A Faster Graph-Based Segmentation Algorithm with Statistical Region Merge

Ahmed Fahad and Tim Morris

School of Informatics, University of Manchester
Manchester, UK, M60 1QD
ahmed.fahad@postgrad.manchester.ac.uk,
tim.morris@manchester.ac.uk

Abstract. The paper presents a modification of a bottom up graph theoretic image segmentation algorithm to improve its performance. This algorithm uses Kruskal's algorithm to build minimum spanning trees for segmentation that reflect global properties of the image: a predicate is defined for measuring the evidence of a boundary between two regions and the algorithm makes greedy decisions to produce the final segmentation. We modify the algorithm by reducing the number of edges required for sorting based on two criteria. We also show that the algorithm produces an over segmented result and suggest a statistical region merge process that will reduce the over segmentation. We have evaluated the algorithm by segmenting various video clips Our experimental results indicate the improved performance and quality of segmentation.

1 Introduction

The problem of image segmentation remains a great challenge for computer vision. A myriad of segmentation techniques have been developed over the years with considerable progress in improving execution time and efficiency. The literature is full of segmentation techniques some of which have showed promising segmentation results. The Normalized Cut algorithm [1] constructs a graph in which the nodes represent the pixels in the image and labelled edges represent affinities between nearby pixels. The image is segmented by minimizing the cost of cutting the graph into sub-graphs where the intra affinity within the sub-graph pixels is greater than the inter affinity of pixels between the sub-graphs. However, this algorithm does not run in linear time with regard to the number of pixels, which makes it an unsuitable algorithm for real time processing of realistic sized images. A fast multi-scale algorithm [2] uses an algebraic multi-grid to approximate the solution to the normalized cut problem using recursive graph coarsening in order to produce an irregular pyramid of region encoding. Cour, et. al. [3] present a multi-scale spectral image segmentation algorithm that solves a cross scale constraint matrix which processes the different spatial scales in parallel by forcing the system to seek an 'average' segmentation across all scales. Macaire, et. al. [4] proposed a pixel classification scheme that constructs a set of classes based on the analysis of the spatial-colour compactness degree which indicates the likelihood of a colour subset, defined by a colour domain, corresponding to an actual region in the

G. Bebis et al. (Eds.): ISVC 2006, LNCS 4292, pp. 286 – 293, 2006.

image. This is done by using a selection procedure that will identify a colour subset as a region in the image by maximizing an objective function which in turn will maximize the connectedness and colour homogeneity properties of the colour subset. The JSEG algorithm proposed by [5] applies a criterion of good segmentation that measures ratio of the distances between different classes to the distances within each class. Mean-Shift segmentation [6] estimates the gradient of the probability density of a colour occurring jointly in the spatial and colour domains of the colour image such that each pixel is associated with the closest local mode of the density distribution. An algorithm that integrates the watershed algorithm with the chain code is used, it simulates raining to generate connected components using chain codes and then simulates flooding to label catchment basins by tracing the chain codes [7]. Seeded Region Growing (SRG), which was proposed by Adams and Bischof [8], starts with assigned seeds, it then grows regions by merging a pixel with the nearest neighbouring seed region. [9] Showed how to automatically select initial seeds as pixels having high similarity to their neighbourhoods which do not fall on the boundaries of two regions. Felzenszwalb and Huttenlocher [10, 11] presented a graph segmentation algorithm based on pairwise region comparison. The comparison predicate considers the minimum weight edge between two regions as an external difference while the minimum spanning tree edges of each component is considered as an internal difference. The predicate merges two regions if their internal difference exceeds their external one. Our algorithm exploits the fact that once two regions are merged, the in-between edges of their boundaries are no longer required and hence the number of edges required for sorting is reduced. We also exploit the fact that the majority of regions' internal weights exceed a fraction of the edge weights which allow us to introduce a preprocessing step to merge regions with quite small edge weights. Although this preprocessing step violates the theoretical aspect of the segmentation algorithm, experimental results show that it speeds up the process and with the statistical region merge it produces better segmentations.

The rest of the paper is organized as follows: in the next section, a brief review of the Felzenszwalb and Huttenlocher graph algorithm is presented. This is followed by a description of our faster segmentation algorithm in section 3. In section 4 we use a statistical region merge predicate to minimize the oversegmentation of the graph algorithm. In section 5, experiments on a variety of image data sets are presented with a conclusion.

