REDUCING GRAPHS IN GRAPH CUT SEGMENTATION

Nicolas Lermé^{1,2}, *François Malgouyres*^{1}, *Lucas Létocart*^{2}

(1) LAGA UMR CNRS 7539,   (2) LIPN UMR CNRS 7030
Université Paris 13 –Avenue J.B. Clément
93430 Villetaneuse - France

{nicolas.lerme,lucas.letocart}@lipn.univ-paris13.fr
malgouy@math.univ-paris13.fr

ABSTRACT

In few years, graph cuts have become a leading method for solving a wide range of problems in computer vision. However, graph cuts involve the construction of huge graphs which sometimes do not fit in memory. Currently, most of the max-flow algorithms are impracticable to solve such large scale problems. In the image segmentation context, some authors have proposed heuristics [1, 2, 3, 4] to get round this problem. In this paper, we introduce a new strategy for reducing graphs. During the creation of the graph, before creating a new node, we test if the node is really useful to the max-flow computation. The nodes of the reduced graph are typically located in a narrow band surrounding the object edges. Empirically, solutions obtained on the reduced graphs are identical to the solutions on the complete graphs. A parameter of the algorithm can be tuned to obtain smaller graphs when an exact solution is not needed. The test is quickly computed and the time required by the test is often compensated by the time that would be needed to create the removed nodes and the additional time required by the computation of the cut on the larger graph. As a consequence, we sometimes even save time on small scale problems.

Index Terms— segmentation, graph cut, reduction.

1. INTRODUCTION

Graph cuts provide a global optimization method based on max-flow/min-cut for solving a wide range of problems encountered in computer vision. Since pioneer work of Greig *et al.* [5], the graph cuts have recently known a quick development with the arrival of a fast max-flow algorithm [6].

At the same time, the resolution of images acquired by digital devices increase constantly. In biomedical imaging, high-resolution data can involve massive graphs containing billion of nodes, which do not fit in memory. For these instances, global optimization methods such as graph cuts are impractical due to memory requirements.

To overcome this problem, Delong and Boykov [7] have recently published a new parallelized max-flow algorithm yielding near-linear speedup with the number of processors. This algorithm is able to segment large volumes while keeping optimality on solutions but remains less effective than standard graph cuts on small graphs. On the other side, some authors have also proposed heuristics based on multi-resolution schemes [3, 2]. The principle is to generate a graph in a narrow band constrained from the segmentation result for a subsampled image. These algorithms reduce drastically speed and memory usage but fail to recover thin structures in images. This drawback is reduced but remains true in [2] for images with low contrast. In medical imaging, this is a real drawback since thin structures like blood vessels are ubiquitous. Other heuristics [1, 4] use adjacency graphs. The idea is to pre-segment the image thanks to a low-level algorithm (e.g watershed [1] or mean shift [4]) and build an adjacency graph where each node corresponds to a pre-segmented region. Results highly depend both on the image structure and the low-level segmentation algorithm. By drastically reducing the number of nodes in the graph, these heuristics greatly increase the speed of graph cuts and reduce the memory usage. Nevertheless, the performances are better when over-segmentation occurs, i.e. when the size of the adjacency graph is equivalent to the size of the graph in standard graph cuts.

In the present work, we propose an algorithm for reducing graphs. The idea is to gradually build the graph by only adding nodes which satisfy a condition in a small window. The rest of the paper is organized as follows. In section 2, we review the graph cuts framework. Next, our approach is detailed in section 3 and compared to standard graph cuts in section 4.

2. BACKGROUND

Let us briefly summarize the work of Boykov and Jolly for N-D image segmentation. An N-D image can be defined by a pair $(\mathcal{P}, I)$ consisting of a finite discrete set $\mathcal{P} \subset \mathbb{Z}^d$ $(d > 0)$ of N-D points (pixels in $\mathbb{Z}^2$, voxels in $\mathbb{Z}^3$, etc.) and a function I that maps each point $p \in \mathcal{P}$ to a value $I(p)$ in some value space. For an image, we can construct the associated directed weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, c)$ consisting of a set of nodes $\mathcal{V} = \mathcal{P} \cup \{s, t\}$, a set of edges $\mathcal{E}$ and a positive weighting function $c : \mathcal{V}^2 \to \mathbb{R}^+$ defining the edge capacity.

