A Reduction Method For Graph Cut Optimization

Nicolas Lermé, François Malgouyres

A Reduction Method For Graph Cut Optimization

N. Lermé, F. Malgouyres

M. Lermé is with the Institut Supérieur d'Electronique de Paris, 28 Rue Notre-Dame des Champs, 75006 Paris, France. E-mail: nicolas.lerme@isep.fr.

M. Malgouyres is with the Institut Mathématiques de Toulouse (IMT, CNRS UMR 5219), Université de Toulouse, 118 route de Narbonne, F-31062 Toulouse Cedex 9, France. Phone: +33 (0)5-61-55-85-83. Fax: +33 (0)5-61-55-75-99. E-mail: francois.malgouyres@math.univ-toulouse.fr.

Abstract

In a few years, graph cuts appeared as a leading method in computer vision and graphics due to their efficiency in computing globally optimal solutions to popular minimization problems. Such an approach remains however impractical for very large-scale problems due to the memory requirements for storing the graphs. Among the strategies to overcome this situation, an existing one consists in reducing the size of these graphs by only adding the nodes which satisfy a local condition. In the image segmentation context, this means for instance that we do not need to consider a node when the unary terms are large in its neighborhood. The remaining nodes are typically located in a thin band around the boundary of the segmented object. In this paper, we detail existing strategies to reduce the memory footprint of graph cuts, describe the proposed reduction criterion and we empirically prove on a large number of experiments that the distance between the minimizer found and the global minimizer remains null or very small. We also provide extra parameters for further reducing the graphs and for removing isolated nodes due to noise.

Index Terms

reduction, graph cuts, segmentation, filtering.

I. INTRODUCTION

Graph cuts are a discrete optimization method based on maximum-flow / minimum-cut (max-flow / min-cut) computations in graphs for minimizing energies frequently arising in computer vision and graphics. Since last decades, this technique is become a cornerstone beside these communities for solving a wide range of problems such as denoising, segmentation, registration, stereo, scene reconstruction, optical flow, etc. We refer the reader to [1] for typical applications of graph cuts. Since seminal work of [2] for denoising binary images, graph cuts have known a quick development (after a ten year silence) mainly due to the introduction of a fast max-flow algorithm [1] and efficient heuristics for solving multi-labels problems [3].

In parallel, technological advances in image acquisition have both increased the amount and the diversity of data to process. As an illustration, in the satellite SPOT-5 launched by Arianespace in 2002, the high geometric resolution sensors can capture multispectral and panchromatic images with an imaging swath of 60 km $\times$ 60 km. Each image has a size of 12000 $\times$ 12000 which amounts to about 1GB of data. Similarly, latest medical imaging systems are able to acquire 3D and 3D+t volume data with several billions of voxels.

Although graph cuts can efficiently solve a wide range of problems, their huge memory consumption remains a drawback. To obtain high-resolution output, graph cuts usually build massive multidimensional grid-like graphs containing billions of nodes and even more edges. These graphs do not fit in memory. Currently, most of the max-flow algorithms are totally impracticable to solve such large scale optimization problems. To overcome this situation, some amount of work has recently been done in this direction and a number of heuristics [4], [5], [6], [7], [8], [9] and exact [1] methods [10], [11], [12], [13] have been proposed. However, the heuristics generally either fail to fully capture shape complexities [4], [5], [6] or strongly depend on a low-level segmentation tool [7], [8], [9].

In this paper, we give a simple condition for testing if a node in a graph is really useful to the max-flow computation [14]. The reduced graph is progressively built by only adding the nodes which satisfy this condition. This leads to a straightforward algorithm with a worst-case additional complexity similar to a convolution. Thus, in the manner of [4], [5], [6], [13], the remaining nodes are typically located in a narrow band surrounding the object edges to segment. However, unlike [4], [5], [7], [8], [9], the proposed method can accurately segment thin structures without requiring any other low-level segmentation tool. Experiments clearly show that the solutions obtained on the reduced graphs are identical to the solutions obtained on the whole graphs. Furthermore, the time required by the reduction algorithm is sometimes compensated by the time for computing the max-flow on the reduced graph. This paper complements the preliminary work of [14] by giving algorithmic details, a detailed bibliography on the subject and a large number of experiments for segmenting multidimensional grayscale and color images. Also, two extra parameters are introduced for both further reducing the size of the graphs and removing small segments in the segmentation due to noise. We want also to mention that the method described in this paper is protected by a patent [15] and has already been applied to minimize an energy designed for interactive lung tumor segmentation [16].

The rest of this document is organized as follows. In Section II, we present the state-of-the-art of methods to overcome the memory problem of graph cuts. We review in Section III the graph cuts framework in the image segmentation context. Then, we detail our strategy for reducing graphs in Section IV and present a large number of experiments for segmenting 2D, 2D+t and

[1] By exact, we mean that the maximum flow value remains unchanged and any solution is guaranteed to be a global minimizer.

3D grayscale/color images using two different energy models. We provide an in-depth look for measuring the influence of the reduction parameters in Section IV. We conclude this paper by reminding the main contributions of this work and discuss about potential perspectives in Section V.

II. State-of-the-art

The methods present in the literature for getting round the well known memory problem of graph cuts can be divided into two main categories: single-machine algorithms and parallelized/distributed algorithms. Let us now review these algorithms in this order.

A. Sequential strategies

To our best knowledge, Li *et al.* seem to be the first ones to tackle the problem of memory consumption of graph cuts [7]. Their algorithm works as follows. First, the image is partitioned into small and numerous homogeneous regions thanks to a low-level segmentation algorithm such as watershed [7], [8] or mean shift [9]. A region adjacency graph is produced where each region corresponds to a node in the graph. Then, the max-flow is computed on this graph for getting the segmentation. The underlying assumption is that the final contours are embedded into the pre-segmentation. While this observation is generally not theoretically guaranteed, it is often verified when working on natural images not corrupted by noise. Although this approach drastically reduces the computational burden of graph cuts (about 6x faster according to [7]), the results strongly depend on the low-level segmentation algorithm used and its noise-sensitivity. Moreover, as fairly observed in [8], this approach generally gives better results when over-segmentation occurs, losing the main benefit of such a reduction.

Others have also reported band-based heuristics using a multi-resolution scheme [5], [4]. The principle is to segment a low-resolution image/volume and propagate the solution to the finer level by only building the graph in a narrow band surrounding the interpolated foreground/background interface at that resolution. More specifically, the acceleration strategy consists of three stages: first, a pyramid of images is built with a coarsening operator (coarsening). Next, the coarsest image is segmented and its contours are extracted (segmentation at coarsest level). Finally, the contours are dilated and interpolated at the next higher resolution for building a new reduced graph (uncoarsening). This process continues until the bottom of the pyramid is reached. Such an

approach greatly reduces time and memory consumption of standard graph cuts (about 8x faster and 4x less memory according to [5]). Nevertheless, it generally fails to recover thin structures and is limited to the segmentation of roundish objects. In medical imaging, this is a real drawback since elongated structures like blood vessels are ubiquitous. Moreover, the parameter controlling the band dilation during the projection, plays an important role. Indeed, one usually needs this parameter to be large enough to fully capture details of various shapes complexities. On the other side, wider bands reduce the computational benefits and may also introduce potential outliers far away from the desired object contours.

To avoid the loss of details, Lombaert *et al.* [4] used the information from a Laplacian pyramid. At each level, the bands are extended by including pixels whose value significantly differs between the image and the "coarsened-uncoarsened image". The idea is to capture thin structures which are not visible in the coarse image. This inclusion is controlled by a thresholding parameter which provides a smooth transition between [4] and traditional graph cuts. Although the previous problem is notably reduced, it is still present for low-contrasted details.

Kohli *et al.* recently proposed a finer band-based technique. In contrast to [5], [4], they first define an energy from the full resolution image instead of the low resolution image. Experiments show that this strategy results in significant improvements in both time and segmentation accuracy. But mostly, they compute uncertainty estimates using min-marginals [2] and use them to determine which regions belong to the reduced graph. Such an approach allows to compute solutions very close to the global optimum.

In words, the motivation of [6] is clearly similar to [5], [4], but the strategy used is slightly different since the pyramid of images is restricted to a single one. Although this heuristic cannot ensure the retrieval of thin structures and details, their experiments indicate they are properly recovered for a wide range of parameters, and more robustly than [5], [4].

Lempitsky and Boykov presented more recently an interesting touch-expand algorithm that is able to minimize binary energy functions with graph cuts in a narrow band, while ensuring the global optimality on the solution [13]. The principle is to make a band evolve around the object to segment by expanding the band when the min-cut touches its boundary. This process

[2]The min-marginal encodes the confidence associated with a variable being assigned the label in the optimal solution. The min-marginal of a variable x corresponds to the energy obtained by fixing it to a particular label and minimizing over all remaining variables. The exact min-marginals can be determined exactly and efficiently by reusing previous max-flow computations.

is iterated until the band no longer evolves. Although the algorithm quickly converges toward the global optimal solution, it strongly depends on the initialization and no bound on the band size is given. Thus, the band can progressively encompass the whole volume in the worst case. However, depending on the initialization, the bands are reasonably small in the context of [13] (volume reconstruction). As far as we know, this strategy has not yet been adapted to image segmentation. In particular, the benefit of this strategy strongly depends on the design of an initial band.

Finally, the approach described in [12] is interesting and complements our work. They simplify the graphs by identifying simple edges. An edge (a, b) is said to be simple if its weight is either greater than the sum of all edge weights adjacent to a (except b) or greater than the sum of all edge weights adjacent to b (except a). This test indeed ensures that these edges are not part of the minimum cut since they cannot be saturated by any flow. Once identified, two nodes connected by a simple edge are merged together with their common edges. This process is repeated until no such edge is found. Experiments reveal that most simple edges lie between terminal nodes and pixel nodes, leading to better sparsification when unary terms are locally strong. Even with weak unary terms (strong regularization) and poor initializations [3], the algorithm performed always faster than graph cuts.

B. Parallelized/distributed strategies

In a recent paper, Delong and Boykov design a method for solving the max-flow problem for graphs which do not fit in memory. They propose a new parallelized max-flow algorithm yielding near-linear speedup with the number of processors [10]. As an illustration, on a standard computer, segmenting a volume of size $512 \times 512 \times 256$ takes about 100 secs on a single core against less than 20 secs on eight cores. However, numerical experiments also show that the acceleration of this scheme is very limited since it needs a large number of processors to reach the near-linear speedup and is sensitive to the amount of physical memory. Furthermore, the proposed algorithm clearly remains less efficient on small graphs than standard graph cuts and can only be applied to grid-like graphs.

[3]The energy model used is the one proposed by Boykov and Jolly in [17] (see Section III-C).

More recently, Strandmark and Kahl in [11] introduced an original approach for minimizing binary energy functions in a parallelized/distributed fashion using the max-flow algorithm of [1]. The idea is to decompose the original problem into optimizable sub-problems, solve them independently and update them according to the results of the adjacent problems. This process is iterated until convergence. The key point is that optimality is guaranteed by dual decomposition.

More precisely, the solutions to the sub-problems are constrained to be equal on an overlap. They solve the original problem by finding a saddle point of the Lagrangian of the constrained problem. This min-max problem is solved by alternating minimization over its primal variables and maximization over its dual variables. The minimizations are done independently of each other on the processing elements. The maximization combines the results obtained on the overlapping bands. It consists in an update of the dual variables. To reflect this change, the weights in the graphs corresponding to the sub-problems are modified and the corresponding solutions are recomputed. This scheme is repeated until the solutions of the variables on the overlap are equal. This iterative scheme is efficient since only a few edge costs change between iterations and then search trees can be efficiently reused [18]. Moreover, the number of edge costs which change decrease as the number of iterations increase.

