Proceedings of the 2005 IEEE
International Conference on Robotics and Automation
Barcelona, Spain, April 2005

# Efficient Closed Contour Extraction from Range Image's Edge Points

Angel D. Sappa
*Computer Vision Center*
*Edifici O Campus UAB*
*08193 Bellaterra, Barcelona, Spain*
angel.sappa@cvc.uab.es

***Abstract* - This paper presents an improvement over a previous contour closure algorithm. Assuming that edge points are given as input, the proposed approach consists of two steps. Similarly than the previous approach, the minimum spanning tree of a partially connected graph is initially computed. Then, a morphological filter removes noisy links and finally open contours are closed by minimizing a linking cost function. Advantages of the proposed technique lie in the lack of user defined thresholds and non-dependency of edge point density. Experimental results with synthetic and real range images are presented showing encouraging results with uniform and non-uniform edge points' distribution.**

***Index Terms* - *Contour Closure. Range Image Segmentation.***

## I. Introduction

Human being can easily extract objects' contours after watching their defining set of points. Unfortunately, this simple and almost trivial action for the human being is a quite difficult task to be automatically performed. A lot of work has been carried out in the computer vision community, some of them using the psychology as an inspiration source. Human visual system can detect many patterns of image elements; the ability to extract significant image relations without any knowledge of the image content and group them to obtain meaningful higher-level structure is usually referred as perceptual grouping. Research in perceptual grouping was started in 1920's by Gestalt psychologists. The hierarchical grouping principles, proposed by Gestalt psychologists, embodied such concepts as grouping by proximity, similarity, continuation, closure, and symmetry [1].

Several techniques have been developed in the range image literature to compute closed contours. Classically, they were inspired from the 2D image processing field; hence, some of the proposed 3D contour closure approaches have been based on the use of morphological operators (e.g., [2], [3], [4]). Other approaches try to link edge points according to local measures of continuity and smoothness, with no a priori information about the object shape. These techniques include several well-know algorithms from different fields (deformable models, constrained clustering and data ordering), see [5] for further details. Differently than these approaches, in [6] a graph-based technique has been proposed. Although interesting experimental results were presented, the major problems of that technique were user defined thresholds and constrains on the density of edge points—they should be uniformly sampled. In this new version, both drawbacks have been finally solved giving rise to a fully unsupervised technique.

This work was supported in part by the Government of Spain under the CICYT project TRA2004-06702/AUT; the author was supported by *The Ramón y Cajal Program*.

Graph theory has long been used in the 2D image segmentation problem (e.g., [5], [7], [8], [9], [10], [11]). Our proposed approach differs from them not only due to the fact that it is focused on range image processing but also in the following aspects. *Firstly*, some of those techniques were meant for partitioning a gray-level image into connected homogeneous regions—region-based approaches ([7], [8], [9])—; for example, [7] introduces a 2D image segmentation algorithm using minimum spanning trees. By minimizing the sum of gray levels variations a minimum spanning tree is partitioned into subtrees, representing different homogeneous regions. Similarly, [8] proposes a graph-partitioning with non-parametric clustering approach for 2D image segmentation. *Secondly*, those approaches, proposed to extract closed contours, were intended to compute the external object boundary ([5], [10]), but not closed boundaries of the object's regions. [5] presents a clustering algorithm based on the minimization of a cost function that depends on several well-known techniques: snakes, Kohonen maps, elastic nets, and hard and fuzzy c-means. Differently than the previous proposal, [10] presents an object contour closure technique by finding the eigenvector with the largest positive real eigenvalue of a transition matrix for a Markov process where edges from the image serve as states. That work incorporates the Gestalt principles of proximity and good continuity. Additionally this approach is able to handle scenes containing several objects. Finally, the use of graph-based representations has been discussed on [11]. The authors introduce a full graph-based active vision system able to solve such tasks as: image segmentation, image perceptual grouping and object recognition (face and 3D object recognition). The system is used to drive an autonomous mobile robot. Segmentation problem has been solved by using a graph partition greedy algorithm with superlinear time complexity.

Up to our knowledge, [6] was the first approach to tackle range image contour extraction problem by means of a graph-based strategy. In the current work improvements over that first work are introduced looking for an unsupervised contour closure technique. The proposed algorithm is described in Section 2. Section 3 presents experimental results by using synthetic and real range images. Conclusions are finally given in Section 4.

