels*, which are obtained from an oversegmentation of an image, comprising possibly millions of pixels, into just a few hundred small regions that become the new primitives on which image processing algorithms work. These regions are well aligned with the semantic boundaries of the image, and their relatively low number greatly improves the tractability of the segmentation problem while also allowing for the consideration of mid-level elements such as texture information.

Some approaches have been introduced to extend the idea of superpixel oversegmentation to surface meshes [26], but, to our knowledge, the extension to the volumetric case has not yet been investigated. These models, often used in tasks such as medical data analysis and simulation, usually consist of a very large number of primitives, often in the tens of millions. For this reason, we believe that an intermediate representation of these models obtained by means of an oversegmentation could prove greatly beneficial to semantic segmentation.

We take a first step towards the application of an oversegmentation as a preprocessing step for tetrahedral meshes, extending a state of-the-art superpixel

V. Murino and E. Puppo (Eds.): ICIAP 2015, Part I, LNCS 9279, pp. 375–386, 2015.
DOI: 10.1007/978-3-319-23231-7_34

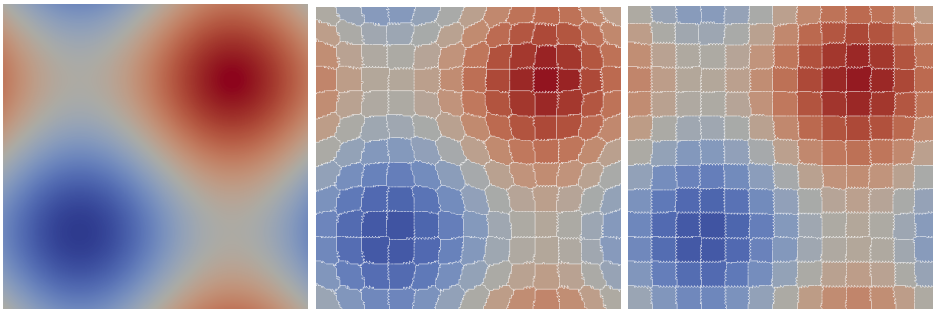

Fig. 1. Field value $f(x, y, z) = sin(x) + sin(y) + sin(z)$, with $x, y, z \in [-\pi, \pi]$ (left), and supertetrahedral segmentation with $w = 0.5$ (center) and $w = 1.5$ (right). Note how the lower w results in regions more aligned with field features, while higher w results in a more regular distribution.

algorithm [1]. The k-means based approach is efficient in time and memory making it suitable for tetrahedral meshes with several million elements for which graph-based methods would not be suitable.

2 Related Work

2.1 Superpixels in Image Processing

Superpixels are often used in Computer Vision as a representation to lower the number of primitives under consideration, thus improving the tractability of the segmentation task [12,14,16,17,24]. *Supervoxels* are the 3D equivalent of superpixels, and have been applied to the segmentation of medical images [19,32] and video sequences [23,36].

Superpixel algorithms use various strategies, including graph-cuts [10,25], watershed [31], mean and medoid shift [7,30], hill-climbing [3], geodesic k-means [33], and reciprocal nearest neighbors [20].

The approach most relevant to our method is the *Simple Linear Iterative Clustering* (SLIC) algorithm [1]. It is a k-means based approach which limits the expansion of each centroid by a constant proportional to the desired superpixel size. This limited expansion results in a complexity which is linear in the image size and is independent of the number of superpixels. The distance metric used is a weighted sum of a Euclidean term to account for the position of a pixel in the image and an intensity term. This formulation yields compact superpixels which adhere well to image contours while being significantly faster than other state-of-the-art methods. The method's low time and space complexity make it an ideal candidate to scale to high-resolution tetrahedral models.