We distinguish two special nodes of $\mathcal{V}$: the source node s specifying the « object » terminal and the sink node t specifying the « background » terminal. Furthermore, we split the set of edges $\mathcal{E}$ in two disjoint sets $\mathcal{E}_n$ and $\mathcal{E}_t$ denoting respectively n-links (neighborhood links) and t-links (terminal links). Next, we associate a neighborhood $\mathcal{N}(p)$ to any point $p \in \mathcal{P}$. In this setting, we will use the following neighborhoods :

$$\begin{aligned} \mathcal{N}_0(p) &= \{q : \textstyle\sum_{i=1}^{d} |q_i - p_i| = 1\} & \forall p \in \mathcal{P}, \\ \mathcal{N}_1(p) &= \{q : |q_i - p_i| \le 1 \ \forall 1 \le i \le d\} & \forall p \in \mathcal{P}, \end{aligned}$$

where p_i denote the i$^{\text{th}}$ coordinate of the point p. For instance, each pixel has 4 and 8 neighbors in 2D, 6 and 26 neighbors in 3D and finally 8 and 80 neighbors in 4D [1]. In the sequel, the terms « connectivity 0 » and « connectivity 1 » will correspond respectively to the use of a $\mathcal{N}_0$ and $\mathcal{N}_1$ neighborhood.

In [8], Boykov and Jolly showed that the image segmentation problem can be efficiently solved by minimizing a Markov Random Field of the form :

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

where $u \in \{0,1\}^N$. As usual, the data fidelity term $E_p(.)$ forces u_p to fit the input data while the smoothness term $E_{p,q}(.)$ penalize neighboring pixels p and q if they have different labels. According to [9], the minimizer of the energy (1) corresponds to a min-cut in a graph and can be efficiently computed by the algorithm described in [6] [2].

3. REDUCING GRAPHS

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 Boykov and Kolmogorov [6] (version 2.2) allocates $24|\mathcal{P}| + 14|\mathcal{E}_n|$ bytes. In table 1, we observe that for a fixed amount of RAM, the maximum volume size decreases quickly as dimension d increases. Nevertheless, we observe on figure 1 that most of the nodes are useless because not traversed by any flow. On the right hand-side of figure 1, we represent the flow passing through the t-links. Light gray pixels (respectively dark gray pixels) indicates that a strictly positive amount of flow is passed from s to node p (respectively from node p to t). Clearly, only a small part of nodes is used during the max-flow computation. When reducing such a graph, one would like extract the smallest possible graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}', c)$ from $\mathcal{G}$ while keeping a solution u' identical or very close to u. Ideally, we want to maximize the reduction rate $\rho = 1 - \frac{|\mathcal{V}'|}{|\mathcal{V}|}$ s.t. $u \simeq u'$. However, the method for determining $\mathcal{G}'$ also needs to be fast and this rules out the resolution of such an optimization problem. Before describing our method for building $\mathcal{G}'$, let us introduce some terminology. In accordance with the graph construction given in [9], we consider (without loss of generality) that a node is linked to at most one terminal, i.e :

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

	Connectivity 0	Connectivity 1
2D	6426	4459
3D	319	219
4D	68	45

Table 1: Maximum values of image size in function of d and connectivity with a fixed amount of RAM of 2GB.

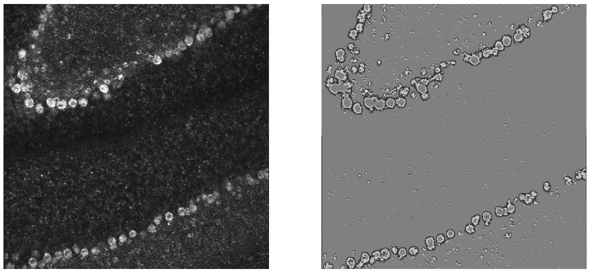


Fig. 1: Illustration of flow passing through t-links (right) for segmenting a 2D image (left).

