Feature Sensitive Mesh Segmentation with Mean Shift

Hitoshi Yamauchi[†] Seungyong Lee[††] Yunjin Lee[††] Yutaka Ohtake[‡]
Alexander Belyaev[†] Hans-Peter Seidel[†]

[†]MPI Informatik, Saarbrücken, Germany
[††]Pohang University of Science & Technology, Pohang, Korea
[‡]Volume-CAD Development Team, RIKEN, Saitama, Japan

{hitoshi,belyaev,hpseidel}@mpi-sb.mpg.de, {leesy,jin}@postech.ac.kr, yu-ohtake@riken.jp

Abstract

Feature sensitive mesh segmentation is important for many computer graphics and geometric modeling applications. In this paper, we develop a mesh segmentation method which is capable of producing high-quality shape partitioning. It respects fine shape features and works well on various types of shapes, including natural shapes and mechanical parts. The method combines a procedure for clustering mesh normals with a modification of the mesh chartification technique in [23]. For clustering of mesh normals, we adopt Mean Shift, a powerful general purpose technique for clustering scattered data. We demonstrate advantages of our method by comparing it with two state-of-the-art mesh segmentation techniques.

1. Introduction

Similar to the role that image segmentation plays in image analysis and understanding, proper surface partitioning is one of the most important operations in geometric modeling and processing. Polygonal meshes remain a preferred representation for surface data because of their ability to approximate complex shapes, algorithmic simplicity, and visualization efficiency. In particular, triangle meshes are widely used in many engineering, medical, and entertainment applications. In this paper, we deal with segmentation of surfaces approximated by triangle meshes.

Mesh segmentation is an active research area and has numerous applications in texture atlas generation [14, 15, 23, 27], shape analysis, recognition, and matching [6,16,18,22], shape simplification [3, 9, 12], and shape modeling and retrieval [8, 13]. Mesh segmentation techniques can be classified to patch-type and part-type [25]. In patch-type segmentation, a mesh is partitioned into patches which are topological disks. Part-type segmentation divides a mesh into meaningful parts, such as a head and legs of a horse, without restricting the part topology. Mesh segmentation considered in this paper is patch-type and generates a chartification of a triangle mesh.

In this paper, we propose a simple but effective technique for improving existing mesh segmentation methods. The main idea behind our approach can be summarized as preliminary clustering of local geometric attributes associated with the surface approximated by a given mesh. We implement a simple variant of this general idea and combine clustering of mesh normals with a modification of the method developed in [23]. It allows us to achieve a mesh segmentation which is more sensitive to salient surface features, compared with the state-of-the-art methods of [3] and [23].

According to our numerical experiments, the Variational Shape Approximation method [3] (we refer to it as the VSA method) is extremely good for segmenting shapes composed of regions with simple geometry and sharp features (typically, models of mechanical parts). On the other hand, the mesh chartification technique in [23, Section 4.1] (we refer to it as the MCGIM method) is very capable of partitioning shapes with complex geometry (typically, natural shapes). It turns out that combining our procedure for clustering mesh normals with a slight modification of the MCGIM method results in a powerful mesh segmentation technique which delivers high-quality results for both types of shapes: natural shapes and mechanical parts.

We employ Mean Shift [2, 4, 7], a powerful general purpose procedure for non-parametric clustering of scattered data, for a practical implementation of the clustering part of our approach. Specifically, given a triangle mesh, we consider the triangle centroids equipped with triangle normals as a scattered set of points in six-dimensional space. Then Mean Shift is used for clustering the 6D points. Finally each triangle is equipped with a modified normal corresponding to the associated cluster center.

Previous geometric modeling application of mean shift clustering includes feature space analysis of unstructured

volume data [24]. The feature space consists of geometry and attributes of volume elements and the analysis with Mean Shift was used to reveal the internal structure of 3D scalar fields. The feature space analysis was later extended to handle polygon meshes [26], where local parameterization is used to implement the Mean Shift operation. In this paper, we demonstrate that triangle normals can be successfully clustered directly using triangle centroids as 3D points without local parameterization.

