Edge Point Linking by Means of Global and Local Schemes

Angel D. Sappa[1] and Boris X. Vintimilla[2]

[1] Computer Vision Center
Edifici O Campus UAB
08193 Bellaterra, Barcelona, Spain
angel.sappa@cvc.uab.es

[2] Vision and Robotics Center
Dept. of Electrical and Computer Science Engineering
Escuela Superior Politecnica del Litoral
Campus Gustavo Galindo, Prosperina, Km 30.5
09015863 Guayaquil, Ecuador
boris.vintimilla@espol.edu.ec

Summary. This chapter presents an efficient technique for linking edge points in order to generate a closed contour representation. The original intensity image, as well as its corresponding edge map, are assumed to be given as input to the algorithm (i.e., an edge map is previously computed by some of the classical edge detector algorithms). The proposed technique consists of two stages. The first stage computes an initial representation by connecting edge points according to a global measure. It relies on the use of graph theory. Spurious edge points are removed by a morphological filter. The second stage finally generates closed contours, linking unconnected edges, by using a local cost function. Experimental results with different intensity images are presented.[3]

1 Introduction

Edge detection is the first and most important stage of human visual process as presented in [6]. During last 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. Unfortunately, computed edge maps usually contain gaps as well as false edge points generated by

[3] This work has been partially supported by the Spanish Ministry of Education and Science under project TRA2004-06702/AUT. The first author was supported by The Ramón y Cajal Program. The second author was partially supported by the ESPOL under the VLIR project, Component 8.

noisy data. Moreover, 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 in 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.

Different 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: *a*) local approaches, which work over every single edge point, and *b*) global approaches, which work over the whole edge map at the same time. Alternatively, algorithms that combine both approaches or use not only edge map information but also enclosed information (e.g., color) can be found (e.g., [10], [15]). In general, most of the techniques based on local information rely on morphological operators applied over edge points. Former works on edge linking by using morphological operators compute closed boundaries by thinning current edge points [14]. However, common problems of thinning algorithms are that in general they distort the shape of the objects, as well as big gaps can not be properly closed. In order to avoid these problems [12] introduces the use of morphological operators together with chamfer maps. Experimental results with simple synthetic-like images with closely spaced unconnected edges, which do not contain spurious neither noisy edge points, are presented.

A real-time edge-linking algorithm and its VLSI architecture, capable of producing binary edge maps at the video rate, is presented in [7]. It is based on local information and, as stated by the authors, has two major limitations. Firstly, it does not guarantee to produce closed contours, actually in every experimental results presented in that paper there are open contours. Secondly, edge-linking process is sensitive to user defined parameters—threshold values.

In [5], a more elaborated edge linking approach, based only on local information, is proposed. Initially, an iterative edge thinning is applied. Thus, small gaps are filled and endpoints are easily recovered and labelled. Finally, endpoints are linked by minimizing a cost function based on a 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—*a)* if the points to be linked are both endpoints; and *b)* if the direction associated to 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 [7], it does not guarantee to produce closed contours.

Differently to 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 linking problem is solved by minimizing some global measure. For instance, [3] presents an edge linking scheme as a graph search problem. A similar scheme was previously introduced in [1]. The methodology consists in 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 later on used for finding the best path among the edge points. Although results presented in [3] are promising, the excessive CPU time together with the large number of image dependent parameters, which have to be tuned by the user, discourage its use.

In [11] a fast and free of user-defined parameters technique, which combines global and local information, is presented. It is close related to the previous approaches ([3], [1]), in the sense that graph theory is also used to compute the best set of connections that interrelate edge points. Differently to the previous ones, it is devised to generate closed contours from 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 labelled as endpoints—points linked once. In a second stage, closed contours are finally obtained by linking endpoints using a local cost function.

In the current work, we propose to adapt [11] 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 modelling algorithms, developed for representing 3D images, have been extended to the 2D image field for different applications (e.g., [4], [8], [13]). In the same way, we propose to adapt the contour closure technique presented in [11] in order to handle intensity images.

The remainder of this chapter is organized as follow. Section 2 briefly introduces the technique proposed in [11] together with the required changes to face up intensity images. Experimental results with several images are presented in Section 3. Finally, conclusions and further improvements are given in Section 4.