2 Graph-Based Image Segmentation

Felzenszwalb and Huttenlocher [10] presented a segmentation algorithm that represented the image as an undirected graph $G = (V,E)$ where each node v_i belonging to V corresponds to a pixel in the image and pairs of pixels are connected by edges $e_{ij} \in E$, $i \neq j$. A segmentation, S, is a partition of V into components such that each component (or region) $C \in S$ corresponds to a connected component in a graph $G' = (V, E')$, where $E' \subseteq E$. The internal difference of a component $C \subseteq V$ is the largest weight in the minimum spanning tree of the component, MST(C, E). That is,

$$Int(C) = \max_{e \in MST(C,E)} w(e). \quad (1)$$

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

$$Dif(C1, C2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w(v_i, v_j). \quad (2)$$

The segmentation evaluates a predicate for the evidence of a boundary between a pair of components by checking whether the difference between the components, $Dif(C_1,C_2)$, is large relative to the internal difference within at least one of the components, $Int(C_1)$ and $Int(C_2)$. A Euclidean colour distance metric is used to represent the edge weight where a small value indicates strong affinity between pixels. The number of edges in E, using 8-adjacency graph, is four times the number of pixels in an image and the overall time complexity is E log E for sorting. Knowing that most of the computation time is devoted to sorting the edges, we show in the next section how to reduce the number of edges required for sorting. We will refer to this algorithm as F&H.

3 Method

F&H's main processing step is building a minimum spanning tree, MST, using Kruskal's algorithm by sorting the graph edges. Kruskal's algorithm starts by sorting all edges in non-decreasing order then scans the edges in this order adding an edge to the MST if either of its vertices do not belong to the MST; otherwise the edge is not included in the final MST. Similarly, two regions that are merged will mean two MSTs will merge into one MST. There is at least one edge connecting two pixels/vertices from different regions/MSTs and the edge with the minimum weight is tested for merging. Once the merge occurs, the two MSTs merge into one MST by adding the edge with the minimum weight and excluding all remaining edges connecting the different MSTs. We exploit the fact that once we merge two regions, the edges connecting their boundaries are no longer required; therefore they can be removed from the set of edges E'. The algorithm computes the graph edge weights as a distance measure in the RGB colour space instead of the absolute intensity difference because experimentally this gives better results [10]. The weights are real values. The edge weights can be sorted into buckets using the integral part of the weight as a bucket index, therefore each bucket holds greater edge weights than the previous one. In other words, the first bucket holds edge weights that are smaller than the other buckets. The weights in a bucket are not sorted. The graph algorithm is applied as normal to the first bucket's weights by sorting the edge weights by non-decreasing order and then applying the merging predicate. Next, all edges in the next bucket whose ending vertices belong to the same region/MST derived from the previous merging step are removed from the bucket (Fig. 1 presents a simple example illustrating the edge removal step). Sorting and merging follow the edge removal step and the

(a) (b) (c) (d)

Fig. 1. Example of edge removal (a) Four vertices belong to different sets (b) The four vertices are merged into two different sets (c) Thick line indicates edge about to merge (d) dashed edges representing removed edges in the next steps

algorithm repeats over all the buckets. We notice that the method does not affect the quality of segmentation i.e. the segmentation is not too coarse and not too fine.

In addition, we add a preprocessing step that will further speedup the algorithm, however this step may affect the quality of segmentation. Pixels belonging to regions that have negligible intensity variation can be merged into small components, thus reducing the number of edges that are required to be sorted. Hence, we perform a region merging process where any two vertices are merged into one component if their edge weight is less than a threshold. The threshold value is quite small in order not to affect the segmentation too much i.e. not too fine and not too coarse segmentation. We show in the experiments section, that this step hardly changes the quality of segmentation. In summary, the segmentation algorithm is:

0. Set each vertex to represent its own component.
1. If the edge weight e connecting any two components is less than a threshold then merge the two components and remove edge e from E.
2. Else add the edge weight to the bucket with the index value equal to the weight without the fraction.
3. Sort the bucket with the minimum index.
4. Let v_i and v_j denote the vertices connected by the edge e_k. The component to which v_i belongs is merged with the component to which v_j belongs if the edge weight e_k is less than or equal to the internal weight of either component and the two vertices belong to different components.
5. Remove all edges from next bucket if the vertices at the edge's two ends belong to the same component.
6. Sort the bucket in non-decreasing order and process step 4.
7. Iterate until all buckets are processed, or until the next bucket's index is greater than all the component's internal weights.
8. Build a spatial adjacency graph and apply statistical region merge.
9. Return the set of components.

