Fast Segmentation of Range Images into Planar Regions by Scan Line Grouping

X. Y. Jiang, H. Bunke
Institute of Informatics and Applied Mathematics
University of Berne, Switzerland

Abstract

In this paper we present a novel technique for rapidly partitioning surfaces in range images into planar patches. Essential for our segmentation method is the observation that, in a scan line, the points belonging to a planar surface form a straight line segment. On the other hand, all points on a straight line segment surely belong to the same planar surface. Based on this observation, we first divide each scan line into straight line segments and subsequently consider only the set of line segments of all scan lines as segmentation primitives. We have developed a simple link-based data structure to efficiently represent line segments and their neighborhood relationship. The principle of our segmentation method is region growing. Three neighboring line segments satisfying an optimality criterion are selected as a seed region, and then a growing is carried out around the seed region. We use a noise variance estimation to automatically set some thresholds so that the algorithm can adapt to the noise conditions of different range images. The proposed algorithm has been tested on real range images acquired by two different range sensors. Experimental results show that the proposed algorithm is fast and robust.

CR Categories and Subject Descriptors: I.4.6 [Image Processing]: Segmentation; I.4.8 [Image Processing]: Scene Analysis.
General Terms: Algorithms.
Additional Key Words: Range data, partitioning, region growing, planar surfaces.

1 Introduction

Recently, three-dimensional vision techniques based on range data have been receiving much attention in computer vision and robotics. Particularly, in such areas as industrial and navigational robotics where a robot must be capable of handling 3-D environment, range information plays a vital role. The development of advanced computer vision systems involves improvements in both data acquisition technology and algorithmic approaches. One advance in data acquisition is the development of direct and effective indirect range imaging techniques [2, 13, 18]. Raw range images in their full dimensionality, however, are just a list of numbers, and can not directly support high-level scene interpretation processes without some type of perceptional organization. One way of perceptional organization is the segmentation of range images into surface patches.

Roughly speaking, the segmentation algorithms known in the literature can be classified into general and dedicated approaches. In a general approach, only general knowledge about surfaces is used to compute a complete segmentation and reconstruction. The algorithm proposed by Besl [3], for instance, assumes only that the range data may be interpreted as noisy samples of a piecewise-smooth surface. On the contrary, dedicated approaches search for particular structures in range data, such as planes, cylinders, cones or solids of revolution. Examples of this class of approaches are [4, 16, 26, 30].

In this work we consider the segmentation of range images into planar regions. Our proposed approach belongs to the second category, i.e., it is a dedicated approach applicable only for planar surfaces. However, for a particular problem, a dedicated approach can be more useful than a general method, mainly for reasons of efficiency. Moreover, the partition into planar regions is perhaps only a possible preprocessing step in a high-level region description approach. By deriving higher order surfaces from the first-order planar regions, the final result is a hierarchical scheme capable of representing regions of arbitrary complexity.

1.1 Segmentation of range images into planar regions

A few algorithms are known in the literature for segmenting range images into planar regions. Yang and Kak [29], for instance, have taken a region growing approach. As their goal was to determine the identity, position and orientation of the topmost object in a pile, they started with an 8×8 topmost region. If this is planar, then it is used as a seed to grow to the boundary of the planar region.

In [20], the split-and-merge paradigm by Horowitz and Pavlidis [12] has been extended from greylevel to range imagery. The approach in [28] is an enhanced version of the split-and-merge technique. It differs from the approach in [20] in a number of ways. In particular, the authors have incorporated a more sophisticated technique for surface parametrization, the use of range continuity checks during the split-and-merge phases, and the option of multiple merge phases.

Traditionally, quadtrees are used in the split-and-merge algorithms as the geometric data structure which serves both for the tessellation of the image domain and the neighborhood referencing. Examples are the methods described in [12, 20, 28]. As a drawback, the quadtree structure often results in some false region boundaries. In [25], the authors used another data structure of a more geometric nature, namely the Delaunay triangulation defined by a set of triangles tessellating the image domain. This data structure, combined with an adaptive surface approximation technique, gives region edges which adapt better to the surface boundaries in the range image. Experimental results were given for three real 256×256 range images, using a VAX 8555 computer. The segmentation takes 79, 104, and 125 seconds, respectively.

Clustering represents another class of algorithms for segmenting range images into planar regions. In any clustering algorithm, two important aspects have to be considered, namely the feature space in which the clustering takes place, and a strategy for partitioning the feature space into clusters without a priori information about the actual number of clusters. In [14], the authors define the feature vector of a pixel (x, y) to be the parameters of the facet model $z = \beta_0 + \beta_1 x + \beta_2 y + \beta_3 x^2 + \beta_4 xy + \beta_5 y^2$ in a small local $(2L + 1) \times (2L + 1)$ neighborhood centered on (x, y). (For the planar facet model $\beta_3 = \beta_4 = \beta_5 = 0$ is used.) The parameters β_i can be computed by least square estimation. Least square, however, is very sensitive to the presence of outliers, e.g. on a discontinuity. To overcome this problem, the robust M-estimators have been used to compute the feature vector. A clustering algorithm based on the minimum volume ellipsoid robust estimator has been proposed which iteratively partitions the feature space into clusters. As clustering in feature space does not take into account the spatial information, the clusters found must be mapped back into the image and analyzed to get the final segmentation result in the spatial domain. For a synthetic 128×128 cube image, a segmentation time of about 2 minutes has been reported on a HP9000 computer.

Among the numerous clustering algorithms, the adaptive distance dynamic clusters algorithm (ADDC) [6] is of particular interest to the problem of partitioning range images into planar regions. ADDC is a "crisp" (hard) clustering algorithm specially designed to search for clusters that lie in subspaces (such as lines or (hyper)planes) of the original feature space. If we define the feature space to be the range image itself, then the clusters found by ADDC correspond to planar regions. In [15], ADDC has been extended to the fuzzy case (FADDC), and a compatible cluster merging technique has been proposed to find the optimum number of clusters. Unfortunately, this special clustering algorithm is computationally very expensive. It has been tested on two 200×200 range images from ERIM. To reduce the computation time, the authors have thresholded out the background in both images before the algorithm was applied. Thus, the amount of data effectively processed was much smaller than the original image data. Even with this reduced data dimension, a segmentation time of 70 (ADDC) and 600 (FADDC) seconds has been reported on a Sun Sparcstation. Despite the computational expense, the (F)ADDC algorithm

has some advantages over other clustering algorithms, such as that in [14]. As the clustering in (F)ADDC proceeds in the original image space, there is no need for the computation of feature vectors from the range image. Since no feature vectors need to be extracted, discontinuity poses no problem, and the resulting segmentation is very sharp. Finally, the mapping from clusters back into the image domain and the subsequent analysis is no more necessary.

The approach in [17] is quite different from other techniques known in the literature. It is based on the observation that the first-order polynomials which determine the planar surfaces in a range image satisfy the Laplace equation. The segmentation is solved by a relaxation process. The great advantage of this approach is surely its parallel nature. On a conventional computer, however, it runs quite slowly. A worst case of 10 minutes CPU time for 256×256 range images on a μVAX 3400 has been reported.

1.2 Scan line grouping technique

In this work we take the region growing paradigm. Our segmentation algorithm differs from other region growing approaches in a fundamental point. Instead of pixels we use straight line segments as growing primitives. This greatly reduces the data dimension to be handled in the growing process. Together with a simple optimality measure for seed region extraction and a simple test for growing decisions, the use of line segments makes our algorithm very fast.

The idea of scan line grouping is not new. In fact, Pavlidis [22] has proposed a similar approach and applied it to the segmentation of images from a scanning electron microscope. In Pavlidis' algorithm, each scan line $z(x, y_0)$ is divided into J straight line segments whose endpoints are adjusted in order to minimize the error norm of approximation of $z(x, y_0)$ by a set of linear functions. This approach, however, requires a good initial guess of the break points. It is also necessary to estimate the number of line segments J, at least for a few of the scan lines. In [22], an estimate of J was obtained through experimental tests on a few scan lines. A subsequent grouping phase merges line segments into larger regions. Two neighboring line segments are merged if the absolute difference between their slopes is smaller than a threshold.