Also, we summarize the capacities of the t-links at any node $p \in \mathcal{P}$ by $c(p) = c(s,p) - c(p,t)$. For any $B \subset \mathbb{Z}^d$ and $x \in \mathcal{P}$, we denote by $\widetilde{B}_x$ the set translation of B by the point $x : \widetilde{B}_x = \{b + x \mid b \in B\}$. Moreover, for $Z \subset \mathcal{P}$ and $B \subset \mathbb{Z}^d$, we define the dilation of Z by B as :

$$\widetilde{Z}_B = \{z + b \mid b \in B, z \in Z\} = \bigcup_{z \in Z} \widetilde{B}_z.$$

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

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

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

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

where $\operatorname{sign}(t) = 1$ if $t > 0$, 0 if $t = 0$ and -1 otherwise.

[1] Typically, larger neighborhood systems yield better results but increase running times and memory consumptions.

[2] An implementation of the max-flow algorithm is freely available at `http://www.cs.cornell.edu/People/vnk/software.html`

Let $B \subset \mathbb{Z}^d$, in order to build $\mathcal{G}'$, we remove from the nodes of $\mathcal{G}$ any $Z \subset \mathcal{P}$ such that either

$$\begin{array}{l} O(\widetilde{Z}_B) = +|\widetilde{Z}_B| \text{ and } A(\widetilde{Z}_B \setminus Z) \geq P_{out}(\widetilde{Z}_B), \text{ or} \\ O(\widetilde{Z}_B) = -|\widetilde{Z}_B| \text{ and } A(\widetilde{Z}_B \setminus Z) \geq P_{in}(\widetilde{Z}_B). \end{array} \tag{2}$$

As an illustration of those conditions, notice for instance that the last condition implies that all the flow that might come in the region $\widetilde{Z}_B$ comes from its boundary and can be absorbed by the band $\widetilde{Z}_B \setminus Z$. Building such sets Z is done by testing each individual pixels of Z. In order to do so, we establish (in a forthcoming paper) that the conjonction of conditions (2) for every $z \in Z$ implies (2) for Z. Considering B, a square window of size $(2r+1)$ $(r > 0)$ centered at the origin, a more conservative test for $z \in Z$ is

$$\left\{ \begin{array}{lll} c(q) \geq +\delta \cdot \gamma & \forall q \in \widetilde{B}_z & \text{or} \\ c(q) \leq -\delta \cdot \gamma & \forall q \in \widetilde{B}_z, & \end{array} \right. \tag{3}$$

where $\gamma \in [0,1]$ and $\delta = \frac{P(B)}{(2r+1)^2-1}$, with

$$\begin{array}{l} P(B) = \max(|\{(p,q), p \in Z, q \notin Z \text{ and } p \in \mathcal{N}(q)\}|, \\ \qquad\qquad |\{(p,q), p \in Z, q \notin Z \text{ and } q \in \mathcal{N}(p)\}|). \end{array}$$

If all the capacities of the n-links are smaller than 1 (which is true for most interesting energies) and (3) holds, the inegality (2) holds for $Z = \{z\}$. Then, $\mathcal{G}'$ is determined by the set of nodes $\mathcal{V}' = \{p \in \mathcal{P} \text{ not satisfying (3)}\} \cup \{s, t\}$. We have theoretical and empirical evidence suggesting that this graph reduction provides an exact solution when $\gamma = 1$. It becomes an heurisitic as γ decreases to 0. Morever, the condition (3) is simple and a straightforward implementation has a worst-case complexity of $O(|B|)$. Decomposing this test along the d dimensions yields an algorithm with complexity $O(1)$, except for image borders.

4. EXPERIMENTAL RESULTS

This section compares the performance of standard graph cuts and our method in terms of speed, memory and segmentation accuracy with two energy models : $TV + L^2$ [10] and Boykov/Jolly [8]. Experiments are performed on an Athlon Dual Core 6000+ 3GHz with 2GB RAM for segmenting 2D/ 3D images in connectivity 1. Times are averaged over 10 runs.

4.1. $TV + L^2$ energy model

Total variation has originally been introduced by Rudin *et al.* for image denoising. The authors of [10] have proposed to use the $TV + L^2$ model for image segmentation.

