Scale-Space Processing of Point-Sampled Geometry for Efficient 3D Object Segmentation

Hamid Laga Hiroki Takahashi Masayuki Nakajima

Graduate School of Information Science and Engineering
Tokyo Institute of Technology
2-12-1 Ookayama, Meguro-ku, Tokyo, 152-8552 JAPAN
{hamid, rocky, nakajima}@img.cs.titech.ac.jp

Abstract

In this paper, we present a new framework for analyzing and segmenting point-sampled 3D objects. Our method first computes for each surface point the surface curvature distribution by applying the Principal Component Analysis on local neighborhoods with different sizes. Then we model in the four dimensional space the joint distribution of surface curvature and position features as a mixture of Gaussians using the Expectation Maximization algorithm. Central to our method is the extension of the scale-space theory from the 2D domain into the three-dimensional space to allow feature analysis and classification at different scales.

Our algorithm operates directly on points requiring no vertex connectivity information. We demonstrate and discuss the performance of our framework on a collection of point sampled 3D objects.

Keywords: *Scale-space, 3D object segmentation, Expectation-Maximization algorithm.*

1 Introduction

Recent developments in modelling and digitizing techniques supported by the fast increase in the performance of available graphics hardware, have resulted in an increasing accumulation of 3D models and scenes. Moreover, the World Wide Web is enabling access to large databases of 3D data providing a mechanism for their wide-spread distribution. This has led to an increasing need for the development of efficient techniques for analyzing, categorization and recognition of 3D objects in large data sets. Unfortunately, objects available on the web have been designed for visualization since they contain only geometric and appearance attributes and usually lack semantic information that would facilitate their automatic analysis.

Unlike 2D images, analyzing 3D free-form surfaces is a much more complex task due to the shape and topology complexity of 3D surfaces. In general, there is no simple representation such as a matrix for 2D images that can be used to analyze 3D surfaces. Issues such as surface sampling resolution, occlusion and high dimension of the pose space further complicate the problem. In this context, features are intrinsic properties of the 3D shape which may encompass local geometry and topology related to design operations.

In this paper we investigate the extension of the scale-space theory [1, 2] combined with the Expectation-Maximization (EM) algorithm to the task of analyzing and segmenting point sampled 3D objects. We focus on some of most important features for 3D surfaces, which are patches that have similar geometric properties such as curvature distribution and point positions. The main advantage of our method is its simplicity and flexibility allowing the incorporation of different surface information such as curvature, color and texture properties. The output of our system can serve as input for many processing applications including 3D reconstruction from scanned data, 3D object matching or 3D object retrieval from large 3D databases.

1.1 Previous work

Our method combines and extends existing techniques from different research fields. In particular, we integrate recent results from image processing, discrete geometric modeling and scale-space theory.

Line features received a lot of attention from many researchers. Hubeli and Gross[3] introduced a multiresolution framework for line-type feature extraction on triangle meshes. Based on various classification operators they identify a set of feature edges and use thinning to extract lines from the set of selected edges. The multiresolution processing is based on the progressive mesh algorithm which is restricted to well generated triangle soup models.

Proceedings of the 2004 International Conference on Cyberworlds (CW'04)
0-7695-2140-1/04 $ 20.00 IEEE