2 Proposed Technique

Let I be a 2D array representing an intensity image with R rows and C columns, where each array element $I(r,c)$ is a value defined as $0 \leq I(r,c) \leq 255$. In order to have a direct application of the approach proposed in [11], intensity values are considered as depth values; so every pixel in $I(r,c)$ becomes a point in 3D space: $(x,y,z) = (r,c,I(r,c))$. Let E be the corresponding edge map computed by an edge point detector algorithm. Each element of $E(r,c)$ is a boolean indicating whether the corresponding image pixel is an edge point or not. In the current implementation edge maps were computed by using Canny edge detector [2]. Additionally, edge points uniformly distributed through the first and last rows and columns were added. Added edge points are useful for detecting a region boundary when it touch an image's border; actually, the

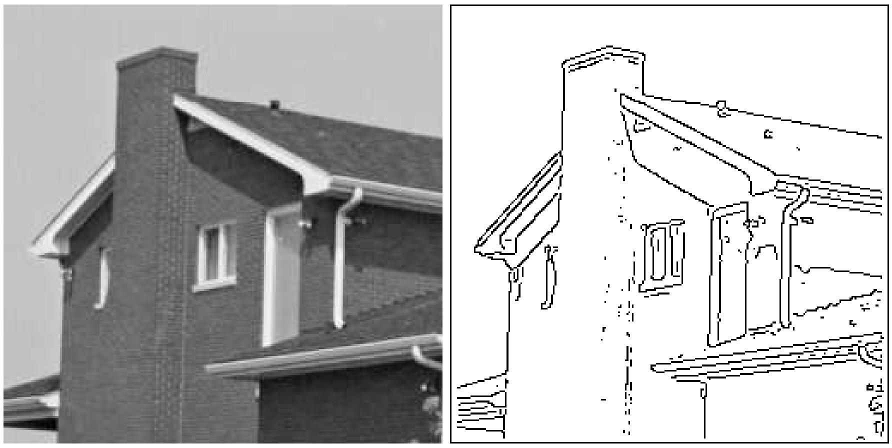

Fig. 1. (*left*) Input intensity image, I. (*right*) Input edge map, E, computed by Canny.

idea of imposing edge points through the image border has already been used in [9].

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 later on filtered by means of a morphological operator. The second stage works locally and is only focussed on points labelled as endpoints. Both stages are further described below.

2.1 Global Scheme: Graph Based Linking

At this stage a single polyline that links all the input edge points, by minimizing the sum of linking costs, is computed. On the contrary to [5], where a linking cost considering the Euclidean distance in the edge map is used (distance in a 2D space), we propose to use also the Euclidean distance but in the 3D space. Neighbor points in the edge map could belong to different regions in the intensity image. In other words, using only point positions in the edge map could drive to wrong results. Therefore, linking cost between edge points $(E_{(i,j)}, E_{(u,v)})$ is defined as:

$$LC_{(i,j),(u,v)} = \|(i, j, I(i, j)) - (u, v, I(u, v))\| \tag{1}$$

In order to speed up further processing, a partially connected graph Γ is computed, instead of working with a fully connected one. Since this partially connected graph should link nearest neighbor edge points, a 2D Delaunay triangulation of the edge map's points is computed. Additionally, every edge is associated with a cost value computed as indicated above, $LC_{(i,j),(u,v)}$.

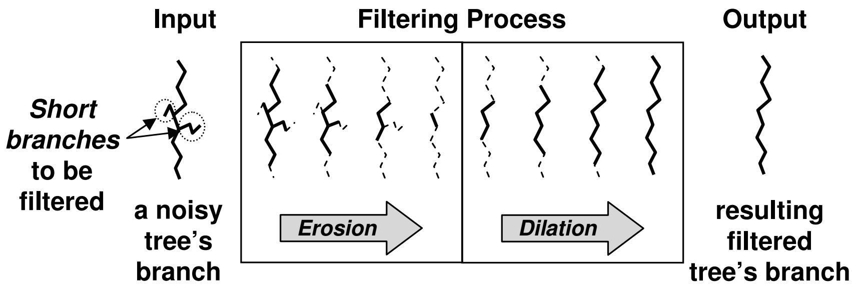

Fig. 2. Filtering process: *opening algorithm.*

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 edges is minimum. Notice that the MST of the Delaunay triangulated input edge points gives the same result than if it were computed over a fully connected graph of those points.

Fig. 3(*top*) shows the triangular mesh and its corresponding MST, computed from Fig. 1; input edge map, Fig. 1(*right*), contains edge points computed by the edge detector [2], as well as edge points added over the first and last rows and columns. As can be appreciated in Fig. 3($top - right$), the resulting MST contains short branches—branches defined by a few edges—, connected with the main path. They belong to information redundancy and noisy data. So, before finishing this global approach stage, and taking advantage of edge point connections structured as a single polyline, a morphological filter is applied. The filter is a kind of *opening algorithm* and consists in performing iteratively erosions followed by the corresponding dilations; the latter applied as many times as the erosion. In brief, the opening algorithm considers segments of the polyline as basic processing elements (like pixels in an intensity image). From the polyline computed by the MST, those segments linked from only one of their defining points—referred as *end segments*—are removed during the erosion stage. After ending the erosion process, dilations are carried out over end segments left. Fig. 2 shows and illustration of this filtering process; in this case it consists of four dilations applied after four erosions. More details about the filtering process can be found in [11].