The idea of limited region expansion has been adopted also in the approach of Papon et al. [22], which is an extension of the superpixel concept to point clouds: after initializing the segmentation superimposing a regular grid to the

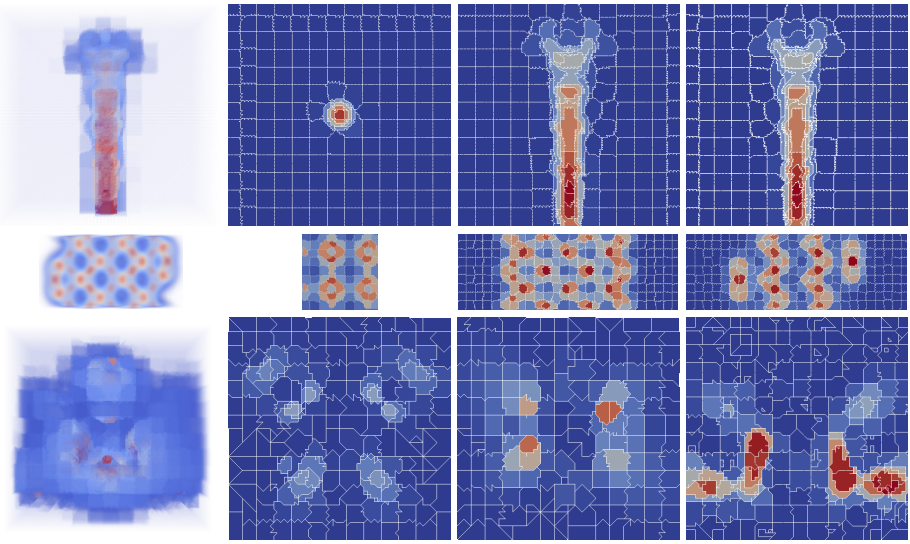

Fig. 2. Volumetric visualization and central x, y, and z slices of physical simulation of a fuel jet (top), silicium material (center), and electron orbit (bottom) segmented into 2000 supertetrahedra with $w = 1.5$.

cloud, they iteratively alternate between a classification step and a centroid update step, until convergence is reached.

2.2 Tetrahedral Domain

Much of the work in the area of tetrahedral mesh research focuses on the quality tetrahedralization of a given volume. This is a difficult task since the regular tetrahedron does not tile 3D space, and state-of-the-art methods usually rely on a Delaunay-based process to tetrahedralize the interior of an object. Tetrahedral meshes are usually computed from point samples [9], regular volume data [8,29], from a set of calibrated images [27], or tetrahedralizing the interior of a triangle mesh [13].

Many techniques have been applied for segmenting a tetrahedral mesh, including traditional unsupervised clustering techniques such as Gaussian mixture models [21], max flow [4], region growing [15], and iterative merging [8], as well as methods based on mathematical concepts such as stable manifolds [2,9], separatrix persistence [11], topological-equivalence regions [6] and discrete Morse theory [35]. Last but not least, many of the best-performing techniques from a semantic point of view apply graph cut methods [27–29].

All of the above methods are designed to produce a segmentation of the input model into a relatively low number of segments, each of relatively large size. To obtain an oversegmentation, applying these techniques would be less than ideal.

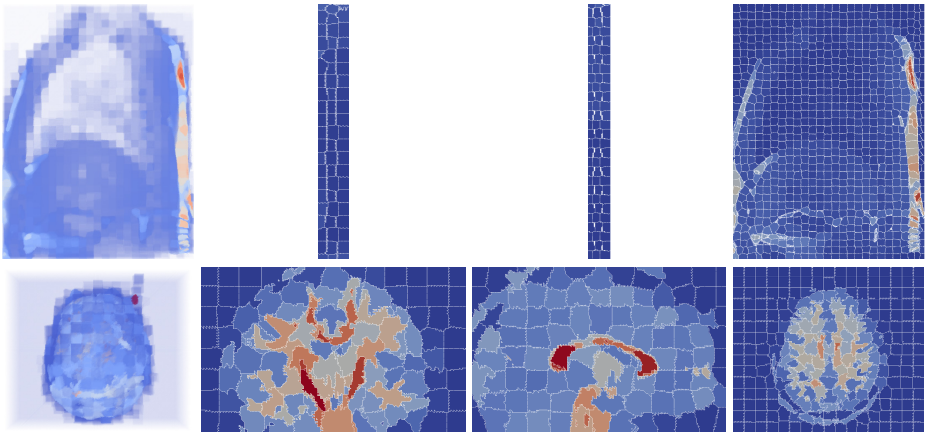

Fig. 3. Volumetric representation and central x, y, and z slices of medical data of a human thorax (top) and brain (bottom) segmented into 2000 segments with $w = 1.5$.