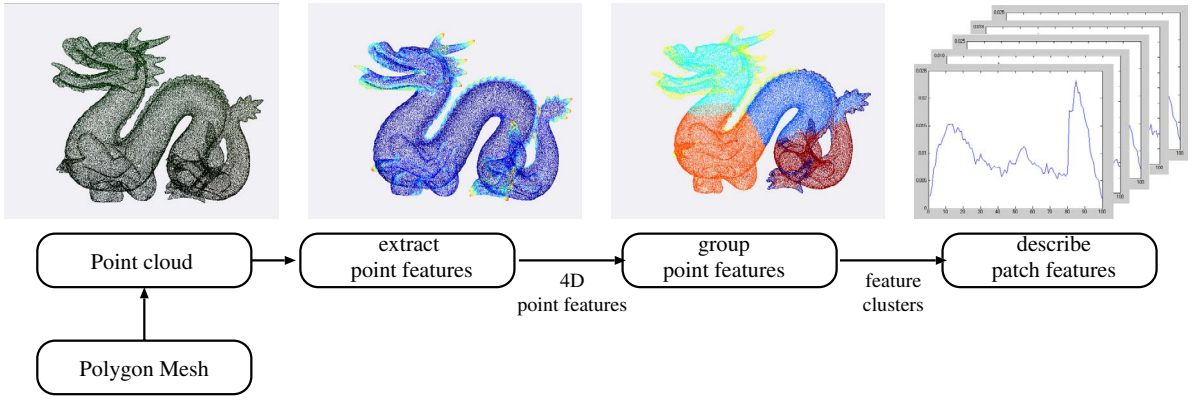

Figure 1. Feature extraction pipeline.

Gumhold *et al.*[4] presented a line feature extraction method using covariance analysis for classification. The line features are represented by a minimum spanning graph. This schema has been later extended by Pauly *et al.* in [5] using a multi-scale classification. The extracted line features are then modelled using snakes to gain more control on the feature smoothness. Geometric snakes have also been used in [6] to extract feature lines in triangle meshes based on normal variation of adjacent triangles. A user intervention is required in the system to specify an initial feature curve.

Bespalov *et. al.* [7] uses hierarchical decomposition of a 3D model into features based on its spectral properties [8]. Then the resulting hierarchical representation is used for topological matching of 3D objects.

Feature extraction has been of interest in many application fields including geometry simplification [9], non-photorealistic rendering [5], topological matching of 3D objects [10, 7], surface extraction from volumetric data [11], etc.

1.2 System overview

Figure 1 gives an overview of our feature extraction pipeline. The input of our system is a set of point cloud $P = \{p_i \in \Re^3\}$ approximating some $3D$ object M. In case a polygon soup model is given as input, we proceed by sampling the object to generate the point cloud P. The algorithm then starts by extracting for each point $p_i \in P$ a feature vector encoding the point position and the surface variation at that point (section 2). This is done by extending the scale-space representation, extensively used in the context of feature detection for images, to point sampled surfaces. Then we group surface points into regions by modeling the distribution of point features with a mixture of Gaussians using Expectation-Maximization (EM) (section 3). The final step is to describe the distribution of the surface properties of each region for use in other applications such as querying 3D objects (section 4).

2 Feature extraction

The goal of the feature extraction phase is to compute for each point $p_i \in P$ a feature vector encoding the surface properties at that point. In particular, we focus on surface variation and position features. Our feature vector extraction algorithm is based on the scale-space framework developed for segmenting 2D images [1].

2.1 Scale space framework for 3D models

Scale-space representation have been studied in the context of feature detection for 2D images [1]. The basic idea is to model an image as a convolution with Gaussian kernels of varying width called *scale*. Given a scale-space representation $L(x,t)$ of an image x we can then apply a classification operator to measure the desired properties based on the color, texture and pixel position.

To transfer these concepts to 3D surfaces, we need to specify a suitable classification operator. Many operators have been proposed in the literature. Guskov *et. al.*[12] uses the Second Order Difference (SOD) and the Extended Second Order Differences (SSOD) operators, constructed respectively from the normals of two adjacent triangles and from the average normals computed from the triangles of one-ring of an edge. Since all computations are carried out on a small region of support, these two operators perform poorly on highly detailed or noisy surfaces. Hubeli *et. al.*[3] introduced the Best Fit Polynomial (BFP) and the Angle Between Best Fit Polynomial (ABBFP) operators. These two operators have the advantage of being flexible because the support can be adapted globally and locally and thus,

are less influenced by noise. However, the main drawback is the computational overhead, due to the polynomial fitting, which becomes expensive for large models.

Various researchers have used surface curvature variation to measure the confidence that p_i belongs to a feature [5]. In our work we combine into one descriptor three features: surface variation (curvature), color and position features. These features are estimated at each point p_i and at different scales k.

To make the notion of scale concrete for irregularly sampled 3D objects, we define the scale to be the size k of the local neighborhood of a sample point p_i, hence, the features are computed on the k-nearest neighbors.

2.2 Surface curvature features

3D objects are different from 2D images. 2D images are sampled on regular grids and are completely characterized by the color and texture values at each pixel. In the other hand, one important property of 3D surfaces is the surface curvature at each sample point. It can be completely described by the distribution of the normals over the local neighborhood. The size of this neighborhood can be used as a discrete scale parameter.

