IMPROVING A GENETIC ALGORITHM SEGMENTATION BY MEANS OF A FAST EDGE DETECTION TECHNIQUE

Angel D. Sappa [†] *Vitoantonio Bevilacqua* [†] *Michel Devy* [‡]

RTS Advanced Robotics Ltd. [†]
Derwent House, Clarence Avenue
Trafford Park, Manchester, M17 1QS, UK
sappa@ieee.org, vito.bevilacqua@robotics.co.uk

LAAS-CNRS [‡]
7, Avenue du Colonel Roche
31077 Toulouse, Cedex 4, France
michel@laas.fr

ABSTRACT

This paper presents a new hybrid range image segmentation approach. Two separate techniques are applied consecutively. First, an edge based segmentation technique extracts the edge points—creases and jumps—contained in the given range image. Then, by using only the edge point position information, the boundaries are computed. Secondly, the points clustered into each region are approximated by single surfaces through a Genetic Algorithm (GA). The GA takes advantage of previous edge representation finding the surface parameters that best fit each region. It works in a local way, according to the boundary information, reducing considerably the required CPU time. Experimental results with different range images are presented; moreover a comparison using either the edge detection stage or not is given.

1. INTRODUCTION

Range image interpretation is usually based in high level representation. Therefore, with this objective in mind, several works have been proposed within the range image segmentation field during the last years. They can be roughly classified into three categories: *edge-based* [1][2], *region-based* [3][4] and *hybrid* techniques [5][6][7]. Although each one of them has its own attractive characteristics, hybrid techniques seem to be the most appropriate option when a general solution is sought. That is because hybrid techniques take advantage of the rich information given by the edge detection algorithms by integrating with the efficiency of the surface based techniques.

Generally, hybrid techniques use the information provided by edge detection techniques to estimate the number of classes or to select optimal region seeds—to initialize clustering or region growing algorithms. In these cases, the edge detection technique is carried out to guide and improve the region based segmentation methods.

In this sense, this paper presents a combined use of two techniques. First, edge points are detected by means of a scan line approach. Then, the edges defined by those points are extracted through a graph representation strategy. These sets of edges are the boundaries of the regions contained in the given range image. Next, the second technique is carried out over each set of enclosed points computing the approximating functions. The surface parameters are found by using a genetic algorithm approach.

In [8] has been shown that optimal surface fitting under geometric constraints is feasible with an evolutionary algorithm. This problem is generally formulated as a non linear programming problem, which tries to optimally fit the data to candidate shape descriptions. But, with non linear constraints it is very difficult to optimize the functions and there is not a known method to guarantee a satisfactory solution. GA allows an efficient exploration of the search space and an exhaustive exploitation of the best solutions as well. However, due to: *1)* the multi-modal nature of the fitting functions, *2)* the presence of noise, and *3)* the very large number of points to be processed at each iteration of the algorithm, it is difficult to use a GA technique to fit the data points contained into a given range image. For the previous reasons, this work proposes first the segmentation of the given range image into individual surfaces—by means of an edge based segmentation technique—and then the approximation of each region by using a GA approach. Additionally, the first stage is responsible to detect the shadow regions which are not considered by the second stage. In this way, a good performance in terms of CPU time is achieved and a satisfactory rate of convergence can be guaranteed.

The paper is organized as follow. Section 2 summarizes the edge extraction technique. Section 3 presents the surface fitting technique. Section 4 shows experimental results and comparisons. Finally, conclusions are given in section 5.

This work has been carried out as part of the *CAMERA* project (*CA*d *M*odelling of *B*uilt *E*nvironments from *R*ange *A*nalysis). *CAMERA* is an EC funded TMR network (ERB FMRX-CT97-0127)

0-7803-6725-1/01/$10.00 ©2001 IEEE

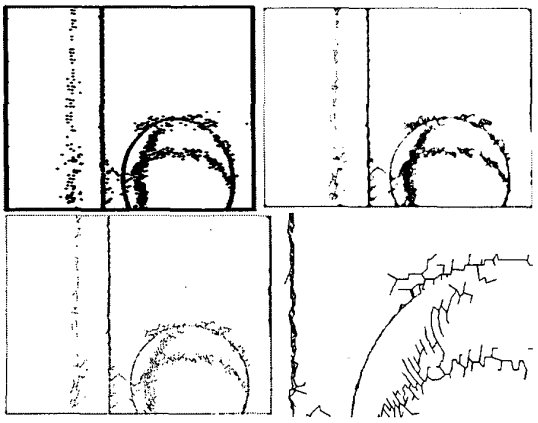

Figure 1. (*top-left*) Binary edge map. (*top-right*) Triangular mesh from the 2D Delaunay triangulation (triangles with long edges have been removed, only triangles at edge or noisy regions are preserved). (*bottom-left*) MST resulting from the graph representation. (*bottom-right*) Enlargement of the MST.

