Cost-based closed-contour representations

Angel D. Sappa
Computer Vision Center
Edifici o Campus UAB
08193 Bellaterra, Barcelona, Spain
E-mail: angel.sappa@cvc.uab.es

Boris X. Vintimilla
Vision and Robotics Center
Department of Electrical and Computer Science Engineering
Escuela Superior Politécnica del Litoral
Campus Gustavo Galindo Km 30.5 vía Perimetral
09015863 Guayaquil, Ecuador

Abstract. *This paper presents an efficient technique for linking edge points in order to generate a closed-contour representation. It is based on the consecutive use of global and local schemes. In both cases it is assumed that the original intensity image, as well as its corresponding edge map, are given as inputs to the algorithm. The global scheme computes an initial representation by connecting edge points minimizing a global measure based on spatial information (3D space). It relies on the use of graph theory and exploits the edge points' distribution through the given edge map, as well as their corresponding intensity values. At the same time spurious edge points are removed by a morphological filter. The local scheme finally generates closed contours, linking open boundaries, by using a local cost function that takes into account both spatial and topological information. Experimental results with different images, together with comparisons with a previous technique, are presented. © 2007 SPIE and IS&T.* [DOI: 10.1117/1.2731799]

1 Introduction

Edge detection is the first and most important stage of human visual process.[1] During the last few decades, several edge-point detection algorithms were proposed. In general, these algorithms are based on partial derivatives (first and second derivative operators) of a given image. Hence, the resulting edge maps are composed by a set of edge points arranged over the boundaries of the different regions contained in the image. Additionally, edge maps usually contain gaps as well as false edge points generated by noisy data. Although useful, edge points alone generally do not provide meaningful information about the image content, so a high-level structure is required (e.g., to be used by scene understanding algorithms). From a given edge map the most direct high-level representation consists of computing closed contours—linking edge points by proximity, similarity, continuation, closure, and symmetry. Something that is very simple and almost a trivial action for the human being becomes a difficult task when it should be automatically performed.

Broadly speaking, two different approaches for linking edge points have been proposed in the literature: (1) *perceptual-based approaches* (e.g., Refs. 2–4) and (2) *general-purpose approaches* (e.g., Refs. 5–7). The former ones are based on finding salient closed boundaries by using Gestalt's law of perception. These approaches are designed to solve a specific grouping problem, such as grouping edge points into smooth closed contours, under quite constrained scenarios. They produce acceptable results on images for which their assumptions hold. On the contrary, the latter ones are able to handle any kind of images. No prior knowledge about the number of objects contained in the scene is required, nor is a constraint about maximum length of the gaps[2] nor about smoothness continuity[8] imposed. It should be noticed that the main target of general-purpose approaches is to compute closed-contour representations by linking edge points, which could be useful for a further high-level processing. Since only edge-point positions are used, computed boundaries do not necessarily correspond to a boundary between two regions. The current paper falls into this second category.

Notice that although perceptual-based approaches and general-purpose approaches pursue the generation of a closed-contour representation, their underlying philosophy is significantly different, since they tackle different goals and applications. In general, perceptual-based approaches incorporate mid-/high-level cues to link edges, while general-purpose approaches could be understood as low-level edge-linking methods that would need a further processing to generate high-level descriptions.

Several general-purpose techniques have been presented for linking edge points in order to recover closed contours. According to the way edge map information is used, they can be divided into two categories: (1) *local approaches*, which work over every single edge point, and (2) *global approaches*, which work over the whole edge map at the same time. Alternatively, *hybrid approaches* that combine both techniques, or use not only edge map information but also enclosed information (e.g., color), can be found.[6,9,10] In general, most of the techniques based on local information

Paper 06120RR received Jul. 5, 2006; revised manuscript received Jan. 22, 2007; accepted for publication Jan. 29, 2007; published online Apr. 24, 2007.
1017-9909/2007/16(2)/023009/9/$25.00 © 2007 SPIE and IS&T.

rely on morphological operators applied over edge points. The former works on edge-point linking by using morphological operators to compute closed boundaries by thinning current edge points.[11] However, common problems of thinning algorithms are that, in general, they distort the shape of the objects and big gaps cannot be properly closed. In order to avoid these problems, Ref. 12 introduces the use of morphological operators together with chamfer maps. Experimental results using simple synthetic-like images, with closely spaced unconnected edges, which do not contain spurious nor noisy edge points, are presented.

A real-time edge-point linking algorithm and its VLSI architecture, capable of producing binary edge maps at the video rate, are presented in Ref. 13. It is based on local information and, as stated by the authors, has two major limitations. First, it does not guarantee producing closed contours; in fact in every experimental result presented in that paper, there are open contours. Second, the edge-point linking process is sensitive to user-defined parameters—threshold values.

In Ref. 7, a more elaborate edge-point linking approach, based only on local information, is proposed. Initially, an iterative edge-point thinning is applied. Thus, small gaps are filled and endpoints are easily recovered and labeled. Finally, endpoints are linked by minimizing a cost function based on local knowledge. The proposed cost function takes into account the Euclidean distance between the edge points to be linked (2D distance) and two reward coefficients—(1) if the points to be linked are both endpoints; and (2) if the direction associated with the points to be linked is opposite. The values of these two reward coefficients are experimentally determined. Since this technique is proposed for linking points, similarly to Ref. 13, it does not guarantee to produce closed contours.

Unlike previous approaches, algorithms based on global information need to study the whole edge-point distribution at the same time. In general, points are represented as nodes in a graph and the edge-point linking problem is solved by minimizing some global measure. For instance, Ref. 5 presents an edge-point linking scheme as a graph search problem. A similar scheme was previously introduced in Ref. 14. The methodology consists of associating to every edge point its corresponding gradient—magnitude and direction. Thus, the initial edge map becomes a graph with arcs between nodes, ideally unveiling the contour directions. A search algorithm, such as A^*, is used later for finding the best path among the edge points. Although the results presented in Ref. 5 are promising, the large number of image-dependent parameters, which have to be tuned by the user, discourages its use.