The first step toward estimating the surface curvature is to compute a tangent plane at each point p_i of the 3D surface. The tangent plane $Tp(p_i)$ associated with the surface point p_i is represented as a point c_i called *center*, together with a unit normal vector $\hat{n}_i$. The center and normal for $Tp(p_i)$ are determined by gathering together the k points of P, the sample surface points, nearest to p_i (the k-neighborhood of p_i). This set is denoted by $Nbhd_k(p_i)$. The center and unit normal vector are computed so that the plane $Tp(p_i)$ is the least squares best fitting plane to the $Nbhd_k(p_i)$. That is the center c_i is the centroid of $Nbhd_k(p_i)$, and the normal $\hat{n}_i$ is determined using the principal component analysis (PCA). First the covariance matrix $CV(p_i)$ of $Nbhd_k(p_i)$ is formed. Then, if $\lambda_1 \leq \lambda_2 \leq \lambda_3$ are the eigenvalues of $CV(p_i)$ associated with unit eigenvectors v_1, v_2, v_3, respectively, the normal vector $\hat{n}_i$ is chosen to be v_1 or $-v_1$.

Appending directly the normal vector v_1 to the feature vector of the sample point p_i will increase the dimension of the feature space. We rather deal with the eigenvalue λ_1 associated to the normal vector. We adopt the same metric as [9] and [5] where the surface curvature, called also variation is defined as:

$$\omega_k(p_i) = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3} \quad (1)$$

Where k is the appropriate scale.

2.2.1 Scale selection

Finding the right scale parameter is often difficult and this is why methods for automatic scale selection have been of interest in many fields. Carson *et. al.*[1] make use of a local image texture property known as *polarity*. It is computed at a given pixel with respect to the dominant orientation in the neighborhood of that pixel. This principle can be extended to the 3D domain by using the surface variation property which is computed at a given point with respect to the dominant eigenvectors, and at different scales.

Pauly *et. al.*[5] avoids the optimal scale selection problem by using the feature persistency over scale. In this case only the critical neighborhood size is required to be estimated to avoid violating the prerequisite that all points of the neighborhood belong to the same connected region. In our case, we start by setting the maximum scale value k_{max}. Then, for each point p_i we evaluate the surface curvatures at different scales ranging from $k = 2$ to $k = k_{max}$. Then we select the scale $k_{optimal}$ which exhibits the strongest local maximum in the surface curvature across the scale axis: $k_{optimal} = argmax_k(\omega_k)$.

2.3 Combining surface curvature and position features

The final descriptor for a given point consists of four values: one for the surface curvature and three for the point position. The surface variation is the $\omega_k(p_i)$ computed at the appropriate scale k. Then we append the (x, y, z) position of the point to the feature vector. Including the position generally decreases over-segmentation and avoids grouping separated patches having the same geometric properties.

Finally, note that this formulation of the feature descriptor is flexible allowing the appending of other properties. Incorporating features such as point color components $(L * a * b)$ and texture properties is straightforward and can lead to better segmentation and analyzing of the 3D object.

3 Surface point grouping

The next step in our process is to group the surface points into meaningful clusters. Recall that the output of previous steps is a set of four dimensional feature vectors which can be considered as points in the four dimensional feature space. In order to divide these points into groups, we make use of the Expectation-Maximization (EM) algorithm [13] to determine the maximum likelihood parameters of a mixture of Gaussians in the feature space.

The EM algorithm is used for finding maximum likelihood parameter estimates when there is missing or incomplete data. In our case, the missing data is the Gaussian cluster to which the points in the 4D feature space belong.

Assume that we are using K Gaussians in the mixture model. The form of the probability density is as follows:

$$f(x|\Theta) = \sum_{i=1}^{K} \alpha_i f_i(x|\theta_i) \quad (2)$$

where x is the 4D feature vector, α_i are the mixing positive weights summing to 1, $\Theta = (\alpha_1, \ldots, \alpha_K, \theta_1, \ldots, \theta_K)$ represents the collection of parameters, and f_i is a multivariate Gaussian density parameterized by $\theta_i = (\mu_i, \Sigma_i)$ with dimension equal four.