4 Region Merge

The segmentation results obtained from F&H algorithm are oversegmented; we apply inference techniques, that are widely accepted in the vision community [12], to minimize this. Fig. 3(a) shows a comparison of the number of regions obtained from graph

segmentation versus a manual segmentation over 100 images. In order to reduce the oversegmentation, step (8) builds a spatial adjacency graph for the segmented regions and applies the merging predicate that tests if two regions C_1 and C_2 of which $C_1 \cup C_2$ come from the same statistical region of a perfect scene [13]. The perfect scene is not known however it is assumed that each pixel is represented by a set of distributions from which each observed colour channel i.e. image to segment, is sampled. The assumptions underlying the merging predicate of a perfect scene are:

- The pixels from the same region have the same expectation for each separate colour channel.
- The expectations of adjacent regions differ by at least one colour channel.

The merging predicate is:

$$P(C_1, C_2) = \left\{ \begin{array}{ll} \text{true} & \text{if } |\overline{C_2} - \overline{C_1}| \leq \sqrt{b^2(C_1) + b^2(C_2)} \\ \text{false} & otherwise \end{array} \right\}, \tag{3}$$

$$b(C) = g\sqrt{1/(2Q\,|C\,|))\ln(|C_{|C|}\,|/\delta)},$$

where each colour channel belongs to the set {1,2,…,g}, Q is the number of random variables representing each colour channel making it a parameter to quantify the statistical complexity of the perfect scene. $|\bullet|$ stands for cardinal and δ is the probability error.

5 Experimental Results

We have designed our experiments to compare the effectiveness of the proposed method with the existing methods. First we demonstrate that the preprocessing step improves the performance of our method without affecting the segmentation results. We applied the preprocessing step on three real videos with 120×100, 352×240 and 240×250 frame size respectively. We will first verify the reduction in processing time by applying the preprocessing step of the graph-based segmentation presented in section 3 on the first 50 frames extracted from each video. All experiments were performed on Pentium M processor 1.5GHz laptop. Fig. 2 (a), (b) and (c) shows the average number of graph edges sorted, the average processing time in milliseconds and the average number of regions obtained using different threshold values. Figure 2 (d) shows the average error of the difference obtained in the number of regions for the threshold values. In all experiments, the segmentation parameters were fixed except for the threshold values.

Second, we demonstrate the enhanced performance of our method in terms of speed and number of segments obtained. Fig. 3(b) demonstrates the over segmentation of the graph algorithm by comparing the results of 160 images manually segmented to the results obtained from graph segmentation. Fig. 3(c) demonstrates the enhanced speed of the algorithm: it is almost twice the speed of the original algorithm.

Finally, we compare quantitatively and qualitatively the results of our method with F&H method and SRM. In Table 1, we show the results of different empirical evaluation methods [14] that have been applied to 100 images segmented manually, with F&H algorithm, SRM algorithm, and our method. The table shows the reduced average number of regions obtained by the statistical region merge. The third column shows average intra-region uniformity where small values indicate better region uniformity. The fourth column shows that our method has the highest average inter-region contrast (adequate segmentation should have higher contrast across adjacent regions). F&H algorithm concentrates on keeping regions' internal variances less than their external variances at any moment which makes region uniformity measure and contrast to be good tools for accessing the quality of the segmentation. Fig. 4 shows some of the segmented images.

We experimentally show that the error incurred from the preprocessing step is very small and generally improves the segmentation result when incorporated with statistical region merge. Furthermore, the statistical region merge reduced the effect of over-segmentation and enhanced the quality of segmentation.

