IJCAT - International Journal of Computing and Technology, Volume 3, Issue 5, May 2016
ISSN : 2348 - 6090
www.IJCAT.org

Effective Image Segmentation using Graph Base Method

[1] Yeshwant Deodhe, [2] Shashant Jaykar, [3] Rohit Himte

[1,2,3] Assist.Prof.Electronics engg.deptt. Rajiv Gandhi college of engg & research. Nagpur, RTMNU,(M.S) India.

Abstract - Graph based image segmentation techniques are considered to be one of the most efficient segmentation techniques. Which are mainly used as time & space efficient methods for real time applications Image segmentation is the first step of image Mining. Due to the limited resources of the Sensor devices, we need time and space efficient methods of image segmentation In this paper, we propose an improving to the graph Base image segmentation method. Already describe in the literature and considered as the most effective method with satisfactory segmentation result. This is the preprocessing step of our online image Mining Approach. We contribute to the method by Re-defining the Internal difference used to define the property of the Components and threshold function the conducted Experiment demonstrates the efficiency and effectiveness of the adjusted method.

Keywords **- Graph Base Image Segmentation Method, New Threshold Function, Sensor Devices.**

1. Introduction

Sensor devices are widely used for monitoring purposes. Useful knowledge is expected to be obtained from the image sequences taken by sensor devices such knowledge may include image data patterns, image data relationships, or other patterns not explicitly stored in the images and between images and other alphanumeric data [9]. Image mining techniques are commonly used to extract such knowledge. However, **an** image is simply a collection of pixels. It is quite difficult to extract the knowledge (the high-level information) Directly from the level pixel collection.Image segmentation is usually the first step in image Mining [6], aiming to partition the image into perceptual meaningful regions. As a result, it is essential to have an image segmentation method as the preprocessing step of any image mining approach. As we are developing an online image mining approach; our initial findings are centered around **an** efficient and effective image segmentation method. In this paper, we describe the basic elements of the image segmentation problem, and compare the existing segmentation approaches in terms of image features, similarity measurement and the segmentation algorithm. Most prior works on image segmentation focus on improving the segmentation results. However, as we put the image segmentation problem in the context of sensor devices with limited resources, we should put equal emphasis on efficiency (time and space complexity) and efectiveness (quality of the segmentation result). We analyze the possible techniques on each element that can be used to improve the efficiency of image segmentation. As the methods described in the literature are concerned, the graph-based image segmentation method proposed recently by Felzenszwalb ***et a1*** [4] is the most efficient with satisfactory segmentation result. However, we have successfully identified several drawbacks and weaknesses of this method. Consequently, we have developed a more effective and efficient method that handles the identified drawbacks and weakness .Explicitly, the method of Felzenszwalb ***et at*** [4] uses internal difference ***as*** a measurement to describe the property of components. It is one of the key elements to guide the segmentation process. In order to make the algorithm fast, Felzenszwalb ***et a1*** [4] defined **internal difference** on the extreme values, which is not an accurate description of the components. So, in this paper we re-define the **internal difference** such that it gives a more accurate and stable description of the components without increasing the complexity of the algorithms. Furthermore, the method of Felzenszwalb et a1 [4] uses a threshold function that require parameter K to control the size of the segmented region.

However, no quantitative relationship between the value of k and the segment size is given. It's very difficult for users to choose an appropriate parameter value for an expected segment size. In this paper, first we analyze how the parameter k affects the segmentation process; and then we identify the stop merging condition with respect to k. Inspired by the stop merging condition, we re-define the threshold function Such that parameter k becomes independent of the edge weight scale. Based on this, we further improve the threshold function by taking the

number of components into consideration. When there are more components than expected, the threshold function "encourage" merging. When there are fewer components than expected, the threshold function "discourage" merging. The reported experimental results demonstrate the efficiency and effectiveness of the proposed approach. The rest of the paper is organized as follows. Section 2 describes image segmentation for sensor monitoring applcations. Section **3** presents the basic improvement to the graph-based method. Section 4 describes the implementation. Section 5 reports the experimental results. Section 6 presents the discussion. Section **7** is summary and conclusions.