## II. Contour Closure Algorithm

Closed contour extraction is a common problem within the edge-based segmentation approaches. Although, several techniques have been proposed, they are focussed on solutions for their specific problems. For example, [12] presents an edge linking algorithm to close a one pixel gap in any one of the four directions.

Some other works tackle the contour extraction problem by analyzing the enclosed surfaces [4], [13]; therefore, contour and region are simultaneously extracted. [2] brought forward an adaptive approach that extracts closed contour by applying a process of hypotheses generation and verification. This algorithm is based on the consideration that any contour gap can be closed by dilating the input edge map. Thus, a single dilation operation followed by region verification is applied until all regions are labeled. The problem is that, as the dilation is performed in all directions, thin regions are liable to disappear, due to the fusion of the contours enclosing them. An extension of the previous approach is presented in [3]. There, the geometry of contours is taken into account in order to apply the dilation—the dilation process is restricted to one direction. Advantages and disadvantages of the aforementioned techniques are presented in [6].

The proposed technique is carried out with no a priori information about the object's surface shape. It takes the contour closure problem as a graph partitioning one and present improvements over the first version [6]. Assuming that a range image, together with its corresponding edge point information are given as inputs, the proposed technique consists of two stages. Firstly, the *minimum spanning tree* (MST) of a partially connected graph is computed. Finally, a post-processing technique, based on a filtering technique plus a cost function minimization, generates a single path describing the object's contours. These stages are described below.

### A. Graph Generation and MST

Let $\mathbf{R}(r, c)$ be a range image with $R$ rows and $C$ columns, where each array element $(r, c)$ ($r \in [0, R)$ and $c \in [0, C)$) is a scalar that represents a surface point of coordinates $(x, y, z)$, referred to a local coordinate system associated with the range sensor. Additionally, the corresponding edge points' information is given as a binary array $\mathbf{B}(r, c)$—edge points are labeled with a 1, while non-edge points with a 0. These points are supposed to be previously computed by some edge-based range image segmentation algorithm (see illustration in Fig. 1(*right*)).

The aim at this stage is to find the best connectivity between edge points. A simple and easy approach could be to start with a fully connected graph; however, in order to speed up further processing a partially connected graph is chosen. Having in mind that the partially connected graph should connects edges linking every couple of nearest neighbors, a 2D Delaunay triangulation algorithm has been finally adopted.

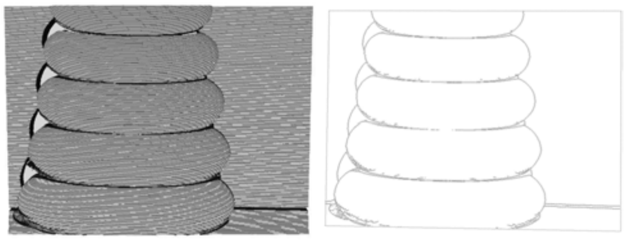


Fig. 1. (*left*) Input range image (*640x480*). (*right*) Input binary array of edge points corresponding to the given range image (*11,075* points).

Let $P = \{B_i = (r_i, c_i) \mid i = 1, ..., n\}$ be a set of edge points in the binary array $\mathbf{B}(r, c)$, its 2D triangular mesh is a piecewise linear partition consisting of triangles connected along their edges. Formally, a 2D triangular mesh $M$ is a set $\{P, E\}$, where $P = \{B_1, \ldots, B_n\}$, $B_i \in \mathbf{R}^2$, is a set of vertex and $E$ is a description of the mesh topology, $E = \{(B_i, B_j) | i, j = 1, \ldots, n; i \neq j\}$. The triangular mesh $M$ is a Delaunay triangulation of $P$ if and only if the circumcircle of any triangle of $M$ does not contain a point of $P$ in its interior [14]. In other words, the Delaunay triangulation of the input edge points $P$ will connect every point with its nearest, as we were looking for.

