A Comparative Study of Image Segmentation by Application of Normalized Cut on Graphs

Anselmo Ferreira
Institute of Computing-IC
University of Campinas- UNICAMP
Campinas, São Paulo, Brazil
Email: ra023169@ic.unicamp.br

Marco A. G. de Carvalho
School of Technology- FT
University of Campinas- UNICAMP
Limeira, São Paulo, Brazil
Email: magic@ft.unicamp.br

***Abstract*—The graph partitioning has been widely used as a mean of image segmentation. One way to partition graphs is through a technique known as Normalized Cut, which analyzes the graph's Laplacian matrix eigenvectors and uses some of them for the cut. This work proposes the use of Normalized Cut in graphs generated by structures based on *Quadtree* and Component Tree to perform image segmentation. Experiments of image segmentation by Normalized Cut in these models are made and a specific *benchmark* compares and ranks the results obtained by other graph-conversion techniques proposed in the literature. The results are promising and allow us to conclude that the use of different graph models combined with the Normalized Cut can yield better segmentations according to the characteristics of images.**

***Keywords*-Image Segmentation; Normalized Cut; Quadtree; Component Tree**

I. Introduction

The goal of computer vision is provide computers human vision habilities, such as recognizing and distinguishing objects in a scene. One image processing technique used for helping this task is the image segmentation, which divides an image into pieces that are suitable for machine operations.

Lots of image segmentation approaches were proposed in literature. The graph-based one uses a graph representation of an image and uses a criteria for splitting it into pieces [1], [2], [3]. In this approach, image features such as pixels or image regions are modelled as graph nodes and graph edges stablishes relations among them. So, a criterion is used to partition this graph, yielding the image segmentation.

One graph partitioning criterion well-used is the sum of edges weight removed to generate two subgraphs, also called the cut. The Normalized Cut (NCut) approach by Shi and Malik [4] is a graph-cut technique responsible for generating balanced subgraphs by removing the minimum edges possible. It is based on a spectral graph theory concept by Fiedler [5], which originally uses the second smallest eigenvector of the graph representative matrix as a guide for graph partitioning. The inherent bias of this technique is that balanced partitions for image segmentation cannot be appropriate for some images when small number of partitions is desired (*e.g*, images with an easily detectible object and a uniform background). This technique was originally proposed to cut pixel-based similarity graphs.

Contributions: This paper proposes two different approaches based on existing ones to create graphs from images and apply NCut image segmentation on them. These approaches are based on images hierarchical information: one is based on Quadtree decomposition and other is based on the Component Tree representation. We argue that these representations can be useful to implement image segmentation based on Normalized Cut for specific images and applications. For that, we use some images from The Berkeley Image Database [6] and classify their NCut segmentations in different graph-conversion approaches, using a wide used benchmark for this task.

This paper is organized as follows: Section II presents the NCut image segmentation approach. In Section III, we show some image-graph conversion approaches applied to NCut segmentation proposed in the literature. Section IV presents the proposed ones and in section V we discuss the image segmentation experiments. Section VI concludes this work.

II. The Normalized Cut framework

Shi and Malik [4] proposed a new criterion for measuring the graph cuts accuracy and performing image segmentation. The authors argue that their technique solves the problem stated by Wu and Leahy in their minimum cut criterion for graph cutting [3]. Given a graph bipartition $G = A \cup B$, the Normalized Cut cost is:

$$NCut(A,B) = \frac{Cut(A,B)}{Vol(A,V)} + \frac{Cut(A,B)}{Vol(B,V)}, \qquad (1)$$

where $Cut(A,B)$ is the sum of edges removed to split the graph and $Vol(A,V)$ and $Vol(B,V)$ are, respectively, the sum of weights in the nodes in A and B to all nodes in the original graph G.

