

Dynamic Image Segmentation Method Using Hierarchical Clustering

Jorge Galbiati¹, Héctor Allende^{2,3,*}, and Carlos Becerra⁴

¹ Pontificia Universidad Católica de Valparaíso, Chile, Department of Statistics
`jorge.galbiati@ucv.cl`

² Universidad Técnica Federico Santa María, Chile, Department of Informatics
`hallende@inf.utfsm.cl`

³ Universidad Adolfo Ibáñez, Chile, Science and Engineering Faculty

⁴ Universidad de Valparaíso, Chile, Department of Computer Science
`carlos.becerra@uv.cl`

Abstract. In this paper we explore the use of the cluster analysis in segmentation problems, that is, identifying image points with an indication of the region or class they belong to. The proposed algorithm uses the well known agglomerative hierarchical cluster analysis algorithm in order to form clusters of pixels, but modified so as to cope with the high dimensionality of the problem. The results of different stages of the algorithm are saved, thus retaining a collection of segmented images ordered by degree of segmentation. This allows the user to view the whole collection and choose the one that suits him best for his particular application.

Keywords: Segmentation analysis, growing region, clustering methods.

1 Introduction

Image segmentation is one of the primary steps in image analysis for object identification. The main aim is to recognize homogeneous regions within an image as distinct objects. It consists of partitioning an image in regions that are supposed to represent different objects within it. Some segmentation methods are automatic, or non-supervised, they do not need human interaction; others are semiautomatic or supervised.

One of the simplest technique used for segmentation is thresholding, in which pixels whose intensity exceeds a threshold value defined by the analyst are said to belong to one region, while those that do not, belong to the other.

A semiautomatic segmentation methods is region growing, that starts off with a group of pixels defined by the analyst as seeds, then other neighborhood pixels are added if they have similar characteristics, according to some criterion. This algorithm is improved with the introduction of a bayesian approach, Pan and Lu [8], by means of a homogeneity criteria of neighbouring pixels.

While image segmentation consists of partitioning an image in homogeneous regions, edge detection is the identification of the lines that define the borders

* This work was supported by Fondecyt 107220 research grant.

between regions determined by a segmentation process. Image segmentation and edge detection are both dual problems, in the sense that solving one provides the solution to the other. Some segmentation methods are based on edge detection. That is the case of morphological watershed methods. They consist on calculating the gradients of an image, and make them to resemble mountains, while simulating a water flood, the water level raising progressively. As the water level increases, it determines thinner zones on the top of the mountains, which are defined as edges, and the zones contained within these edges are segments. Frucci et al. [4] developed a segmentation method based on watersheds using resolution pyramids, formed by a set of images of decreasing resolution, under the premise that the most relevant aspects of an image are perceived even at low resolution.

The use of wavelets has spread as a segmentation method. Wavelets can model the behavior pattern of the pixels in the image. The difference with the Fourier transformation is that these are not periodic. Perhaps the most widely used wavelet transformation in images is the Gabor wavelet, the product of a sinusoid and a gaussian. Wang et al. [10] introduced a texture segmentation method on the basis of Gabor wavelets. It is used for the extraction of texture characteristics.

Image segmentation by hierarchic clusters uses the classic hierarchic cluster analysis methods that groups the nearest clusters at each stage. Martinez-Use et al. [7] segment an image by means of a hierarchic process that maximizes a measurement that represents a perceptual decision. It can be also used for multispectral images. Another clustering method is k-means, that iteratively moves elements to the cluster whose centroid is the closest, the process ends when no elements change places. Fukui et al. [5] apply hierarchic clustering for the detection of objects, in particular, face detection. They use k-means clustering with color space features. Allende and Galbiati [2] developed a method for edge detection in contaminated images, based on agglomerative hierarchical clusters, by performing a cluster analysis on a 3×3 pixel moving window; more than one cluster denotes the presence of an edge. A segmentation method based on cluster analysis is shown in Allende et al. [3]. It runs a 3×3 neighbourhood window along the image, performing a cluster analysis each time and deciding whether the central pixel belongs to one of the clusters as the surrounding pixels. If it does, it assigns it to the corresponding cluster. If not, it creates a new cluster. At the end, each cluster is considered a segment.

Image segmentation is usually combined with other image processing methods, like image enhancement and restoration, which are usually performed before segmenting; and like feature extraction, object recognition and classification, and texture analysis, for which segmentation is a previous step.

