Extracting Plane Graphs from Images*

Émilie Samuel[1], Colin de la Higuera[2], and Jean-Christophe Janodet[1]

[1] Université de Lyon, F-42023, Saint-Étienne, France
CNRS, UMR5516, Laboratoire Hubert Curien, 42023, Saint-Étienne, France
Université de Saint-Étienne, Jean Monnet, F-42023, Saint-Étienne, France
{emilie.samuel,janodet}@univ-st-etienne.fr
[2] CNRS, UMR6241, LINA, 44322, Nantes, France
Université de Nantes, F-44322, Nantes, France
cdlh@univ-nantes.fr

Abstract. In order to use structural techniques from graph-based pattern recognition, a first necessary step consists in extracting a graph in an automatic way from an image. We propose to extract *plane graphs*, because of algorithmic properties these graphs have for isomorphism related problems. We also consider the problem of extracting semantically well-founded graphs as a compression issue: we get simple graphs from which can be rebuilt images similar to the initial image. The technique we introduce consists in segmenting the original image, extracting interest pixels on the segmented image, then converting these pixels into pointels, which in turn can be related by region-based triangulation. We show the feasibility and interest of this approach in a series of experiments.

Keywords: Plane graphs, images, interest pointels, segmentation, Delaunay triangulation.

1 Introduction

Representing images with graphs is an approach followed in pattern recognition in which it is hoped to benefit from the structural properties of the graphs, but also to be able to use the robustness associated with them [2,10]. A number of ideas have been analysed in order to use these graphs once built, involving edit distances [17] or graph matching [14]. Moreover, databases of synthetically generated graphs [19] or graphs related to pattern recognition benchmarks [16] have been created.

Many different models of graphs extracted from images have already been proposed. A first alternative consists in segmenting the image, in representing each region by a vertex and putting an edge between two vertices whenever the regions have a common border. Further arguments can be added to indicate the size of the region, the length of the border, and so forth. The obtained graphs

* This work was partially supported by an ANR grant BLANC 07-1-184534 (project SATTIC), and the IST Programme of the European Community, under the PASCAL 2 Network of Excellence, IST–2006-216886.

E.R. Hancock et al. (Eds.): SSPR & SPR 2010, LNCS 6218, pp. 233–243, 2010.

are called Region Adjacency Graphs [18] and have been improved, yielding dual graphs [11], ordered graphs [9], combinatorial maps [12]. A second option consists in extracting interest points (they will now be the vertices) and in relating them according to some neighbourhood relationship (the Delaunay triangulation may be a way to do this). Again, numerical attributes can be added to the representation, either attached to the vertices or to the edges.

Yet neither of the two aforementioned techniques is satisfying: typical drawbacks in the first case are that the structure of the graph actually has little to do with the topological nature of the image (i.e. the spatial disposition of depicted objects, their borders and connexity), in particular when the segmentation is rough. In the second case, not only do many of the so-called interest points not seem to be that interesting (requiring to extract many more points than necessary in order to be sure to have all the important ones), but also, since these points are actually pixels (with non null thickness), the construction is hindered by discrete geometry issues (e.g. what is the frontier between two regions?).

What should be the features of a good graph extracted from an image? On one hand, it should be robust: if the image is slightly distorted, the graph should not be deeply modified. It should also have good algorithmic properties: typically the central isomorphism problem, the related problems of sub-graph isomorphism, maximum common subgraph and graph edit distance should be solvable with polynomial algorithms. Furthermore what we may call the semantic properties of the image should be present in the graph: this might mean that the graph, when drawn, actually is a symbolic representation of this image; for instance, the boundaries between the regions could coincide with the edges of the graph. Finally we would like to be able to rebuild an image from the graph and for this new image to be a good compression of the initial image. In other terms, the *loss* due to the extraction process should be minimal.

In this paper we explore the possibility of combining the advantages of both the techniques presented above: on the one hand, segmentation introduces robustness; on the other hand, extraction of interesting pixels is a good idea and much less pixels are going to be needed from a segmented image. The method we suggest is thus as follows: (1) segment the image into regions, (2) extract s interest pixels, (3) add s' intersection pixels (s' being generally less than s), (4) compute the corresponding interest *pointels*, (5) relate them according to the boundaries of the regions and (6) triangulate the pointels of each region.

