Unsupervised Contour Closure Algorithm for Range Image Edge-Based Segmentation

Angel Domingo Sappa, *Member, IEEE*

***Abstract*—This paper presents an efficient technique for extracting closed contours from range images' edge points. Edge points are assumed to be given as input to the algorithm (i.e., previously computed by an edge-based range image segmentation technique). The proposed approach consists of three steps. Initially, a partially connected graph is generated from those input points. Then, the minimum spanning tree of that graph is computed. Finally, a postprocessing technique generates a single path through the regions' boundaries by removing noisy links and closing open contours. The novelty of the proposed approach lies in the fact that, by representing edge points as nodes of a partially connected graph, it reduces the contour closure problem to a minimum spanning tree partitioning problem plus a cost function minimization stage to generate closed contours. Experimental results with synthetic and real range images, together with comparisons with a previous technique, are presented.**

***Index Terms*—Image edge analysis, image segmentation, range image.**

I. Introduction

A RANGE image is generally represented by means of a two-dimensional (2-D) array $\mathbf{R}$, where each element $\mathbf{R}(r,c)$ is a scalar representing a surface point of coordinates: $(x,y,z) = (f_x(r), f_y(c), f_z(\mathbf{R}(r,c)))$ referred to a local coordinate system [1]. Range image segmentation algorithms concern the grouping of three-dimensional (3-D) input data points into disjoint homogeneous regions. Formally, a segmentation of a range image $\mathbf{R}$ into n regions $r_1, \ldots, r_n$ is defined by the following properties [2]: *1)* $r_1 \cup r_2 \cup \cdots \cup r_n = \mathbf{R}$; *2)* r_i is a connected region $\forall r_i \in \mathbf{R}$; *3)* $r_i \cap r_j = \emptyset, \forall i \neq j$; *4)* $\mathrm{P}(r_i) = \mathrm{TRUE}, \forall r_i \in \mathbf{R}$; *5)* $\mathrm{P}(r_i \cup r_j) = \mathrm{FALSE}, \forall i \neq j$; where (r_i) is a similarity predicate over the points in set r_i—such as they belong to the same surface—and $\emptyset$ is the null set.

Most of the proposed approaches, in general, can be divided into two categories: *a) region-based approaches* and *b) edge-based approaches*. Alternatively, edge-based approaches can be used to support region-based segmentation algorithm—usually referenced as hybrid approaches. Each one of them has its own advantages and disadvantages. While region-based techniques have threshold problems, or surface fitting's definition, edge-based approaches have as a weakest point the contour closure extraction; although most of the edge-based range image segmentation approaches succeed in finding a good sampling of object's edges, most of them fail in computing regions' closed boundaries. The work in this paper is focused on extracting closed contours from range images' edge points.

Human being can easily extract closed 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 1920s by Gestalt psychologists. The hierarchical grouping principles proposed by Gestalt psychologists embodied such concepts as grouping by proximity, similarity, continuation, closure, and symmetry [3].

From the 3-D computer vision point of view, several techniques have been proposed in the literature following different strategies (some of them will be summarized in the next section). Classically, they were inspired from the 2-D image processing field; hence, some of the proposed 3-D contour closure approaches have been based on the use of morphological operators (e.g., [4]–[6]). Other approaches, close related to the topic tackled in this paper, are focused on extracting the external contour of an object (object's boundary) but not on extracting the contour of every single region, r_i, defining the object. These approaches try to link edge points according to local measures of continuity and smoothness, with no a prior information about the object shape. These techniques include several well-know algorithms from different fields (deformable models, constrained clustering and data ordering); see [7] for further details. Differently than these approaches, in this paper, we tackle the problem from another point of view by proposing a graph-based technique.

Graph theory has long been used in the 2-D image segmentation problem (e.g., [7]–[12]). Our proposed approach differs from them not only due to the fact that it is focussed on range image processing but also on the following aspects. *First*, some of those techniques were meant for partitioning a gray-level image into connected homogeneous regions—region-based approaches [8]–[10]; for example, [8] introduces a 2-D image segmentation algorithm using minimum spanning trees. By minimizing the sum of gray levels variations a minimum

Manuscript received October 10, 2004; revised January 17, 2005. This work was supported in part by the Government of Spain under the CICYT project TRA2004-06702/AUT and in part by The Ramón y Cajal Program. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Eli Saber.

The author is with the Computer Vision Center, 08193 Bellaterra, Barcelona, Spain (e-mail: angel.sappa@cvc.uab.es).

Digital Object Identifier 10.1109/TIP.2005.860612

spanning tree is partitioned into subtrees, representing different homogeneous regions. Similarly, [9] proposes a graph-partitioning with nonparametric clustering approach for 2-D image segmentation. *Second*, as mentioned above, other approaches were proposed for extracting the external contours of a single object (object's boundary); hence, they are not able to compute a closed contour for every single region r_i (e.g., [7] and [11]). [7] 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, [11] 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, but again one single closed contour per object is computed.

The use of graph-based representations and techniques for image processing and analysis has been discussed on [12]. 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 3-D 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.

Similarly to graph-based approaches mentioned above, in this paper, we propose to tackle the contour closure problem as a graph-related problem. This work is an extension of a previous work presented in [13]. Improvements in front of that work are in two key points. First, the current version avoids user defined parameters, such as the number of iterations during the filtering stage. Second, the new version of the algorithm is intended for extracting closed contours, while [13] was only able to compute a set of open polylines defining the contours. The next section will summarize some of the approaches proposed in the literature to compute contour closure from range images' edge points.