2 Method

The agglomerative clustering algorithm starts defining each element as one cluster, so at the starting point we have as many clusters as the number of elements. Then the distances between elements are computed, forming a symmetric $N \times N$

distance matrix, where N is the number of elements. Distances like a Minkowski type distance are frequently used,

$$d(\underline{x}, \underline{y}) = \left(\sum_{i=1}^p |x_i - y_i|^k \right)^{1/k}$$

where \underline{x} and \underline{y} are p -dimensional vector elements, and k is any positive real number. $k = 1$ gives the "city block" distance, $k = 2$ corresponds to the euclidean distance.

Then we need to define a distance measure between clusters, as a function of the distance between elements. A convenient distance measure is the average of all the distances between pairs of elements, one from each cluster. Other distances are, the smallest distance between pairs of elements, or the largest distance, or the distance between the averages of the elements of both clusters.

The second step consists of finding the smallest distance, and joining the corresponding two clusters to form a new cluster with two elements. The distances from this new cluster to the remaining clusters are then computed, and the distances from the original two clusters that merged are eliminated from the set of distances, so we have a new $(N-1) \times (N-1)$ distance matrix. The previous procedure is repeated until we have one single cluster containing all the elements. At each stage, the distance at which two clusters are merged increases, thus building clusters in increasing degree of heterogeneity. A record is kept of all the stages, the way they join to form larger clusters and their merging distances, which form an increasing sequence, and can be represented by a graph called dendrogram, which illustrates the way the hierarchical procedure developed.

As can be noticed, the complexity of the problem grows fast as the number of elements to be clustered increases. In fact, in the case of N elements, the number of distances, and consequently the number of steps, is $\frac{1}{2}N \cdot (N-1)$.

This procedure can be applied to images, to develop a segmentation method, where each pixel is a vector element. In the case of color RGB images, the vectors are three dimensional, the coordinates representing the Red, Green and Blue intensities. In monochrome images, the elements are scalars, and the Minkowski distance turns up to be equal to the absolute value difference.

The algorithm becomes particularly complex in the case of images. An image consisting of n columns and m rows has $npix = n \cdot m$ pixels, so the number of distances is $\frac{1}{2}n \cdot m \cdot (n \cdot m - 1) = \frac{1}{2}n^2 \cdot m^2 - n \cdot m$. Suppose a 1000×1000 image, which is not too large, the number of distances is approximately $5 \cdot 10^{11}$, which would make the task of forming clusters almost impossible. But, unlike general situations where cluster analysis is applied, in images the pixels to be clustered are elements which have an order. And the cluster forming procedure should consider this fact, so as to merge only adjacent clusters. As a consequence, at the initial step the distances to be considered are the distances between each pixel and two of its neighbors, the one at its left and the one beneath. That makes the number of initial distances to be scanned at the first step equal to $ndist = 2m \cdot m - n - m$. In the example where $n = m = 1000$, this number is approximately equal to 2×10^6 , considerably smaller than in the general case.

1	2	3	4
5	6	7	8
9	10	11	12

Fig. 1. Segmentation process

Figure 1 illustrates how the algorithm works for an image with three clusters. The pixels that are to be compared are the following: First row, 1-2, 1-5, 2-3, 2-6, 3-4, 3-7, and 4-8; second row, 5-6, 5-9, 6-7, 6-10, 7-8, 7-11, and 8-12; third row, 9-10, 10-11, and 11-12. There are two light gray clusters, but although they are the same colour, their pixels are never going to be compared, as both clusters are not adjacent. Thus, although the same, the algorithm considers them as different clusters.

The algorithm uses three arrays of size $npix$: $Frec(s)$, $Avg(s)$ and $Pos(p)$. $Frec(s)$ keeps the frequency, or number of pixels, of the cluster or segment s ; $Avg(s)$ has the average value of the intensity of each color of all pixels within segment s ; $Pos(p)$ contains a label assigned to pixel p , indicating the segment to which it belongs and a label giving it a number within the segment. The algorithm also uses three arrays of size $ndist$: $dist(d)$, $e(d)$ and $f(d)$, where $dist(d)$ keeps the d^{th} distance value between segments $e(d)$ and $f(d)$.

At each step, after the smallest distance is obtained, say $dist(d_0)$, the two segments $e(d_0)$ and $f(d_0)$ are merged into one, which retains the label $e(d_0)$. The frequency of the new segment $e(d_0)$ is the sum of the two frequencies, while the frequency of the $f(d_0)$ is set to zero. The average value of this new segment is obtained as a weighted average of the averages of the two concurrent segments, and the average of $f(d_0)$ is now zero. The labels in $Pos(p)$ are updated for each pixel p that belonged to segment $f(d_0)$.

