Martín-Rodríguez F. (2013) New Tools for Gray Level Histogram Analysis, Applications in Segmentation. In: Kamel M., Campilho A. (eds) Image Analysis and Recognition. ICIAR 2013. Lecture Notes in Computer Science, vol 7950. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-39094-4_37

New Tools for Gray Level Histogram Analysis, Applications in Segmentation

Fernando Martín-Rodríguez

Communications and Signal Theory Department, University of Vigo,
E.E.T., C/Maxwell S/N (Ciudad Universitaria), 36310 Vigo, Spain
fmartin@tsc.uvigo.es

Abstract. This paper summarizes three algorithms used for the analysis of gray level histograms. Two of them are developed generalizing standard techniques and the other is a completely new method. We will study the advantages of each and give examples of real-world use. Gray level histogram analysis (mainly threshold computation) is a known technique that allows easy and fast segmentation of the regions of interest in an image [1]. Many methods have been proposed for this problem [2], but almost all of them focus only on the problem of bimodal histograms. We will deal with multimodal histograms and, besides, we improve the time efficiency of the most widely used method (Otsu's [3]).

Keywords: image analysis, gray level histograms, threshold computation, multimodal histogram, image segmentation.

1 Introduction

The origin of this work was the study of electronic components images (Fig. 1). We wanted to segment these images into their different elements: background, body of chip, characters on the chip and pins. If we apply the classical method from Otsu [3] we will only distinguish the brighter objects (characters and pins in Fig. 2). This is reasonable because Otsu method is designed for bimodal histograms (one threshold, two regions). This image has a more complex histogram (Fig. 3).

In section 2, we will apply three different methods to this problem. The first two methods are based on simple modifications on Otsu's original one and the third one is completely new. In section 3, we will apply our new method to other problems.

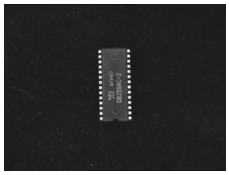

Fig. 1. Chip image

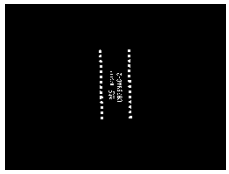

Fig. 2. Otsu method gives a threshold to segment brighter objects

M. Kamel and A. Campilho (Eds.): ICIAR 2013, LNCS 7950, pp. 326–335, 2013.

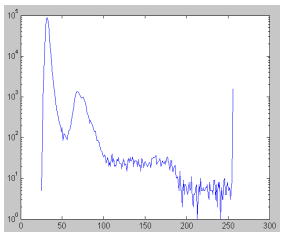

Fig. 3. Gray level histogram (vertical scale is logarithmic)

2 Chip Images Analysis

2.1 Generalized Otsu Method

In this problem, we want to compute more than one threshold. In fact we want to segment four regions (three thresholds). The classical Otsu method computes a discriminator function that measures threshold quality. For a hypothetical threshold U, this quantity is computed treating the histogram as a probability density function that is divided in two partial distributions by U. The discriminator is then: $D = \sigma^2 - (w_1\sigma^2_1 + w_2\sigma^2_2)$ called the "between-class variance"; where σ is the variance of the whole histogram, σ_i are the partial variances and w_i are the total probabilities of each partial distribution. It is not difficult to prove that the former equation is equivalent to[1]:

$$D = w_1(\mu - \mu_1)^2 + w_2(\mu - \mu_2)^2 \tag{1}$$

Where μ and μ_i are the global and partial means (respectively). Equation 1 can also be seen as the scalar version of a "between-class scatter matrix" [4]. This equation can be further simplified to: $D = w_1 w_2(\mu_1 - \mu_2)^2$, which is much faster to compute[2]. Practical implementations make a recursive computation of the last expression for all possible values of U (generally 0-255) and then they select the threshold that maximizes D.

To compute N thresholds, we should consider the histogram divided into N+1 partial probability functions. Applying Otsu's philosophy, the expression for D will be:

$$D = \sigma^2 - \sum_{i=1}^{N} w_i\sigma^2_i \tag{2}$$

Which, with analogous method, we can transform into:

$$D = \sum_{i=1}^{N} w_i(\mu - \mu_i)^2 \tag{3}$$

---

[1] To prove this we have to develop both expressions (original Otsu and equation 1), making simple substitutions such as $E[x^2] = w_1E[x^2|1]+w_2E[x^2|2]$, we will get the same.