II. Previous Approaches

Edge detection is the first and most important stage of human visual process as discovered by [14]. Contour extraction is a common problem within the edge-based approaches (2-D or 3-D). However, so far, it did not get so much attention. Classically, most of the authors have preferred to develop solutions for their specific problems. For example, [15] 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 [6], [16]; therefore, contour and region are simultaneously extracted. Reference [6] deals with the understanding of natural scene from range images; in the area close to the ending point of an edge, a local analysis of the depth is performed in order to select an optimal edge dilation direction. [4] 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

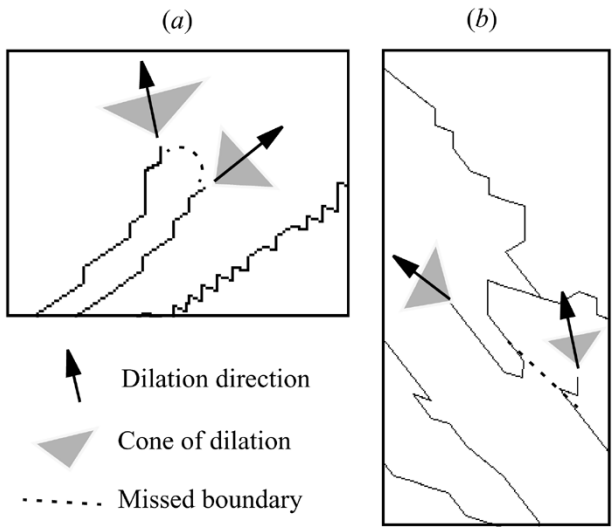

Fig. 1. Examples of some atypical open contour configurations that have appeared after processing real range images.

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 [5]. There, the geometry of contours is taken into account in order to apply the dilation—the dilation process is restricted to one direction. This direction-guided dilation is applied only over the ending points of the contours. Direction of dilation is determined from the last three points connected with the ending point. Thus, from these four points, one gets an average direction which will be considered as the direction of dilation. Since some uncertainty still remains about the direction vector, it is suggested to include surrounding points to that direction. Instead of being carried out along a single straight line, the dilation is applied along a conic region; the apex of that cone is the ending point of the open contour. This approach can deal with thin regions. Sometimes, however, considering the last three segments of an open contour only—the last point plus three previous ones over that contour—cannot be enough to obtain the good boundary direction. Normally, this technique cannot handle most of the possible pathological cases emerging when real range images are processed (see illustration shown in Fig. 1). Those problems may appear when final points are affected by noisy data or when open contours describe curved shapes; in this last case, the direction vector used for the dilation may not be representative of the boundary direction—in addition to the noisy data that direction vector is also dependent on the boundary's curvature and the edge point density.

Most of the aforementioned techniques are based on the consideration that closed contours can be extracted by dilating the input edge map and that every contour gap will be closed. It may be good to deal with the typical open contour but the common issue of these techniques is to find the good dilation direction in a general way.

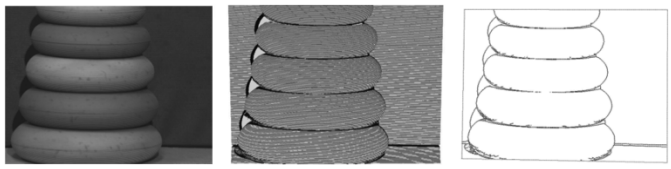

Fig. 2. (Left) Intensity image (640 × 480). (Middle) Input range image (640 × 480). (Right) Input binary array of edge points corresponding to the given range image (11 075 points).

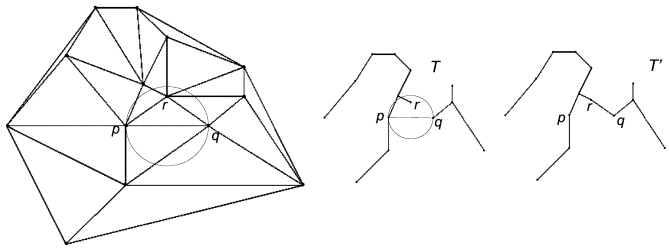

Fig. 3. (Left) 2-D Delaunay triangulation of a given set of points tree connecting all the input points. (Right) The MST computed from the 2-D Delaunay triangulation.

III. Contour Closure Algorithm

The proposed technique is carried out with no a prior information about the object's surface shape. It treats the contour closure problem as a graph partitioning one. Assuming that a range image, together with its corresponding edge point information are given as input, the proposed technique consists of three consecutive stages. First, edge points, represented in a 2-D space, are triangulated. Then, that triangular mesh is depicted as a weighted graph where points of the mesh are the nodes and the mesh's edges become graph's links. Weight (or cost) associated to each edge corresponds to the 3-D distance between the two range image points connected by that edge. Second, the minimum spanning tree (MST) is computed from the weighted graph. Third, a postprocessing technique, based on a filtering technique plus a cost function minimization, generates a single path describing the region's boundaries. These stages are described below.

A. Two-Dimensional Triangulation and Graph Generation

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 nonedge points with a 0. These points are supposed to be previously computed by some edge-based range image segmentation algorithm [see illustration in Fig. 2 (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 2-D Delaunay triangulation algorithm has been finally adopted.

