State-of-the art of Graph-based approaches for Image Segmentation

Thomas DAMBRIN
INSA Lyon
March 2020

***Abstract*—In our daily life, we instantaneously identify objects with their shape, color or texture to use, classify or ignore them. Combining this exercise with computers' speed, automation and accuracy, draws the path for the development of various high-level applications. Therefore, many algorithms based on different representations have been proposed throughout the years, each having its strengths and weaknesses. This paper presents a multi-criteria analysis of three techniques for segmentation : the Graph Cuts [1], the Watershed Cuts [2] and the Random Walks [3].**

I. Introduction

Segmenting an image into its composing objects is a well-known problem in Computer Vision and has many applications. It may be involved in fields such as medical image processing, facial recognition, photo/video editing, autonomous driving and more. Although the task is effortless for a human eye, all a computer can see is a set of intensities without any signification. To compute the detection of several regions in a single image, we can count different approaches. Among these, one representation seems to be very seductive : graphs. Using graphs and considering a 2D (resp. 3D) image as a set of pixels (resp. voxels), discretization errors are ruled out as no continuous modeling approximations are performed. That is the first main advantage for graph modeling. Besides, it offers data representation flexibility as it can be applied to 2D or 3D images and videos (see [1]) independently of the topology (i.e. geometry). Finally, graphs are a widely used structure for methods developed in other contexts than image processing. Communication and transportation networks are notable instances. According to [4], it's no rare event that these methods can be adapted to an image setting.

Then, whatever the representation is, segmentation goes through either one of the following schemes. The background/foreground segmentation consideration associates each pixels with the ground it is most likely to belong to. This principle is illustrated in Figure 1. In most cases, foreground (resp. background) region is not necessarily composed of a one and only block but can be a set of isolated areas. The K-way image segmentation is a generalization of the background/foreground idea. It consists of considering K regions to be segmented in an image. The value of K may or not be predefined before computation depending on whether or not the method requires markers. If so, we can talk about supervised segmentation and more precisely interactive segmentation if these markers are user-defined. All of the three algorithms mentioned earlier are designed for interactive segmentation (Graph Cut, Random Walker) or can be adapted to it (Watershed Cuts). Their theoretical principle will be exposed in later sections as it requires some mathematical and graph-theory knowledge first. A quick overview of the criteria of analysis will also be proposed to ease the understanding of the vocabulary and concepts.

II. Requirements

A. Mathematical and Graph-theory concepts

1) for Graph and Watershed Cuts:

In order to avoid confusion, let's set the notations we will use in this paper. A graph G is described by its vertices $v \in V$ (commonly known as nodes) and the edges $e \in E \subseteq V \times V$ between these nodes. Therefore, it can be written as a pair $G = (V, E)$. A graph G can be decomposed into several distinct or not subgraphs G_i. Note that the objective of a graph-based segmentation algorithm is to find a partition of G which matches the human interpretation of the image. In order to understand correctly the principle behind the cuts algorithms, a few definitions are to be known.

Definition 1. Extension. Let X, Y subgraphs of G. Y is an extension of X if $X \subseteq Y$ and any component of Y contains exactly one component of X.

Definition 2. Graph cut. Let $X \subseteq G$, $S \subseteq E$. S is a cut of X if $\overline{S}$ is an extension of X and S is minimal for this property (i.e. if $T \subseteq S$ and $\overline{T}$ is an extension of $X \Rightarrow T = S$).

Figure 2 is a recommended illustration to understand these definitions.

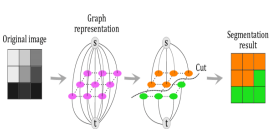

Figure 1. Illustration of Graph Cuts principle[5]. Nodes 's' and 't' will be explained in subsection III.A

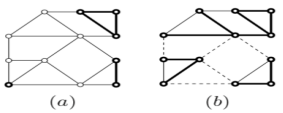

Figure 2. Illustration of graph properties [2]. In bold : (a) a subgraph X of G; (b) an extension Y of X. The set of dashed edges of (b) is a cut of X.

