PAPER

# 3D Model Segmentation and Representation with Implicit Polynomials

**Bo ZHENG**[†a)], **Jun TAKAMATSU**[†], ***Nonmembers***, ***and*** **Katsushi IKEUCHI**[†], ***Member***

**SUMMARY** When large-scale and complex 3D objects are obtained by range finders, it is often necessary to represent them by algebraic surfaces for such purposes as data compression, multi-resolution, noise elimination, and 3D recognition. Representing the 3D data with algebraic surfaces of an implicit polynomial (IP) has proved to offer the advantages that IP representation is capable of encoding geometric properties easily with desired smoothness, few parameters, algebraic/geometric invariants, and robustness to noise and missing data.

Unfortunately, generating a high-degree IP surface for a whole complex 3D shape is impossible because of high computational cost and numerical instability. In this paper we propose a 3D segmentation method based on a cut-and-merge approach. Two cutting procedures adopt low-degree IPs to divide and fit the surface segments simultaneously, while avoiding generating high-curved segments. A merging procedure merges the similar adjacent segments to avoid over-segmentation. To prove the effectiveness of this segmentation method, we open up some new vistas for 3D applications such as 3D matching, recognition, and registration.

***key words:*** *Implicit polynomial (IP), 3D segmentation, 3D representation.*

## 1. Introduction

Recently, 3D models obtained by scanning actual objects with range finders are widely applied in computer vision. The progress of the modeling techniques enables us to obtain 3D models of large objects whose dimension is over 100 meters with precision within a few centimeters [1]. Thus the model data is becoming large, complex, and of high quality.

However, this expansion might cause problems in data processing, such as modeling 3D shapes, representing data in fine detail, and sharing the huge amount of data on the Internet. Therefore there is a need to represent the obtained 3D models by algebraic surfaces for such purposes as rendering, automatic registration, compression, multi-resolution, and noise elimination.

Representation with algebraic surfaces defined by IPs is effective because it has many good properties such as fast fitting, few parameters, algebraic/geometric invariants, and robustness against noise and occlusion. Furthermore, the method for estimating the coefficients of an IP from the given data set is very simple.

Unfortunately, two disadvantages frustrate the direct application of high-degree IPs to complex shape representations. One is global instability and the other one is computational inefficiency. The former results in generating some redundant zero sets around the desired zero set (see [2]). The latter is that increasing the number of the coefficients makes the computation too time consuming. Although these disadvantages prevent us from fitting a high-degree IP to a whole complex date set, low-degree IPs lead to undesired loss of accuracy.

We propose a 3D segmentation method based on a cut-and-merge approach. In this method, a complex surface can be divided into some meaningful segments and each segment can be well represented by a low-degree IP. Advantages of our method are that it is capable of 1) finding the appropriate segmentation for various desired accuracies; 2) achieving stable fitting since we avoid using high-degree IPs; and 3) decomposing heavy computation from one high-degree IP fitting into light computations from multiple low-degree IPs. The segmentation result opens up some new vistas for 3D applications such as 3D matching, recognition, and registration. Note this method can be applied to general 3D data formats such as polygonal mesh models.

This paper is organized as follows: Section 2 gives a review of prior works on 3D representation and segmentation; Section 3 provides a new segmentation method employed IP techniques; Section 4 reports experimental results; Section 5 discusses the further possible applications extended from our method, followed by conclusions in section 6.

## 2. Related work

### 2.1 The methods for 3D representation

There have been a number of 3D representation methods for applications such as rendering, recognition, registration, and shape retrieval. Some of them specifically dealt with feature encoding for 3D object retrieval but not for the shape representation [3]. Others gave the complete geometric description of the 3D shapes in some mathematical forms.

Parametric surfaces, such as Non-Uniform Rational B-Spline (NURBS), Rational Gaussian (RaG) [4], etc., are widely used in computer aided design and manufacturing (CADM). But they have the disadvantage that it is difficult to make a surface defined on

[†]Institute of Industrial Science, The University of Tokyo Komaba 4–6–1, Meguro-ku,Tokyo 153-8505 Japan

a) E-mail: zheng@cvl.iis.u-tokyo.ac.jp

a parametric region fit an arbitrary region. Radial basis function (RBF) [5] for representing a large-scale dense data set has too many parameters, which leads to heavy computation for fitting and surface reconstruction. Superquadrics and Generalized cylinders are difficult (or impossible) to use to represent a complex shape model, unless there exists an appropriate segmentation method that makes each segment lie within a single Superquadric or a Generalized cylinder. For a detailed survey, the reader is referred to [6].

Algebraic surfaces are generally defined in a polynomial form, and its representation for 3D data sets has proved useful for computer vision applications. In contrast to other functions based representations such as B-spline, NURBS and RBF, IP representation may not be able to give a relatively accurate model. But this representation is more attractive for applications requiring fast registration and recognition (see the works [2], [7]–[10]), because of its algebraic/geometric invariants [11]. Sahin also showed some experimental results to prove the robustness for noising and missing data in [12].

Implicit polynomial (IP) is an implicit function defined in multivariate polynomial form. For example, an IP for 3D space is defined as:

$$\begin{aligned} f_n(\boldsymbol{x}) &= \sum_{0\le i,j,k;i+j+k\le n} a_{ijk}x^i y^j z^k \\ &= (\underbrace{1 \;\; x \;\; \ldots \;\; z^n}_{\boldsymbol{m}(\boldsymbol{x})^T})(\underbrace{a_{000} \;\; a_{100} \;\; \ldots \;\; a_{00n}}_{\boldsymbol{a}})^T, \end{aligned} \quad (1)$$

where $\boldsymbol{x} = (x \;\; y \;\; z)$ is a data point. $f_n(\boldsymbol{x})$'s zero set $\{\boldsymbol{x}|f_n(\boldsymbol{x}) = 0\}$ is used to represent the given data set. Representing a 3D data set with IP belongs to the conventional fitting problem. For solving this problem, the prior methods are classified into nonlinear methods [13]–[15] and linear methods [2], [12], [16], [17]. Because the linear methods are simpler and much faster than the nonlinear ones, they are receiving more attention.

As described above, an IP often suffers from the difficulty of representing complex models. The low-degree IP leads to undesired inaccuracy, whereas the high degree leads to global instability. Therefore, in order to accurately describe such a complex model without losing IP's advantages, the method to segment its surfaces and to represent each of them using an IP separately is often used.

### 2.2 The methods for 3D segmentation

For 3D segmentation, there mainly exist two categories of the methods. One is a geometrical method, and the other one is a clustering method. The former segments the surface according to the local geometrical properties of 3D surface, such as Gaussian and mean curvatures. For triangulated meshes, the Gaussian and mean curvature at a vertex is approximated from its adjacent triangles [18], [19], or from the locally approximated functions discussed in [20]–[22]. The latter segments them by minimizing a certain energy function with topological information and some metric of distance. The minimization is done by some clustering technique, such as the graph cut method.