Finally, the distances from the new segment to all its adjacent segments are updated. The adjacent segments are easily identified, considering all pairs $(e(d), f(d))$, where one of the two numbers is the label of any one of the two segments that merged; the other corresponds to an adjacent segment. To start, every pixel is defined as a segment, numbered from 1 to $npix$.

The iterations are carried on until the merging distance reaches a threshold value $thrs$ given initially as an input, an integer in the interval $(0,255)$. With this threshold value the degree of the segmentation can be controlled. Small values will give finer segmentations, while higher values will give coarser segmentation, with a small number of large segments.

The sequence of segment union distances follows the same pattern, independent of the threshold value. The difference is that if we choose two segmentations with different threshold values, say $T1$ and $T2$, with $T1 < T2$, then the sequence of steps followed to reach $T1$ is the same as the initial part of the sequence of steps followed to reach $T2$.

We can take advantage of this fact in order to obtain a series of segmentations, with different degrees of detail, instead of just one. In this way, after the image is processed, we can sequentially visualize all the segmented images, each with a

different level of segmentation, from the finest, equivalent to the original image, to the coarsest level, which corresponds to a one color rectangle, the average color. To do this, the dimensionality of the arrays has to be increased, but it is not necessary to save each entire image for every segmentation level, we only have to save the changes made from one step to another, and this reduces the amount of memory required.

The method presented here differs from seeded region growing because here, the algorithm starts merging pixels which are closest together, according to the distance being used, while in the other method the merging starts with pixels which are adjacent to seed pixels that were previously determined by the user.

3 Results

In the experimental results presented in this paper, the distance used was the city block distance. The distance between clusters is the sum of the absolute differences between the average of each colour, in both clusters. In each case we saved 24 segmented images, with threshold values for the maximum merging distance ranging from the minimum to the maximum observed distances in each case, at intervals of one 23rd of the range between them. In Figures 2 to 6 we show a few of the results, indicating the threshold value in each case. Some of the images were taken from the Berkeley Data Set for Segmentation and Benchmark [6].

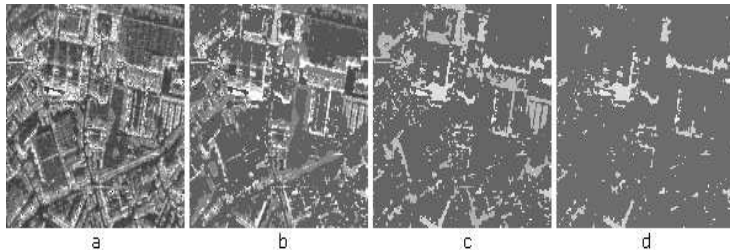


Fig. 2. City air photograph. (a) Original image. Segmentation threshold values: (b) 40, (c) 80, (d) 96.

4 Discussion

The method involves a large amount of computing, that makes it relatively slow, compared to some other methods. Figure 5 is a 131×135 image, with a total of 17685 pixels, that makes 35104 initial distances. It took 6 minutes and 28 seconds to process with a Visual Basic program, running on an Intel Duo Core processor at 3.0 GHz, with 3.25 GB Ram. The output was a series of 15 segmented images, four of which are shown here. An optimized computer program would contribute to make it work faster. It is also memory consuming, but a careful management of the stored information can help to reduce the memory consumption.

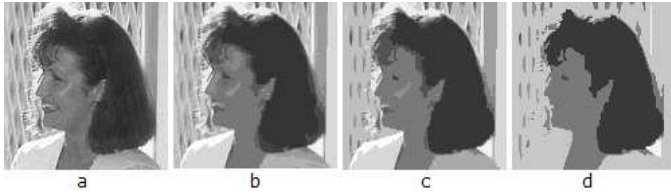


Fig. 3. Woman's face 1. (a) Original image. Segmentation threshold values: (b) 21, (c) 28, (d) 50.

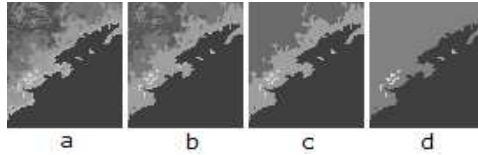


Fig. 4. Air photograph of coast. (a) Original image. Segmentation threshold values: (b) 27, (c) 37, (d) 59.



Fig. 5. Female lion. (a) Original image. Segmentation threshold values: (b) 19, (c) 32, (d) 52, (e) 58.

