Thresholding by Voting on Text Based Documents

A concrete analysis on existing algorithms and methods on evaluating them

Atanasiu Alexandru Marian
Masters GMRV
Fac. Automatics si Computer Science
University Politehnica Bucharest

Damian Petrisor Alin
Masters GMRV
Fac. Automatics si Computer Science
University Politehnica Bucharest

Panaitescu Cristian
Masters GMRV
Fac. Automatics si Computer Science
University Politehnica Bucharest

Abstract* — *Cuurent paper review present techniques of thresholding on text content images and proposes solutions on evaluation using voting techniques.

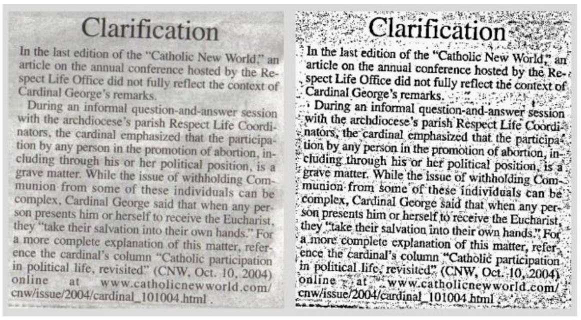
Key words* — *thresholding, voting, images, OTSU, Adaptive Gaussian, Adaptive Integral Image, Souvola's method, Lu's method, mean, spread, variation, python, histogram, entropy

I. Introduction

The focus of this paper is proposing methods of qualitative evaluation of existing implementations of thresholding algorithms and how can we improve chosing the greatest output given a fixed set of algorithms.

A. Thresholding

Thresholding is a class of segmentation algorithms which aims to split the information in an image into 2 classes: a foreground class and a background class (as opposed to general segmentation which aims to generate k classes representing k distinct entities in the image).



Clarification

In the last edition of the "Catholic New World," an article on the annual conference hosted by the Respect Life Office did not fully reflect the context of Cardinal George's remarks.

During an informal question-and-answer session with the archdiocese's parish Respect Life Coordinators, the cardinal emphasized that the participation by any person in the promotion of abortion, including through his or her political position, is a grave matter. While the issue of withholding Communion from some of these individuals can be complex, Cardinal George said that when any person presents him or herself to receive the Eucharist, they "take their salvation into their own hands." For a more complete explanation of this matter, reference the cardinal's column "Catholic participation in political life, revisited" (CNW, Oct. 10, 2004) online at www.catholicnewworld.com/cnw/issue/2004/cardinal_101004.html .

Clarification

In the last edition of the "Catholic New World," an article on the annual conference hosted by the Respect Life Office did not fully reflect the context of Cardinal George's remarks.

During an informal question-and-answer session with the archdiocese's parish Respect Life Coordinators, the cardinal emphasized that the participation by any person in the promotion of abortion, including through his or her political position, is a grave matter. While the issue of withholding Communion from some of these individuals can be complex, Cardinal George said that when any person presents him or herself to receive the Eucharist, they "take their salvation into their own hands." For a more complete explanation of this matter, reference the cardinal's column "Catholic participation in political life, revisited" (CNW, Oct. 10, 2004) online at www.catholicnewworld.com/cnw/issue/2004/cardinal_101004.html .

Fig.1 Thresholding on a photo of a text document

The simplest definition of a threshold algorithm - let I be an Image defined as a function:

$$I: \{0,1, \dots w\} \times \{0,1, \dots h\} \rightarrow [0,1] \tag{1}$$

Where w is the width of the image and h is the height of the image, the thresholding of the image would transform it in a function:

$$T: \{0,1, \dots w\} \times \{0,1, \dots h\} \rightarrow \{0,1\} \tag{2}$$

0 and 1 being the two distinct classes we want to distinguish each pixel of the image. Usually 0 is the foreground class and it is represented by a full black pixel and 1 is the background class and is represented by a full white pixel.

B. Thresholding algorithm types

Threshold algorithms classify into the following categories:

- Histogram shape algorithms
- Clustering-based algorithms
- Entropy-based algorithms
- Local algorithms

Histogram shape algorithms analyze the histogram of the image (like peaks, valleys and curvatures). The histogram can be defined as the distribution of discrete values (usually pixels have a descrete representation eg: 0-255), mathematically can be described as:

$$h: \{0,1, \dots 255\} \rightarrow \mathbb{N} \tag{3}$$

with the property of:

$$h(v) = c \tag{4}$$

if there are exactly c pixels in the image that have the value v.

