MVA'94 IAPR Workshop on Machine Vision Applications Dec. 13-15, 1994, Kawasaki

An Efficient Agglomerative Clustering Algorithm for Region Growing

T. Kurita
Electrotechnical Laboratory

1-1-4 Umezono, Tsukuba, Japan 305
E-mail: kurita@etl.go.jp

Abstract

This paper proposes an efficient agglomerative clustering algorithm for region growing. To speed up the search of the best pair of segments which is merged into one segment, dissimilarity values of all possible pairs of segments are stored in a heap. Then the best pair can be found as the element of the root node of the binary tree corresponding to the heap. Since the only adjacent pairs of segments are possible to be merged in image segmentation, this constraints of neighboring relations are represented by sorted linked lists. Then we can reduce the computation for updating the dissimilarity values and neighboring relations which are influenced by the merging of the best pair. The proposed algorithm is applied to the segmentations of a monochrome image and range images.

1. Introduction

Image segmentation is the one of the most fundamental and important techniques in image processing and pattern recognition. There are three approaches, (1) characteristic feature thresholding or clustering, (2) edge based segmentation and (3) region based segmentation[3]. In characteristic feature thresholding or clustering, pixels are classified into several groups based on some characteristic features of the pixels but neighboring constraints of pixels are usually neglected. In edge based segmentation, edges or boundaries are detected by some edge detection operator. Region based approach can be classified into three categories, (1) region merging, (2) region dividing, and (3) a combination of region merging and dividing[3]. In region based approach, only adjacent pixels can be grouped into one region in the clustering process.

In [2], J.M.Beaulieu and M.Goldberg proposed a hierarchical stepwise optimization algorithm for region merging which is based on stepwise optimization and produces a hierarchical decomposition of the picture. The algorithm starts with an initial picture partition, and at each iteration, merges two segments that minimizes a criterion C_{ij} corresponding to the cost of merging the segment S_i with the segment S_j. In this stepwise optimization, the algorithm searches the whole picture context before merging two segments and finds the optimal pair of segments. This means that the most similar segments are merged first. The algorithm gradually merges the segments and produces a sequence of partitions. The sequence of partitions reflect the hierarchical structure of the picture.

From the viewpoint of clustering, this algorithm for region growing can be regarded as agglomerative clustering, which is a typical example of hierarchical clustering. It is known that the agglomerative clustering produces definitely one grouping result for a given data and the final grouping is usually satisfactory, while non-hierarchical clustering algorithm such as k-means method produces different results depending on the choice of initial seed points and may be trapped at a local optimal. However, the computation time of the algorithm is $O(N^3)$ when the algorithm is implemented straightforwardly.

In [8], the author proposed an efficient agglomerative algorithm for general clustering problem. The algorithm uses a heap in which distances of all pairs of clusters are stored. Then the nearest pair of clusters is given by the element of the root node of the binary tree corresponding to the heap. The heap can be easily updated at each stage of the hierarchy by shifting up or down the elements of the heap along the path of the heap tree. The computation time of the algorithm is at most $O(N^2 \log N)$ for the clustering of N objects.

For image segmentation, however, this algorithm can not be directly applied because there are neighboring constraints on merging, namely, only adjacent pairs of segments can be candidates for merging while all pairs of clusters are possible to be merged in general clustering.

In this paper, an efficient agglomerative clustering algorithm for region growing is presented. The algorithm uses sorted linked lists to maintain neighboring relations of segments and also heap structure to store dissimilarities of all possible pairs of segments.

2. The Proposed Algorithm

The proposed agglomerative clustering algorithm starts with an initial partitions of the given image into N segments $\{S_1, S_2, \ldots, S_N\}$, and sequentially reduces the number of segments by merging the best pair of segments among all possible pairs of segments in terms of a given criterion. This merging process is repeated until the required number of segments is obtained.