The best Normalized Cut in a graph is the one that minimizes the $NCut$ value. The problem in minimizing Equation (1) is that it has a NP-Hard complexity[4]. In their paper, Shi and Malik extended this equation and founded a well-known equation in linear algebra called the Rayleigh Quotient [7]:

[0]This work is related to a master thesis presented at UNICAMP's School of Technology in January 2011

$$RQ = \frac{v^T A v}{v^T v}, \quad (2)$$

where v is orthogonal to the $n-1$ smallest eigenvectors $v_1 ... v_{n-1}$ of A. Shi and Malik defined $A = \frac{D-W}{D}$ [4]. One fact about the Rayleigh Quotient is that it is minimized by the second smallest eigenvector v_2 of A, and its minimum value is λ_2, the second smallest eigenvalue of A [4]. To do so, the following generalized eigenvalue system has to be solved:

$$(D - W)v = \lambda v. \quad (3)$$

The graph splitting is guided by v_2, where each value $v_2[i]$ will represent a graph node i and it is called the *characteristic value* of node i. To split a graph, a threshold value is used and the graph nodes are partitioned in two subsets. The most common threshold values are zero, the median value in v_2 or the one that minimizes the $NCut$ value. Shi and Malik used the latter approach by checking l possible splitting points and calculating the best $NCut$ among them. They proved that their approach is reliable even with small l [4]. The cut can be recursively done in the two partitioned parts and stops when a previously given $NCut$ value is reached, resulting a previously unknown number of partitions. This technique is known as Recursive Two-Way Cut and its steps are described as in the following:

1) Given a weighted graph G, build the weight matrix W and the degree matrix D.
2) Solve $(D - W)v = \lambda Dv$.
3) Threshold v_2, where $v_2[i]$ is a node in the graph, yielding partitions A and B.
4) Calculate $NCut$.
5) Repeat the previous steps in each subgraph only if the $NCut$ value is below a prespecified threshold.

If one wants to simultaneously generate a K fixed partitions, so the K first eigenvectors can be used as a K dimensional vector for each pixel. The partitioning process using K-way Cut uses the two first steps of the Recursive Two-Way Cut, the difference is the use of the K first discretized eigenvectors as the guide for partitioning [4], [8].

Shi and Malik also proposed their method as a way of image segmentation by first modelling an image into a graph. For that, an affinity graph must be builded using image structural features (e.g, pixels, regions, among others) as graph nodes and a similarity function to weight its edges.

For image segmentation, this technique has two problems that must be dealt: the high computational cost for graphs with high connections and the correct choice of the image-graph conversion method to segment specific images. In the following two Sections we discuss some image graph conversion approaches applied in NCut Image segmentation and present two other based on existing ones.

III. Image Graph Conversion Approaches to Normalized Cut in the literature

Several image-graph modelling techniques were proposed as input for the Normalized Cut. In this section, we review three different approaches proposed in the literature.

1) Pixel Affinity Graph: In this modelling, each pixel is taken as a graph node, and two pixels within a r distance are connected by an edge. The edges weights should reflect the similarity between the pixels connected by them. The grouping cue used in the similarity function will reflect the overall quality of the segmentation. Some of them are the intensity, position and contours [4], [8], [9].

The intensity and position grouping cue Wip assumes that close-by pixels with similar intensity are most probably to belong to the same object.The measure of similarity regarding this grouping cue is given by Equation (4) [4], [8]:

$$Wip(i,j) = \begin{cases} e^{-\left(\frac{\alpha^2}{d_p}\right) - \left(\frac{\beta^2}{d_i}\right)}, & \text{if } \alpha < r \\ 0, & \text{otherwise} \end{cases}, \quad (4)$$

where $\alpha = ||P_i - P_j||$ and $\beta = ||I_i - I_j||$ are, respectively, the position and intensity difference between pixels i and j; r is a given distance (also called graph connection radius), α is the distance between i and j; and d_p and d_i are the corresponding scale parameters which control the tradeoff between the brightness likehood and spatial proximity.

This grouping cue used separately often gives bad segmentations because some natural images are affected by texture clutter. So, another grouping cue Wic is regarded to the intervening contours, given by Equation (5) [8]:

$$Wic(i,j) = \begin{cases} e^{-\left(\frac{\max_{(x \in \text{line} f(i,j))} \epsilon}{d_c}\right)}, & \text{if } \alpha < r \\ 0, & \text{otherwise} \end{cases}, \quad (5)$$

where line$f(i,j)$ is a straight line joining pixels i and j and $\epsilon = ||Edge(x)||^2$ is the square of edge strength at location x. In this grouping cue, two pixels have a high affinity if a straight line across them doesn't cross an image edge.

These two grouping cues can be combined as shown by Equation (6) [8]:

$$Wipc(i,j) = \sqrt{Wip(i,j) \cdot Wic(i,j)} + \alpha_2 \cdot Wic(i,j), \quad (6)$$

where α_2 is a constant given by user [8].