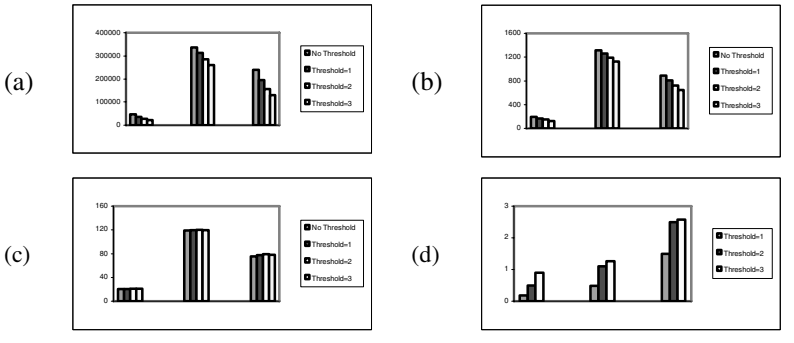

Fig. 2. Comparison of results using different threshold values for the three video clips, 'No Threshold' indicates F&H algorithm. (a) Average number of graph edges sorted (b) Average speed in millisecond (c) Average number of regions (d) Average error in the number of regions.

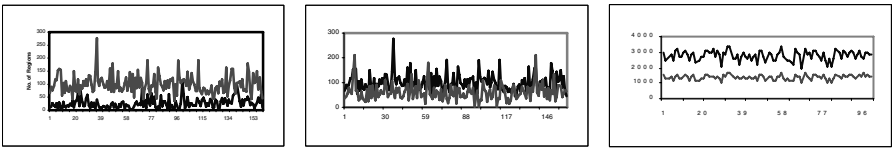

Fig. 3. (a) Regions obtained from F&H algorithm (oversegmentation) compared with manual segmentation (b) Number of Regions obtained from F&H algorithm compared with out method (c) Speed comparison of F&H algorithm and our algorithm in milliseconds over 160 images

Table 1. Results of applying empirical evaluation methods. GU: intra-region uniformity, GC: inter-region contrast, ND: discrepancy based on the position of mis-segmented pixels, D-PE: discrepancy based on the number of mis-segmented pixels.

Average of	Regions	GU	GC	ND	D-PE
F&H	100.59	0.888	1.651	0.587	25.52
SRM	46.55	0.692	1.828	0.662	39.30
Our Method	61.37	0.803	1.885	0.571	28.75

Fig. 4. Segmented Images

References

1. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. on Pattern Analysis and Machine Intelligence. 22 (2000) 888-905
2. E. Sharon, A. Brandt, and R. Basri: Fast multiscale image segmentation. In IEEE Conference on Computer Vision and Pattern Recognition. 1 (2000) 70-77
3. Cour, T., Benezit, F., Shi, J.: Spectral segmentation with multiscale graph decomposition. In IEEE Conf. on Computer Vision and Pattern Recognition. 2(2005) 1124-1131
4. L. Macaire, N. Vandenbroucke, and J.-G. Postaire: Color image segmentation by analysis of subset connectedness and color homogeneity properties. Computer Vision and Image Understanding. 102 (2006) 105-116
5. Y. Deng, B. S. Manjunath, and H. Shin: Color Image Segmentation. In IEEE Conference on Computer Vision and Pattern Recognition. 2 (1999) 446-451
6. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE Trans. on Pattern Analysis and Machine Intelligence 24 (2002) 603 – 619
7. H. Sun, J. Yang, and M. Ren: A fast watershed algorithm based on chain code and its application in image segmentation. Pattern Recognition Letters. 26 (2005) 1266-1274
8. R. Adams and L. Bischof: Seeded region growing. IEEE Transactions on Pattern Analysis and Machine Intelligence. 16 (1994) 641-647
9. F. Y. Shih and S. Cheng: Automatic seeded region growing for color image segmentation. Image and Vision Computing. 23 (2005) 877-886
10. Felzenszwalb, P., Huttenlocher, D.: Efficient graph-based image segmentation. Int. J. of Computer Vision 59 (2004) 167-181
11. Haxhimusa, Y., Kropatsch, W.: Segmentation graph hierarchies. Joint IAPR Int. Workshops SSPR and SPR, Springer-Verlag, Berlin Heidelberg (2004) 343-351
12. Forsyth, D.A., Ponce, J.: Computer vision a modern approach. Prentice Hall (2003)
13. Nock, R., Nielsen, F.: Statistical region merge. IEEE Trans. on Pattern Analysis and Machine Intelligence 26 (2004) 1452-1458
14. Zhang, Y. J.: A survey on evaluation methods for image segmentation. Pattern Recognition 29 (1996) 1335-1346