2. Image Segmentation for Sensor Monitoring Applications

Most prior works on image segmentation focus on improving the segmentation results as long as the time complexity is not worse than polynomial. However, as we put the image segmentation problem in the context of sensor devices with limited resources, we should put equal emphasis on efficiency (time and space complexity) and effectiveness (quality of the segmentation result). In a sensor monitoring application, the images are constantly streaming in, the segmentation process is expected to compete in real time (a fraction of a second, at most few seconds), we need a linear time (at least nearly linear time) method that can produce satisfactory results.

The techniques to speed up the segmentation process can be classified as follows:
1. Use simple features [ll];
2. Reduce the resolution of the image [l0];
3. Use fast transform algorithm (like one-pass Wavelet decomposition [18]);
4. Use efficient similarity measurement [4];

Using simple features is definitely a good choice if the image itself is quite simple; for example, for images consisting of solid color regions. Bruce *et* al [ll] proposed a fast and cheap image segmentation method for interactive robots with O(m) time complexity, where m is the number of pixels in the image. However, real world images are seldom such simple as a combination of constant regions. More often they contain both texture and non-texture components. Using one simple local feature usually makes little sense

Due to the advantages of wavelet transform, it is widely used in image segmentation. Kim ***et al*** [l0] proposed a fast image segmentation method based on multi-resolution analysis and wavelet. Multi-resolution analysis is to view the original signal at various levels of resolution. When applying a wavelet transform to the original signal, the trend of the signal is approximated by the scaling coefficients. The scaling coefficients generated at level ***m*** give the approximation of the original signal at & of the original resolution. The following processing is based on the low-resolution signal. It is much faster than on the original signal. However, resolution-reduction misses some information; it simply sacrifices effectiveness in exchange of efficiency. The wavelet transformation brings extra overhead to the segmentation process. In the context of sensor devices with limited resources, we can not afford the extra cost. The recently proposed one-pass wavelet decomposition technique [18] might help addressing this problem. Furthermore, finding appropriate similarity measurement is an important way to improve the efficiency. In image segmentation methods, the similarity measurement is the key factor in determining the segmentation quality. Although simple measurement usually means fast processing, it produces poor results under most circumstances because simple measurement only considers local properties of the pixels. However, robust measurement, which considers the global properties of the image often implies high computation [2]. So, it is a big challenge to choose an appropriate measurement to enable fast processing while maintaining acceptable results. The recently proposed graph-based method [4] uses a good measurement, which looks like a kind of "upper bound"; it maintains the algorithm at complexity of O(m1ogm). But, a slight relaxation of the condition would make the problem NP-hard. More important, this measurement captures the global properties of the image and produces good result. To the best of our knowledge, this method [4] is the fastest one with satisfactory segmentation results. It is claimed to be suitable for real time applications. In the following section, we analyze the key element of its similarity measurement, namely the threshold function, and propose an interesting improvement.

3. Improvements to the Graph-Based Method

The image segmentation algorithm described in [4] starts with a trivial segmentation, with each component containing one pixel, and repeatedly merges pairs of components based on the following merge condition:

$$\text{Di f f (Cl, C2)} \leq \text{Int(C1)} + \text{T(C1) and}$$
$$\text{Dif f (Cl, C2)} \leq \text{Int(C2)} + \text{T(C2)} \qquad (1)$$

Where Diff (Cl,C2) is the difference between components Cl and C2; Int(Cl) and Int(C2) are the internal differences of Cl and C2, respectively; T (C) = k/|Cl is the threshold function. Parameter k controls the size of the components in the segmentation. Felzenszwalb et al [4] state that large k favors large regions; but they do not give any quantitative relationship between k and the size of the

regions. Therefore, it is hard for the users to give an appropriate value of parameter k for an expected component size. For example, they use two different k values, 150 and 300. No explanation is given on why 150 or 300 are chosen instead of other Values. If the k value is determined by trial and error for each particular image, this method becomes infeasible in real time applications, mainly in the context of sensor devices.