2) Multiscale Graph Decomposition: In their paper, Cour, Benzit and Shi [8] argue that high graph connection radius r generally makes *better segmentations* because it facilitates the propagation of local grouping cues along image regions. But it demands a high computational cost, once it makes the graph's similarity matrix denser. They propose the graph decomposition in multiple scales. For that, in the first scale W_1, every pixel is a graph node and are connected if they are within r distance apart. In the second scale, pixels are sampled at $2r+1$ distance apart and, in scale s, pixels are sampled at $(2r+1)^{s-1}$.

Their algorithm works on these scales to capture coarse and fine level details. The construction of the graph is then

given according to $W = W_1 + W_2 + \ldots + W_s$, where W represents the graph similarity matrix and s, the scale, i.e., each W_s is an independent subgraph. As r value is a tradeoff between the computation cost and segmentation result, W_s can be compressed using recursive sub-sampling of image pixels. This compression is not perfect, but he has the advantage of the computational efficiency.

3) Watershed regions based similarity graph: The Watershed Transform is a region-based image segmentation that treats a grayscale image or the image gradient as a topographic surface. The image is then flooded from a set of selected sources (also called regional minima) until the whole image has been flooded. Dams are then built between different lakes before them meet, generating, in this way, the watershed lines and watershed regions [10].

One problem with this technique is the image supersegmentation due to the high number of regional minima in the images. Meyer [11] dealt with this problem by suggesting the hierarchical Watershed. In this case, the flooding process begins in a given threshold value tv that represents some relief feature, for instance, the altitude. So, some initial regions will be flooded, yielding in this way the number of partitions desired [12].

The hierarchical watershed regions can be modeled using graphs. The flooded gradient image is represented by a full-connected weighted neighborhood graph, where a node represents a catchment basin of the topographic surface. Hierarchical Watershed can be used in order to reduce the number of nodes (super segmentation problem) that is originated by the primitive Watershed in the correspondent graph. After the conversion, one weighting function that can be used is the mean intensity, as proposed in [13]:

$$W_{IW}(i,j) = e^{-|Im_i - Im_j|}, \tag{7}$$

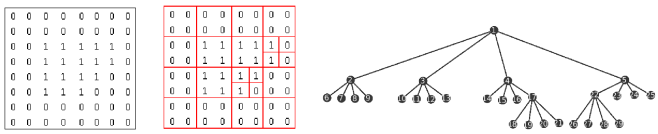
where Im is the mean intensity of Watershed regions i and j. Another interesting approach to modelling image graphs using watershed regions uses centroid pixels and can be found in [14].

IV. Quadtree and Component Tree based Similarity Graphs

Hierarchical information about images can give good clues about image graph constructions if one wants to segment images into meangniful regions. In this section, we describe two image graph construction approaches previously presented in [15], [16].

A. Quadtree based similarity gaph

The term Quadtree is used to describe a class of hierarchical data structures created by the recursive decomposition of space [17]. In order to represent an image through a Quadtree, it should be recursively decomposed into exact four new disjoint regions when they satisfy a defined criterion. The initial region corresponds to the whole image and is associated to the tree root node [17], [18]. Fig. 1 shows a Quadtree decomposition example.



(a) Original Image (b) Quadtree Decomposition (c) Quadtree

Fig. 1. Quadtree decomposition example.

Defining the criterion to Quadtree decomposition is not a trivial task. There are different criteria proposed, such as the standard deviation or entropy of image graylevels [18]. We proposed applying the Canny filter in the image before the decomposition. This Filter, proposed by Canny in [19] is a more robust filter because it is less sensitive to noise in images. By removing pixels with low gradient and thresholding the resulting ones, this process produces a binary image with border pixels highlighted and apply Quadtree decomposition on them. This procedure was chosen because:

1) the filter results in a binary matrix. Then, it became trivial to define that a region should be decomposed when it is not formed entirely by 1's or 0's [17].
2) the edge detection operation drastically reduce the size of data to be processed, while at the same time preserves the structural information about object boundaries [19];

The main goal of using a Quadtree image representation is to reduce the similarity graph size. For this purpose, the input graph will be generated using the regions associated to the Quadtree leaves. Each region will be associated to a graph node and in each region a *centroid pixel* is defined as the representative pixel for that region. The connection radius is given by Equation (8):

$$Radius = \frac{\max(RS_a, RS_b)}{2} + r, \tag{8}$$

where RS_a and RS_b are the sizes of the two regions being connected and r is the radius given by the user. The edges weighting function will be the same as used in pixels similarity graph in Equation (5), but now we use only the regions centroid pixels and the ones reached by the graph connection radius as the graph nodes.