In turn, hybrid approaches usually enrich edge-point information with additional sources such as color, region descriptions, or surface geometry. An edge-point linking approach by merging spatial edge-point information and regions resulting from adaptive Bayesian color segmentation is presented in Ref. 6. Initially, a color-based segmentation split the original image into a set of contiguous regions. Later regions are merged and labeled. Finally, boundaries of the obtained regions constitute the linked edge map. On average, the execution time of this hybrid approach is on the order of a few minutes on a Sparc 20. A similar approach, in the sense that edge points are linked according to generated regions, is proposed by Ref. 10. This technique is intended for processing range images and is based on the assumption 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.

In Ref. 15, a fast technique that is free of user-defined parameters, and combines global and local information is presented. It is closely related to the previous approaches[5,14] in the sense that graph theory is also used to compute the best set of connections that interrelate edge points. In contrast to the previous ones, it is devised to generate closed contours from a range image's edge points, instead of classical intensity images. Initially, edge points are linked by minimizing a global cost function. At the same time, noisy data are easily removed by means of an efficient morphological filter. It does not have to go through the whole list of points contained in the input edge map, but only over those points labeled as endpoints—points linked once. In a second stage, closed contours are finally obtained by linking endpoints using a local cost function.

Other approaches, different from those presented above but 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 from the given image. These 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-known algorithms from different fields (deformable models, constrained clustering, and data ordering); see Ref. 16 for further details.

In the current work, we propose to adapt Ref. 15 in order to process intensity images. Range image processing techniques can be customized to work with 2D images considering intensity values as depth values. For instance, mesh modeling algorithms, developed for representing 3D images, have been extended to the 2D image field for different applications.[17–19] In the same way, we propose to adapt the contour closure technique presented in Ref. 15 in order to handle intensity images.

In general, adaptation of 3D image processing algorithms to the 2D image field has been carried out by representing the pixels of a given intensity image $I(u,v)$ in a 3D space $(u,v,I(u,v))$. However, in the particular case we are tackling in this paper, the adaptation of Ref. 15 to the intensity image domain requires not only the assumption of intensity values as depth values but also specific changes in the original technique in order to reach the best performance. In this context, (1) new cost functions that are proposed are specifically intended for handling intensity images; (2) the originally proposed noisy data filtering approach is modified in order to avoid problems with high textured regions.

The remainder of this paper is organized as follows. Section 2 briefly presents the technique proposed in Ref. 15 to highlight the required changes to handle intensity images. Experimental results with several images, as well as comparisons with a previous technique by using a publicly

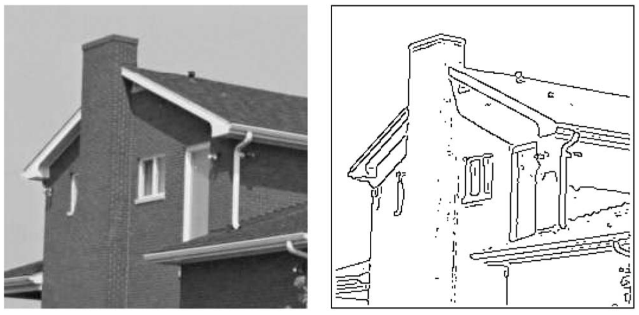

(a) (b)

Fig. 1 (a) Input intensity image, *I*. (b) Input edge map, *E*, computed by Canny.

available algorithm,[5] are presented in Section 3. Finally, conclusions and further improvements are given in Section 4.

2 Proposed Approach

Let I be a 2D array representing an intensity image with R rows and C columns, where each array element $I(r,c)$ ($0 \leq r < R$ and $0 \leq c < C$) is a value defined as $0 \leq I(r,c) \leq 255$. In order to use the approach proposed in Ref. 15, the first assumption is to consider intensity values as depth values; so every pixel in I becomes a point in 3D space: $(x,y,z)=(r,c,I(r,c))$. Let E be the edge map, corresponding to I, computed by some edge-point detector. Each element $E(r,c)$ is a Boolean indicating whether or not the corresponding image pixel is an edge point. Although in the current implementation edge maps were computed by using the Canny edge detector,[20] the proposed technique can be used with other edge detector algorithms (see Ref. 21 for a detailed comparison of different edge detector algorithms). In addition to the edge points computed by the chosen algorithm, edge points uniformly distributed through the first and last rows and columns are added. Added edge points are useful for detecting a region boundary when it touches an image's border; actually, the idea of imposing edge points through the image border has already been used[22] for segmenting moving objects in video sequences.

Assuming both arrays, I and E (see Fig. 1), are given as inputs, the proposed technique consists of two stages. The first stage links edge points by minimizing a global measure. Computed connections are filtered later by means of a morphological operator. Notice that by using a global approach, all edge points are linked, avoiding threshold definition problems usually related to local approaches (e.g., number of connections per edge points, maximum distance). The second stage works locally and is only focused on points labeled as endpoints. Both stages are further described ahead.

2.1 *Global Scheme: Graph-based Edge-Point Linking and Noisy Data Filtering*

At this stage a set of connected polylines linking all the input edge points, by minimizing the sum of linking costs, is computed. In order to compute that set of polylines, and following the proposal presented in Ref. 15, a partially connected graph Γ is used instead of working with a fully connected one.* Additionally, this helps to reduce the CPU time considerably. Since this partially connected graph should link nearest-neighbor edge points, a 2D Delaunay triangulation of edge points contained in E is computed.[23] The resulting 2D triangular mesh M is defined by a set $\{P,S\}$, where $P=\{E_{(i,j)},E_{(f,g)},\ldots,E_{(u,v)}\}$ corresponds to edge points in the aforementioned edge map E, and $S=\{S_1,S_2,\ldots,S_n\}$ corresponds to segments linking two edge points [e.g., $S_1=(E_{(i,j)},E_{(u,v)})$]. Additionally, every segment is associated with a linking cost value computed as indicated below.