In Condition (I), the internal difference of a component, Int(C), is defined to be the largest edge weight of the minimal spanning tree. The difference between two components, Diff (Cl, C2), is defined to be the smallest edge weight between them. The only purpose to take extreme values in these definitions is to make the algorithm fast, with complexity O(n1ogn). The extreme values are not accurate descriptions of the components. Moreover, they are sensitive to noise. Although it is difficult to improve the definition of Diff (Cl, C2) as it has been proven that a slight relaxation would turn the problem NP-hard, we may improve the definition of Int(C) such that it gives a more accurate description of the component and reduce the influence from noise pixels. The only requirement is that the computation of Int(C) must take constant time at each step.

3.1 Analysis of the Threshold Function

First, we analyze how the threshold function affects the segmentation process. In the algorithm, the edges are sorted in a non-decreasing weight order. This means the weight of edges connecting two components is always larger than the weight of edges inside each Component. The threshold function T(C) = k/|C| is necessary for the algorithm to work because without the threshold function the difference between two components Cl and C2 is always larger than the internal differences of Cl and C2. As the edges are processed in a non-decreasing eight order, the edge causing the merge of Cl and C2 must be the smallest weighted edge connecting C1 and C2; the weight of this edge is Di f f (Cl, C2). After merging Cl and C2, this edge becomes the largest weighted edge in the merged component Cl U C2; we have Diff (Cl, C2) = Int(Cl U C2). Similarly, Int(C1) and Int(C2) are weights of the last processed edges in forming Cl and C2, respectively.
Let E = {el, e2 . . . en) be the set of edges in nondecreasing weight order, W(ei) ≤ W(ei + 1) for i = 1,2.. . n - 1. Condition (1) can be rewritten as

K ≥ |C1| (W (ep) - W (ei)) and

K |C2| (W (ep) – W (ei)) (2)

where p > i and p > ***j*** **,** e, is the current edge connecting Cl and C2, and ei is the last processed edge in forming Cl, and ej is the last processed edge in forming C2 .Lemma 1 in **[4]** states that if a component Cl does not merge with component C2 as a result of the fact that Diff(C1,C2) > Int(C1) + k/ |Cl| , Cl will not merge with any other components. Based on our rewritten merge condition, it is equivalent to say that:

If
K < | C| (W (ep) – W (ei)) (3)

then component Cl will not merge with any other components and it must be in the final segmentation because the weight of edges after e, is greater than W(e,) .So we can generalize Lemma 1 from **[4]** and get stop merging condition for all the components.
Theorem 1: Assume at certain step, the minimum size of current components is | | |C|min, and the current edge under consideration is et; then the stop-merging condition is:

K<|C|min(W (et) - W (et-1)) (4)

In other words, if Condition (4) is satisfied, then all current components are in the final segmentation. We do not need to consider the edges after the current edge and the algorithm can stop.
Proof: : For an arbitrary pair of components Cx, and Cy, we have |Cx| ≥ |Cx|min and
|Cy| ≥ |C|min- Let ex be the largest weighted edge in Cx, and ey the largest weighted edge in Cy. As ex and ey were processed before et, W(ex) ≤ W(et-1) and W(ey) ≤ W(et-1).

Let exy, be any unprocessed edge connecting Cx, and Cy, W(exy) ≥ W(et). We have k < |C|min(W(et) -W(et-1))≤ |Cx|(W(exy)- W(ex)) and k<|C| min(W(et)- W(et-1))≤ |Cy|(W(exy)-W(ey)).

From Lemma 1, both Cx and Cy, will not merge with other components. Since Cx, and Cy, are arbitrarily chosen, no merge will happen in the following steps. By analyzing Theorem 1, we may think of using Condition (4) as the stopping condition of the algorithm. If Condition (4) is satisfied, the edges after the current edge will not cause merges any more, and the algorithm can stop without affecting the final segmentation.Theoretically, this is true. However, as real world images are concerned, there exist noisy pixels, which do not merge with any other components. As a result, some single-pixel components exist in the final Segmentation. Even such noisy pixel could make the stop-merging condition practically meaningless. Although the stop merging condition can not be directly applied in practice, it gives a quantitative

description of parameter k in terms of component size and edge weight differences. From this, we know that the size of components in the final segmentation depends on parameter k and the difference in edge weight. As the edge weight is defined based on properties of the pixels, users would not know the magnitude of edge weights. For a given image, different definitions of the edge weight require different k values to produce the same result. Therefore, to determine an appropriate parameter k, users have no idea of the relationship between component size and a particular k value. In next section, we redefine the threshold function T(C) such that parameter k is independent of the scale of edge weight. That is, no matter how the edge weight is scaled, same k value produces same segmentation.

