Extraction of Geometric Primitives from Point Cloud Data

Sung Il Kim* and Sung Joon Ahn**

*Department of Golf Systems, Tamna University, 697-703 Seogwipo, Korea
(Tel: +82-64-735-2122; Fax: +82-64-735-2033; Email:sunk@tnu.ac.kr)
**School of Information and Communication Engineering, Sungkyunkwan University, 440-746 Suwon, Korea
(Tel: +82-31-290-7977; Fax: +82-31-290-7179; Email:finger@skku.edu)

Abstract: Object detection and parameter estimation in point cloud data is a relevant subject to robotics, reverse engineering, computer vision, and sport mechanics. In this paper a software is presented for fully-automatic object detection and parameter estimation in unordered, incomplete and error-contaminated point cloud with a large number of data points. The software consists of three algorithmic modules each for object identification, point segmentation, and model fitting. The newly developed algorithms for orthogonal distance fitting (ODF) play a fundamental role in each of the three modules. The ODF algorithms estimate the model parameters by minimizing the square sum of the shortest distances between the model feature and the measurement points. Curvature analysis of the local quadric surfaces fitted to small patches of point cloud provides the necessary seed information for automatic model selection, point segmentation, and model fitting. The performance of the software on a variety of point cloud data will be demonstrated live.

Keywords: segmentation, object recognition, automation, point cloud, orthogonal distance fitting, differential geometry

1. Introduction

Model reconstruction from point cloud data is a highly relevant subject to computer vision and reverse engineering with which the real objects should be represented by their geometric model parameters. A fully automatic and generally applicable solution to model reconstruction might be realized only through a highly sophisticated hardware and software technique analyzing all the available information on the objects, such as point cloud, object database, object surface color and texture. Certainly, all the necessary technical and theoretical tools will not be available in the near future. Nevertheless, the aim to increase the degree of automation of model recognition and reconstruction is ever-present.

If we restrict our interest field to industrial environment, we find that large portion of man-made objects (e.g. manufacturing facilities and workpieces) can be modeled as combination of *exact features*, i.e. plane, sphere, cylinder, cone, and torus [7]. Thus, even if we limit the range of model features to exact features, there is still large demand on the automatic techniques for model reconstruction, e.g., for reverse engineering, intelligent robot, and digital factory. As far as we are aware, there is no supplier of such technique [6], [11].

Under the circumstances mentioned above, we have developed a software tool for automatic extraction of exact features from point cloud, based on our previous work on a semi-automatic solution [2]. The functionality of the software tool is analogous to that of human intelligence searching for exact features in a dark room environment. In this paper we describe the algorithmic techniques implemented in the software tool. Beside the experimental result given in this paper, we will demonstrate live the performance of the software on a variety of point clouds generated by different 3-D measuring techniques.

2. Parametric Model Reconstruction

2.1. Real Object, Point Cloud and Model Feature

Before we begin describing the algorithmic details of the software tool, we like to survey the conditions for parametric model reconstruction from point cloud (Fig. 1). To bear in mind is:

- Point cloud is subject to measurement errors
- Model feature represents only roughly the true surface of real object
- Model parameters are estimated from point cloud, of which results are subject to the applied estimation method.

This means all the three passages between the three parties (real object, point cloud, and model feature) cause inevitably errors during the reconstruction process comprised of 3-D measurement, model selection and parameter estimation. In order to layout a feasible technical solution to this inconvenient situation, we review the very fact of parametric model reconstruction.

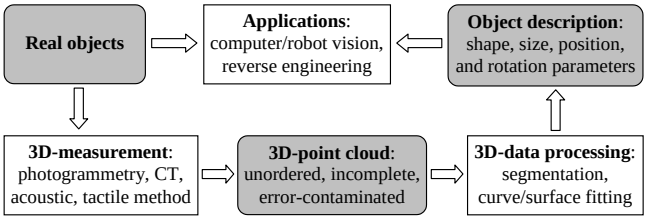

Fig. 1. Parametric model reconstruction of real objects.

2.1.1 Model Features for Real Object

For representing an object surface we can consider three ways: point model, facet model, and analytic model. The point model represents an object surface through a set of points. The usability of the point model is limited practically to the registration of a second point set, usually, a set of measurement points. The facet model consists of a set of

polygons. Although the facet model is suitable for object visualization and lithography, it does not provide applications with information on shape, size, position and orientation of objects. The analytic model describes an object surface by using mathematical formulas, which is employed by our software tool.