Techniques such as watershed, region merging, and their mathematically-related extensions allow for little control over the variance in size and shape of the resulting segments, usually producing irregular and unevenly-sized regions that are less than ideal as intermediate representations in the style of super-pixels. Graph-cut algorithms, on the other hand, require quadratic storage for an all-pairs affinity matrix, and so do not scale well to the case of graphs with millions of nodes. This makes them inappropriate since the size and resolution of the tetrahedral datasets used in practical applications is increasing. In contrast, as previously mentioned, our adaptation of the SLIC approach [1] produces semantically accurate results while keeping the complexity linear in the number of primitives.

It should be emphasized that our approach is not meant to replace the previously-mentioned tetrahedral segmentation algorithms, but rather to be used as a pre-processing step to enable their execution on a domain consisting of much fewer, well-shaped primitives conforming to semantic boundaries.

3 Algorithm

Our aim is to develop an adaptation of the SLIC algorithm [1] from the image domain to 3D scalar fields defined over the vertices of a tetrahedral mesh, considering tetrahedra rather than pixels. The concept of oversegmentation has already been extended to triangle meshes in [26], but the domain presents several differences with respect to the tetrahedral one: first, only the surface of the object is considered, and therefore every distance calculation has to be constrained to lie on the surface. Second, triangle meshes do not have a scalar field defined on them. Rather, information about the angle between adjacent faces is added to the spatial information given by the vertices as a "curvature" term. Given that

in the case of the tetrahedral mesh the field is defined over vertices rather than tetrahedra, we consider the centroids of the latter, assigning as a field value the average field value at the tetrahedron's vertices. While pixels lie on a regular grid and adjacencies are implicitly given, in the tetrahedral mesh case we must encode and use adjacency information explicitly during the clustering process. To store the mesh, we use a structure that stores information about tetrahedra and their vertices together with the adjacency relations between them [5]. The segmentation process is based on the k-means algorithm, and thus consists of an initialization step followed by centroid update and a classification steps, with the latter two steps repeated until convergence. We describe each one of these steps more in detail below.

Field Normalization Preprocess: To be robust across models with varying ranges of field values, we normalize the scalar field defined over the vertices to have unit variance, taking the field $\bar{f}_i = f_i/\sigma_f$, where f_i is the field value of the i-th tetrahedron (taken as the mean field value of its vertices) and σ_f is the standard deviation of the set of all field values of the model.

Initialization: We superimpose a virtual regular grid on the domain with cells of width $S = \sqrt[3]{v/k}$, where v is the total volume of the mesh's 3D domain and k is the number of desired regions. Each tetrahedron is assigned to its corresponding discrete cell taking the integer division of its 3D coordinates and cell width S.

Centroid Update: Each supertetrahedral region's centroid is calculated as the volume-weighted average of the barycenters of the tetrahedra belonging to said region. This centroid need not fall exactly on any of the region's barycenters, and is free to lie within the continuous 3D domain.

Tetrahedral Re-Classification This step assigns each tetrahedron to the region associated with the nearest region centroid using a breadth-first expansion that starts from the tetrahedron closest to the centroid. The metric used to evaluate the distance is described in Section 3.1. Limiting the expansion to $2S$ allows the complexity of this step to remain linear in the number of tetrahedra regardless of the number of regions.

3.1 Distance Metric

The distance metric we use is an adaptation of the formula presented in SLIC [1] to the case of scalar field defined over the vertices of a tetrahedral mesh. As a consequence, it consists of the weighted sum of a term which carries spatial information and a term for the field value. The spatial term is the Euclidean distance between the 3D barycenters of tetrahedra i and j: $||c_i - c_j||$. Since the fields we consider are scalar, we define the field term as $|\bar{f}_i - \bar{f}_j|$. So that these terms are robust to variations in mesh scale and scalar field intensity across models, they are normalized and combined as follows:

$$d_{i,j} = (\bar{f}_i - \bar{f}_j)^2 + \left(w \frac{||c_i - c_j||}{S} \right)^2 \quad (1)$$