In order to obtain the shortest—cheapest—path linking all the edge points, that triangular mesh is now considered as a partially connected weighted planar graph $G = \{P, E_w\}$; every edge is associated with a cost value computed as the 3D distance between the range image points linked by that edge:

$$E_w = \{(B_i, B_j, w_{i,j}) | (w_{i,j} = dist(R_{(r_i, c_i)}, R_{(r_j, c_j)})) \wedge (i, j = 1, \ldots, n, i \neq j)\}$$

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. The MST of a graph $G$, defined by $m$ edges and $n$ vertices can be efficiently computed in O($m$log$n$) by applying Kruskal's algorithm [15]. In the current implementation, due to the fact that $G$ is a 2D Delaunay triangulation of the $n$ input edge points, but not a fully connected graph, the cost can be bounded by O ($n$log$n$), assuming that the average number of edges is proportional to the density of points [16]. Notice that the MST of the Delaunay triangulated input data points gives the same result than if it were computed over a fully connected graph of those input data points.

Fig. 2,(*top-right*) shows the MST corresponding to the triangular mesh (Fig. 2(*top-left*)) computed from the edge points presented in Fig. 1(*right*); enlargements are presented in Fig. 2(*bottom*). As expected, the generated tree goes along edge points unveiling regions' contours. In addition, the algorithm generates several short branches which are removed during the following stage. Finally, as mentioned above, the MST is the acyclic subgraph of G so no closed boundaries will appear at this stage. These open contours are easily detected and connected during the next stage.

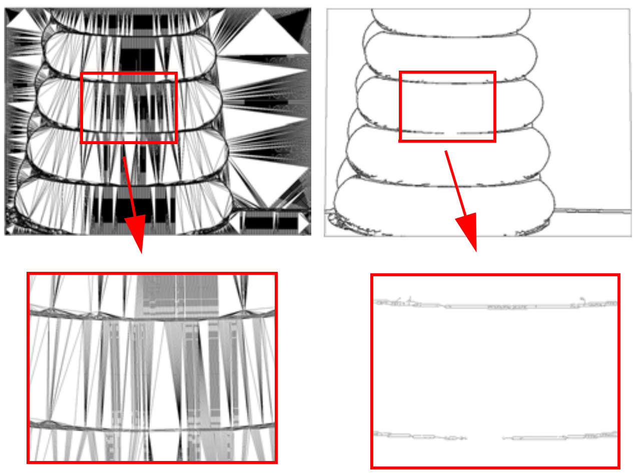
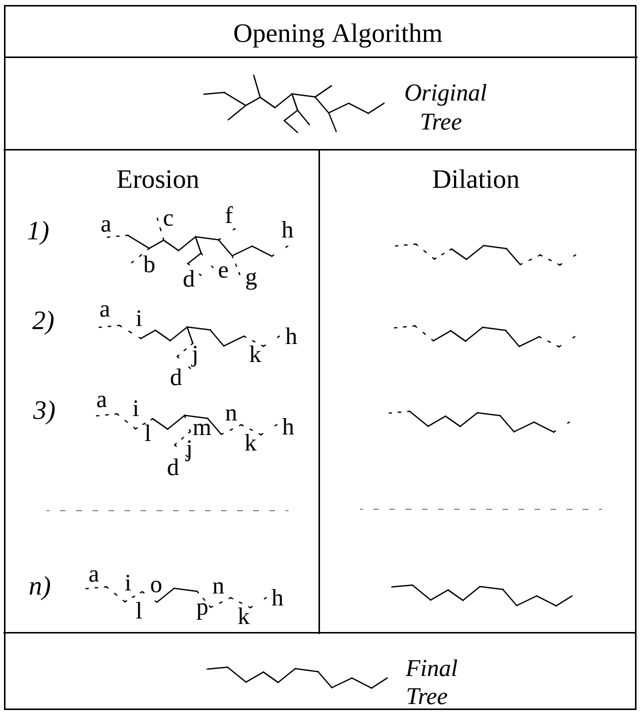


Fig. 2. (*top-left*) 2D triangular mesh of the binary edge map (*20,387* triangles). (*top-right*) Resulting MST (*11,074* edges). (*bottom-left*) Enlargement of a 2D triangular mesh section. (*bottom-right*) Enlargement of the MST corresponding to the same region.

