A Greedy Merging Method for User-Steered Mesh Segmentation

Jong-Sung Ha
Department of Game and Contents
Woosuk University, Wanju, Korea

Young-Jin Park
Department of Information Industrial Engineering
Chungbuk National University, Cheongju, Korea

Kwan-Hee Yoo*
Department of Computer Education, and Department of Information Industrial Engineering
Chungbuk National University, Cheongju, Korea

ABSTRACT

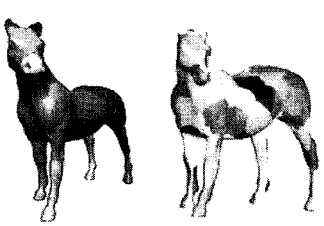
In this paper, we discuss the mesh segmentation problem which divides a given 3D mesh into several disjoint sets. To solve the problem, we propose a greedy method based on the merging priority metric defined for representing the geometric properties of meaningful parts. The proposed priority metric is a weighted function using five geometric parameters, those are, a distribution of Gaussian map, boundary path concavity, boundary path length, cardinality, and segmentation resolution. In special, we can control by setting up the weight values of the proposed geometric parameters to obtain visually better mesh segmentation. Finally, we carry out an experiment on several 3D mesh models using the proposed methods and visualize the results.

Keywords: *Mesh Segmentation, Greedy Merging, Feature, User-Steered*

1. INTRODUCTION

In general, mesh segmentation is formulated as an optimization problem that divides a given mesh into disjoint parts satisfying some criteria to the maximum. This is equivalent to a graph partitioning problem if the faces of the mesh and their adjacency relations are represented with the nodes and the edges of a graph respectively. This optimization is an NP-class problem; at present, any known algorithm requires time that is super-polynomial in the input size. Furthermore, the mesh segmentation is also input-sensitive; the efficacy of a segmentation method varies from artifacts to natural objects. By those reasons, most mesh segmentation techniques have been developed for a specific application such as mesh modeling, metamorphosis, compression, simplification, shape matching, collision detection, texture mapping, and skeleton extraction.

Mesh segmentation methods can be categorized into the bottom-up approaches [1-3] that merge the faces of a mesh increasingly, and the top-down approaches [4-6] that divide the mesh continuously. Other approaches have been developed: the topological approach [7], the physics-based approach [8], error metric-based approach [9], and so on. Each segmentation method is aimed at generating mesh parts in meaningful shapes: feature-type and disc-type as illustrated in Fig. 1(a) and (b) respectively. Many segmentation methods have been presented for the disc-type, and recently the feature-type-oriented approaches have gained a great interest because they are more natural in mesh applications such as re-meshing and shape matching.



(a) feature-type (b) disc-type

Fig. 1: Two types of segmented parts

Corresponding author. E-mail : khyoo@chungbuk.ac.kr
Manuscript received May 30, 2007 ; accepted June 18, 2007

This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD) (KRF-2006-D00413).

It is, however, difficult to formally define the meaningful parts of a mesh in a quantitative manner for evaluating the different segmentation methods[10]. This paper defines several geometric properties that can be measured and develop a user-steered mesh segmentation method, in which users can interactively control the part shapes. Our method exploits a greedy technique that merges adjacent parts increasingly by finding a pair with the maximum metric value of the defined geometric properties, which will be described in Section 2 and 3 concretely. In order to preserve the geometric shape as possible, the convex patches composed of a set of faces are, instead of the faces, used as the smallest units to be merged. Our experiments in Section 4 show that users can extract the well-defined parts of feature-types from a mesh by empirically setting up the number size of parts and the weights of the defined geometric parameters.

2. MERGING POLICY OF CONVEX PATCHES

A set of faces in a mesh is called the convex patch if they are connected and all of them exit on their convex hull. Chazelle *et al.* [11] proved that it is a NP-complete problem to compute the minimum number of disjoint convex patches for a given mesh. These convex patches are called exact convex patches (ECP) against approximate convex patches (ACP) that can be defined for efficient and practical computing in [12]. We can permit a tolerance angle in checking the convexity of an edge for allowing that a mesh is usually an approximation of a curved surface, and select one of these two types of convex patches:

① ECP: Chazelle *et al.* [11] have categorized the methods for computing ECP into three: space partitioning, space sweeping, and flooding. We take a simple flooding technique that visits the faces of a mesh in breadth-first-search (BFS), which can construct ECP in a disc shape if possible.

② ACP: In this paper, a set of faces that are connected via convex edges is defined as an ACP. Our ACP can include some concave edges but there is at least a convex edge between every pair of adjacent faces. This ACP is always determinant, and can be computed easily with the graph searching technique of BFS and depth-first-search (DFS).