First, the middle row of figure 2 describes the influence of the window parameter r. For $r = 0$, time and memory usage correspond to standard graph cuts. The general trend is that the amount of allocated memory first decreases and then increases as r increases. This corresponds to the fact that in (3), each individual test $|c(q)| \geq \delta$ is easier to satisfy when r is large, because δ decreases with r. However, the test on the signs are more difficult to satisfy since the window is larger. Note that the value of r which minimizes the memory usage depends both on the image structure and the model's parameters. Except for the image « plane », our algorithm is generally faster than standard graph cuts.

Second, figure 2 also illustrates the role of the γ parameter. The window radius is chosen to minimize both normalized time and memory usage. The differences between the reference and the segmentation are evaluated using the Dice Similarity Coefficient (DSC) and the Hausdorff distance [3]. For all images, the memory usage can be significantly reduced by lowering the γ parameter while getting nearly the same solution up to a certain value.

4.2. Boykov and Jolly's energy model

Introduced in [8], this model has quickly become a standard in applications. From a user viewpoint, it consists of marking some parts of the image as « object » and « background ». For more information, we refer the reader to [8].

Figure 3 compares time and memory usage between standard graph cuts and our method for segmenting real images. The second image represents a simulated brain MRI generated by Brainweb with 3% of noise [4] while the third image shows an abdominal CT with a pulmonary tumor.

In these experiments, the model's parameters are optimized for better visualization while γ parameter is set to 1. The window radius is chosen such that memory usage is minimized. Seeds were placed by hand but are not represented here because space limitations. For all images, the amount of allocated memory for the graph is reduced by a factor ranging from 4.8x to 7.7x. For the first image, our algorithm is 1.7x faster and require 4.8x less memory while getting exactly the same result. Moreover, altough the graphs induced by the volumes « brain » and « ct-thorax » do not fit in memory when no reduction is performed, we observe that our algorithm is able to segment them in less than 10 seconds.

5. REFERENCES

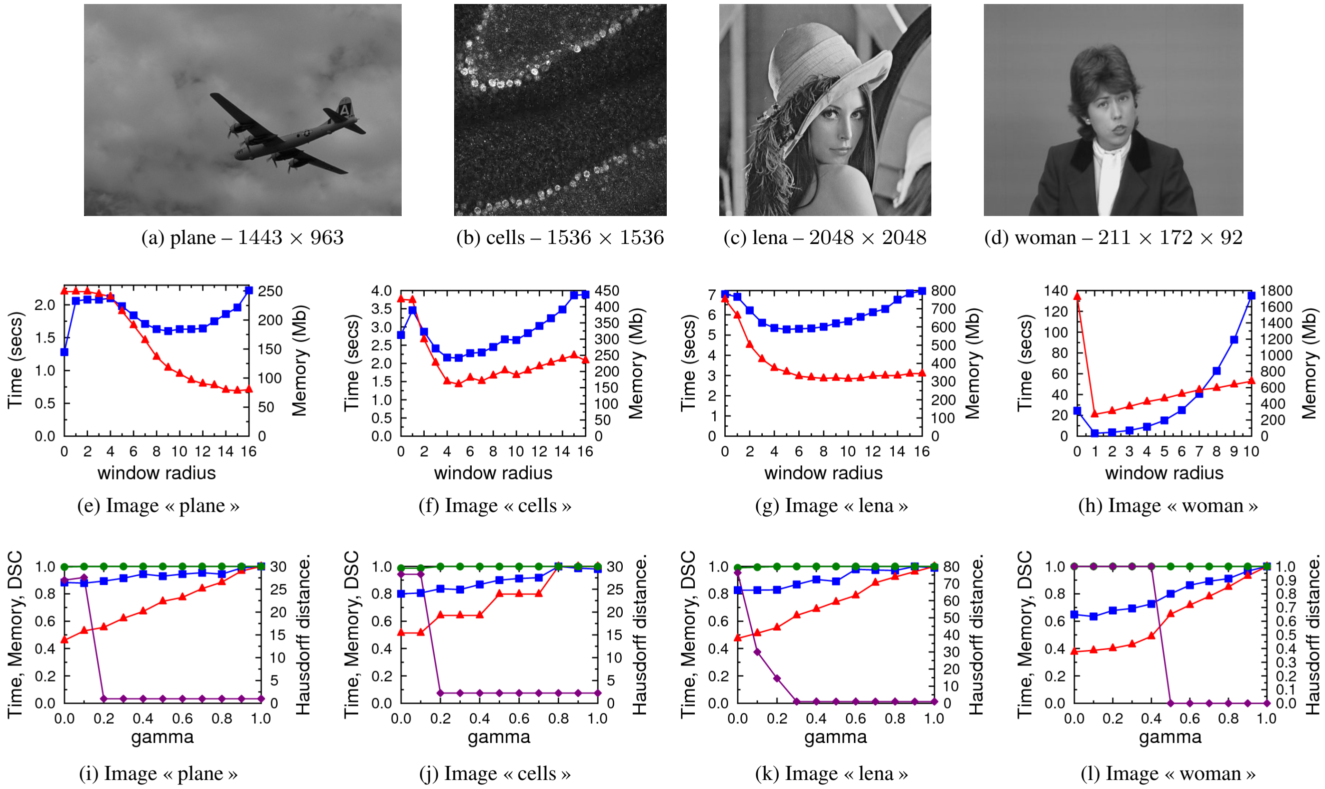
[1] Yin. Li, Jian. Sun, Chi-Keung. Tang, and Heung-Yeung. Shum, "Lazy snapping," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 303–308, 2004.