In contrast to Ref. 7, where a linking cost only considering the Euclidean distance in the edge map is used, we propose a new cost function that also takes into account information from the intensity image space. Furthermore, we introduce the idea of using information related to those pixels traversed by the segments. Hence, two major advantages are reached. First, it prevents linking neighbors points in the edge map, which could belong to different regions in the intensity image. In other words, using only point positions in the edge map[7] could derive wrong results. Second, it avoids breaking/cutting real boundaries by using information related to traversed points. The proposed linking cost function is defined as follows:

$$\mathrm{LC}_{(i,j),(u,v)} = \|(i,j)-(u,v)\| * (1+\sigma_I/\mu_I), \qquad (1)$$

where $(1+\sigma_I/\mu_I)$ can be assumed as a weighting factor; σ_I represents the standard deviation of the intensity values associated with the set of pixels defining the segment S_k, which links points $E_{(i,j)}$ and $E_{(u,v)}$, both included. μ_I corresponds to the mean intensity value of that set of pixels. Points defining S_k are easily computed by using Bresenham's algorithm.[24] The main idea of this weighting factor is to avoid linking edge points with a similar intensity value but through an area with different intensity values. The variance of the intensity values of points defining S_k is scaled by μ_I in order to obtain a weighting value relative to the intensity of that area. Hence, the proposed linking cost takes into account more information than those presented in Refs. 7 and 15. At the same time, added CPU time for obtaining the set of points defining a straight line or computing their mean and standard deviation values could be disregarded.

Finally, the shortest path in Γ that links all the edge points is extracted by computing the minimum spanning tree (MST) of Γ. The MST of Γ is the acyclic subgraph of Γ that contains all the nodes and such that the sum of the costs associated with its segments is minimum. In the current implementation, Kruskal's algorithm has been used to compute the MST.[25] Notice that the MST of the Delaunay triangulated input edge points gives the same result as if it were computed over a fully connected graph of those points.

Figure 2 (top) shows the triangular mesh and its corresponding MST, computed from Fig. 1; the input edge map, Fig. 1(b), contains edge points computed by the edge detector[20] as well as those added over the first and last rows and columns. Notice that the resulting MST, Fig. 2(b), con-

*In order to avoid confusion, a graph will be referred to as *edge points* and *segments* instead of as *nodes* and *edges* as usually referred to in the literature.

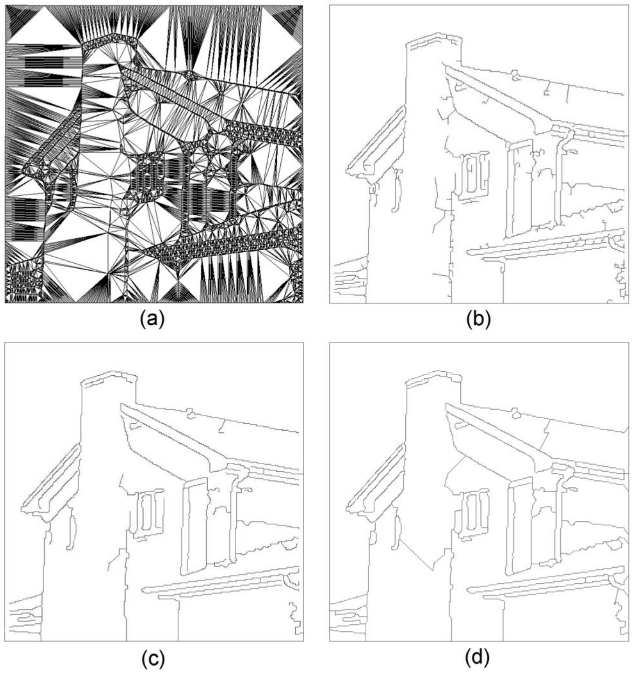


Fig. 2 (a) Triangular mesh of the edge points presented in Fig. 1(b). (b) Minimum spanning tree. (c) Filtered MST—opening algorithm. (d) Final linked edge-point representation.

tains *short branches* connected with the main path. A short branch is a branch of the tree defined by a few segments; this number of segments is not known beforehand and depends on the image to be processed. It is automatically computed as explained below. Short branches are generated by linking information redundancy and noisy data. So, before finishing this global stage, and taking advantage of edge-point connections, structured as single segments, a morphological filter is applied. The filter is a kind of opening algorithm and consists of performing iteratively erosions followed by the corresponding dilations; the latter is applied as many times as the erosion. In brief, the opening algorithm considers segments of the tree's branches as basic structuring elements (like pixels in an intensity image). Actually, the filtering algorithm is only performed over those segments linked from only one of their defining points—referred to as *end segments*. In the illustration presented in Fig. 3, since four iterations are performed during the erosion process (the stopping criterion is presented below), branches defined by less than four segments are removed (in this particular illustration two short branches are removed, one defined by two segments and the other by three segments).

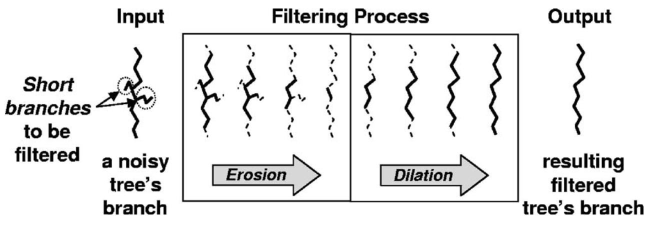
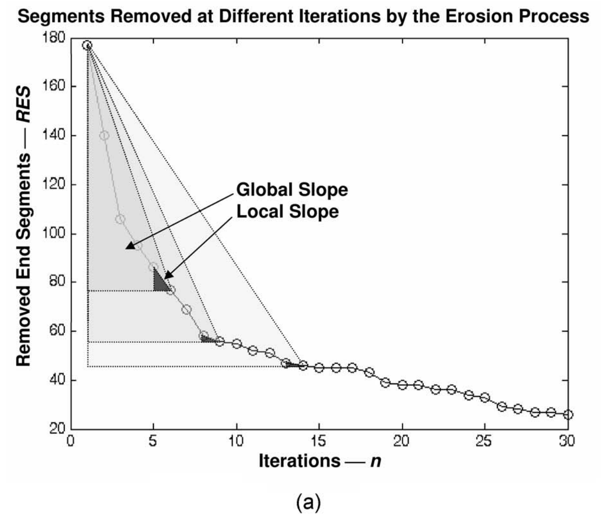


Fig. 3 Filtering process: opening algorithm.