A deficiency of the geometrical methods is that they are generally vulnerable to data noises and occlusions, since the geometric property associated with each point has to be decided by a differential operation. On the other hand, clustering methods bias segmentation according to topological information. The result might be useful to object partition, such as the fact that a body can be divided into a head, two arms, two legs etc., but it cannot guarantee that each segment is suitable for a low-degree IP. For more detail surveys, we refer to [23], [24].

Some other methods are proposed in [25]–[27], where they used low-degree implicit functions for modeling voxel-piecewise or sphere-piecewise segments. However, since these methods are mainly for rendering, their segmentation results are not Euclidean invariant and thus not suitable for solving general problems such as object recognition and registration (for example [11]).

Since all the above methods cannot satisfy our purpose, we propose a novel method for automatically segmenting 3D shapes. All the segments have a one-to-one correspondence to the low-degree IPs.

## 3. Segmentation with IP

Starting from this section, we present a novel method for 3D segmentation, which uses the cut-and-merge strategy to simplify the surface into regions satisfying three requirements as follows:

(1) Every region can be well fitted by a low-degree IP with desired accuracy.

(2) The number of the regions is as few as possible;

(3) Each boundary is cut as clearly as possible on object edges.

Therefore, an advantage in our method is that it can generate meaningful regions that have a one-to-one correspondence to IPs. That is, the geometric properties of each region can be well encoded with an IP. A clear boundary of each region also can be extracted easily and automatically.

First, we define the degree of the fitness, that is, the similarity between an IP and a point data set. We use this similarity to decide how well the fitting is performed. Next, we describe our cut-and-merge strategy in detail, which consists of the three following procedures:

- Unfit cutting procedure:
  Iteratively cutting the regions until no unfit region

exists.
- High-curved cutting procedure:
  Iteratively cutting the regions until neither high-curved nor distorted region exists.
- Merging procedure:
  Iteratively merging the regions which are acceptable for the same IP.

### 3.1 Similarity

Before we give the details of our cut-and-merge strategy let us define two functions to measure the similarity between the IPs and the regions as follows:

$$D_{dist} = \frac{1}{N}\sum_{i=1}^{N} e_i \tag{2}$$

$$D_{smooth} = \frac{1}{N}\sum_{i=1}^{N} (\boldsymbol{N}_i \cdot \boldsymbol{n}_i), \tag{3}$$

where

$$e_i = \frac{\mid f(\boldsymbol{x}_i) \mid}{\| \bigtriangledown f(\boldsymbol{x}_i)\|}, \quad x_i \in \{data\ set\} \tag{4}$$

$$\boldsymbol{n}_i = \frac{\bigtriangledown f(\boldsymbol{x}_i)}{\| \bigtriangledown f(\boldsymbol{x}_i)\|}, \quad x_i \in \{data\ set\}. \tag{5}$$

$N$ is the number of vertices, and $\boldsymbol{N}_i$ is the normal associated with the point. Although $e_i$ in (4) is not a real Euclidean distance, it is proved to be useful for approximating the Euclidean distance from vertex $\boldsymbol{x}_i$ to the zero set of $f(\boldsymbol{x})$ [2]. $\boldsymbol{n}_i$ is the normalized gradient vector of $f$ at vertex $\boldsymbol{x}_i$. Thus it represents the normal vector of $\boldsymbol{x}_i$ on the zero set.

$D_{dist}$ and $D_{smooth}$ can be considered as two measurements of distance and smoothness between a region and the zero set of an IP. From the definition of the similarity measures, the smaller $D_{dist}$ and the larger $D_{smooth}$ are, the more the region is similar to IP. The perfect case is $D_{dist} = 0 \wedge D_{smooth} = 1$. Accordingly two thresholds $T_1(T_1 > 0)$ and $T_2(T_2 < 1)$ are set into the constraint of

$$(D_{dist} < T_1) \ \wedge \ (D_{smooth} > T_2). \tag{6}$$

If this constraint is satisfied, we then say the current region can be described by the corresponding IP.

### 3.2 Unfit cutting procedure

The first procedure is named *Unfit cutting procedure*, and its main task is to divide the IP-unrepresentable data set into IP-representable segments. First, we use a low-degree IP to measure a surface region with constraint (6). Second, if the constraint is satisfied, this region is regarded as undividable and will be outputted. Otherwise, this region is regarded as dividable and will be divided into two parts: Inner part ($\{InnerRg\}$) and Outer part ($\{OuterRg\}$) defined as:

$$\begin{aligned} \{InnerRg\} &:= \{\boldsymbol{x}_i | f(\boldsymbol{x}_i) \le 0\}, \\ \{OuterRg\} &:= \{\boldsymbol{x}_i | f(\boldsymbol{x}_i) > 0\}; \end{aligned} \tag{7}$$

Third, we repeat the above operation until there are no more dividable regions. In short, the algorithm is as follows:

**Algorithm 1: Unfit cutting procedure**

Input $I$: point data set, $T_1$, $T_2$: thresholds
Output $\{\tilde{R}\}$: regions, $\{\tilde{IP}\}$:polynomials
Working Variables $IsAnyCut$: A flag, if any region is cut in the main loop, it will be set true, otherwise false.
Step 1 Initialization:
  $\{\mathcal{R}\} \leftarrow \{whole\ data\ set\}$;
  Initialize $\{InnerRg\}$ and $\{OuterRg\}$ with two empty stacks;
Step 2 Main loop:
  $IsAnyCut \leftarrow false$;
  **for** $R_i \in \{\mathcal{R}\}, i = 1, 2, \ldots$ **do**:
    $[IP_i] \leftarrow$ ***3LFitting***$(R_i)$;
    $[Dd, Ds] \leftarrow$ ***Measuring***$(R_i, IP_i)$;
    **if** Dd $< T1$ and Ds $> T2$
      push $R_i$ into $\{\tilde{R}\}$;
      push $IP_i$ into $\{\tilde{IP}\}$;
    **else**
      [InnerPart, OuterPart] $\leftarrow$ ***BiSegment***$(R_i, IP_i)$;
      add InnerPart into $\{InnerRg\}$;
      add OuterPart into $\{OuterRg\}$;
      $IsAnyCut \leftarrow true$;
    **end**
  **end**
Step 3 Checking end:
  **if** $IsAnyCut$
    $\{\mathcal{R}\} \leftarrow \{InnerRg\} \cup \{OuterRg\}$;
    **Clear** $\{InnerRg\}$ and $\{OuterRg\}$;
    **GOTO** Step 2;
  **else**
    Output $\{\tilde{R}\}$, $\{\tilde{IP}\}$;
    **Return**;
  **end**

Here, the function ***3LFitting*** uses the 3L method [16] to fit a region and return an IP as a result; the function ***Measuring*** is used to measure the similarity between a region and an IP referring to (2, 3); and function ***BiSegment*** divides a region into two parts, Inner part and Outer part defined in (7).