Let $P = \{B_i = (r_i,c_i) | i = 1,\ldots,n\}$ be a set of edge points in the binary array $\mathbf{B}(r,c)$, its 2-D triangular mesh is a piecewise linear partition consisting of triangles connected along their edges. Formally, a 2-D 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 [17]. In other words, the Delaunay triangulation of the input edge points P will connect every point with its nearest, as we were looking for.

Finally, 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 3-D distance between the range image points linked by that edge, $\{E_w = (B_i,B_j,w_{i,j}) | (w_{i,j} = \mathrm{dist}(R_{(r_i,c_i)},R_{(r_j,c_j)})) \wedge (i,j = 1,\ldots,n, i \neq j)\}$.

B. Minimum Spanning Tree Generation

During this stage, the shortest—cheapest—path linking all the edge points will be computed. Problems such as these have been called in the literature as the traveling salesman problems. 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 $\mathrm{O}(m \log n)$ by applying Kruskal's algorithm [18]. In the current implementation, due to the fact that G is a 2-D Delaunay triangulation of the n input edge points, but not a fully connected graph, the cost can be bounded by $\mathrm{O}(n \log n)$, assuming that the average number of edges is proportional to the density of points [19]. 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. In other words, as it has been stated in [20], the MST of a set of points P (in any dimension) is a subgraph of the Delaunay triangulation. This can be briefly proved as follow. Let T be the MST of a given set of points and $\mathrm{w}(T)$ its corresponding total cost; let p and q be any two points such that $\overline{pq}$ is an edge of T. Suppose to the contrary that $\overline{pq}$ is not an edge in the Delaunay triangulation. This implies that there is no empty circle passing through p and q, and in particular, the circle whose diameter is the segment $\overline{pq}$ contains another point, let call it r (see illustration in Fig. 3). The removal of $\overline{pq}$ from the MST splits the tree into two subtrees. Assume without loss of generality that r lies in the same subtree as p. Now, remove the edge $\overline{pq}$ from the MST and add the edge $\overline{rq}$ in its place. The result will be a spanning tree T' whose cost can be easily computed from $\mathrm{w}(T)$ as: $\mathrm{w}(T') = \mathrm{w}(T) - \overline{rq} + \overline{pq} < \mathrm{w}(T)$; the last inequality follows because $\overline{pq}$ is the diameter of the circle, implying that $\overline{pq} < \overline{rq}$. This contradicts the hypothesis that T is the MST, completing the proof.

Fig. 4 (top right) shows the MST corresponding to the triangular mesh [Fig. 4 (top left)] computed from the edge points presented in Fig. 2 (right); enlargements are presented in Fig. 4 (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 contours will appear at this stage. These open contours are easily detected and connected during the next stage.

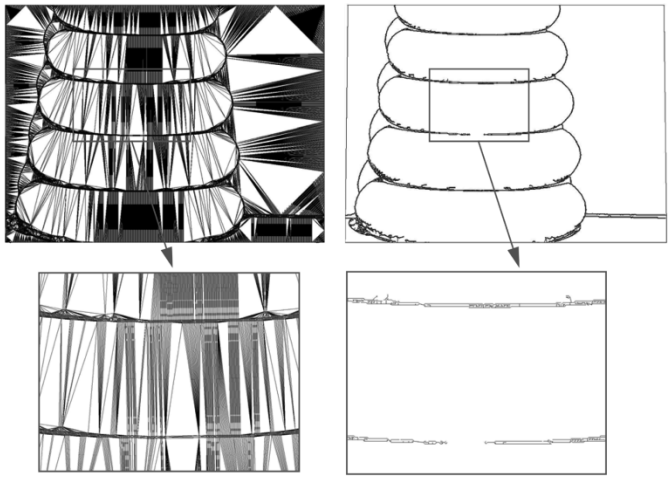

Fig. 4. (Top left) 2-D triangular mesh of the binary edge map (20 387 triangles). (Top right) Resulting MST (11 074 edges). (Bottom left) Enlargement of a 2-Dtriangular mesh section. (Bottom right) Enlargement of the MST corresponding to the same region.

C. MST Filtering and Contour Closure

The resulting MST can be understood as a single polyline linking all the input points (differently than in [13] where several polylines, defining open contours, 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; first, to remove those short branches (see enlargement shown in Fig. 4 (bottom right)), and, second, to close open contours.

In order to perform the removal process, and by using mathematical morphology concepts, a kind of *opening algorithm* has been implemented. This algorithm consists of 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 n times and at each iteration, all the end segments of that configuration are removed. The number of iterations depends on the input binary edge map. On the contrary with [13], where n was a fixed value (ten iterations), in the current implementation n is automatically computed according to the difference between removed elements at each iteration: $\Delta = \mathrm{Re}_{(t)} - \mathrm{Re}_{(t-1)}$ ($\mathrm{Re}_{(t)}$ represents the elements removed in the iteration t). The erosion process ended when that difference (Δ) is null in at least τ consecutive iterations, τ was set to four in the current implementation. Although in the current version we have to define the threshold value τ, we consider that it is more appropriate than before hand defining the number of iterations [13]. We could assume that after τ 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 ending the erosion process, n dilations are performed.