2. EDGE EXTRACTION

This section describes the technique used for extracting the edges contained in the given range image. It consists of four stages. First, a binary edge map is generated by extracting crease and jump edges. Then those selected points are triangulated over their 2D space by means of a Delaunay triangulation algorithm. The resulting triangular mesh is considered as a weighted graph, next the *Minimum Spanning Tree*—MST—is computed. Finally, after filtering that MST, the edges defining each region are obtained and the enclosed points are labelled. Each stage is summarized below; a complete description of each one of them can be found in [9].

2.1. Binary Edge Map Generation

The aim of this first stage is to generate a two dimensional array R, where each element is a binary value indicating whether that point is an edge point or not. This stage has been implemented by using a scan line technique similar to the one presented in [1][2]. A scan line can be understood as a profile digitalization where its 2D projection can be represented by a planar curve. In [1] and [2], rows, columns, and diagonals are considered as scan lines. On the contrary, in the current work, only rows and columns are considered.

At this stage, every row and column (hereinafter called scan lines) is approximated by a set of quadratic functions. These functions are obtained by a recursive splitting algorithm. It computes the parameters of the approximating function by using three points—the first, middle and last point of the considered scan line. Then, the approximation error between the obtained quadratic function and every point of the scan line is computed. If this error is above a given threshold, the considered scan line is split into two curves at the point where the biggest error appears. Next the parameters of the quadratic curve are computed again, by using the new set of points. This splitting algorithm is applied recursively while the approximation error is above the given threshold.

The result of this recursive algorithm is a set of quadratic curves approximating the given scan line. From each quadratic curve, the first and last points are selected and their positions in a binary map are labelled. The 3D coordinates associated with each selected point are kept and will be used during the next stage. Fig. 1(*top-left*) shows an example of a binary map from the proposed scan line approximation algorithm.

2.2. 2D Triangulation and Graph Generation

At this stage, the points of the binary map are triangulated through a 2D Delaunay algorithm. Then, the obtained planar triangular mesh is considered as a weighted graph G. The vertices of the mesh are the nodes of that graph and the edges defining the triangles give the edges of that graph. Each edge of the graph has associated a cost which is defined by the 3D distance between the points joined by this edge. In order to speed up further processing, edges with a cost higher than some given threshold are removed (Fig. 1(*top-right*) shows an example of a planar triangular mesh, edges with a high cost have been removed).

2.3. Minimum Spanning Tree Generation

Given a weighted graph G, the MST of G is the acyclic subgraph of G that contains all the nodes and such that the sum of the costs associated with its edges is minimum. It can be computed by applying *Kruskal's algorithm* [10].

Fig. 1(*bottom-left*) shows the MST obtained from the triangular mesh showed in Fig. 1(*top-right*), an enlargement of that MST is presented in Fig. 1(*bottom-right*). As expected, the generated trees go along the edge points defining the boundaries by linking them. However, the algorithm also generates several short branches which have to be removed during the next step.

2.4. MST Filtering and Region Labelling

The resulting MST can be understood as a set of independent trees. Each of these trees is a polyline where its segments are the graph's edges. As shown in Fig. 1(*bottom-left*), several short branches—isolated or linked with the boundaries—were generated from the MST. Then, in order to remove those short branches, a kind of *opening algorithm* is performed. It consists in applying an iterative *erosion* process followed by a *dilation* process. The latter is applied as many times as those of the erosion. The opening algorithm (erosion and dilation process) considers the segments of the polyline as basic processing elements (like pixels for the classical opening operator of intensity image processing). Those segments linked from only one of their ending points, *end segments*, are removed during the erosion stage. This stage is

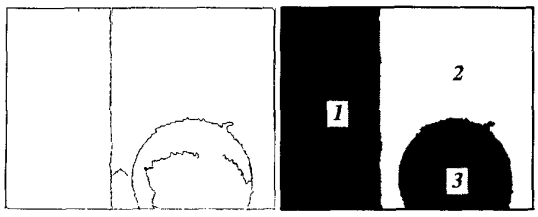

Figure 2. (*left*) Boundaries obtained from the opening stage. (*right*) The three labelled regions.

applied t times and at each iteration, all the end segments of that configuration are removed. Once the erosion process is completed, a dilation process is performed. This algorithm is carried out over the end segments left by the erosion process. It consists in putting back the segments connected with each one of the end segments present at each iteration. The number of dilations is the same as the number of erosions.

Next, the different boundaries are extracted and labelled. Finally, the enclosed points are also labelled as items of that region. Fig. 2(*left*) shows the boundaries obtained from the opening stage. Notice that some branches, resulting from linking noisy data, still remain. Due to they do not define any closed boundary are labelled as items of their surrounding region. The obtained boundaries are used to extract the three regions showed in Fig. 2(*right*).