The number of regions obtained by the proposed technique will vary depending on the image data and size. Also, the parameters of the edge detection filter can be manually specified, in order to change its sensibility. It means that the number of nodes on the similarity graph can be influenced by the choice of the edge detector parameters.

B. The Component Tree-based similarity graph

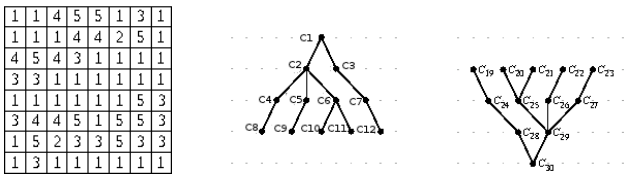
The Component Tree (CT) is a hierarchical representation of a grayscale image after several thresholding operations between its minimum and maximum graylevels. There is a relation of inclusion between components at sequential cross-sections of the image. A cross-section is defined as a binary image given by Equation (9) [20]:

$$F_k = \{x \in F / F(x) \geq k\}, \tag{9}$$

where F is an image and F_k is a section k (level) of F. The levels in k are in the range $[GMin, GMax]$, where $GMin$ is the smallest graylevel and $GMax$ is the higher graylevel in the image.

The Connected Components (CC) of the different cross-sections are organized in a tree structure. Two CCs C_{k+1} and C_k are linked in this tree when C_{k+1} is a subset of C_k. The CC of the first cross-section F_{min} corresponds to the whole image domain and it's called root.

The traditional CT is formed only by the *1's* CCs, once the cross-section used for the root corresponds to the minimal graylevel. However, there is still information on the cross-sections related to the *0's* CCs that are not included in the traditional CT. For some particular cases these CCs hold more relevant information than the *1's* CCs does. Therefore, we build a Reverse Component Tree (RCT), where two CCs C_k and C_{k-1} are linked when C_{k-1} is a subset of C_k. In this case, the root of the tree is formed by the CC of the last cross-section F_{max}. Fig. 2 shows a simple example of CT and RCT construction.



(a) original image (b) Component Tree (c) Reverse Component Tree

Fig. 2. Example of buiding the Component Tree and Reverse Component Tree

The final similarity graph $G = \{G^1_{cs}, G^2_{cs}, \ldots, G^n_{cs}\}$ is a set of n complete subgraphs $G^n_{cs} = (C_k, E)$, where: n is the subgraph identifier; C_k is the set of all connected components from cross-section k; and E is the set of edges that connects mutually every node from G^n_{cs}. Finally, the disconnected subgraphs are linked together by CT and RCT edges.

It is not necessary to include all the cross-sections in the final graph G. In general, the cross-sections at extreme gray levels do not contain much information about the image and also there is a lot of redundant information among all cross-sections. Thus, one method to choose which cross-sections should be added to G can be done by selecting a cross-section correspondent to a middle gray level with more CCs; then, the adjacent cross-sections can be added to G until a desired number of nodes is achieved.

The G weights should reflect the similarity between the CCs and can be obtained from combinations of both CC and image attributes; but the edges that links CCs from different cross-sections needs to receive a higher similarity value to ensure that very similar CCs on subsequent cross-sections are kept on the same region.

V. Experiments

A. Database and benchmarking

We chose the *Berkeley Image Segmentation Dataset* and its open source benchmark to evaluate the experiments. Two reasons tells us to use the Berkeley Image Dataset in the experiments:

1) The images available were previously segmented by humans, so, the machine segmentations can be compared to the human-made ones.
2) An open-source benchmark is available to classify the segmentations, according to their similarity to the manual segmentations.

The benchmark uses two metrics in the classification task: Precision (P) and Recall (R). Precision is the probability that the pixel marked as a border pixel in fact be a border pixel; Recall is the probability that the border pixels marked by the machine is the same as the border pixels marked by humans. These two metrics are summarized in the F-measure [21]:

$$F = 2 \cdot \frac{P \cdot R}{P + R}. \tag{10}$$

The process of Precision-Recall calculation for the image segmentation comparison occurs 30 times or the number of times defined by user. In each time, the segmented image is thresholded and the final image is then compared to the manual segmentations, yielding the Precision-Recall values. The greater F-measure value in all of these points is then defined as the algorithm ranking.