Recent studies [3, 11, 20] show that exploring geometric information contained in the surface normal field is a key ingredient for feature sensitive shape processing. Our idea of using clustered mesh normals can be considered as a further development of a mollification technique of [10], employed as a preprocessing step for feature preserving mesh smoothing and based on Gaussian smoothing of mesh normals. Due to its solid statistical foundation, Mean Shift can generate more robust filtering of noisy mesh normals than Gaussian smoothing. Moreover, mean shift filtering has an advantage that we do not need to care about both dimensionality of data set and mesh connectivity.

2. Mesh Normal Clustering with Mean Shift

Given a set of points (samples) $\mathcal{X} = \{\mathbf{x}_1, \dots \mathbf{x}_N\} \subset \mathbb{R}^d$ drawn from a probability density $f(\mathbf{x})$, we want to estimate the probability density function $f(\cdot)$ at point $\mathbf{x}$. The kernel density estimation method (Parzen-window estimation) estimates the probability density function by

$$\hat{f}(\mathbf{x}) = \frac{1}{Nh^d} \sum_{i=1}^{N} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right), \qquad (1)$$

where $K(\mathbf{x})$ is the so-called kernel function satisfying

$$K(\mathbf{x}) \geq 0 \quad \text{and} \quad \int_{\mathbb{R}^d} K(\mathbf{x})\, d\mathbf{x} = 1$$

and h is a smoothing parameter called the bandwidth. Figure 1 shows an example of a reconstruction of a one-dimensional probability density function using a Gaussian-like kernel $K(\mathbf{x})$.

Assume now that we are interested in subdividing scattered data $\mathcal{X}$ into a set of clusters. It is natural to consider the points where $\hat{f}(\mathbf{x})$ defined by (1) has local maxima as centers of the clusters. The simplest method to find the local maxima of (1) is to use a gradient-ascent process.

Assume that the kernel function $K(\mathbf{x})$ is rotationally invariant

$$K(\mathbf{x}) = c \cdot k\left(\|\mathbf{x}\|^2\right),$$

where $k(x)$ is called the profile function and c is a normalizing constant

$$\int_{\mathbb{R}^d} k\left(\|\mathbf{x}\|^2\right) d\mathbf{x} = \frac{1}{c}.$$

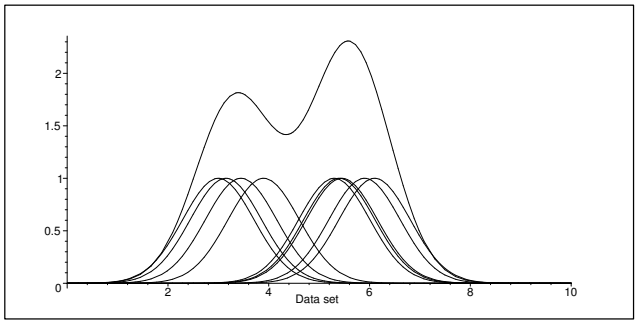


Figure 1. Probability density function is estimated from scattered data as the sum of kernel functions associated with each point of the data.

It is easy to see that

$$\nabla \hat{f}(\mathbf{x}) = \frac{2c}{Nh^{d+2}} \sum_{i=1}^{N} g\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right) \mathbf{m}(\mathbf{x}),$$

where $g(x) = -k'(x)$. The so-called mean shift vector $\mathbf{m}(\mathbf{x})$ is given by

$$\mathbf{m}(\mathbf{x}) = \frac{\sum_{i=1}^{N} \mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{N} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)} - \mathbf{x}.$$

A gradient-ascent process with an adaptive step size

$$\mathbf{y}^{[j+1]} = \mathbf{y}^{[j]} + \mathbf{m}(\mathbf{y}^{[j]}), \quad j = 0, 1, 2, \dots \qquad (2)$$

constitutes the core of the mean shift clustering procedure. For clustering $\mathcal{X} = \{\mathbf{x}_1, \dots \mathbf{x}_N\}$ with Mean Shift, the following two steps are performed on each $\mathbf{x}_i \in \mathcal{X}$:

1. Initialize $\mathbf{y}_i^{[0]}$ with $\mathbf{x}_i$;
2. Compute $\mathbf{y}_i^{[j]}$ according to (2) until convergence.

It can be shown in [4] that under some general assumptions the sequences $\left\{\mathbf{y}_i^{[j]}\right\}$ converge to the points where $\hat{f}(\mathbf{x})$ defined by (1) attains its local maxima.