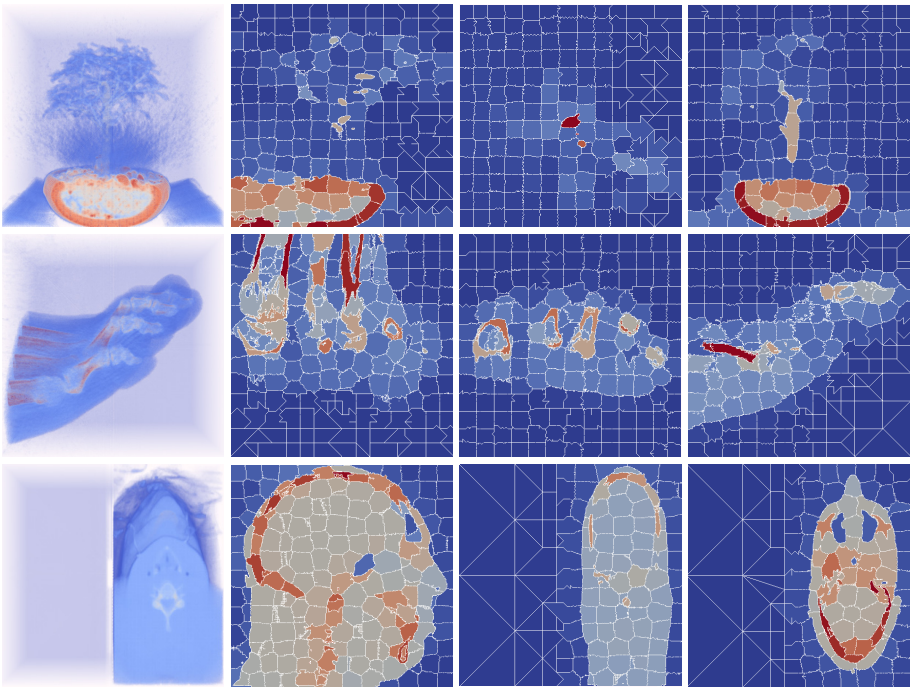

Fig. 4. Volumetric representation and central x, y, and z slices of meshes with non-regular domains segmented into 2000 segments with $w = 1.5$.

where S is the previously-defined supertetrahedral diameter, and w is a user-specified weight. Lower values of w will give more importance to the field values, and will favor more irregular regions that better adapt to the field's features. In contrast, higher values of w will result in more regularly-shaped regions. This effect can be observed in Figure 1. We have found experimentally that a good range of values for w is $[0.5, 2.0]$.

3.2 Notes on Time and Space Complexity

Analogously to the original SLIC algorithm, the complexity of the expansion step is linear in the number of tetrahedra, because the expansion of each region is limited by a constant proportional to the region size. Achieving this complexity in the tetrahedral case requires a regular spatial grid to which all tetrahedral elements register, resulting in a constant number of tetrahedra in each grid cell. With such a spatial index, the tetrahedron closest to each supertetrahedral region's centroid can be found in constant time during the reclassification step. A k-d tree or octree can be used instead, with a very modest reduction in performance with respect to the previous alternative.

The supertetrahedral representation is effective for compactly encoding simplicial meshes. In our experiments we compare the latter with another compact

Table 1. IA and supertetra comparison and running times for the latter.

Model	#V	#T	#ST	#Arcs	IA (MB)	ST (MB)	ST (s)	Ncut (s)
Silicium	113K	634K	2.1K	12.2K	22.80	13.22	21.76	245.43
Neghip	129K	728K	2.2K	11.3K	26.14	15.12	47.22	284.93
Fuel	262K	1.5M	2.2K	9K	53.78	30.96	40.49	332.15
Thorax	768K	4.1M	2.7K	14.7K	149.08	86.37	137.80	712.81
Brain	860K	5M	2K	12.4K	178.54	102.49	451.40	480.15
Trigonom.	1M	5.8M	2.1K	13.6K	208.18	119.45	193.18	591.52
Bonsai	4.5M	25.8M	2.2K	13.2K	926.18	531.92	4613.90	1774.81
VisMale	4.9M	28M	1.6K	8.8K	1,007.11	578.70	6,529.66	1,712.95
Foot	5.3M	31.2M	2.1K	12.5K	1,115.44	638.90	5,788.87	2,054.03