Experiments in [11] for image segmentation and stereo clearly demonstrate that both faster processing on multi-core computers and the ability to solve large scale problems over a distributed network. As an illustration, such an approach is able to segment a graph requiring 131GB of memory in 38 secs. To our best knowledge, the proposed work is the first to segment 4D volume data of moderate size using graph cuts while keeping optimality on the solution. Furthermore, in the image segmentation context, the algorithm is stable over a large range of values of the regularization parameter. Nevertheless, the algorithm is slower for solving some instances where the object to segment is not uniformly spread over the image. Also, notice that the proposed strategy is only effective for graphs for which the max-flow algorithm of [1] is. In particular, the latter becomes less effective than a push-relabel algorithm for dense graphs.

III. GRAPH CUTS FRAMEWORK

A. *Energy minimization via graph cuts*

Consider an image $I : \mathcal{P} \subset \mathbb{Z}^d \rightarrow [0,1]^c$ ($d > 0$, $c > 0$) as a function, mapping each point (called pixel) $p \in \mathcal{P}$ to a value $I_p \in [0,1]^c$. We define a binary segmentation as a mapping

u assigning each element of $\mathcal{P}$ with the value 0 for the background and 1 for the object and we write $u \in \{0,1\}^{\mathcal{P}}$. In the energy minimization approach, a popular strategy consists in minimizing a Markov Random Field of the form [17]:

$$E(u) = \beta \sum_{p \in \mathcal{P}} E_p(u_p) + \sum_{(p,q) \in \mathcal{N}} E_{p,q}(u_p, u_q), \tag{1}$$

among $u \in \{0,1\}^{\mathcal{P}}$ and for a fixed $\beta \in \mathbb{R}^+$. The neighborhood system $\mathcal{N} \subset (\mathcal{P} \times \mathcal{P})$ is a subset of all pixel pairs $(p,q) \in (\mathcal{P} \times \mathcal{P})$. In this context, we will use the following standard neighborhoods:

$$\begin{aligned} \mathcal{N}_0 &= \{(p,q) \in (\mathcal{P} \times \mathcal{P}) \mid \textstyle\sum_{i=1}^{d} |q_i - p_i| = 1\} \quad \text{or,} \\ \mathcal{N}_1 &= \{(p,q) \in (\mathcal{P} \times \mathcal{P}) \mid |q_i - p_i| \leq 1 \ \forall 1 \leq i \leq d\}, \end{aligned}$$

where p_i denotes the i^{th} coordinate of p. In what follows, "connectivity 0" and "connectivity 1" respectively refer to the use of $\mathcal{N}_0$ and $\mathcal{N}_1$ neighborhoods [4].

In equation (1), the data term $E_p(\cdot)$ is the cost for assigning the label u_p to the pixel p without regards to its neighbors. On the other hand, the smoothness term $E_{p,q}(\cdot)$ assumes that the boundaries of the segmentations are smooth. More precisely, it penalizes pixel pairs (p,q) having different labels. It can also be used to better align boundaries of the segmentation on the image edges having a strong gradient. Notice that we only consider pairwise interactions between pixels and do not consider models using higher order terms.

Consider now a weighted directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, c)$ where $\mathcal{V} = \mathcal{P} \cup \{s,t\}$ is the set of nodes, $\mathcal{E} \subset (\mathcal{V} \times \mathcal{V})$ is the set of edges and $c : \mathcal{E} \to \mathbb{R}^+$ is a weighting function defining the edge capacities. The terminal nodes s and t are respectively called the source and the sink. Additionally, we split the set of edges $\mathcal{E}$ into two disjoint sets $\mathcal{E}_n$ and $\mathcal{E}_t$ denoting respectively n-links (neighborhood links) and t-links (terminal links):

$$\begin{aligned} \mathcal{E}_n &= \{(p,q) \in \mathcal{E} \mid (p,q) \in (\mathcal{P} \times \mathcal{P})\}, \\ \mathcal{E}_t &= \{(s,p) \in \mathcal{E} \mid p \in \mathcal{P}\} \bigcup \{(p,t) \in \mathcal{E} \mid p \in \mathcal{P}\}. \end{aligned} \tag{2}$$

In accordance with the graph construction given in [19], we consider (without loss of generality) that a node is linked to at most one terminal:

$$(s,p) \in \mathcal{E}_t \Rightarrow (p,t) \notin \mathcal{E}_t, \qquad \forall p \in \mathcal{P}.$$

[4] As an illustration, each pixel has respectively 4 and 8 neighbors in 2D, 6 and 26 neighbors in 3D and finally 8 and 80 neighbors in 4D, for $\mathcal{N}_0$ and $\mathcal{N}_1$ neighborhoods.

We also summarize the t-links capacities for any node $p \in \mathcal{P}$ by

$$c(p) = c(s,p) - c(p,t). \tag{3}$$

We denote by $\mathcal{C} = (\mathcal{S}, \mathcal{T})$ a s-t cut which is a partition of $\mathcal{V}$ such that $s \in \mathcal{S}$ and $t \in \mathcal{T}$. For any s-t cut $\mathcal{C}$ in $\mathcal{G}$, we define the value of $\mathcal{C}$ by

$$v(\mathcal{C}) = \sum_{\substack{(p,q)\in\mathcal{E} \\ p\in\mathcal{S}, q\in\mathcal{T}}} c(p,q).$$

We also define $u^{\mathcal{C}} \in \{0,1\}^{\mathcal{P}}$ as the underlying segmentation of $\mathcal{C}$ in $\mathcal{G}$:

$$u_p^{\mathcal{C}} = \begin{cases} 0 & \text{if} \quad p \in \mathcal{T} \\ 1 & \text{if} \quad p \in \mathcal{S} \end{cases}, \qquad \forall p \in \mathcal{P}.$$

In other words, pixels in $\mathcal{S}$ belong to the object and those in $\mathcal{T}$ to the background.

Notice first that $\mathcal{C} \rightarrow u^{\mathcal{C}}$ makes a one-to-one correspondence between s-t cuts in $\mathcal{G}$ and the segmentations of the image. Then, the key idea of graph cuts is to build a graph $\mathcal{G}$ such that for any s-t cut $\mathcal{C}$ in $\mathcal{G}$, we have

$$v(\mathcal{C}) = E(u^{\mathcal{C}}) + K, \tag{4}$$

for some additional constant $K \in \mathbb{R}$ independent of $\mathcal{C}$. When pairwise terms are submodular, i.e. when they satisfy

$$E_{p,q}(0,0) + E_{p,q}(1,1) \leq E_{p,q}(0,1) + E_{p,q}(1,0), \qquad \forall (p,q) \in \mathcal{N},$$

$\mathcal{G}$ can be constructed as described in [19]. Then, (4) guarantees that the min-cut in $\mathcal{G}$ corresponds to a minimizer of (1). Moreover, as it is well known, the min-cut can be efficiently computed using a max-flow algorithm such as [1]. Again, once the min-cut $\mathcal{C}^*$ is computed in $\mathcal{G}$ with a max-flow algorithm, $u^{\mathcal{C}^*}$ minimizes (1). In the next sections, we review two classical energy models for segmenting images with graph cuts.

B. TV+L^2 energy model

Initially introduced by Rudin, Osher and Fatemi [20], the TV+L^2 (ROF) model and its variants have been a very active research topic in image restoration. This model has also successfully demonstrated its efficiency for segmenting cars in video [21]. It is only defined on grayscale

images but can of course be applied to a grayscale image resulting from a multichannel image. In the image segmentation context, the solution is taken as a level-set [5] of the minimizer u^* of

$$\underbrace{\sum_{\mu=0}^{L-2} \sum_{(p,q)\in\mathcal{N}} w_{p,q}|u_p^\mu - u_q^\mu|}_{TV(u)} + \beta\|u - I\|_2^2, \qquad \beta \in \mathbb{R}^+, \tag{5}$$

where L denotes the maximum intensity of I, $\|.\|_2$ denotes the Euclidean distance in $\mathbb{R}^{\sharp\mathcal{P}}$, $I \in \mathbb{R}^{\mathcal{P}}$ is initial data $TV(u)$ denotes the Total Variation of $u \in \mathbb{R}^{\mathcal{P}}$ and $w_{p,q}$ is a weight proportional to the distance between p and q. While the second term maintains a proximity to a level-set of I, the solution is regularized by the first one. Expressing the two terms of (5) in terms of level-sets, we observe that the μ level-set of u^* is a minimizer of the binary energy

$$\underbrace{\sum_{(p,q)\in\mathcal{N}} w_{p,q}|u_p^\mu - u_q^\mu|}_{TV(u^\mu)} + 2\beta \sum_{p\in\mathcal{P}} u_p^\mu[(\mu - \frac{1}{2}) - I_p] + I_p, \tag{6}$$

among $u^\mu \in \{0,1\}^{\mathcal{P}}$ (see [22]). The latter problem has the form described in (1) and can be minimized by a graph cut. We remind that this formulation cannot handle color images. In practice, color images need to be converted into grayscale images before they are segmented.

C. Boykov-Jolly energy model

In [17], authors introduced an energy model for segmenting images using graph cuts. Unlike the previous model, the user can provide object ($\mathcal{O} \subset \mathcal{P}$) and background seeds ($\mathcal{B} \subset \mathcal{P}$) in an interactive fashion. The role of these seeds is twofold: reducing the cuts space and computing probability distributions of the intensity for the object and the background. Formally, we have

$$\begin{cases} E_p(1) = -\log\ \mathbb{P}(I_p \mid p \in \mathcal{O}) \\ E_p(0) = -\log\ \mathbb{P}(I_p \mid p \in \mathcal{B}) \end{cases} \quad \text{and} \quad E_{p,q}(u_p, u_q) = B_{p,q}|u_p - u_q|, \tag{7}$$

where $\mathbb{P}(.)$ is a probability density function, $I_p \in [0,1]^c$ denotes the intensity at voxel p and $B_{p,q}$ is a weighting function used to map similarity between voxels to graph weights. The distribution of the object and the background are generally estimated either using normalized histograms or Gaussian mixture models. As usual, the data term favors the belonging of each pixel $p \in \mathcal{P}$ to the object or the background class while the smoothness term penalizes neighboring pixels p

[5] In this setting, we denote by $u^\mu = \{p \in \mathcal{P} \mid u_p^\mu = 1\}$ the μ level-set of u at the level μ where $u_p^\mu = \mathbf{1}_{\{u_p \geq \mu\}}$.

and q having different labels. In its simplest form, the weight of this penalization only depends on the gradient and favors boundaries with a strong gradient. Notice that the weight can also embed more complex features such as textures or gradient direction. A popular choice is to use (and we use it for all the experiments presented in this paper)

$$B_{p,q} = \frac{1}{\|p-q\|_2} \exp\Big(- \frac{\|I_p - I_q\|_2^2}{2\sigma^2} \Big), \tag{8}$$

where $\sigma \in \mathbb{R}^+$ is a free parameter and $\|.\|_2$ is the Euclidean norm (either in $\mathbb{R}^d$ or $\mathbb{R}^c$).