3. SURFACE FITTING BY MEANS OF A GENETIC ALGORITHM

The outcome of the previous stage is a set of regions covering all the surface of the given range image. In the example used so far these regions are: a cylinder *(1)*, a plane *(2)* and a sphere *(3)* (see Fig. 2(*right*)). At this stage, the objective is to fit each one of these sets of points by means of a single surface by using a GA. This stage is applied independently over each region. In order to speed up the convergence of the GA, only one out of twenty five points defining each region have been considered. It is owing to the knowledge that each region is defined by a single surface, then it is not necessary to take into account all the contained points.

In the current implementation a *real-valued chromosome* representation has been used. Each gene corresponds to a coefficient in a shape equation and it is defined by a floating point number. It has been considered three surface primitives: planes, spheres and cylinders. In the case of planes, a four genes parametric representation: $\{n_1, n_2, n_3, d\}$ has been used; n_1, n_2, n_3 are the components of the unit normal vector associated with the plane and d is the constant defining its minimum distance from the origin of the reference frame. In the case of the spheres, again a four genes parametric representation $\{p_1, p_2, p_3, r\}$ has been used; p_1, p_2, p_3 are the coordinates of the centre point and r is the radius of the sphere. In the case of cylinders, a seven genes representation $\{c_1, c_2, c_3, N_1, N_2, N_3, r\}$ has been used; c_1, c_2, c_3 are the coordinates of the starting point of the cylinder, N_1, N_2, N_3 are the components of the unit vector defining the axis direction and r is the radius.

The used GA is *Genocop III* [11] which is specialized to:

a) handle constrained function optimizations with non linear constraints in order to ensure that solutions do not fall outside the range;

b) avoid trivial null solutions;

c) include information about the mutual positions of the different shapes in the images;

The main characteristic of this algorithm is the presence of two populations: a *reference set* and a *search set*. The reference population is a set of fully feasible individuals which satisfy all constraints whereas the search population may not. At each iteration, the search population is allowed to move around the solution space; an individual S of the search set is combined randomly with an individual R of the reference set to generate a new one that will replace R if it is better than it. Moreover *Genocop III* uses operators to ensure that any mutation and cross-over operation always produce a child from the constrained solution space.

The fitting function to be minimized is the sum, for all the points, of the true geometric distances to each theoretical surface, being known a prior the classification of the surfaces in the data. For example in the case of the sphere the function to be minimized is the following:

$$\sum_{i=0}^{n} \left(\sqrt{(x_i - p_1)^2 + (y_i - p_2)^2 + (z_i - p_3)^2} - r\right)^2$$

where x_i, y_i, z_i are the coordinates of the point i, and n is the number of points belonging to the sphere.

4. EXPERIMENTAL RESULTS

The proposed technique has been tested with different noisy range images, defined by *480x640* (rows x columns) points from the K2T data base. Moreover, a comparisons using either the edge extraction stage or not is presented. CPU times have been measured on a Sun Ultra 5. Fig. 1(*top-left*) shows the binary map obtained by using the scan line processing approach; it contains 4,549 points. This binary map was generated in 7.56 sec. The points of the binary map were used by the 2D Delaunay algorithm to generate a triangular mesh with 7,940 triangles. The 2D Delaunay triangulation was computed in 0.53 sec. Fig. 1(*top-right*) shows the triangular mesh (graph) obtained after removing long edges. This graph contains 8,558 edges. From that graph, the MST was generated in 3.22 sec. It is shown in Fig. 1(*bottom-left*) (an enlargement is shown in Fig. 1(*bottom-right*)). Finally, Fig. 2(*left*) shows the boundaries obtained from the opening process. There, some open boundaries still remain; they are removed by the final labelling stage. The opening process took 0.3 sec. generating a representation with 3,204 segments. Fig. 2(*right*) shows the three regions labelled and extracted from the previous boundary representation. The labelling process took 1.8 sec. The surface parameters fitting

Cylinder *(1)*	Plane *(2)*	Sphere *(3)*
c_1 = -5.61	n_1 = -0.282	p_1 = -8.15
c_2 = -3.25	n_2 = -0.957	p_2 = -9.13
c_3 = 5.75	n_3 = 0.062	p_3 = 8.86
r = 4.32	d = 6.75	r = 4.05
N_1 = -0.92		
N_2 = 0.129		
N_3 = -0.368		

Table 1: Parameters of the surfaces showed in Fig. 2 (*right*) (these parameters are close to the real parameters of the original surfaces).

