# Improved Image Segmentation Algorithm Using Graph-Edges

**Imad Kassar Akeab***

Received on:30/8/2009
Accepted on:1/4/2010

**Abstract**

In this paper an efficient algorithm for segment digital image has been developed by measuring the evidence for a boundary between two regions in an image using (graph-edges). The regions in the image were sorted as components, where each region in an image represents a component in the graph. The region comparison predicate evaluates if there is evidence for a boundary between a pair of components by checking if the difference between the components, is large relative to the internal difference within at least one of the components. A threshold function is used to control the degree the difference between components must be larger than minimum internal difference. An important characteristic of the method is its ability to preserve detail in important image regions while ignoring detail in unimportant regions. The classical methods depend just on external difference and ignore the internal difference, when segment two neighboring regions.

## تقسيم الصورة الرقمية باستخدام خوارزمية مطورة لترسيم الحافات غرافيا

**الخلاصة**

في هذا البحث تم تطوير خوارزمية كفؤة لتقسيم الصورة الرقمية بناءا على قياس دليل على وجود حدود بين المناطق داخل الصورة الرقمية. حيث تعتمد هذه الخوارزمية على بناء (graph-edges) ثم ترتيب المناطق في الصورة على اساس المركبات, حيث ان كل منطقة تمثل بمركبة في ال (graph). ونجري عملية مقارنة بين كل منطقة للتاكد من وجود حد بينهما عن طريقة التاكد من ان الفرق الخارجي بين كل مركبتان داخل الصورة هو اكبر من اصغر احد الفروق الداخلية لكل منطقة على حدة لكي يتم اعتبار المنطقتان متجاورتان والا يتم دمج المنطقتان المتجاورتان واعتبارهما منطقة واحدة. تم السيطرة على مدى الاختلاف بين الفرق الخارجي للمركبات والفرق الداخلي عن طريق دالة العتبة. ان الطرق الكلاسيكية تاخذ بنظر الاعتبار الفرق الخارجي للمركبات وتهمل الفرق الداخلي. من اهم خصائص هذه التقنية هو قدرتها على استخلاص المناطق المهمة في الصورة وتتجاهل المناطق الغير مهمة فيها.

***Astronomy Department, University of Baghdad/ Baghdad**



https://doi.org/10.30684/etj.28.11.14
2412-0758/University of Technology-Iraq, Baghdad, Iraq

## 1. Introduction

Image Segmentation is used to partition a given image into a number of regions, so that each region corresponds to an object. (intensity, color, texture ...). Image segmentation stays a big problem in image processing. Higher-level problems such as image recognition and image indexing can also make use of segmentation results in matching. Our aim is to improve computational approaches to image segmentation that are Broadly useful, better than other low-level techniques like edge detection are widely used in computer vision. Any segmentation algorithm should have two major properties. First one, separate important regions in an image and make a decision of what is important to segment and what is not. Also algorithm running time should be linear with the number of image pixels. Segmentation method should run in low time similar to low-level technique like edge detection. Although the last few years have seen considerable progress in eigenvector-based methods of image segmentation ([4], [6]), these methods are too slow to be Practical for many applications. The method improved in this paper uses in large image database applications. The segmentation technique developed here both captures certain perceptually important non-local image characteristics and is computationally efficient and can run at video frames rates. Unlike the classical techniques, our technique adaptively adjusts the segmentation criterion based on the degree of variability in neighboring regions of the digital image. This results in a method that, while making greedy decisions, can be shown to obey certain non-obvious global properties. We also show that other adaptive criteria, closely related to the one developed here,
Result in problems that are computationally difficult. [1, 3, 9]

There are researchers work on image segmentation and clustering, like Marco Saerens who study The Principal Components Analysis of a Graph, and Its Relationships to Spectral Clustering. [5]
And Yining Deng who make, a new approach to fully automatic color image segmentation, called JSEG. [7]
The organization of this paper is as follows. In the next section graph-based formulation has been described. Section (3) describes our proposed method. In section (4) some of important properties of our method. Finally, in section (5) we have been discussed the results, which is applied on some different images and conclusions.