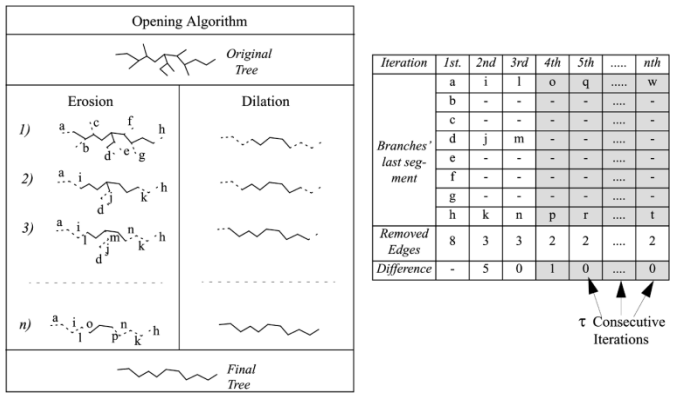

Fig. 5. (Left) Illustration of the opening algorithm. (Right) Removal stages used during the dilation process.

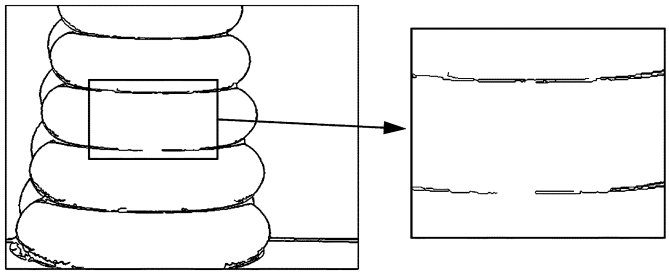

Fig. 6. (Left) MST filtered by the opening process (9 352 edges). (Right) Enlargement of the same region presented in Fig. 4 (bottom).

Dilations are carried out over end segments left by the erosion process. It consists of putting back the segment connected with each one of the end segments present at each 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. Removed points that are not recovered during the dilation process are also removed from the binary array $\mathbf{B}(r, c)$. Fig. 5 (left) shows an illustration of the proposed opening algorithm while Fig. 5 (right) illustrates the data structure used during the erosion and dilation stages. Experimental results with the MST presented in Fig. 4 (top right) are presented in Fig. 6 (left). An enlargement of the resulting contours, obtained after the opening stage, is given in Fig. 6 (right).

Finally, after removing short branches, the last stage of the algorithm focuses on detecting and closing open contours. This is the second improvement in front of [13], where only a set of open polylines were computed but not closed contours. Open contours were originated by construction, due to the fact that a MST is an acyclic subgraph so that it does not contain any closed contours. First, 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 $\mathbf{B}(r, c)$. Finally, the point with a minimum linking cost is chosen to close that open contour. 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

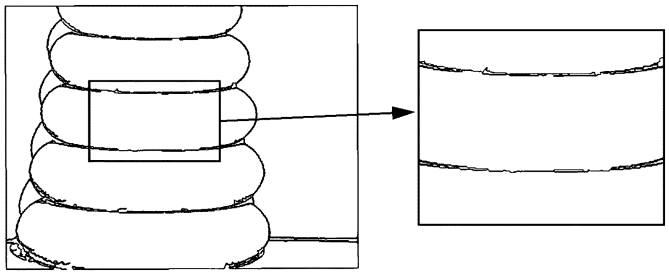

Fig. 7. (Left) Final contour after the closure stage—minimum linking cost—(9 426 edges). (Right) Enlargement showing that the large gap presented in previous figures has also been closed.

binary array, contained in the window $B_{(m,n)}$ are chosen as candidates: $(m = \{i, i \pm 1, \ldots i \pm s, \ldots, i \pm t\}, n = \{j, j \pm 1, \ldots j \pm s, \ldots, j \pm t\}, t = s + \tau$ 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 the user defined value τ; threshold τ depends on the density of edge points in the binary array, in the current implementation τ 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:

$$\mathrm{Cost}_{(i,j),(u,v)} = \frac{\mathrm{dist3}D_{(i,j),(u,v)}}{\mathrm{PathLength}_{(i,j),(u,v)}} \tag{1}$$

$\mathrm{dist3}D$ represents the 3-D distance between the corresponding range image points $[\mathbf{R}(i,j), \mathbf{R}(u,v)]$ while $\mathrm{PathLength}$ 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 $\mathrm{PathLength}$ values from those candidates to the given end point were equal or smaller than t, the size of the dynamic windows is increased by τ, 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 edge point in the 3-D space $(\mathrm{dist3}D)$, avoiding those points already connected in its neighborhood $(\mathrm{PathLength})$. Fig. 7 presents the final result after closing all open boundaries, by linking end points with their corresponding minimum linking cost points. Although a regular sampling of edge points is expected to be given, notice how the big gap presented in Fig. 6 (right) has been also correctly closed.

IV. Experimental Results

Closed contour extraction is a task highly dependent on the quality of detected edge points. Therefore, in order to test the proposed approach, independently of the quality of the given edge points, three different test methodologies have been proposed. First, experimental result from synthetic scenes, containing a single object each, will be presented. These synthetic range images together with their corresponding edge points (uniformly sampled through the object's edges) are considered as inputs to the system. Second, experimental results by using the same synthetic scenes mentioned above but now adding noisy data points are presented. Finally, the proposed approach is applied to real range images; this last test includes several real range images from two different range scanners. Edge points were computed by means of two edge-based range image segmentation techniques [4], [13], in addition by using different approximation errors—approximation error defines the edge points' density.