Since only adjacent pairs of segments can be candidates for merging, we have to maintain such constraints during the merging process. The constraints can be represented as a planer graph in which nodes and edges denote segments and pairs of adjacent segments, respectively. To update the graph efficiently after merging a pair of segments, the graph is represented as sorted linked lists, where the adjacency lists are sorted so that the indecies of nodes appear in increasing order. Then the merging of two lists can be performed in time linearly proportional to the maximum length of the two lists.

To speed up the search of the best pair of segments which is merged into one segment, dissimilarity values of all possible pairs of segments are stored in a heap. A heap is a data structure invented by Williams[10]. The most important characteristic of a heap is that the root node of the associated binary tree has the least element. Thus we can know its least element by simply checking the root node. We utilize this characteristic of a heap in the searching of the best pair of segments.

The general procedure for agglomerative clustering is sammarized as follows;

STEP I Initialization:

1. Start with an initial segments $\{S_1, S_2, \ldots, S_N\}$.
2. Construct a graph which represents adjacency relations of segments using sorted linked lists.
3. Calculate dissimilarities of all adjacent pairs of segments and store the values in a heap.

STEP II Merge the best pair of segments

1. Search the best pair of segments. The pair can be found as the element of the root node of the binary tree corresponding to the heap.
2. Merge the pair of segments into one segment and update the features related with the segment.
3. Update the graph by merging the sorted linked lists.
4. Update the dissimilarities influenced by the merging by shifting up or down the modified elements of the heap along the path of the heap tree.

STEP III Stopping condition

1. Stop if no more mergings are required.
2. Otherwise, go to Step II.

The computation to construct a heap in **STEP I-3** is at most $(mN \log(mN))$, where m is the number of adjacent segments per one segment and N is the number of initial segments. The value mN is the initial number of possible pairs of segments and is gradually decrease as the algorithm proceeds. In the following experiment we set each pixle as intitial segment and 4 nieghboring pixels as the adjacent segments. In this case, mN is equal to 4 times the number of pixels. To update one element in the heap in **STEP II-4**, computations of at most $O(\log(mN))$ is necessary and elements influenced by the merging are at most $2M$, where M is the maximum number of adjacent segments per one segment and usuall is much smaller than the number of initial segments N. Since **STEP II** is repeated at most N times, the total computation time is at most $O(MmN \log(mN))$.

3. Experiments

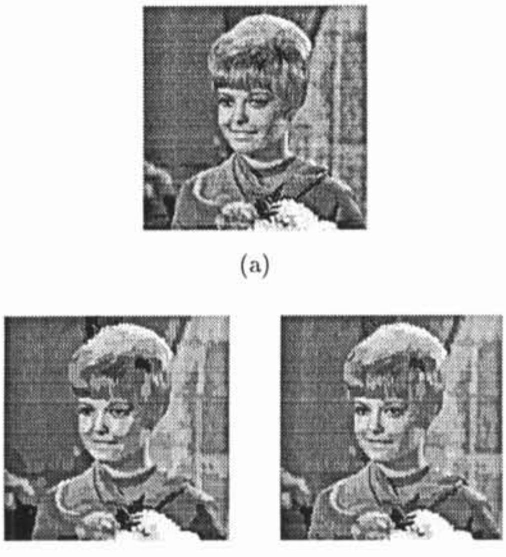
To illustrate the efficiency of the proposed algorithm, we have done experiments on segmentations of a monochrome image and range images[9].

3.1. Monochrome image

At first the proposed algorithm was applied to a monochrome image shown in Figure 1 (a). The size of the image is of 128 by 128 pixels. In the initial partition, each segment consists of only one pixel. The number of initial segments is $N = 16384$. As a global criterion of the clustering, the mean squared errors between the original image and the segmented image

$$\varepsilon^2(S_1, \ldots, S_k) = \frac{1}{N} \sum_{i=1}^{K} \sum_{(x,y)\in S_i} (g(x,y) - \bar{g}_i)^2 \quad (1)$$

is used, where $g(x, y)$ is the intensity of the original image at the pixel (x, y) and $\bar{g}_i$ denotes the average



(a)

(b) (c)