2 Definitions

A (digital) *image* I is an array of $n*m$ *pixels*. Each pixel p has a colour described by a 3 dimensional vector $c_I(p)$ in standard RBG colour system; that latter hypothesis corresponds to our experimental settings but any other colour system could be used. The coordinates of the pixels consist in pairs (i, j); their range goes from $(0, 0)$ to $(n-1, m-1)$. Each pixel is delimited from each of his four neighbours by a *linel*, and the four corners of the pixels are called *pointels*. Such pointels and linels are mathematical points and line respectively, with null

thickness. This enables us to associate the coordinates of the pointels to those of the pixels; their range goes from $(0,0)$ to (n,m).

We now introduce the main criterion used in this paper to measure the loss between an original image I_1 and an image I_2 obtained after a series of transformations. We assume that both I_1 and I_2 have the same size $(n * m)$. The *loss* is the L_1 distance between I_1 and I_2 divided by the number $n * m$ of pixels. Formally, we first define the distance d_p between any pixel p_1 in image I_1 and pixel p_2 in image I_2 as the normalised distance between corresponding colour vectors. That is, if $c_{I_1}(p_1) = (r_1, g_1, b_1)$ and $c_{I_2}(p_2) = (r_2, g_2, b_2)$, then $d_p(p_1, p_2) = (|r_2 - r_1| + |g_2 - g_1| + |b_2 - b_1|)/(3 * 255)$. And we state:

$$\text{loss}(I_1, I_2) = \frac{\sum_{i=0}^{i=n-1} \sum_{j=0}^{j=m-1} d_p\big(p(I_1, i, j), p(I_2, i, j)\big)}{n * m}, \tag{1}$$

where $p(I_\ell, i, j)$ denotes pixel with coordinates (i, j) in image I_ℓ.

We now introduce *plane graphs*. We recall that a graph $G = (V, E)$ is *planar* if it can be embedded in the plane, in such a way that no two edges cross. Note that several non homeomorphic embeddings may exist for a given planar graph (i.e. several incomparable drawings of the same graph) [3]. Moreover, it is always possible to move the vertices so that the edges are drawn with straight-line segments [6]. Thus by a *plane graph*, we mean a planar graph for which the embedding is fixed and such that (1) every embedded edge is a straight-line and (2) no two embedded edges intersect, except at their endpoints.

Plane graphs are interesting to represent images for at least two reasons. On one hand, they can be nicely represented as combinatorial maps which also provide us with an elegant data structure [20]. On the other hand, isomorphism and subisomorphism problems were recently studied for combinatorial maps [5], with the following interesting consequence:

Theorem 1. *Let $G = (V, E)$ and $G = (V', E')$ be two plane graphs. Deciding whether G and G' are isomorphic is solvable in time $\mathcal{O}(|V| \times |V'|)$. Moreover, if G is face-connected, deciding if G is a pattern of G' is solvable in time $\mathcal{O}(|V| \times |V'|)$.*

A pattern of a graph can be seen as a subgraph obtained by erasing faces and then edges (called *bridges*) separating 2 erased faces and finally singular vertices (called *hinges*) separating 2 non-contiguous erased faces [4]. Other positive algorithmic results for special classes of graphs are described in [9].

In this paper, we develop techniques to extract a plane graph G from an image. In order to rebuild an image, we also need to have (1) a vector P mapping each vertex with the coordinates of corresponding pointel and (2) a vector C mapping each face of G with the colour in which it should be drawn[1]. Although G, P and C are known, a variety of images can still be drawn. Indeed, embedded edges will cross many pixels and divide them into several parts, so which colour should these pixels be given? We choose the following rule: for each split pixel, we consider the colours of its four corners (pointels) (inherited from the faces in which they

[1] Faces of plane graphs can be denoted and saved using directed edges, and a convention stating that such or such face is on the right of such or such directed edge.

stand - pointels that lie between regions are not taken into account), compute the average colour and assign it to the pixel.