B. Parameters

In the experiments, the images dimensions had to be resized to 256x256 because of the Quadtree requirement for squared images with size 2^n. So, we decided to do this to make a fair comparison with other approaches. The comparison with the ground-truth segmentations is done in the benchmark after performing a bicubic interpolation, to resize the segmented images to their original sizes. The Pixel affinity graph was configurated to use the intervening contours weighting function in Equation (5) with $d_c = 0.1$ and a connection radius $r = 10$. We will call the segmentation in this graph NCutPixel.

The Pixel affinity graph using the multiscale decomposition was configurated to use the intensity and contours weighting function as presented in Equation (6), with $\alpha_2 = 1$, $d_i = 0.12$ and $d_c = 0.08$. In this case, the graph was decomposed in three scales. The segmentation algorithm works simultaneously across these graph scales, with an inter-scale constraint to ensure communication and consistency between the segmentations at each scale [8]. We will call the segmentation in this graph NCutPixelMS.

The Watershed regions affinity graph uses a full-connected input graph with 1000 regions generated by the hierarchical watershed, and the weighting function uses the difference between the mean graylevel intensity values of each region, as cited in Equation (7). We will call the segmentation in this graph NCutWshed.

The Quadtree affinity graph uses the Quadtree leaf nodes as the similarity graph nodes. The radius used was $r = 10$ and the weighting function uses the intervening contours described in Equation (5). We will call the segmentation in this graph NCutQT.

The Component Tree based similarity graph uses a full connected graph using the connected components from the CT and RCT cross-sections as graph nodes. The weighting function uses the difference of mean graylevels, standard deviation, distance and area of two connected component. The segmentation in this graph is spread among all the cross sections, and the cross section with best segmentation is chosed manually to be the final segmentation. We will call the segmentation in this graph NCutCT.

C. results and discussion

The number of segments for each image was chosen to be the same as one chosen by one manual segmentation for that image. To do that, we accessed the segmentation data for a randomly chosen volunteer. The segmentation was then performed by the K-way Cut described in Section II.

Originally, the Berkeley benchmark classifies only border-based image segmentations. So, we use a border detector in the images segmented by its regions to benchmark the results. We decide to use the Canny filter [19] also available in the benchmark because it is faster than the others. The result of this process is a graylevel image used for benchmarking. Table I shows the final ranking for 50 images segmented using the F-Measure.

TABLE I
FINAL SCORES FOR THE NORMALIZED CUT ALGORITHM WITH DIFFERENT IMAGE-GRAPH CONVERSION APPROACHES.

Position	Score	
1	0.52	NCutPixelMS
2	0.50	NCutPixel
3	0.48	NCutQT
4	0.48	NCutCT
5	0.44	NCutWshed

Despite the proposed methods were not the best ones when applied to NCut segmentation, the Table I shows only the mean F-Measure and tells nothing about the segmentation of each image separately. The Table II shows the F-measure per image where the proposed image-graph conversion approaches got best or equal NCut segmentations than the Pixel graph with multiscale decompositon, according to the benchmark. Fig. 3 shows the region-based segmentation results.

As shown in Tables I and II, the NCutWshed approach had the worse performance than others. This fact is explained by the use of a full connected graph. In this case, regions not phisically neighboring are neighbors in the similarity graph, so the Normalized Cut can put them in the same region.

Because of its inherent characteristic of generating different segmentations in each cross section using connected components as graph nodes, the NCutCT is appropriated to segment some of the images, *e.g*, 58060, 3096, 208001 and 253027.