Figure 1. Segmentation of a monochrome Image. (a) Original image (128 by 128 pixels). (b) Segmented image (128 segments). (c) Segmented image (256 segments).

intensity within the segment S_i. In this case, the criterion to select a pair of segments for merging is given by

$$C_{ij} = \delta\varepsilon^2(S_i, S_j) = \frac{n_i n_j}{n_i + n_j}(\bar{g}_i - \bar{g}_j)^2, \qquad (2)$$

where n_i and n_j are respectively the numbers of pixels in the segments S_i and S_j. The clustering algorithm which uses this criterion is known as the Ward's method. In merging step of the proposed algorithm, a new segment is created by merging the pair which has the minimum of this criterion among all possible pairs of segment. Then the neighboring constraints represented as sorted linked lists and the values of this criterion stored in a heap are updated. The segmented images are shown in Figure 1 (b) and (c). The numbers of segments are 128 and 256, respectively. The computation time for this segmentation was about 6 seconds on the SparcStation 10. This shows that the efficiency of the proposed algorithm. The segmentation of 16384 pixels can be performed within reasonable computation time.

Figure 2 shows the relation between the number of segments and the mean squared errors. The mean squared error monotonically increase as the number of segment decrease.

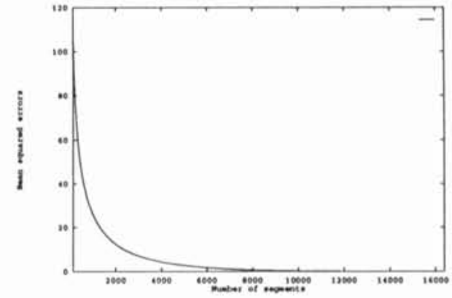
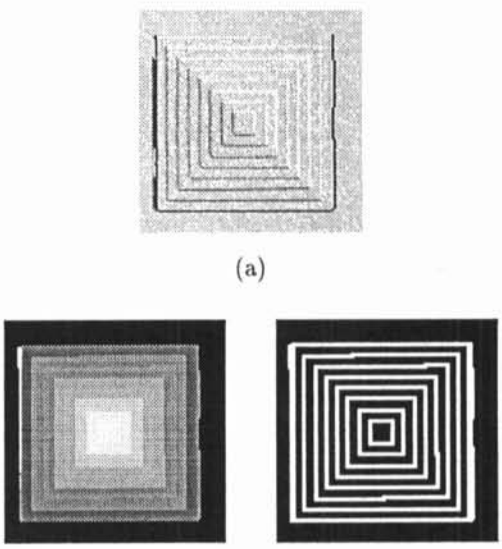


Figure 2. Relation between the number of segments and the mean squared errors.

3.2. Range images

3.2.1. Segmentaion based on hights



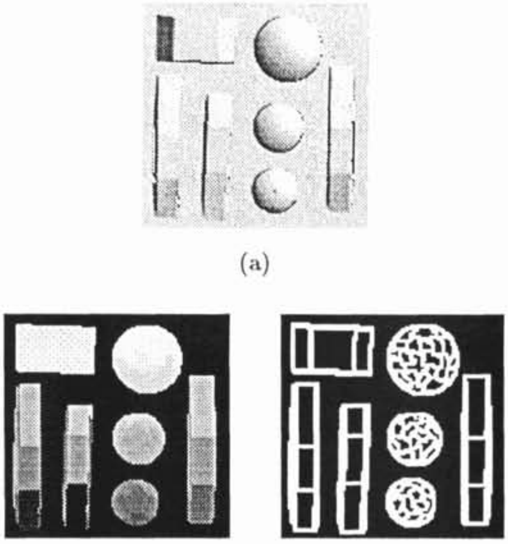
(a)

(b) (c)

Figure 3. Segmentation of a range image based on local hights. (a) Shaded rendering of range image (128 by 128 pixels). (b) Segmentation Result. (c) Region boundaries of the segmented image (b).