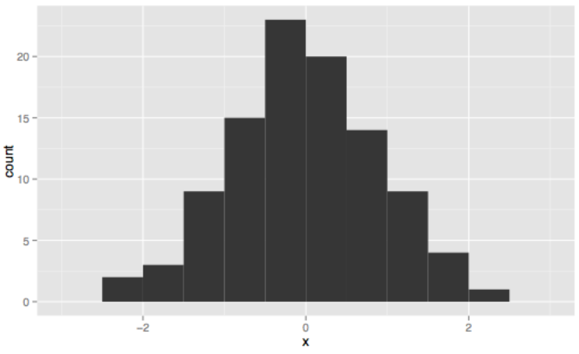


Fig.2 Example of histogram which details a gaussian distribution

and we will define the probability function which normates on the total count of the pixels

$$p: \{0,1, \dots 255\} \rightarrow [0,1] \tag{5}$$

$$p(v) = \frac{h(v)}{w \cdot h} \tag{6}$$

Clustering techniques investigate splitting two clusters of greylevels which are gaussian distributed in order to separate the background and the foreground class.

Entropy algorithms compute entropies of foreground and background images and compare the cross entropy of the result image in order to compute the variation of the entropy.

Local algorithms split the image into windows and apply diverse algorithms in order to compute a local threshold. Usually this applies if lighting conditions of the image imply a nonlinear thresholding finding problem (there are regions that differ drastically in the thresholding point).

In that case a local algorithm is required since there can't be a single threshold value that separates foreground for background, and there is a threshold value for each region that is *differently lighted*.

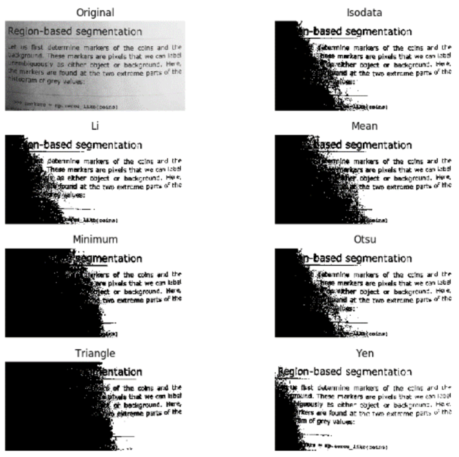


Fig.3 Example of results of trying a global thresholding algorithm on a non-linear thresholding problem

II. THE ALGORITHMS

A. *Thresholding Algorithms used*

For implementation we have studied and tweaked several thresholding algorithms, from very simple and basic techniques to complex heuristics and formulas in order to detect the thresholding of the image. Important to note is that the current article reviews thresholding on text rich images that do not have images or other kind of media content besides text.

The algorithms in our system tend to search in 256 levels greyscale images using different methods a global threshold level (let it be called $t \in \{0,1 \dots 256\}$) which will separate the classes as follows: pixels with values below t will be part of the foreground (and will have the value 0 in the resulting image) and the pixels with the value above t will be part of the background (and will have the value 1 in the resulting image).

Our implementation starts with **Otsu's method** as the starting point for searching a thresholding point. Otsu's method computes the threshold value by separating the classes such that their combined spread is maximal (minimizing the inner class spread).

Let I be an image defined like in (1), we define ${\sigma_0}^2(t), {\sigma_1}^2(t)$ the variance of the two classes for a given t threshold value and $\omega_0(t), \omega_1(t)$ - probabilities of the two classes (pixel count of each class / total number of pixels). Otsu defines intra-class varience as:

$$ {\sigma_\omega}^2(t) = \omega_0(t){\sigma_0}^2(t) + \omega_1(t){\sigma_1}^2(t) \qquad (7)$$

$$\begin{cases} \omega_0(t) = \sum_{i=0}^{t-1} p(i) \\ \omega_1(t) = \sum_{i=t}^{255} p(i) \end{cases} \qquad (8)$$

The algorithm implies the steps:

1. Compute $h(v)$ and $p(v)$
2. Set initial $\omega_i(0)$ and $\mu_i(0)$
3. for each i in available values (0…255)
    a. Update ω_i and μ_i
    b. Compute ${\sigma_b}^2(i)$
4. Use the threshold where ${\sigma_b}^2(t)$ is maximum

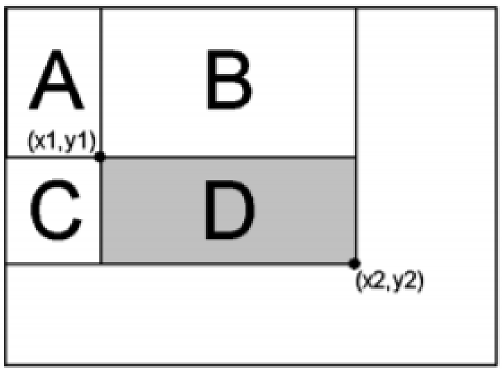
Integral Image here

An integral image (also known as a summed-area table) is a tool that can be used whenever we have a function from pixels to real numbers f(x,y) (for instance, pixel intensity), and we wish to compute the sum of this function over a rectangular region of the image.