Note that if the edges are weighted (i.e. a cost/weight is associated to each of them), the graph is called edge-weighted. If the costs are associated to vertices, it is called vertex-weighted. In this work, all presented methods are based on edge-weighted graphs[1], the weights mark changes in intensity.

[1]The Graph Cuts algorithm is based on a graph whose vertex and edges are all weighted.

The first two algorithms' purpose is to find an optimal[2] cut of the overall graph representing the image. From this cut, it can identify the different segments composing the image. More method specific knowledge will be given in the related sections.

2) for Random Walks:

Léo Grady proved that simulating the random walks principle is equivalent to resolving the Dirichlet problem. You may all be familiar with partial differential equations (PDE) but not with the problem so here is a brief explanation.

Definition 3. Dirichlet Problem. Let D be an open of $\mathbb{R}^N$, ∂D the boundary of D and $f : \partial D \to \mathbb{R}^N$ a function. The dirichlet problem is to find a harmonic function $u : D \to \mathbb{R}^N$ of class $\mathscr{C}^2$ as $x \in \partial D \Rightarrow u(x) = f(x)$

Remark. A function $f : E \to E$ is a harmonic function if it satisfies $\bigtriangledown^2 f = 0$ (Laplace's equation)

B. Criteria revealing

An non-exhaustive list of the algorithms' properties will be detailed. First and foremost, an important thing for an end-user is how easy and time-saving it is to use the segmentation service. Based on this, the influence of markers location and quantity on the obtained results will be studied. Setting of the implementation must also be easily mastered and brief. For a more universal comparison, computation times will be exposed. The quality of the segmentation obtained is hard to define so we will focus on the following points : modularity and location (and processing) of weak boundaries (i.e. interrupted dividing lines). To pore over modularity, a spotlight will fall on the segmentation supervision and first-result editing if the method allows it.

III. Article Overview

In any of the three articles we will explore, the algorithm can be ordered as (1) Generate the graph, (2) Perform the segmentation. Each of the presented articles uses a different weighting method to generate the graphs.

A. Interactive Graph Cuts

The overall principle of this article is to divide an image into two segments (backgroung/foreground segmentation, see Section I). Before performing, the algorithm requires the user to set seeds (i.e. pixels/voxels which constitute the starting point of the segmentation) and associate to each of these a label (i.e. 'background' or 'object') indicating which region the seed belongs to. Seeds associated to 'object' are regrouped in $\mathscr{O}$ and the ones marked as 'background' in $\mathscr{B}$. The set of defined seeds will be called 'hard constraints'. Besides, as for the Random Walker algorithm, the solution is quite independent of the positioning of the seeds in an objects [1] [6].

[2]'optimal' refers to different properties for different algorithms

After seeding the image, the algorithm uses the intensity of the pixels (resp. voxels) to get histograms of intensity distributions for the two regions. These histograms are then used to set respective regional penalties R_p("obj") and R_p("bkg") as negative log-likelihoods variables. These will be involved in edge-weighting. Once it's done, it generates the graph associating each pixel (resp. voxel) to a node, forming the set $\mathscr{P}$, and linking it to its 8 (resp. 26) direct neighbors. In order to perform the segmentation (i.e. associate 'background' or 'object' to each pixel), two additional nodes are generated : a source S (object) a sink T (background). Each pixel p of the image will be linked to S represented by p,S and T represented by p,T, we will call these edges t-links.

The edges' weights are also affected with $\lambda \geq 0$, the only free parameter of the algorithm which set the cost function. Therefore, it is called 'soft constraints'. For a more complete perception, it specifies the relative importance of regions versus boundaries mostly based on a distinctness property.