We denote by $\mathrm{Im}(G, C, P)$ the resulting image and can now measure the loss between $\mathrm{Im}(G, C, P)$ and original image I using Eq.(1). Obviously, the loss between an image obtained through some compression or representation by graphs could also be looked into *w.r.t.* the gain in terms of space. We claim that whereas the space needed to encode an image is $n \cdot m \cdot K$, where K is a constant corresponding to the size needed to encode one colour vector (here 24), to encode $\mathrm{Im}(G, C, P)$ the space is about $|V| \cdot (\log n + \log m + 3 + K)$.

3 Construction

In this section, we propose a new way to represent an image as a graph, by taking advantage of both image preprocessing tasks introduced in Section 1: the segmentation task, combined to interest pixels extraction. This allows us to obtain graphs that preserve the semantics and the topology of the original images, while providing us with a valid approximation that minimises the loss. Remember that the goal is to extract, given an image I, a graph G such that, with correct P and C functions, we have $\mathrm{loss}(I, \mathrm{Im}(G, P, C))$ as small as possible.

3.1 Segmentation

Any segmentation process aims at defining disjoint regions made of homogeneous sets of pixels (in some predefined way). Many segmentation algorithms exist, and it is generally possible to target an approximate number of regions by tuning the parameters of any chosen algorithm (see Fig. 1).

Aiming at minimising $\mathrm{loss}(I, \mathrm{Im}(G, P, C))$, it would be suitable to more or less preserve the regions of segmented image while building the graph: the boundaries between them should match, in as close a way as possible, the edges of the graph. We recall that two adjacent pixels are separated by a *linel* (see Section 2), so the regions are delimited by boundaries, sets of consecutive separating linels that form cycles. Of course, one given region may be delimited by more than one cycle, and the borders of the image need to be included.

Now our thesis is that the vertices of the graph should be selected among the endpoints of the separating linels described above, that is, the *pointels*. Indeed,

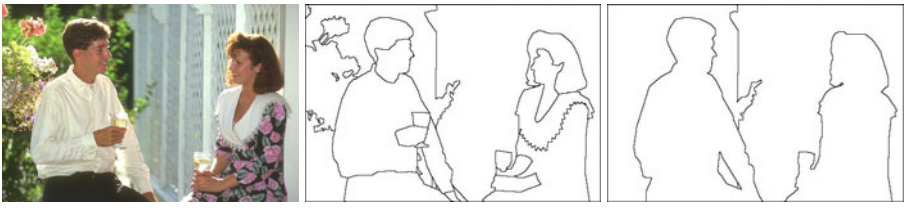

Fig. 1. An image and two segmentations at different levels, coming from the Berkeley Segmentation Dataset [15]

the vertices (and then the edges) must be chosen in such a way as to cover all the important elements structuring the boundaries. Obviously, too many vertices will make us deal with large graphs (which for combinatorial reasons is not recommended). Conversely, too few vertices will result in embedded edges (straight lines) which will be far apart from the boundaries. This means that we have to compress the information displayed in the segmented regions (which was already a compression of the original image): we are not interested in the exact shape of the regions, but in a rough although lifelike sketch (see Fig. 3 w.r.t. Fig. 1).

3.2 Extracting and Cleaning Up Interest Pixels

In order to select the vertices of the graph, we now make use of an interest pixels detector. No consensus exists on what an interest pixel is: each detector (e.g., [8,13]) performs its own measures of local information (based on texture, colour, shape,...) on the image and then extracts most stable and rich sets of pixels w.r.t. these features. While usually applied on original images, we run interest pixels detectors on segmented images. Indeed, segmented images are simplified, and detected pixels will thus be situated on, or close to, boundaries. This will prevent us from detecting pixels on textured areas, salient corners pixels situated inside a region, or pixels corresponding to noise. Note that if the number of extracted interest pixels is low, some regions (e.g. regions strictly included in others) may not be detected and thus may not be represented in the final graph.

Despite these precautions, the set of pixels obtained is generally not satisfying for two reasons. On the one hand, due to the principles of detectors, it is still possible for a pixel to be selected while it is far from any region boundary. We aim at eliminating them: for each interest pixel, we consider a mask containing this pixel and its eight neighbours, and then remove the interest pixel from the set if all the pixels in the mask belong to the same region.