encoding for simplicial meshes: the IA data structure [5]. For the sake of clarity, we distinguish among the three types of entities encoded in the IA data structure: the geometry, which is represented by the vertex coordinates, the connectivity, which encodes the relations between tetrahedra and vertices, and the dual graph. Considering a tetrahedral mesh Σ, its dual graph is the graph $G = (N, A)$ where the nodes N are in one-to-one correspondence with the tetrahedra in Σ. An arc in A connects two nodes if and only if the corresponding tetrahedra are face-adjacent (i.e. they share a triangle).

In our IA implementation, vertices are indexed in a single array containing, for each of them, their coordinates and an additional scalar value, thus storing $4v$ float values, where v is the number of vertices in Σ. Tetrahedra are also indexed in a single array. Connectivity information is encoded by storing, for each tetrahedron, the four indices of its boundary vertices, thus encoding $4t$ indexes in total, where t is the number of tetrahedra in Σ. Lastly, the dual graph can be efficiently represented exploiting the regularity of the tetrahedra adjacencies. Since each tetrahedron has at most four adjacent tetrahedra, we can use a constant-size array of length $4t$ storing, for each tetrahedron, the four indices of the adjacent tetrahedra. The resulting overall storage cost of the IA structure is then $4v + 8t$.

The information represented in the supertetra structure can be conceptually subdivided as we have done for the IA representation. The geometric and combinatorial information still consists of three coordinates per vertex plus a scalar field value and $4t$ indexes, which reference each tetrahedron's vertices. However, adjacency relations are now considered between supertetra regions rather than tetrahedra. The dual graph $G = (N, A)$ now encodes one node for each supertetra region and one arc for each adjacency relation between two supertetra. Thus, each supertetra encodes the list of adjacent regions as a list of pointers to the latter. Since each adjacency relation is referred twice (i.e. a pointer is encoded on both the adjacent regions) the total number of pointers is $2|A|$. As a consequence, the storage cost of this representation becomes $4v + 4t + 2|A|$.

We have estimated the spatial occupation of the two data structures encoding real datasets and we show the obtained results in Table 3.2. We are assuming a storage cost of 8 bytes for each coordinate and scalar value and a storage cost of 4 bytes for each pointer or index over an array. We note that the supertetra representation is approximately 40% more compact than the IA representation.

4 Results

In the following, we show the results of applying our oversegmentation algorithm to sample datasets. Figure 1 illustrates the effect of the w parameter on the resulting regularity, Figure 2 shows results on models of physical simulations, and Figure 3 illustrates results on anatomical data. Lastly, Figure 5 demonstrates the application of our supertetrahedral representation to a semantic segmentation. Since the objects are volumetric, we visualize the slices obtained by cutting the mesh along its central axes. Please refer to figure captions for more details. In each case, note how the "empty space" (marked by constant field values) is divided into an approximately regular grid, while in detailed areas the supertetrahedral boundaries follow the contours of the field's features. Most of the meshes we use are regular, but our approach also work on non-regular or semi-regular meshes, as illustrated in Figure 4.

4.1 Enabling Normalized Cuts on Large Datasets

Usually, the results of an oversegmentation are computed as a preprocess, the results of which are fed as input to some subsequent algorithm, improving its results and/or efficiency. Here, we apply the Normalized Cuts algorithm of Shi and Malik [25]. This graph cut method produces high quality results, but its drawback are its $O\left(n^{\frac{3}{2}}\right)$ complexity and quadratic storage costs. The latter, in particular, would make the application of the normalized cuts approach to one of these high resolution models completely intractable. Let us 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). The brain dataset, comprising approximately five million tetrahedra, would require $\frac{1}{2}(5M)^2 \times 4$ bytes ≈ 50 terabytes to store the distance matrix alone.