IV. Reducing graphs

A. Principle

As we have seen before, the memory usage for segmenting high-resolution data by graph cuts can be prohibitive. As an illustration, the max-flow algorithm of [1] (version 3.0) allocates $28\sharp\mathcal{P} + 16\sharp\mathcal{E}_n$ bytes [6], where the operator '$\sharp$' stands for the cardinality of a set. One can easily observe that for a fixed amount of RAM, the maximum volume size decreases quickly as the dimension d increases. Nevertheless, as showed in [14], most of the nodes in the graph are useless during the max-flow computation since they are not traversed by any flow (see Figure 1). Ideally, one would like to extract the smallest possible graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ from $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ while keeping the max-flow value f'^* in $\mathcal{G}'$ identical or very close to the max-flow value f^* in $\mathcal{G}$. In words, we want to minimize the relative size of the reduced graph defined as

$$\rho = \frac{\sharp\mathcal{V}'}{\sharp\mathcal{V}}, \tag{9}$$

under the constraint that $f^* \simeq f'^*$. In fact, this is an ideal optimization problem which we will not try to solve since the method for determining $\mathcal{G}'$ also needs to be (very) fast. In order to represent the potential of this idea, we represent in the right hand-side of Figure 1 the flow passing in the graph for segmenting an image with the TV+L^2 model (see Section III-B). In this image, we assign to each pixel p the sum of incoming and outgoing flow passing through p multiplied by $\mathrm{sign}(c(p))$ (where $c(p)$ is defined in (3)) and the sign function is defined by:

$$\mathrm{sign}(t) = \begin{cases} +1 & \text{if } t > 0, \\ -1 & \text{if } t < 0, \\ 0 & otherwise. \end{cases}$$

[6] We remind that $\mathcal{P}$ is the set of pixels/voxels and $\mathcal{E}_n$ denotes the set of n-links (see (2)).

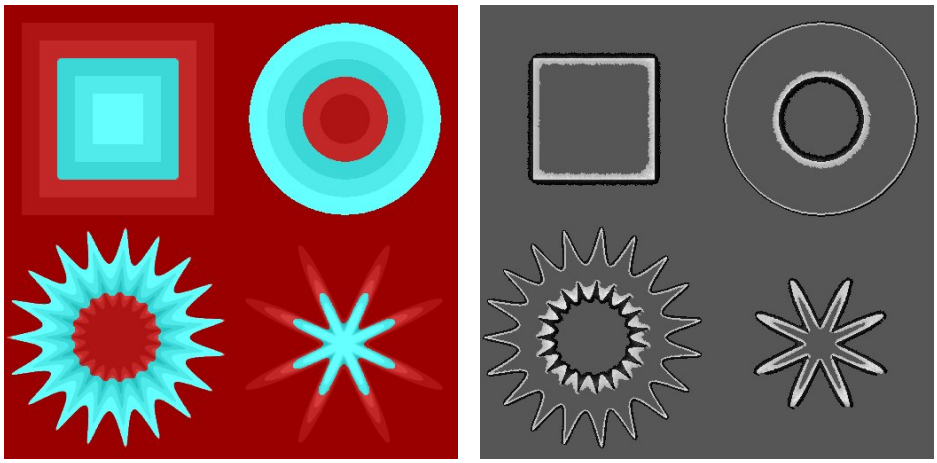


Fig. 1. Illustration of the flow passing in a graph for segmenting a synthetic 2D image using a TV+L^2 model. On the left image, the segmentation is superimposed on the image by transparency. For every pixel p of the right image, when some flow is passing from s to p (resp. from p to t) we display a white pixel (resp. black pixel). Gray pixels are the nodes not traversed by any flow in the graph.

Then, we adopt the following representation: white pixels (resp. black pixels) denote that an amount of flow passed from the source s to a node p (resp. from a node p to the sink t). Gray pixels are the nodes not traversed by any flow in the graph. Clearly, only a small part of the nodes is used during the max-flow computation.

First, let us introduce some terminology before describing our method for building $\mathcal{G}'$. For any $B \subset \mathbb{Z}^d$ (in practice, B will be a square centered at the origin) and $p \in \mathcal{P}$, we denote by B_p the set translation of B by the point p:

$$B_p = \{q + p \mid q \in B\}.$$

For $Z \subset \mathcal{P}$ and $B \subset \mathbb{Z}^d$, we denote by Z_B the dilation of Z by the structuring element B as:

$$Z_B = \{p + q \mid q \in B, p \in Z\} = \bigcup_{p \in Z} B_p.$$

We also define, for any $Z \subset \mathcal{P}$, the maximal amount of flow that might come in and go out through the n-links by

$$P_{in}(Z) = \sum_{\substack{p \notin Z, q \in Z \\ (p,q) \in \mathcal{N}}} c(p,q), \quad P_{out}(Z) = \sum_{\substack{p \in Z, q \notin Z \\ (p,q) \in \mathcal{N}}} c(p,q).$$

Finally, we define the maximum amount of flow passing through the t-links and the flow

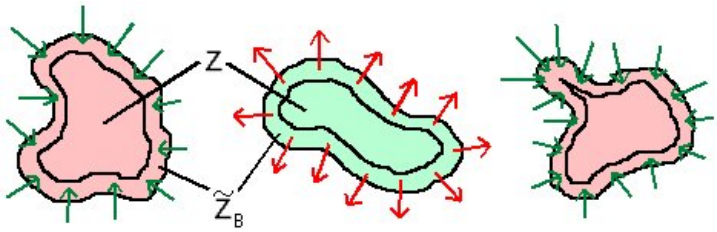


Fig. 2. Principle of the reduction. Red area and arrows (resp. green area arrows) denote the flow which get in (resp. out of) Z_B. The nodes from Z are removed since Z satisfies (10). Remaining nodes are typically located in the narrow band $Z_B \setminus Z$.

orientation by

$$A(Z) = \sum_{p \in Z} |c(p)|, \qquad O(Z) = \sum_{p \in Z} \mathrm{sign}(c(p)),$$

The intuitive idea for building $\mathcal{G}'$ is to remove from the nodes of $\mathcal{G}$ any $Z \subset \mathcal{P}$ such that

$$\left\{ \begin{array}{ll} \text{either} & \Big(O(Z_B) = +\sharp Z_B \text{ and } A(Z_B \setminus Z) \geq P_{out}(Z_B)\Big), \\ \text{or} & \Big(O(Z_B) = -\sharp Z_B \text{ and } A(Z_B \setminus Z) \geq P_{in}(Z_B)\Big). \end{array} \right. \tag{10}$$

As an illustration of these conditions, the last (resp. first) condition of the test (10) implies that all the flow that might come in (resp. go out) the region Z_B does so by traversing its boundary and can be absorbed (resp. provided) by the band $Z_B \setminus Z$ (see Figure 2).

Building such sets Z is done by testing each individual pixel $p \in Z$. In order to do so, we know that the conjunction of conditions (10) for every set $\{p\}$, where $p \in Z$, implies (10) for Z. Considering B, a square window of size $(2r+1)$ $(r > 0)$ centered at the origin, an even more conservative test for $p \in Z$ is:

$$\left\{ \begin{array}{ll} \text{either} & \Big(\forall q \in B_p,\ c(q) \geq +\delta\Big), \\ \text{or} & \Big(\forall q \in B_p,\ c(q) \leq -\delta\Big), \end{array} \right. \tag{11}$$

where $\delta = \frac{P(B)}{(2r+1)^2 - 1}$ and $P(B)$ denotes the perimeter of B with

$$\begin{aligned} P(B) = \max(&\sharp\{(p,q) \ : \ p \in B, q \notin B \text{ and } (p,q) \in \mathcal{N}\}, \\ &\sharp\{(q,p) \ : \ p \in B, q \notin B \text{ and } (q,p) \in \mathcal{N}\}). \end{aligned}$$

The main advantage of (11) is that it can be easily computed. If moreover

$$c(p,q) \leq 1 \qquad (p,q) \in \mathcal{E},$$

(which is true for the energies described in Section III-B and III-C [7]) and (11) holds, one can easily check that the condition (10) holds for $Z = \{p\}$. For instance, the first condition of (11) implies:

$$\begin{aligned} A(B_p \setminus \{p\}) &= \textstyle\sum_{q \in B_p \setminus \{p\}} |c(q)| \\ &\geq [(2r+1)^2 - 1]\delta \\ &\geq P(B) \\ &\geq P_{out}(B_p). \end{aligned}$$

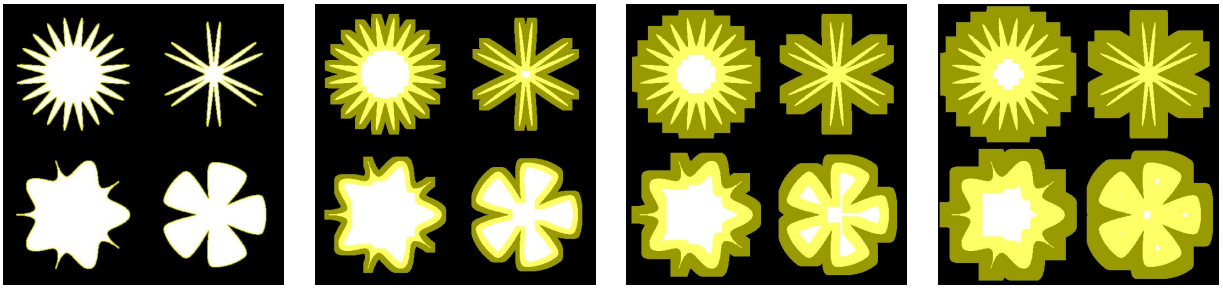
In words, for any node $p \in Z$ satisfying the first (resp. second) condition of (11), all its neighbors $q \in B_p$ are only linked to the source s (resp. the sink t) and the flow that might come in (resp. go out) through t-links in $B_p \setminus \{p\}$ suffices to saturate the n-links going out of (resp. in) B_p. The node p is useless and can be removed from $\mathcal{G}$. Therefore, we consider $\mathcal{G}'$ a subgraph of $\mathcal{G}$ such that $\mathcal{V}' = \mathcal{P}' \cup \{s, t\}$, where:

$$\mathcal{P}' = \{p \ \in \ \mathcal{P} \mid (11) \text{ does not hold for } p\}.$$

The experiments presented in Section IV-C.3 confirm the dependence between the size of the reduced graph and the model parameters (see Figure 3). Indeed, when minimizing (1) via graph cuts as described in Section III-A, the t-links capacities are all multiplied by β. Thus, it is straightforward to observe that the condition (11) is more difficult to satisfy as β decreases. In such a situation, we need a larger window radius for decreasing δ in order to diminish the size of the reduced graph. This results in wider bands around the object contours. Notice that when β is small, the role of the regularization term $E_{p,q}(\cdot)$ is increased. Notice also that when the window radius is increased, there is generally less chance that the test is satisfied for the entire window. Conversely, we can afford large δ and therefore small window radius when β is large. Thus, the reduced graph consists of narrow bands around the object edges.

Additionally, we investigate some ways to relax the condition (11) for further reducing the size of the reduced graph. A simple way is to multiply δ by a factor $\gamma \in [0, 1]$. Then, as γ decreases to 0, the condition (11) can be satisfied for a larger number of nodes. Typically, when $\gamma = 0$, we only test the sign of contracted capacities (see (11)). Another way is to allow some nodes in B_p to fail complying with the test. The proportion of nodes satisfying the test is controlled by a

[7] If the condition (11) does not hold, δ can for instance be multiplied by $\max_{(p,q) \in \mathcal{N}} c(p, q)$.



(a) $\rho = 5.75\%$ (b) $\rho = 32.58\%$ (c) $\rho = 50.24\%$ (d) $\rho = 64.77\%$

Fig. 3. Tuning of the window radius for segmenting a synthetic 2D image with a TV+L^2 model in connectivity 1. From left to right: reduced graphs are superimposed in yellow on the original image for the window radius $r = 1, 8, 15, 22$. The relative size of the reduced graph is indicated below each image.

parameter called $\eta \in [0, 1]$. Then, as η decreases, the condition (11) can be satisfied more easily since a larger proportion of nodes can be connected to opposite terminals. Embedding these two extra parameters leads to the following condition:

$$\begin{cases} \text{either} & \Big(\sharp\{q \in B_p \mid c(q) \geq +\delta\gamma\} \geq \eta\sharp B_p\Big), \\ \text{or} & \Big(\sharp\{q \in B_p \mid c(q) \leq -\delta\gamma\} \geq \eta\sharp B_p\Big). \end{cases} \tag{12}$$

Unlike the window radius parameter, γ and η parameters can further decrease the graph size but do not offer any guarantee on the final segmentation. However, for time-critical applications, this can be particularly useful when optimality does not represent a major constraint. As regard to the parameter η, it can also be used to remove noise in the segmentation.