[2] Again we must develop both equations knowing also that $w_1+w_2 = 1$.

This can again be seen as a 1x1 "between-class scatter matrix" [4]. The problem with equation 2 (that does not admit a simpler form this time) is that we have to compute it for all the possible N-tuples of thresholds and that will typically be a very large number. For a typical image of 256 gray levels, we would have a big number of N-tuples:

$$\text{order} = \frac{256!}{N!(256-N)!} \tag{4}$$

That is the number of different combinations of N different numbers (assuming thresholds are always ordered so that combinations must differ in at least one element)... For the chip problem, we need 3 thresholds (N=3): 2.7 millions of 3-tuples. This process was executed on an Intel Core i3 3.07 GHz (Matlab implementation) taking 4 minutes and 36 seconds. Solution is optimum (Fig. 4) but not good for real time.

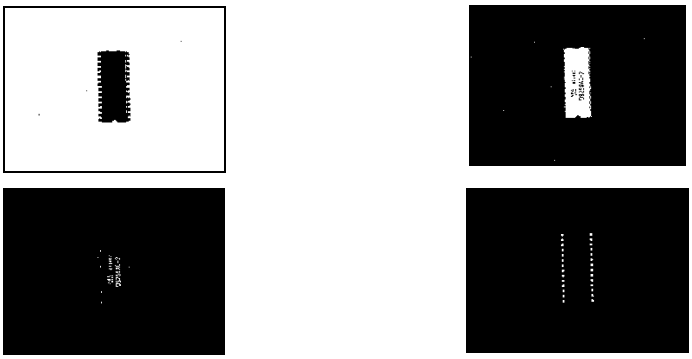

Fig. 4. Segmentation of chip image using generalized Otsu, 4 detected regions: background, chip body, chip text and chip pins

To check validity of this result we performed a more thorough test. We ran this test over 23 images containing one single chip and over other 22 images containing more than one chip. Method got the correct result for the whole set of 45 images. Average execution time was 4 minutes and 47 seconds.

Afterwards, we developed a method to improve the computing time, based on a reduction of the number of BINS used in histogram computation. The thresholds found are not optimal and the resulting segmentation is not very good (Fig. 5). But we can refine this result testing in a neighborhood of these non accurate thresholds (choosing the maximum D from equation 3). With his method we have got the optimum result (the same values that produced Fig. 4) in only 5.47 seconds using 32 BINS for the first approach and neighborhoods of 33 points.

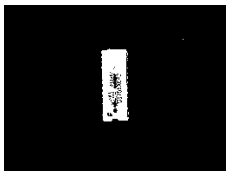

Fig. 5. Chip body segmentation with reduced histogram of 32 BINS

Testing with the whole test set we discovered that it did not work for some images. We thought that the problem was due to the excessive preponderance of main histogram peaks over secondary ones. Thus we tried some histogram normalization schemes. Best results were obtained for a very simple non-linear normalization: computing logarithm of histogram values (the idea came from the fact that we always needed logarithmic scales to make graphic representations of these histograms while linear scale graphics show only the two dominant peaks). Running now a test on the 45 test images yielded correct segmentations for all the images. Average execution time was 5.54 seconds (including log normalization). Computed thresholds were equal to the optimal ones in 15 images (one of each 3). In the rest of the cases, thresholds were close to optimal and still able to segment the images. Average difference from optimal thresholds to approximate ones was 1.9%.

Conclusion up to this point is that we have got to compute thresholds for multimodal histograms in a "reasonable" time (provided that implementing this algorithm in C/C++ reduces execution time by at least one order of magnitude as it is usual).

2.2 Sequential Otsu

Our next idea was to apply Otsu algorithm "on the partial distributions that result from Otsu algorithm": doing so twice, we would get three thresholds. Are those thresholds useful? The answer is yes. The first time (global Otsu) we will get a threshold that splits the darker parts (chip body and background) from the brighter ones (pins and characters). In the first partial distribution, we will segment the background and chip body. In the second, we will segment characters (darker) and pins (brighter).

Execution time was that of three Otsu processes[3] (total: 14 milliseconds, always Matlab code on Intel Core i3 3.07 GHz). The segmented images are almost identical to those of figure 4 (new thresholds differ 2.6% from the optimal ones). We are always considering "optimal" the results with maximum Otsu parameter (D).