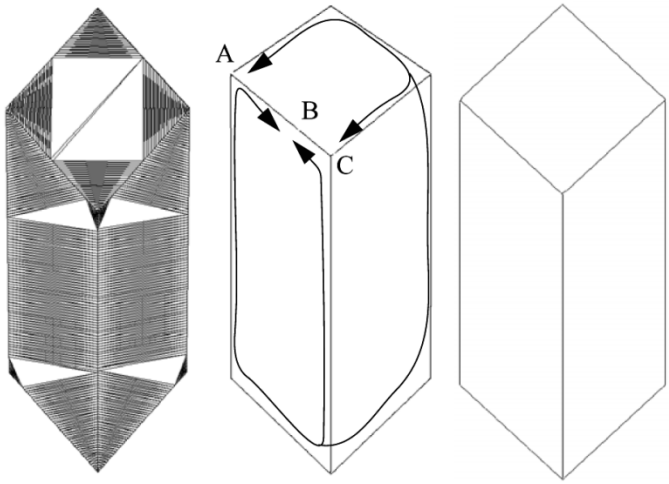

Fig. 8. Different steps of the proposed algorithm when a scene containing a single object (synthetically generated) is processed. (Left) 2-D triangular mesh (843 triangles). (Middle) MST of the previous triangular mesh (597 edges). (Right) Final result (600 edges).

A. Synthetic Data Points

Synthetic range images defined by 480×640 points have been used. From those synthetic range images, uniformly distributed edge points were computed according to the surface orientation discontinuities. Fig. 8 shows results from different algorithm's steps, when a scene containing a single object is processed. Fig. 8 (left) displays the triangular mesh generated by means of the 2-D Delaunay triangulation algorithm. Fig. 8 (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 in the illustration) were closed by using the proposed closing approach (minimum linking cost). In this particular example, there are no short branches to be removed—noisy data or redundant information—hence, the opening morphological operator only consists of the 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 3-D 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. 9 presents results obtained after adding noisy data points, and removing some edge points, to the synthetic range image presented in Fig. 8. Fig. 9 (top) contains

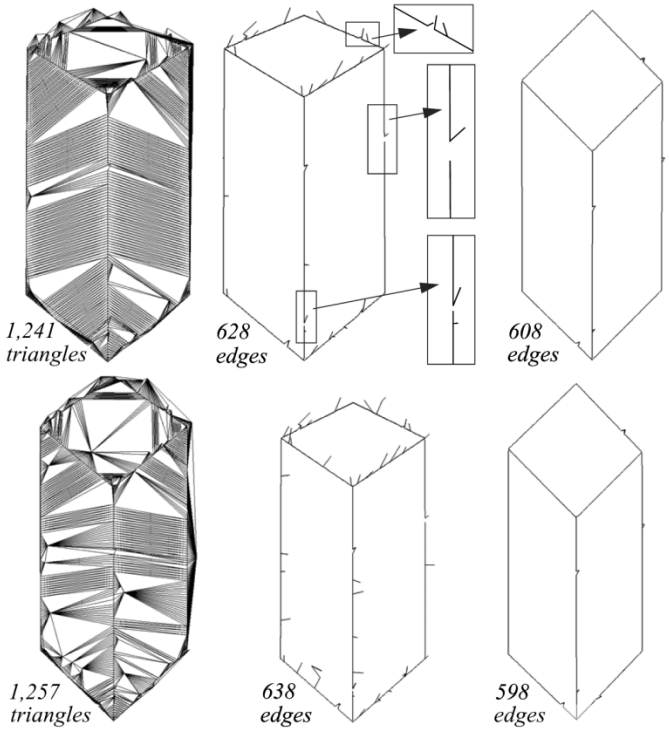

Fig. 9. Scene presented in Fig. 8 after adding noisy data points and removing some edge points. (Left) Triangular meshes of the input data points. (Middle) Resulting MST, short branches are generated by noisy data points (enlargements show open boundaries). (Right) Final result after filtering noisy data points and closing open boundaries.

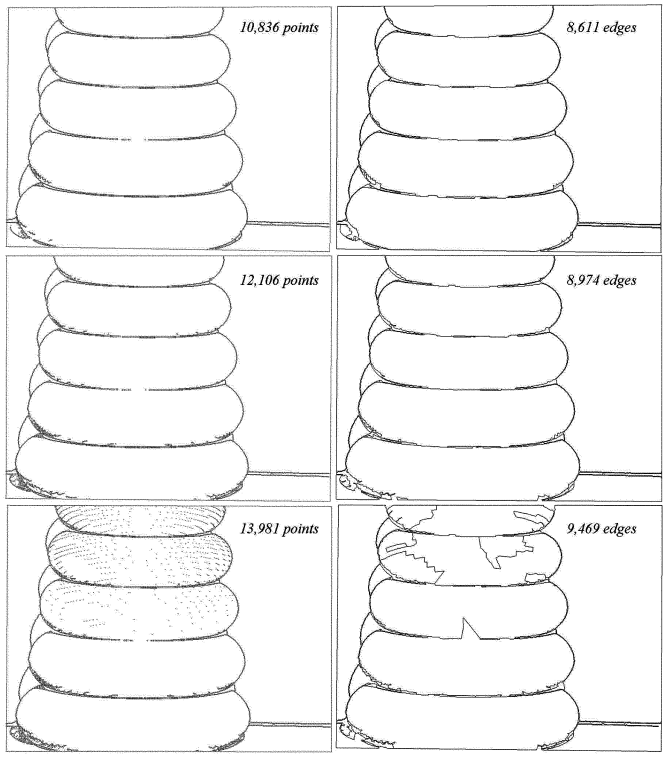

Fig. 10. (Left) Edge points computed at different approximation errors. (Right) Final representations corresponding to those edge points.