There are three description forms of analytic models (curves and surfaces), i.e. explicit, implicit and parametric form [3], [4], [9]. With diverse applications handling dimensional models or objects, the usual description form of curve/surface is the implicit or the parametric form. In addition, many applications (e.g. the bin-picking or the obstacle-avoidance task in robotics) involve describing real objects in terms of shape, size, position and orientation. Thus, we group the model parameters $\mathbf{a}$ of a curve/surface into form $\mathbf{a}_\mathrm{g}$, position $\mathbf{a}_\mathrm{p}$, and rotation parameters $\mathbf{a}_\mathrm{r}$ as follows:

$$\begin{cases} f(\mathbf{a}_\mathrm{g}, \mathbf{x}) = \mathbf{0} & : \text{implicit feature} \\ \mathbf{x}(\mathbf{a}_\mathrm{g}, \mathbf{u}) & : \text{parametric feature} \ , \end{cases} \tag{1}$$

$$\mathbf{X} = \mathbf{R}^{-\mathbf{1}}_{\omega,\varphi,\kappa}\mathbf{x} + \mathbf{X}_\mathrm{o} \quad \text{or} \quad \mathbf{x} = \mathbf{R}_{\omega,\varphi,\kappa}(\mathbf{X} - \mathbf{X}_\mathrm{o}) \ , \tag{2}$$

with

$$\begin{aligned} \mathbf{a}^\mathrm{T} &\triangleq (\mathbf{a}_\mathrm{g}^\mathrm{T}, \mathbf{a}_\mathrm{p}^\mathrm{T}, \mathbf{a}_\mathrm{r}^\mathrm{T}) \\ &= (a_1, \ldots, a_l, X_\mathrm{o}, Y_\mathrm{o}, Z_\mathrm{o}, \omega, \varphi, \kappa) \ . \end{aligned} \tag{3}$$

The form parameters determine the shape and size of the canonical model feature (1) defined in a model coordinate frame xyz and, are invariant to the rigid body motion (2) of the model feature. Our software tool extracts automatically the exact features (plane, sphere, cylinder, cone, and torus) from a given point cloud and, estimates their model parameters in terms of form, position, and rotation parameters (1)–(3).

2.1.2 Point Cloud

Optical 3-D measuring devices available on the current market can generate millions of dense 3-D points in a few seconds [11]. However, the point cloud is usually not dense enough to cover the details of object surface. This means we should hold back from undersampling the point cloud. And, it is generally assumed the point cloud is not ordered. Furthermore, because of the limited accessability of measuring devices to object surface, the point cloud covers only partially the object surface. On the other hand, with the point cloud generated by CT technology enjoying a full accessability to object surface, the segmentation of closely neighboring object surfaces is a challenging task. Our software tool can handle an unordered incomplete and complex point cloud with a large number of data points.

A measurement point is the probable observation of an unknown *nearest* point on object surface to the measurement point [1]. The distance between the measurement point and the unknown object point is the true measurement error. In practice, because the true object surface is unknown, it is substituted by an associating model feature [7]. The true measurement error is substituted by the minimum distance (geometric distance, Euclidean distance) between the model feature and the measurement point. This error definition solely outlines the algorithmic functionalities which should be implemented in the software tool for a reliable and accurate parametric model reconstruction from point cloud. The minimum distance should be used not only as the decision measure between the inliers and the outliers of a model feature (segmentation), but also as the error measure to be minimized by the procedures of parameters estimation (model fitting) [1], [7], [8]. Although the calculation and minimization of the minimum distances are computationally expensive, they are of vital importance to a reliable and accurate parametric model reconstruction from point cloud.

2.2. Orthogonal Distance Fitting

We briefly describe the *orthogonal distance fitting* (ODF) that estimates model parameters by minimizing the square sum of error distances between the model feature and the given points. Interested readers are referred to [3] for a complete description of the ODF algorithms that estimate the model parameters in terms of form, position and rotation parameter (1)–(3).

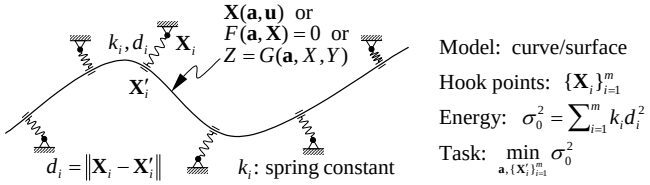

Fig. 2. Interpretation of the orthogonal distance fitting as an energy minimization problem.