A simple 2D example is shown in Fig. 1. In this case, the 1-degree IPs (lines) are used. First, a 1-degree IP is used to fit the whole data set. Second, by measuring the similarity, the data set is divided into two parts, an inner part shown with region I and an outer part shown with regions II and III. Then all of these regions are fitted again with new IPs. Since regions II and III can be fitted well by 1-degree IPs, they will be outputted in the next step. On the other hand, since

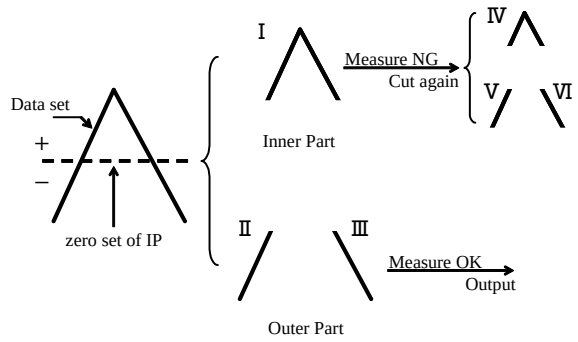



**Fig. 1** Cutting procedure

region I cannot be well fitted by a 1-degree IP, it will be cut again. The procedure is done iteratively, until every region is acceptable to a 1-degree IP with constraint (6) of two certain thresholds.

### 3.3 High-curved cutting procedure

The task of the high-curved cutting procedure is to find the high-curved regions resulting from the unfit cutting procedure, and to divide them into low-curved regions. In the results from the unfit cutting procedure, there may be highly curved or distorted regions, e.g., Region IV in Fig. 1. But from the view of segmentation, the two edges of this angle should be divided into two parts, and each part belongs to a different IP. For this reason, we design the high-curved cutting procedure to find the curved regions and cut them again.

#### 3.3.1 How to find the curved regions

To find the curved regions, we have to know the geometric characteristics of the surface first. IPs can provide a convenient way to find the surface curvatures quickly and robustly.

As defined in (5), the normal vector $\boldsymbol{n}_i$ at vertex $\boldsymbol{x}_i$ is represented by the first order partial derivative of $f$. If $\boldsymbol{g}_i = \bigtriangledown f(\boldsymbol{x}_i) = (\frac{\partial f(\boldsymbol{x}_i)}{\partial x} \ \frac{\partial f(\boldsymbol{x}_i)}{\partial y} \ \frac{\partial f(\boldsymbol{x}_i)}{\partial z})^T$, then $\boldsymbol{n}_i = (n_x \ n_y \ n_z)^T = \boldsymbol{g}_i / \|\boldsymbol{g}_i\|$. The second order derivatives in (8) contain information about the curvature of isosurfaces of the implicit function.

$$\bigtriangledown \boldsymbol{n}_i^T = \begin{pmatrix} \frac{\partial n_x}{\partial x} & \frac{\partial n_x}{\partial y} & \frac{\partial n_x}{\partial z} \\ \frac{\partial n_y}{\partial x} & \frac{\partial n_y}{\partial y} & \frac{\partial n_y}{\partial z} \\ \frac{\partial n_z}{\partial x} & \frac{\partial n_z}{\partial y} & \frac{\partial n_z}{\partial z} \end{pmatrix}. \tag{8}$$

It can be solved as follows (see the derivation in [20]):

$$\bigtriangledown \boldsymbol{n}_i^T = \frac{1}{\|\boldsymbol{g}_i\|} \mathrm{GH}, \tag{9}$$

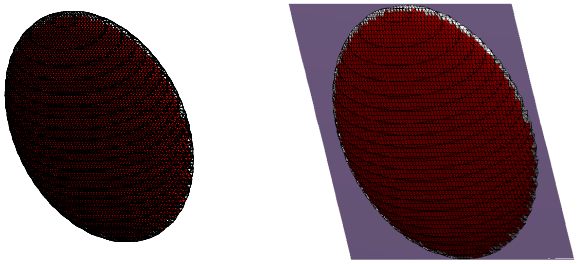
where $\mathrm{G} = \mathrm{I} - \boldsymbol{n}_i \boldsymbol{n}_i^T$; I is the $3 \times 3$ identity matrix; and H is the Hessian matrix:

$$\mathrm{H} = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} \\ \frac{\partial^2 f}{\partial x \partial x} & \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial y \partial z} \\ \frac{\partial^2 f}{\partial x \partial z} & \frac{\partial^2 f}{\partial y \partial z} & \frac{\partial^2 f}{\partial z^2} \end{pmatrix}.$$

From the eigenvalues of the matrix $\bigtriangledown \boldsymbol{n}_i^T$, the so-called principal curvatures, $k_1$ and $k_2$ are obtained. And the maximum/minimum absolute curvatures are defined as

$$\kappa_{max} = \max(|k_1|, |k_2|) \tag{10}$$

$$\kappa_{min} = \min(|k_1|, |k_2|). \tag{11}$$



(a) (b)

**Fig. 2** (a) White points show the high curvatures of a "cookie" object. (b) "cookie" object is cut along the edge by a 1-degree IP.

What we are now interested in are the angled regions that contain the Ridge and Valley, namely the region where $\kappa_{max}$ is large and $\kappa_{min}$ is relatively small. As an example shown in Fig. 2 (a), the $\kappa_{max}$s on the edge vertices of "cookie" object have high values and their $\kappa_{min}$s have relatively low values; thus, the edge shown with white points is a ridge.

#### 3.3.2 Cutting the curved region

After finding the high-curved regions, the rest of the task is to cut them into low-curved regions. Following the procedure discussed above, we can extract the points existing on the high-curved place by a function

$$[\kappa_{max}^j, \ \kappa_{min}^j] = c(\boldsymbol{x}_j, f_i), \tag{12}$$

where vertex $\boldsymbol{x}_j \in R_i$; $R_i$ is the found curved region, and $f_i$ is the corresponding IP function of region $R_i$. The function $c$ returns the maximum/minimum absolute curvatures $\kappa_{max}^j$, $\kappa_{min}^j$ at $\boldsymbol{x}_j$. We extract the points if they satisfy the constraint:

$$\kappa_{max}^j / \kappa_{min}^j > K, \tag{13}$$

where $K$ is a certain threshold for the ratio of maximum and minimum absolute curvatures. Thus we call the extracted points valley or ridge.

From the knowledge of differential geometry, we

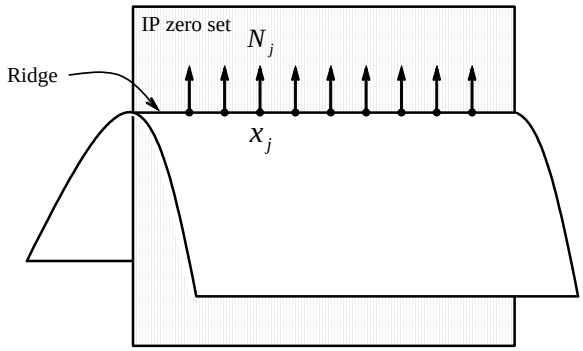



**Fig. 3** An example of cutting the curved region with a new IP

understand that for an $n$-degree IP region, there might be continuous valley/ridge curves that can be fitted with an $(n-1)$-degree IP.

Using the information of the vertices and their normals on the valley/ridge curves, we try to fit a new IP, and cut the curved region again with this IP. A simple example is shown in Fig. 3. If we let vertices on the ridge and the points along their normals be the points passed through by the IP zero set, we can fit a new IP whose zero set crosses the ridge curve. Thus, according to the sign of each point measured by the IP function, the region can be cut into two parts, the inner part and the outer part with (7).