In contrast to the test described in [23], we do not have any theoretical guarantees about the nature of the minimizer found with the proposed test. Nevertheless, the numerical experiments presented in Section IV-C.3 exhibit important memory gains while keeping a null distance between the segmentations obtained with and without reduction. But most importantly, these experiments show that the relative max-flow error between f^* and f'^* (see Appendix I) is null (or extremely small). This tendency is also confirmed by the experiments of [23] where both tests are compared and depict similar performance. This clearly means that the proposed heuristic is of very good quality. In the next section, we detail a fast algorithm for building $\mathcal{G}'$.

B. Algorithmic considerations

1) Unoptimized algorithm

From the Section IV-A, an easy-to-implement non-optimized algorithm emerges: for each pixel p of the input image, we can check if the condition (12) holds by browsing the window of radius r centered at p. If so, we do not add the node to $\mathcal{G}'$. Since the neighborhood of each pixel is visited exactly once, the graph construction resembles a convolution and has a worst-case complexity of $O(\sharp\mathcal{P}\sharp B)$. Notice that δ is computed only once. When a node cannot be removed from $\mathcal{G}$, we connect it to its constructed neighbors. We keep track of these neighbors with an array of dimension $(d-1)$.

2) Incremental algorithm

For large window radii, the unoptimized algorithm becomes inefficient as soon as the image size and d increase. Nevertheless, one can observe that the condition (12) can be decomposed as sums along the dimensions d yielding an algorithm with a complexity of $O(\sharp\mathcal{P})$, except for image borders. We now detail this incremental algorithm in the 2D case with a connectivity 0. We consider a square window B of size $(2r+1)$, $(r>0)$. First, for any point $p \in \mathcal{P}$ and $\delta' \geq 0$, we define

$$g_{\delta'}(p) = \begin{cases} 1 & \text{if} \quad c(p) \geq +\delta', \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

We either denote $g_{\delta\gamma}(p)$ or $g_{\delta\gamma}(i,j)$ for any pixel $p=(i,j) \in \mathcal{P}$ (it will never be ambiguous once in context). In what follows, we only describe the computation of $\sharp\{q \in B_p \mid c(q) \geq +\delta\gamma\}$. The other case can easily be deduced by adapting the definition of (13). The key idea is to decompose $\sharp\{q \in B_p \mid c(q) \geq +\delta\gamma\}$ as two sums where the first one sums over each row in a column while the second one sums over all columns. First, we introduce an array M whose size is the image width, where each element contains the sum of the values of $g_{\delta\gamma}(\cdot)$ over a vertical segment of B_p. More precisely, if we denote M_{i_0,j_0} the state of table M at the beginning of the computation at the pixel $p=(i_0,j_0) \in \mathcal{P}$, we have:

$$M_{i_0,j_0}[i] = \begin{cases} \sum_{l=-r}^{+r} g_{\delta\gamma}(i, j_0+l) & \text{if} \quad i \leq i_0 + r, \\ \sum_{l=-r}^{+r} g_{\delta\gamma}(i, j_0+l-1) & \text{if} \quad i > i_0 + r, \end{cases} \tag{14}$$

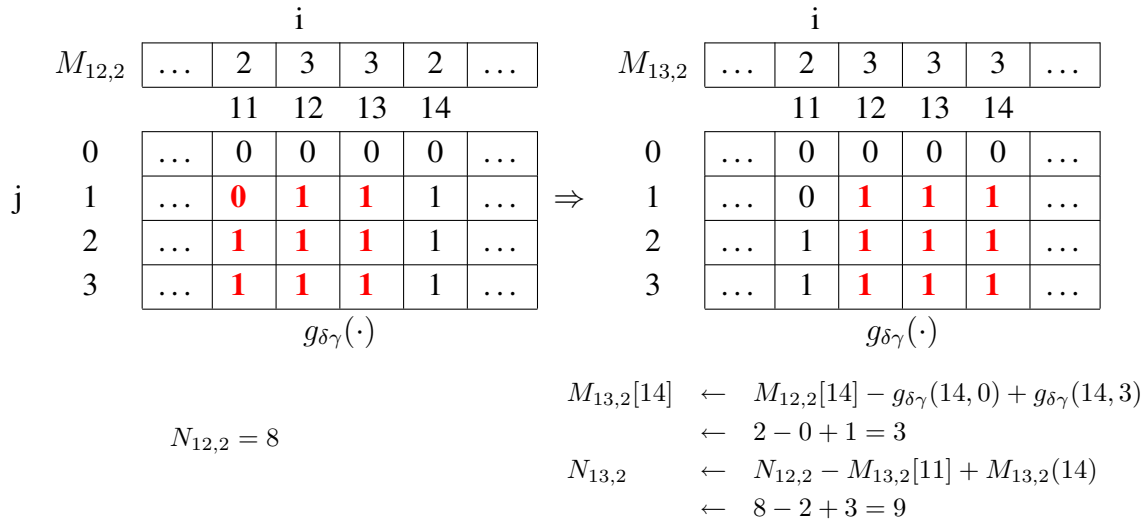


Fig. 4. Illustration of the incremental algorithm for a 2D image with $r = 1$, $\gamma = 1$ and $\eta = 1$. In this example, only the node corresponding to the pixel $p = (13, 2)$ is added to $\mathcal{G}'$ since $|N_{13,2}| = (2r + 1)^2 = 9$.

except for image borders. Additionally, we maintain a variable N_{i_0,j_0} summing the elements of M along an interval of size $2r + 1$:

$$N_{i_0,j_0} = \sum_{c=-r}^{+r} M_{i_0,j_0}[i_0 + c], \qquad \forall (i, j) \in \mathcal{P}.$$

We trivially have $N_{i_0,j_0} = \sharp\{q \in B_p \mid c(q) \geq \delta\gamma\}$, for $p = (i_0, j_0)$. Then, for ensuring the property (14) at the next pixel $p = (i_0 + 1, j_0) \in \mathcal{P}$, we update M before N with:

$$\begin{aligned} M_{i_0+1,j_0}[i_0 + r + 1] &\leftarrow M_{i_0,j_0}[i_0 + r + 1] - g_{\delta\gamma}(i_0 + r + 1, j_0 - r - 1) + g_{\delta\gamma}(i_0 + r + 1, j_0 + r) \\ N_{i_0+1,j_0} &\leftarrow N_{i_0,j_0} - M_{i_0+1,j_0}[i_0 - r] + M_{i_0+1,j_0}[i_0 + r + 1] \end{aligned}$$

The contracted capacities are only evaluated once: when shifting from one position to the next one. Therefore, the optimized algorithm builds the reduced graph with a complexity of $O(\sharp\mathcal{P})$, except for image borders. In particular, the complexity becomes independent of the window radius. Also, one can notice that the cost of such an algorithm is directly proportional to the cost for evaluating the contracted capacities. However, for the energy models presented in this document, these capacities can be efficiently pre-computed and stored in lookup tables. The memory storage required by the incremental graph construction algorithm lies in the table M which is of dimension $d - 1$. Thus, the extra memory usage is negligible with respect to both the image and the graph size.

The unoptimized algorithm remains quite general and can be extended in various ways. For instance, one can imagine an adaptive version where r varies automatically according to the image content. This forces us to guess the optimal window radius r^* for each node. This can be done by examining all window radii in a fixed range $\{0, \ldots, r_{max}\}$ ($r_{max} > 1$) until r^* is found. Unlike the unoptimized algorithm, the worst-case complexity now becomes $O(\sharp\mathcal{P} \cdot T(r_{max}))$ where $T(r_{max})$ denotes the cost for examining all nodes for an increasing radius up to r_{max}. Although this approach permits to build smaller graphs, the construction of the graphs suffers from a higher computational cost. It becomes also more difficult to avoid repetitive evaluations of the contracted capacities like in the incremental algorithm.

Notice also that the reduction can easily be parallelized. First, due to the locality of data and operations, the unoptimized algorithm could also be easily parallelized on GPU since the result of condition (12) can be independently evaluated on each node. Secondly, when the reduced graph contains several connected components, we could solve the max-flow on each connected component independently. In some situations (such as the segmentation of noisy images), this approach could be very efficient since the max-flow computation would become trivial for a large amount of connected components whose nodes are all linked to the same terminal [8].

C. Numerical experiments

In the subsequent sections, all experiments are performed on an Intel Xeon X5690 @ 3.47GHz with 12 processors and 64GB of RAM using the max-flow algorithm [9] of [1]. Running times include the graph construction, the max-flow computation as well as the construction of the solution. Times are expressed in seconds and averaged over ten runs.

1) The window radius parameter

The Figure 5 measures the impact of the window radius with respect to speed and memory usage and compares these results to standard graph cuts (bottom row) for segmenting 2D and 2D+t

[8]Indeed, the condition (12) does not imply that both terminals are linked to the grid nodes unless we have $\gamma = 0$ and $\eta = 1$.

[9]The code is freely available at `http://pub.ist.ac.at/~vnk/software.html`. We also want to point out the existence of specialized implementations taking advantage of grid graphs using less memory. For instance, a possible specialization of [1] on 32-bit targets can be reduced to 24 bytes per node and 4 bytes per edge. This diminishes the amount of memory needed of 2/3 in 2D and 3/4 in 3D.

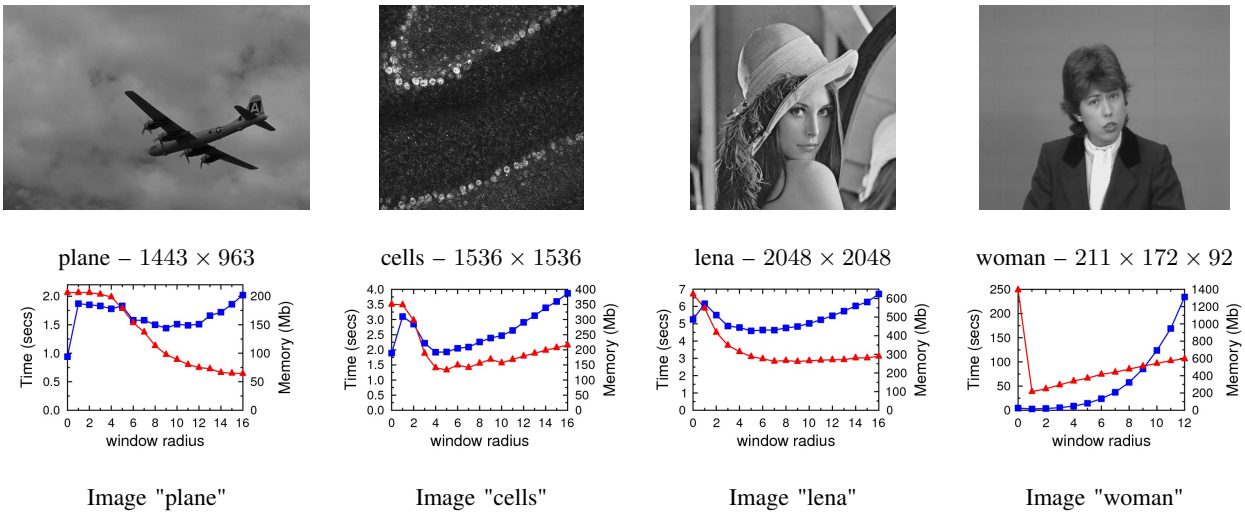


Fig. 5. Influence of window radius (bottom row) for segmenting 2D and 2D+t images (top row) with a TV+L^2 model in connectivity 1. On the bottom row, blue curve with squares and red curve with triangles correspond respectively to execution time and the amount of memory allocated for the graph. Standard graph cuts correspond to a window radius equal to 0.

data (top row) in connectivity 1. On the bottom row, the blue curves with squares correspond to time consumption and the red curves with triangles correspond the memory of the reduced graphs. Standard graph cuts correspond to $r = 0$.