6.5% of wrong edge points (noisy data plus removed), while Fig. 9 (bottom) contains 9.4% of wrong edge points. Noisy data points generate short branches [Fig. 9 (middle)] that are removed during the opening stage. Fig. 9 (right) presents final results after filtering MST and closing open boundaries. Some of those open boundaries (see enlargement in Fig. 9 (top middle)) cannot be closed by those techniques only based on dilation process—Section II.

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)—and with the OSU's Minolta 700 range scanner (http://sample.eng.ohio-state.edu/~sample/data/3DDB/RID/minolta). Edge points were computed by using the code presented in [4] and [13]; the difference between them is that [4] not only deals with rows and columns 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 480×640 points. Fig. 2 (right) shows input edge points (11 075 points), while the corresponding triangular mesh and MST are presented in Fig. 4. The triangular mesh contains 20 387 triangles, while its MST is defined by 11 074 edges. Notice as the low density of points in the middle of the image (enlargement area) produces a gap by the MST algorithm. This gap is closed during the last stage, just after removing noisy 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. 7 and it consists of 9 426 edges. Fig. 10 presents experimental results with the same range image, but after modifying the edge points' density (by increasing and reducing the approximation error during the edge point extraction, [4] and [13]). Fig. 10 (left) presents input edge points while Fig. 10 (right) shows final results.

Fig. 11 presents three experimental results obtained after processing a range image defined by 480×640 points. Input edge points (computed at different approximation errors) are presented in Fig. 11 (left), while the computed closed contours are shown in Fig. 11 (right). Another example, from the same database, is presented in Fig. 12. The original range image is defined by 480×640 points (top left), its corresponding intensity image is presented in Fig. 12 (top right). Input edge point representations, computed at different approximation errors, are presented in Fig. 12 (middle left) and (bottom left). They were computed with the code presented in [4], while in the previous examples both codes were alternatively used ([4] and [13]). Final results are shown in Fig. 12 (middle right) and (bottom right).

Fig. 13 presents experimental results by using range images from the OSU's Minolta 700 range scanner. These range images are defined by 200×200 points. Edge points [Fig. 13 (middle)], computed by using [13] are used as input to the proposed algorithm. The resulting closed contours are presented in Fig. 13 (right).

Finally, a comparison between the proposed technique and the one presented in [4] has been carried out. The objective

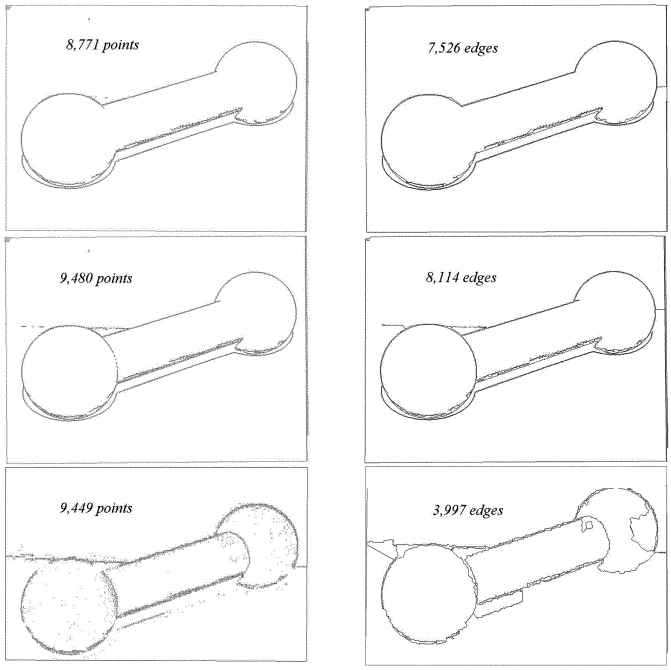

Fig. 11. (Left) Input edge points computed at different approximation errors. (Right) Final results computed by the proposed technique.