The above clustering procedure can be generalized in a number of ways [5, 19, 24, 26, 30, 31]. One simple extension consists of dealing with a set $\mathcal{X}$, each element of which has two components of different nature,

$$\mathcal{X} = \{\mathbf{x}_i = (\mathbf{p}_i, \mathbf{q}_i) \,:\, \mathbf{p}_i \in \mathcal{P},\ \mathbf{q}_i \in \mathcal{Q}\}.$$

In such a situation, it is convenient to use the mean shift clustering procedure with separable kernels

$$\hat{f}(\mathbf{x}) = \frac{1}{Nh_1^{d_1}h_2^{d_2}} \sum_{i=1}^{N} K_1\left(\frac{\mathbf{p} - \mathbf{p}_i}{h_1}\right) K_2\left(\frac{\mathbf{q} - \mathbf{q}_i}{h_2}\right).$$

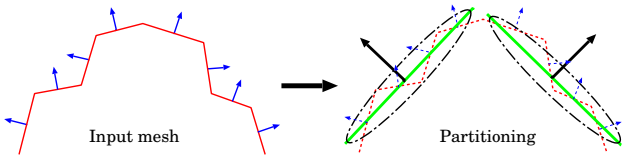


Figure 2. Geometric idea behind the use of clustering of mesh normals for mesh partitioning purpose.

In this paper, we employ the mean shift procedure for clustering mesh normals. Given a triangle mesh $\mathcal{M}$ consisting of triangles $\{T_i\}$, we consider the triangle centroids $\{\mathbf{c}_i\}$ equipped with triangle normals $\{\mathbf{n}_i\}$ as scattered data $\mathcal{X} = \{(\mathbf{p}_i, \mathbf{q}_i) = (\mathbf{c}_i, \mathbf{n}_i)\}$ in $\mathbb{R}^6$. For each cluster center $\mathbf{y}_i^{[c]} = (\mathbf{p}, \mathbf{q})$, we normalize its 'mesh normal' component $\mathbf{q}$ and assign it to all the mesh triangles T corresponding to the cluster.

Figure 2 demonstrates our idea of using clustering of normals for mesh partitioning. For the sake of readability and consistency, in the rest of the paper we use the notations h_g and h_n instead of h_1 and h_2, respectively.

In our current implementation, for both the triangle centroids $\mathbf{p}_i = \mathbf{c}_i$ and triangle normals $\mathbf{q}_i = \mathbf{n}_i$ we use the so-called Epanechnikov kernel which has a simple profile

$$k_E(x) = \begin{cases} 1 - x & 0 \leq x < 1 \\ 0 & x > 1. \end{cases}$$

To accelerate the mean shift computation, we store the geometry information (3D locations of $\{\mathbf{c}_i\}$) using a k-D tree. The point query process consists of the following two steps.

1. Given a query $(\mathbf{p}, \mathbf{q})$, we detect all points $\mathbf{p}_i$ such that $\|\mathbf{p}_i - \mathbf{p}\| < h_g$.
2. For each detected point $\mathbf{p}_i$, we check whether its 'normal' counterpart $\mathbf{q}_i$ satisfies $\|\mathbf{q}_i - \mathbf{q}\| < h_n$.

In our experiments, we have observed that our Mean Shift based clustering of mesh normals can also be used for adaptive mesh smoothing. Once the modified (clustered) normals are obtained, a new mesh can be reconstructed from the modified normals, as described in [21] (see also [28, 29] for similar approaches). As shown in Figure 3, varying the bandwidth parameter h_n used in the kernel for mesh normals leads to adaptive mesh smoothing with simultaneous sharpening of surface creases at different geometric scales. The parameter h_n defines which normals are involved for averaging in the mean shift procedure. On the other hand, altering h_g affects how many neighboring mesh vertices are considered in the mean shift procedure. Choosing a larger value for h_g will lead to smooth out smaller mesh features.

3. Iterated Region Growing Segmentation

Once the mesh normals are prefiltered with Mean Shift, we follow a general approach proposed in [23] and inspired by the iterative Lloyd quantization method [17].