To compute the integral image, we store at each location, I(x,y), the sum of all f(x,y) terms to the left and above the pixel (x,y). This is accomplished in linear time using the following equation for each pixel (taking into account the border cases). (ec1) Once we have the integral image, the sum of the function for any rectangle with upper left corner (x1,y1), and lower right corner (x2,y2) can be computed in constant time using the ec2 . Figure ? illustrates that computing the sum of f(x,y) over the rectangle D using Equation 2 is equivalent to computing the sums over the rectangles (A+B+C+D)-(A+B)-(A+C)+A.

I(x,y) = f(x,y) +I(x−1,y) +I(x,y−1)−I(x−1,y−1) (ec1)

$$\sum_{x=x_1}^{x_2} \sum_{y=y_1}^{y_2} f(x,y) = \mathrm{I}(x_2,y_2) - \mathrm{I}(x_2,y_1-1) - \mathrm{I}(x_1-1,y_2) + \mathrm{I}(x_1-1,y_1-1) \quad \text{(ec 2)}$$



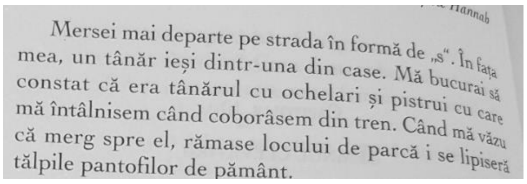
Our adaptive thresholding technique is a simple extension of Wellner's method. The main idea in Wellner's algorithm is that each pixel is compared to an average of the surrounding pixels. Specifically, an approximate moving average of the last s pixels seen is calculated while traversing the image. If the value of the current pixel is t percent lower than the average then it is set to black, otherwise it is set to white. Our technique is clean, straightforward, easy to code, and produces the same output independently of how the image is processed. Instead of computing a running average of the last s pixels seen, we compute the average of an s x s window of pixels centered around each pixel. This is a better average for comparison since it considers neighboring pixels on all sides

```
for $i = 0$ to $w$ do
  for $j = 0$ to $h$ do
    $x1 \leftarrow i - s/2$ {border checking is not shown}
    $x2 \leftarrow i + s/2$
    $y1 \leftarrow j - s/2$
    $y2 \leftarrow j + s/2$
    $count \leftarrow (x2 - x1) \times (y2 - y1)$
    $sum \leftarrow intImg[x2, y2] - intImg[x2, y1 - 1] - intImg[x1 - 1, y2] + intImg[x1 - 1, y1 - 1]$
    if $(in[i, j] \times count) \leq (sum \times (100 - t)/100)$ then
      $out[i, j] \leftarrow 0$
    else
      $out[i, j] \leftarrow 255$
    end if
  end for
end for
```

B. *Voting System and Heuristics*

In order to be able to choose a fit threshold we studied several heuristics to be able to vote out the most fit thresholdings that our algorithms can do. Our voting system takes into consideration **multiple elections** in order to come up with a certain set of proper thresholds on the given image.

The first election of the voting algorithm is based on eliminating trivial candidates that don't represent a viable threshold. Since we started with the assumption that our input images are photos taken from books/documents with text only, we can impose that whatever image we will apply our algorithms on, the ratio between foreground and background pixels cannot be beyond a certain ratio (eg: 10%) and above a certain ratio (eg: 90%) thus, eliminating images like:


Mersei mai departe pe strada în formă de „s". În fața
mea, un tânăr ieși dintr-una din case. Mă bucurai să
constat că era tânărul cu ochelari și pistrui cu care
mă întâlnisem când coborâsem din tren. Când mă văzu
că merg spre el, rămase locului de parcă i se lipiseră
tălpile pantofilor de pământ.