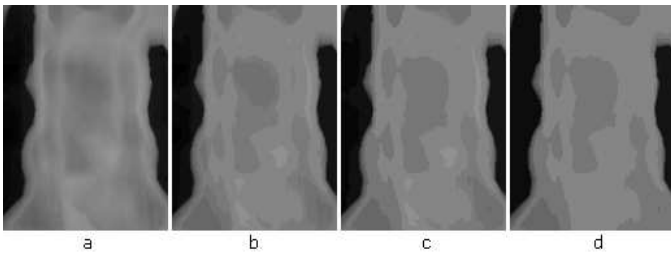


Fig. 6. Vertebrae X-ray. (a) Original image. Segmentation threshold values: (b) 9, (c) 12, (d) 15.

The particular way that the segments are merged together, that is, only considering adjacent segments, results in the fact that sometimes two disjoint segments look very similar in color, but remain as separate segments, while others, that are not as similar, are merged earlier. This is correct, because if two

segments remain like that, it is because they are not adjacent. In the image, that means that they are separate, and even if they look as if they are similar in color, they do represent different objects. That can be clearly appreciated in the lion's leg and body, in Figure 5.

If the image is too noisy, the method does not give a good result. But applying a previous smoothing, like, for example, a median smoothing, or the method shown in [1] the quality of the result is significantly increased.

5 Conclusions and Future Studies

In case we want one particular segmentation we have to set a threshold value at the start of the segmentation algorithm. If we want the algorithm to give us a set of segmented images with different degrees of segmentation, from which to choose the one we want for our specific application, then we do not have to set a threshold value, but we do need to choose the segmentation we want. So in both cases there is a human intervention, at the start or at the end. That means that this is not a fully automatic segmentation method.

The fact that the user has the possibility of looking at various segmentation levels as a result of processing the image only once, makes the method versatile and gives it a dynamic characteristic, giving him the possibility of choosing the segmentation level that suits his particular needs. An important characteristic of this method is that it can work fairly well with images that show little contrast. This can be seen in Figure 6.

As future studies, the authors intend to investigate the efficiency of this method, in terms of computational resources.

Another aspect to study is obtaining optimal values for the optimal threshold parameter.

Also the authors are going to perform comparisons of the results of this method with other comparable methods.

References

1. Allende, H., Galbiati, J.: A non-parametric filter for digital image restoration, using cluster analysis. *Pattern Recognition Letters* 25, 841–847 (2004)
2. Allende, H., Galbiati, J.: Edge detection in contaminated images, using cluster analysis. In: Sanfeliu, A., Cortés, M.L. (eds.) *CIARP 2005. LNCS*, vol. 3773, pp. 945–953. Springer, Heidelberg (2005)
3. Allende, H., Becerra, C., Galbiati, J.: A segmentation method for digital images based on cluster analysis. *Lecture Notes In Computer Science: Advanced And Natural Computing Algorithms*, vol. 4431(1), pp. 554–563 (2007)
4. Frucci, M., Ramella, G., di Baja, G.S.: Using resolution pyramids for watershed image segmentation. *Image and Vision Computing* 25(6), 1021 (2007)
5. Fukui, M., Kato, N., Ikeda, H., Kashimura, H.: Size-independent image segmentation by hierarchical clustering and its application for face detection. In: Pal, N.R., Kasabov, N., Mudi, R.K., Pal, S., Parui, S.K. (eds.) *ICONIP 2004. LNCS*, vol. 3316, pp. 686–693. Springer, Heidelberg (2004)

6. Martin, D., Fowlkes, C., Tal, D., Malikand, J.: A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring. In: Ecological Statistics, Proc. 8th Int'l Conf. on Computer Vision, vol. 2, pp. 416–423 (2001)
7. Martinez-Uso, A., Pla, F., Garcia-Sevilla, P.: Unsupervised image segmentation using a hierarchical clustering selection process. In: Yeung, D.-Y., Kwok, J.T., Fred, A., Roli, F., de Ridder, D. (eds.) SSPR 2006 and SPR 2006. LNCS, vol. 4109, pp. 799–807. Springer, Heidelberg (2006)
8. Pan, Z., Lu, J.: A Bayes-based region-growing algorithm for medical image segmentation. *Computing In Science and Engineering* 9(4), 32–38 (2007)
9. Pauwels, J., Frederix, G.: Finding salient regions in images. *Computer Vision and Image Understying* 75, 73–85 (1999)
10. Wang, K.B., Yu, B.Z., Zhao, J., Li, H.N., Xie, H.M.: Texture image segmentation based on Gabor wavelet using active contours without edges. Department of Electronic Engineering, Northwestern Polytechnical University, Xi'an 710072, China