## 2. Graph-Based Formulation

Image model is given as graphs $g_{mn} = \{V_n, E_m\}$ where (for example, gm is an aspect graph) $(V_n)$ is a set of vertices and $(E_m)$ is a set of arcs.Graph with vertices $v_i$ is part of $v_i \in V_n$ , the set of elements to be

segmented, and edges $(v_i, v_j) \in E$ corresponding to pairs of neighboring vertices. Each edge $(v_i, v_j) \in E$ has a corresponding weight $w(v_i, v_j)$, which is a non-negative measure of the difference between neighboring elements $v_i$ and $v_j$. In the case of image segmentation, the elements in $V$ are pixels and the weight of an edge is some measure of the difference between the two pixels connected by that edge (e.g., the difference in intensity, color, motion, location or some other local attribute). Segmentation $S$ is a partition of $V$ into components thus each component $C \in S$ corresponds to a connected component in a graph $G' = (V, E')$, where $E' \subseteq E$. This mean, any segmentation is induced by a subset of the edges in $E$. There are many ways to measure the quality of a segmentation but in general we want the elements in a component to be similar, and elements in different components to be dissimilar. Or in other words, that edge between two vertices in the same component should have relatively low weights, and edges between vertices in different components should have higher weights. Thus,

- Construct an edge from every element to every other.
- Associate with this edge a weight representing the extent to which the elements are similar.
- Cut the graph into connected components with relatively large interior weights "which correspond to clusters" by cutting edges with relatively low weights. As shown in Fig (1).
- The weights of the within-component edges will be large compared to the across-component edges.
- As a result, image segmentation problem becomes a graph cut problem.[7]

## 3. Image Segmentation Algorithm Using Graph-Edges

In this research we used color images thus we separate the color images into its three bands i.e. (red, green, blue) bands. And manipulate each band alone.

Some of images have noise in it, and that make the problem of image segmentation very difficult thus we must clear the image from these noise by smoothing it with low-pass filter. In this method we used Gaussian filter to smooth the image. The mathematical formula for this filter is below:

$$G(i, j) = e^{-(x_i - x_j)^2 / 2s^2} \quad \text{.....(1)}$$

Where $1 \le i, j \le n$

The useful of this filter not just smooth the input image, but also to close the components of the image. For evaluating whether or not there is

evidence for a boundary between two components in segmentation we define a predicate *P.* This predicate is based on measuring the difference between elements along the boundary of the two components relative to a measure of the difference among neighboring elements within each of the two components. The resulting predicate compares the inter-component differences to the within component differences and is thereby adaptive with respect to the local characteristics of the image data.

The *internal difference* of a component $C \subseteq V$ is the largest weight in the minimum spanning tree of the component. That means:

*Internal difference(C)=largest weight(e)* …… (2)

Where *e is* a short of edges of the image.

One intuition underlying this measure is that a given component *C* only remains connected when edges of weight at least *Internal difference(C)* are considered.

The difference between two components $C_1, C_2 \subseteq V$ is the minimum weight edge connecting the two components. That means,

$$Dif(C_1, C_2) = Min \; w(v_i, v_j) \text{.....(3)}$$

Where,

$v_i \in C_1$

$v_j \in C_2$

$(v_i, v_j) \in E$

To extract edges from the image, we used differences in all the directions not just in horizontal or vertical, to make sure it's a real edge not false.

If there is no edge connecting the two component $C_1, C_2$ we let $Dif(C_1, C_2)$ =infinity. This measure of difference could in principle be problematic, because it reflects only the smallest edge weight between two components. In practice we have found that the measure works quite well in spite of this apparent limitation. Moreover, changing the definition to use the median weight, or some other quantile, in order to make it more robust to outliers, makes the problem of finding a good segmentation NP-hard. Thus a small change to the segmentation criterion vastly changes the difficulty of the problem.