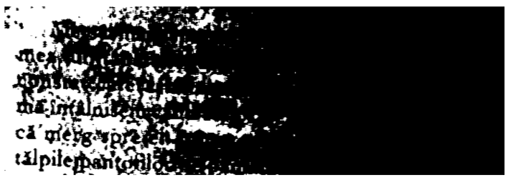


Fig.4 The input image and a trivial case of elimination where the threshold chosen was too high and the ratio was too high

A second election of the voting algorithm makes a similar approach, but it applies it locally on windows approximative size of either 1 character or multiples of that size (2 x 2, 4 x 4 characters in pixels) thus assuring that although globally the threshold applies the (eg: the ratio of classes is around 60-70% - there are regions where it is over 95% which clearly means). Although here we may use fixed thresholds like in the first step, this might most likely fail if we have for example page margins (where we don't have foreground elements).

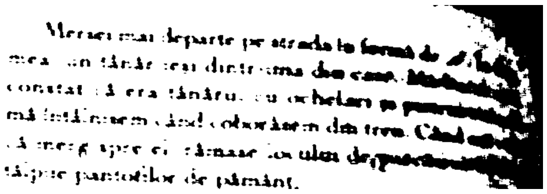


Fig.5 Although this binarization passes the first step, there are windows of size of a character where we can detect that the threshold is not proper

So, in order to consider this, we compute the histogram of the input image (which for convenience will have 256 posible values) on the current window and we execute two eliminatory tests:

- the average test
- the entropy test

The average test computes the ponderated average value of the histogram of the initial window of the image:

$$A_i = \frac{\sum I(i,j)}{w^2} \tag{9}$$

where w is the window's length; and the average between the threshold values:

$$A_t = \frac{h_t(0) + h_t(1)}{w^2} \tag{10}$$

The test implies that the ratio between the two averages should be in the same range as the ratios set in the first step.

The entropy test computes the entropy of the entry window:

$$S_i = -\sum p(i) log\,[p(i)] \tag{11}$$

And the entropy of the generated by the generated threshold

$$S_t = -p(0)\log\big(p(0)\big) - p(1)\log\big(p(1)\big) \tag{12}$$

and we impose that the ratio between entropies should be more that 2 (meaning that once we group more values into a single value, we can consider that the entropy mai in the worst case halve).

Although this step is computationally more expensive that the first, it is easy to parallelize on SIMD architectures (using for instance GPGPU), since we have fixed size of chunks, and we compute the same steps on different chucks of data, improving this way the overall performance of the algorithm.

The third election is a tournament-based step where we intend to do in 2 steps: once globally (on all thresholds) and one on groups of data (eg 16 player in a batch); where intend to eliminate a ratio of the players (eg 25% of the oddest players).

In order to do that we compute a probability image for each pixel which will be the ratio between how many times that pixel was foreground and total number of images in a batch. This image we will threshold in the middle (50% +/- 10%). Whatever pixels are below will be class 0 and whatever are above they are class 1. Now we can evaluate each image to see how many pixels match this thresholded probability image. The least 25% are dropped from the tournament.

Doing one such step globally and two such steps locally and again one globally assure keeping around only 30% of the original thresholds (which succeeded into this tournament). After each of these steps we need to reshuffle the data batches in order to not propagate same

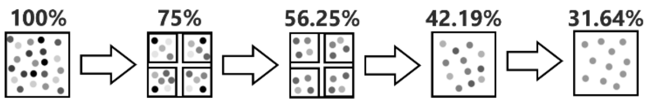


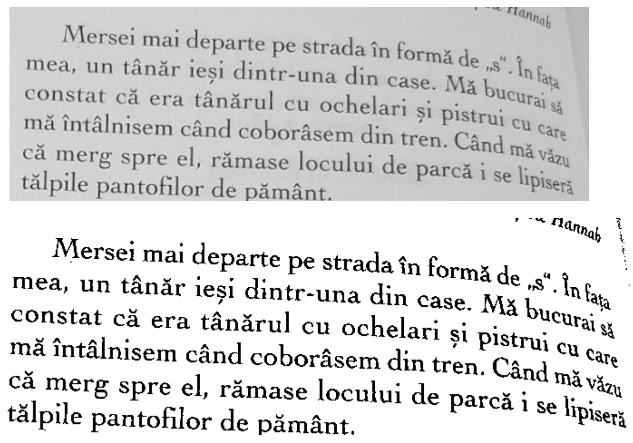
Fig.5 The tournament election that removes around 70% unfit candidates

The last step which is not an election takes the remaining images, makes a similar image to the last step which will become grayscale, and similarly to the last step we will threshold at the middle value (50%), and the result image will be the final chosen as the final threshold.