The ODF task can be interpreted as an energy minimization problem illustrated in Fig. 2, with which the energy (cost) function is defined as

$$\sigma_0^2 \triangleq (\mathbf{X} - \mathbf{X}')^\mathrm{T}\mathbf{P}^\mathrm{T}\mathbf{P}(\mathbf{X} - \mathbf{X}') \quad \text{or} \quad \boldsymbol{\sigma}_\mathbf{0}^\mathbf{2} \triangleq \mathbf{d}^\mathrm{T}\mathbf{P}^\mathrm{T}\mathbf{P}\mathbf{d} \ , \tag{4}$$

where $\mathbf{X}^\mathrm{T} = (\mathbf{X}_\mathbf{1}^\mathrm{T}, \ldots, \mathbf{X}_\mathbf{m}^\mathrm{T})$ and $\mathbf{X}'^\mathrm{T} = (\mathbf{X}_\mathbf{1}'^\mathrm{T}, \ldots, \mathbf{X}_\mathbf{m}'^\mathrm{T})$ are the coordinate row vectors of the m given (measurement) points and of the m corresponding points on the model feature, respectively. The diagonal elements of the weighting matrix $\mathbf{P}^\mathrm{T}\mathbf{P}$ correspond to the spring constants $\{k_i\}_{i=1}^m$ in Fig. 2. To minimize the cost function (4) the ODF algorithm minimizes not only the square sum but also every single distance $\{d_i\}_{i=1}^m$ between the model feature and the given points. Because the minimum distances $\{d_i\}_{i=1}^m$ are nonlinear to the model parameters, the ODF task is inherently a *nonlinear minimization problem* that must be solved through iteration. The computing cost and the memory space usage of the ODF algorithms in [3] are proportional to the number of data points, thus the algorithms are suitable for processing a massive point cloud. By investigating the resulting cost (4) and the parameter covariance matrix, we can test the overall performance and the reliability of the model selection and model fitting.

3. Automatic Model Extraction

Given a set of points, the feature extraction procedure consists of two substantial sessions of segmentation and model fitting, respectively (Fig. 1). At this point we confront a

chicken-and-egg dilemma. Namely, without the geometric information on model feature we cannot decide between the inliers and the outliers of model feature (segmentation). Reversely, without the inliers we cannot get the geometric information on model feature (model fitting). To resolve this information deadlock, either of the two sides should provide the seed information triggering the other side.

3.1. Model Selection and Initial Model Parameters

We obtain the geometric seed information on model feature from a small patch of point cloud, which is comparable with touching an object then guessing its geometry in a dark room environment:

1. Cut (touch) a small initial patch from the point cloud.
2. Fit a plane to the patch through the moment method (noniterative linear ODF) [8].
3. Fit a quadric surface to the patch through ODF starting from the plane parameters.
4. Get the orthogonal footing point on the surface from the mass center of the patch.
5. Calculate the surface normal, principal curvatures and axes at the footing point [5].
6. Choose the model type for the patch by analyzing the signed curvature radii (Fig. 3).
7. Derive the size, center and orientation parameters of the chosen model feature from the surface normal, curvature radii and principal axes at the footing point.
8. Fit the initial model feature to the patch through ODF starting from the model parameters derived in the last step.

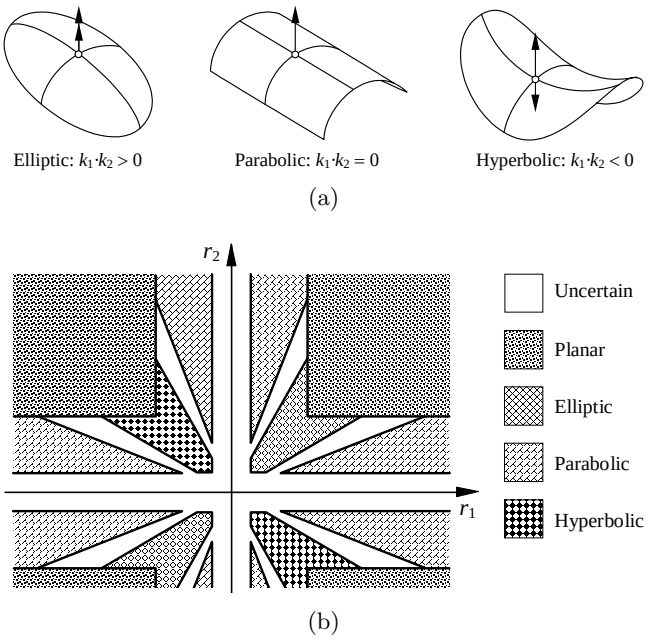

Fig. 3. Classification of local surface types according to the two principal curvatures k_1 and k_2. (a) Flat for plane, elliptic for sphere or torus, parabolic for cylinder or cone, and hyperbolic for torus; (b) Curvature radius map for local surface types ($r_1 = 1/k_1, r_2 = 1/k_2$).