For the greedy merging of parts, we define a priority metric as a weighted function of some geometric parameters controlling the shape of merged parts according to user-steering. Since we define the following geometric parameters mainly from the viewpoint of the feature-type parts, other geometric parameters can be added without any modification of the scheme of our method.

① Distribution of Gaussian Map: The Gaussian mapping is the movement of each normal vector of faces to a point on a unit sphere. The flatter a part is, the smaller the region of its Gaussian map is. This distribution degree can be checked by computing the spherical convex hulls of the Gaussian maps, or the smallest spheres enclosing the Gaussian maps. Our method takes the radii of the smallest enclosing spheres as a geometric parameter. A great weight for this parameter is useful for extracting feature-type parts.

② Boundary Path Concavity: The boundary path between a pair of parts is the consecutive edges shared by the pair. The concavity between a pair of parts is measured by averaging the angles of concave edges in the boundary path. Since the inner product of face normals is used for measuring the angles, a great weight for this parameter is also useful for extracting feature-type parts.

③ Boundary Path Length: The boundary path length is the number of edges, which is related with the coherence of faces within a part. The boundary path length of a part with a zigzag patterned boundary is bigger than that of a disc-type part. Hence, disc-type parts can be extracted by setting up a great weight for this parameter and small weights for the above two parameters.

④ Cardinality: If the number of faces is less than a threshold, the part is not meaningful even though it has other strong geometric properties. The weight for this parameter defines the values of this threshold.

⑤ Segmentation Resolution: The merging iteration can be stopped when the merged parts satisfy a given criterion or a given number of parts to be obtained.

Finally, by considering the above file geometric parameters, we defined a priority metric Mij for a pair of adjacent two parts Pi and Pj as Eqn. (1).

$$Mij = w_g \cdot (2 - g_{ij} - \frac{g_i}{2} - \frac{g_j}{2}) + w_e \cdot e_i + w_l \cdot (\frac{l_{ij}}{L_i} + \frac{l_{ij}}{L_j}) + w_f \cdot (2 - \frac{f_i + f_j}{F \cdot S}) \quad \text{(Eqn. 1)}$$

Each symbol in Eqn. (1) is defined as the follows.

g_i : the radius of the smallest sphere enclosing the Gaussian map of P_i

g_{ij} : the radius of the smallest sphere enclosing the Gaussian map of P_i and P_j

e_{ij} : the average of the inner product of normals of two faces sharing a concave edge in the boundary path between P_i and P_j

l_{ij} : the boundary path length between P_i and P_j

f_i : the cardinality of P_i

w_m : the weight of each parameter m

L_i : the total boundary path length of P_i

F : the total number of faces of a mesh

S : the number of parts to be obtained

3. A GREEDY MERGING ALGORITHM

Our greedy algorithm for merging the mesh parts is developed by using a set of tuples and a priority queue for inserting active tuples and deleting dead tuples. A tuple includes a pair of adjacent parts, its merging priority and other information. Firstly, for each pair of adjacent convex patches, we create a tuple and insert it into the priority queue. And then, a loop iterates until the stopping condition is satisfied; delete an active tuple with the maximum metric value from the priority queue, merge the two parts in the deleted tuple, and updates or removes the tuples including the merged parts. We can describe our algorithm with a pseudo code as.

procedure GREEDY_MERGER
input: a set of convex patches $\{CP1, \cdots, CPn\}$
output: a set of merged parts $\{P1, \cdots, Ps\}$
begin
  Initialize the tuple priority queue *PQueue*;
  foreach a pair of adjacent parts P_i and P_j **do**
    PQueue.push (CREATE_TUPLE(i, j));
  end foreach
  loop
    $TP(P_i, P_j, M_{ij}) \leftarrow$ *PQueue.pop* ();
    Merge the parts P_i and P_j into P_i;
    Initialize a temporary tuple priority queue *tmpPQueue*;
    while *PQueue* is not empty **do**
      $TP(P_a, P_b, M_{ab}) \leftarrow$ *PQueue.pop* ();
      if $a = i$ **or** $b = i$ **then**
        tmpPQueue.push (CREATE_TUPLE(a, b));
      else if $a = j$ **or** $b = j$ **then**
        if $a = j$ **then** $k = b$ **else** $k \leftarrow a$;
        if P_i was not adjacent with P_j **then**
          tmpPQueue.push (CREATE_TUPLE(i, k));
        end if
      else
        tmpPQueue.push ($TP(P_a, P_b, M_{ab})$);
      end if
    end while
    PQueue $\leftarrow$ *tmpPQueue*;
  until a given criterion is satisfied
end procedure
function CREATE_TUPLE(a, b)
begin
  Compute the priority metric M_{ab} of Eqn. (1) for a pair of adjacent two parts P_a and P_b;
  Create a tuple $TP(P_a, P_b, M_{ab})$;
  return $TP(P_a, P_b, M_{ab})$;
end function

4. IMPLEMENTATION AND EXPERIMENTS

Our greedy merging method has been implemented by using CGAL (computational geometry algorithm library) [13] in the environments of Microsoft Windows and Visual Studio .NET 2003. Mesh data for testing our implementation were obtained almost from the Princeton shape benchmark [14], but only the 3D models of half-edge data structures with one component were tested. The models with many components are more clearly segmented.