III. The Implementation and Results

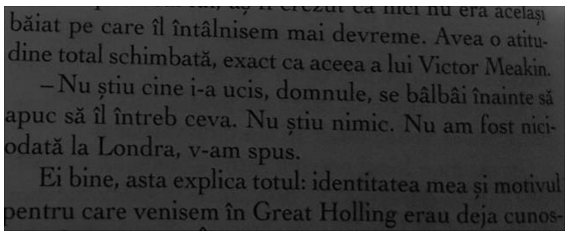
The system that was implemented for this article was written in Python and GNU Linux Bash. There is a central bash script which tests on multiple images the algorithms.

As python implementation there is a script that implements all the thresholding variation to create the pool of tests and one script used for evaluation. For opening and computing on images we used OpenCV.


Mersei mai departe pe strada în formă de „s". În fața mea, un tânăr ieși dintr-una din case. Mă bucurai să constat că era tânărul cu ochelari și pistrui cu care mă întâlnisem când coborâsem din tren. Când mă văzu că merg spre el, rămase locului de parcă i se lipiseră tălpile pantofilor de pământ.

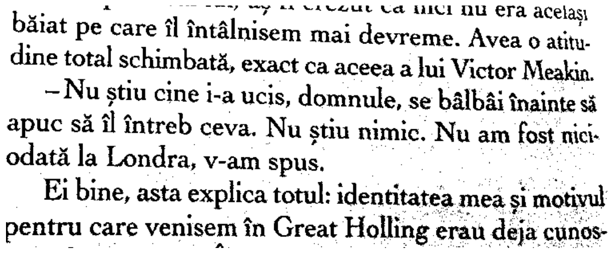
Mersei mai departe pe strada în formă de „s". În fața mea, un tânăr ieși dintr-una din case. Mă bucurai să constat că era tânărul cu ochelari și pistrui cu care mă întâlnisem când coborâsem din tren. Când mă văzu că merg spre el, rămase locului de parcă i se lipiseră tălpile pantofilor de pământ.

Fig.5 The basic test - good lighting


băiat pe care îl întâlnisem mai devreme. Avea o atitu-dine total schimbată, exact ca aceea a lui Victor Meakin.

–Nu știu cine i-a ucis, domnule, se bâlbâi înainte să apuc să îl întreb ceva. Nu știu nimic. Nu am fost nici-odată la Londra, v-am spus.

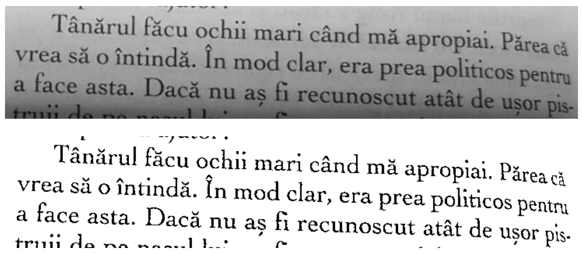
Ei bine, asta explica totul: identitatea mea și motivul pentru care venisem în Great Holling erau deja cunos-


băiat pe care îl întâlnisem mai devreme. Avea o atitu-dine total schimbată, exact ca aceea a lui Victor Meakin.

–Nu știu cine i-a ucis, domnule, se bâlbâi înainte să apuc să îl întreb ceva. Nu știu nimic. Nu am fost nici-odată la Londra, v-am spus.