## B. *MST Filtering and Contour Closure*

The resulting MST can be understood as a single polyline liking all the input points (differently than [6] where several polylines were computed). As mentioned above, several short branches, connected with the main path, were generated from the MST. They belong to information redundancy and noisy data, mainly in jump edge regions. The aims at this stage are two; firstly, to remove those short branches (see enlargement shown in Fig. 2(*bottom-right*)) and secondly to close open contours.

In order to perform the removal process, and by using mathematical morphology concepts, a kind of *opening algorithm* has been used. This algorithm consists in performing an iterative *erosion* process followed by a *dilation* stage applied as many times as the erosion requires. The opening algorithm assumes segments of the polyline—i.e. edges from the graph—as basic processing elements (like pixels in an intensity image). Those segments linked from only one of their defining points—so called *end segments*—are removed during the erosion stage. This stage is applied *t* times; at each iteration all the end segments of that configuration are removed. The number of iterations depends on the input binary edge map; this is one of the differences with the previous version ([6]), where *t* was a fixed value (ten iterations). In the current implementation *t* was automatically defined according to the difference between removed elements at each iteration: $\Delta = Re_{(t)} - Re_{(t-1)}$ ($Re_{(t)}$ represents the elements removed in the iteration *t*). The erosion process ended when that difference ($\Delta$) is null in at least $\tau$ consecutive iterations, $\tau$ was set to four in the current implementation. Although in the current version we have to define the threshold value $\tau$, we consider that it is more appropriate than define before hand the number of iterations [6]. We could assume that after $\tau$ consecutive iterations without changes in the number of removed edges, the erosion process has finished removing short branches and has arrived to a stability point where edges belonging to the main path are being processed. After



| *Iteration* | *1st.* | *2nd* | *3rd* | *4th* | *5th* | ..... | *nth* |
|---|---|---|---|---|---|---|---|
| *Branches' last segment* | a | i | l | o | q | ..... | w |
| | b | - | - | - | - | .... | - |
| | c | - | - | - | - | .... | - |
| | d | j | m | - | - | .... | - |
| | e | - | - | - | - | .... | - |
| | f | - | - | - | - | .... | - |
| | g | - | - | - | - | .... | - |
| | h | k | n | p | r | .... | t |
| *Removed Edges* | 8 | 3 | 3 | 2 | 2 | .... | 2 |
| *Difference* | - | 5 | 0 | 1 | 0 | .... | 0 |

$\tau$ Consecutive Iterations

Fig. 3. (top) Illustration of the opening algorithm. (bottom) Removal information used during the dilation process.

ending the erosion process, *t* dilations are performed.

Dilations are carried out over end segments left by the erosion process. It consists in putting back the segment connected with each one of the end segments present at that iteration. The number of dilations is the same than the number of erosions. Thus, in order to perform the dilation process, it is necessary to store previous stages of those end segments left by the erosion process at each iteration. Removed points that are not recovered during the dilation process are also removed from the binary array **B**(*r*, *c*). Fig. 3(*top*) shows an illustration of the proposed opening algorithm while Fig. 3(*bottom*) illustrates the data structure used during the erosion and dilation stages. Experimental results with the MST presented in Fig. 2(*top-right*) are presented in Fig. 4(*left*). An enlargement

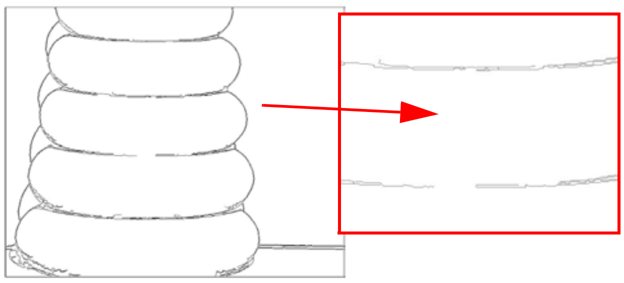


Fig. 4. *(left)* MST filtered by the opening process (*9,352 edges*). (*right*) Enlargement of the same region presented in Fig. 2(*bottom*).