Our segmentation method is an enhanced version of Pavlidis' algorithm. It differs from his method in a number of ways. First, we use a much simpler method for partitioning the scan lines into straight line segments. Secondly, we extract optimal seed regions for the growing process. Moreover, the similarity criterion for two line segments used in [22] is not feasible and sufficient for our application. Therefore, we will give a modified similarity measure. Finally, we introduce a postprocessing step to get clean edges between regions.

This paper is organized as follows. The next section gives an overview of the algorithm. Then all steps of the algorithm are described in detail in subsequent sections. In section 7 we report some experimental results. And finally, a discussion concludes the paper.

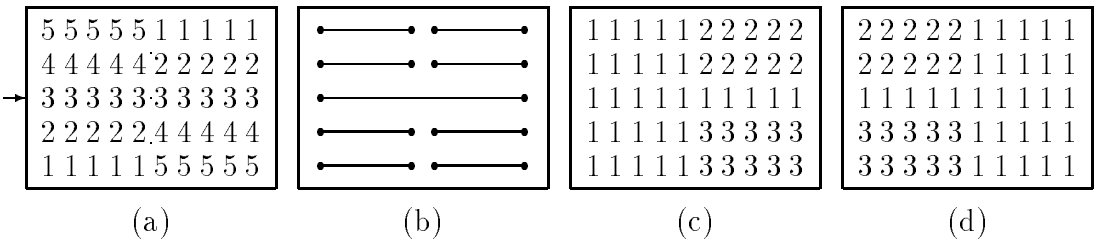

Figure 1: A pathological case: (a) Range image. (b) The extracted line segments. (c) One possible segmentation. (d) Another possible segmentation.

2 Overview of the algorithm

As input to our algorithm we assume that a dense range image $z(x, y)$ is available. That is, the points are sampled on a regular grid. For the sake of description clarity and without loss of generality, we assume furthermore the sampling interval in both x and y direction to be one. Under these two assumptions, a $N \times N$ range image is given by a discrete function $z(x, y), x, y \in I_N = \{1, 2, \ldots, N\}$, which contains N scan lines $z(x, y_0), y_0 \in I_N$.

2.1 Basic idea

Principally, our algorithm is a region growing process based on the straight line segments in the scan lines. The feasibility of the use of line segments as segmentation primitives results from the following observation. A planar surface S can be described by a first-order polynomial

$$z = Ax + By + C. \tag{1}$$

In a scan line $z(x, y_0), y_0 \in I_N$, the points belonging to S are

$$z = Ax + By_0 + C = Ax + B_0 \tag{2}$$

that clearly form a straight line segment in the $x - z$ plane. Different line segments of S have the same slope A but different intercepts B_0 which are dependent on y_0. On the other hand, all points on a straight line segment in the $x - z$ plane surely belong to the same planar surface. Thus, instead of an individual pixel, we can treat a complete line segment as an atomtic entity. In the growing process we must only decide whether a new line segment can be added to the surface patch already approximated or not.

The assertion that all points on a line segment belong to the same planar surface is true, expect for some pathological cases. One such case is shown in Fig. 1. The two possible segmentation results are shown in Fig. 1(c), (d). Neither of them agrees with our expectation. The reason is that the assertion above is violated at the scan

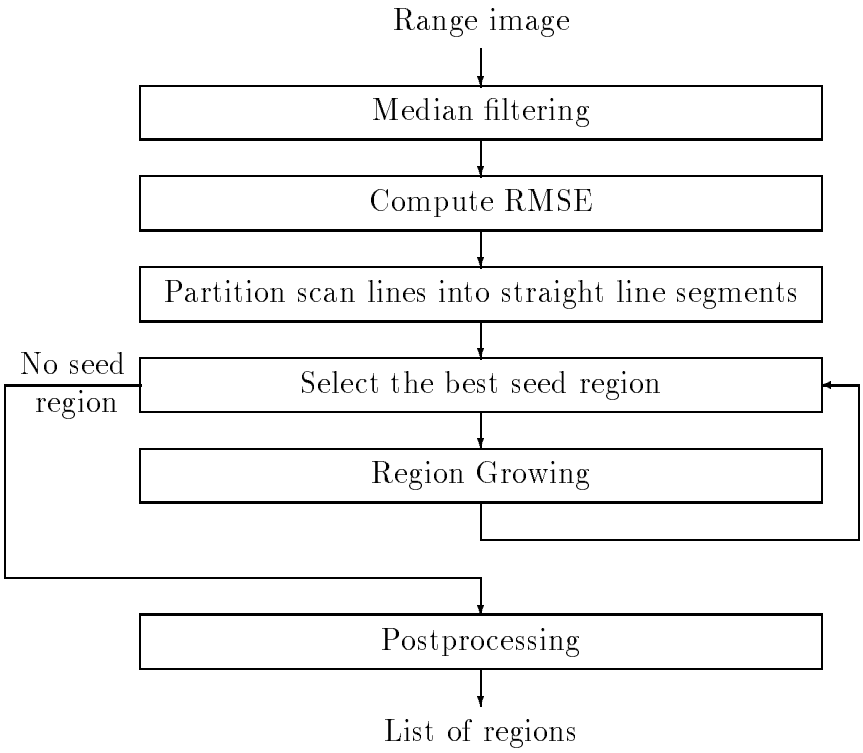

Figure 2: Overview of the segmentation algorithm.

line $y_0 = 3$ where a line segment goes across two planar surfaces. However, this is not an inherent problem of our scan line grouping technique but rather a problem of any segmentation method based solely on function approximation. Pictorial similarity (or homegeneity) is not always a reliable criterion for segmenting an image into regions that correspond to meaningful surfaces of objects. Without additional knowledge, such as the boundary shape of surfaces, this problem is unavoidable. In the following discussion we won't consider this problem for two reasons. First, the occurrence of this kind of pathological cases is extremely rare. Secondly, as one can not expect always to get a perfect segmentation result, the usual practice is to let the segmentation method produce an over-segmented image. Using high-level knowledge, the over-segmentation can then be remedied in a later interpretation phase by merge operations. In our algorithm, the pathological case in Fig 1 will result in an over-segmentation. Thus, the segmentation error introduced through the violation of the fundamental assumption of our scan line grouping technique is not a serious problem.

2.2 Algorithm description

An overview of our algorithm is given in Fig. 2. Real range images suffer from sensor noise. Before the actual segmentation takes place, it would be advantageous

to reduce the noise level of an image while preserving edge information. The Median filter turned out to be a good candidate for this purpose [11]. Thus, we use the Median filter on a 3×3 neighborhood as a standard preprocessing step.

The first step of the algorithm is the partition of each scan line into straight line segments. Through the use of line segments as segmentation primitives, however, we loose the natural neighborhood relationship of the grid structure. To overcome this problem we have developed a simple link-based data structure to provide very efficient access to the neighborhood of each line segment.

The actual segmentation is a region growing process. A small number of line segments satisfying an optimality criterion are selected as a seed region, and the region is expanded around the seed region. For each line segment neighboring the actual region, which can easily be accessed using the link-based data structure, we decide whether it can be added to the planar surface approximated so far. This is done by a very simple test. The growing process is continued until no new line segment can be found surviving the test.

The algorithm above does not always yield clean edges between regions. We introduce a postprocessing step to overcome this problem. The final result is a list of first-order polynomials of all planar regions, and a label image in which each pixel contains the number of the region it belongs to. In the algorithm some thresholds are needed. For all experiments, we use a fixed set of thresholds. The value of each threshold will be given where it is introduced.

3 Noise estimation for threshold selection

In order for the segmentation algorithm to group pixels based on their underlying planar surfaces, it needs to know how well the approximating surface functions should fit the input data. This kind of information should be derived in a data-driven manner from the input data so that the algorithm can automatically adapt to the noise conditions of different range images.