As presented in Ref. 15, the erosion process stops when the number of end segments, removed at a given iteration, keeps constant during at least the n previous iterations. The criterion used to define this stopping condition is based on the assumption that after n consecutive iterations without changes in the number of removed segments, the erosion process has finished removing short branches. In other words, it has arrived to a stability point where segments belonging to the main path are being processed. Although initially we tried to use the same criterion for the intensity image processing, it was noticed that in some cases it does not work properly. Figure 6(d) illustrates the filtering result obtained with this stopping criterion—it can be appreciated that the filtering process removes not only segments from the MST, associated with noisy data, but also region boundaries useful for further processing (e.g., recovering the different contours defining the image regions). It is due to the fact that the texture of an intensity image generally produces a large number of segments [e.g., trees in Fig. 7(b)]; this effect was not noticed in Ref. 15 since surfaces defining a 3D object are usually larger than those small details appreciated in intensity images. Hence, in order to solve the aforementioned problem, a new criterion is introduced to stop the erosion process—avoiding fixing the number of iterations beforehand.

The proposed stopping criterion is based on the ratio between a global and a local slope, $\Upsilon = Sg/Sl$. $Sg = (\text{RES}(1) - \text{RES}(n))/n$ and $Sl = \text{RES}(n-1) - \text{RES}(n)$, where RES represents the number of removed end segments in the current (n), the first (1) and the previous $(n-1)$ iterations respectively (see illustrations of dark and light triangles presented in Fig. 4). The stopping threshold has been set experimentally as $\Upsilon \geqslant 6.3$. The use of the threshold, instead of defining beforehand the number of iterations, allows eroding end segments independently of the total amount of removed segments at a given iteration. Moreover, notice that although the shapes of the curves presented in Fig. 4 are similar in both plots, the number of segments processed at every iteration is completely different. In none of the processed images was the criterion presented in Ref. 15 reached while short branches were removed [see illustration in Fig. 6(d)]. Finally, after ending the erosion process, the same number of dilation iterations is carried out over the end segments left. The filtering process consists of four dilations applied after four erosions in the illustration presented in Fig. 3. Edge points defining those segments removed during the filtering are also removed from the input edge map E.

Before going to the next stage, a brief study of the filtering capability of the proposed stopping criterion to remove noisy short branches is presented. It consists of studying the robustness of the proposed criteria when the input edge map is corrupted with noisy data. Noisy data are added in the neighborhood of current edge points after removing some of the original edge points. The remove/add procedure works as follows. Initially, a set of r points is randomly selected and removed from the input edge map. The resulting edge map is now used for selecting a new set of random points, a. Every point contained in a is used as a center point of a 5×5 window, where a randomly selected point is introduced. Figure 5 presents three illustrations of the image used through the paper. The left column shows



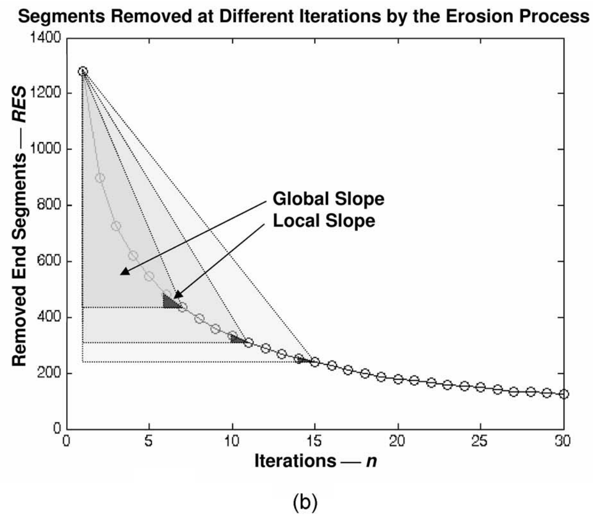


Fig. 4 End segments removed at different iterations by the erosion process. Dark and light triangles are used to compute the ratio between the global and local slopes, at each iteration. It is used as a threshold for stopping the erosion process. (a) House image erosion—Fig. 1. (b) Lenna image erosion—Fig. 6.

input edge maps resulting after removing/adding data. In these examples r and a have been set to (119, 154), (199, 220), and (279, 273), respectively, which correspond to about (3%, 5%), (5%, 7%), and (7%, 9%) of the total amount of points contained in the input edge map at that iteration. The middle column depicts the corresponding MSTs, while the right column presents the filtered MSTs obtained by using the proposed stopping criterion. Notice that a tradeoff between removing noisy segments and image features is reached by using the experimentally defined stopping threshold.