Fig. 2 illustrates the results of our user-steered mesh segmentation that can control the part shapes by setting up parametric weights. The number of parts to be obtained is varied for a model in Fig. 2(a)~(e), while the parametric weights are varied for a model in (e) and (f). Usually, the geometric parameters related with the Gaussian map and boundary path concavity is used for the feature-typed segmentation as illustrated in Fig 2.(d), (e), and (g)~(i), but the parametric weights can be empirically saturated as illustrated in (f) and (j)~(l) for a better feature-type. Tab 1. shows the parametric values that were set up interactively through a user interface for the models of Fig 2.

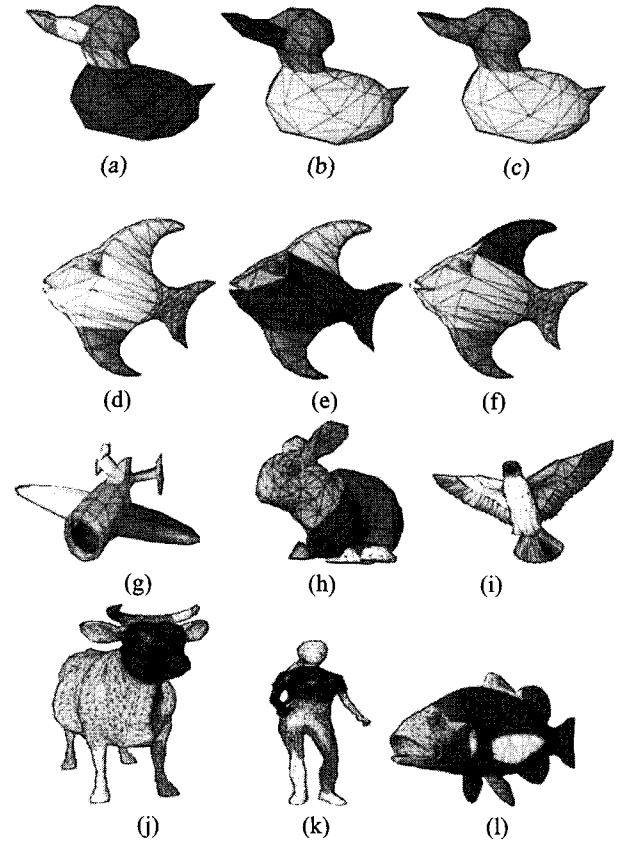


Fig. 2: Examples for greedy merging of convex patches

Table 1: Parametric values for the models of Fig 2.

Parameters	*(a)*	*(b)*	*(c)*	*(d)*	*(e)*	*(f)*
F	182			478		
S	7	3	2	7	4	
wg	100			100	100	20
we	100			100	100	20
wl	20			20	20	100
wf	10			10	10	10

Parameters	*(g)*	*(h)*	*(i)*	*(j)*	*(k)*	*(l)*
F	160	902	1130	5804	2500	3208
S	6	7	7	10	8	10
wg	100	100	100	100	10	50
we	100	100	100	100	100	50
wl	20	20	20	100	100	20
wf	10	10	10	100	10	20

5. CONCLUSION AND FUTURE RESEARCHES

Our greedy method for merging convex patches were motivated from the application of partial shape matching that compares the geometric similarities of mesh parts, where the feature-type segmentation is preprocessed. The greedy algorithm is based on the merging priority metric defined with five geometric properties: segmentation resolution, distribution of Gaussian map, boundary path concavity, boundary path length, and cardinality. From the experiments of our implementation, we could get the feature-type parts by weighting these geometric parameters appropriately. Its efficacy can be potentially enhanced without any modification of the scheme by developing other formal parameters for defining the geometric properties of meaningful parts. For example, the extreme points of convex hulls of models or parts can be used for defining a new parameter of the merging priority metric in order to reflect more global geometric properties.