If the noise in the image is approximately stationary, we can compute an estimation of the noise variance σ^2_{img} (that should be applicable at almost all pixels) by averaging the estimates of the noise variance at each pixel. To compute the noise variance at each pixel p, we perform a least-square plane fitting $z = Ax + By + C$ in the 3×3 neighborhood of p. If the pixel lies in the interior portion of a planar surface, the error in the plane fitting will be primarily due to noise. By contrast, if a pixel is at or near to a step discontinuity where plane fitting is always poor, we should discard it. Such a pixel is detected by testing whether the difference between the maximal and minimal range values in the neighborhood is greater than a preset threshold (10 in our experiments). The mean image noise variance can be expressed as

$$\sigma^2_{img} = \frac{1}{|I'|} \sum_{p \in I'} \sigma^2_{W_3}(p) \tag{3}$$

where I' is the set of pixels actually considered, and $\sigma^2_{W_3}(p)$ is the root-mean-square-error (RMSE) of the plane fitting in the 3×3 window W_3 around p

$$\sigma^2_{W_3}(p) \;=\; \frac{1}{9} \sum_{(x,y) \in W_3} (z(x,y) - (Ax + By + C))^2 \tag{4}$$

Although the regions themselves are not known at the time the noise variance is estimated, we get a good approximation of σ^2_{img}. This allows us to automatically set some thresholds used in the algorithm.

4 Extraction and representation of line segments

As a preparation step for the segmentation, each scan line $z(x, y_0), y_0 \in I_N$, is divided into straight line segments. We use the classical splitting algorithm proposed by Duda and Hart [8]. It splits a curve into two parts at the point most distant from the chord between the two end-points of the curve whenever this maximal distance is greater than some preset threshold. The algorithm proceeds recursively until the curve can not be divided further. In our experiments the threshold is set to $1.0 + 0.5\sigma_{img}$.

In order to be useful in our algorithm, the original splitting algorithm has been adapted in two ways. In the context of polygonal approximation of curves, the distance of a point to the chord between the two endpoints of the curve is the perpendicular distance of a point to a straight line. In our case we use the approximation error, i.e. for two endpoints (x_0, y_0) and (x_2, y_0), and a point (x_1, y_0), the distance is computed by $|z(x_1, y_0) - f_{02}(x_1)|$ where f_{02} is the equation of the straight line going through (x_0, y_0) and (x_2, y_0). In a polygonal approximation of a curve, a break point always belongs to two line segments, namely the left and right neighboring line segments. For the purpose of segmentation, however, the line segments should be disjoint. So, we must assign each break point to one of the neighboring line segments. We do this assignment by a simple heuristic. Let (x_0, y_0) be the break point. Then (x_0-1, y_0) $((x_0+1, y_0))$ must be the immediate neighbor of (x_0, y_0) on the left (right) neighboring line segment. If $|z(x_0, y_0) - z(x_0 - 1, y_0)| < |z(x_0, y_0) - z(x_0 + 1, y_0)|$, then we assign the break point to its left and otherwise to its right neigboring line segment.

It is well known that the simple splitting method of Duda and Hart produces sometimes superfluous line segments. Pavlidis and Horowitz [21] tried to solve this problem by introducing a merge step. As stated in [7], however, this algorithm is computationally much more expensive. For complex curves, it produces sometimes worse results than the simple splitting method. As the second stage of our segmentation algorithm is based on region growing which is a special kind of merging, the potential problem with the simple splitting method will automatically be overcome. Hence, we use the algorithm of Duda and Hart without any merging. Consequently,

our scan line partition method is much simpler than that of Pavlidis [22]. However, this simple method does work well as the experimental results in section 7 show.

We denote a line segment by $((x_1, x_2), y_0)$ where y_0 is the y-coordinate of the corresponding scan line and x_1 (x_2) is the x-coordinate of the leftmost (rightmost) pixel of the line segment. Each segment is represented by a record containing x_1, x_2, and the equation of the segment $z = ax + b$ which is determined by the leftmost and rightmost pixel. The y-coordinate y_0 is saved somewhere else as will be explained later.

As stated earlier, our segmentation algorithm follows the region growing paradigm. Because a region growing process always proceeds in the neighborhood of the region found so far, a simple and efficient access of the neighborhood relationship of the segmentation primitives is of crucial importance. In a regular image grid this is guaranteed in a natural way. The 8-neighborhood of a pixel (x, y), for instance, is simply the set $\{(x \pm 1, y \pm 1)\} - \{(x, y)\}$. Through the use of line segments as segmentation primitives, however, we loose the natural neighborhood relationship of the grid structure. In the following we describe a simple link-based data structure which enables a fast access to the neighborhood of each segment.

All line segments of a scan line are stored in a double-linked list in their natural order. Furthermore, we use an array of dimension N in which each entry $y_0 \in I_N$ points to the first line segment of the scan line $z(x, y_0)$. As all line segments of this scan line have the same y-coordinate y_0 which is in fact the index of the array entry, we do not need to store this coordinate explicitly.

Two line segments s and s' are said to be neighbors if there exist $p \in s$ and $p' \in s'$ such that p and p' are neighbors in a 4-neighborhood. Clearly, each line segment $s = ((x_1, x_2), y_0)$ has exactly one left neighbor (except the first segment of a scan line) and one right neighbor (except the last segment of a scan line). They are simply the left and right neighbor in the double-linked list. The number of neighbors in the scan lines $z(x, y_0 \pm 1)$, however, is not constant. We note that a segment $s' = ((x'_1, x'_2), y_0 \pm 1)$ is a neighbor of s if and only if $[x_1, x_2] \cap [x'_1, x'_2] \neq \emptyset$ holds which can be easily tested. Moreover, if s' and s'' on the same scan line are both neighbors of s, then all line segments between s' and s'' are also neighbors of s. Thus, all neighbors of s in the scan line $z(x, y_0 \pm 1)$ build a continuous sequence in the double-linked list, and we need only to store two pointers to the first and last segment of this sequence. (Naturally, they can be the same.)

In summary, all line segments of a scan line are stored in a double-linked list in their natural order. Each line segment $((x_1, x_2), y_0)$ is represented by a record which contains x_1, x_2, the slope a, the intercept b, pointers to the left and right neighbor, and pointers to the first and last neighbor in the scan line $z(x, y_0 \pm 1)$. An array is used to record the first line segment of each scan line. Using this data structure, we can easily go through all line segments of a scan line. For each line segment, its neighboring segments can be found by simply following the pointers.

5 Region growing based on line segments

The region growing paradigm may be stated as: make an initial guess and then iteratively refine the solution. In our algorithm, the initial guess is a small set of line segments. The iterative refinement is based on function approximation and region growing. Once a planar surface has been fitted to the kth group of neighboring line segments, the $(k+1)$th group of segments is obtained by finding all those neighboring segments that are compatible with the fitted planar surface of the previous group. When no new line segment can be found, the iteration terminates yielding an extracted region. The process of seed region finding and region growing is iterated until no seed region more can be found.

5.1 Seed region extraction

It is important to choose reliable seed regions from which the growing process begins. In this work we choose a small number of neighboring line segments as a seed region. Because working with only two line segments is relatively unreliable, we use three neighboring line segments on three different scan lines. As very short line segments are generally not reliable, we define a length threshold (10 pixels in our experiments). If one of the three line segments is shorter than this threshold, we discard this seed region candidate. The total number of potential seed region candidates corresponds to the number of all instances of three neighboring line segments on three neighboring scan lines surviving the length constraint.

Next we have to decide which one is the "optimal" seed region among all the candidates. One possible measure of optimality could be based on a least-square plane fitting for each candidate. Then, the approximation error could be computed, and the candidate with the smallest error selected as the optimal seed region. In this way, however, we need to consider all pixels of the three line segments, for all candidates. It would be thus very time-consuming. Moreover, the advantage of scan line grouping is lost as we consider now individual pixels which are the primitives before the grouping process. In the following we formulate an optimality criterion directly based on line segments.