2.2 Local Scheme: Cost Based Closure

The outcome of the previous stage is a single polyline going through almost all edge points (some edge points were removed during the last filtering stage). Fig. 3($bottom - left$) presents the result obtained after filtering the MST of Fig. 3($top - right$). Notice that although this polyline connects edge points it does not define closed contours—recall that the MST is an acyclic subgraph

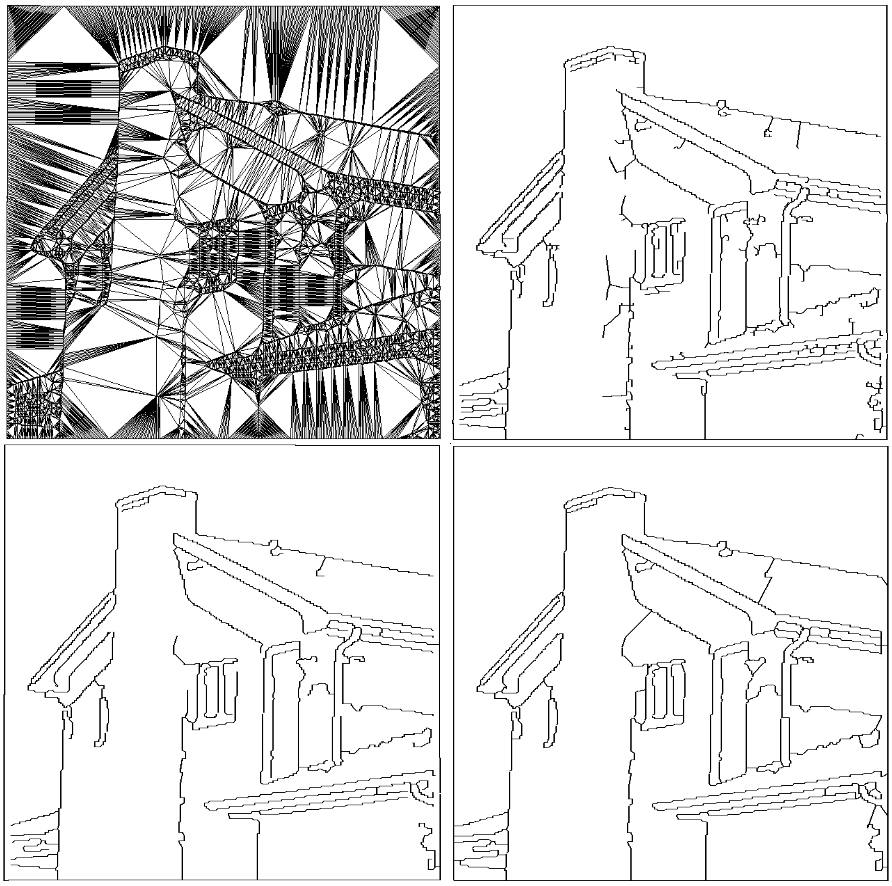

Fig. 3. $(top - left)$ Triangular mesh of the edge points presented in Fig. 1$(right)$. $(top - right)$ Minimum spanning tree. $(bottom - left)$ Filtered MST—opening algorithm. $(bottom - right)$ Final linked edge point representation.

so that it does not contain any closed contours. 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. For every endpoint a list of candidate points from the edge map E is extracted. Finally, the point with a minimum closure cost is chosen to close the given endpoint. These stages are detailed below.

Given an end point $E(i, j)$, 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; 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:

$$CC_{(i,j),(u,v)} = \frac{LC_{(i,j),(u,v)}}{PathLength_{(i,j),(u,v)}} \tag{2}$$

$LC_{(i,j),(u,v)}$ is the linking cost defined in (1), which represents the 3D distance between the points to be linked; while $PathLength_{(i,j),(u,v)}$ 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 $PathLength_{(i,j),(u,v)}$ values from those candidates to the given endpoint were equal or smaller than t, the size of DW 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 closure cost is chosen to be linked with the endpoint $E(i,j)$.

3 Experimental Results

The proposed technique has been tested with different intensity images. As mentioned above, in all the cases edge maps were computed by using Canny edge detector [2]. Additionally, a set of edge points uniformly distributed over the image border (first and last rows and columns) was added. The CPU time to compute the different stages have been measured on a 1.86 GHz Pentium M PC with a non-optimized C code.