The classification of local surface types (Fig. 3a [5]) according to the local curvatures (including the mean and Gaussian curvatures) of a curved surface fitted to a small point patch is known in literature [10]. Instead of the curvatures, our software tool employs the curvature radii which correspond to the feature radius (Fig. 3b). During the above procedures the ODF algorithm plays an important role in determining the parameters of the intermediate quadric surface and of the initial model feature from the small point patch. Other fitting algorithms than the ODF algorithm, which minimize some error measures other than the minimum distance, are prone to fail to fit a model feature to a small point patch. We address the influencing factors on the procedures shown above:

Model scale. The aperture radius of the small patch must be chosen by considering the curvature radius of the model features to be extracted. At the same time, it must be at least 5–10 times larger than the accuracy of point measurement.

Model uncertainty. If the rms distance error of quadric fitting is larger than the measurement accuracy, we ignore the current small patch. If either of the two absolute curvature radii is too small relatively to the aperture radius (see Fig. 3b), we ignore the current small patch. We put some tolerances to the ratio of curvature radii for classifying surface types. If both the absolute curvature radii are larger than 10–30 times of the aperture radius, we assume a plane. In the case of elliptic surface, we choose the smaller curvature radius as the sphere radius.

Model ambiguity. In the case of parabolic surface, we assume a cone. When the resulting vertex angle becomes small enough at the end of the overall process (see Fig. 4), we assume a cylinder and continue the process. Torus should be extra handled, since it contains all the local surface types (elliptic for outer face, parabolic for top/bottom face, and hyperbolic for inner face of either of the two chained tori).

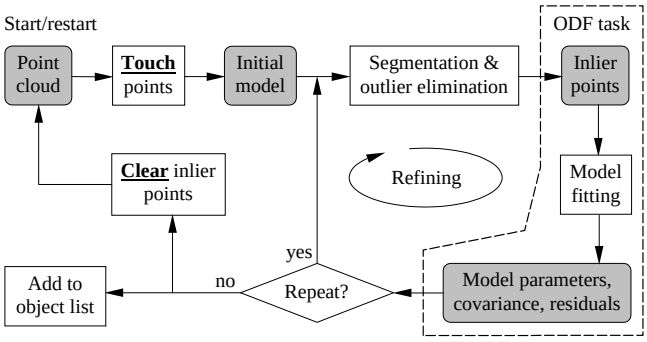

Fig. 4. "Touch & Clear" in point cloud for automatic feature extraction.

3.2. Overall Process and Experimental Result

Once the model type and parameters are initialized, the interaction loop between the segmentation and the model fitting can be triggered. As noted in Sect. 2.1, the minimum distance of a given point to the model feature should be used as the decision measure whether the point is an inlier of model feature. However, with regard to a specific model feature, the large part of a point cloud is occupied by plain outliers, causing a high computing cost of unnecessarily calculating the minimum distances. Through utilizing

the parameter grouping (1)–(3) and the properties of implicit model description, we can efficiently eliminate the plain outliers from the point cloud. The overall process of automatic feature extraction can be described as follows (see Fig. 4):
1. Initialize the model feature (model type, size, position and orientation) (Sect. 3.1).
2. Stamp all the points lying outside the domain box as plain outlier.
3. Except for the inlier candidates lying between the two (inner and outer) iso-features of model feature, stamp all points as plain outlier.
4. Evaluate the rms distance of the inlier candidates to the model feature.
5. Stamp only the inlier candidates as inlier, of which distances to the model feature are not larger than 2–3 times of the rms distance.
6. Update the model parameters through ODF to the inliers.
7. If necessary, repeat from the second step ('Refining' in Fig. 4).
8. Save the model parameters, and, clear the inliers from the point cloud.
9. Repeat from the first step until no more dense point patch can be found (touched).
As an experimental example for automatic feature extraction from point cloud, we applied our software tool to a point cloud of about 68,600 points generated by stripe-projection method (structured-light method) (Fig. 5a). All the relevant model features could be extracted fully automatically and correctly (Fig. 5b).