We recall that, in the scan line $z(x, y_0)$, the straight line segment due to a planar surface

$$z = Ax + By + C \tag{5}$$

is

$$z = Ax + By_0 + C = Ax + B_0, \tag{6}$$

In the next scan line $z(x, y_0 + 1)$, the line segment due to (5) is

$$z = Ax + B(y_0 + 1) + C = Ax + B_0 + B = Ax + B_1, \tag{7}$$

and in the scan line $z(x, y_0 + 2)$, the corresponding line segment is

$$z = Ax + B(y_0 + 2) + C = Ax + B_1 + B = Ax + B_2. \tag{8}$$

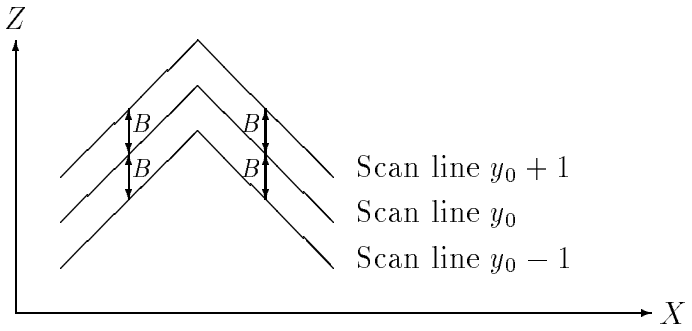

Figure 3: Graphical representation of the relationship of three line segments.

Thus, the three line segments have the same slope but different intercept. The difference in the intercept of two neighboring line segments is always B. These two constraints are graphically shown in Fig. 3.

For each seed region candidate formed by three line segments

$$s_i : \; z = a_i x + b_i, \quad i = 0, 1, 2 \tag{9}$$

if s_0, s_1 and s_2 are really due to the same planar surface (5), then

$$a_0 = a_1 = a_2 = A, \quad b_2 - b_1 = b_1 - b_0 = \frac{b_2 - b_0}{2} = B \tag{10}$$

holds in the ideal case. In reality, this will never be true. So, we need to test how close this relation is satisfied by each seed region candidate.

A simple test like

$$\sum_{i \neq j} (a_i - a_j)^2 + \sum_{i \neq j} (b_i^* - b_j^*)^2 \tag{11}$$

where $b_0^* = b_2 - b_1$, $b_1^* = b_1 - b_0$, and $b_2^* = \frac{b_2 - b_0}{2}$, is not feasible because the parameters a_i and b_i^* are not appropriate metrics for comparison purpose. The problem is that the absolute difference in space is a nonlinear function of the relative difference of such kind of parameters. For the 2-dimensional case, for instance, the angle θ between two lines of slopes m_1 and m_2 is a nonlinear function of their slopes:

$$\theta \;=\; tan^{-1}(m_1) - tan^{-1}(m_2). \tag{12}$$

In this work we use the following strategy for checking the similarity of line segments. The normals of two line segments s_i and s_j in the $x - z$ plane are $\mathbf{m}_i = (a_i, -1)$ and $\mathbf{m}_j = (a_j, -1)$, respectively. A good test for whether they have approximately the same slope is based on $\frac{\mathbf{m}_i \cdot \mathbf{m}_j}{|\mathbf{m}_i||\mathbf{m}_j|}$, being the cosine of the angle between $\mathbf{m}_i$ and $\mathbf{m}_j$. In the ideal case, $b_i^* = B$ holds. If we define a scan line as $z(x_0, y), x_0 \in I_N$ instead of $z(x, y_0), y_0 \in I_N$, then all the discussions above for the comparison of a_i are valid for b_i^*. So, if we define $\mathbf{n}_i = (b_i^*, -1)$, $\mathbf{n}_j = (b_j^*, -1)$, the

similarity between b_i^* and b_j^* can be measured by $\frac{\mathbf{n}_i \cdot \mathbf{n}_j}{|\mathbf{n}_i||\mathbf{n}_j|}$. From this discussion, we define the optimality measure of a seed region candidate as

$$\frac{1}{12}[\sum_{i \neq j} \frac{\mathbf{m}_i \cdot \mathbf{m}_j}{|\mathbf{m}_i||\mathbf{m}_j|} + \sum_{i \neq j} \frac{\mathbf{n}_i \cdot \mathbf{n}_j}{|\mathbf{n}_i||\mathbf{n}_j|}] + 0.5 \tag{13}$$

which falls into the interval $[0, 1]$.

In summary, we compute for each seed region candidate the optimality measure defined in (13), and select the candidate with the largest optimality value as the seed region for the growing process.

Here some comments on Pavlidis' algorithm [22] are adequate. In his algorithm, two neighboring line segments are merged if the absolute difference between their slopes is smaller than a threshold. As stated above, however, slope is not an appropriate metric for direct comparison. Thus, the simple similarity test of Pavlidis is not feasible for our application. Moreover, different line segments of a particular planar surface not only have the same slope, but also their intercepts are subject to the relation $\frac{b_i - b_j}{y_i - y_j} = B$. Thus, it is not sufficient to consider the slope alone. Rather both the slope and the intercept must be taken into account to guarantee that a set of line segments really belong to the same planar surface. Our optimality criterion (13) fulfills this requirement.

5.2 Iterative region growing

After a seed region R^0 is found, a region growing process takes place around R^0. We represent the region after the kth growing iteration by R^k, and use a simple-linked list L to record all line segments of R^k. A least-square plane fitting is performed to get the plane $P^k : \ z = A^k x + B^k y + C^k$ that best fits R^k. The task is to find A^k, B^k and C^k so that the approximation error

$$\varepsilon^2 \ = \ \sum_{(x,y) \in R^k} (z(x,y) - (A^k x + B^k y + C^k))^2 \tag{14}$$

is minimized. This is done by the least square technique.

The $(k+1)$th iteration proceeds in the following way. We go through the list L once, and for each line segment $s \in R^k$, we consider all neighboring line segments of s. For each neighboring line segment s' of s, if it is still unlabeled, we test whether s' is compatible with P^k. s' is compatible with P^k if, for all $(x, y) \in s'$,

$$|z(x,y) - (A^k x + B^k y + C^k)| \ \leq \ \theta \tag{15}$$

holds where θ is a threshold. In our experiments we set $\theta = 1.5 + 1.75\sigma_{img}$. We notice that, on the straight line segment s', the maximal approxation error will occur either at the leftmost pixel (x_l, y_0) or the rightmost pixel (x_r, y_0) of s'. Thus, the simple test

$$|z(x_l, y_0) - (A^k x_l + B^k y_0 + C^k)| \leq \theta \ \wedge \ |z(x_r, y_0) - (A^k x_r + B^k y_0 + C^k)| \leq \theta \tag{16}$$

is sufficient. This is a large saving as only two approximation error checks are needed instead of one check for each pixel of s', showing another advantage of the scan line grouping technique. If s' survives the test, it is added to the region and the list L. The growing process is iterated until, for some k, $R^k = R^{k+1}$.

The region growing above is quite simple, but it introduces some inefficiency because no deletion operation is performed on the list L. If all neighboring line segments of some $s \in L$ are already labelled, for instance, s needn't be considered in the subsequent iterations. We have tried two strategies to overcome this problem. First, we delete a line segment s from the list L if all neighboring line segments of s are already labelled. In the second strategy we go a step further. If, for each neighboring line segment s' of s, s' is either labeled or the approximation error is very large, say

$$|z(x_l, y_0) - (A^k x_l + B^k y_0 + C^k)| > 2\theta \ \vee \ |z(x_r, y_0) - (A^k x_r + B^k y_0 + C^k)| > 2\theta \quad (17)$$

such that the probability of s' being added to the region in a later iteration is almost zero, we delete s from L. These two strategies seem to be useful. Experimental results, however, show that no speedup could be achieved. This is because both strategies introduce extra housekeeping work. On the other hand, the additional cost resulting from keeping the unnecessary line segments in L is very small. So the saving from using deletion operations and the newly introduced housekeeping work are approximately the same. Based on this observation we use only the simple version without deletion.