The region comparison predicate evaluates if there is evidence for a boundary between a pair of components by checking if the difference between the components, $Dif(C_1, C_2)$, is large relative to the internal difference within at least one of the components, *Internal difference* $(C_1)$ and *Internal difference* $(C_2)$. A threshold function is used to control the degree to which the difference between components must be larger than minimum internal difference. We define the pairwise comparison predicate as,

$P(C_1, C_2)$

=

$$\begin{cases} true.....Dif(C_1,C_2) \ \mathbf{f}\ MInt(C_1,C_2) \\ false.....else \end{cases}$$

…... (4)

Where the minimum internal difference, *MInt*, is defined as,

$MInt(C_1,C_2)$ =

$min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2))$ ..... (5)

The threshold function $\tau$ controls the degree to which the difference between two components must be greater than their internal differences in order for there to be evidence of a boundary between them (*P* to be true). For small components, *Int(C)* is not a good estimate of the local characteristics of the data. In the extreme case,

When $|C| = 1$, *Int(C)* = 0. Therefore, we use a threshold function based on the size of the component,

$\tau(C) = k/|C|$

Where $|C|$ denotes the size of *C*, and k is some constant parameter. That is, for small components we require stronger evidence for a boundary. In practice *k* sets a scale of observation, in that a larger *k* causes a preference for larger components. Note, however, that *k* is not a minimum component size. Weak components are allowed when there is a large difference between neighboring components.

It is possible to have the segmentation method prefers components of certain shapes, by defining a $\tau$ which is large for components that do not fit some desired shape and small for ones that does.

**Algorithm**

The following algorithm illustrates the above method.

We must separate the image into three bands (red, green, blue) if the input image is color image. And smoothes it by using Gaussian filters to reduce noise effect. We can change the smoothness of Gaussian filter by controlling *s* in Gaussian function.

Suppose you have a graph $g_{mn} = \{V_n, E_m\}$ with *n* vertices and *m* edges. the result image is a segmentation of $V_n$ into components $S = \{C_1, C_2, C_3, .....C_n\}$

**1.** Arrange *E by* increasing edge weight.

**2.** Start with segmentation $S^0$, where each vertex $v_i$ is in its own component.

**3.** Build $S^r$ given $S^{r-1}$ as follows.

Let $v_i$ and $v_j$ denote the vertices connected by the r-th edges in the ordering, i.e., $e_{r-1} = (v_i, v_j)$. If $v_i$ and $v_j$ are in disjoint component of $S^{r-1}$ and $w(e_{r-1})$ is small

compared to the internal difference of both those components, then merge the two components else do nothing. More formally, let $C_i^{r-1}$ be the component of $S^{r-1}$ containing $v_i$ and $C_j^{r-1}$ contains component$v_j$. If $C_i^{r-1} \neq C_j^{r-1}$ and $w(e_{r-1}) \leq MInt(C_i^{r-1}, C_j^{r-1})$ then $S^r$ is obtained from $S^{r-1}$ by merging $C_i^{r-1}$ and $C_j^{r-1}$(give them same color). Else $S^r = S^{r-1}$.(run this step for r = 1 to m).

**4. Properties of the algorithm**

We will describe the above algorithm in this section. And show that a segmentation resulted by our algorithm obeys the properties of being neither so coarse nor so smooth. The segmentation produced by the algorithm 1 dose not depend on which non-decreasing weight order of the edges is used. So any ordering can be changed into another one only by only swapping adjacent elements. Thus it is sufficient to show that swapping the order of two adjacent edges of the same weight in the increasing weight ordering dose not change the result produced by the algorithm 1. Let *e1* and *e2* be two edges of the same weight that are adjacent in some increasing weight ordering. Clearly if when the algorithm considers the first of these two edges they connect disjoint pairs of components or exactly the same pair of components, then the order in which the two are considered does not matter. The only case we need to check is when $e_1$ is between two components $C_1$ and $C_2$ and $e_2$ is between one of these components, say $C_2$, and some other component $C$. Another property of algorithm 1 is a segmentation $S$ is so coarse when there exists a proper refinement of $S$ that is not so smooth. That means if regions of segmentation can split and yield segmentation where there is evidence for a boundary between all pairs of neighboring regions, then the initial segmentation has so few components. The last property we discussed her is a segmentation S is so smooth if there is some pair of regions $C_2$ and $C_2$ for which there is no evidence for a boundary between them.