On the other hand, the detector is often not able to come up with all the pixels that act as boundaries to three regions or more (i.e. pixels that belong to one region and have at least two other regions among their eight neighbours). Yet these pixels are necessary in order to have embedded edges of the graph close to the boundaries, so we aim at adding them: we consider 3x3 masks again, this time for all the pixels of the image, and add as interest pixels those whose mask intersect three different regions (or at least two, if the pixel is on image's border). We finally add the four corners of the image to the set.

We are clearly now in possession of a "clean" set of interest pixels that could be the vertices of the graph. However, we are still faced with a problem, well-known to researchers in discrete geometry: when tracing the lines corresponding to the edges of such a graph, because of the discrete nature of the pixels, we can end up with zones that are not colourable, or even to edges which intersect. Thus the vertices of the graph should not be pixels but pointels (with null thickness).

3.3 From Pixels to Pointels

In order to define the vertices, we now draw a parallel between linels separating segmented regions and interest pixels. Since linels lie between pixels, we will

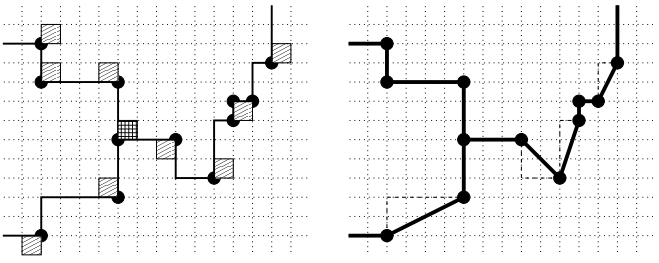

Fig. 2. On the left, the border between 3 regions: 9 (hatched) interest pixels + 1 (striped) intersection pixel yield 12 interest pointels (big dots). On the right, the extracted plane graph: the edges are built following regions boundaries.

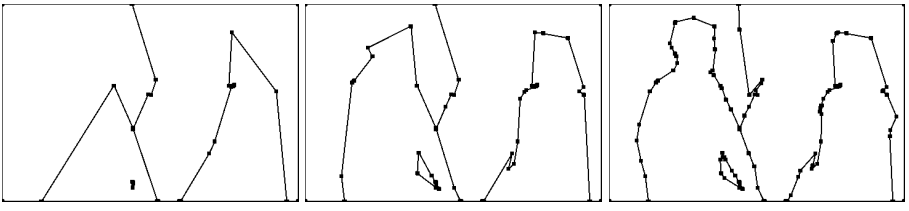

Fig. 3. Several plane graphs obtained from rightmost segmentation in Fig. 1. For the leftmost (resp. middle, rightmost) graph, 25 (resp. 50, 100) interest pixels were detected (and none of them deleted using the uniform mask rule), then 37 (resp. 36, 36) intersection pixels were added, generating 68 (resp. 108, 194) vertices (pointels).

analyse the four corners (pointels) of each interest pixel. Then the goal is to select among these pointels, those which are the most relevant. The rule is as follows: (1) Consider each pointel π of each interest pixel separately; (2) Let p_0, p_1, p_2 and p_3 be the 4 pixels which share pointel π; (3) If $\exists p_i$ such that $c_I(p_{i-1}) \neq c_I(p_i)$ and $c_I(p_i) \neq c_I(p_{i+1})$ (modulo 4) then the pointel is selected.

At this stage, the selected pointels become the vertices of the graph. Note that function P is also defined straightforwardly (we use coordinates of the pointels in the image). They are yet to be related: the edges are built by linking vertices, following the boundaries of each region (see Fig. 2 and 3).

3.4 Triangulation

To complete our method, we finally make use of a kind of Delaunay triangulation. The reason is threefold. (1) Such a technique will improve the robustness of the graph: if the positions of the vertices are slightly modified by transformations applied to the original image, the Delaunay triangulation will remain stable. (2) In a problem such as the plane graph isomorphism discussed in Section 2, triangulation edges will prevent one from matching faces with similar boundaries but different shapes. Note that the benefit of Delaunay triangulation has been

investigated for other problems in Pattern Recognition [7]. (3) The first stage of our technique is a segmentation. Yet the resulting regions may be strictly included one into another (see Fig. 1). On the other hand, in the last stage, the edges are built following the boundaries of the regions. So the graph we get is generally not connected (as shown in Fig. 3), which is not combinatorially suitable; the use of the Delaunay triangulation will settle this issue.