In initialization, the number of seeds starts from one and a seed triangle is selected randomly. First, *Phase 1* below is performed. If the number of seeds has not reached the user specified number of charts, the farthest triangle is marked as a new seed. Otherwise, no new seed is added. Second, *Phase 2* below is used to find a center seed for each chart. This combination of *Phase 1* and *Phase 2* procedures is repeated until all the seed positions are converged or the number of iterations exceeds a user defined threshold.

Phase 1: Chart growth. The distance between two adjacent triangle faces, T_i and T_j, is defined by

$$\text{distance}(T_i, T_j) = \|\mathbf{N}_{\mathbf{m}i} - \mathbf{N}_{\mathbf{m}j}\|, \tag{3}$$

where $\mathbf{N}_{\mathbf{m}}$ is the triangle normal obtained by mean shift filtering. By performing the multi-source Dijkstra's shortest path algorithm with this distance, we can determine the closest seed for each triangle.

In [23], a different definition of the distance is given by

$$\text{distance}(T_i, T_j) = (1 - (\mathbf{N}_r \cdot \mathbf{N}_j))(\|\mathbf{c}_i - \mathbf{c}_j\|), \tag{4}$$

where T_i is a triangle already included in a chart and T_j is a candidate triangle to be added to the chart. $\mathbf{N}_r$ is the representative normal of the chart, which is the average of normals of the current chart triangles. $\mathbf{N}_j$ is the normal of triangle T_j. $\mathbf{c}_i$ and $\mathbf{c}_j$ are the centroids of T_i and T_j, respectively.

Since the representative normal $\mathbf{N}_r$ is the average of current chart triangles, Equation (4) does not fully respect features of a mesh. On the other hand, $\mathbf{N}_r$ introduces the stability of the chart growing algorithm. If the distance is simply defined with the normal deviation of two adjacent triangles, the effect of noise may be emphasized, e.g., on a bumpy surface.

In contrast, the normals used in Equation (3) are the results of the mean shift filtering. As demonstrated in Figure 3, mean shift filtering removes noises in triangle normals while preserving the features. Hence, the normals used in Equation (3) can be considered reliable and we do not need a stabilizing term such as in Equation (4). As a result, the distance definition in Equation (3) is more feature-sensitive than Equation (4) without introducing instability.

Note that we chose $\|\mathbf{N}_{\mathbf{m}i} - \mathbf{N}_{\mathbf{m}j}\|$ in Equation (3) instead of usually used $\|\mathbf{N}_{\mathbf{m}i} - \mathbf{N}_{\mathbf{m}j}\|^2$. Simple calculation shows that $\|\mathbf{N}_{\mathbf{m}i} - \mathbf{N}_{\mathbf{m}j}\|^2 = 2(1 - (\mathbf{N}_{\mathbf{m}i} \cdot \mathbf{N}_{\mathbf{m}j}))$. With this choice, the distance in Equation (3) is made more sensitive to the deviation of normals, especially when two normals are very similar.

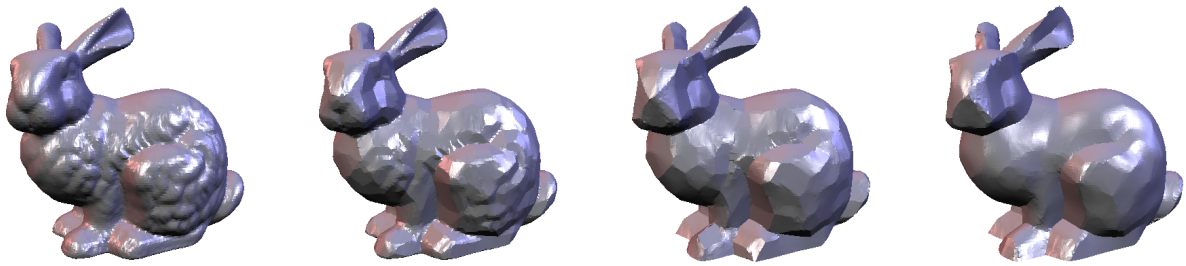


Figure 3. Left: original Stanford bunny. The other three images demonstrate that different kernel bandwidths for mesh normals result in adaptive mesh smoothing at different geometric scales. The kernel bandwidth increases from left to right. ($h_g = 0.03$ in all three images. $h_n = 0.2, 0.5$, and 1.0, respectively.)