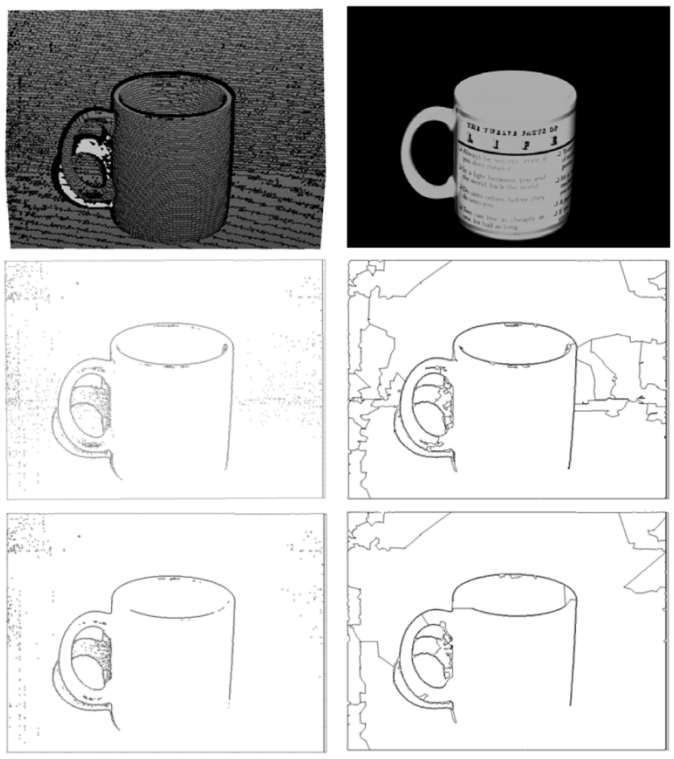

Fig. 12. (Top) Original range image (480 × 640 points) with the corresponding intensity image. (Middle left) Input edge points (9 481 points). (Middle right) Closed boundaries extracted with the proposed technique (6 194 points). (Bottom left) Input edge points (8 941 points). (Bottom right) Final result (6 082 points).

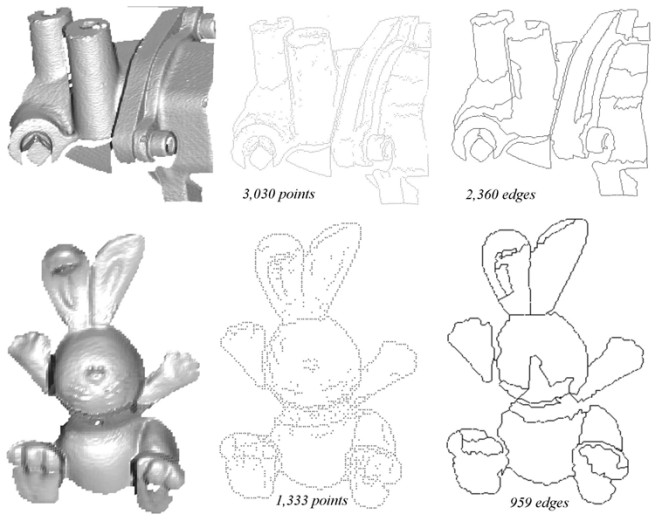

Fig. 13. (Left) Original range images (200 × 200 points). (Middle) Input edge points, computed with [13]. (Right) Final closed boundaries computed with the proposed technique.

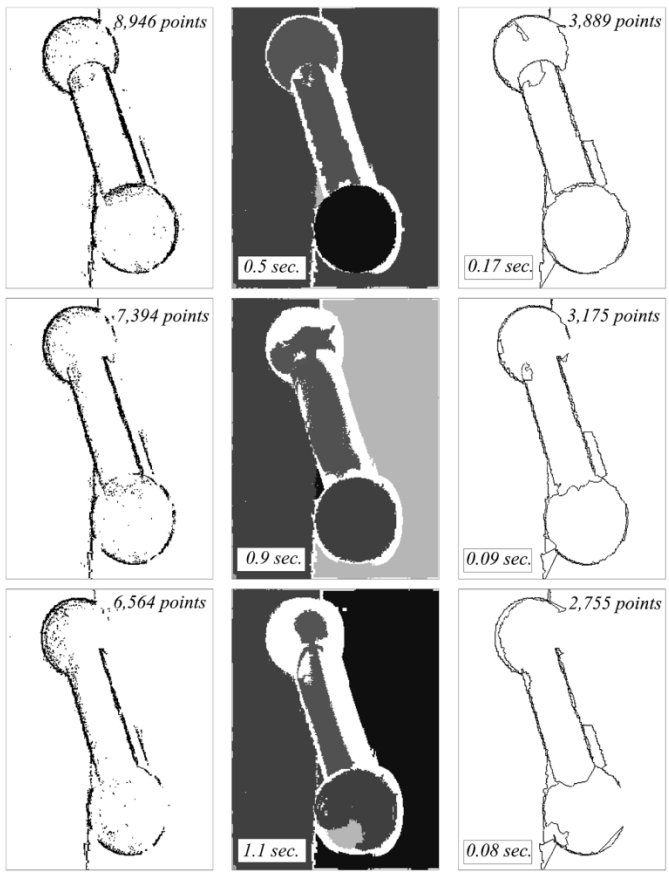

Fig. 14. (Left) Input edge points (computed from [4]). (Middle) Regions corresponding to the given inputs, computed with the region-based approach presented in [4] (not only the edge points presented at the left are considered as input but also the corresponding range image's points are needed in order to compute surfaces' parameters). (Right) Final result computed with the proposed technique (only edge points are considered as input).

of the comparison has been to compute the final representation by using the same input (edge points). Fig. 14 (left) presents different input edge point representations computed by [4]—they were also used as input by the contour closure algorithm presented in [4] and by the algorithm proposed in this paper. Fig. 14 (top left) corresponds to the highest density of edge points (8 946 points), while Fig. 14 (middle left) and

(bottom left) contain a lower density of edge points, 7 394 and 6 564 points, respectively. Regions computed with [4] are presented in Fig. 14 (middle column), while closed contours generated by the proposed technique are presented in Fig. 14 (right). Additionally, CPU time required to compute each one of these final representations is also provided. Notice that the objective is to compare the final representation computed by each algorithm when the same set of edge points are used as inputs. Remember that the proposed technique is not focused on edge point generation but in closed contour extraction, independently of the input quality. Another point that deserves to be highlighted is the fact that the results presented in Fig. 14 (middle column) were computed by using a kind of region-based algorithm [4] (see Section II for further details). On the contrary, the technique proposed in this paper only takes into account the edge points given as input; any information concerning the surface shape is required. This explains why the CPU processing time decreases when the amount of edge points contained in the input also decreases (Fig. 14). Finally, no careful tuning of the user-defined parameters required by [4] has been performed; in other words, the results presented in Fig. 14 (middle column) were not intentionally computed. They were generated by using the tuning provided in the code of [4].

