A Linear Time Implementation of k-Means for Multilevel Thresholding of Grayscale Images

Pablo Fonseca and Jacques Wainer

RECOD Lab, Institute of Computing, University of Campinas, Campinas, Brazil*

Abstract. In this paper we present a method based on the k-means algorithm for multilevel thresholding of grayscale images. The clustering is computed over the histogram rather than on the full list of intensity levels. Our implementation runs in linear time per iteration proportional to the number of bins of the histogram, not depending on the size of the image nor on the number of clusters/levels as in a traditional implementation. Therefore, it is possible to get a large speedup when the number of bins of the histogram is significantly shorter than the number of pixels. In order to achieve that running time, two restrictions were exploited in our implementation: (I) we target only grayscale images and (II) thresholding does not use spatial information.

1 Introduction

Thresholding is an important task in image processing. It is one of the most basic approaches to image segmentation, as it aims to separate regions of an image according to their intensity levels. These thresholds will determine which pixels belong to each group, and the determination of their actual values can be done automatically. K-means [1] is one of the several methods suited for that task, as it separate data points into k clusters, implicitly determining $k-1$ thresholds.

We take advantage of the fact that thresholding does not take into account the spatial relationships among pixels, and thus it is possible to fully determine the values on the histogram of the image, which is a more succinct representation. Other techniques that run on the histogram include the well known Otsu's method [2] as well as others [3] [4] [5] [6] [7]. In this paper we present an implementation of the k-means algorithm for multilevel image thresholding that computes the clustering on the histogram.

The running time for each iteration of the k-means in our implementation is $\mathcal{O}(h)$, where h is the number of bins of the histogram. It does not depend on the number of clusters nor on the size of the image. It is possible to obtain a large speedup for the cases where the histogram is significantly shorter than the full list of pixels when compared to a traditional implementation of k-means.

The paper continues as follows. In section 2, related work is briefly reviewed. In section 3 we describe our implementation of the k-means algorithm for running

* Thanks to the National Council for Scientific and Technological development - CNPq (Brazil) for providing funding.

E. Bayro-Corrochano and E. Hancock (Eds.): CIARP 2014, LNCS 8827, pp. 120–126, 2014.

on the histogram and its relationship with the Otsu's method. Finally, we report the experiments and results, conclusions and discuss future work.

2 Related Work

Liu et al. [8] proved the equivalence of the Otsu's method [2] with k-means for multilevel image thresholding based on a previous proof for the bilevel case. They pointed out that when the number of dimensions grows, it is better to use the list of points while using k-means than doing it in the histogram with an algorithm such as Otsu's method.

We go the other direction, adapting k-means to run in the histogram. We did this to derive a faster method for multilevel image thresholding, but targeting grayscale images and aiming a method that runs in linear time, depending just on the number of bins of the histogram.

Other related work include application of the k-means method for image segmentation such [9] and [10] in the medical image segmentation domain, as well as satellite image processing [11] where Otsu's method was compared to grayscale k-means with the conclusion that the running time was critical to the presented application, making k-means a better choice over an exhaustive Otsu's method.

2.1 The k-Means Clustering Algorithm

The k-means algorithm is an unsupervised learning method for clustering similar objects into k groups. In order to do so, it uses the euclidean distances between the representations of the objects as an indicator of similarity. It can be seen as a method that solves an optimization problem where the intra-cluster distance should be minimized. Each cluster is represented by its center, which is obtained by averaging every object assigned to the cluster.

It has two main steps in every iteration (1) cluster assignment and (2) new center calculation. In the first step, every object is assigned to the nearest cluster center by computing the euclidean distance of every point to each cluster center. When every object is assigned to its closest cluster, the new cluster center is recomputed. This iterative process is performed until convergence or other stopping criteria.

A more detailed description of the algorithm can be found in [12], where the application to image segmentation and quantization is also discussed.

3 Methods

In this section we address our implementation of k-means for image thresholding on the histogram. First, we describe the histogram computation and then we discuss the details of our implementation and present its pseudocode.