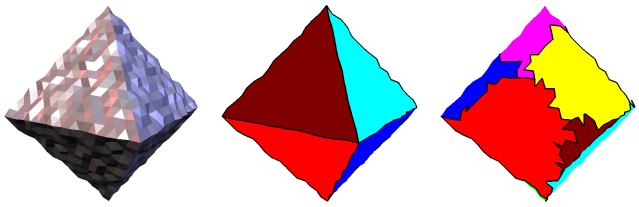


Figure 4. Mean shift effect for our segmentation method. Left: noisy octahedron. Middle: segmentation with our approach. Perfect segmentation of the noisy octahedron is achieved. Right: our approach without mean shift filtering.

Phase 2: Seed recomputation. Once chart growing has been completed, the seed is updated for each chart. The relocated seed should be the most interior triangle from the chart boundary. The distance among adjacent triangles in this phase is approximated by the geodesic distance;

$$\text{distance}(T_i, T_j) = \|\mathbf{c}_i - \mathbf{c}_j\|, \tag{5}$$

where $\mathbf{c}_i$ is the centroid of triangle T_i. The same distance is used in [23] for the seed recomputation.

Figure 4 shows the segmentation result of our approach applied to a noisy octahedron model. Our segmentation procedure delivers the perfect segmentation of the model. However, without mean shift filtering, our approach is too sensitive to noise. The parameters used for Figure 4 are $(h_g, h_n, N_c) = (0.3, 0.5, 8)$, where h_g is the bandwidth for geometry (1.0 is the maximum edge length of the input model's bounding box), h_n is the bandwidth for normals, and N_c is the number of charts.

In [23] it was mentioned that their discrete optimization procedure sometimes fails to converge because of cycles. This could also happen in our case. Our implementation includes the cycle detection and user specified maximum number of iterations. Therefore, the iteration process will be stopped if a cycle is detected for seed positions or the number of iterations is too large.

Model	# of Tris.	MCGIM	VSA	MS + growth
Mannequin	21,680	67.5	32.1	11.4 + 90.4
Fandisk	12,944	31.2	12.7	0.96 + 26.6
Cow	23,216	185	22.9	0.95 + 225
Camel	78,144	1659	256	11.5 + 1598

Table 1. Elapsed time (sec.) of segmentation processing (see Figure 6). MS+growth stands for the elapsed times for mean shift clustering and chart growth procedures.

4. Experimental Results and Discussion

Figure 6 shows the comparison of our method with [3] and [23]. We implement the method in [3] with the $L^{2,1}$ metric and region teleportation. We usually observed better segmentation results with region teleportation. We experimentally chose 0.01 for the threshold of cluster merging in region teleportation. The same number of charts (N_c) was used for segmenting a model with different methods. The number of triangles of each model in Figure 6 and the processing time are shown in Table 1. The feature enhancing effects of Mean Shift are observed in many cases. For example, some models have eyes and they are separated in our approach, whereas other methods do not successfully capture them.

The VSA method [3] is originally proposed as a shape approximation method but attacks the approximation problem by way of surface segmentation. The method produces

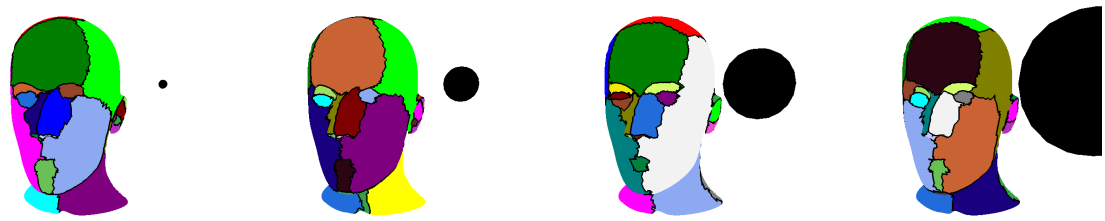


Figure 5. Influence of bandwidth h_g (for the sake of visualization, black disks whose radii are proportional to h_g are shown). From left to right, geometry bandwidth sizes are 0.0125, 0.05, 0.1, and 0.2. The bandwidth h_n is set to 0.3 and the number of charts is 30 in all cases. Notice how different values of bandwidth h_g affect separation of the eyebrows and mouth.

quite good segmentation results, especially for mechanical objects, because it obtains a piecewise linear approximation of a shape. However, some features are missing on non-mechanical objects compared with our method, as demonstrated in Figure 6. The method uses the representative normal calculated by averaging all normals belonging to a chart. This approach is very successful in detecting anisotropy in a surface shape but has a smoothing effect that makes the segmentation less sensitive to small features.