Now, in order to preserve existing edges, a Delaunay triangulation takes place in each face of the embedded graph independently of the others. As a region may not be convex and often contains other included regions, the triangulation edges that are not totally included in the region are eliminated. Finally, we merge together the triangulated faces, and obtain the target plane graph G.

4 Experiments

The public benchmark of the Berkeley Segmentation Dataset [15] contains 300 images (481x321 RGB) of natural scenes, coming from the Corel dataset (widely used in computer vision). Each image contains at least one discernible object, and was segmented at varying levels by several individuals without any particular instruction on the type of criterion to use (see Fig. 1 for an example). The number of regions is at least 2, for an average of 20. As for the interest pixels detector, we have used the one from [1], based on multi-resolution contrast information. Below, I denotes an image, I_k its segmentation into k regions, and $\mathrm{Bg}(I_k, s)$ the graph obtained by extracting s interest pixels from the segmented image and following all the steps described in Section 3 (Bg=*Build Graph*).

4.1 Losses Due to the Graph Extraction Process

Firstly, we have evaluated $\mathrm{loss}(I, I_k)$, that is the loss due to the segmentation of I into k regions. For the 1633 available segmentations, the average loss is 10.0%. We also wanted to study its correlation with the number of regions. So, for regions ranging from 3 to 30, we selected 10 images (for a total of 280 segmentations) and computed the corresponding loss, together with the standard deviation. The results are shown on the left of Fig. 4: no correlation can be found. As the individuals who divided the images did not have any particular instructions, the segmentation was mostly semantic. Regions thus depict objects or parts of objects, and they are not necessarily homogeneous in terms of colours, whatever the level of segmentation.

Next, we have studied how the number of interest pixels affected the loss from I_k (the image segmented into k regions) to $\mathrm{Bg}(I_k, s)$ (the graph obtained by extracting s interest pixels from I_k). To this extent, we have extracted 10 to 500 interest pixels and built corresponding graphs. The results are shown on the right of Fig. 4. The loss ranges from 3.1% for 10 interest pixels to 0.2% for 500. Basically, the larger the extracted graph is, the closer the boundaries of the regions are followed by the edges, thus the lower $\mathrm{loss}(I_k, \mathrm{Bg}(I_k, s))$ is.

Finally note that the processing of interest pixels induces widely less loss than the segmentation stage ($< 3\%$ w.r.t. 10%). To confirm this, we have considered

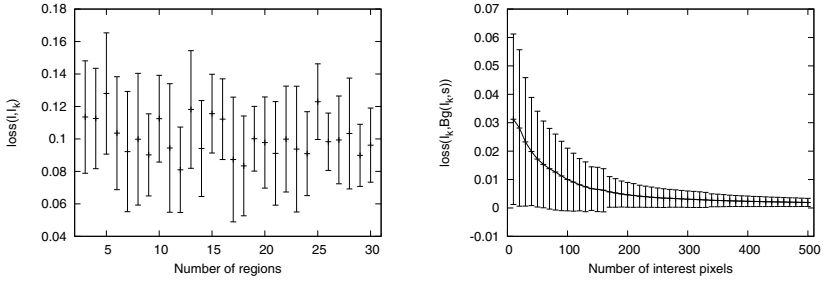

Fig. 4. $\text{loss}(I, I_k)$ w.r.t. the number of regions (on the left) and $\text{loss}(I_k, \text{Bg}(I_k, s))$ w.r.t. the number of interest pixels (on the right)

Table 1. For one image, losses corresponding to different levels of segmentation

s	k	$\lvert V\rvert$	$\text{loss}(I, I_k))$	$\text{loss}(I_k, \text{Bg}(I_k, s))$	$\text{loss}(I, \text{Bg}(I_k, s))$
50	3	104	0.084	0.006	0.089
50	7	123	0.084	0.005	0.089
50	9	175	0.084	0.006	0.090
50	13	149	0.084	0.007	0.091
50	20	207	0.083	0.007	0.090

one particular image at different levels of segmentation k and extracted a fixed number of interest pixels. See Table 1.

