Abstract – This paper describes an efficient edge detection algorithm that can be used as a plug-in for digital image processing systems. The proposed algorithm uses a method based on iterative clustering targeting a reduced number of operations. The algorithm splits the image into two parts, background and foreground, and calculates the mean value for each of them. Based on these results, the new threshold value will be obtained and looped until the mean values remain unchanged. The only pixels affected by the change are the pixels with values between the previous two thresholds, so only they have to be redistributed to a new class. As a result, only few operations are needed in order to obtain the desired threshold. All the algorithms and results obtained in this paper are developed and tested using the C# programming language.

Keywords—image processing; edge detection; thresholding algorithms; non-maximum suppression; cluster based methods.

I. INTRODUCTION

The problem of edge detection has a wide variety of uses for obtaining the complex information regarding an image, for pattern recognition as well as computer vision. As these methods are usually used in real time processing, the need of better edge detection methods arises. Faster methods are a necessity, but there is a feeble balance between speed and performance for such an application. For some steps usually used in edge detection techniques, such as filtering the noise and computing the gradient, applying a big, time consuming filter is a must for getting a certain degree of quality. For the grayscale conversion, non-maximum suppression and hysteresis, there are basic and rather quick algorithms, so they have a low priority for optimization. As opposed to these, the automatic thresholding methods have a wide variety of solutions, some with slower but better results, and others with quicker but non-optimal results. Because of this, appear the question of which algorithms are better suited for detection the value of the threshold for an image composed out of edges and having most pixel intensities of either close to zero or over a medium value.

The proposed algorithm is a clustering based method, in which, the pixels are clustered into two classes, which is efficient given the high variations of values between pixels in this type of image, followed by calculating the mean value for the pixels in each class and re-categorizing of some pixels.

The document is structured as follows. A quick overview of the algorithms used in each step of the edge detection, for which the thresholding method was tested and, also, a summary of the automatic thresholding methods currently in use are presented in section II. The next section presents the functionality of the entire algorithm proposed by this paper. Section IV contains experimental results of the edge detection in which the threshold value was calculated using the proposed method. In the last section, the paper will be concluded as well as presenting the future research goals.

II. BASICS OF EDGE DETECTION ALGORITHMS

A. Edge Detection

As presented in [1], in the edge detection there are three mandatory steps: grayscale conversion, calculating the gradient values for image and thresholding these values. For a better quality of the output image, two more steps can be added: noise filtering and non-maximum suppression.

Grayscale conversion is a simple process and is done by calculating a new value for each pixel based on the following formula.

$$Y = 0.289*R + 0.587*G + 0.114*B \quad (1)$$

The second step, if added, is noise filtering. Most images, especially those taken using a camera, have a fair amount of image noise, that will, most of the time, get in the way of detecting the edges properly or even lead to detecting false edges. There are various types of noises, each being susceptible to different types of filters [2], [3].

The next step is obtaining the gradient values for each pixel. A gradient is an increase or decrease in the magnitude of a certain property. For a pixel, the gradient is composed of two values: magnitude and direction. One way of calculating these is by applying gradient operators. Classic operators [4] such as Sobel and Prewitt calculate the gradient magnitude in two
...directions (vertical and horizontal), the magnitude and direction of the gradient are then calculated using (2) and (3).
\[ G = \sqrt{G_x^2 + G_y^2} \]  
\[ \theta = \tan^{-1} \frac{G_x}{G_y} \]  

where \( G_x \) and \( G_y \) are the previously calculated horizontal and vertical gradients.

If the image will be displayed as it is now, the edges would be as thick as a few pixels. The role of the next step named non-maximum suppression, is edge thinning, the process of removing the unwanted extra pixels from the edges of an image. After this operation, the image will have most or all edges with a thickness of one pixel. The non-maximum suppression algorithm has two steps that are applied for each pixel in the image:

- Compare the edge strength of the current pixel with the strength of the neighboring pixels in the same direction of the gradient.
- If the current pixel’s edge strength is higher, the value is kept. Otherwise, the value is suppressed.

B. Thresholding Methods

The end result of the edge detection process is a binary image and the purpose of thresholding is determining which pixels fall into each category [5]. The simplest thresholding method uses a single value of intensity and, depending on it, every pixel can belong to one of two categories:

- If the intensity of the pixel is lower than the threshold value, then the pixel is set to black.
- If the intensity of the pixel is higher than the threshold value, then the pixel is set to white.

The algorithm in itself is simple but the problem that appears in making the process automatic is having a method of calculating the optimal threshold value for the respective image.

There are six types of thresholding methods, as categorized by Sezgin and Sankur in [6]:

1. Histogram shape-based methods, where certain aspects of the smoothed histogram are analyzed.
2. Clustering-based methods, where the pixels are clustered in two parts, either background or foreground.
3. Entropy-based methods, that use the expected value of the information contained in each message received.
4. Object Attribute-based methods, based on the similarities between the newly obtained image and the grayscale one, such as edge coincidences and so on.
5. Spatial methods, that use the higher-order probability distribution or the correlation between pixels.
6. Local methods, where a threshold value is computed for each pixel, based on the neighboring pixels.