Figure 5 shows the effects of the bandwidth size in our approach. We fix the normal bandwidth (h_n) to 0.3 and alter the geometry bandwidth, which is shown as a black disk in the figure. We observed the optimal value stays around 0.1 for this model (the third image from the left). With this parameter value, we can successfully separate both the eyebrows and the mouth. Basically, the choice of h_g affects the detected feature size of a mesh and the choice of h_n determines the feature angle to be detected. In Figure 5, $h_n = 0.3$ approximately corresponds to 20 degrees of the dihedral angle.

According to our numerical experiments, selecting proper values for the bandwidths h_g and h_n is rather simple. Although non-linear diffusion effect may generate unexpected results, in general, we observe that the segmentation results rather intuitively depend on h_g and h_n in our method. Small changes of the parameter values usually have not much affect on the segmentation result.

Currently, no optimization has been carried out in our implementation. All timings were measured on a 1.7 GHz Pentium 4 PC. The time to segment a mesh heavily depends on the number of triangles and the bandwidth size. Our algorithm took about 30 seconds to 25 minutes to complete the segmentation. Table 1 shows the details of the processing time.

One problematic case for our approach is that the surface has no features. If a surface is very smooth, the distance function in Equation (3) is close to zero in all directions. One of the pathetic cases is a plane. In this case, the triangle normals are the same everywhere, and there is no distance to grow. For preventing this phenomenon, the geodesic distance would help but may smooth out the feature-based distance in some cases. Figure 6(b.1) shows the effect of a too strong geodesic distance, where we can observe some charts cross over the features. A weighted sum of geodesic and normal deviation terms seems a good idea but introduces one extra parameter. The same problem also occurs in [3], because they use only normal deviation for the $L^{2,1}$ metric. Their region teleportation method resolves this problem. For our implementation, we add small $\epsilon(> 0)$ to Equation (3), where we use $\epsilon = 1.0 \times 10^{-5}$ determined by experiments.

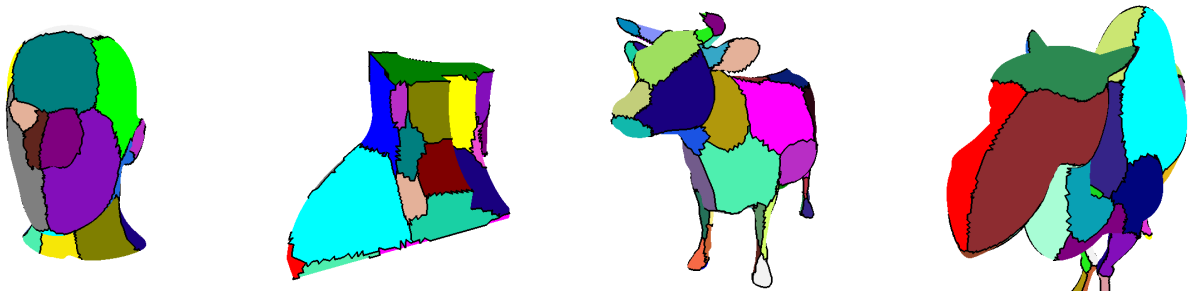
5. Conclusion and Future Work

We have proposed a simple scheme for clustering mesh normals and demonstrated its efficiency for generating feature sensitive mesh segmentation. According to our experiments, the best results are obtained when clustering of mesh normals is combined with a modification of the mesh chartification approach proposed in [23].

The mean shift clustering is based on the Parzen-window estimation and well suited for dealing with complex, noisy, and uncertain data sets. This property helps a lot to robustly process unstable normals resulting from erroneous normal estimation for meshes with complex shapes.