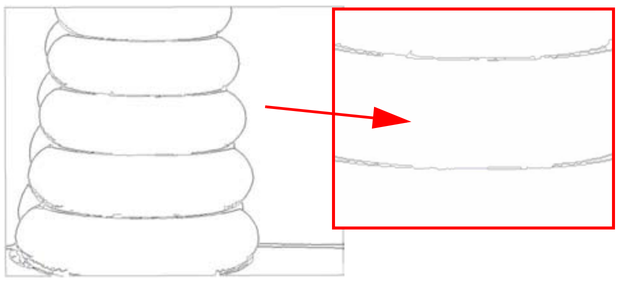


Fig. 5. (*left*) Final contour after the closure stage—minimum linking cost—(*9,426* edges). (*right*) Enlargement showing that the large gap presented in previous figures has also been closed.

of the resulting contours, obtained after the opening stage, is given in Fig. 4(*right*).

Finally, after removing short branches, the last stage of the algorithm focuses on detecting and closing open contours. Open contours were originated by construction, due to the fact that a MST is an acyclic subgraph so that it will not contain any closed contours. First of all, edge points only connected once are detected—they are easily identified from end segments left by the opening algorithm. For each one of those end points a list of candidate points is extracted from the binary array **B**(*r*, *c*). Finally, the point with a minimum linking cost is chosen to close that open boundary. These stages are further detailed below.

Given an end point $B_{(i,j)}$, defining an end segment, the set of candidate points to be linked with $B_{(i,j)}$ are selected by means of an iterative process over a dynamic window centered at that point. Initially, all those edge points, from the binary array, contained in the window $B_{(m,n)}$ are chosen as candidates: ($m = \{i, i\pm1, \ldots i\pm s, \ldots, i\pm t\}$, $n = \{j, j\pm1, \ldots j\pm s, \ldots, j\pm t\}$, $t = s+\tau$ and $\{(s<m<t)\vee(s<n<t)\}$); during the first iteration $s$ is set to zero, then after each iteration $s$ is increased by a user defined value $\tau$; threshold $\tau$ depends on the density of edge points in the binary array, in the current implementation $\tau$ was set to four.

After extracting the set of candidate points a linking cost, representing the cost of connecting each one of those candidates with the given end point $B_{(i,j)}$, is computed according to the following expression:

$$\text{Cost}_{(i,j),(u,v)} = \frac{dist3D_{(i,j),(u,v)}}{PathLength_{(i,j),(u,v)}} \qquad (1)$$

$dist3D$ represents the 3D distance between the corresponding range image points (**R**(*i*,*j*), **R**(*u*,*v*)) while $PathLength$ measure the length of the path—number of edges—linking those two points. In case of no candidate points were extracted from the current window or the *PathLength* values from those candidates to the given end point were equal or smaller than *t*, the size of the dynamic window is increased by $\tau$; so that *s* and *t*, and the process starts again by extracting a new set of candidate points. The new set of candidate points does not contain those previously studied due to the fact that the new window is only defined by the outside band. Otherwise, the point with lowest linking cost is chosen to be linked with the point $B_{(i,j)}$.

The philosophy of the proposed cost function is to link an end point with its nearest point in the 3D space, avoiding those points already connected in its neighborhood. Fig. 5 presents the final result after closing all open boundaries, by linking end points with their corresponding minimum linking cost points. Notice how the big gap presented in Fig. 4(*right*) has been correctly closed. This kind of gap could not be closed with an approach such as the one proposed by [6] where a uniform distribution of edge points is assumed.

## III. Experimental Results

Closed contour extraction is a task highly dependent on the detected edge points. Therefore, in order to test the proposed approach, independently of the given edge points, three different test methodologies have been proposed. Firstly, experimental result from a synthetic scene, containing a single object, will be presented. This synthetic range image together with its corresponding edge points (uniformly sampled through the object's edges) are considered as inputs to the system. Secondly, experimental results by using the same synthetic scene mentioned above but now adding noisy data points are presented. Finally, the proposed approach is applied to real range images, this last test include several real range images from Vision lab data base at the University of South Florida (K2T structured light sensor). Edge points were computed by means of two edge-based range image segmentation techniques ([2] and [6]), in addition by using different approximation errors—approximation error defines the density of edge points.