Here we choose the renormalization method [15] for fitting the new IP. The reason for not choosing the 3L method is that it is not easy to obtain the information of connectivity between the extracted high-curved points. Thus the tangent at each point cannot be calculated for the generation of the other 2 layers. The Fig. 2 (b) shows the cutting result of "cookie". Along the ridge of the "cookie", the new IP shown as a plane in this case, is obtained. And according to the signs (7), the "cookie" is cut into two parts in the different sides of the plane.

### 3.4 Merging procedure

The task of the merging procedure is to merge the over-segmented regions to an integral region. The regions resulting from the unfit/high-curved cutting procedure may be over-segmented, as in the simplest case shown in Fig. 1. Although the resulting regions, Nos. II, III, V, and VI, were fitted well by 1-degree IPs respectively and outputted as independent regions, No. II and No. V should belong to the same region and served by the same IP. In the regions No. III and No. VI, the situation is the same. For solving this problem, we designed a procedure to merge the over-segmented regions.

This procedure makes use of the region-grow strategy [23] where the seed region is selected and then is merged with its neighbors if possible. The criterion for judging whether a neighbor should be merged into the seed region is the same as the constraint (6). The difference is that we use the seed's IP to measure the neighbors. Thus the measurement described in (4) and (5) should be replaced by:

$$e_i = \frac{\mid f_{seed}(\boldsymbol{x}_i) \mid}{\parallel \bigtriangledown f_{seed}(\boldsymbol{x}_i) \parallel}, \quad x_i \in \{R_{neighbor}\}, \tag{14}$$

$$\boldsymbol{n}_i = \frac{\bigtriangledown f_{seed}(\boldsymbol{x}_i)}{\parallel \bigtriangledown f_{seed}(\boldsymbol{x}_i) \parallel}, \quad x_i \in \{R_{neighbor}\}, \tag{15}$$

where $f_{seed}$ is the IP function corresponding to the seed region, and $R_{neighbor}$ is a neighbor region of the seed region. The constraint in (6) is still used as the measurement of the similarity between a seed IP and its neighbor. In this procedure, the larger region is first to be chosen as a seed region. In short, the algorithm is described as follows:

**Algorithm 2: Merging procedure**

Input $\{\mathcal{R}\}$: segmented regions; $\{IP\}$: IPs corresponding to the regions; $T_1$ and $T_2$: thresholds
Output $\{\mathcal{R}\}$, $\{IP\}$
Working Variables *mergedFlag*: a flag checking whether merging occurred in the loop; $\{\hat{R}\}$: a stack preserving the regions
Initialization
  $\{\hat{R}\} \leftarrow \{\mathcal{R}\}$;
Main Loop
  **while** $\{\hat{R}\}$ is not empty **do**:
    Find the largest region $R_{max} \in \{\hat{R}\}$ and its corresponding IP $IP_{max}$;
    $\{R_{neighbor}\} \leftarrow \boldsymbol{FindNeighbors}(R_{max})$;
    $mergedFlag \leftarrow false$;
    **for** $R_i \in \{R_{neighbor}\}, i = 1, 2, \ldots$ **do**:
      $[Dd, Ds] \leftarrow \boldsymbol{Measuring}(R_i, IP_{max})$;
      **if** Dd $< T1$ and Ds $> T2$
        $[R_{max}] \leftarrow merge(R_i, R_{max})$;
        $[IP_{max}] \leftarrow \boldsymbol{3LFitting}(R_{max})$;
        $mergedFlag \leftarrow true$;
        Updating $\mathcal{R}$ with $R_{max}$ and removing $R_i$ from $\mathcal{R}$
      **end**
    **end**
    **if** *mergedFlag* is false
      pop $R_{max}$ from $\{\hat{R}\}$;
    **else**
      $\{\hat{R}\} \leftarrow \{\mathcal{R}\}$;
    **end**
  **end**

Here, ***FindNeighbors*** is a function for finding the neighbors of a certain region.

## 4. Experimental results

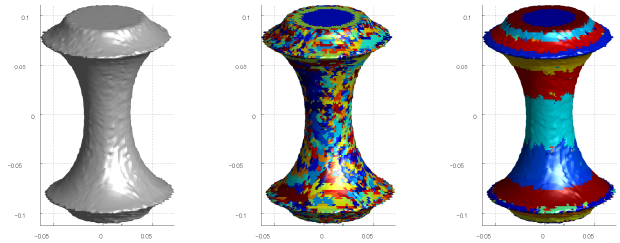
In this section, we first introduce a simple experiment step by step, and then we report some other results.

### 4.1 A simple experiment

To clear the effectiveness of each step of our method, let us show an example of how to segment a noisy model shown in Fig. 4 (a) step by step.

#### 4.1.1 Experiments with unfit cutting procedure

In the first step, unfit cutting procedure (Algorithm 1), the two thresholds $T_1$ and $T_2$ are essential for adjusting the accuracy and the number of segments. It is given that the more tolerant $T_1$ and $T_2$ are, the more dissimilar the IP is to a region. According to a certain demand, if high accuracy is required, $T_1$ should be set close to 0 and $T_2$ close to 1, and vice versa. But note that the immoderate setting may lead to oversegmentation or undesired inaccuracy. In practice, for choice of the thresholds, the ratio between a scale of object and degree of data noise should be considered. Also essentially the choices should be different according to the applications.



(a) (b) (c)

**Fig. 4** The results of the unfit cutting procedure with different thresholds. (a) Original range data. (b) 4365 segments. (c) 267 segments.

Let us show an example of differing choices. The target object shown in Fig. 4 (a) is segmented with results of different thresholds. Fig. 4 (b) shows the result of 4365 segments, in the case of setting $T_1 = 0.0001$ and $T_2 = 0.9$, while Fig. 4 (c) shows the result of 267 segments, in the case of setting $T_1 = 0.001$ and $T_2 = 0.75$. Because of the coarse surface and the strict thresholds, the resulting segments in the former case are cut into small pieces to be suited for locally high accuracy. On the other hand, in the latter case, because of the tolerant thresholds, the resulting segments are larger and more noises are admissible into the regions to be suited for a relatively global accuracy.

#### 4.1.2 Experiments with high-curved cutting procedure

Although the results in Fig. 4 (b) and (c) achieve IP fitting with desired accuracy, high-curved regions still exist, as shown in Fig. 5. Considering easy visualization of the result, we choose the result shown in Fig. 4 (c).

As shown in Fig. 5, first, the high-curved cutting procedure checks out the high-curved regions (shown in yellow) by setting $K = 10$ in constraint (13). Note the threshold setting refers to the curvature radius estimated from scale of object by experience.