First, the segmentations obtained by standard graph cuts and reduced graph cuts are identical. We also observe that the reduced graph cuts are always faster and requires less memory than the former except for the image "plane". One can also observe that both curves are approximately convex and the minimal relative size of the reduced graph (denoted by ρ^*) is reached for some radius $r^* > 0$. Notice that r^* naturally depends both on the image structure and the model parameters. The intuitive reason for both curves to be approximately convex is that each individual test of (12) can be satisfied more easily when r increase, since δ decreases with r. Nevertheless, when r is larger, the condition becomes more and more difficult to satisfy because a larger number of individual test must hold. Notice that this experiment is chosen to illustrate the behavior when r changes. However, we generally take $r = 1$ for most of the images used (see Tables I and II).

2) Estimation of the distributions

Before presenting numerical experiments, we detail how the distributions $\mathbb{P}(I_p \mid p \in \mathcal{O})$ and $\mathbb{P}(I_p \mid p \in \mathcal{B})$ are estimated in (7) using Normalized Histograms (NH). Since we use the same strategy for the object and the background, we only describe it for $\mathcal{B}$. Let $N_b \in \mathbb{N}^*$ denotes the number of bins. We call, for $q \in \{0, \ldots, N_b - 1\}^c$,

$$H_k = \frac{\sharp\{p \in \mathcal{B} \mid \frac{q_i}{N_b} \leq (I_p)_i < \frac{q_{i+1}}{N_b}, \forall 1 \leq i \leq c\}}{\sharp\mathcal{B}}$$

where we remind that $I_p \in [0,1]^c$ and $(I_p)_i$ is the i^{th} channel of I_p. We then approximate $\mathbb{P}(I_p|p \in \mathcal{B})$ by

$$(G_{\sigma'} * H)_{I_p}$$

where $G_{\sigma'}$ is a Gaussian kernel of standard deviation σ'. In what follows, we always take $\sigma' = 1$. In practice, we use the same number of bins N_b for the object and the background and for all channels.

Notice that, as it is well known, when the number of bins N_b is too large, H_q is null for most $q \in \{0, \ldots, N_b - 1\}^c$. Such observation grows as the number of channels c increases. As a result, $\mathbb{P}(I_p \mid p \in \mathcal{B})$ is not accurately estimated and (the learned distribution law overfits the samples) most contracted capacities of the graph are set to 0. In practice, the model behaves as if we had $\beta = 0$. On the other hand, when N_b is too small, the best possible estimate approximates $\mathbb{P}(I_p \mid p \in \mathcal{B})$ by a piecewise constant function made of large square pieces. This time, H_q is not null for a larger part of $q \in \{0, \ldots, N_b - 1\}^c$ but $\mathbb{P}(I_p \mid p \in \mathcal{B})$ is roughly approximated. Therefore, the number of bins N_b should be a trade-off between these two situations. In practice, we adapt the number of bins to the number of channels. Following results of the Figure 6, we empirically choose a number of cells $N_b = 256$ and $N_b = 50$ for grayscale and color images, respectively. Smoothing distributions allows to further increase the number of cells where H_q is not null and can further reduce the size of the graph.

3) Numerical experiments on 2D, 2D+t and 3D images

For segmenting 2D, 2D+t and 3D grayscale/color images in connectivity 1, we compare the performance of standard graph cuts against reduced graph cuts in terms of speed, memory

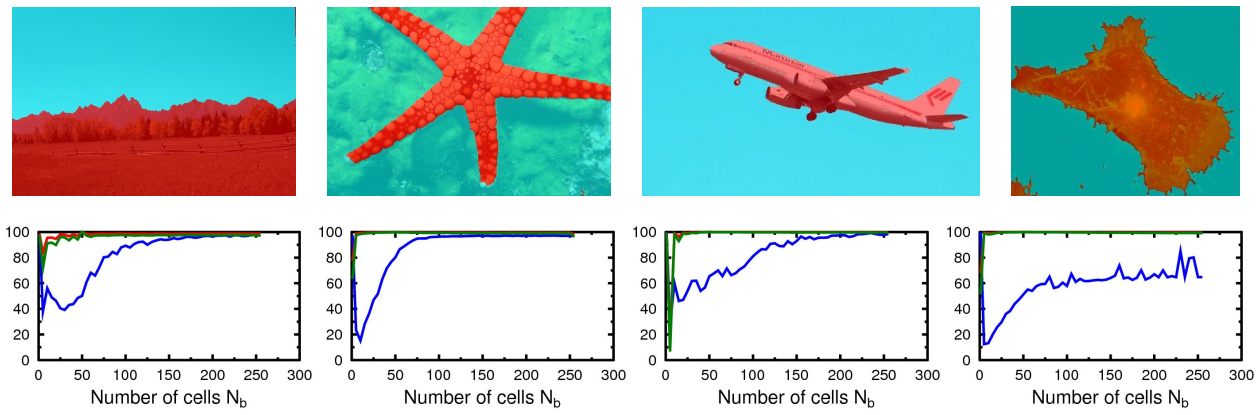


Fig. 6. Evolution of the relative size of the reduced graph (blue curve) and the distance between a ground truth image obtained from Table II with the number of bins N_b. The distance between both segmentations is measured with the DSC (red curve) and the SRMSSD (green curve). These measures are detailed in Appendix I. Seeds and parameters are the same as those used in Table II.

consumption and provide measures for assessing the differences between the segmentations obtained with standard graph cuts and reduced graph cuts.

Let us now describe our experimental procedure. For each image, the seeds and the model parameters are manually optimized for getting the best segmentation. Using these seeds and parameters, a reference segmentation is computed with standard graph cuts: the same seeds and parameters are then used for the reduced graph cuts. Also, we indicate the window radius r^* for which the relative size of the graph ρ^* is minimum. This is simply achieved by an exhaustive search for $r \in \{1, \ldots, 30\}$. The differences between the segmentations are estimated using two evaluation measures (DSC and VO, see Appendix I) as well as on the value of the flow in the graphs (see Appendix I). Notice that some 2D images comes from the Berkeley segmentation dataset [10].

The results obtained using a TV+L^2 model (see Section III-B) are summarized in Table I. Similarly, the results obtained using a Boykov-Jolly model (see Section III-C) are summarized in Table II. Segmentation results are also illustrated using a TV+L^2 and a Boykov-Jolly model in Figure 7 and 8, respectively.

Let us now analyze the time and memory consumption of our algorithm. For both models,

[10]The dataset is freely available at `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/`

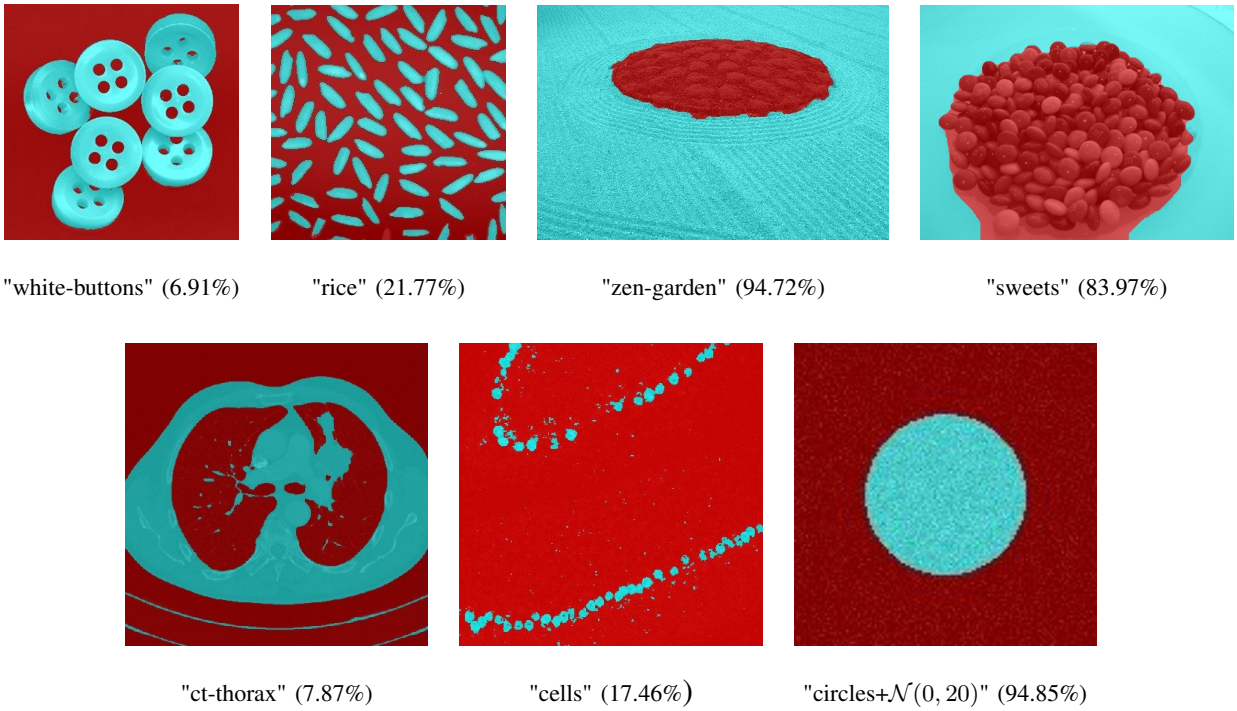


Fig. 7. Segmentation results using a TV+L^2 model in connectivity 1. The segmentation is superimposed on the original image by transparency. The minimal relative size of the reduced graph is indicated in parenthesis below each image.

we first observe that our algorithm mostly outperforms standard graph cuts in terms of memory while keeping a null (or extremely small) distance between segmentations. In that situation, the reduced graph cuts are also generally faster since the time used for testing each node is compensated by the graph reduction. However, notice that the difference in time is generally small and diminishes as r^* increases.