### A. *Synthetic Data Points*

A synthetic range image defined by *480x640* points has been used. From this synthetic range image, uniformly distributed edge points were computed according to the surface orientation discontinuities. Fig. 6 shows results from different algorithm's steps. Fig. 6(*left*) displays the triangular mesh generated by means of the 2D Delaunay triangulation algorithm. Fig. 6(*middle*) presents the computed tree, together with a sketch of the tree's branches. Finally, during the postprocessing stage, open contours (A, B and C on the illustration) were closed by using the proposed closing approach (minimum linking cost). In this particular example there are not short branches to be removed—noisy data or redundant information—hence the opening morphological operator only consists of the

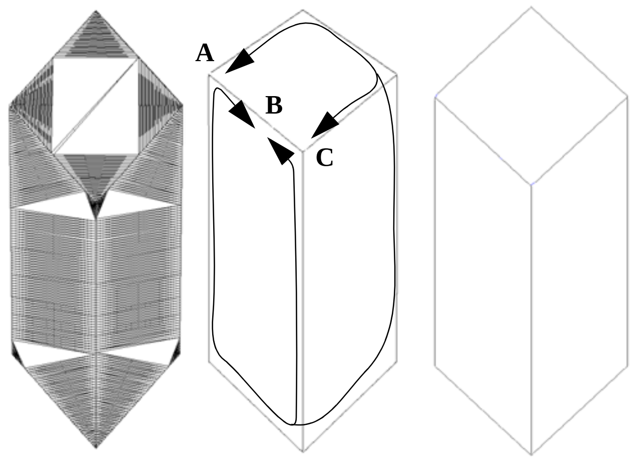


Fig. 6. Different steps of the proposed algorithm when a scene containing a single object (synthetically generated) is processed. (*left*) 2D triangular mesh (*843* triangles). (*middle*) MST of the previous triangular mesh (*597* edges). (*right*) Final result (*600* edges).

first five iterations (five erosions and five dilations).

## B. *Synthetic Data Points Plus Noisy Data*

The aim at this stage is to test the proposed technique in presence of noise. Noisy data were randomly introduced in the binary array (edge points); the corresponding 3D values were extracted from the range image. Additionally some edge points are removed assuming they are not correctly extracted by the segmentation algorithm. Different synthetic range images have been considered, Fig. 7 presents results obtained after adding noisy data points, and removing some edge points, to the synthetic range image presented in Fig. 6. Fig. 7(*top*) contains 6.5% of wrong edge points (noisy data plus removed), while Fig. 7(*bottom*) contains 9.4% of wrong edge points. Noisy data points generate short branches (Fig. 7(*middle*)) that are removed during the opening stage. Fig. 7(*right*) presents final results after filtering MST and closing open boundaries. Some of those open boundaries (see enlargement in Fig. 7(*top-middle*)) can not be closed by those techniques only based on dilation process [6].

## C. *Real Range Images*

Finally the proposed approach has been tested with several real range images obtained with the K2T structured light sensor—from the Vision lab data base at the University of South Florida (http://marathon.csee.usf.edu/range/DataBase.html). Edge points were computed by using the code presented in [2] and [6]; the difference between them is that in [2], not only rows and columns are considered but also diagonals. Edge points were computed at different approximation errors. The range image used through the paper, to illustrate the different stages of the proposed technique, consists of *480x640* points. Fig. 1(*right*) shows input edge points (*11,075* points), while the corresponding triangular mesh and MST are presented in Fig. 2. The triangular mesh contains *20,387* triangles, while its MST is defined by *11,074* edges. Notice as the MST produces a gap in the middle of the image (enlargement area) due to the fact of the low density. This gap is closed during the last stage, just after removing noisy