The oversegmentations we produce, illustrated in Figure 5, are composed of approximately 2000 regions, and therefore it is possible to perform a normalized cut considering the regions as nodes of the graph. We use the same distance metric defined in equation (1), computing it between the centroids of adjacent supertetrahedral regions, and then find all-pairs shortest path distances over the adjacency graph of these regions. Finally, we apply Normalized Cuts. Figure 5 illustrates the results of this procedure on regular and irregular tetrahedral domains respectively. The supertetrahedral representation is what enables graph-cut results such as these at this resolution.

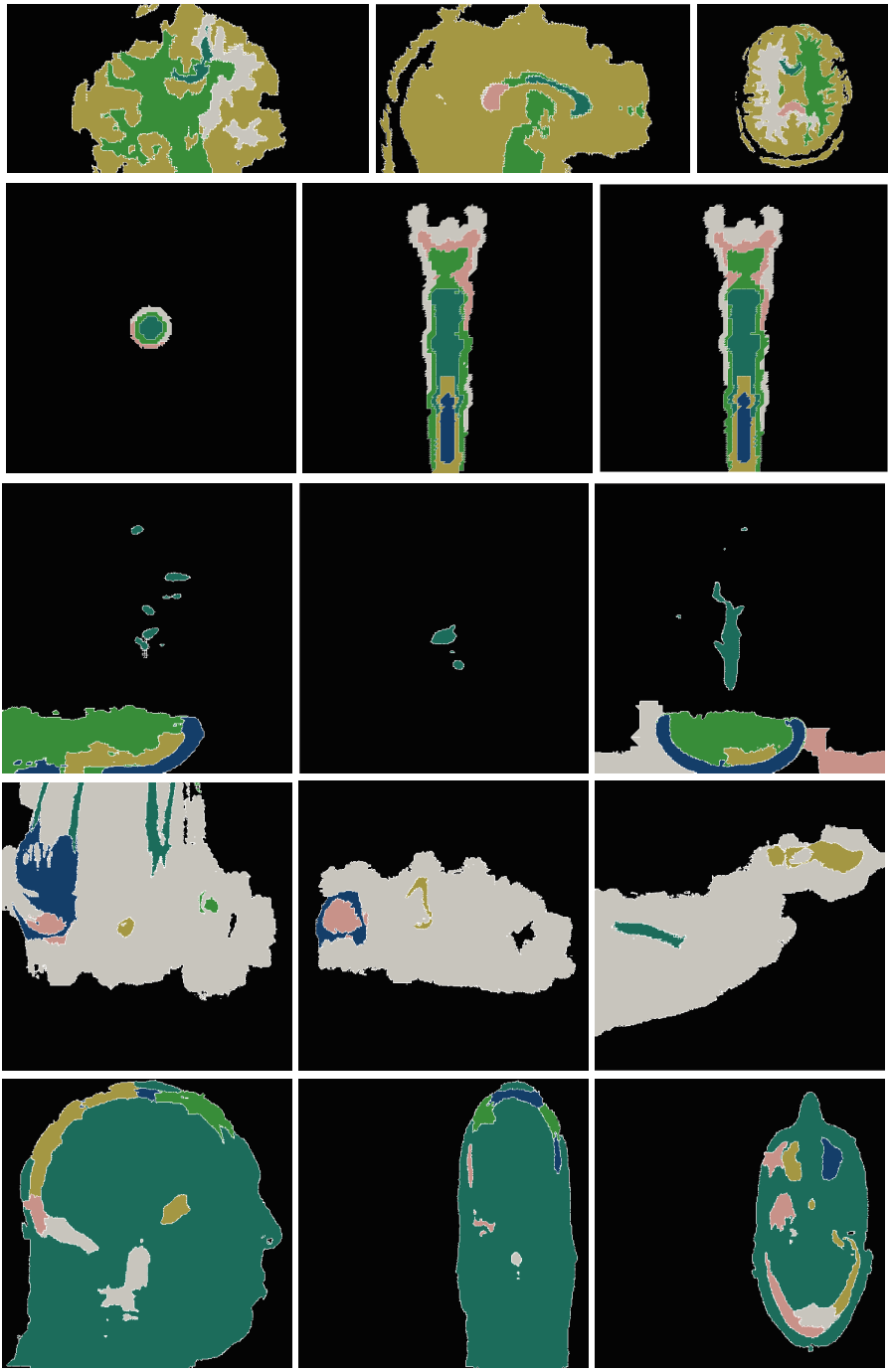

Fig. 5. Results of applying Normalized Cuts on our supertetrahedral representation. A direct application of such a graph-cut algorithm would be completely intractable due to the prohibitive storage costs of the all-pairs distance matrix.