Next, segmentations of range images were performed. Figure 3 (a) shows shaded rendering of a range image. The size of the range image is also 128 by 128 pixels. There is a pyramidal object in the image. The segmentation was performed based on the height. The mean squared error criterion was also used as the global criterion. The segmentation result is shown in Figure 3 (b). Figure 3 (c) shows the boundaries of the segments by applying a simple edge operator to the segmented image. For this range image, the proposed algorithm worked very well. The computation time of this segmentation was also about 6 seconds on SparcStation 10.

3.2.2. Segmentation based on normals



(a)

(b) (c)

Figure 4. Segmentation of a range image based on local surface normals. (a) Shaded rendering of range image (128 by 128 pixels). (b) Segmentation Result. (c) Region boundaries of the segmented image (b).

Finally, the proposed algorithm was applied to the range image shown in Figure 4 (a). This image contains 7 objects; 4 planer objects and three objects with curved surface. For this range image, local surface normal vector is used as features of each pixel. The global criterion

$$\varepsilon^2(S_1,\ldots,S_k) = \frac{1}{N}\sum_{i=1}^{K}\sum_{(x,y)\in S_i}(\boldsymbol{n}(x,y)-\bar{\boldsymbol{n}}_i)^2 \quad (3)$$

is used, where $\boldsymbol{n}(x,y)$ is the local normal vector at the pixel (x,y) and $\bar{\boldsymbol{n}}_i$ denotes the average normal vector within the segment S_i. The normal vectors of each pixel were computed by fitting a plane within a local 3 by 3 window. The segmentation result is shown in Figure 4 (b) and the boundaries of the segments in Figure 3 (c). Each planar region is correctly classified into one segment but curved regions are partitions into several patches. This is reasonable because the local normal vectors are used as primitive features. The computation time for this segmentation was about 10 seconds on SparcStation 10.

4. Conclution

We proposed an efficient agglomerative clsutering algorithm for region growing which uses a heap to store dissimilarity values of all possible pairs of segments and sorted linked lists to maintain constraints on adjacency relations of segments. Expelimental results show that this algorithm makes the stepwise optimization approach for region growing feasible on the current computer.

Acknowledements

The authors would like to thank M. Rioux, L. Cournoyer, and J. Domey of the National Research Council of Canada who kindly provided access to the range image database.

References

[1] A.V.Aho, J.E.Hopcroft, and J.D.Ullman, "*The Design and Analysis of Computer Algorithms*," Addision-Wesley (1976).

[2] J.-M.Beaulieu and M.Goldberg, "Hierarchy in picture segmentation: A stepwise optimization approach," IEEE Trans. on Pattern Analysis and machine Intelligence, Vol.11, No.2, pp.150-163, 1989.

[3] K.S.Fu and J.K.Mui, "A survey on image segmentation," Pattern Recognition, Vol.13, pp. 3-16, 1981.

[4] R.M.Haralick, "Edge and region analysis for digital image data," Compute Graphics and Image Processing, 12, pp.60-73, 1980.

[5] R.M.Haralick and L.G.Shapiro, "Image segmentation techniques," Computer Vision, Graphics, and Image Processing, 29, pp.100-132, 1985.

[6] R.Hoffman and A.K.Jain, "Segmentation and classification of range images," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.PAMI-9, No.5, pp.608-620, 1987.

[7] M.Kocher and R.Leonardi, "Adaptive region growing technique using polynomial functions for image approximation," Signal Processing, Vol.11, pp.76-60, 1986.

[8] T.Kurita, "An efficient agglomerative clustering algorithm using a heap," Pattern Recognition, Vol.24, No.3, pp.205-209, 1991.

[9] M. Rioux, F. Blais, J.A. Beraldin and P. Boulanger, "Range image sensors development at NRC laboratories", Proc. of the Workshop on Interpretation of 3D scenes, Austin, TX, November 27–29 (1989).

[10] J.W.J.Williams, "Heapsort," Comm. of ACM, Vol.7, No.6, pp.347-348 (1964)

[11] S.W.Zucker, "Region grouping: childhood and adolescence," Compute Graphics and Image Processing 5, pp.382-399, 1976.