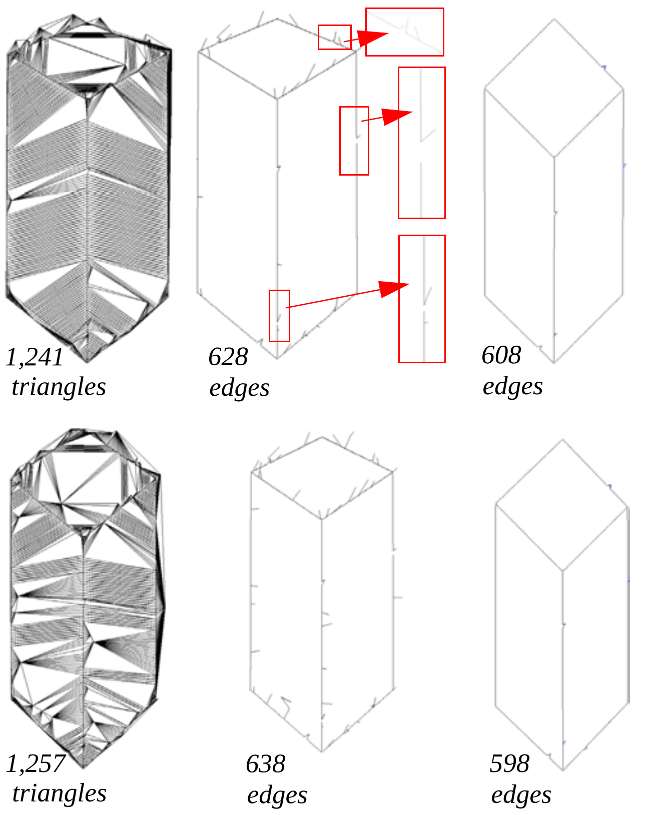


Fig. 7. The scene presented in Fig. 6 after adding noisy data points and removing some edge points. (*left*) Triangular meshes of the input data points. (*middle*) Resulting MST, short branches are generated by noisy data points (enlargements show open boundaries). (*right*) Final result after filtering noisy data points and closing open boundaries.

points. Again, techniques based on edge dilation are not able to close gaps so big like that. After removing short branches, by means of the proposed opening algorithm, those couples with minimum linking cost are connected. This final result is presented in Fig. 5 and it consists of *9,426* edges.

Fig. 8 presents three experimental results obtained after processing a range image defined by 480x640 points. Input edge points (computed at different approximation errors) are presented in Fig. 8(*left*), while the computed closed contours are shown in Fig. 8(*right*). Finally, Fig. 9 presents a range image defined by *480x640* points (*top-left*); its corresponding intensity image is presented in Fig. 9(*top-right*) as an illustration. Input edge point representations, computed at different approximation errors are presented in Fig. 9(*middle-left*) and Fig. 9(*bottom-left*). They were only computed with the code presented in [2], while in the previous examples both codes were used alternatively ([2] and [6]). Final results are showed in Fig. 9(*middle-right*) and Fig. 9(*bottom-right*).

## IV. Conclusions

This paper presents improvements over a previous graph-based approach to deal with the classical contour closure problem. This problem can be understood as the last stage of edge-based range image segmentation techniques. The proposed technique only require the

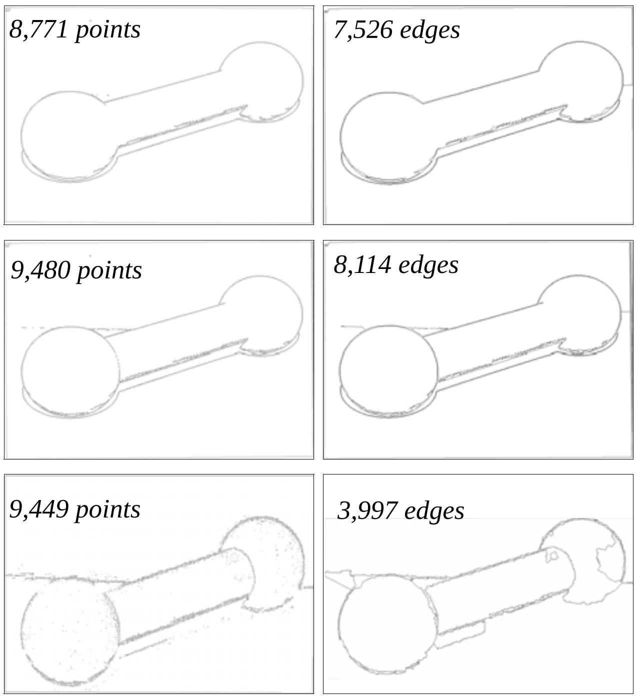


Fig. 8. (*left*) Input edge points computed at different approximation errors. (*right*) Final results computed by the proposed technique