5 Future Work

We have introduced a tetrahedral mesh segmentation algorithm, which can be regarded as an extension of a state-of-the-art superpixel algorithm to tetrahedral mesh representations with scalar fields defined over their vertices. We illustrated the results of our method by applying it to several models, including analytic, physical simulation, and medical datasets.

Since oversegmentation is not a final aim in image and mesh processing, but rather an intermediate step applied to improve the tractability of some subsequent task, we demonstrated the utility of our approach by showing how it could be used to enable the application of a graph cut approach to high resolution models, a task which would have been intractable otherwise.

Future work will focus on exploring other uses of the supertetrahedral representation, as well as extending the approach to higher-dimensional and time-varying meshes, and augmenting the regions with feature vectors that encode texture and other mid-level perceptual cues.

Acknowledgments. This work was supported in part by the National Science Foundation under grant number IIS-1116747.

References

1. 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), 2274–2282 (2012)
2. Alexa, M., Rusinkiewicz, S., Dey, T.K., Giesen, J., Goswami, S.: Shape segmentation and matching from noisy point clouds
3. Van den Bergh, M., Boix, X., Roig, G., de Capitani, B., Van Gool, L.: SEEDS: Superpixels extracted via energy-driven sampling. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part VII. LNCS, vol. 7578, pp. 13–26. Springer, Heidelberg (2012)
4. Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Trans. Pattern Anal. Mach. Intell. **26**(9), 1124–1137 (2004)
5. Canino, D., De Floriani, L., Weiss, K.: IA*: An adjacency-based representation for non-manifold simplicial shapes in arbitrary dimensions. Computer and Graphics **35**(3), 747–753 (2011)
6. Chiang, Y.J., Lu, X.: Progressive simplification of tetrahedral meshes preserving all isosurface topologies. Comput. Graph. Forum **22**(3), 493–504 (2003)
7. Comaniciu, D., Meer, P., Member, S.: Mean shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**, 603–619 (2002)
8. Cuadros-vargas, A.J., Nonato, L.G., Tejada, E., Ertl, T.: Generating segmented tetrahedral meshes from regular volume data for simulation and visualization applications. In: Proceedings of CompIMAGE. Taylor and Francis Group (2006)
9. Dey, T.K., Giesen, J., Goswami, S.: Shape segmentation and matching with flow discretization. In: Dehne, F., Sack, J.-R., Smid, M. (eds.) WADS 2003. LNCS, vol. 2748, pp. 25–36. Springer, Heidelberg (2003)

10. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. Int. J. Comput. Vision **59**(2), 167–181 (2004)
11. Gunther, D., Seidel, H.P., Weinkauf, T.: Extraction of dominant extremal structures in volumetric data using separatrix persistence. Computer Graphics Forum **31**(8), 2554–2566 (2012)
12. He, X., Zemel, R.S., Ray, D.: Learning and incorporating top-down cues in image segmentation. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part I. LNCS, vol. 3951, pp. 338–351. Springer, Heidelberg (2006)
13. Jimenez, J.J., Segura, R.J.: Collision detection between complex polyhedra. Comput. Graph. **32**(4), 402–411 (2008)
14. Kim, S., Nowozin, S., Kohli, P., Yoo, C.D.D.: Higher-order correlation clustering for image segmentation. In: Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K. (eds.) Advances in Neural Information Processing Systems, vol. 24, pp. 1530–1538 (2011)
15. Kurita, T.: An efficient agglomerative clustering algorithm for region growing (1994)
16. Levinshtein, A., Sminchisescu, C., Dickinson, S.: Optimal contour closure by superpixel grouping. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part II. LNCS, vol. 6312, pp. 480–493. Springer, Heidelberg (2010)
17. Levinshtein, A., Stere, A., Kutulakos, K.N., Fleet, D.J., Dickinson, S.J., Siddiqi, K.: Turbopixels: Fast superpixels using geometric flows. IEEE Transactions on Pattern Analysis and Machine Intelligence **31**(12), 2290–2297 (2009)
18. Liu, M.Y., Tuzel, O., Ramalingam, S., Chellappa, R.: Entropy rate superpixel segmentation. In: Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, pp. 2097–2104. IEEE Computer Society, Washington, DC (2011)
19. Lucchi, A., Smith, K., Achanta, R., Knott, G., Fua, P.: Supervoxel-Based segmentation of EM image stacks with learned shape features. Tech. rep, EPFL (2010)
20. LV, J.: An approach for superpixels using uniform segmentation and reciprocal nearest neighbors clustering. Journal of Theoretical and Applied Information Technology **47**(3) (2013)
21. kay Ng, S., Mclachlan, G.J.: On some variants of the em algorithm for fitting finite mixture models. Australian Journal of Statistics (2003)
22. Papon, J., Abramov, A., Schoeler, M., Worgotter, F.: Voxel cloud connectivity segmentation - supervoxels for point clouds. In: IEEE (ed.) CVPR, pp. 2027–2034 (2013)
23. Vazquez-Reina, A., Avidan, S., Pfister, H., Miller, E.: Multiple hypothesis video segmentation from superpixel flows. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 268–281. Springer, Heidelberg (2010)
24. Ren, X., Malik, J.: Tracking as repeated figure/ground segmentation. In: Conference on Computer Vision and Pattern Recognition (CVPR). IEEE Computer Society (2007)
25. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **22**, 888–905 (1997)
26. Simari, P., Picciau, G., De Floriani, L.: Fast and scalable mesh superfacets. Computer Graphics Forum **33**(7), 181–190 (2014)
27. Sinha, S.N., Mordohai, P., Pollefeys, M.: Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh. In: ICCV, pp. 1–8. IEEE (2007)
28. Sondershaus, R., Straßer, W.: View-dependent tetrahedral meshing and rendering using arbitrary segments. Journal of WSCG **14**(1–3), 129–136 (2006)

29. Takahashi, S., Takeshima, Y., Fujishiro, I.: Topological volume skeletonization and its application to transfer function design. Graphical Models **66**(1), 24–49 (2004)
30. Vedaldi, A., Soatto, S.: Quick shift and kernel methods for mode seeking. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part IV. LNCS, vol. 5305, pp. 705–718. Springer, Heidelberg (2008)
31. 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), 583–598 (1991)
32. Wang, H., Yushkevitch, P.A.: Multi-atlas segmentation without registration: A supervoxel-based approach. Medical image computing and computer-assisted intervention **16**(3), 535–542 (2013)
33. Wang, P., Zeng, G., Gan, R., Wang, J., Zha, H.: Structure-Sensitive superpixels via geodesic distance. International Journal of Computer Vision **103**, 1–21 (2012)
34. Wang, S., Lu, H., Yang, F., Yang, M.H.: Superpixel tracking. In: Metaxas, D.N., Quan, L., Sanfeliu, A., Gool, L.J.V. (eds.) International Conference on Computer Vision (ICCV), pp. 1323–1330. IEEE (2011)
35. Weiss, K., Iuricich, F., Fellegara, R., De Floriani, L.: A primal/dual representation for discrete Morse complexes on tetrahedral meshes. In: Proceedings of the 15th Eurographics Conference on Visualization, vol. 32(3), pp. 361–370 (2013)
36. Xu, C., Xiong, C., Corso, J.J.: Streaming hierarchical video segmentation. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part VI. LNCS, vol. 7577, pp. 626–639. Springer, Heidelberg (2012)
37. Zhang, Y., Hartley, R.I., Mashford, J., Burn, S.: Superpixels via pseudo-boolean optimization. In: Metaxas, D.N., Quan, L., Sanfeliu, A., Gool, L.J.V. (eds.) International Conference on Computer Vision (ICCV), pp. 1387–1394. IEEE (2011)