The illustrations used through the chapter correspond to an intensity image of 256×256 pixels (Fig. 1($left$)) and an edge map defined by 4784 points (Fig. 1($right$)); its MST contains 4783 edges and was computed in 0.56 sec (Fig. 3($top - right$)). The opening algorithm filters 378 edges from the computed MST giving rise to a representation with 4405 edges in 0.03 sec (Fig. 3($bottom - left$)). The 378 removed edges correspond to those ones linked with noisy data or redundant edge points. Finally, 52 open contours are closed in 0.05 sec. This final representation contains 4457 edges, Fig. 3($bottom - right$)

Other images were processed with the proposed approach. Fig. 4($top - left$) presents the input edge map, 21393 points, corresponding to an image of 512×512 pixels. Intermediate results, such as Delaunay triangulation of input edge points and its MST, are also presented in Fig. 4. The MST is defined by 21392 edges. The final closed contour representation is presented in Fig. 4($bottom - right$); it contains 18517 edges and was computed in 28.4 sec. Have

Fig. 4. ($top - left$) Input edge points, 21393 points. ($top - right$) Triangular mesh. ($bottom - left$) Minimum spanning tree, 21392 edges. ($bottom - right$) Final closed contour representation (after filtering the MST and closing open boundaries), 18517 edges.

a look at those gaps on the shoulder and top of the hat that are successfully closed in the final representation.

Finally, Fig. 5(*top*) shows edge maps defined by 6387 and 34827 points respectively; the corresponding intensity images are defined by 256×256 pixels (girl) and 512×512 pixels (car). The results from the global approach stage are presented in Fig. 5(*middle*)—filtered MST. Final results are given in Fig. 5(*bottom*); they are defined by 5052 and 27257 edges respectively. Information regarding CPU time for the different examples are presented in Tab. 1. As can be appreciated in all the examples about 85% of the time is spent by the triangular mesh and MST generation. Since a non-optimized C code is used, it is supposed that there is a room for improvement.

Fig. 5. (*top*) Input edge points (6387 and 34827 points). (*middle*) Filtered MST (4964 and 26689 edges). (*bottom*) Final closed contour representation (5052 and 27257 edges).

Table 1. CPU time (sec)

	Global Scheme		Local Scheme	Total Time
	Triangular Mesh and MST Generation	Filtering	Contour Closure	
House	0.56	0.03	0.05	0.65
Lenna	23.9	0.76	3.74	28.41
Car	80.43	2.35	11.67	94.46
Girl	0.98	0.07	0.13	1.19

4 Conclusions and Further Improvements

This chapter 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, experimental results proved that it is also useful for processing intensity images.

Further work will be focused on improving MST generation, for instance by generating it at the same time that the triangular mesh. Additionally, the development of new linking cost and closure cost functions, specifically designed for handling intensity images, will be considered. It is supposed that cost functions that take into account information such as intensity of pixels crossed by the graph edges could improve the final results. Finally, comparisons with other approaches will be done.

References

1. D. Ballard and C. Brown. *Computer Vision*. Prentice-Hall, Inc., 1982.
2. J. Canny. Computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.
3. A. Farag and E. Delp. Edge linking by sequential search. *Pattern Recognition*, 28(5):611–633, May 1995.
4. M.A. Garcia, B. Vintimilla, and A. Sappa. Approximation and processing of intensity images with discontinuity-preserving adaptive triangular meshes. In *Proc. ECCV 2000, D. Vernon, Ed. New York: Springer, 2000, vol. 1842, LNCS, Dublin, Ireland*, pages 844–855, June/July 2000.
5. O. Ghita and P. Whelan. Computational approach for edge linking. *Journal of Electronic Imaging*, 11(4):479–485, October 2002.
6. W.E. Grimson. *From Images to Surfaces*. Cambridge, MA: MIT Press, 1981.
7. A. Hajjar and T. Chen. A VLSI architecture for real-time edge linking. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(1):89–94, January 1999.

8. L. Hermes and J. Buhmann. A minimum entropy approach to adaptive image polygonization. *IEEE Trans. on Image Processing*, 12(10):1243–1258, October 2003.
9. C. Kim and J. Hwang. Fast and automatic video object segmentation and tracking for content-based applications. *IEEE Trans. on Circuits and Systems for Video Technology*, 12(2):122–1129, February 2002.
10. E. Saber and A. Tekalp. Integration of color, edge, shape, and texture features for automatic region-based image annotation and retrieval. *Journal of Electronic Imaging*, 7(3):684–700, July 1998.
11. A.D. Sappa. Unsupervised contour closure algorithm for range image edge-based segmentation. *IEEE Trans. on Image Processing*, 15(2):377–384, February 2006.
12. W. Snyder, R. Groshong, M. Hsiao, K. Boone, and T. Hudacko. Closing gaps in edges and surfaces. *Image and Vision Computing*, 10(8):523–531, October 1992.
13. Y. Yang, M. Wernick, and J. Brankov. A fast approach for accurate content-adaptive mesh generation. *IEEE Trans. on Image Processing*, 12(8):866–881, August 2003.
14. T. Zhang and C. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, March 1984.
15. S. Zhu and A. Yuille. Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(9):884–900, 1996.