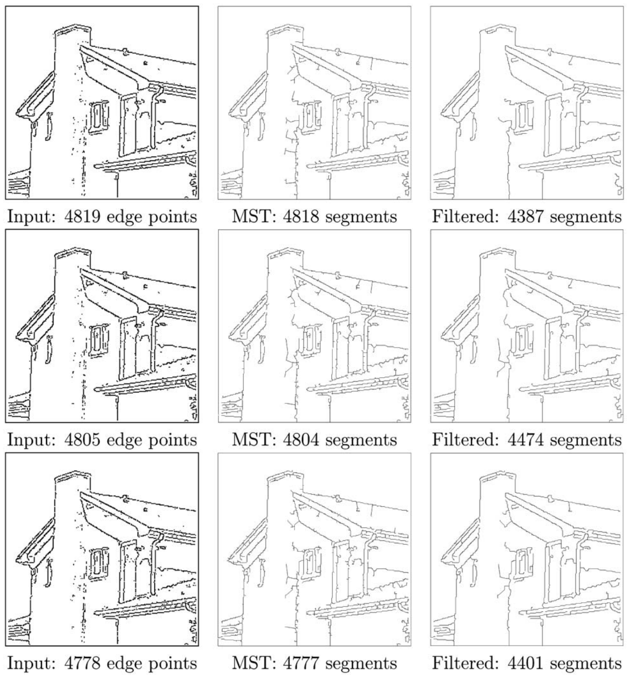


Fig. 5 (left) Input edge maps corrupted with noisy data. (middle) Resulting MSTs. (right) Filtered MSTs.

2.2 *Local Scheme: Cost-based Closure*

The outcome of the previous stage is a single tree, whose branches go through almost all the input edge points (some edge points were removed from the edge map during the last filtering stage). Figure 2(c) presents the result obtained after filtering the MST of Fig. 2(b). This representation connects edge points without defining closed contours, since it is a tree—the MST. Therefore, the objective at this last stage focuses on closing open contours.

Open contours are characterized by edge points linked once—*endpoints*. Since the previous filtering stage was carried out over end segments, endpoints are easily identified; there is no need to go through the whole list of edge points to find those only linked once. Moreover, it is not necessary to find a proper definition of endpoint, such as those proposed in Ref. 26 or Ref. 27, where several variants are introduced, since, as mentioned above, endpoints are only linked once. For every endpoint a list of candidate points from the filtered edge map is extracted. Notice that the list of candidate points is directly obtained from the filtered edge map; no information about the current connections is used; therefore, endpoints and internal points are equally considered—an internal point is a non-endpoint contained in the MST. Furthermore, the proposed closure function does not include reward factors (e.g., Ref. 7) since it could affect the correct connections of particular topologies (e.g., *tee-junction*). Finally, the point with the minimum closure cost is chosen to link with the given endpoint, thus closing the open contour. These stages are detailed below.

Given an endpoint $E(i,j)$, which belongs to the filtered MST, its set of candidate edge points is selected by means of an iterative process over a dynamic window, DW, centered at that point—$DW_{(i\pm m,j\pm n)}$, where $m=\{1,\ldots,t\}$; $n=\{1,\ldots,t\}$; $t=s+\tau$; and $\{(s<m<t)\vee(s<n<t)\}$. During the first iteration s is set to zero. Then after each iteration it is increased by τ. The threshold τ depends on the density of edge points in the given edge map; similarly to Ref. 15, in the current implementation τ was set to four.

After extracting the set of candidate points from the current iteration, a closure cost, CC, is computed. It represents the cost of connecting each one of those candidates with the given endpoint $E(i,j)$. It is computed according to the following expression:

$$CC_{(i,j),(u,v)} = \frac{\mathrm{LC}_{(i,j),(u,v)}}{\text{Path Length}_{(i,j),(u,v)}}, \tag{2}$$

where $\mathrm{LC}_{(i,j),(u,v)}$ is the linking cost defined in Eq. (1), which takes into account the distance between the points to be linked as well as the mean and standard deviation of the intensity values of those pixels defining the straight line between (i,j) and (u,v); while $\mathrm{PathLength}_{(i,j),(u,v)}$ measures the length of the path—the number of segments—linking those two points. In case no candidate points were extracted from the current window or the $\mathrm{PathLength}_{(i,j),(u,v)}$ values from those candidates to the given endpoint were equal to or smaller than t, the size of DW is increased by τ, as well as 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 the lowest closure cost is chosen to be linked with the endpoint $E(i,j)$.

The closure cost function, Eq. (2), adapted from the one proposed in Ref. 15, merges intensity image information from both points to be linked and traversed points (numerator) together with topological information related to the boundary that is being generated (denominator).

3 Experimental Results and Comparisons

The proposed technique has been tested with different images. As mentioned, in all cases edge maps were computed using the Canny edge detector.[20] Additionally, a set of edge points uniformly distributed over the image border (first and last rows and columns) was added. The CPU times to compute the different stages have been measured on a 3.2-GHz Pentium IV PC with a non-optimized C code.

The illustrations used throughout the paper correspond to an intensity image of 256×256 pixels [Fig. 1(a)] and an edge map defined by 4784 points [Fig. 1(b)]; its MST contains 4783 segments and was computed in 0.57 s [Fig. 2(b)]. The opening algorithm filters 234 segments from the computed MST, giving rise to a representation with 4549 segments in 0.01 s [Fig. 2(c)]. The 234 removed segments correspond to those linked with noisy data or redundant edge points. Finally, 74 open contours were closed in 0.11 s. This final representation contains 4623 segments, Fig. 2(d).

Other images were processed with the proposed approach. Figure 6(a) presents the input edge map corresponding to an image of 512×512 pixels. Intermediate results, such as the Delaunay triangulation of input edge points and its MST, are also presented in Fig. 6. The final closed-contour representation obtained after filtering the MST with the proposed criterion is presented in Fig. 6(f); it contains 17,900 segments and was computed in 27.41 s. Look at those gaps on the shoulder, forehead, and top of the hat that are successfully closed in the final representation.