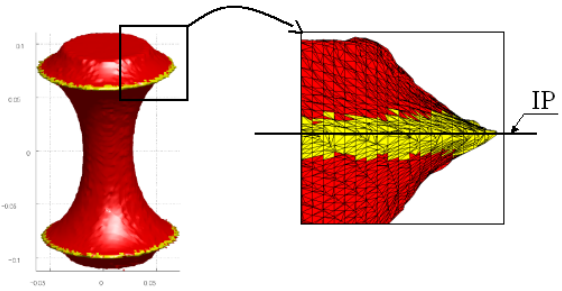



**Fig. 5** Cutting the high-curved regions (yellow regions)

Second, this procedure cuts this high-curved region into two low-curved regions so as to improve the result. By extracting the points with high curvature in the high-curved regions using the constraint (13), a new IP is obtained as shown in the right part of Fig. 5. Finally the new IP divides the high-curved region into two parts in its different sides respectively. Thus the 267 segments in Fig. 4(c) become 269 segments.

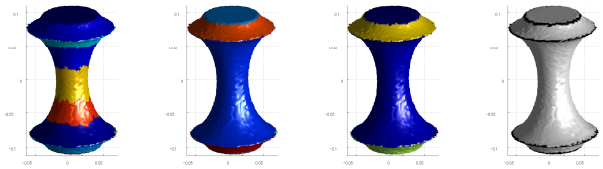
#### 4.1.3 Experiments with merging procedure

The merging procedure is designed to merge similar IPs for further effective representation. Let us show this effectiveness by merging the result of the 269 segments derived from the high-curved cutting procedure.

The example shown in Fig. 6 (a) is the case where the thresholds are set the same as the cutting procedure with $T_1 = 0.001$ and $T_2 = 0.75$. Fig. 6 (b) shows the result obtained by setting $T_1 = 0.007$ and $T_2 = 0.7$. In these two cases, the 2-degree IPs are used to solve the fitting problem of the $3L$ method. We also give the merging case of 4-degree IPs shown in Fig. 6 (c) with settings of $T_1 = 0.001$ and $T_2 = 0.75$. Fig. 6 (c) obtains the same result as Fig. 6 (b). However, since 4-degree IPs can describe more complex surfaces than 2-degree IPs, merging by 4-degree IPs can be done with higher accuracy but is also more time-consuming than merging by 2-degree IPs. Fig. 6 (d) shows the black boundary points of the regions extracted from (b) or (c).

#### 4.1.4 Visualization of IP segments

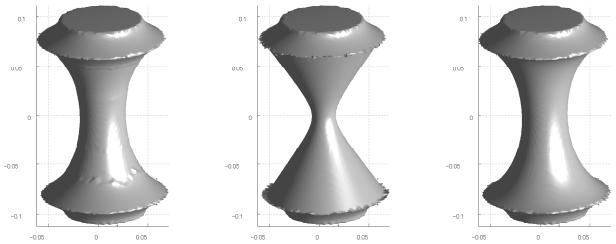
Having obtained the segmentation result, segmented regions and their one-to-one corresponding IPs, now let



(a) (b) (c) (d)

**Fig. 6** The results of the merging procedure with different thresholds. (a) 9 segments. (b) 5 segments with 2-degree IPs. (c) 5 segments with 4-degree IPs. (d) boundaries.

us show the segments with IP zero sets. To simplify the visualization of the IP segmentation result, we choose a simple way that projects the original mesh onto the zero set surfaces of IP segments. But we refer the interested reader to [21], [28] for more details about mesh reconstruction for implicit functions.



(a) (b) (c)

**Fig. 7** IP surfaces representation. (a) 9 segments with 2-degree IPs. (b) 5 segments with 2-degree IPs. (c) 5 segments with 4-degree IPs.

Fig. 7 (a)(b)(c) show the IP surfaces (zero sets) from the segmentation results shown in Fig. 6 (a)(b)(c) respectively. Note although Fig. 6 (b) shows the same segmentation result as Fig. 6 (c), their IP surfaces are rather different. Obviously the 4-degree IPs represent more accurate surfaces than 2-degree IPs.

### 4.2 Other results

In this subsection, we show some other experimental results to prove the effectiveness of our method.

A simple example of segmented IP surface representation of a cube is shown in Fig. 8; the shape of the cube is contaminated by adding noises to one percent of its edge length. Six segmented IP surfaces for the cube are shown in Fig. 8 (b), and its corresponding segmentation result is shown in Fig. 8 (c).

Other complex examples are shown in Figs. 9-11. The original range data are shown in column (a) respectively. Then each of these images is represented with IP surfaces in different segmentation levels, by changing the thresholds of (6) and (13), shown in columns (b) and (c) of each figure. The figures in column (d) show the segmentation results corresponding to the IP surfaces shown in column (c) respectively.

## 5. Applications

To clarify the effectiveness of our segmentation results, here we show some simple applications based on the aspects of rendering, 3D recognition, and registration. By these applications, we demonstrate various possible uses for our method.

### 5.1 An improved rendering method

The problem frustrating our results of the application for rendering is the errors existing on the boundaries of the segments. Especially when the threshold $T_1$ and $T_2$ are too tolerant, the fitting accuracy is accordingly decreased, and thus the boundaries connecting two neighbor segments become more coarse (see Fig. 9 (c)). To solve this problem, we propose a method based on smoothing the segment boundaries. First we define a metric from a point to the boundary of the segment which the point belongs to as:

$$d(\boldsymbol{x}) = \min_{\boldsymbol{b}_i \in Bdry} \| \boldsymbol{x} - \boldsymbol{b}_i \|, \tag{16}$$

where $\boldsymbol{x}$ is the point in a segment and $\boldsymbol{b}_i$ is the point at the boundary of the corresponding segment. Then by defining two weights $w_1$ and $w_2$, we can combine the current IP and its neighbor IP to one implicit function as

$$F(\boldsymbol{x}) = \frac{w_1 * f + w_2 * f_{neighbor}}{w_1 + w_2}, \tag{17}$$

where $f$ is the IP function of the current segment, $f_{neighbor}$ is the IP function of a neighbor segment connected by the nearest boundary point. The weights $w_1$ and $w_2$ can be seen as two Gaussian-like functions respective to $d(\boldsymbol{x})$, such as those shown in Fig. 12.

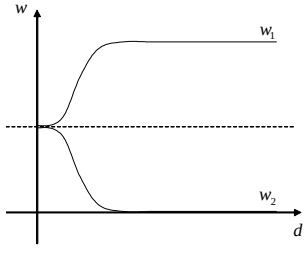
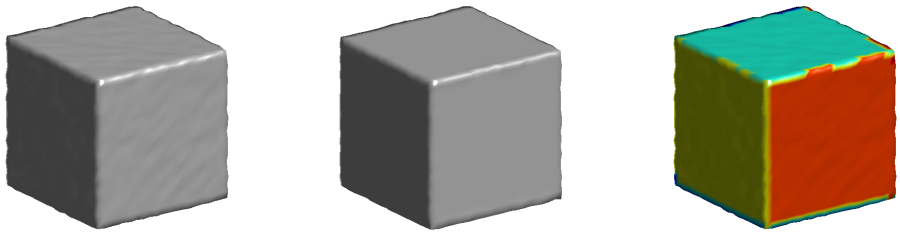