4.2 Comparison with Another Plane Graph Extraction Method

In order to compare our graph extraction method, we use another more classic one: interest pixels are directly extracted from original images (without pre-segmentation), then converted into interest pointels as described in Sect. 3.3 (with, added, the 4 corners pointels in order to fit whole image) and related by the Delaunay triangulation. We denote $\text{Bg}(I, t_s)$ this graph and compare $\text{loss}(I, \text{Bg}(I, t_s))$ with $\text{loss}(I, \text{Bg}(I_k, s))$ (see Fig. 5): our method induces slightly less loss, while providing a graph that preserves the semantics of original image.

4.3 Size of the Graph

Given an image, in order to obtain a graph of a given size $|V|$, we can work on the segmentation level and the number of extracted pixels. To assess the trade-off between both, we compare $\text{loss}(I, \text{Bg}(I_k, s))$ for different values of k and s. Of particular interest is the case where $\text{Bg}(I_k, s)$ and $\text{Bg}(I_{k'}, s')$ have similar sizes.The situation for 2 images A and B is depicted in Table 2. $\text{loss}(I, \text{Bg}(I_k, s))$ is quite similar whether we are considering an important number of interest pixels and a small number of regions, or conversely. Thus any of parameters k and s can be tuned to obtain of graph of a given size, with a slight preference for the first due to the decrease of $\text{loss}(I_k, \text{Bg}(I_k, s))$ when s becomes larger.

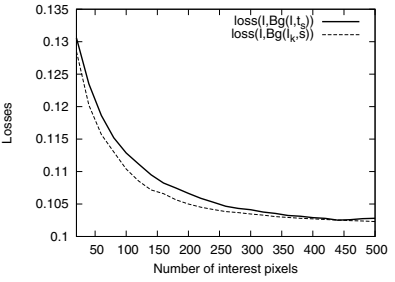

Fig. 5. $\text{loss}(I, \text{Bg}(I, t_s))$ and $\text{loss}(I, \text{Bg}(I_k, s))$ w.r.t. the number of interest pixels

Table 2. Influence of k and s on the losses for extracting a graph of size $|V|$

Image	s	k	$\lvert V\rvert$	$\text{loss}(I, I_k)$	$\text{loss}(I_k, \text{Bg}(I_k, s))$	$\text{loss}(I, \text{Bg}(I_k, s))$
A	80	7	182	0.084	0.004	0.088
	20	18	188	0.076	0.023	0.098
B	80	11	146	0.166	0.009	0.175
	20	25	145	0.166	0.022	0.188

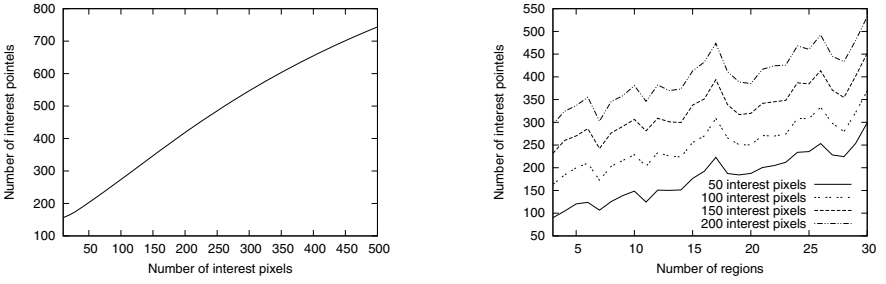

Fig. 6. Number of pointels w.r.t. the number of interest pixels (on the left) and to the number of regions (on the right)

More precisely, to assess the influence of the number s of detected interest pixels on $|V|$, we extracted 10 to 500 interest pixels from 1633 segmented images, added the intersection pixels and computed the corresponding pointels; left-hand curve of Fig. 6 shows that $|V|$ is a quasi-linear function of s.

Finally, concerning the influence of the level of segmentation k on $|V|$, we selected 10 segmentations for a number of regions varying from 3 to 30 (yielding 280 segmentation files); for each of them, we extracted 50, 100, 150 and 200 interest pixels and computed the corresponding pointels (see right of Fig. 6): Even if the curves do not increase steadily, for some fixed s, the more regions there are, the more selected pointels there are; this is due to the intersection pixels, that grows up with the number of regions.

Finally, even though the method is not intended as a compression algorithm, it can be noted that the original images have a theoretical size of 450kB (encoded

as JPEG, they are about 70kB). The segmented files have an average size of 28kB, and a graph with 150 vertices requires less than 1kB.