3.1 Histogram Computation

In order to compute the histogram of a grayscale image it is necessary to visit every pixel in the image and increase the frequency count in the histogram vector. With that in mind, it is easy to see that the computational cost of the histogram building is $\mathcal{O}(mn)$ where m is the height and n is the width of the image. It also worth noticing that it has a memory usage of $\mathcal{O}(2^{bits})$, that is, proportional to two to the power of the number of bits used to represent the intensity levels. A normalized histogram can be obtained by dividing each bin count by the total amount of pixels mn.

3.2 K-Means over the Histogram Representation

Implementing the k-means on the histogram is possible because the information about spatial relationships among pixels is not used for thresholding and we restricted our problem to grayscale images. We exploit these characteristics in order to derive an implementation of the k-means algorithm running in $\mathcal{O}(h)$ per iteration. For grayscale images, the distance between two pixels in terms of color is a simple rest of their intensity levels. In the context of the histogram representation, this distance can be obtained from the index of the histogram bins.

The k-means algorithm, as discussed above, has two main operations in every iteration. The first, the cluster assignment demands distances of every element to the k cluster centers to be computed. A simple implementation of this will lead to a computational cost of $\mathcal{O}(mnk)$ meaning that it would be proportional to

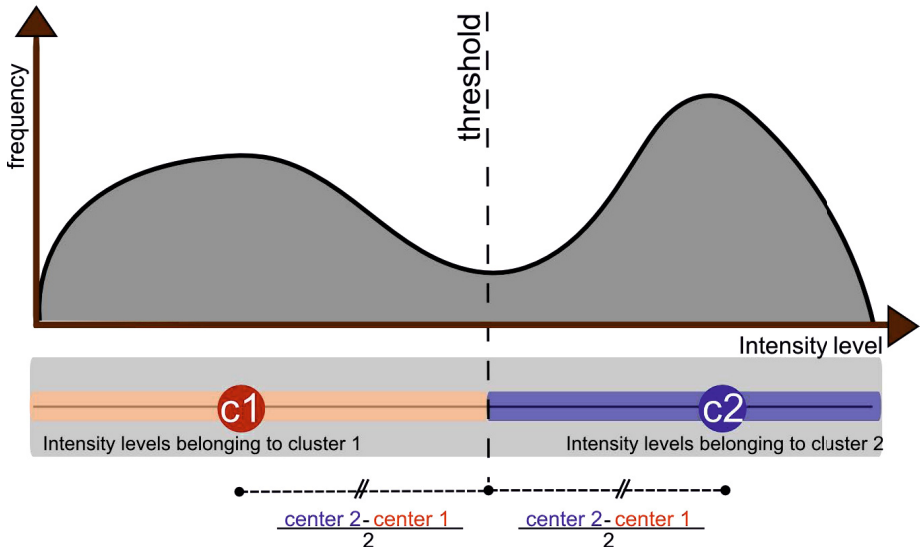

Fig. 1. K-means in the histogram

the amount of pixels in the image multiplied by the number of clusters. However, in the histogram representation, an intensity level is going to be assigned to the cluster center that is nearer to that histogram bin.

Moreover, in order to assign an intensity level to the nearest cluster center it is not required to compute distances. If cluster centers are initialized sorted, it is easy so see that the area of influence of a cluster goes up to the middle point between consecutive cluster centers, as shown in Fig. 1. Beyond that point, the intensity level will be closer to the other cluster center.

This allows to avoid explicit distance calculation between points and cluster centers in the k-means algorithm. Intensity levels in the range $(center_{i-1} + center_i)/2 < x < (center_i + center_{i+1})/2$ are going to be assigned to the cluster i and used for the new mean computation directly. It worth noticing that the region from 0 to the first center will belong entirely to the first cluster, as well as the region from the last cluster center to the maximum value will belong to the last cluster. This, of course, demands the cluster centers to be sorted when initialized. Nevertheless, this does not require us to sort the clusters after each iteration, as they will keep sorted. This has a computational cost of $\mathcal{O}(h)$, because for each cluster only its influence region is going to be visited, leading to a unique visit to each histogram bin per iteration. The Algorithm 1 shows how this method is implemented. For an actual implementation, we have made available GNU Octave/Matlab code online.