Once the graph is well adjusted, the solution is sought by computing a minimum cost cut on the graph G through a max-flow algorithm. Note that the solution obtained is not necessarily unique, there can be several minimum cost cuts in G. One of the flaws of this method is that it may take several trials and more seeds than the other ones. Fortunately, it has a backup plan. If the solution obtained first is not accurate enough, users can add or remove seeds to adjust the intuitive boundaries. If so, t-links' weights are increased in order to recompute the solution without doing it from scratch. To illustrate this editing efficiency, the authors of [1] segmented a video by considering frames as part of a 3D structure. The first result computation may take several minutes for big volumes data but re-computation' results after adjustments were obtained in few seconds. Such a scenario is expected to often happen as the Graph Cuts method can cause small objects segments as it is biased to minimize boundary length [7]. So combining re-edition with adequate soft constraints may produce very accurate segmentation. However, the max-flow algorithm runs in low-order polynomial time relative to the size of the image.

B. Watershed Cuts

The Watershed Cuts method is part of the watershed methods. In such approaches, an image is view as a topographic surface, the grey value of a pixel becomes its elevation level so most of these approaches are based on vertex-weighted graphs. The problem is to find a topological-cut of the graph. Here, it is based on edge-weighted graphs but as the problem is similar, it inherits the connection properties of the vertex-weighted graphs' approaches (i.e. coherence of an object even with color/intensity variations in it). We define a catchment-basin as a region associated to a minimum whose explanation will be given later. The exposed process doesn't require any seeding but note that such an implementation is possible according to [2]. Besides, a control of where to set the boundary on a wide dividing line is achievable through a linear-time preprocessing [8]. By default, the dividing line is placed to the middle. In the method described in [2], the construction of the graph

is based on a 4-adjacency neighboring schemes but as said in section I, it can be generalized to any other scheme. The weights of the edges of the generated graph are settled as the absolute difference of intensity of the associated pixels it is composed of. What we will refer by the 'altitude' of a node is the minimum weight of its edges. Thus, we suppose that the contours are located on the highest edges (i.e. greatest intensity differential). Contrary to the Random Walker algorithm that we will delve into, it does not need any sorting step, nor any use of sophisticated data structure to be computed. In order to explain the intuitive behavior of the algorithm, let's keep one definition in mind.

Definition 4. Minimum. Let $X \subseteq G, k \in \mathbb{Z}$ and $F : E \to \mathbb{Z}$ be the map used to weight the edges. The subgraph X of G is a minimum for F at altitude k if :

- X is connected
- k is the weight of any edge of X
- the weight of any edge adjacent to X is greater than k

Each region to be segmented is associated with a minimum. The aim is to find a set of Watershed-cuts that correctly segment the image. These are found thanks to the intuitive 'Drop of water' principle. Consider that the boundary of a region is a set of edges from which water can flow down towards two distinct minima. The same result can be obtained through finding minimum spanning forests relative to subgraphs of G which each induce a unique watershed-cut. This duality of methods is imaginable thanks to the fact that with an edge-weighted graph, a watershed-cut can be equally defined by its catchment-basins or its dividing lines. With the dividing lines approach, the optimal cuts can be found in quasi-linear time thanks to a Minimum Spanning Tree algorithm [9] (e.g. Prim's algorithm). Besides, although it is not a supervised method, it supports the restriction of the number of resulting segments. It can be done by filtering the weights to reduce the number of minima. However, noise handling needs to be performed through image preprocessing. Finally, the proposed weighting function is well-adapted for segmenting an image into its 'homogeneous' zones but an alternative one can be used for a more classical segmentation i.e. segmenting darker zones separated by brighter ones.