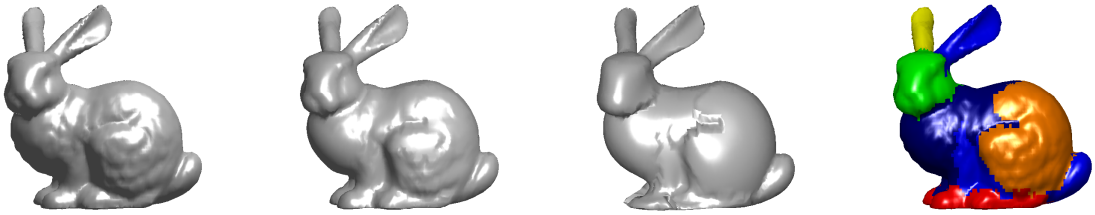

**Fig. 12** Weight functions

We show an example in Fig. 13. Fig. 13 (a) shows a segment result the same as that in Fig. 9(d). Fig. 13 (b) shows the weight calculated by (17) according to the boundaries. Fig. 13 (c) shows the rendering result by directly rendering the segments without using (17). And Fig. 13 (d) shows the rendering result by using the weighted function in (17).



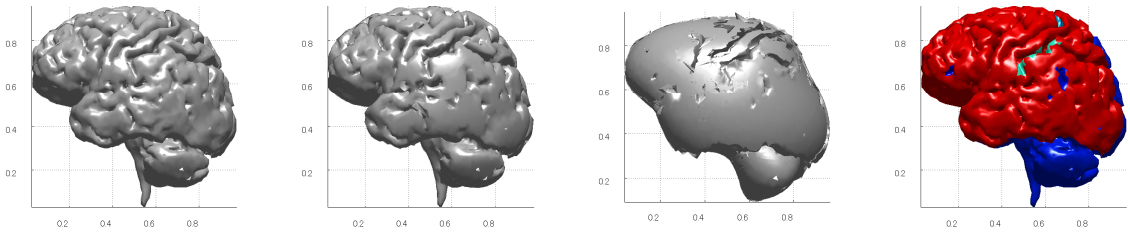
(a) (b) (c)

**Fig. 8** (a) Original range data. (b) 6 2-degree IP surfaces. (c) 6 segments with different colors referring to (b).



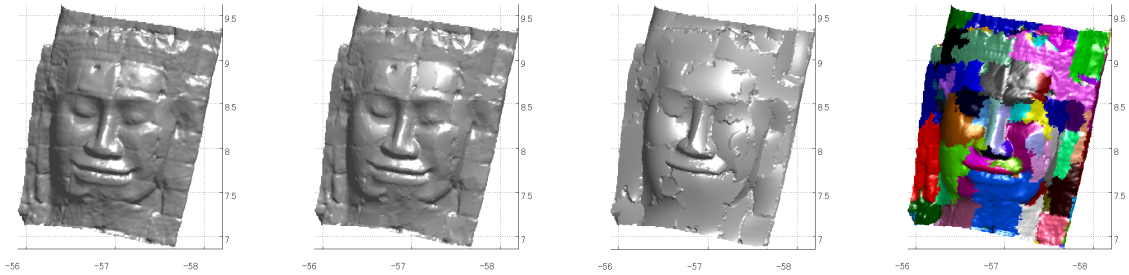
(a) (b) (c) (d)

**Fig. 9** (a) Original range data. (b) 100 4-degree IP surfaces. (c) 8 4-degree IP surfaces. (d) 8 segments referring to (c).



(a) (b) (c) (d)

**Fig. 10** (a) Original range data. (b) 323 4-degree IP surfaces. (c) 19 4-degree IP surfaces. (d) 19 segments referring to (c).



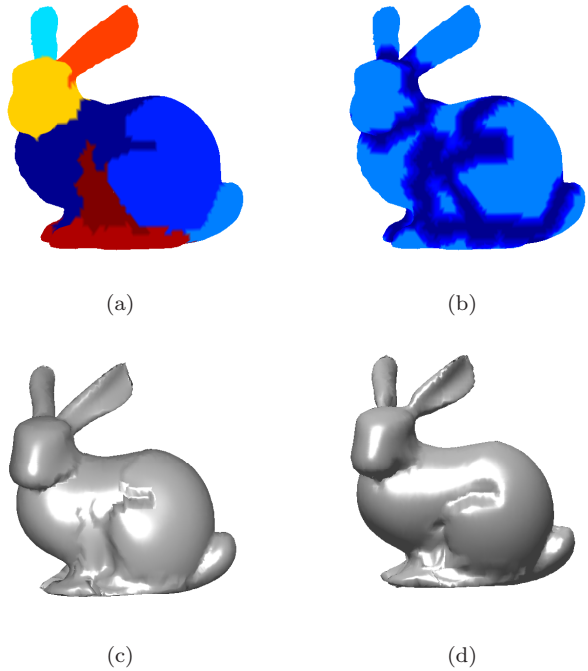
(a) (b) (c) (d)

**Fig. 11** (a) Original range data. (b) 763 4-degree IP surfaces. (c) 113 4-degree IP surfaces. (d) 113 segments referring to (e).

### 5.2 An application for 3D recognition

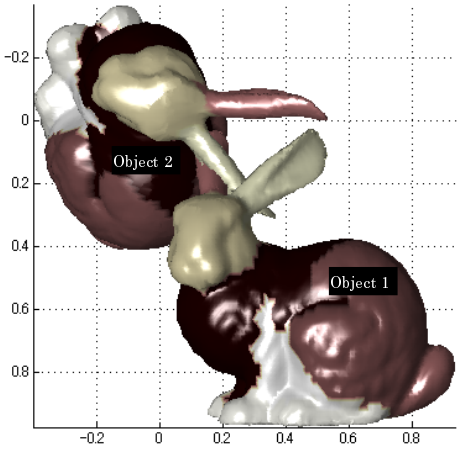
One of the most important properties of IP is its algebraic invariants, which has been introduced by Taubin and Cooper [11]. They pointed out that some Euclidean invariants can be extracted from the coefficients of the leading form of an IP by certain symbolic calculations. The invariants enable us to apply our segmented results to 3D feature matching or recognition. A recent survey on 3D recognition is given by [3].

An example is given in Fig. 14. First, we took the



(a) (b) (c) (d)

**Fig. 13** Rendering example by our improved method. (a) Segment result. (b) Weight calculation according to boundary. (c) Rendering without the weight function. (d) Rendering with the weight function.