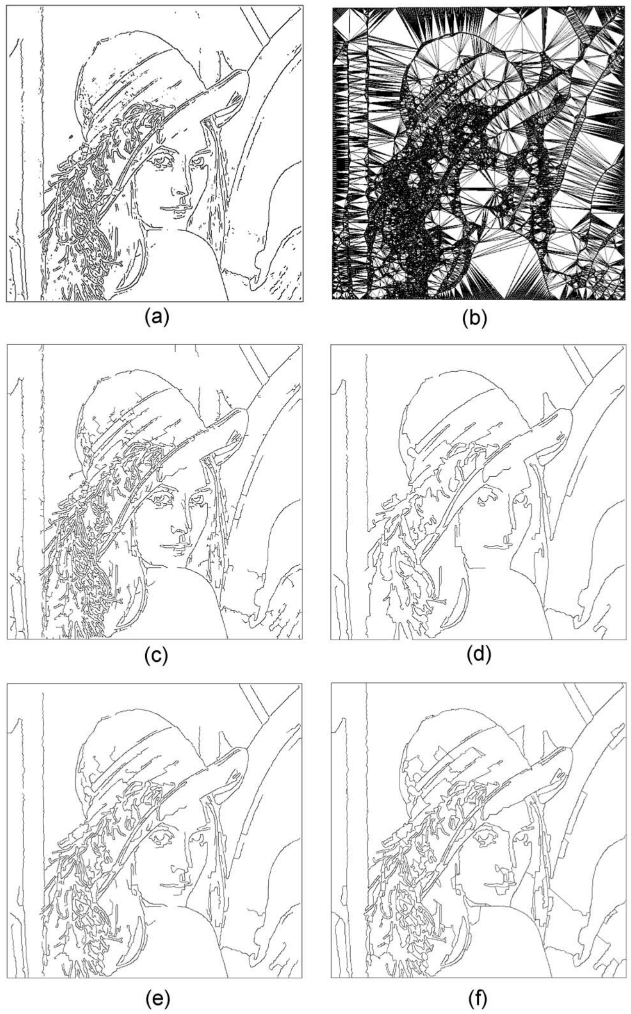


Fig. 6 (a) Input edge points, 21,393 points. (b) Triangular mesh. (c) Minimum spanning tree, 21,392 segments. (d) Filtered MST obtained with the previous proposal,[15] 14,059 segments. (e) Filtered MST obtained with the new proposal, 17,673 segments. (f) Final closed contour representation (after filtering the MST with the new proposal and closing open boundaries), 17,900 segments.

Figures 7(a) and 7(b) show edge maps corresponding to intensity images of 256×256 pixels (girl) and 512×512 pixels (car). The results from the global stage are presented in Figs. 7(c) and 7(d)—filtered MST. Final results are given in Figs. 7(e) and 7(f). Information regarding CPU time for the different examples is presented in Table 1. In all examples, about 85% of the time is spent on the triangular mesh and MST generation. There is room for improvement since a non-optimized C code is used.

Since the proposed technique does not use as prior knowledge the number of objects contained in the scene, and high-level post-processing is not applied (e.g., color-/texture-based region merging), and no constraint about the maximum length of gaps is imposed,[2] notice that there is no way to avoid single non-overlapped objects being connected in the final solution. In other words, the proposed technique is only intended for finding the best way to link all the edge points; isolated objects will be connected to some other object in the scene or to the image boundary. Therefore, closed contours do not necessarily correspond to

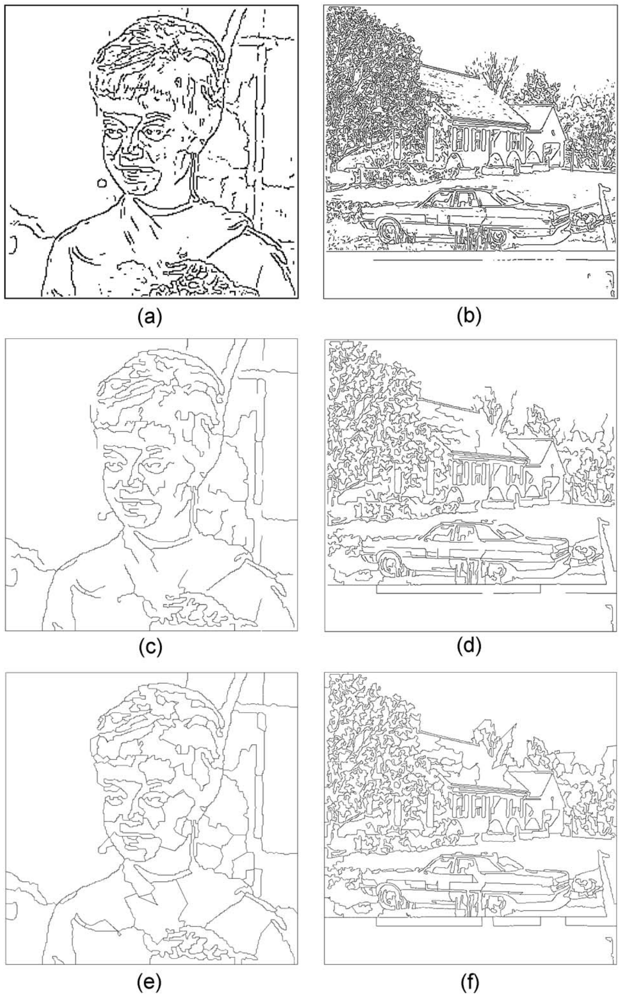


Fig. 7 (a) and (b) Input edge points (6,387 and 34,827 points). (c) and (d) Filtered MST (5,568 and 24,106 segments). (e) and (f) Final closed-contour representation (5,724 and 24,457 segments).

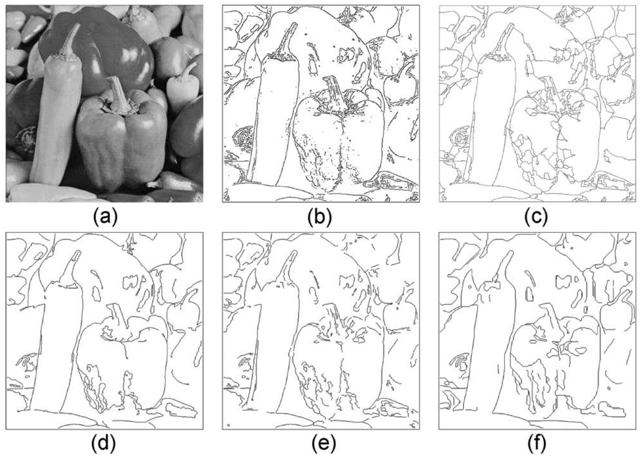


Fig. 8 Comparison between the proposed technique and the one presented in Ref. 5. (a) Input intensity image. (b) Edge map computed by Canny. (c) Result obtained with the proposed technique. (bottom) Edges computed and linked by Ref. 5, by using (d) the default set of parameters; (e) the default set of parameters and setting $l=1$; (f) the default set of parameters and setting $s=2$.

boundaries between two regions; they correspond to a representation where every single edge point is linked with a minimum total cost, ideally unveiling the different regions of the image.