For some instances, the algorithm however does not reduce the underlying graphs significantly. This is for instance the case of the images "circles+$\mathcal{N}(0, 20)$", "zen-garden" or "cells" for which the relative reduced graph size is indeed large. This result reflects the fact that a lot of neighboring nodes are connected to opposite terminals in the graph. The density of nodes connected to the source s and the sink t is therefore correlated to the amount of noise in the image. Thus, an ideal situation consists of large areas of nodes linked to the same terminal and separated by smooth borders. Notice that when these areas contain a lot of nodes connected to wrong terminals, we can further gain memory by relaxing (11) with the parameter η (see (12)). When the graphs are not significantly reduced, we might also loose time compared to standard graph cuts. In that

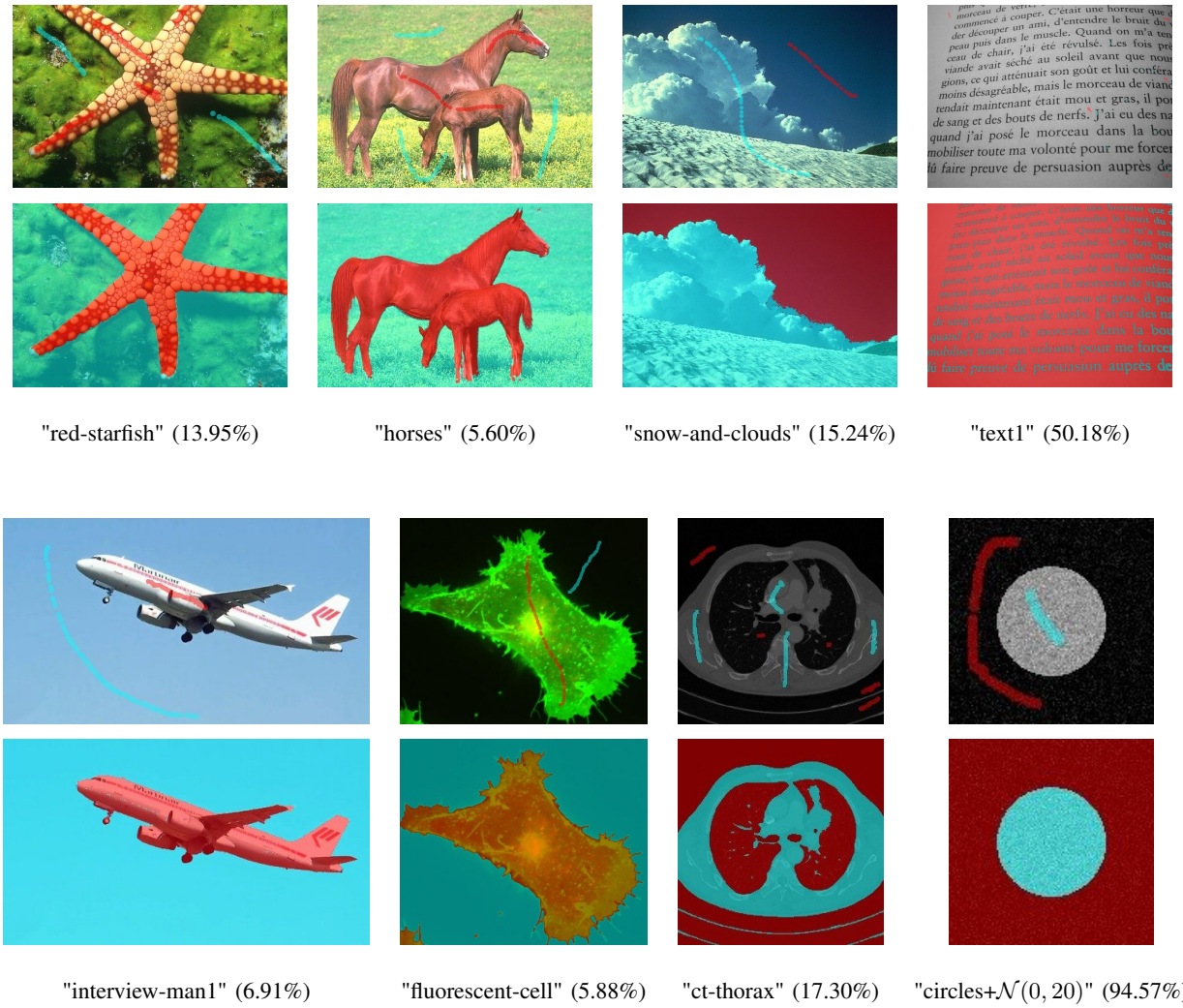


Fig. 8. Segmentation results using a Boykov-Jolly model in connectivity 1. The segmentation (bottom row) as well as seeds (top row) are superimposed on the original image by transparency. The minimal relative size of the reduced graph is indicated in parenthesis below each image.

case, the time of the reduction is not anymore balanced by the time for computing the max-flow in the reduced graph. However as illustrated in the previous experiments, this situation does not occur often since we generally have $r^* = 1$ for most of the presented images. We also believe that times could be substantially increased by adopting a better management of the borders in the reduction algorithm.

Let us now describe the results obtained in Tables I and II. For the TV+L^2 model, the average relative size of the reduced graph is 33.5% with a standard deviation of 34.01%. For 26 images out of 28, reduced graph cuts allocate less memory than standard graph cuts. Similarly, for 22

images out of 28, reduced graph cuts are faster than standard graph cuts. For some instances, the optimal window radius is far from being equal to one because the boundary of the segmentation is very rigid for avoiding undesired objects (see image "zen-garden" and "sweets" in Figure 7). This leads to a small value of β and therefore a large window radius r^* for lowering δ in order to obtain a smaller graph.

For the Boykov-Jolly model, the average relative size of the reduced graph is 19.44% with a standard deviation of 24.46%. For all images, reduced graph cuts allocate less memory than standard graph cuts. Similarly, for 30 images out of 31, reduced graph cuts are faster than standard graph cuts.

	Volume name	Size	Standard GC		Reduced GC		ρ^* (%)	r^*	RME	RSE	DSC (%)	VO (%)
			Time	Memory	Time	Memory						
2D	plane	481 ×321	0.09	43.44 Mb	0.17	27.88 Mb	49.89	14	0	0	100	100
	zen-garden	481 ×321	0.07	43.44 Mb	0.13	43.19 Mb	94.72	10	0	0	100	100
	oriental-man	321 ×481	0.19	43.44 Mb	0.24	43.19 Mb	79.85	13	0.0007	0	100	100
	ct-thorax-z	512 ×512	0.16	73.81 Mb	0.07	3.87 Mb	5.74	1	0	0	100	100
	book	3012 ×2048	2.56	1.70 Gb	1.74	148.75 Mb	8.64	1	0	0	100	100
	cells-z	512 ×512	0.18	73.81 Mb	0.18	65.95 Mb	76.39	6	0.0011	0	100	100
	beans	256 ×256	0.06	18.41 Mb	0.06	18.41 Mb	99.83	4	0.0001	0	100	100
	sweets	800 ×600	0.64	135.24 Mb	1.18	135.24 Mb	83.97	22	0	0	100	100
	text1	1600 ×1200	0.79	541.48 Mb	0.57	47.46 Mb	10.56	1	0	0	100	100
	text2	1024 ×768	0.33	221.67 Mb	0.31	66.11 Mb	28.78	1	0	0	100	100
	yeasts1	512 ×512	0.20	73.81 Mb	0.20	73.81 Mb	95.32	3	0.0016	0	100	100
	yeasts2	512 ×512	0.17	73.81 Mb	0.09	13.06 Mb	18.83	1	0.0001	0.007	99.9	99.9
	angiography1	512 ×512	0.18	73.81 Mb	0.08	5.80 Mb	7.97	1	0.0001	0	100	100
	angiography2	350 ×643	0.13	63.35 Mb	0.06	4.17 Mb	7.51	1	0	0	100	100
	f117	588 ×392	0.10	64.89 Mb	0.05	1.15 Mb	1.71	1	0	0	100	100
	black-cat	600 ×400	0.10	67.57 Mb	0.06	782.65 Kb	1.14	1	0	0	100	100
	viking-symbol2	660 ×740	0.33	137.61 Mb	0.22	66.11 Mb	37.06	3	0	0	100	100
	white-buttons	300 ×300	0.06	25.30 Mb	0.03	1.72 Mb	6.91	1	0	0	100	100
	rice	256 ×256	0.04	18.41 Mb	0.02	3.87 Mb	21.77	1	0.0002	0	100	100
	blood-cells	425 ×280	0.05	33.46 Mb	0.04	5.80 Mb	18.84	1	0	0	100	100
	poppy	409 ×613	0.16	70.59 Mb	0.06	1.85 Mb	2.93	1	0.0002	0	100	100
2D+t	interview-girl	320 ×240 ×150	17.91	9.27 Gb	6.64	1.48 Gb	15.04	1	0	0	100	100
	interview-old-man	256 ×256 ×128	17.08	6.74 Gb	6.38	1.48 Gb	18.83	1	0.0017	0	100	100
	interview-woman	352 ×288 ×154	21.14	12.57 Gb	8.35	2.21 Gb	13.36	1	0	0	100	100
3D	ct-thorax	409 ×409 ×252	79.91	34.03 Gb	18.30	2.27 Gb	7.87	1	0.8439	0.001	100	100
	circles-pyramid+sigma20	128 ×128 ×128	5.06	1.68 Gb	4.55	1.67 Gb	94.85	1	0.0138	0	100	100
	cells	409 ×409 ×101	40.49	13.59 Gb	10.66	2.21 Gb	17.46	1	0.0006	0	100	100
	brain-p3	181 ×217 ×181	15.12	5.71 Gb	3.96	671.62 Mb	12.63	1	0.0339	0	100	100

TABLE I

STANDARD GRAPH CUTS (STANDARD GC) ARE COMPARED TO OUR ALGORITHM (REDUCED GC) IN TERMS OF SPEED (IN SECS) AND MEMORY FOR SEGMENTING GRAYSCALE IMAGES USING A TV+L^2 MODEL IN CONNECTIVITY 1.

	Volume name	Size	Standard GC		Reduced GC		ρ^* (%)	r^*	RME	RSE	DSC (%)	VO (%)
			Time	Memory	Time	Memory						
2D	eagle-c	481 ×321	0.24	43.44 Mb	0.07	2.58 Mb	5.54	1	0	0	100	100
	zen-garden-c	481 ×321	0.26	43.44 Mb	0.27	43.19 Mb	90.75	1	0	0	100	100
	columns-c	481 ×321	0.20	43.44 Mb	0.06	521.76 Kb	1.17	1	0	0	100	100
	red-flowers-c	481 ×321	0.25	43.44 Mb	0.12	13.06 Mb	23.30	1	0	0	100	100
	snow-and-clouds-c	481 ×321	0.23	43.44 Mb	0.10	8.71 Mb	15.24	1	0	0	100	100
	marker-c	481 ×321	0.23	43.44 Mb	0.07	1.15 Mb	2.46	1	0	0	100	100
	pyramid-c	481 ×321	0.23	43.44 Mb	0.06	561.84 Kb	1.46	1	0	0	100	100
	red-starfish-c	481 ×321	0.24	43.44 Mb	0.09	5.80 Mb	13.95	1	0	0	100	100
	black-cow-c	481 ×321	0.24	43.44 Mb	0.07	561.84 Kb	1.51	1	0	0	100	100
	church2-c	481 ×321	0.23	43.44 Mb	0.07	2.58 Mb	5.07	1	0	0	100	100
	snake2-c	481 ×321	0.25	43.44 Mb	0.07	1.85 Mb	4.95	1	0	0	100	100
	birds2-c	321 ×481	0.28	43.44 Mb	0.08	2.58 Mb	6.40	1	0.0001	0	100	100
	eagle2-c	481 ×321	0.23	43.44 Mb	0.06	1.15 Mb	2.25	1	0	0	100	100
	greek-temple-c	481 ×321	0.23	43.44 Mb	0.08	3.87 Mb	8.03	1	0	0	100	100
	horses4-c	481 ×321	0.23	43.44 Mb	0.07	2.58 Mb	5.60	1	0	0	100	100
	meadow-and-mountains-c	481 ×321	0.25	43.44 Mb	0.18	29.38 Mb	56.00	1	0	0	100	100
	traditional-houses-c	481 ×321	0.23	43.44 Mb	0.07	1.72 Mb	4.30	1	0	0	100	100
	ct-thorax-z	512 ×512	0.43	73.81 Mb	0.13	8.71 Mb	10.85	1	0	0.03	99.78	99.5
	book	3012 ×2048	8.49	1.70 Gb	2.97	148.75 Mb	8.18	1	0	0	100	100
	cells-z	512 ×512	0.45	73.81 Mb	0.29	44.07 Mb	48.91	2	0.0007	0	100	100
	text1	1600 ×1200	2.90	541.48 Mb	2.02	334.68 Mb	50.18	1	0	0	100	100
	viking-symbol2	660 ×740	0.85	137.61 Mb	0.21	6.25 Mb	5.13	1	0.0023	0	99.99	99.9
2D+t	interview-man1-c	320 ×240 ×203	63.03	12.56 Gb	10.77	1007.43 Mb	6.91	1	0	0	100	100
	interview-man2-c	426 ×240 ×180	73.66	14.83 Gb	11.20	447.75 Mb	3.21	1	0	0	100	100
	plane-take-off-c	492 ×276 ×180	99.16	19.70 Gb	15.85	1.01 Gb	6.20	1	0	0	100	100
	talk-c	370 ×276 ×190	78.81	15.64 Gb	21.44	2.21 Gb	15.44	1	0	0	100	100
	fluorescent-cell-c	478 ×396 ×121	97.79	18.44 Gb	18.68	1007.43 Mb	5.88	1	0	0	100	100
3D	ct-thorax	245 ×245 ×151	42.90	7.29 Gb	16.73	2.21 Gb	23.88	1	9.9501	0.01	99.94	99.8
	circles-pyramid+sigma20	128 ×128 ×128	10.52	1.68 Gb	9.26	1.48 Gb	87.23	1	0.0004	0	100	100
	cells	230 ×230 ×57	13.72	2.41 Gb	8.32	1.48 Gb	58.38	1	0.0055	0	100	100
	brain-p3	181 ×217 ×181	38.37	5.71 Gb	10.90	1.48 Gb	24.38	1	0.0186	0	100	100