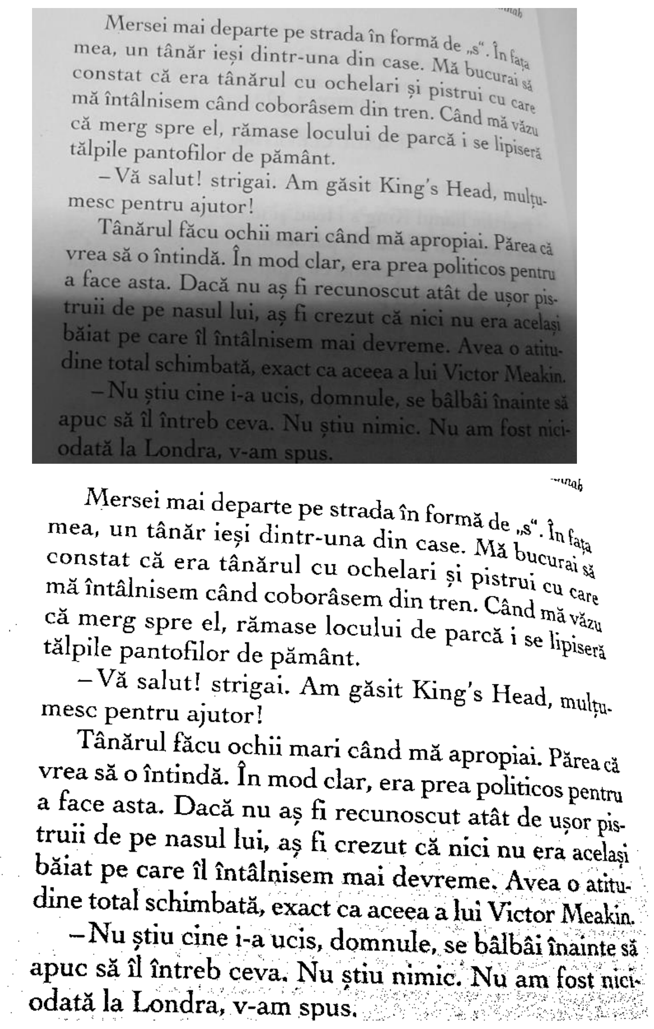
Ei bine, asta explica totul: identitatea mea și motivul pentru care venisem în Great Holling erau deja cunos-

Fig.6 Poor lighting test - threshold is lower


Tânărul făcu ochii mari când mă apropiai. Părea că vrea să o întindă. În mod clar, era prea politicos pentru a face asta. Dacă nu aș fi recunoscut atât de ușor pis-

Tânărul făcu ochii mari când mă apropiai. Părea că vrea să o întindă. În mod clar, era prea politicos pentru a face asta. Dacă nu aș fi recunoscut atât de ușor pis-

Fig.7 Light varies on the page a little


Mersei mai departe pe strada în formă de „s". În fața mea, un tânăr ieși dintr-una din case. Mă bucurai să constat că era tânărul cu ochelari și pistrui cu care mă întâlnisem când coborâsem din tren. Când mă văzu că merg spre el, rămase locului de parcă i se lipiseră tălpile pantofilor de pământ.

–Vă salut! strigai. Am găsit King's Head, mulțu-mesc pentru ajutor!

Tânărul făcu ochii mari când mă apropiai. Părea că vrea să o întindă. În mod clar, era prea politicos pentru a face asta. Dacă nu aș fi recunoscut atât de ușor pis-truii de pe nasul lui, aș fi crezut că nici nu era același băiat pe care îl întâlnisem mai devreme. Avea o atitu-dine total schimbată, exact ca aceea a lui Victor Meakin.

–Nu știu cine i-a ucis, domnule, se bâlbâi înainte să apuc să îl întreb ceva. Nu știu nimic. Nu am fost nici-odată la Londra, v-am spus.

Mersei mai departe pe strada în formă de „s". În fața mea, un tânăr ieși dintr-una din case. Mă bucurai să constat că era tânărul cu ochelari și pistrui cu care mă întâlnisem când coborâsem din tren. Când mă văzu că merg spre el, rămase locului de parcă i se lipiseră tălpile pantofilor de pământ.

–Vă salut! strigai. Am găsit King's Head, mulțu-mesc pentru ajutor!

Tânărul făcu ochii mari când mă apropiai. Părea că vrea să o întindă. În mod clar, era prea politicos pentru a face asta. Dacă nu aș fi recunoscut atât de ușor pis-truii de pe nasul lui, aș fi crezut că nici nu era același băiat pe care îl întâlnisem mai devreme. Avea o atitu-dine total schimbată, exact ca aceea a lui Victor Meakin.

–Nu știu cine i-a ucis, domnule, se bâlbâi înainte să apuc să îl întreb ceva. Nu știu nimic. Nu am fost nici-odată la Londra, v-am spus.

Fig.8 Testing local thresholding (because there isn't a single threshold value)

References

[1] *"Image Segmentation Based on 2D Otsu Method with Histogram Analysis"* (2008) Jun Zhang , Junglu Hu

[2] *"Voting based image segmentation"(2013),* Costin Boiangiu, Radu Ioanitescu

[3] *"Applying localized Otsu for Watershed Segmented Images" (2015)* , Costin Boiangiu, Andrei Tigora

[4] *"Adaptive document image binarization"(2000)* - J. Souvola, M Pietikainen

[5] *"Voting algorithms" (1994)* - B. Parhami

[6] *"Adaptive Thresholding using Integral Image"(2007) -Derek Bradley, Gerald Roth*