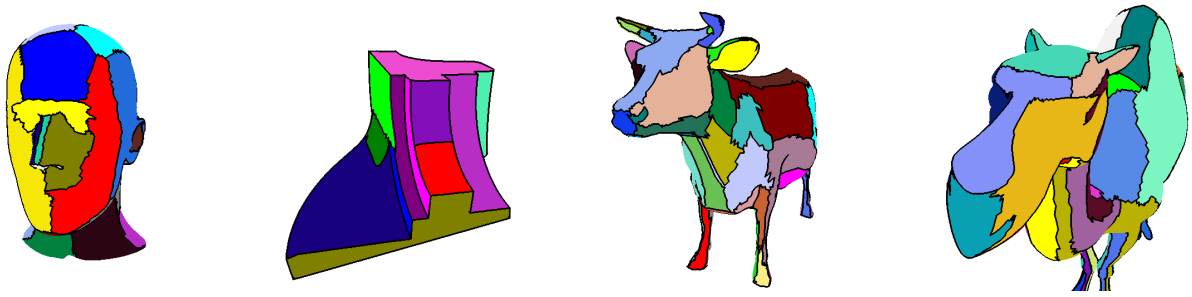
We have found that determining appropriate bandwidth parameters h_g and h_n to generate feature sensitive mesh partitioning with Mean Shift is rather simple. Roughly speaking, one recommended value for h_g is two to five times the average edge length of a mesh. At least, h_g should be large enough to contain some neighbor vertices for exploiting mean shift filtering effects. In future, we plan to develop a procedure for adaptive selection of the bandwidth parameters. We also think that the number of charts can be

MCGIM [23]



(a.1) (b.1) (c.1) (d.1)

VSA [3]



(a.2) (b.2) (c.2) (d.2)

our approach

(a.3) (0.1, 0.3, 30) (b.3) (0.05, 0.3, 20) (c.3) (0.015, 0.3, 90) (d.3) (0.015, 0.3, 80)

Figure 6. Results of feature sensitive mesh segmentation. The list of three numbers (h_g, h_n, N_c) represents our parameter set. h_g is the geometry bandwidth size, which is relative to the maximum edge length of the input model's bounding box. h_n is the normal bandwidth size. N_c is the number of charts. The same number of charts (N_c) was used for the results of MCGIM [23] and VSA [3].

selected automatically according to some statistical criteria. See [1] for relevant ideas.

The mean shift clustering does not allow a user to prescribe the number of clusters explicitly. Thus, if we want to have the number of charts as a user-specified parameter, k-means least-squares partitioning seems preferred in comparison with Mean Shift. However k-means clustering will reduce the ability to segment small-sized surface features.

Another idea that intrigues us in this field is to use high-order surface descriptors (curvature, shape index, etc.) for mesh partitioning with a richer set of primitives (e.g., planar, cylindrical, conical, and spherical parts).

Acknowledgments

We thank the anonymous reviewers for their thoughtful and constructive suggestions. The research of the MPI Informatik authors was supported in part by European FP6 NoE grant 506766 (AIM@SHAPE). The authors at POSTECH were supported in part by the Korean Ministry of Education through the BK21 program and the Korean Ministry of Information and Communication through the

ITRC support program.

References

[1] H. Bischof, A. Leonardis, and A. Selb. MDL principle for robust vector quantization. *Pattern Analysis and Applications*, 2(1):59–72, 1999.

[2] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:790–799, 1995.

[3] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. *ACM Transactions on Graphics*, 23(3):905–914, August 2004. Proceedings of SIGGRAPH 2004.

[4] D. Comaniciu and P. Meer. Mean Shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002.

[5] D. Comaniciu, V. Ramesh, and P. Meer. The variable bandwidth mean shift and data-driven scale selection. In *Proc. International Conference on Computer Vision (ICCV'01), Volume 1*, pages 438–445, 2001.

[6] T. K. Dey, J. Giesen, and S. Goswami. Shape segmentation and matching with flow discretization. In *Proc. Workshop on Algorithms Data Structures (WADS 03), LNCS 2748*, pages 25–36, 2003.

[7] K. Fukunaga and L. D. Hostetler. The estimation of the gradient of a density function with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21:32–40, 1975.

[8] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin. Modeling by example. *ACM Transactions on Graphics*, 23(3):652–663, 2004. Proceedings of SIGGRAPH 2004.