Making the thorough test with this sequential method we again met problems due to peak unbalance. Applying again log normalization we got correct segmentations for all the images. Average execution time was now of 13 milliseconds (an impressive improvement from previous method). Nevertheless results were a bit worse: we got

[3] To make fair comparisons, we implemented the original Otsu algorithm in Matlab code (`.m` functions). If using **`graythresh`** function we would get better results for Otsu because this functions uses compiled C code in Matlab libraries.

the exact optimal thresholds in no images. Average difference from optimal thresholds to approximate ones was 3.5%.

At this moment we can say that we can implement multimodal thresholding in real time.

2.3 Peak Detection Method

Doing a heuristical study of the histogram curve (Fig. 3), we can identify the different regions that we want to distinguish (Fig. 6). We discovered that regions can be detected if we know the most outstanding points (particularly the local maxima of the histogram, marked LM). Notice that the threshold between two regions will be the local minimum between two local maxima. See also that each region has one local maximum[4].

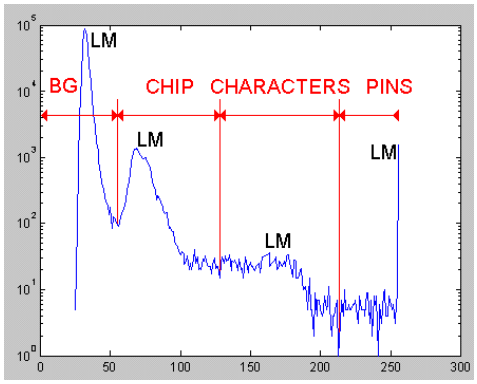

Fig. 6. Heuristical study of histogram: there are 4 regions (background, chip, characters and pins). Local maxima are marked LM, region frontiers approximately coincide with local minima

We have designed a method to detect local maxima and use them for threshold detection. The method is based on non-linear neighborhood filtering. We define a sliding neighborhood and we center it on all the values of L (luminance). When the center value is the absolute maximum of the neighborhood the output vector is this maximum value, otherwise output vector is zero. This idea was inspired on one-dimensional mathematical morphology [5]. Applying this method, with a neighborhood of 51 points, we locate the 4 local maxima labeled as LM in figure 6 (neighborhood filtering is enough to erase local fluctuations). Finding the minimum between each two maxima, we will have 3 thresholds. Applying these thresholds to image segmentation yields good results (Fig. 7).

Following the classification in [2] this method is a “Histogram-Shape” method (Otsu method, and derivative ones are “Clustering Based Methods”).

[4] See that global maximum is also a local maximum.

In the example image, thresholds computed this way differ 7.5% from the optimal ones (generalized Otsu). This could seem an important error but it is not so much if these thresholds are in "almost flat" regions of the histogram. If we compute Otsu discriminators for both the optimal and suboptimal values the difference is only 0.60%. Speed is the biggest advantage of this new method. Execution time for this method is 2 milliseconds (on the same machine of the preceding examples).

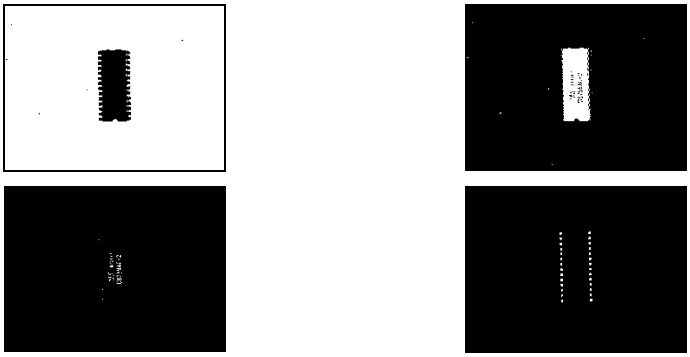

Fig. 7. Results of the peak detection method

Performing the complete test, we found several errors, after much testing we developed the following algorithm:

- Compute histogram effective length, defined as: number of values from first to last non-zero values.
- Histogram is first normalized in the following way: each histogram value is divided by the histogram mean in a neighborhood of that point. Neighborhood size is the histogram effective length divided by 4. This avoids dominant peak to hide others, in this case logarithm is not effective because is a monotonic function.
- Peak detection is performed as explained in the above paragraph. Neighborhood size begins by being half the histogram effective length.
- As we know that we need N thresholds, we are searching for M = N+1 local maxima. If we already have enough ones, stop searching. Otherwise we repeat search with half neighborhood size searching new maxima.
- Now we have enough maxima, but we could have more than M. We will prioritize those found in earlier stages and in the same stage we prioritize local maxima with larger histogram value.
- Now we have M = N+1 maxima. We will compute N thresholds; each of them will be between two consecutive maxima. Between each to maxima we compute three values: the central point, the local minimum and the median (the point that leaves equal histogram area before and after itself). We order the three numbers and we elect the central one.