Algorithm 1. Histogram based k-means image thresholding

Require: $histogram \leftarrow (bin_1, bin_2, bin_3 ... bin_h)_h$
  $centers \leftarrow (c_1, c_2, c_3 ... c_k)_k$ //Sorted
  repeat
    $newCenters \leftarrow (0, 0, .., 0)_k$
    for $(i = 0; i < k; i++)$ **do**
      $start = determineStartInfluence(i, k, centers)$
      $end = determineEndInfluence(i, k, centers)$
      $freqCount \leftarrow 0$
      for $(j = start; j <= end; j++)$ **do**
        $newCenters[i] \leftarrow newCenters[i] + j * histogram[j]$
        $freqCount \leftarrow freqCount + histogram[j]$
      end for
      $newCenters[i] \leftarrow newCenters[i]/freqCount$
    end for
    $centers \leftarrow newCenters$
  until Stopping Criteria

3.3 K-Means and Otsu's Method

K-means and the Otsu's method are equivalent according to [8] for the bilevel case as well as the multilevel case. However, the Otsu's method uses a exhaustive search procedure and thus can be very expensive for multilevel thresholding. On

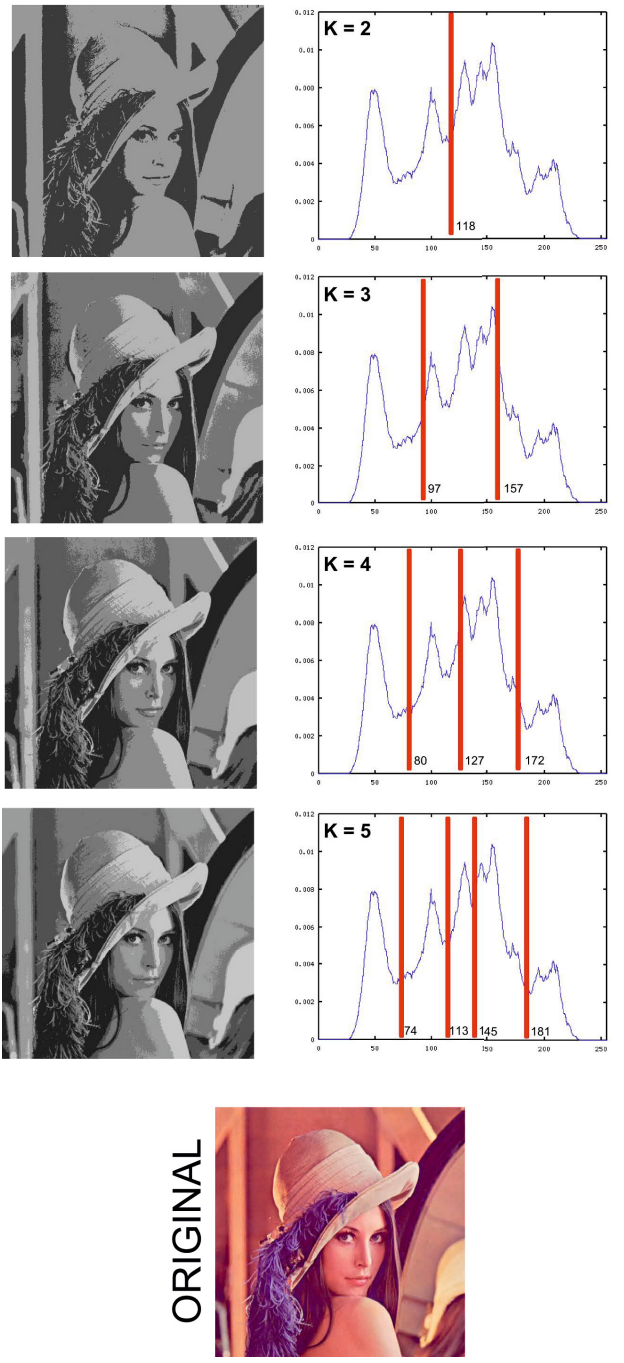

Fig. 2. Multilevel thresholding with k-means for k=2,3,4 and 5

the other side, even if k-means has convergence guarantee, it can converge to local minima, depending on the initialization of the cluster centers. Nevertheless, the time taken by k-means is significantly shorter and the trade off should be analyzed on an application basis.