In order to define the complementary notion of what it means for a segmentation to be so coarse, we first introduce the notion of a refinement of segmentation. Given two segmentations $S_1$ and $S_2$ of the same base set, we say that $S_2$ is a refinement of $S_1$ when each component of $S_2$ is contained in (or equal to) some component of$S_1$. Also we say that $S_2$ is a proper refinement of $S_1$ when$S_2 \neq S_1$. Note that if $S_2$ is a proper refinement of$S_1$, then $S_2$ can be obtained by splitting one or more regions of$S_1$. When $S_2$ is a proper refinement of $S_1$ we say that $S_2$ is smoother than $S_1$ and that $S_1$ is

coarser than $S_2$ the running time of the algorithm 1 can be factored into two parts. First in Step 1, it is necessary to arrange the weights into increasing order. For integer weights this can be done in linear time using counting sort. Steps 2-4 of the algorithm take little time. In order to check whether two vertices are in the same component we use set-find on each vertex, and in order to merge two components we use set-union. Thus there are at most three disjoint-set operations per edge. The computation of *MInt* can be done in constant time per edge if we know *Int* and the size of each component. Maintaining *Int* for a component can be done in constant time for each merge.

**5. Results and Conclusions**

We used an edge weight function based on the absolute intensity difference between the pixels connected by an edge,

$$w((v_i, v_j)) = |I(i) - I(j)|$$

Where $I(i)$ is the intensity of the pixel *i*. We use a Gaussian filter to smooth the image slightly before computing the edge weights, in order to compensate for digitization artifacts. We always use a Gaussian with $s = 0.8$, which do not affect on the input image but helps remove noise.

For color images we run the algorithm three times, once for each of the red, green and blue color bands, and then combine these three sets of components.

We put two neighboring pixels in the same component when they appear in the same component in all three of the color plane segmentations. Alternatively one could run the algorithm just once on a graph where the edge weights measure the distance between pixels in some color space, however experimentally we obtained better results by intersecting the segmentations for each color plane in the manner just described.

There is one runtime parameter for the algorithm, which is the value of *k* that is used to compute the threshold function $\tau$. Recall we use the function

$$\tau(C) = k/|C|$$

Where $|C|$ is the number of elements in component C. Thus *k* effectively sets a scale of observation, in that a larger *k* causes a preference for larger components. Thus, in 300 x 266 images like house and bear images we used *k*=500 as shown in table 1.

The first image in Fig (2) shows a house image. Note that windows of the house segmented efficiently by the algorithm and the small variation in the sky and grassy areas colors neglected. That means the algorithm makes good decisions to segment the image objects and neglect the isolated points. Also note that when we increase the number of connect components from 50 to 150 the number of output connected components increase also. If we need to get more objects from the image we can increase the number of connected components. We used more than on image in this paper to prove that the algorithm works well with different

kinds of images. We used image have more details like bear image Fig (3) and the algorithm still satiable to extract the important objects in the image and neglects the unimportant ones.

For Fig (4) we get good result we make number of connected component equal to 20 and minimum size of components equal to 50 for fruit image as shown in Fig (4) a and b. And we got good result when we make number of connected components equal to 20 and minimum size of components equals to 20 as shown in Fig (4) c and d. See that even if there is small changing in intensity, the algorithm could capture the object and segmented it.

After the algorithm capture the object in an image the algorithm give random color to each object so that we could segment it by its color.

We conclude that proposed segmentation algorithm makes simple greedy decisions, and yet produces segmentations that obey the global properties of being not so coarse and not so fine according to a particular region comparison function. The method runs in little time for $m$ graph edges and is also fast in practice, generally running in a fraction of a second.

The pairwise region comparison predicate we use considers the minimum weight edge between two regions in measuring the difference between them. Thus algorithm 1 will merge two regions even if there is a single low weight edge between them.

We have illustrated image segmentation algorithm 1 with two different kinds of graphs. The first of these uses the image grid to define a local neighborhood between image pixels, and measures the difference in intensity (or color) between each pair of neighbors. The second of these maps the image pixels to points in a feature space that combines the (x; y) location and (r; g; b) color value. Edges in the graph connect points that are close together in this feature space. The algorithm yields good results using both kinds of graphs.