See that with this version we do not need to compute or estimate a neighborhood size but we have to know how many thresholds we are searching (equal to the previous methods).

Results were not as good as in the previous examples. For 45 test images we got 37 correct segmentations and 8 with some problem (almost always in the "chip text" part), this is an 82% correct segmentation rate. Execution time results are outstanding: 6 milliseconds per image (average execution time). Thresholds for this method differ 10% in average from the optimal ones.

Perhaps the chip segmentation is too difficult for this method, that we think that is anyway promising.

3 Bimodal Histograms

In this section we will describe some tests that we performed to see if our peak detection algorithm is a real alternative to Otsu method.

3.1 True Bimodal Histogram

The first test image was an election paper, used to vote for the regional government (Fig. 8). This image has a very clear bimodal histogram (Fig. 9). In Fig. 9 we have shown the thresholds obtained by the two methods: peak detection and Otsu. We can see that both thresholds are not very different (they differ 2.8%) and that binarized images are almost equal (Fig. 10).

Execution time is 7.2 milliseconds for Otsu (Matlab `.m` code, not `graythresh` function) and 5.1 milliseconds for peak detection. Peak detection still outperforms Otsu in computing time but advantage is not very high.

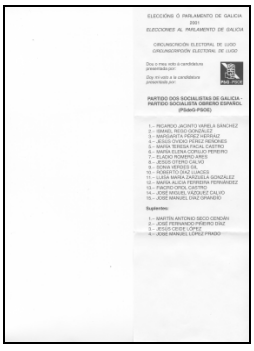

Fig. 8. Election paper

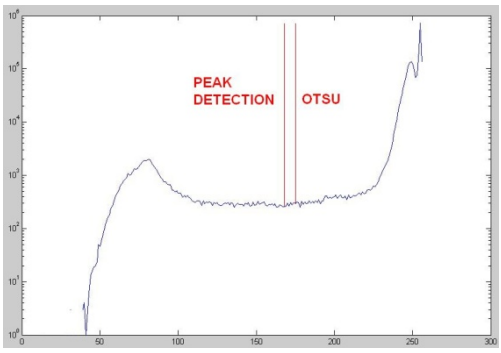

Fig. 9. Histogram and thresholds

Fig. 10. Binarized portions (above: Otsu, below: Peak Detection)

To perform a thorough test we used a collection of 69 scanned documents. Although we got correct segmentations for both Otsu and Peak Detection methods, we saw that in the Peak Detection case we could simplify the histogram normalization stage: instead of dividing histogram by its own mean we did a mean filtering with the same neighborhood size. Working in this manner we got correct segmentations for all the images. Threshold computing time was of 6.3 milliseconds for Otsu method and 4.9 milliseconds for Peak Detection. We see that Peak Detection is an eligible method for bimodal segmentation but advantage in computing time is small.

3.2 Complex Histogram

As our final test, we have used car plate images. Nowadays there are a lot of car plate number readers, like [6]. Typically these systems will be able to segment an image like figure 12 from a complete car image (Fig. 11).

Fig. 11. Car Image

Fig. 12. Segmented plate

In this case histogram is bimodal but very noisy (Fig. 13). We decided to test the same method of the previous section, trusting that the mean filter normalization would smooth histogram and still get good results.

Testing with 11 plate images, we got correct segmentations for all with both methods: Otsu and Peak Detection. Threshold computing time was of 7.2

milliseconds for Otsu method and 4.4 milliseconds for Peak Detection. Again, we see that Peak Detection is an eligible method; advantage in computing time is now a bit larger[5].