(a)

| Segments | Object 1 | Object 2 |
|---|---|---|
| | 1.0e+004 * (0.3009, 0.8786, 2.1953) | 1.0e+004 * (0.3009, 0.8786,2.1953) |
| | 1.0e+004 * (0.7127, 2.0522, 6.2591) | 1.0e+004 * (0.7127, 2.0522, 6.2591) |
| | 1.0e+007 * (0.0157, 0.0513, 1.7007) | 1.0e+007 * (0.0157, 0.0513, 1.7007) |
| | 1.0e+008 * (0.0213, 0.2080, 1.7999) | 1.0e+008 * (0.0213, 0.2080, 1.7999) |
| | 1.0e+005 * (0.7364, 1.5098, 8.0651) | 1.0e+005 * (0.7364, 1.5098, 8.0651) |
| | 1.0e+008 * (0.0180, 0.2021, 2.1230) | 1.0e+008 * (0.0180, 0.2021, 2.1230) |
| | 1.0e+008 * (0.0006, 0.0028, 2.1964) | 1.0e+008 * (0.0006, 0.0028, 2.1964) |
| | 1.0e+009 * (0.0004, 0.0026, 2.7701) | 1.0e+009 * (0.0004, 0.0026, 2.7701) |

(b)

**Fig. 14** Matching two segmented objects: (a) Object 1 is the original and Object 2 is Euclidean transformed. (b) The calculated invariants of each segment of the two segmented objects.

Euclidean transformation (rotation and translation) of the original object shown in Fig. 9 (a), and the transformed object is shown in Fig. 14 (a) marked as Object 2. Second, we segmented both objects with the same parameters as the example shown in Fig. 9 (d). We obtained the same two segmented results: 1) both of them are segmented into 8 segments, and 2) both of their segments are one-to-one responded in different Euclidean coordiantes. Third, we calculated the Euclidean invariants shown in Fig. 14 (b). Here the invariants are calculated as the eigenvalues of $F_{[3,1]}^T F_{[3,1]}$, and this symbol notation and calculation are as noted in [11].

This example shows two important conclusions: 1) our segmentation method is Euclidean invariant, and 2) the segmented regions of the both are robustly one-to-one matched by using the invariants extracted from IP coefficients. Although large numbers of examples are needed to prove the effectiveness, it opens up new vistas for 3D matching for recognition.

### 5.3 Application for 3D registration

We extend the previous result to the application of 3D registration (alignment). 3D registration is one of the important sub-steps for large-scale 3D modeling. Most of these registrations need to manually align the corresponding scans for the initial process, and then use an ICP-based algorithm such as [29] to achieve further accuracy in alignment. We refer this survey to [30].

Taubin and Cooper gave a simple method for calculating IP's orientation and center of mass from the coefficients of IP's leading form [11], which made the registration (or pose estimation) fast and simple. Fortunately, this method can be applied to the results of our segmentation method.

Fig. 15 shows an example of this application, where we take half of the original "bunny" and transform it to another position using random Euclidean transform, and then test the registration between it and the original one. The process can be described in three steps: 1) **Segmentation**: segmenting the objects with the same thresholds; 2) **Matching**: calculating and finding the matching pairs in the same way as in the recognition application; 3) **Registration**: calculating the orientation and center of the matched segments by Taubin's method [11] and then estimating the transformation.

There are some errors in the registration results, since our segmentation method cannot give exactly the same segments between the original "bunny" and the partial "bunny." But this result is acceptable as the result of the initial alignment process (coarse registration) to be succeeded by a fine registration such as the ICP process [29]. The advantages of this method are that it is fully automatic, fast, simple and free to any point-to-point corresponding procedures.

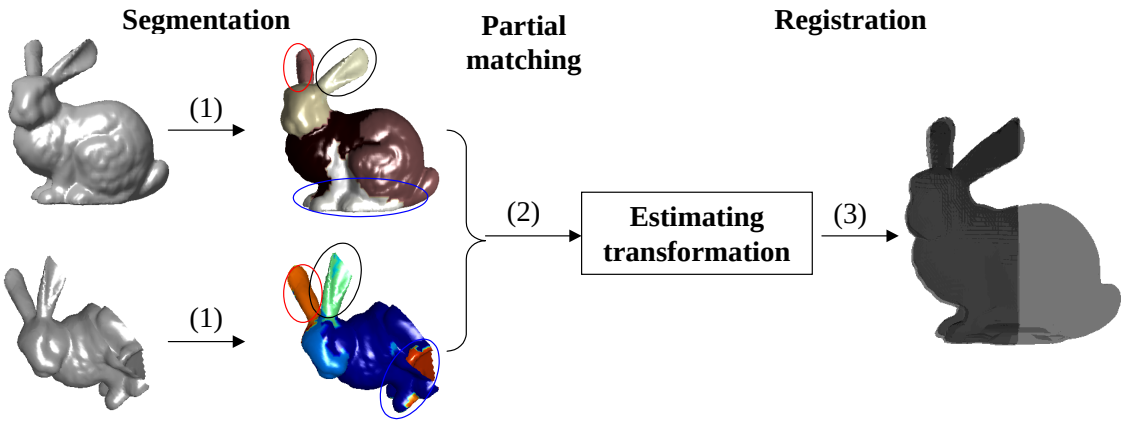



**Fig. 15** Partial 3D registration

## 6. Conclusions

In this paper, we bring some major contributions to the field of implicit representation for range data. First, we propose a three-step cut-and-merge approach for obtaining a low-degree IP segment combination. Segmenting a whole object into small parts and then fitting them to moderate lower-degree IPs avoids the time-consuming computation of fitting a high-degree IP to a complex object. As a result, the IPs assigned to segments provide the geometric properties such as curvatures and Euclidean invariants. By adjusting the thresholds using different tolerance levels, IP surfaces with different accuracy and smoothness are obtained. Degrees of accuracy and smoothness can be freely selected so as to be suitable for the requirements for various vision applications.

### References

[1] K. Ikeuchi, K. Hasegawa, A. Nakazawa, J. Takamatsu, T. Oishi, and T. Masuda, "Bayon Digital Archival Project," Proceedings of the Tenth International Conference on Virtual Systems and Multimedia, pp.334–343, 2004.

[2] T. Tasdizen, J.P. Tarel, and D.B. Cooper, "Improving the Stability of Algebraic Curves for Applications," IEEE Trans. on Image Processing, vol.9, no.3, pp.405–416, 2000.

[3] B. Bustos, D. a. Keim, D. Saupe, T. Schreck, and D. v. Vrani'c, "Feature-Based Similarity Search in 3D Object Databases," ACM Computing Surveys, vol.37, no.4, pp.345–387, 2005.

[4] A. Goshtasby, "Design and Recovery of 2-D and 3-D Shapes Using Rational Gaussian Curves and Surfaces," Int. J. Computer Vision, vol.10, no.3, pp.233–256, 1993.

[5] G. Turk and J.F. O 'Brieni, "Variational Implicit Surfaces," Technical Report GIT-GVU-99-15, Graphics, Visualization, and Useability Center, 1999.

[6] R.J. Campbell and P.J. Flynn, "A Survey Of Free-Form Object Representation and Recognition Techniques," Computer Vision and Image Understanding, vol.81, pp.166–210, 2001.

[7] J.P. Tarel and D.B. Cooper, "The Complex Representation of Algebraic Curves and Its Simple Exploitation for Pose Estimation and Invariant Recognition," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol.22, no.7, pp.663–674, 2000.

[8] J.P. Tarel, H. Civi, and D.B. Cooper, "Form 3d objects without point matching using algebraic surface models," In Proceedings of IEEE Workshop Model Based 3D Image Analysis, pp.13–21, 1998.