The first step in applying the EM algorithm is to initialize the K mean vector $\mu_1, \ldots, \mu_K$ and the K covariance matrices $\Sigma_1, \ldots, \Sigma_K$ to represent each of the K groups. The initial covariances are set to the identity matrix. To initialize the means, we partition the object's bounding box into N equal boxes, and then we find the average feature vector on each of the K boxes. The update equations are:

E-step:

$$\alpha_i^{new} = \frac{1}{N} \sum_{j=1}^{N} p(i|x_j, \Theta^{old}) \quad (3)$$

$$\mu_i^{new} = \frac{\sum_{j=1}^{N} x_j p(i|x_j, \Theta^{old})}{\sum_{j=1}^{N} p(i|x_j, \Theta^{old})} \quad (4)$$

$$\Sigma_i^{new} = \frac{\sum_{j=1}^{N} p(i|x_j, \Theta^{old})(x_j - \mu_i^{new})(x_j - \mu_i^{new})^T}{\sum_{j=1}^{N} p(i|x_j, \Theta^{old})} \quad (5)$$

where N denotes the number of four dimensional feature vectors.

M-step:

$$p(i|x_j, \Theta) = \frac{\alpha_i f(x_j|\theta_i)}{\sum_{k=1}^{K} \alpha_k f(x_j|\theta_k)} \quad (6)$$

Where K denotes the number of clusters. We have thus far not discussed how to choose the number of mixture components K. Many approaches have been proposed in the literature for an automatic estimation of K [1]. In our current implementation, the number of clusters is set manually by the user.

4 Describing the regions

After point grouping, the 3D object is partitioned into K clusters we call *regions*. To describe each region characteristics, we store a simple region descriptor.

The i^{th} region descriptor's main components are: its centroid, the mean μ_i and the covariance matrix Σ_i of the i^{th} Gaussian mixture to which it is classified. In order to represent the shape distribution of the object's surface in the i^{th} region we store also the distance histogram of the points. This histogram is based on the $L2$ distance and is computed in the same manner as described by Osada *et. al.* in [14]. Incorporating the shape distribution is important for tasks such as feature comparing, matching and querying 3D objects.

Finally, we append to the region descriptor the mean surface curvature in that region.

Table 1. Timing in seconds for different stages of the processing pipeline on a 2.0GHz Athlon with 1.0GByte of main memory. The maximum scale $kmax = 120$.

Model	Dragon	Cow	Dino	bunny
#vertices	100,250	11,610	23,984	34,823
Extraction	1,151.3	106.12	250.82	336.11
Grouping	6.36	0.65	1.45	3.92
Description	1.24	0.12	0.32	0.83
Total	1,158.90	106.89	252.59	340.86

5 Results and Applications

In this section we present some of the results obtained by our framework. Experiments were conducted on a collection of 3D objects. We use both well known models such as the "Dragon" and models from the public domain such as the 3DCafe [15] collection and our original data. The computer used is an AMD Athlon(tm) 64, 2.0Ghz with $1.0GB$ of RAM and WindowsXP operating system. The system is implemented using Matlab.

Figure 2 shows some segmentation results on a set of different 3D objects. Throughout these experiments, the maximum scale k_{max} is set to 120. The number of Gaussians in the mixture K (corresponding to the number of classes) is set manually. The figure 2 shows segmentation results for $K = 4, 5, 6$. From these results, we can see that our segmentation algorithm performs well for different classes of objects and is independent from the surface complexity and topology. The tests performed on common objects show that our approach is robust against topological errors on the 3D surface. Our method recovers faithfully the salient parts of the 3D object.

Performance. Table 1 summarizes the processing time in seconds of each stage of our algorithm when using six Gaussian components during the classification. We use the kd-trees for computing the set of k-nearest neighbors of a point p_i. The table shows clearly that the surface curvature

Figure 2. Segmentation results on different point sampled models. Some of the objects have been tesselated for the visualiztion purpose. (a) input objects. (b) Segmentation results with $K = 4$, (c) with $K = 5$ and (d) with $K = 6$.

estimation step is the slowest one. One way to speed it up is to compute and then store the points connectivity prior to feature extraction.