3.2 New Threshold Function

Given an image represented by a weighted graph, we denote the largest edge weight by Wma,, the smallest edge weight by Wmi,. We define the threshold function as:

T(C) = K (Wmax- Wmin) / |C| (5)

Based on Condition ***(5),*** we may rewrite the merge condition as:
K ≥ |C1| (W (ep) – W (ei) / (Wmax – Wmin)
and
K ≥ |C2| (W (ep) – W (ej) / (Wmax – Wmin) (6)

In Condition (6), the current edge weight difference is normalized by total edge weight differences. Therefore, parameter k becomes independent of the edge weight scale. More important, in the sorted-edge list, Wmax - Wmin is equal to W (en) - W (el). The new

Threshold function specified in Condition (5) can be computed in constant time; and thus would not increase the complexity.Given an image; we expect a moderate number of components in the segmentation. The number of components should also be considered in the threshold function. In the segmentation process, when there is a large number of components, the threshold function should "encourage" merging. When the number of components decreases, the threshold function should "discourage" merging. In other words, the more components there are the stronger evidence we need for a boundary between two components. Based on this idea, we modified the threshold function as:

T(C) = (Wmax – Wmin / |C|) * (Num / K) (7)

Where *Num,* is the number of components. Parameter k can be regarded as the expected number of components. Large k produces more components. Note that k does not equal to the number of components in the segmentation. The algorithm starts with *Numc* = |V| (|V| is the number of pixels), and each merge decreases *Num,* by one. Computing Function (7) takes constant time; it does not increase the complexity. Based on Function (7), the stop merge condition for a component C becomes:

K>((Wmax- Wmin)Num / |C| (W (ep)- W(ei)) (8)

3.3 New Internal Difference

In this section, we redefine the internal difference, Int(C), such that it gives a more accurate description of component C. We define the internal difference of component C as the average edge weight in the minimal Spanning tree of C. Formally,

$$Int(C) = 1/N * \sum_{e \in MST(C,E)} W(e) \quad (9)$$

Where *N* is the number of edges in the minimal spanning tree of C.

In Equation (9), Int(C) takes the average value instead of the maximum value. Few noisy pixels would not have great impact. In other words, Equation (9) reduces the influence from noisy pixels. It is more stable than the original definition. More important, it does not increase the time complexity. As we know, n-nodes tree has n - 1 edges; here we have N = ICI - 1. Since each component ***C*** maintains its size ICI, computing Equation (9) takes constant time.

4. The Implementation

The author of [4] provided an implementation for the grid graph. In this implementation, the image is smoothed by a Gaussian filter to remove noise. The edge weight is defined to be the color difference between neighbor pixels. After the graph is segmented, the segmentation result is post-processed by combining all the components with size below a user-specified Threshold. We borrowed some of the code from the author's implementation, like the Gaussian filter and the data structure to store the components. We did the following modifications:

1. We used the threshold function specified in Equation (7);
2. We used the internal difference specified in Equation (9);
3. We implemented the nearest neighbor graph: each pixel has edges connecting 10 nearest neighbors on the 5-dimensional feature space (r, g, b, x, y), where the triplet (r, g, b) represent the color value of the pixel and the pair *(x,*y) represent the location of the pixel. The edge weight

is the Euclidean distance on that space. For efficiency, we use the nearest neighbors within distance 10.