4 Results

We implemented the algorithm in the Matlab/Octave programming language, and made it available online[1]. We show in figure 2 a sample result with the standard Lena testing image for $k = 5, 4, 3$ and 2. The thresholds found are reported in table 1. The reason why there are floating point values in this table, is because k-means implies thresholds which are in the middle point between two adjacent cluster centers.

Table 1. Thresholds found with k-means

K=5	K=4	K=3	K=2
74.91	80.99	97.25	118.29
113.61	127.37	157.01	-
145.60	172.03	-	-
181.06	-	-	-

In order to show how it relates to the multilevel Otsu's, we show in table 2 the thresholds found with the Matlab's implementation (which is not the exhaustive version).

Table 2. Thresholds found with Matlab's implementation of Otsu's method

N=4	N=3	N=2	N=1
74	80	90	116
113	127	149	-
145	171	-	-
179	-	-	-

It worth noticing that k clusters would generate $k - 1$ thresholds.

5 Conclusions

We presented an implementation of the k-means algorithm for multilevel grayscale image thresholding using the histogram instead of running over the list of intensity levels. It is a fast although equivalent k-means implementation, however subject to the deficiencies of k-means.

The main gain in computing cost because of the usage of the histogram representation and also because we restricted to grayscale images. Succinctness of the histogram, allowed us to assign intensity levels (groups of pixels) at the same time, and without explicit distance calculation.

[1] `https://bitbucket.org/palefo/kmeans`

5.1 Future Work

Our future work will focus on researching how to extend this to color images, eventually by projecting color images into grayscale with dimensionality reduction techniques. We also plan to explore convergence issues of the k-means for different type of histograms, so some guarantees can be derived.

Acknowledgments. The authors would like to thank the National Council for Scientific and Technological development - CNPq (Brazil) for providing funding for the project. We also want to thank Dr. Benjamin Castaneda and Rodrigo Velarde from the Pontifical Catholic University of Peru for valuable discussions on the project.

References

1. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, California, USA, vol. 1, p. 14 (1967)
2. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man and Cybernetics 9(1), 62–66 (1979)
3. Tsai, D.M., Chen, Y.H.: A fast histogram-clustering approach for multi-level thresholding. Pattern Recognition Letters 13(4), 245–252 (1992)
4. Tsai, W.H.: Moment-preserving thresolding: A new approach. Computer Vision, Graphics, and Image Processing 29(3), 377–393 (1985)
5. Yen, J.C., Chang, F.J., Chang, S.: A new criterion for automatic multilevel thresholding. IEEE Transactions on Image Processing 4(3), 370–378 (1995)
6. Wang, S., Haralick, R.M.: Automatic multithreshold selection. Computer Vision, Graphics, and Image Processing 25(1), 46–67 (1984)
7. Sahoo, P., Soltani, S., Wong, A.: A survey of thresholding techniques. Computer Vision, Graphics, and Image Processing 41(2), 233–260 (1988)
8. Liu, D., Yu, J.: Otsu method and k-means. In: Ninth International Conference on Hybrid Intelligent Systems, HIS 2009, vol. 1, pp. 344–349 (August 2009)
9. Ng, H.P., Ong, S.H., Foong, K.W.C., Goh, P.S., Nowinski, W.: Medical image segmentation using k-means clustering and improved watershed algorithm. In: 2006 IEEE Southwest Symposium on Image Analysis and Interpretation, pp. 61–65 (2006)
10. Pham, D.L., Xu, C., Prince, J.L.: Current methods in medical image segmentation 1. Annual Review of Biomedical Engineering 2(1), 315–337 (2000)
11. Soares, F., Catalao, J., Nico, G.: Using k-means and morphological segmentation for intertidal flats recognition. In: 2012 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), pp. 764–767 (July 2012)
12. Bishop, C.M., et al.: Pattern recognition and machine learning, vol. 1. Springer, New York (2006)