5 Discussion, Open Questions and Conclusion

The bottleneck problem to using graphs in pattern recognition for image recognition tasks is to extract good graphs, where *good* means both reasonably small and semantically stable. We have introduced a new method for extracting graphs from images, which compresses the original information while preserving the semantics. Experiments show that the loss due to this process is limited. Moreover, these graphs have nice properties that are interesting for isomorphism-related problems: they are planar and connected, and their size is low. Moreover, this method is general enough to be applied to any kind of images (while usual methods are often adapted to particular classes of images (e.g see [16]).

It would be interesting to study alternative loss functions, to experiment with different segmentation algorithms and other ways of creating connexity and structural rigidity. Finally, we aim at proving the validity of the extracted graphs in pattern recognition tasks, and contributing to the field with a free distribution of the described software.

References

1. Bres, S., Jolion, J.-M.: Detection of interest points for image indexation. In: Huijsmans, D.P., Smeulders, A.W.M. (eds.) VISUAL 1999. LNCS, vol. 1614, pp. 427–434. Springer, Heidelberg (1999)
2. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition. Pattern Recogn. and Artificial Intell. 18(3), 265–298 (2004)
3. Cori, R.: Un code pour les graphes planaires et ses applications. In: Astérisque, vol. 27, Soc. Math. de France, Paris, France (1975)
4. Damiand, G., de la Higuera, C., Janodet, J.-C., Samuel, E., Solnon, C.: A polynomial algorithm for subisomorphism of open plane graphs. In: MLG 2009 electronic proceedings (2009)
5. Damiand, G., de la Higuera, C., Janodet, J.-C., Samuel, E., Solnon, C.: A polynomial algorithm for submap isomorphism: Application to searching patterns in images. In: Torsello, A., Escolano, F., Brun, L. (eds.) GbRPR 2009. LNCS, vol. 5534, pp. 102–112. Springer, Heidelberg (2009)
6. Fàry, I.: On straight-line representation of planar graphs. Acta Scientiarum Mathematicarum 11, 229–233 (1948)
7. Finch, A.M., Wilson, R.C., Hancock, E.R.: Matching delaunay graphs. Pattern Recognition 30(1), 123–140 (1997)
8. Harris, C., Stephens, M.: A combined corner and edge detection. In: Proceedings of the Fourth Alvey Vision Conference, pp. 147–151 (1988)
9. Jiang, X., Bunke, H.: Optimal quadratic-time isomorphism of ordered graphs. Pattern Recognition 32(7), 1273–1283 (1999)
10. Kandel, A., Bunke, H., Last, M. (eds.): Applied Graph Theory in Computer Vision and Pattern Recognition. Studies in Computational Intelligence, vol. 52. Springer, Heidelberg (2007)

11. Kropatsch, W., Macho, H.: Finding the structure of connected components using dual irregular pyramids. In: Proc. DGCI 1995, pp. 147–158 (1995)
12. Lienhardt, P.: Topological models for boundary representation: a comparison with n-dimensional generalized maps. Computer-Aided Design 23(1), 59–82 (1991)
13. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
14. Lozano, M.A., Escolano, F.: Protein classification by matching and clustering surface graphs. Pattern Recognition 39(4), 539–551 (2006)
15. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proc. 8th Int. Conf. Computer Vision, July 2001, vol. 2, pp. 416–423 (2001)
`http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/`
16. Riesen, K., Bunke, H.: IAM graph database repository for graph based pattern recognition and machine learning. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) S+SSPR 2008. LNCS, vol. 5342, pp. 287–297. Springer, Heidelberg (2008)
17. Riesen, K., Bunke, H.: Approximate graph edit distance computation by means of bipartite graph matching. Image Vision Comput. 27(7), 950–959 (2009)
18. Rosenfeld, A.: Adjacency in digital pictures. Infor. and Control 26(1), 24–33 (1974)
19. Santo, M.D., Foggia, P., Sansone, C., Vento, M.: A large database of graphs and its use for benchmarking graph isomorphism algorithms. Pattern Recognition Letters 24(8), 1067–1079 (2003)
20. Tutte, W.: A census of planar maps. Canad. J. Math. 15, 249–271 (1963)