Finally, in our implementation, we did not do any post-processing and achieved outstanding results as reported next in Section **5.**

5. Experimental Results

In the experiment, we compare our method with the work described in [4], which is the most fast one with satisfactory segmentation result; it is also the most similar because ours is mainly based on handling the weaknesses of the approach described in [4]. we have proven that our method has the same time complexity with that described in [4], we do not compare the running time of the the approaches. We only compare the segmentation result and the segmented images. We use the test images from the Most of the images are outdoor pictures with texture components. In order to compare the actual segmentation performance with the work of [4], we use the same piece of code for image pre-processing, i.e., a Gaussian filters to remove image noise.

For the same purpose, our Implementation does not perform any post-processing. We compare our segmented images before the post processing stage. In the experiment, we compare the images in both grid graph and nearest neighbor graph settings. For grid graph, each pixel has edges connecting its eight physically adjacent neighbors. The edge Weight is defined to be the color difference between the neighboring pixels. Since the authors of [4] provided an implementation for the grid graph, we use as much the same code as possible and only modify the threshold function as specified in Function (7) and the internal difference specified by Equation (9).For nearest neighbor graph, each pixel has edges connecting 10 nearest neighbors on the 5-dimensional feature space ***(r,*** g, ***b,*** x, y), where (r, g, b) represent the color value of the pixel and (x, y) represent the location of the pixel. The edge Weight is the Euclidean distance on that space.

In establishing the edges connections, we use a simple nearest neighbor search algorithm and search nearest neighbors within distance 10.We found that for both methods, the nearest neighbor graph setting produces much better results than the grid graph setting. Here, we compare three of the segmented images using nearest neighbor graph. The first example is shown in Figure la; the input is the ground-water image. According to human perceptton, We expect that the sky in the segmented image is identified as one component, the buildings in distance is a component, and the water is a component. We do not expect the whole piece of land close to the viewer to be one component because the difference between various parts is perceptually significant (e.g., the difference between the green grass from other parts). Figure l b is the segmented image produced by the method described in [4] with parameter value 300. In Figure l b , the sky is still in two parts, but the green grass on the land has been merged with the surrounding parts. In order to make the sky one component, we need to increase the parameter value. Figure l is the segmented image using 400 as the parameter value. However, to separate the green grass from the other Part, we need to decrease the parameter value.

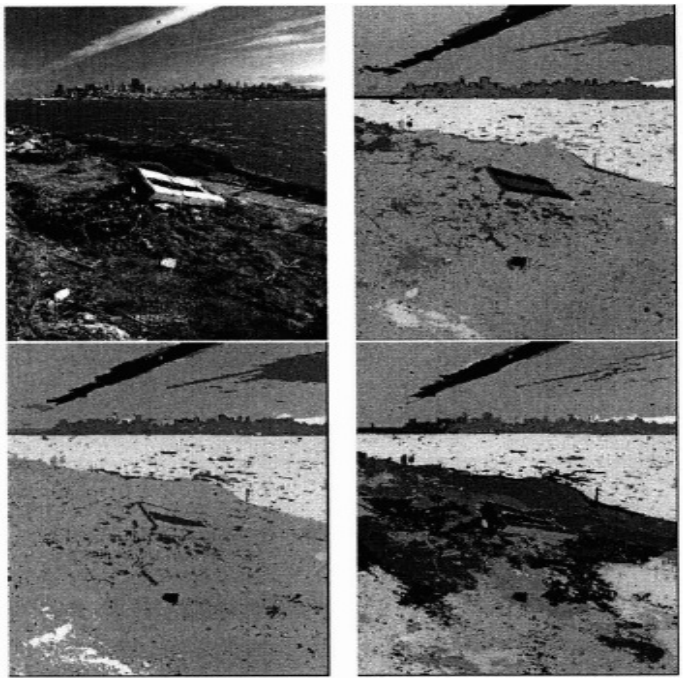

Figure 1. Ground-water: a) The original input image; b) Segmented image by the original method with =300; c) Segmented image by the original method with k=400; d) Segmented image by our method.