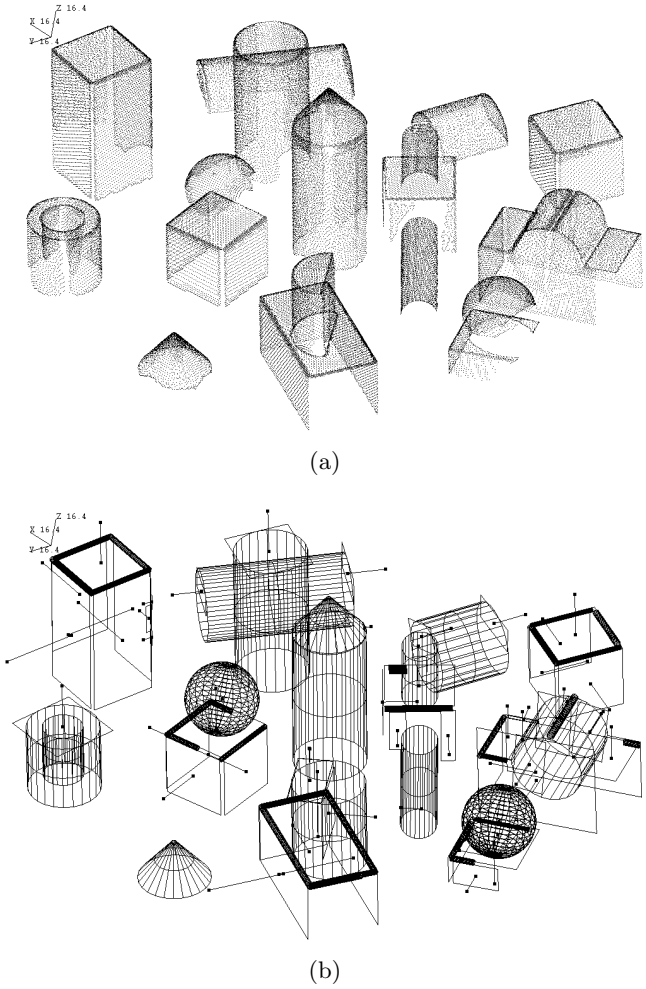

Fig. 5. Automatic extraction of geometric primitives from point cloud. (a) Unordered and incomplete point cloud of about 68,600 points; (b) Geometric primitives extracted fully automatically from the point cloud. The computing time with a Pentium 3.0 GHz PC is a total of 48 s.

4. Summary

We have developed a software tool for a fully-automatic extraction of exact features from unordered incomplete and error-contaminated point cloud. The necessary seed information for the model selection, segmentation and model fitting could be obtained from an intermediate quadric surfaces fitted to small patches of point cloud. The geometric error measure is of vital importance to both the segmentation and the model fitting, although the required computing cost is relatively high. In order to save the computing cost of segmentation, we exploited the parameter grouping and the properties of implicit model description. We demonstrated the outstanding performance of the software tool on a set of real measurement points generated by stripe-projection method.

References

[1] R. J. Adcock, "Note on the method of least squares," *The Analyst*, vol. 4, pp. 183–184, 1877.

[2] S. J. Ahn, I. Effenberger, S. Roth-Koch, and E. Westkämper, "Geometric Segmentation and Object Recognition in Unordered and Incomplete Point Cloud," *Lect. Notes Comput. Sc.*, vol. 2781, pp. 450–457, 2003.

[3] S. J. Ahn, *Least-Squares Orthogonal Distance Fitting of Curves and Surfaces in Space*, Springer, Heidelberg, 125 pages, *Lect. Notes Comput. Sc.*, vol. 3151, 2004.

[4] R. M. Bolle and B. C. Vemuri, "On Three-Dimensional Surface Reconstruction Methods," *IEEE Trans. Pattern Analy. and Mach. Intell.*, vol. 13, pp. 1–13, 1991.

[5] M. P. do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice-Hall, 1976.

[6] A. Hoover, G. Jean-Baptiste, X. Jiang, P. J. Flynn, H. Bunke, D. B. Goldgof, K. Bowyer, D. W. Eggert, A. Fitzgibbon, and R. B. Fisher, "An Experimental Comparison of Range Image Segmentation Algorithms," *IEEE Trans. Pattern Analy. and Mach. Intell.*, vol. 18, pp. 673–689, 1996.

[7] ISO 10360-6:2001, *Geometrical Product Specification (GPS) - Acceptance test and reverification test for coordinate measuring machines (CMM) - Part 6: Estimation of errors in computing Gaussian associated features*, ISO, 2001.

[8] K. Pearson, "On Lines and Planes of Closest Fit to Systems of Points in Space," *The Philosophical Magazine: Ser. 6*, vol. 2, pp. 559–572, 1901.

[9] D. H. von Seggern, *CRC standard curves and surfaces*, 2nd Ed., CRC Press, 1993.

[10] B. C. Vemuri, A. Mitiche, and J. K. Aggarwal, "Curvature-based representation of objects from range data," *Image and Vision Computing*, vol. 4, pp. 107–114, 1986.

[11] www.pobonline.com, "2004 3D Laser Scanner Hardware Survey," *Point of Beginning*, pp. 20–25, January 2004.