TABLE II
F-MEASURES OF NORMALIZED CUT WHEN APPLIED TO DIFFERENT IMAGE GRAPHS ACCORDING TO THE BERKELEY BENCHMARK. THE BEST ONES ARE HIGHLIGHTED IN GRAY

Image/Algorithm	58060	3096	208001	253027	42049
NCutPixel	0,37	0,27	0,48	0,41	0,76
NCutPixelMS	0,40	0,38	0,55	0,41	0,81
NCutWshed	0,37	0,28	0,53	0,54	0,54
NCutQT	0,36	0,29	0,52	0,36	0,82
NCutCT	0,46	0,71	0.57	0,56	0,71

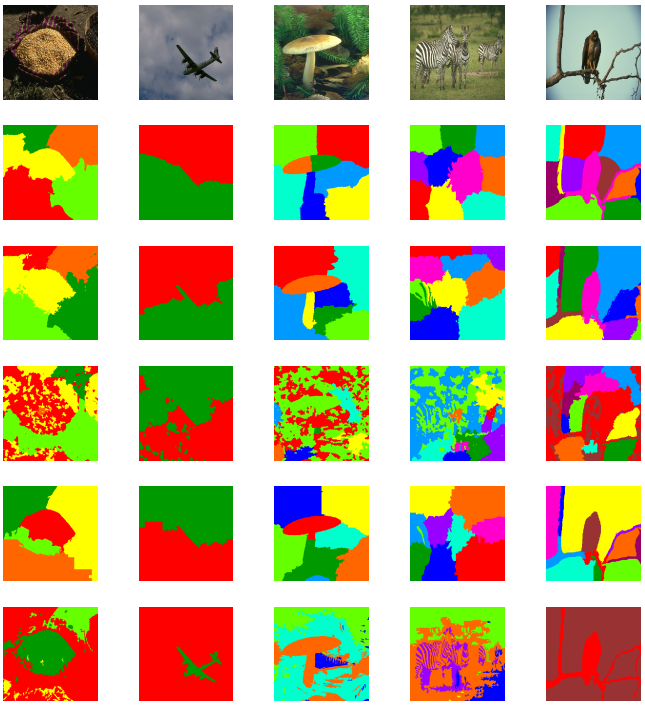


Fig. 3. Images and NCut segmentations that achieves better results when applied in the proposed image-graph conversion approaches: First Row: original images 58060, 3096, 208001, 253027 and 42049 respectively. Second, third, fourth, fifth and sixth rows shows the NCut segmentations using as input the pixel affinity graph, the multiscale graph decomposition, the Watershed-based affinity graph, the component tree and Quadtree based similarity graphs.

Fig. 4 shows the NCutCT Precision-Recall plot for image 3096. The behavior of this graph shows the Precision increasing and remaining constant before the F-measure chosed point and decaying after this point, indicating that the number of false positives are low only in the threshold values before this point.

According to Table I, the NCutQT has the same score from the NCutCT, but its advantage is using lower number of graph nodes (only the quadtree leaves). Fig. 5 shows the NCutQT Precision-Recall plot for image 42049. In the Figure, the greater F-measure point was $F = 0.82$ in the threshold value $t = 0.35$. The graph behavior before this point shows that the Precision remains high and Recall starts and remain increasing in most parts of the graph. After this point, the Recall remains constant and Precision decays, indicating that the number of false positives increases.

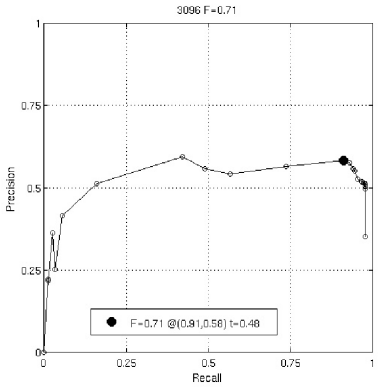


Fig. 4. Precision-Recall plot of the Normalized Cut in the component tree similarity graph of image 3096.

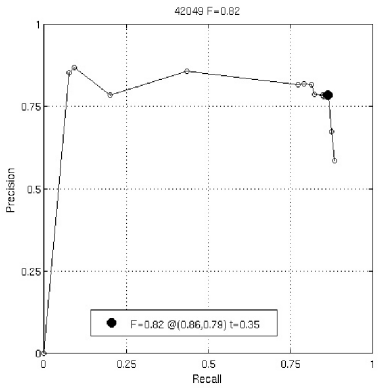


Fig. 5. Precision-Recall plot of the Normalized Cut in the quadtree based similarity graph of image 42049.

VI. Conclusion

In this work, we propose the use of hierarchical information for building image similarity graphs and discuss the image segmentation on them when used as input to the Normalized Cut approach. Our graph construction methodologies uses concepts from some already known image representations: Quadtree and Component Tree.