Therefore, it is impossible to produce a satisfactory result no matter what parameter value is chosen. The reason is that the method described in [4] uses the maximum edge weight to describe a component, each time a pixel is merged with a component; the newly added edge weight is used to describe the component. In this example, although the difference between the green grass with surrounding parts on the land is perceptually significant, the boundary is not clear. There exist pixels that are between the typical pixels of the two adjacent parts. These pixels work as a transition and incorrectly merge the two perceptually different components. Our average edge weight description of component is more stable. Figure Id shows the segmented image by our method. Here, the sky is one component and the green grass is separated from the surrounding parts on the land.

6. Summary and Conclusion

In this paper, we compared the existing segmentation approaches in terms of image features, similarity measurement and segmentation algorithm and discussed the possible techniques to improve the efficiency of image segmentation for sensor monitoring applications. We analyzed the graph-based image segmentation method described in [4], which is reported in the literature as the fastest one with satisfactory segmentation result we proposed major improvement to this method.. We re-defined the internal difference to give a more accurate and stable description of components with no increase of time complexity .We re-define the threshold function such that it Can adaptively guide the segmentation process in-dependent of the edge weight scale. Finally, the reported experimental results on a well Learn known database of images demonstrate the effective-ness and efficiency of the proposed approach.

References

[1] Y.A.Deodhe, “wavelet based segmentation of remotely sensed images using graph based method” International conference on computer applications ICCA 2012.Pondicherry

[2] Z. Wu and R. Leahy, "Optimal graph theoretic approach to data clustering: theory and its application to image segmentation," IEEE Trans. PatternAnalysis and Machine Intelligence, Vo1.15, No.11pp.1101-1113, 1993

[3] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," IEEE Trans. Pattern Analysis and Machine Intelligence, Vo1.22, No.8, pp.888-905, 2000

[4] Y. Weiss, "Segmentation using eigenvector unifying view," Proceedings of International Conference on Computer Vision, pp.975-982, 1999.

[5] P.F. Felzenszwalb and D.P. Huttenlocher, "Efficient Graph-Based Image Segmentation," Interna tional Journal of Computer Vision, Vo.59, No.2, 2004.

[6] W. Hsu, M.L. Lee, and J. Zhang, "Image Mining:Trends and Developments," Journal of IntelligentInformation Systems, Vo1.19, No.1, pp.7-23, 2002.

[7] B. Kim, J. Shim and D. Park, "Fast Image Segmentation based on Multi-resolution Analysis and Wavelets," Pattern Recognition Letters, Vo1.24,N0.16, pp.2995-3006, 2003.

[8] A.C. Gilbert, Y. Kotidis, S. Muthukrishnan and M.J. Strauss, "One-Pass Wavelet Decomposition of Data Streams," IEEE IPrans Knowledge and Data Engineering, Vo1.15, No.3, 2003.

[9] B. Kim, J. Shim and D. Park, "Fast Image Segmentation based on Multi-resolution Analysis and Wavelets," Pattern Recognition Letters, Vo1.24,N0.16, pp.2995-3006, 2003.

[10] J. Bruce, T. Balch and M. Veloso, "Fast and Cheap Image Segmentation for interactive Robots," Proceedings of the Workshop on InteractiveRobotics and Entertainment, 2000.

[11] H. Choi and R.G. Baraniuk, "Multi-scale Image Segmentation Using Wavelet-Domain Hidden Markov Models," IEEE Trans. Image Processing, VOl.lO1 No.9, 2001.

Bibliography

Prof. Yeshwant A.Deodhe ,Assist. Professor, Deptt. Of Electronics,RGCER, Nagpur has competed B.E. Electronics in 1996 from Nagpur,university . M.Tech Electronics in 2011 from Nagpur university. Five Research publications in IEEE international conferences in India.and Three papers in international journal in India in the Area of specialization is VLSI and communication engineering and Image processing.

Prof.Shashant Jaykar Assist. Professor, Deptt. Of Electronics,RGCER, Nagpur has competed B.E. Electronics in 2008 from Amarawati,university . M.Tech Electronics in 2011 from Nagpur university. Six Research publications in IEEE international conferences in India.in the Area of specialization is VLSI and Signal processing.Ten papers in International journal in India.