6 Postprocessing

The algorithm above does not always yield clean edges between regions. There are two reasons for this behaviour. First, the simple splitting algorithm for line segment extraction finds sometimes nonoptimal separating points. This leads to some short line segments across two different planar surfaces. In fact, even more sophisticated approaches can not avoid this phenomenon. Another reason for noisy edges is that the region growing process *sequentially* extracts planar regions from range data. If a line segment s is added to the region R found so far, it is merely guaranteed that the approximation error of s is below a certain threshold. However, there might exist another region R' that gives a better approximation of s. This is a problem of growing order.

We solve the problem of noisy edges using an iterative refinement technique. For each line segment s, we check whether the leftmost (rightmost) pixel of s is better approximated by the left (right) neighbor of s. If this is the case, the pixel is deleted from s and added to the neighboring line segment. This simple heuristic is iteratively used until a stable solution is achieved. In our experiments we have observed that, in most cases, the refinement process converges after a few iterations.

In the region growing process we treat a line segment as a single entity, and only merging but no splitting of line segments is possible. The iterative refinement as

described above could be considered as some kind of splitting operation for improving nonoptimal segmentation results.

7 Experimental results

The segmentation algorithm has been implemented on a Sun Sparcstation 1 and tested on about 50 real range images acquired by two different range sensors. In this paper we give the results for five test scenes.

The first sensor is based on the projection of binary-coded patterns [27]. The 256×256 greylevel and range image of a scene bloc1 are shown in the top of Fig. 4. The result after line segment extraction is shown in the middle where the endpoints of the line segments are marked. The segmentation results before and after the postprocessing step are shown in the bottom where four grey levels are used to color the regions in such a way that no two neighboring regions get the same grey level. The results for two other range images acquired by the same sensor are shown in Fig. 5 and 6, respectively. Notice that the backgound plane in the scene bloc3 has been segmented into two planar regions. This is not an over-segmentation. Rather it is due to a column in which the range values have been erroneously measured by the sensor. Some thin regions of bloc3 (the black regions) have not been detected by our algorithm. This is because we require all the three line segments of a seed region have a minimum length (10 pixels in our experiments). In such a thin region, the line segments are so short that no seed region can be found.

The second set of test images was acquired by a Technical Arts scanner at Michigan State University. As an example, the range image of a scene curvblock-3 is shown in the top of Fig. 7. As no greylevel image of the scene is available, a 3-D plotting is given for better understanding of the scene. The extracted line segments are shown in the middle, and the segmentation results before and after postprocessing in the bottom. Another scene propane-5 is shown in Fig. 8. It contains two planar and one curved surface. Our algorithm has successfully detected the two planar regions and partitioned the curved (cylindrical) region into a few parallel surfaces which agree with our expectation.

Some statistics for the test scenes are recorded in Table 1. The first six rows show the computation time in seconds of the main steps of the algorithm. The total computation time is given in the seventh row. Some more statistics are recorded in rows eight to ten. We see that the final refinement process usually converges after a small number of iterations. The effect of edge cleaning can be easily observed in the final result images. We want to mention again that the same set of thresholds have been used for all images, i.e. for both types of range scanner.

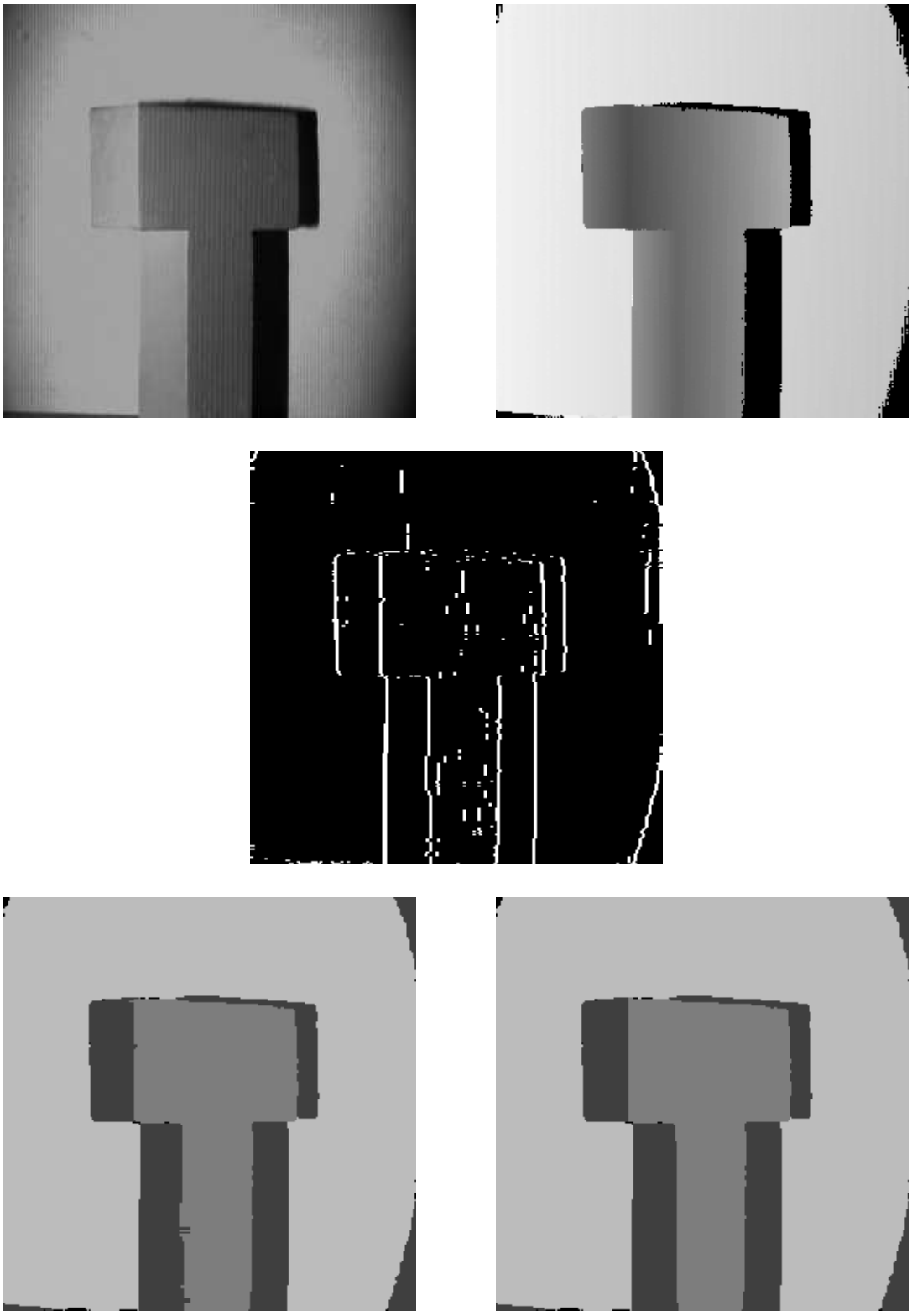

Figure 4: The test scene bloc1.