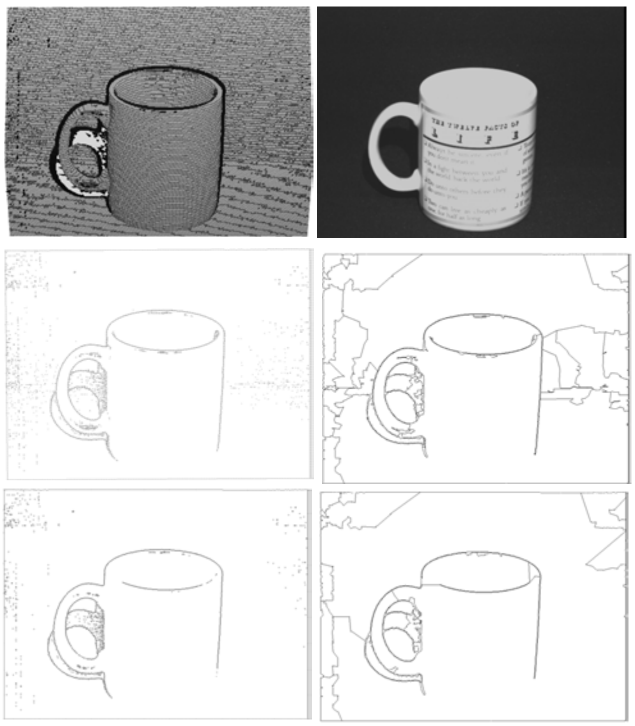


Fig. 9. (*top*) Original range image (*480x640* points) with the corresponding intensity image. (*middle-left*) Input edge points (*9.481* points). (*middle-right*) Closed boundaries extracted with the proposed technique (*6,194* edges). (*bottom-left*) Input edge points (*8.941* points). (*bottom-right*) Final result (*6,082* edges).

information about edge point positions, no other assumptions neither about the enclosed surfaces nor about the uniform edge points distribution have to be made. Assuming a range image with its corresponding edge points are given as inputs, the proposed technique consists of two stages. Firstly, the MST of a partially connected graph is computed. Next, after unveiling objects' contours a post-processing stage removes short branches and finally open boundaries are efficiently closed by minimizing a linking cost function.

## REFERENCES

[1] D. Lowe, *Perceptual Organization and Visual Recognition*, Kluwer Academic publishers, 1985.

[2] X. Jiang and H. Bunke, "Edge detection in range images based on scan line approximation", Computer Vision and Image Understanding, vol 73 No. 2, February, 1999.

[3] X Jiang, "An adaptive contour closure algorithm and its experimental evaluation", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol 22, No 11, November 2000.

[4] S. Betgé-Brezetz, R. Chatila and M. Devy, "Natural scene understanding for mobile robot navigation", IEEE Int. Conf. on Robotics and Automation, San Diego, USA, May 1994.

[5] A. Abrantes and J. Marques, "A class of constrained clustering algorithms for object boundary extraction", IEEE Trans. on Image Processing, vol. 5, No 11, November 1996.

[6] A. Sappa and M. Devy, "Fast range image segmentation by an edge detection strategy", IEEE Int. Conf. on 3-D Digital Imaging and Modeling, Quebec City, Canada, May/June 2001.

[7] Y. Xu and E. Uberbacher, "2D image segmentation using minimum spanning trees", Image and Vision Computing, 15 (1997).

[8] A. Martínez, P. Mittrapiyanuruk and A. Kak, "On combining graph-partitioning with non-parametric clustering for image segmentation", Computer Vision and Image Understanding 95 (2004).

[9] Z. Wu and R. Leahy, "Image segmentation via edge contour finding: a graph theoretic approach", IEEE Int. Conf. on Computer Vision and Pattern Recognition, Champaign, IL USA, June 1992.

[10] S. Mahamud, L. Williams, K. Thornber and K. Xu, "Segmentation of multiple salient closed contours from real images", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol 25, No 4, April 2003.

[11] A. Sanfeliu, R. Alquézar, J. Andrade, J. Climent, F. Serratosas and J. Vergés, "Graph-based representations and techniques for image processing and image analysis", Pattern Recognition 35 (2002).

[12] 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, December 1990.

[13] 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.

[14] M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, Second Edition, 2000.

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

[16] O. Faugeras, Three-Dimensional Computer Vision, A Geometric Viewpoint, The MIT Press, 1993.