The proposed approach has been compared with a publicly available algorithm based on a multiresolution, sequential edge-linking algorithm, M-SEL.[5] Figures 8(a) and 9(a) present intensity images used to compare both techniques. They are defined by 512×512 pixels and 512×480 pixels, respectively. Figures 8(b) and 9(b) present their corresponding edge maps computed by the Canny algorithm and used as input by the proposed technique. Experimental results obtained with the proposed technique are presented in Figs. 8(c) and 9(c), respectively. Since Ref. 5 depends on a large number of user-tuned parameters—nine parameters—three different representations were computed to compare with the proposed technique. In the first case the nine parameters were set with the values given by default with the downloaded code[5] [Figs. 8(d) and 9(d)]. In the second and third experiments, only one of the parameters was changed while the others were set with the default values. Figures 8(e) and 9(e) present results obtained by

Table 1 CPU time (s).

	Global Scheme		Local Scheme	
	Triangular Mesh and MST Generation	Filtering	Contour Closure	Total Time
House	0.57	0.01	0.11	0.69
Lenna	23.76	0.82	2.83	27.41
Car	80.51	2.47	5.61	88.59
Girl	0.99	0.06	0.28	1.33

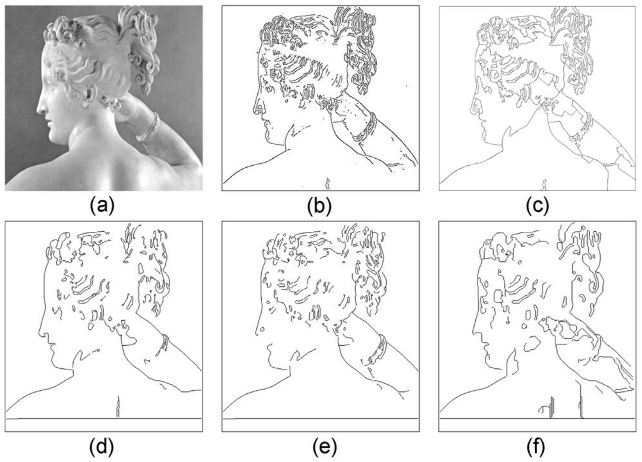


Fig. 9 Comparison between the proposed technique and the one presented in Ref. 5. (a) Input intensity image. (b) Edge map computed by Canny. (c) Result obtained with the proposed technique. (bottom) Representations computed with Ref. 5, by using (d) the default set of parameters; (e) the default set of parameters and setting $l=1$; (f) the default set of parameters and setting $s=2$.

Table 2 Comparison between M-SEL algorithm[5] and the proposed technique (CPU time in s).

	Proposed Technique	M-SEL Algorithm (default parameters)	M-SEL Algorithm (l=1, standard SEL)	M-SEL Algorithm (s=2)
Peppers	31.2	7.9	13.5	12.6
Paolina	18.5	5.3	8.8	9.7

setting $l=1$. Finally, Figs. 8(f) and 9(f) show linked edge points computed by setting $s=2$ and the other parameters with the default values. No careful tuning of the nine parameters has been done, because no hints are given for deciding their values in order to obtain optimal performance. Moreover, the relationship among them makes the tuning process tedious.

In most of the cases, Ref. 5 generates closed loops of edges instead of finding the right path among edge points. Comparative CPU times are given in Table 2. Notice that although the CPU time required by the proposed technique is on average more than twice the M-SEL time, the usefulness of the obtained representations is more than that. For instance, in none of the representations presented in Fig. 9 (bottom) can the different regions contained in the given image—background, hair, face, arm, back—be discerned; in the three cases, all of them are connected, defining a single region that includes almost all the image's pixels—only a few small regions defined by closed loops in the head are generated. On the contrary, the result obtained with the proposed technique, Fig. 9(c), can be more useful than those obtained with M-SEL for a high-level scene understanding algorithm since the given image is segmented in different regions. This behavior can also be appreciated in the results presented in Fig. 8 (bottom), where different regions are merged together.

Finally, although it was not possible to find a publicly available implementation of Ref. 7, a visual and qualitative comparison can be done since almost the same set of images presented in Ref. 7 were processed with the proposed approach. Notice, as highlighted above, that the proposed technique can successfully close gaps such as those on the top of the hat, forehead, or vertical edges on the left of Fig. 6(f), while they remain open in Ref. 7, in addition to other open gaps or minor artifacts left in Ref. 7.

4 Conclusions and Further Improvements

This paper presents the use of global and local schemes for computing closed contours from edge points of intensity images. The global stage is based on graph theory while the local one relies on values computed by a local cost function. Noisy and redundant edge points are removed by means of an efficient morphological operator. Although this approach has been initially proposed to handle range images, the different tests performed with the new cost functions, together with the adaptation of the noisy data filtering, proved that it is also useful for processing intensity images. Experimental results and comparisons with previous techniques show that the proposed technique is able to handle different kind of images; it does not require the prior knowledge of the scene, nor parameters that have to be carefully tuned.

Most of the effort on further work will be focused on improving triangular mesh and MST generation, which are the most expensive parts of the proposed approach. The use of optimized code, as well as the joint generation of triangular mesh and MST, could speed up the whole process. Additionally, further work will include the study of more elaborate cost functions. Finally, the use of smooth curve fitting for representing currently computed closed contours (i.e., piecewise linear interpolations) will also be considered.