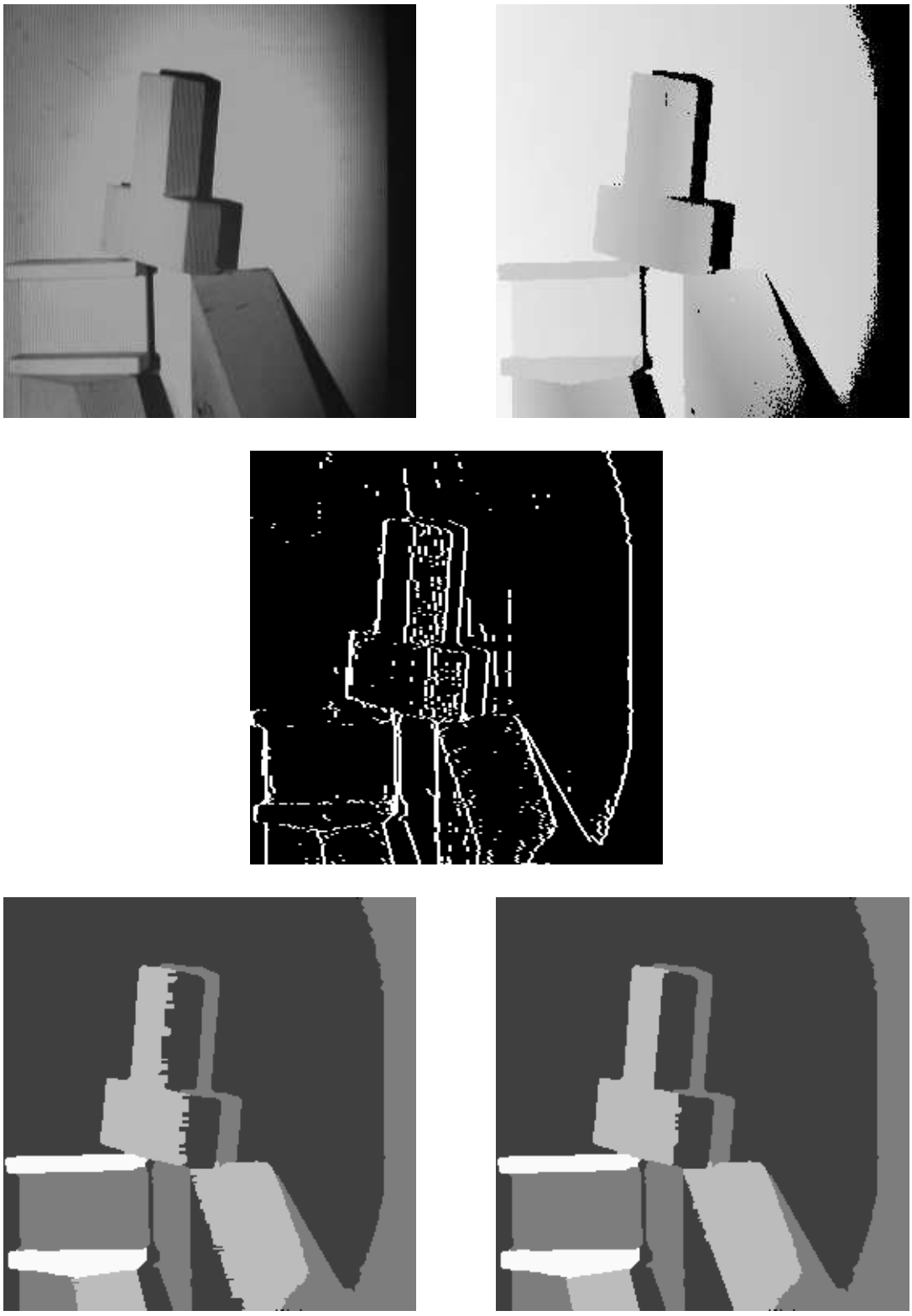

Figure 5: The test scene bloc2.

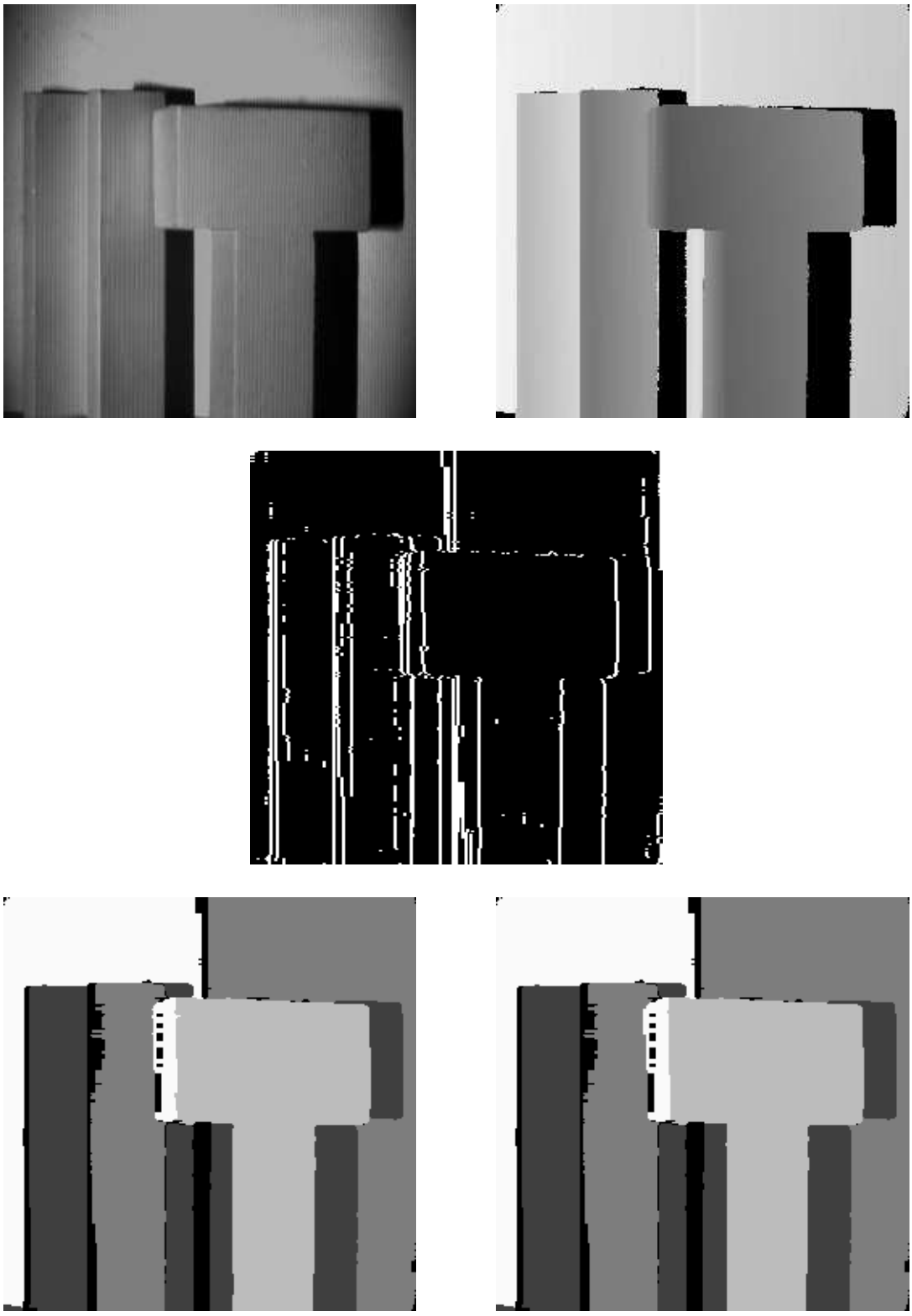

Figure 6: The test scene bloc3.