In our experiments, we segment images of general purpose and show that some images can be better segmented with Normalized Cut in the similarity graphs proposed. So, this image segmentation approach proves to be very flexible, because different image graph representations yield different segmentations, showing its potential to different applications. Benchmark results showed that, because of the general purpose nature of the images used to segmentation, the final result tells nothing about the algorithm performance in each one separately.

Extensions of this work are proposing these approaches to a specific application, using an image database of a specific kind of image, use different weighting functions and use physical neighborhood in the Watershed Similarity Graph.

Acknowledgment

This work was supported by CAPES Brazilian Agency.

References

[1] L. Grady, "Random walks for image segmentation," in *IEEE Transactions os Pattern Analysis and Machine Intelligence*, vol. 18, 2006, pp. 1–17.

[2] L. Grady and E. Schwartz, "Isoperimetric graph partitioning for image segmentation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, 2006, pp. 469–475.

[3] Z. Wu and R. Leahy, "An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation," in *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, November 1993, pp. 1,101–1,113.

[4] J. Shi and J. Malik, "Normalized cuts and image segmentation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 731–737.

[5] M. Fiedler, "A property of eigenvectors of nonnegative symmetric matrices and its applications to graph theory," *Czech. Math. Journal*, vol. 25, no. 100, pp. 619–633, 1975.

[6] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th Int'l Conf. Computer Vision*, vol. 2, July 2001, pp. 416–423.

[7] G.H.Golub and C. Loan, *Matrix Computations*. John Hopkins Press, 1989.

[8] T. Cour, F. Bénézit, and J. Shi, "Spectral segmentation with multiscale graph decomposition," in *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition - CVPR'05*, vol. 2, 2005, pp. 1124–1131.

[9] J. Malik, S. Belongie, J. Shi, and T. Leung, "Textons, contours and regions: cue integration in image segmentation," in *Proc. of IEEE International Conference on Computer Vision*, Corfu - Greece, 1999, pp. 918–925.

[10] S. Beucher, "The watershed transformation applied to image segmentation," in *Scanning Microscopy International*, 1991, pp. 299–314.

[11] F. Meyer, "Hierarchies of partitions and morphological segmentation," in *Proc. of Scale-Space*, 2001, pp. 161–182.

[12] M. A. G. Carvalho, "Análise hierárquica de imagens através da árvore dos lagos críticos," Ph.D. dissertation, Faculdade de Engenharia Elétrica e Computação- FEEC, 2004.

[13] M. A. G. Carvalho, A. C. B. Ferreira, T. W. Pinto, and R. M. Cesar-Jr., "Image segmentation using watershed and normalized cuts," in *Proc. of 22th Conference on Graphics, Patterns and Images (SIBGRAPI)*, Rio de Janeiro - Brazil, 2009.

[14] F. C. Monteiro and A. Campilho, "Watershed framework to region-based image segmentation," in *Proc. of IEEE 19th International Conference on Pattern Recognition - ICPR*, 2008, pp. 1–4.

[15] M. A. G. Carvalho, A. C. B. Ferreira, and A. Costa, "Image segmentation using quadtree-based similarity graph and normalized cuts," in *Proc. Of XV Iberoamerican Congress On Pattern Recognition*, São Paulo-SP, Brazil, 2010, pp. 329–337.

[16] M. A. G. Carvalho, A. C. B. Ferreira, A. Costa, and R. M. Cesar-Jr., "Image segmentation using component tree and normalized cuts," in *Proceedings of the XXIII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI2010)*, Gramado - Brazil, 2010, pp. 317–322.

[17] H. Samet, "The quadtree and related hierarchical structures," *ACM Computing Surveys*, vol. 16, no. 2, pp. 187–261, 1984.

[18] L. A. Consularo and R. M. Cesar-Jr., "Quadtree-based inexact graph matching for image analysis," in *Proceedings of the XVIII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI2005)*, Natal - Brazil, 2005, pp. 205–212.

[19] J. Canny, "A computational approach to edge-detection," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986, pp. 679–700.

[20] V. Mosorov and T. M. Kowalski, "The development of component tree for grayscale image segmentation," in *Proc. of International Conference on Moderns Problems of Radio Engineering, Telecommunications and Computer Science – TECSET*, Slavsko – Ukraine, 2002, pp. 252–253.

[21] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *ICML '06: Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 233–240.