[9] M. Garland, A. Willmott, and P. S. Heckbert. Hierarchical face clustering on polygonal surfaces. In *Proc. Symposium on Interactive 3D Graphics*, pages 49–58, 2001.

[10] T. R. Jones, F. Durand, and M. Desbrun. Non-iterative, feature-preserving mesh smoothing. *ACM Transactions on Graphics*, 22(3):943–949, 2003. Proceedings of SIGGRAPH 2003.

[11] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. *ACM Transactions on Graphics*, 21(3):339–346, July 2002. Proceedings of SIGGRAPH 2002.

[12] A. D. Kalvin and R. H. Taylor. Superfaces: Polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Applications*, 16(3):64–77, 1996.

[13] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics*, 22(3):954–961, 2003. Proceedings of SIGGRAPH 2003.

[14] A. W. F. Lee, W. Sweldens, P. Schröder, L. L. Cowsar, and D. Dobkin. MAPS: Multiresolution adaptive parameterization of surfaces. In *Proc. ACM SIGGRAPH 1998*, pages 95–104, 1998.

[15] B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics*, 21(3):362–371, 2002. Proceedings of SIGGRAPH 2002.

[16] R. Liu and H. Zhang. Segmentation of 3D meshes through spectral clustering. In *Proc. Pacific Graphics*, pages 298–305, 2004.

[17] S. P. Lloyd. Least square quantization in PCM. *IEEE Transaction on Information Theory*, 28(2):129–137, March 1982.

[18] A. P. Mangan and R. T. Whitaker. Partitioning 3D surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308–321, October 1999.

[19] P. Mrázek and A. Weickert, J.and Bruhn. On robust estimation and smoothing with spatial and tonal kernels. In R. Klette, R. Kozera, L. Noakes, and J. Weickert, editors, *Proc. Dagstuhl Seminar: Geometric Properties from Incomplete Data*, 2004.

[20] Y. Ohtake and A. G. Belyaev. Dual/primal mesh optimization for polygonized implicit surfaces. In *Proc. 7th ACM Symposium on Solid Modeling and Applications*, pages 171–178, June 2002.

[21] Y. Ohtake, A. G. Belyaev, and I. A. Bogaevski. Mesh regularization and adaptive smoothing. *Computer-Aided Design*, 33(11):789–800, 2001.

[22] D. L. Page, A. Koschan, and M. Abidi. Perception-based 3d triangle mesh segmentation using fast marching watersheds. In *Proc. Intl. Conf. on Computer Vision and Pattern Recognition, Vol. II*, pages 27–32, 2003.

[23] P. V. Sander, Z. J. Wood, S. J. Gortler, J. Snyder, and H. Hoppe. Multi-chart geometry images. In *Proc. Symposium on Geometry Processing*, pages 146–155, 2003.

[24] A. Shamir. Feature-space analysis of unstructured meshes. In *Proc. IEEE Visualization 2003*, pages 185–192, 2003.

[25] A. Shamir. A formulation of boundary mesh segmentation. In *Proc. 2nd International Symposium on 3D Data Processing, Visualization, and Transmission*, September 2004.

[26] A. Shamir, L. Shapira, D. Cohen-Or, and R. Goldenthal. Geodesic mean shift. In *Proc. 5th Korea Israel conference on Geometric Modeling and Computer Graphics*, pages 51–56, October 2004.

[27] O. Sorkine, D. Cohen-Or, R. Goldenthal, and D. Lischinski. Bounded-distortion piecewise mesh parameterization. In *Proc. IEEE Visualization 2002*, pages 355–362, 2002.

[28] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher. Geometric surface smoothing via anisotropic diffusion of normals. In *Proc. IEEE Visualization 2002*, pages 125–132, November 2002.

[29] G. Taubin. Linear anisotropic mesh filtering. IBM Research Report RC22213 (W0110-051), IBM, October 2001.

[30] J. Wang, B. Thiesson, Y. Xu, and M. Cohen. Image and video segmentation by anisotropic kernel mean shift. In *Proc. European Conference on Computer Vision (ECCV'04), Volume 2*, pages 238–249, 2004.

[31] J. Wang, Y. Xu, H.-Y. Shum, and M. F. Cohen. Video toning. *ACM Transactions on Graphics*, 23(3):574–583, August 2004. Proceedings of SIGGRAPH 2004.