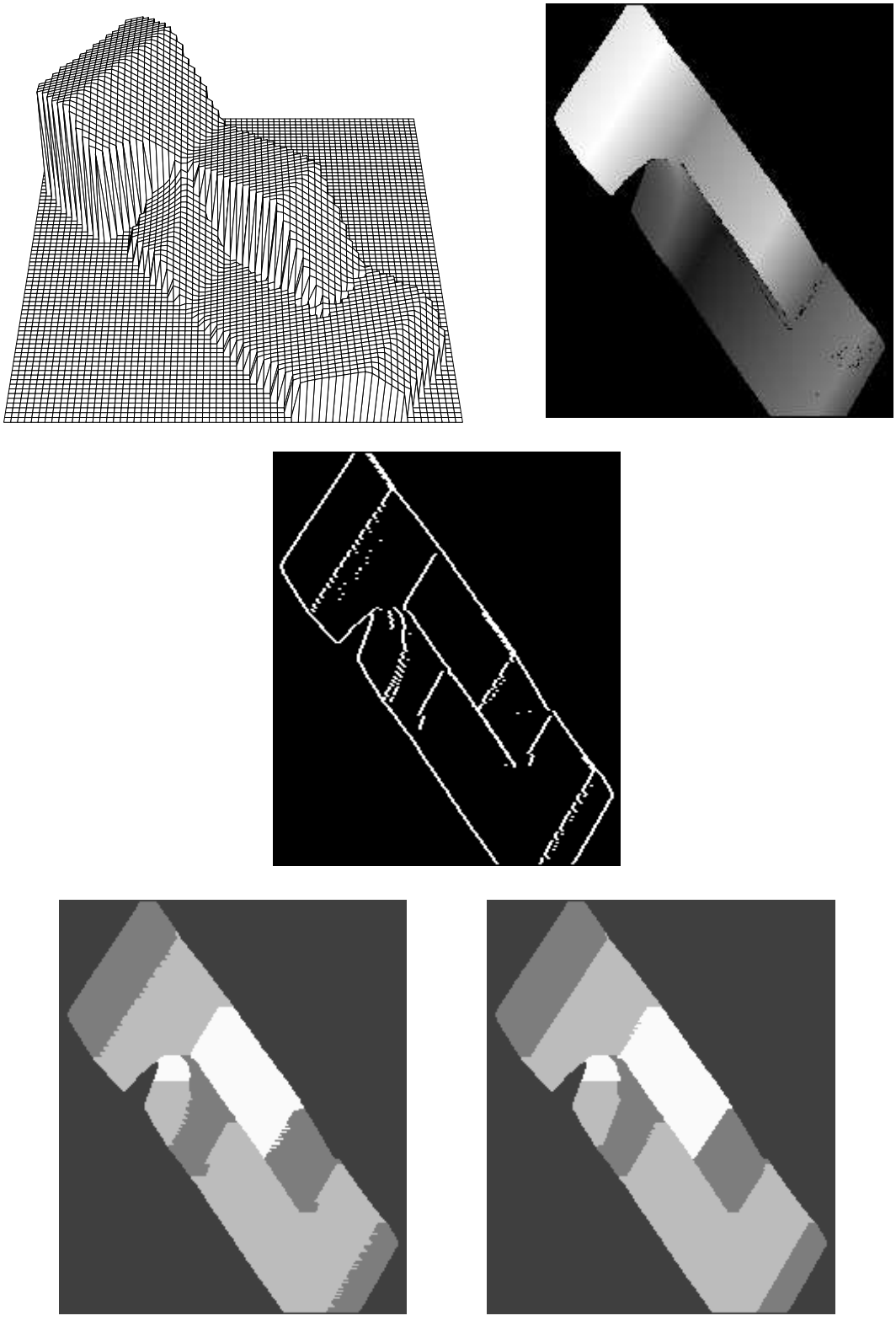

Figure 7: The test scene curvblock-3.

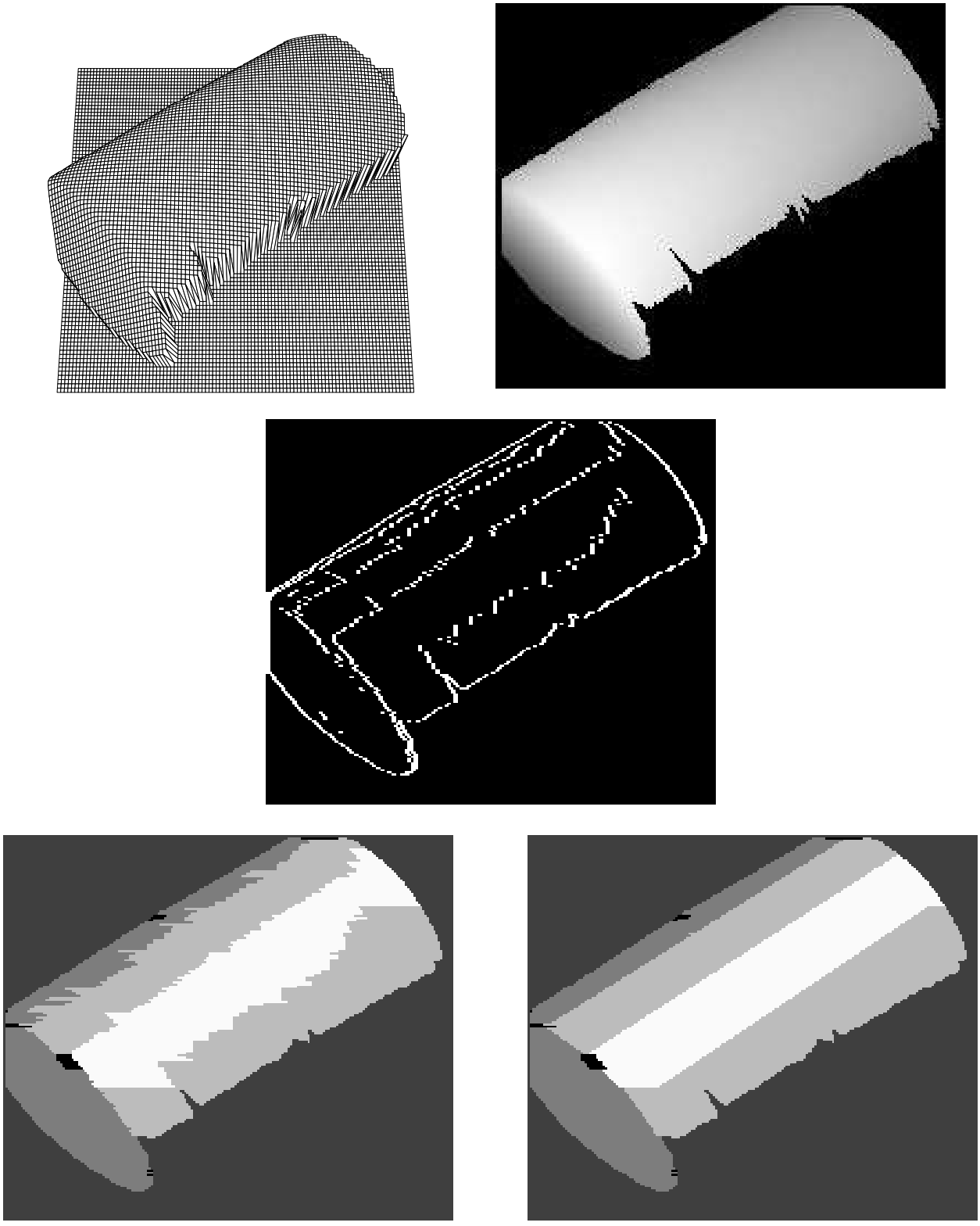

Figure 8: The test scene propane-5.

Table 1: Statistics for the test scenes. RMSE = Computation of RMSE. EOLS = Extraction of line segments. EOSR = Extraction of seed regions.

	bloc1 256×256	bloc2 256×256	bloc3 256×256	curvblock-3 240×203	propane-5 174×204
Median	3.50	3.50	3.50	2.62	1.88
RMSE	3.08	2.98	3.02	2.25	1.65
EOLS	0.80	1.07	1.20	0.65	0.43
EOSR	0.72	1.17	1.07	0.68	0.50
Growing	0.66	0.80	0.76	0.48	0.38
Postprocessing	0.05	0.15	0.07	0.05	0.10
Total (s)	8.81	9.67	9.62	6.73	4.94
σ_{img}	0.87	1.41	0.55	0.50	0.40
Line segments	1740	2882	3704	1333	1000
# Iterations	5	6	4	6	21

8 Discussions

From Table 1 we see that our algorithm is very fast. As a matter of fact, we have not paid special attention to efficiency in our current implementation. The Median filter and the splitting algorithm for line segment extraction, for instance, were directly implemented. The Median filtering could be done more efficiently [1, 19]. We could also use the faster pseudo-Median filter [24]. Moveover, there are more efficient ways for the iterative splitting [5, 9, 10]. Thus, our implementation could be further speeded up.