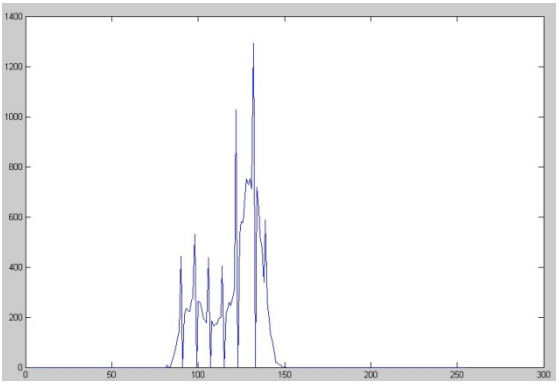

Fig. 13. Complex histogram (vertical scale is linear)

Fig. 14. Example of binarized plates (left: Otsu, right: Peak Detection)

4 Discussion

4.1 General Conclusions

We have studied the automatic computation of thresholds for segmenting gray level images. Threshold computation is based on image histogram. We have designed methods for multimodal histograms. Two of them are based on generalizing the classical Otsu method; the last one (peak detection) is simpler and more intuitive. On the issue of bimodal histograms: we have made a set of test cases to compare peak detection and Otsu. We have found that peak detection is almost always a valid alternative.

In multimodal analysis, the optimum (generalized Otsu method) is very slow, needing some approximation to be usable in real time (the other Otsu based methods could be the election). Peak detection is very fast for multimodal analysis but it is also the less accurate. Peak detection continues to be faster in bimodal analysis, but here its advantage is lesser.

[5] Although we compute histograms with 256 bins, resulting histogram may have a lesser effective length (due to zeros in histogram), both methods take advantage of this resulting in slight differences in computing time when input histogram changes.

Peak detection can be a useful method in other circumstances: whenever we want to locate peaks in a signal (time domain or frequency analysis), this technique can be helpful.

4.2 Comparison to Other Methods

Out main source of comparison has been Otsu method because it is the standard (at least for bimodal histograms). In [7] and [8] we found other methods for multimodal thresholding. In the first one, they convert thresholding in an iterative process getting similar results but they do not publish execution speed data. In the other paper, they make a histogram pre-segmentation based on minima computation and then they model each segment as a Gaussian. Again they get interesting results but do not mention time efficiency.

4.3 Future Lines

Beyond the scope of this paper, we have tested sequential Otsu method (section 2.2) as a tool for optimal quantization design. See that an optimal quantizer will divide a scalar value in N regions, where the region means (μ_i) will minimize the MSE (Mean Square Error). MSE equation ($mse = \sum_{i=1}^{N} p(reg-i)E[(x-\mu_i)^2 \mid reg-i]$) is equivalent to the subtracted sum in equation 2. The conclusion is that minimizing MSE is equivalent to maximizing Otsu discriminant. Id EST: selecting N-1 thresholds via sequential Otsu, we will desing a quantizer with N regions (tipically $N=2^n$).

Other possible future lines for this work are:

- Testing peak detection in other examples to continue exploring the strengths and weaknesses of this algorithm.
- Exploring better optimizations for generalized Otsu.

References

1. González, R.C., Woods, R.E.: Digital Image Processing, 3rd edn. Prentice Hall, Upper Saddle River (2008)
2. Sankur, B., Sezgin, M.: Image Thresholding Techniques: A Survey over Categories. Pattern Recognition 34(2), 1573–1607 (2001)
3. Otsu, N.: A Threshold Selection Method for Gray Level Histograms. IEEE Transactions on System, Man and Cybernetics 9(1), 62–66 (1979)
4. Fukunaga, K.: Introduction to Statistical Pattern Recognition. Academic Press, S. Diego (1990)
5. Serra, J.: Image Analysis and Mathematical Morphology. Academic Press, London (1982)
6. Hermida, X.F., Rodríguez, F.M., Lijó, J.L.F., Sande, F.P., Iglesias, M.P.: Automatic and Real Time Recognition of V.L.P.'s (Vehicle License Plates). In: Del Bimbo, A. (ed.) ICIAP 1997. LNCS, vol. 1311, pp. 552–559. Springer, Heidelberg (1997)
7. Assidan, E., El-Zaart, A.: Fast Optimal Multimodal Thresholding on Between-Class Variance Using a Mixture of Gamma Distributions. In: Proceedings of IEEE-ICSMC 2009 (2009)
8. Chang, J.H., et al.: Multi-Modal Gray-Level Histogram Modeling and Decomposition. Image and Vision Computing 20, 203–216 (2002)