Currently, the parametric weights are intuitively set up and saturated by try-and-error for the weighting function of merging priority metric. These manual operations seem to be a little necessary for user-steered mesh segmentation since almost mesh segmentation methods are input-sensitive. But it may take much time for obtaining usable weight values. In future, it is needed to automatically analyze the statistics of the defined geometric properties for a given mesh in order to provide information to users or automatically determine the usable weight values. It is also needed to post-process the segmented parts for obtaining more natural boundary shapes, since the convex patch as the merging unit causes more zigzag patterns than the face unit even though it can preserve the convex properties well.

REFERENCES

[1] T. Srinak, C. Kambhamettu, "A Novel Method for 3D Surface Mesh Segmentation," **Proceedings of the 6th IASTED International Conference on Computers, Graphics, and Imaging**, 2003, pp. 212-217.

[2] M. Garland, A. Willmott, P.S. Heckbert, "Hierarchical Face Clustering on Polygonal Surfaces," **Proceedings of the 2001 Symposium on Interactive 3D Graphics**, 2001, pp. 49-58.

[3] D.L. Page, A.F. Koschan, M.A. Abidi, "Perception-based 3D Triangle Mesh Segmentation using Fast Matching Watersheds," **Conference on Computer Vision and Pattern Recognition,** 2003, pp. 27-32.

[4] S. Katz, A. Tal, "Hierarchical Mesh Decomposition using Fuzzy Clustering and Cuts, **ACM Transactions on Graphics**, Vol. 22, No. 3, 2003, pp. 954-961.

[5] T. Kanungo, D.M. Mount, N.S. Netanyahu, A.D. Piatko, R. Silverman, A.Y. Wu, "An Efficient k-Means Clustering Algorithm: Analysis and Implementation," **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 2002, pp. 881-892.

[6] A.P. Mangan, R.T. Whitaker, "Partitioning 3D Surface Meshes using Watershed Segmentation," **IEEE Transactions on Visualization and Computer Graphics**, 1999, pp.308-321.

[7] T.K. Dey, J. Giesen, S. Goswami, "Shape Segmentation and Matching with Flow Discretization," **Lecture Notes in Computer Science 2748,** 2003, pp. 25-36.

[8] K. Wu, M.D. Levine, "3D Part Segmentation using Simulated Electrical Charge Distributions," **IEEE International Conference on Pattern Recognition**, 1996, pp. 14-18.

[9] D. Cohen-Steiner, P. Alliez, M. Desbrun, "Variational Shape Approximation," **Proceedings of the 31st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)**, 2004, pp. 27-34.

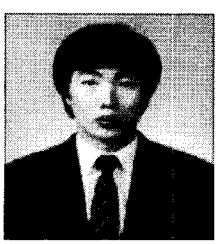
[10] M. Attene, S. Katz, M. Mortara, G. Patane, M. Spagnuolo, A. Tal, "Mesh Segmentation - A Comparative Study," **IEEE Int. Conf. on Shape Modeling and Applications**, 2006, pp. 14-25.

[11] B. Chazelle, *et al.*: "Strategies for Polyhedral Surface Decomposition: An Experimental Study," **Computational Geometry: Theory and Applications**, Vol. 7, 1997, pp. 327-342.

[12] J.M. Lien, N. M. Amato, "Approximate Convex Decomposition of Polyhedra," **Technical Report TR06-002, Dept. of Computer Science, Texas A&M University**, 2006.

[13] Computational Geometry Algorithm Library, http://www.cgal.org

[14] P. Shilane, M. Kazhdan, P. Min, T. Funkhouser, "The Princeton Shape Benchmark," **In Proc Shape Modeling International**, Genoa, Italy, 2004.



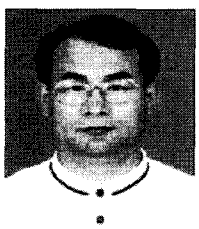
Jong-Sung Ha

He is a professor of game and contents at Woosuk University, Korea. He received the B.E. in computer engineering from Seoul National University, Korea in 1984, and also received M.S., Ph.D. in computer science from KAIST(Korea Advanced Institute of Science and Technology), Korea in 1986 and 1996,

respectively. His main research interests include computational geometry, computer graphics, and CAD/CAM. He is particularly interested in robust geometric algorithms that are practical in application fields.

Young-Jin Park

He is a Ph.D. candidate of information industrial engineering at Chungbuk National University, Korea. He received the B.S., M.S. in computer science from Chonbuk National University, Korea in 1986 and 1988, respectively. His research interests include computational geometry and computer graphics.



Kwan-Hee Yoo

He is a professor of computer education and IIE(information industrial engineering) at Chungbuk National University, Korea. He received the B.S. in computer science from Chonbuk National University, Korea in 1985, and also received M.S., Ph.D. in computer science from KAIST(Korea Advanced Institute of Science and Technology), Korea in 1988 and 1995, respectively. His research interests include computational geometry, computer graphics, 3D character animation, dental/medical applications.