[9] N. Khan, "Silhouette-Based 2D-3D Pose Estimation Using Implicit Algebraic Surfaces," Master Thesis in Computer Science, Saarland University, 2007.

[10] C. Unsalan, "A Model Based Approach for Pose Estimation and Rotation Invariant Object Matching," Pattern Recogn. Lett., vol.28, no.1, pp.49–57, 2007.

[11] G. Taubin and D. Cooper, Symbolic and Numerical Computation for Artificial Intelligence, chapter 6, Computational Mathematics and Applications. Academic Press, 1992.

[12] T. Sahin and M. Unel, "Fitting Globally Stabilized Algebraic Surfaces to Range Data," Proceedings of the Tenth IEEE International Conference on Computer Vision, vol.2, pp.1083–1088, 2005.

[13] D. Keren and D. Cooper, "Describing Complicated Objects by Implicit Polynomials," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol.16, no.1, pp.38–53, 1994.

[14] G. Taubin, "Estimation of Planar Curves, Surfaces and Nonplanar Space Curves Defined by Implicit Equations with Applications to Edge and Range Image Segmentation," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol.13, no.11, pp.1115–1138, 1991.

[15] K. Kanatani, "Renormalization for Computer Vision," The Institute of Elec., Info. and Comm. eng. (IEICE) Transaction, vol.35, no.2, pp.201–209, 1994.

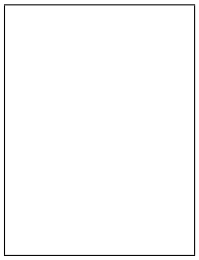
[16] M. Blane, Z.B. Lei, and D.B. Cooper, "The 3L Algorithm for Fitting Implicit Polynomial Curves and Surfaces to Data," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol.22, no.3, pp.298–313, 2000.

[17] A. Helzer, M. Barzohar, and D. Malah, "Stable Fitting of 2D Curves and 3D Surfaces by Implicit Polynomials," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol.26, no.10, pp.1283–1294, 2004.

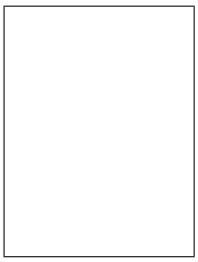
[18] T. Srinark and C. Kambhamettu, "A Novel Method for 3D Surface Mesh Segmentation," Proceedings of the 6th IASTED International Conference on Computers, Graphics, and Imaging, pp.212–217, 2003.

[19] H. Yamauchi, S. Gumhold, R. Zayer, and H.P. Seidel, "Mesh Segmentation Driven by Gaussian Curvature," Pacific Graphics 2005, Visual Computer, vol.21, no.8-10, pp.649–658, 2005.

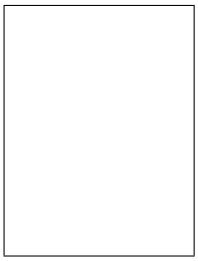
[20] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Moller, "Curvature-based transfer functions for direct volume rendering: methods and applications," Visualization, 2003. VIS 2003. IEEE, pp.513– 520, 2003.
[21] B. de Araujo and J. Jorge, "Curvature dependent polygonization of implicit surfaces," Computer Graphics and Image Processing, 2004. Proceedings, pp.266– 273, 2004.
[22] M.W. Jones, J.A. Baerentzen, and M. Sramek, "3D Distance Fields: A Survey of Techniques and Applications," IEEE Trans. on Visualization and Computer Graphics, vol.12, no.4, pp.581–599, 2006.
[23] S. Petitjean, "A Survey of Methods for Recovering Quadrics in Triangle Meshes," ACM Computing Surveys, vol.34, no.2, pp.211–262, 2002.
[24] A. Shamir, "A Formulation of Boundary Mesh Segmentation," Second International Symposium on 3D Data Processing, Visualization and Transmission, pp.82–89, 2004.
[25] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.P. Seidel, "Multi-level Partition of Unity Implicits," ACM Transactions on Graphics (SIGGRAPH 2003), vol.22, no.3, pp.463–470, 2003.
[26] Y. Ohtake, A. Belyaev, and M. Alexa, "Sparse Low-degree Implicit Surfaces with Applications to High Quality Rendering, Feature Extraction, and Smoothing," Proc. of Eurographics Symposium on Geometry Processing 2005, pp.149–158, 2005.
[27] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C.T. Silva., "Computing and rendering point set surfaces," IEEE Transactions on Visualization and Computer Graphic, vol.9, no.1, pp.3–15, 2003.
[28] W. Lorensen and H. Cline, "Marching cube: A high resolution 3-D surface construction algorithm," Computer Graphics Proceedings, vol.21, no.4, pp.163–169, 1997.
[29] T. Oishi, A. Nakazawa, R. Kurazume, and K. Ikeuchi, "Fast Simultaneous Alignment of Multiple Range Images using Index Images," Proc. The 5th International Conference on 3-D Digital Imaging and Modeling (3DIM 2005), pp.476–483, 2005.
[30] J. Salvi, C. Matabosch, D. Fofi, and J. Forest, "A review of recent range image registration methods with accuracy evaluation," Image and Vision Computing, vol.25, no.5, pp.578–596, 2007.



**Bo Zheng** He graduated from Beijing University of Technology, Beijing, China, in 2001. He received the M.S. degree from the Department of Applied Physics School of Engineering, the University of Tokyo, Japan, in 2005. He worked for Fujitsu CO., LTD., Japan, during the Summer of 2005. He is currently pursuing the Ph.D. degree in graduate school of Information Science and Technology, The University of Tokyo. His research interests include computer vision, pattern recognition, algebraic geometry, and Krylov subspace method.



**Jun Takamatsu** He received the Ph.D. degree in Computer Science from the University of Tokyo, Japan, in 2003. At present, he is a project lecturer at the Institute of Industrial Science, the University of Tokyo. His research interests include 3D shape analysis, physics-based vision, and robotics including learning-from-observation, task planning using feasible motion analysis.



**Katsushi Ikeuchi** He received the Ph.D. degree in Information Engineering from the University of Tokyo, Tokyo, Japan, in 1978. After working at the Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Electrotechnical Laboratory of Japanese Ministry of International Trade and Industry, and the School of Computer Science, Carnegie Mellon University, he joined the university in 1996. At present, he is a Professor at the Interfaculty Initiative in Information Studies Graduate School of Interdisciplinary Information Studies, the University of Tokyo. His research interests include modeling-from-reality, object recognition, and learning-from-observation. He has received several awards, including the David Marr Prize in computational vision, and IEEE R&A K-S Fu memorial best transaction paper award. In addition, in 1992, his paper, "Numerical Shape from Shading and Occluding Boundaries," was selected as one of the most influential papers to have appeared in Artificial Intelligence Journal within the past ten years. His IEEE activities include General Chair, IROS95, ITSC00, IV01; Program Chair, CVPR96, ICCV03; Associate Editor, IEEE TRA, IEEE TPAMI; distinguished lecture SPS (2000-2002) , RAS (2004-2006). Dr. Ikeuchi was selected to IEEE Fellow in 1998. He is the EIC of International Journal of Computer Vision.