Acknowledgments

This work has been partially supported by the Spanish Ministry of Education and Science under project TRA2004–06702/AUT. The first author (A.D.S.) was supported by The Ramón y Cajal Program. The second author (B.X.V.) was partially supported by the ESPOL under the VLIR project, Component 8.

References

1. W. Grimson, *From Images to Surfaces*, Cambridge, MA, MIT Press (1981).
2. D. Jacobs, "Robust and efficient detection of salient convex groups," *IEEE Trans. Pattern Anal. Mach. Intell.* **18**(1), 23–37 (1996).
3. E. Saund, "Finding perceptually closed path in sketches and drawings," *IEEE Trans. Pattern Anal. Mach. Intell.* **25**(4), 475–491 (2003).
4. S. Wang, T. Kubota, J. Sisking, and J. Wang, "Salient closed boundary extraction with ratio contour," *IEEE Trans. Pattern Anal. Mach. Intell.* **27**(4), 546–561 (2005).
5. A. Farag and E. Delp, "Edge linking by sequential search," *Pattern Recogn.* **28**(5), 611–633 (1995).
6. E. Saber, A. Tekalp, and G. Bozdagi, "Fusion of color and edge information for improved segmentation and edge linking," *Image Vis. Comput.* **15**, 769-780 (1997).
7. O. Ghita and P. Whelan, "Computational approach for edge linking," *J. Electron. Imaging* **11**(4), 479–485 (2002).
8. S. Mahamud, L. Williams, K. Thornber, and K. Xu, "Segmentation of multiple salient closed contours from real images," *IEEE Trans. Pattern Anal. Mach. Intell.* **25**(4), 433–444 (2003).
9. S. Zhu and A. Yuille, "Region competition: Unifying snakes, region growing, and Bayes/MDL for multiband image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.* **18**(9), 884–900 (1996).
10. X. Jiang, "An adaptive contour closure algorithm and its experimental evaluation," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(11), 1252–1265 (2000).
11. T. Zhang and C. Suen, "A fast parallel algorithm for thinning digital patterns," *Commun. ACM* **27**(3), 236–239 (1984).
12. W. Snyder, R. Groshong, M. Hsiao, K. Boone, and T. Hudacko, "Closing gaps in edges and surfaces," *Image Vis. Comput.* **10**(8), 523–531 (1992).
13. A. Hajjar and T. Chen, "A VLSI architecture for real-time edge linking," *IEEE Trans. Pattern Anal. Mach. Intell.* **21**(1), 89–94 (1999).
14. D. Ballard and C. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ (1982).
15. A. Sappa, "Unsupervised contour closure algorithm for range image edge-based segmentation," *IEEE Trans. Image Process.* **15**(2), 377–384 (2006).
16. A. Abrantes and J. Marques, "A class of constrained clustering algorithms for object boundary extraction," *IEEE Trans. Image Process.* **5**(11), 1507–1521 (1996).
17. M. Garcia, B. Vintimilla, and A. Sappa, "Approximation and processing of intensity images with discontinuity-preserving adaptive triangular meshes," *Proc. ECCV 2000*, 1842, 844–855 (2000).
18. L. Hermes and J. Buhmann, "A minimum entropy approach to adaptive image polygonization," *IEEE Trans. Image Process.* **12**(10), 1243–1258 (2003).
19. Y. Yang, M. Wernick, and J. Brankov, "A fast approach for accurate content-adaptive mesh generation," *IEEE Trans. Image Process.* **12**(8), 866–881 (2003).
20. J. Canny, "Computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.* **8**(6), 679–698 (1986).

21. H. Heath, S. Sarkar, T. Sanocki, and K. Bowyer, "Comparison of edge detectors: A methodology and initial study," *Proc. IEEE Comp. Vis. Patt. Recog.*, pp. 143–148 (1996).
22. C. Kim and J. Hwang, "Fast and automatic video object segmentation and tracking for content-based applications," *IEEE Trans. Circuits Syst. Video Technol.* **12**(2), 1122–1129 (2002).
23. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, New York (2000).
24. T. Pavlidis, *Algorithms for Graphics and Image Processing*, Marcel Dekker, New York (1982).
25. K. Rosen, *Discrete Mathematics and Its Applications*, 2nd ed., McGraw-Hill, New York (1990).
26. J. Gómez, N. Ilhami, P. Luque Escamilla, J. Martínez Aroza, and R. Román Roldán, "Improved entropic edge-detection," *Proc. IEEE Int. Conf. Image Anal. Proc.*, pp. 180–184 (1999).
27. L. Lam, S. Lee, and C. Suen, "Thinning methodologies—a comprehensive survey," *IEEE Trans. Pattern Anal. Mach. Intell.* **14**(9), 869–885 (1992).

Angel D. Sappa received his electromechanical engineering degree in 1995 from National University of La Pampa, General Pico-La Pampa, Argentina, and his PhD degree in industrial engineering in 1999 from Polytechnic University of Catalonia, Barcelona, Spain. From 1999 to 2002, he undertook a post-doctorate research position at LAAS-CNRS, Toulouse, France, and at Z+F UK Ltd., Manchester, UK. 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, Spain, as a Ramón y Cajal Research Fellow. His research interests are focused on range image analysis, 3D modeling, and model-based segmentation.

Boris X. Vintimilla received his mechanical engineering degree in 1995 from the Escuela Superior Politécnica del Litoral, Guayaquil, Ecuador, and his PhD degree in industrial engineering in 2000 from Polytechnic University of Catalonia, Barcelona, Spain. In 2001, he joined the Department of Electrical and Computer Science Engineering of the Escuela Superior Politécnica del Litoral, Guayaquil, Ecuador, as an associate professor and has been the head of the Vision and Robotics Center since August 2005. His research interests include image processing, 3D modeling, and mobile robotics.