V. Conclusion

This paper presents a 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 requires the information about edge point positions, no other assumption about the enclosed surfaces has to be considered. Assuming a range image with its corresponding edge points are given as inputs, the proposed technique consists of three stages. First, a 2-D triangular mesh of those input edge points is generated. Next, the corresponding MST of that partially connected graph is obtained. Finally, a postprocessing stage removes short branches and connects open boundaries. Experimental results with several range images and comparisons with a previous technique show the performance of the proposed technique when different inputs (edge points) are considered.

References

[1] M. A. Garcia and A. Sappa, "Efficient generation of discontinuity-preserving adaptive triangulations from range images," *IEEE Trans. Syst., Man Cybern. B, Cybern.*, vol. 34, no. 5, pp. 2003–2014, Oct. 2004.

[2] A. Hoover *et al.*, "An experimental comparison of range image segmentation algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 7, pp. 673–689, Jul. 1996.

[3] D. Lowe, *Perceptual Organization and Visual Recognition*. Norwell, MA: Kluwer, 1985.

[4] X. Jiang and H. Bunke, "Edge detection in range images based on scan line approximation," *Comput. Vis. Image Understand.*, vol. 73, no. 2, pp. 183–199, Feb. 1999.

[5] X. Jiang, "An adaptive contour closure algorithm and its experimental evaluation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1252–1265, Nov. 2000.

[6] S. Betgé-Brezetz, R. Chatila, and M. Devy, "Natural scene understanding for mobile robot navigation," presented at the IEEE Int. Conf. Robotics and Automation, San Diego, CA, May 1994.

[7] A. Abrantes and J. Marques, "A class of constrained clustering algorithms for object boundary extraction," *IEEE Trans. Image Process.*, vol. 5, no. 11, pp. 1507–1521, Nov. 1996.

[8] Y. Xu and E. Uberbacher, "2D image segmentation using minimum spanning trees," *Image Vis. Comput.*, vol. 15, pp. 47–57, 1997.

[9] A. Martínez, P. Mittrapiyanuruk, and A. Kak, "On combining graph-partitioning with nonparametric clustering for image segmentation," *Comput. Vis. Image Understand.*, vol. 95, pp. 72–85, 2004.

[10] Z. Wu and R. Leahy, "Image segmentation via edge contour finding: A graph theoretic approach," presented at the IEEE Int. Conf. Computer Vision and Pattern Recognition, Champaign, IL, Jun. 1992.

[11] S. Mahamud, L. Williams, K. Thornber, and K. Xu, "Segmentation of multiple salient closed contours from real images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 4, pp. 433–444, Apr. 2003.

[12] A. Sanfeliu *et al.*, "Graph-based representations and techniques for image processing and image analysis," *Pattern Recognit.*, vol. 35, pp. 639–650, 2002.

[13] A. Sappa and M. Devy, "Fast range image segmentation by an edge detection strategy," presented at the IEEE Int. Conf. 3-D Digital Imaging and Modeling, Quebec City, QC, Canada, May/Jun. 2001.

[14] W. E. Grimson, *From Images to Surfaces*. Cambridge, MA: MIT Press, 1981, pp. 3–5.

[15] E. Al-Hujazi and A. Sood, "Range image segmentation with applications to robot bin-picking using vacuum gripper," *IEEE Trans. Syst. Man. Cybern.*, vol. 20, no. 6, pp. 1313–1325, Dec. 1990.

[16] O. P. Bellon, A. Direne, and L. Silva, "Edge detection to guide range image segmentation by clustering techniques," presented at the IEEE Int. Conf. Image Processing, Kobe, Japan, Oct. 1999.

[17] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, 2nd ed. New York: Springer-Verlag, 2000.

[18] K. Rosen, *Discrete Mathematics and Its Applications*, 2nd ed. New York: McGraw-Hill, 1990.

[19] O. Faugeras, *Three-Dimensional Computer Vision, a Geometric Viewpoint*. Cambridge, MA: MIT Press, 1993.

[20] F. Preparata and M. Shomos, *Computational Geometry: An Introduction*. New York: Springer-Verlag, 1985.

Angel Domingo Sappa (S'94–M'00) received the electro-mechanical engineering degree from the National University of La Pampa, General Pico-La Pampa, Argentina, in 1995, and the Ph.D. degree in industrial engineering from Polytechnic University of Catalonia, Barcelona, Spain, in 1999.

From 1999 to 2002, he undertook postdoctorate research at the LAAS-CNRS, Toulouse, France, and at Z+F UK, Ltd., Manchester, U.K. From September 2002 to August 2003, he was with the Informatics and Telematics Institute, Thessaloniki, Greece, as a Marie Curie Research Fellow. Since September 2003, he has been with the Computer Vision Center, Barcelona, as a Ramón y Cajal Research Fellow. His research interests are focused on range image analysis, 3-D modeling, and model-based segmentation.