Limitations. One limitation of our algorithm is the automatic selection of the maximum scale k_{max}. k_{max} indicates the maximum neighborhood size. Wrong selection of this value can lead to inconsistencies in the surface orientation. As a result, features can be misclassified. Such example appears in the dragon tail of figure 2. In our current implementation, the maximum scale value is set manually. An automatic setting should be considered in the future work.

Another remarkable drawback is that the number of clusters K is set manually, thus preventing a fully automatic segmentation. This issue has been extensively studied in the context of image segmentation. However, for $3D$ objects, further research must be carried out to overcome this limitation.

6 Conclusion and future work

We have presented a complete point sampled 3D object analysis and segmentation algorithm. Our main contribution is the scale-space classification and the embedding of surface curvature and point position into the four dimensional feature space. The features are extracted by clustering the set of 4D features using the Expectation-Maximization algorithm. And then, each cluster is represented by its centroid, the associated mean and covariance matrix and finally, the $L2$ shape distribution of the cluster.

This representation offers many advantages. First the method is robust in the presence of noise due to the scale-space analysis. Second, it allows a hierarchical representation of a 3D object, as well as the combination of object features using the set theory operators. Also, it can be useful for many applications such as 3D shape matching and shape-based 3D reconstruction.

As future work, it is necessary to solve the problem of automatic maximum scale selection to avoid misclassifications such as the tail of the dragon model in figure 2. Also, our framework allows incorporating other object properties. For instance, many of the 3D models available on the web relies on using texturing to simulate some effects. Hence, it is necessary to incorporate in the feature space such information.

Acknowledgements. The test data of bunny and dragon models are from the *Stanford 3D Scanning Repository*. We would like to aknowledge the Stanford Computer Graphics Laboratory for making those data available. We would like also to thank Hugues Hoppe for the cow and mechanical part models. Finally, we thank the anonymous reviewers for their valuable comments and suggestions.

References

[1] Chad Carson, Serge Belongie, Hayit Greenspan, and Jitendra Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(8):1026–1038, 2002.

[2] R.Gavilan David, Hiroki Takahashi, and Masayuki Nakajima. Image categorization using color blobs in a mobile environment. *Computer Graphics Forum*, 22(3):427–432, 2003.

[3] Andreas Hubeli and Markus Gross. Multiresolution feature extraction for unstructured meshes. In *Proceedings of the conference on Visualization '01*, pages 287–294. IEEE Computer Society, 2001.

[4] Stefan Gumhold, Xinlong Wang, and Rob MacLeod. Feature extraction from point clouds. In *Proceedings, 10th International Meshing Roundtable, Sandia National Laboratories*, pages 293–3050, 2001.

[5] Mark Pauly, Richard Keiser, and Markus Gross. Multi-scale feature extraction on point-sampled surfaces. *Computer Graphics Forum*, 22(3):281–281, 2003.

[6] Lee Yunjin and Lee Seungyong. Geometric snakes for triangular meshes. *Computer Graphics Forum*, 21(3), 2002.

[7] Dmitriy Bespalov, Ali Shokoufandeh, William C. Regli, and Wei Sun. Scale-space representation of 3d models and topological matching. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 208–215. ACM Press, 2003.

[8] Mark Pauly and Markus Gross. Spectral processing of point-sampled geometry. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 379–386. ACM Press, 2001.

[9] Mark Pauly, Markus Gross, and Leif P. Kobbelt. Efficient simplification of point-sampled surfaces. In *Proceedings of the conference on Visualization '02*, pages 163–170. IEEE Computer Society, 2002.

[10] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 203–212. ACM Press, 2001.

[11] Leif P. Kobbelt, Mario Botsch, Ulrich Schwanecke, and Hans-Peter Seidel. Feature sensitive surface extraction from volume data. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 57–66. ACM Press, 2001.

[12] Igor Guskov, Wim Sweldens, and Peter Schroder. Multiresolution signal processing for meshes. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques SIGGRAPH'99*, pages 325–334. ACM Press/Addison-Wesley Publishing Co., 1999.

[13] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of Royal Statistical Society*, 39:1–38, 1977.

[14] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Trans. Graph.*, 21(4):807–832, 2002.

[15] 3Dcafe:. http://www.3dcafe.com.