A closer look at Table 1 reveals that a large portion (65% or even more) of the run time is devoted to the Median filtering and the computation of RMSE. The actual segmentation takes merely a few seconds. This is a large speedup against almost all of the known algorithms for partitioning range images into planar regions.

The speed of our algorithm lies surely in the scan line grouping. The use of straight line segments as segmentation primitives greatly reduces the data dimension that must be handled in the region growing process. The test scene bloc2, for instance, has 65536 pixels, but only 2882 line segments. Hence, a reduction ratio of 22 has been achieved[1]. On the other hand, due to the use of the data structure described in section 4, the high-level primitive line segments do not result in more handling costs. We can access the neighborhood of a line segment almost as easily as in a regular grid structure. Moreover, the investigation of special properties of

[1] As a matter of fact, the x-coordinates and the data of the points at the end of the line segments provide the basis for a powerful method of image compression [23]. Our interest here, however, lies not in efficient storage and transmission of pictorial information but in an image representation which allows a fast segmentation.

line segments enables us to formulate the simple optimality measure in (13) for seed region extraction and the growing criterion in (16).

Our segmentation algorithm treats a line segment as a single entity and provides only very limited ability of line segment splitting in the postprocessing step. This means that an over-partition of scan lines is needed so that no powerful line segment splitting technique is required. The empirically determined threshold for the scan line splitting algorithm $1.0 + 0.5\sigma_{img}$ is low enough in order to always obtain an over-partition. This can be easily seen in the middle of Fig. 4. The experiments show that the region growing process has successfully grouped the over-partitioned line segments into global regions.

We have used the noise variance estimation σ_{img} to automatically set two thresholds. In this way the thresholds are directly tied to the geometrical and statistical properties of range data providing good performance for different images. We have tested the algorithm on real range images acquired by two different sensors. The use of a fixed set of thresholds for all the test images shows the robustness of the algorithm.

Our segmentation algorithm is a dedicated approach. It searchs for planar surfaces in range data. It is interesting to ask whether the idea of scan line grouping could be extended to other types of geometrical entities, like spheres or cylinders. This topic is currently being evaluated.

Acknowledgements

The first author was supported by the Swiss National Science Foundation under the NFP-23 program, Grant No. 4023-027026. We thank F. M. Wahl and T. G. Stahs of the Technical University of Braunschweig, F. R. Germany, for providing some of the test range images, and the Pattern Recognition and Image Processing Lab of Michigan State University for making a database of range images for public usage. Further thanks are due to U. Meier who prepared the 3-D plottings and to R. Robmann for valuable discussions.

References

[1] J. T. Astola, T. G. Campbell, On computation of the running median, IEEE Trans. on ASSP, Vol. 37, No. 4, 572–574, 1989.

[2] P. J. Besl, Active, optical range imaging sensors, Machine Vision and Applications, Vol. 1, 127–152, 1988.

[3] P. J. Besl, Surfaces in range image understanding, Springer-Verlag, 1988.

[4] R. C. Bolles, M. A. Fischler, A RANSAC-based approach to model fitting and its application to finding cylinders in range data, Proc. of 7th Int. Conf. on Artificial Intelligence, Vancouver, 637–643, 1981.

[5] L.-M. Chien, et al., Fast corner detection using linear search, Proc. of 7th Scand. Conf. on Image Analysis, Aalborg, Denmark, 354–361, 1991.

[6] E. Diday, J. C. Simon, Clustering analysis, in Digital pattern recognition, K. S. Fu (Ed.), Springer, New York, 47–94, 1976.

[7] J. G. Dunham, Optimum uniform piecewise linear approximation of planar curves, IEEE Trans. on PAMI, Vol. 8, No. 1, 67–75, 1986.

[8] R. O. Duda, P. E. Hart, Pattern classification and scene analysis, Wiley, New York, 1972.

[9] R. Epselid, I. Jonassen, A comparison of splitting methods for the identification of corner-points, Pattern Recognition Letters, Vol. 12, 79–83, 1991.

[10] M.-H. Han, D. Jang, J. Foster, Identification of cornerpoints of two-dimensional images using linear search, Pattern Recognition, Vol. 22, No. 1, 13–20, 1989.

[11] A. Härkönen, H. Ailisto, I. Moring, Noise analysis and filtering of range images produced by a scanning laser range finder, Proc. of 6th Scand. Conf. on Image Analysis, Oulu, Finland, 481–491, 1989.

[12] S. L. Horowitz, T. Pavlidis, Picture segmentation by a direct split and merge procedure, Proc. of 2nd Int. Conf. on Pattern Recognition, 424–433, 1974.

[13] R. A. Jarvis, A perspective on range finding techniques for computer vision, IEEE Trans. on PAMI, Vol. 5, No. 2, 122–139, 1983.

[14] J.-M. Jolion, P. Meer, S. Bataouche, Robust clustering with applications in computer vision, IEEE Trans. on PAMI, Vol. 13, No. 8, 791–802, 1991.

[15] R. Krishnapuram, C.-P. Freg, Fuzzy algorithms to find linear and planar clusters and their applications, Proc. of CVPR'91, 426–431, 1991.

[16] T. Lozano-Pérez, W. E. L. Grimson, S. J. White, Finding cylinders in range data, Proc. of IEEE Conf. on Robotics and Automation, 202–207, 1987.

[17] G. Maître, H. Hügli, F. Tièche, J. P. Amann, Range image segmantion based on function approximation, Proc. of ISPRS-Conference, SPIE Vol. 1395, Zurich, 275–282, 1990.

[18] D. Nitzan, Three-dimensional vision structure for robot applications, IEEE Trans. on PAMI, Vol. 10, No. 3, 291–309, 1988.

[19] A. W. Paeth, Median finding on a 3×3 grid, in Graphics Gems, A. S. Glassner (Ed.), Academic Press Inc., 171–175, 1990.

[20] B. Parvin, G. Medioni, Segmentation of range images into planar surfaces by split and merge, Proc. of Computer Society Conf. on Computer Vision and Pattern Recognition, 415–417, 1986.

[21] T. Pavlidis, S. L. Horowitz, Segmentation of plane curves, IEEE Trans. on Comput. Vol. C-23, 860–870, 1974.

[22] T. Pavlidis, Segmentation of pictures and maps through functional approximation, Computer Graphics and Image Processing, Vol. 1, 360–372, 1976.

[23] D. T. Pham, M. Abdollahi, Image compression using polylines, Pattern Recognition, Vol. 21, No. 6, 631–637, 1988.

[24] W. K. Pratt, Digital image processing, Second edition, John Wiley & Sons, Inc., 1991.

[25] F. Schmitt, X. Chen, Fast segmentation of range images into planar regions, Proc. of CVPR'91, 710–711, 1991.

[26] L. Shao, R. Volz, Finding cones from multi-scan range images, SPIE Vol. 1608, Intelligent Robots and Computer Vision X: Neural, Biological and 3-D Methods, 378–384, 1991.

[27] T. G. Stahs, F. M. Wahl, Fast and robust range data acquisition in a low-cost environment, Proc. of ISPRS-Conference, SPIE Vol. 1395, Zurich, 496–503, 1990.

[28] R. W. Taylor, M. Savini, A. P. Reeves, Fast segmentation of range imagery into planar regions, Computer Vision, Graphics, and Image Processing, Vol. 45, 42–60, 1989.

[29] H. S. Yang, A. C. Kak, Determination of the identity, position and orientation of the topmost object in a pile, Computer Vision, Graphics, and Image Processing, Vol. 36, 229–255, 1986.

[30] N. Yokoya, M. D. Levine, Volumetric description of solids of revolution in a range image, Proc. of 10th Int. Conf. on Pattern Recognition, 303–307, 1990.