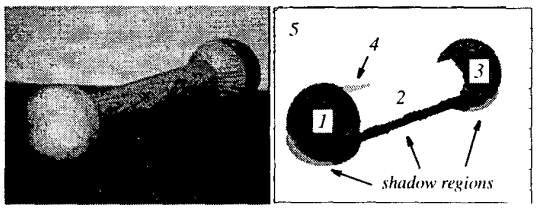

Figure 3. (*left*) Intensity image. (*right*) Regions obtained by the edge extraction stage (Section 2).

each of these regions were obtained by the GA approach presented in Section 3. They are shown in Table 1. These parameters were obtained in 251 sec. by the GA.

Fig 3 shows another result obtained from a range image of the same set. The CPU time to obtain the regions showed in Fig. 3(*right*) was 84.65 sec. From this time: 14.92 sec. were used by the binary map generation; then, 1.43 sec. by the 2D Delaunay triangulation; 56.34 sec. were used by the MST generation; 4.79 sec. by the opening stage; and finally 7.17 sec. were required to extract and label the regions. The edge extraction stage succeeds in obtaining the regions and boundaries that define the object contained in the scene, but on the contrary, some crease edges defined by the intersection of the floor with the wall are missed. Then, the opening algorithm removes that unconnected boundary linking the floor with the wall. At the second stage, the GA is the responsible to obtain the surface parameters fitting the different regions (five regions in the current example, the shadow regions are not considered). The GA finds the parameters of each surface in 3,231.1 sec.

Finally, the range image corresponding to the example of Fig. 1 has been used to compare the CPU times of the proposed technique—edge based segmentation plus GA—with a segmentation without using the edge extraction stage—only using the GA stage. The segmentation results (surface parameters) are comparable but the segmentation which does not involve edge extraction took almost eight times the time spent by the proposed technique (Edge Detection plus GA: 264.41 sec.; Genetic Algorithm 2,000 sec.). In the second case, the adopted fitting function maximizes the number of points instead of minimize least square error.

5. CONCLUSIONS

This paper proposes an hybrid technique which merge the speed of an edge based segmentation technique with the flexibility of surface fitting by using a GA approach.

The use of GA for segmenting range images is an appealing option. However, when large range images are considered, the required CPU time makes it difficult to use. For that reason this paper propose the use of a fast edge based segmentation stage at the first stage. Then, the GA is used only in a second stage over each detected region. It is carried out in a local way reducing considerably the required CPU time. Moreover, another advantage of the proposed hybrid technique is that shadow regions are detected at the first stage and left out, avoiding spend time trying to fit them.

Further work will consist of the automatic determination of the kind of surface primitives to be used for fitting each region.

6. REFERENCES

[1] X. Jiang and H. Bunke, Edge Detection in Range Images Based on Scan Line Approximation, Computer Vision and Image Understanding, Vol. 73, No. 2, pp. 183-199, February 1999.

[2] X. Jiang and H. Bunke, Range Image Segmentation: Adaptive Grouping of Edges into Regions, Computer Vision-ACCV'98 (R. Chin and T. Pong, Eds.), pp. 299-306, Springer-Verlag, Berlin/New York, 1998.

[3] M. A. García and L. Basañez, Fast Extraction of Surface Primitives from Range Images, 13th IAPR Int. Conf. on Pattern Recognition, vol. III, pp. 568-572, Vienna, Austria, August 1996.

[4] P. Besl and R. Jain, Segmentation Through Variable-Order Surface Fitting, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 10, No. 2, pp. 167-192, 1988.

[5] O. Pereira Bellon, A. Direne and L. Silva, Edge Detection to Guide Range Image Segmentation by Clustering Techniques, IEEE Int. Conf. on Image Processing, Kobe, Japan, October 1999.

[6] E. Al-Hujazi and A. Sood, Range Image Segmentation with Applications to Robot Bin-Picking Using Vacuum Gripper, IEEE Trans. on SMC, Vol. 20, No. 6, pp. 1313-1325, December 1990.

[7] K. Koster and M. Spann, MIR: An Approach to Robust Clustering-Application to Range Image Segmentation, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 22, No. 5, pp. 430-444, May 2000.

[8] C. Robertson, D. Corne, R. Fisher, N. Werghi, A. Ashbrook, *Investigating Evolutionary Optimization of Constrained Functions to Capture Shape Descriptions from Range Data*, Advances in Soft Computing Engineering Design and Manufacturing, eds. Roy, Furuhashi and Chawdhry, Springer-Verlag, 1998.

[9] A. Sappa and M. Devy, Fast Range Image Segmentation by and Edge Detection Strategy, Int. Conf. on 3D Digital Imaging and Modeling, Quebec, Canada, May-June 2001.

[10] K. Rosen, *Discrete Mathematics and its Applications*. McGraw-Hill, Inc., New York, second edition, 1990.

[11] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 3rd edition.