TABLE II

STANDARD GRAPH CUTS (STANDARD GC) ARE COMPARED TO OUR ALGORITHM (REDUCED GC) IN TERMS OF SPEED (SECS) AND MEMORY FOR SEGMENTING GRAYSCALE / COLOR IMAGES USING A BOYKOV-JOLLY MODEL IN CONNECTIVITY 1. COLOR IMAGES ARE SUFFIXED BY "C" IN THEIR NAMES.

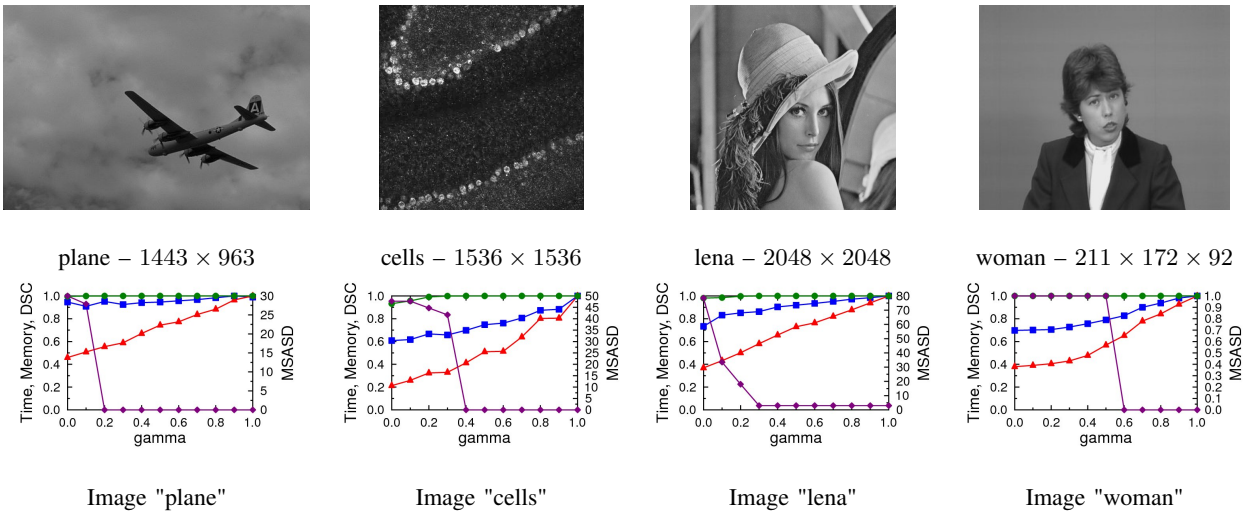


Fig. 9. Influence of the parameter γ (bottom row) for segmenting 2D and 2D+t images (top row) with a TV+L^2 model in connectivity 1. On the bottom row, blue curve with squares and red curve with triangles correspond respectively to the gain in time and the amount of memory allocated for the reduced graph. Green curves with circles and purple curves with diamonds correspond respectively to the DSC and the MSASD between γ-parametrized segmentations and the segmentations obtained with standard graph cuts. The DSC and MSASD measures are detailed in Appendix I.

4) The parameter γ

The Figure 9 illustrates the role of the parameter γ using the same model, images and parameters as in Figure 5. This experiment shows how far the condition (11) can be relaxed while nearly having an exact solution. In Figure 9, the window radii are chosen to minimize the memory consumption. The differences between the segmentations with the standard graph cuts and the reduced graph cuts are estimated using two evaluation measures: the DSC and the MSASD (see Appendix I). Then, we display the DSC (green curve), the MSASD (purple curve) as well and the execution time (blue curve) and the memory consumption (red curve) over a fixed range of γ values from 0 to 1. As γ decreases to 0, we naturally observe that we get a coarser approximation of the solution. In practice, we obtain nearly exact solutions for $\gamma \geq 0.5$. For $\gamma < 0.4$, the solution is slightly different but remains close from the original segmentation.

5) The parameter η

A lower bound for η

For a fixed window radius, notice first that the value of η must be sufficiently large for keeping the graph in a whole piece (see Figure 10). Below some value (denoted by η_{min}), the reduced

graph can be unfortunately split into several connected components and pixels can be wrongly labeled in the segmentation. A lower bound of such a value can be easily be computed by considering an image consisting of two contrasted areas separated by a vertical edge. An easy estimate of η_{min} is to impose that η_{min} permits to segment these two contrasted areas. In order to do so, we want the test (12) to be false for any pixel p located on the boundary between these areas. For such a pixel, we have (e.g. if we assume $c(p) \geq +\delta\gamma$):

$$\sharp\{q \in B_p \mid c(q) \geq +\delta\gamma\} = (r+1)(2r+1)^{d-1}.$$

As a consequence, if

$$\eta \leq \frac{(r+1)(2r+1)^{d-1}}{(2r+1)^d},$$

the pixel p does not belong to the reduced graph $\mathcal{G}'$. Since we want to avoid the situation, we therefore must have:

$$\begin{array}{rcl} \eta & > & \frac{(r+1)(2r+1)^{d-1}}{(2r+1)^d} \\ & = & 1 - \frac{r}{2r+1} = \eta_{min}. \end{array} \tag{15}$$

Thus, as r tends to infinity, the maximum proportion of nodes allowed being connected to opposite terminals tends to 50%. Notice that this lower bound no longer holds in connectivity 0. Indeed, the lower bound can be too small in areas with high-curvature and the reduced graph would be disconnected into multiple pieces (see Figure 10). Consequently, the min-cut is no longer ensured of being fully contained in $\mathcal{G}'$.

Further reducing the graphs using η

We now detail how the parameter η can be used for reducing the memory usage. The Figure 11 illustrates how far the condition (11) can be relaxed for further reducing graphs while getting nearly the same segmentation. In this experiment, the segmentation and the reduced graph are shown for segmenting a synthetic noisy 2D image with a Boykov-Jolly model using connectivity 1. Since the condition (12) becomes easier to satisfy when η decreases, the graph around the object contours becomes thicker.

Denoising using η

The parameter η can be also used for denoising the segmentation. Indeed, it can be tuned to remove small regions of the segmentation and therefore denoise it. This behavior is illustrated in Figure 12 for segmenting a 3D noisy image from a confocal microscope with a Boykov-Jolly

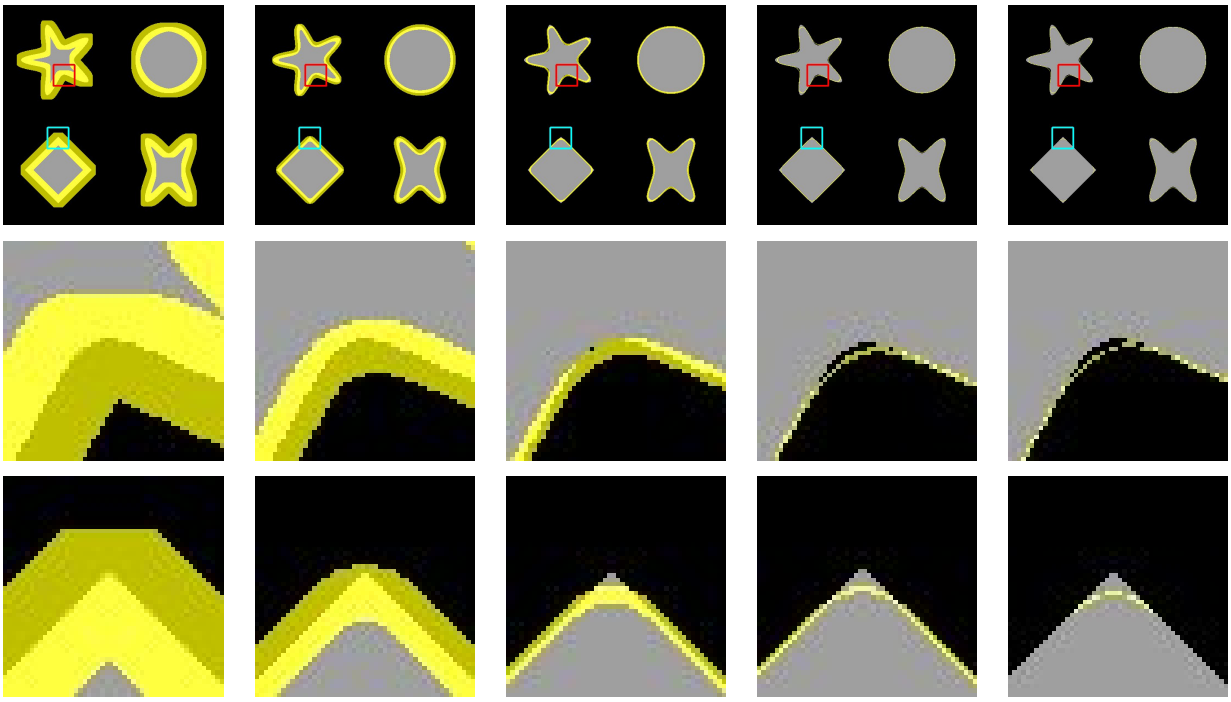
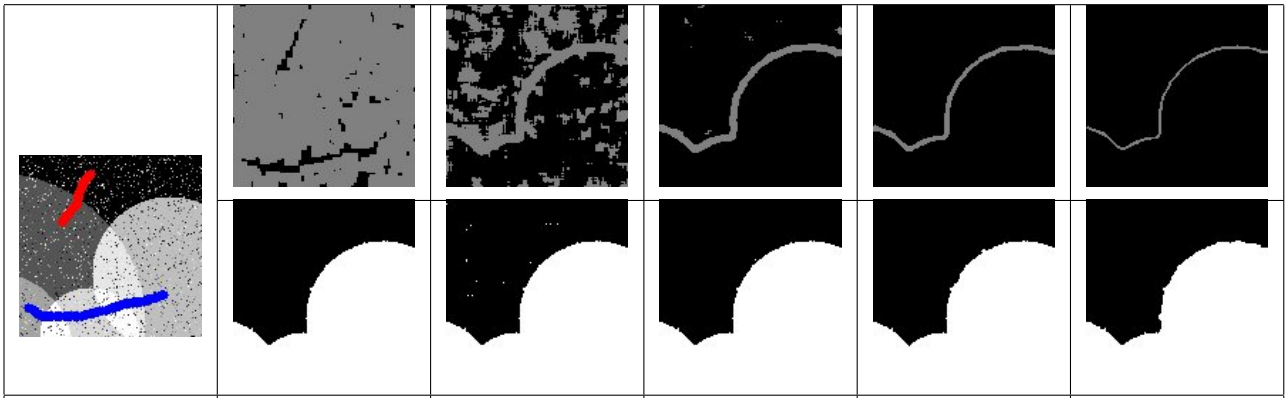
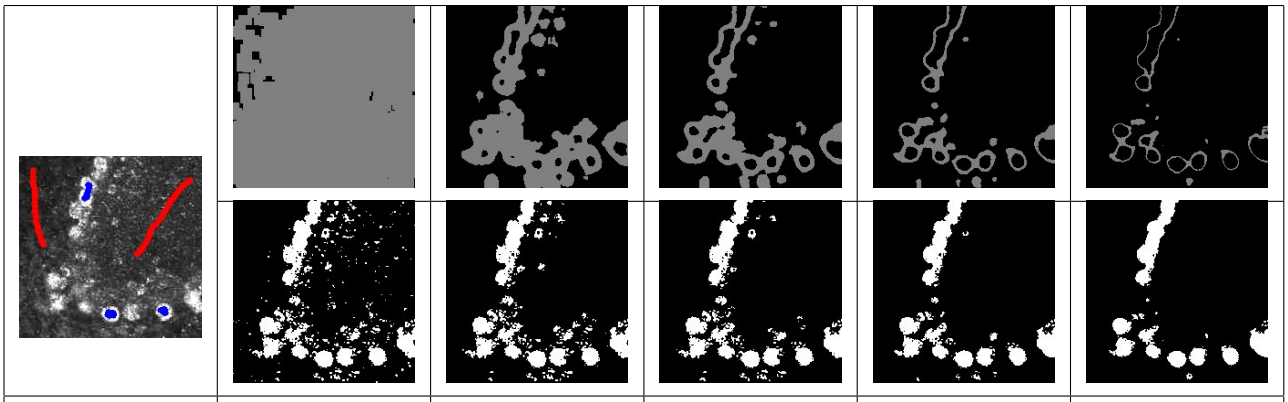