[2] A.K. Sinop and L. Grady, "Accurate banded graph cut segmentation of thin structures using laplacian pyramids," in *MICCAI*, 2006, vol. 9, pp. 896–903.

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

[3] Details on these evaluation measures are available at `http://lts08.bigr.nl/about.php`

[4] The Brainweb simulator is freely accessible at `http://mouldy.bic.mni.mcgill.ca/brainweb/`

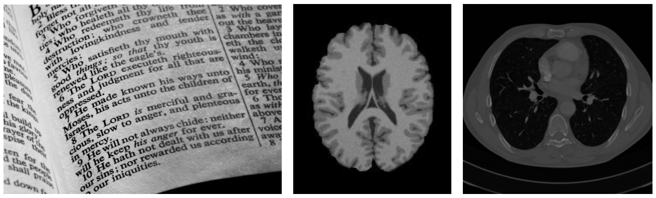


(a) plane – 1443 × 963 (b) cells – 1536 × 1536 (c) lena – 2048 × 2048 (d) woman – 211 × 172 × 92

(e) Image « plane » (f) Image « cells » (g) Image « lena » (h) Image « woman »

(i) Image « plane » (j) Image « cells » (k) Image « lena » (l) Image « woman »

Fig. 2: Influence of window radius (middle row) and γ parameter (bottom row) for segmenting 2D/3D images (top row) with a $TV + L^2$ energy model. On the two last rows, blue curve with squares and red curve with triangles correspond respectively to time execution and amount of memory allocated by the graph. On bottom row, green curve with circles and purple curve with diamonds correspond respectively to the DSC and the Hausdorff distance between normal and γ-parametrized segmentations.



(a) book – 3012 × 2048 (b) brain – 181 × 217 × 181 (c) ct-thorax – 245 × 245 × 151

	Original		Our algorithm	
Image	Time	Memory	Time	Memory
book	5.58	1.08 Gb	3.25	231.25 Mb
brain	/	3.59 Gb	9.02	734.64 Mb
ct-thorax	/	4.58 Gb	8.25	606.27 Mb

Fig. 3: Speed (secs) and memory usage compared to standard graph cuts for segmenting 2D/3D images (top) with a Boykov/Jolly's energy model [8].

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

[5] 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.

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

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

[8] Y. Boykov and M-P. Jolly, "Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images," in *ICCV*, 2001, vol. 1, pp. 105–112.

[9] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts ?," *IEEE Transactions on PAMI*, vol. 26, no. 2, pp. 147–159, 2004.

[10] F. Ranchin, A. Chambolle, and F. Dibos, *Total Variation Minimization and Graph Cuts for Moving Objects Segmentation*, pp. 743–753, 2007.

[11] L. Dice, "Measure of the amount of ecological association between species," *Ecology*, vol. 26, no. 3, pp. 297–302, 1945.