**References**

[1] S. Arya and D. M. Mount, Approximate nearest neighbor searching, Proc. 4th Annual ACM-SIAM Symposium on Discrete Algorithms, pp (271-280) (1993)

[2] Chaur-Chin Chen, Introduction to Cluster Analysis, IEEE presses, (2002)

[3] I. Jermyn and H. Ishikawa, "Globally Optimal Regions and Boundaries as Minimum Ratio Weight Cycles", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, pp (1075-1088) (2001).

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

[5] M. Saerens and F. Fouss, "the Principal Components Analysis of a Graph, and Its Relationships to Spectral Clustering", IEEE presses (2004),.

[6] Y. Weiss, "Segmentation using eigenvectors a unifying view", University of California, San Diego (2002).
[7] S. Candemir, "Graph-based Algorithms for Segmentation", Algoritmalar ve Teorisi Alannda Özel Konular (2005).
[8] K. Yoon, "Segmentation", presentation from internet (2002).
[9] Y. Deng and H. Shin," Color Image Segmentation", IEEE presses (2001).
[10] X. Yang, "Image Segmentation", IP seminar for fall semester (2004).

**Table (1) Effect of input coefficients on the number of segmented components**

| Image Name | k | Min size of comp. | *S* | Num_ccs | Segm Comp. |
|---|---|---|---|---|---|
| House | 500<br>500 | 20<br>20 | 0.8<br>0.5 | 50<br>150 | 135<br>170 |
| Bear | 500<br>500 | 20<br>50 | 0.8<br>0.8 | 50<br>20 | 89<br>62 |
| Soccer | 500 | 20 | 0.8 | 20 | 154 |
| Fruit | 500 | 50 | | | 73 |

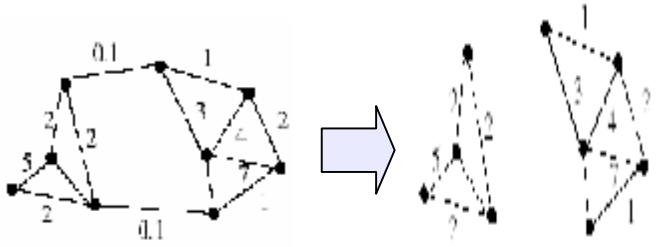


**Figure (1): separate similar edge component according to weights.**

a. Input color Image (house).

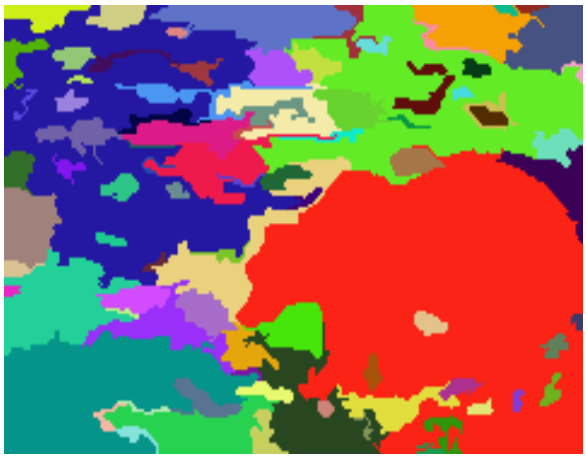
b. Segmented image (k=500, sigma=0.8, num_ccs=50)

c. Segmented image (k=500, sigma=0.8, num_ccs=20)

**Figure (2): House Image segmentation by graph-edges algorithm**

a. input color image (bear).

c. Segmented image (k=500, sigma=0.5, num_ccs=150



b. Segmented image (k=500, sigma=0.8, num_ccs=50)

Figure (3): Bear Image segmentation by graph-edges algorithm

a. Input color image (soccer).

b. Segmented image (k=500, sigma=0.8, num_ccs=20)

c. Input color image (fruit).

d. Segmented image (k=500, sigma=0.8, num_ccs=20)

**Figure (4): Soccer & fruit Images segmentation by graph-edges algorithm**