Fig. 10. Illustration of the lower bound η_{min} for segmenting a 2D synthetic image using a TV+L^2 model. In this experiment, $\eta_{min} \simeq 0.523$ and we set $r = 10$ using connectivity 1. On all images, the pixels belonging to $\mathcal{V}'$ are superimposed in yellow to the original image by transparency. The middle and the bottom rows correspond respectively to close-ups of the red and cyan areas. Observe how the reduced graph split into multiple pieces as soon as $\eta \leq \eta_{min}$.



η	1.0	0.9	0.8	0.7	0.6
ρ	93.28%	30.99%	5.74%	3.65%	2.00%

Fig. 11. Memory gain when segmenting a 2D synthetic image corrupted by 10% of impulsive noise, using a Boykov-Jolly model (left). Top row shows the nodes of the reduced graph in light gray while bottom row shows the corresponding segmentation. In this experiment, we set $r = 3$, $\gamma = 1$ and use connectivity 1. In this experiment, $\eta_{min} \simeq 0.571$.



η	1.0	0.9	0.8	0.7	0.6
ρ	55.70%	37.15%	18.26%	12.65%	8.87%

Fig. 12. Simultaneous segmentation and denoising of a 3D image using a Boykov-Jolly model (left) in connectivity 1. In this picture, white spots correspond to cell nuclei in a mouse cerebellum. Top row shows the nodes of the reduced graph in light gray while bottom row shows the corresponding segmentation. In this experiment, we set $r = 5$ and $\gamma = 1$.

model. In this picture, white spots correspond to cell nuclei in a mouse cerebellum. Observe how the denoising acts for small values of η: small regions in the graph and in the segmentation are progressively removed as η decreases. Typically, this parameter is useful for denoising images corrupted by a noise behaving like an impulsive noise. Finally, unlike traditional filters, our method does not require any pre or post-processing steps.

V. Conclusion

In this paper, we have presented a new band-based method for reducing the graphs involved in binary graph-cut optimization. We empirically proved on comprehensive experiments that the minimizer found is very near from the global one. When the unary terms in the energy are large, this reduction strategy clearly outperforms standard graph cuts in terms of memory. Thanks to a fast implementation, the algorithm runs also faster in that situation. When the graphs are not significantly reduced, an extra parameter is proposed to both further reduce them and remove segments due to noise.

An interesting perspective to this work would be to predict if a graph can fit in memory. Another point could be also to predict the window radius r^* for which the reduced graph size is minimum (ρ^*). Knowing r^* and ρ^* could be for instance useful to decide if the reduction must be applied or not to minimize the running time. Then, it would not be difficult to apply reduction

when ρ^* is greater than some threshold (probably around 75% according to the experiments presented in Section IV). In fact, answering to the second question allow to answer to the first one: if we are able to find r^*, we are able to predict if the graph can fit in memory or not. However, computing r^* is probably too expensive since we must run several times the algorithm until r^* is found. We therefore believe that trying to find a reliable estimate of r^* is a better choice. This could be for instance achieved by randomly selecting a few pixels in the image, computing their best window radius (by progressively increasing r) and averaging these values to get an approximate value of r^*.

Another perspective is to extend the proposed reduction strategy in the multi-labels setting (i.e. when the variables in the energy to minimize take values in more two labels) and analyze the memory gains on different problems.

REFERENCES

[1] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[2] D. M. Greig, B. T. Porteous, and A. H. Seheult, "Exact maximum a posteriori estimation for binary images," *Journal of the Royal Statistical Society*, vol. 51, no. 2, pp. 271–279, 1989.

[3] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," in *ICCV*, vol. 1, 1999, pp. 377–384.

[4] H. Lombaert, Y. Sun, L. Grady, and C. Xu, "A multilevel banded graph cuts method for fast image segmentation," in *ICCV*, vol. 1, 2005, pp. 259–265.

[5] A. Sinop and L. Grady, "Accurate banded graph cut segmentation of thin structures using laplacian pyramids," in *MICCAI*, vol. 9, no. 2, 2006, pp. 896–903.

[6] P. Kohli, V. Lempitsky, and C. Rother, "Uncertainty driven multi-scale energy optimization," in *DAGM*, 2010, pp. 242–251.

[7] Y. Li, J. Sun, C. Tang, and H. Shum, "Lazy Snapping," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 303–308, 2004.

[8] J. Stawiaski, E. Decencière, and F. Bidault, "Computing approximate geodesics and minimal surfaces using watershed and graph-cuts," in *ISMM*, 2007, pp. 349–360.

[9] C. Cigla and A. Alatan, "Region-based image segmentation via graph cuts," in *ICIP*, 2008, pp. 2272–2275.

[10] A. Delong and Y. Boykov, "A scalable graph-cut algorithm for N-D grids," in *CVPR*, 2008, pp. 1–8.

[11] P. Strandmark and F. Kahl, "Parallel and distributed graph cuts by dual decomposition," in *CVPR*, 2010, pp. 2085–2092.

[12] B. Scheuermann and B. Rosenhahn, "Slimcuts: Graphcuts for high resolution images using graph reduction," in *EMMCVPR*, July 2011.

[13] V. Lempitsky and Y. Boykov, "Global optimization for shape fitting," in *CVPR*, 2007, pp. 1–8.

[14] N. Lermé, F. Malgouyres, and L. Létocart, "Reducing graphs in graph cut segmentation," in *ICIP*, 2010, pp. 3045–3048.

[15] ——, "Reduction of "vision graphs"," Patent No. FR2955408 (A1), January 2010.

[16] N. Lermé, F. Malgouyres, and J.-M. Rocchisani, "Fast and memory efficient segmentation of lung tumors using graph cuts," in *MICCAI – Workshop on Pulmonary Image Analysis*, 2010, pp. 9–20.

[17] Y. Boykov and M.-P. Jolly, “Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images,” in *ICCV*, vol. 1, 2001, pp. 105–112.

[18] P. Kohli and P. H. S. Torr, “Dynamic graph cuts for efficient inference in markov random fields,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 12, pp. 2079–2088, 2007.

[19] V. Kolmogorov and R. Zabih, “What energy functions can be minimized via graph cuts?” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147–159, 2004.

[20] L. I. Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms,” *Physica D*, vol. 60, pp. 259–268, 1992.

[21] F. Ranchin, A. Chambolle, and F. Dibos, “Total variation and graph cuts approaches for variational segmentation,” in *Proceedings of SSVM*, June 2007, pp. 743–753.

[22] J. Darbon and M. Sigelle, “Exact optimization of discrete constrained total variation minimization problems,” in *IWCIA*, vol. 3322, 2004, pp. 548–557.

[23] F. Malgouyres and N. Lermé, “A non-heuristic reduction method for graph cut optimization,” 2012, submitted.

APPENDIX I
EVALUATION AND REDUCTION MEASURES

Nicolas Lermé received the B.Sc. degree in Computer Science from the University Paris 13 in 2006 and the M.Sc. degree from the Université de Marne-la-Vallée in 2008 in the same domain. He is currently working as a postdoctoral fellow at the Institut Supérieur d’Electronique de Paris. His research interests primarily focus on developing efficient and robust methods in discrete optimization for solving computer vision tasks with an emphasis on "graph algorithmics".

François Malgouyres Francois Malgouyres received his PhD in applied Mathematics from the Ecole Normale Supérieure de Cachan in 2000. After one year at the Mathematics department of the University of California at Los Angeles, he has been appointed by the University Paris 13 as an assistant professor in Mathematics, Computer science and image processing, from 2001 to 2011. He is now full Professor at the Institut de Mathématiques de Toulouse, at the University Paul Sabatier.

Let $SG, GT \subset \{0,1\}^N$ ($N > 0$) denote respectively a machine-obtained segmentation and the ground truth. The function $\partial S : \{0,1\}^N \rightarrow \{0,1\}^N$ will correspond to the border of any set $S \subset \{0,1\}^N$ which is defined as $\partial S = \{p \in S | \ \exists (p,q) \in \mathcal{N}, q \notin S\}$. We also remind that f^* (resp. f'^*) denote the max-flow value in the non-reduced graph $\mathcal{G}$ (resp. in the reduced graph $\mathcal{G}'$).

- **Dice Similarity Coefficient (DSC):**

$$DSC(SG, GT) = 2 \cdot \frac{\sharp(SG \bigcap GT)}{\sharp SG + \sharp GT} \times 100 \in [0, 100].$$

- **Volumetric Overlap (VO):**

$$VO(SG, GT) = \frac{\sharp(SG \bigcap GT)}{\sharp(SG \bigcup GT)} \times 100 \in [0, 100].$$

- **Symmetric RMS Surface Distance (SRMSSD):**

$$SRMSSD(SG, GT) = \sqrt{\left(\frac{\sum_{p \in \partial SG} \min_{q \in \partial GT} d(p,q)^2 + \sum_{q \in \partial GT} \min_{p \in \partial SG} d(p,q)^2}{\sharp \partial SG + \sharp \partial GT} \right)}.$$

- **Maximum Symmetric Absolute Surface Distance (MSASD or Hausdorff distance):**

$$MSASD(SG, GT) = \max\{ \max_{p \in \partial SG} \min_{q \in \partial GT} d(p,q), \max_{q \in \partial GT} \min_{p \in \partial SG} d(p,q) \}.$$

- **Relative Max-flow Error (RME):**

$$RME(\mathcal{G}, \mathcal{G}') = \frac{(f^* - f'^*)}{f^*} \times 100 \in [0, 100].$$