C. Random Walks

The Random Walker algorithm is an interactive segmentation technique. As for the Graph Cuts, it requires user-defined or predefined seeds to perform the algorithm. The requirements are quite heavy as there has to be one label, being composed of one or more seeds, for each region to be segmented. Otherwise, the segmentation will either decompose an object (too much labels) or band several together (too few labels). The overall principle can be summed up in the following instruction : assign the label of the seed for which a random walker starting at an unmarked pixel has the greatest probability of first reach. Indeed, the problem is posed as it is biased to avoid crossing objects' boundaries thanks to edge weights. According to [3], computing the very real principle of random walks "would be completely impractical". The good news is that the correspondence between circuit theory and random walks gives us a way to implement it. In any ways, the weighting function for the edge containing pixels p_i and p_j of a gray scale (resp. colorful) image is $w_{ij} = exp(-\beta(g_i - g_j)^2)$ (resp. $||g_i - g_j||^2$). Then, we set matrices thanks to which we will be able to solve the Dirichlet Problem. Thanks to the way these matrices are related, solving a sparse, symmetric and positive-definite system of equations is enough to get the resulting segments. This resolution is straightforward to implement. As for the Graph Cuts, it can be corrected with modifying the seed points and using the previous solution as an initial one for the iterative matrix solver. Most of the properties of this algorithm are very attractive. First, as it's the intuitive situation would happened in your imagination, weak boundaries are simply handled by assigning the closest seeded pixel. Its design guarantees the avoidance of fragmented segmentations sometimes obtained by other algorithms.

IV. Overall Comparison

Whereas one algorithm is performing in a low-order polynomial time (Graph Cuts), others may run in a quasi-linear time (Random Walks, Watershed Cuts) still being an attractive alternative to machine learning approaches. These theoretical properties are proven by Figure 3. The setting of the parameters presents different aspects. Besides, the analog electric circuit implementation of the Random Walker should perform way faster than the 'CPU speed'. According to the criteria given in subsection II.B, the ranking could be the subsequent. (1) Watershed Cuts : selection of type of segmentation, no seeding, no sorting step (2) Random Walks : one free parameter with minimal influence, at least one label per region (3) Graph Cuts : one free parameter with great influence, only two labels required. Noise robustness can be handled with preprocessing (Random Walks, Watershed Cuts) or setting (Random Walks, Graph Cuts).

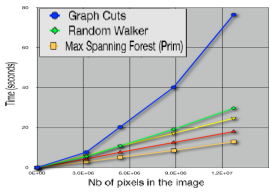

Figure 3. Computation times 2D [7][3]

V. Conclusion

In this work, we presented three algorithms performing image segmentation. Their overall principle and characteristics were reviewed in a very brief way. The supervised methods, Graph Cuts and Random Walks, requires a time-spending preprocessing to offer more flexibility. In opposition to boundaries marking, used in [10], it can be applied to any N-dimension image including video. Thus, according to the content of this article, the three algorithms are suited for different environments. For instance, we may imagine local (as image sending can induce noise) image classification using Watershed Cuts, photo and video edition using Graph Cuts and Random Walks. In any case, each method has its own advantages as a framework for easy setting and use of the three algorithm has been developed in [7].

[3]Max Spanning Forest method correspond to Watershed Cuts

REFERENCES

[1] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, volume 1, pages 105–112. IEEE, 2001.

[2] Jean Cousty, Gilles Bertrand, Laurent Najman, and Michel Couprie. Watershed cuts: Minimum spanning forests and the drop of water principle. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(8):1362–1374, 2008.

[3] Leo Grady. Random walks for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(11):1768–1783, November 2006.

[4] Xiaofang Wang. *Graph based approaches for image segmentation and object tracking*. PhD thesis, Ecully, Ecole centrale de Lyon, 2015.

[5] Romane Gauriau. *Shape-based approaches for fast multi-organ localization and segmentation in 3D medical images*. PhD thesis, 06 2015.

[6] Ali Kemal Sinop and Leo Grady. A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[7] Camille Couprie, Leo Grady, Laurent Najman, and Hugues Talbot. Power watershed: A unifying graph-based optimization framework. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(7):1384–1399, July 2011.

[8] Jos B.T.M. Roerdink and Arnold Meijster. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundam. Inf.*, 41(1,2):187–228, April 2000.

[9] Bernard Chazelle. A minimum spanning tree algorithm with inverse-ackermann type complexity. *J. ACM*, 47(6):1028–1047, November 2000.

[10] Eric N. Mortensen and William A. Barrett. Intelligent scissors for image composition. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, page 191–198, New York, NY, USA, 1995. Association for Computing Machinery.