One of the most notable automatic thresholding algorithms is Otsu’s method presented in [7]. This algorithm is based on the assumption that the image contains two types of pixels, either background or foreground pixels. After doing so, it calculates the optimum threshold for separating the two classes so that their combined spread is minimal. Another histogram-based thresholding method is the Balanced histogram threshold [8]. It starts from the same assumption as Otsu’s method, separating the pixels in the two categories. Aside from the simple thresholding there is the double thresholding that uses two values, a high threshold and a low threshold. Depending on their value relative to the two thresholds pixels can be:

- Strong pixels, when their value is higher than the high threshold.
- Weak pixels, when their value is higher than the low threshold but lower than the high threshold.
- Very weak pixels when their value is lower than the low threshold. Their value will be set to 0, as they will be black pixels in the final image.

III. PROPOSED THRESHOLDING ALGORITHM

The thresholding algorithm proposed by this paper is devised into two steps.

- Calculating the initial threshold.
- Splitting the pixels into two classes and recalculating the new threshold value based on these.

The first step calculates the threshold value based on the following formula.

\[ t = \frac{\sum_{i=0}^{255} n_i \ast i}{\sum_{i=0}^{255} n_i} \]  

In equation (4), \( i \) represents the number of possible intensities of pixels from the digital image, \( n_i \) represents the number of pixels with intensity equal to \( i \) and \( t \) is the calculated threshold value.

The diagram of the proposed clustering based method to solve the automatic thresholding is presented in Fig. 1.

The first loop calculates the initial threshold value based on histogram of the image and using equation (4). In the second and third loop, the partial sums for each region are saved in the following variables: \( S_b \) or \( S_f \) for the total value of the pixel’s intensity in the...
background and respectively foreground regions; Nb or Nf for the total numbers of pixels in the two regions (background and foreground).

\[ h = \text{histogram(image)} \]
\[ i = 0 \]
\[ i < 256 \]
\[ S += h[i] \times i \]
\[ N += h[i] \]
\[ i++ \]

\[ \text{Yes} \]
\[ i = 0 \]
\[ i < t \]
\[ Sb += h[i] \times i \]
\[ Nb += h[i] \]
\[ i++ \]

\[ \text{Yes} \]
\[ Sb -= S \]
\[ \text{No} \]
\[ Nb -= N \]
\[ Sf += S \]
\[ Nf += N \]

\[ \text{No} \]
\[ Mb = Sb / Nb \]
\[ Mf = Sf / Nf \]
\[ Mb = Mb \]
\[ Mf = Mf \]
\[ T = (Mb + Mf) / 2 \]
\[ T > Ti \]
\[ i = T \]

\[ \text{No} \]
\[ Sb = S \]
\[ Nb = N \]
\[ Sf = S \]
\[ Nf = N \]
\[ i++ \]

\[ \text{Yes} \]
\[ Sb += h[i] \times i \]
\[ Nb += h[i] \]
\[ i++ \]

\[ \text{No} \]
\[ Sf += h[i] \times i \]
\[ Nf += h[i] \]
\[ i++ \]

\[ \text{No} \]
\[ Mb += Mb \]
\[ Mf += Mf \]

\[ \text{Yes} \]
\[ \text{STOP} \]

Figure 1. The diagram of the automatic thresholding algorithm

The results obtained after the first three loops will be used to obtain the desired threshold value that consists of:

a) Categorizing the pixels that might have to change classes: Initially, the pixels aren't categorized, so all of them are tested to find which class to be distributed to. From the second iteration onward only a part of the pixels have to be tested using the following logic: If \( T > Ti \) the only class of pixels that can be affected are the ones in the foreground. If \( T < Ti \) the only class of pixels that can be affected are the ones in the background. As a result of this logic, only few operations are needed in order to obtain the desired threshold.

b) Calculating the mean value for each class: We mention that the mean value is calculated as the sum value for each region divided by the total number of pixels in the respective region.

c) Calculating the new threshold value based on the results obtained in each class of pixels: In this step, we compute the new mean values for each class and compare them to the old ones. If the values obtained differ then the threshold will be calculated as a mean value between the two new mean values from each region and the process will be repeated. Otherwise, stop the detection process.

IV. TESTING AND RESULTS

The classical and the proposed method for image edge detection were fully implemented in software using the C# programming language. The main interface of the application is presented in Fig. 2.
The proposed thresholding algorithm will be applied for all of the selected filters (Canny, Prewitt, Sobel and Kirsch) and the results will be presented directly in the graphical visual interface of the application. For example, in Fig. 3, the implementation was tested for all algorithms on a digital image. In the right side of the picture is presented the image used for edge detection.

Another example, as can be seen in Fig. 4, shows a more complex scene, with overlapping edges and some regions where the colors of the foreground and background may match.

In this test also, most of the edges were detected correctly.

From the algorithm execution time point of view, it was tested on different images and the results are presented in Table I.

V. CONCLUSION

The automatic thresholding method presented in this paper, as seen from the tests, offers good results in detecting the edges of a digital image. The obtained image can be used further on in devices such as surveillance cameras or different software applications that use image processing for certain functions.

In the near future, in order to obtain high speed, the algorithm will be implemented in hardware, on a FPGA (Field Programmable Gate Arrays) device, with the possibility of sending/receiving images from the surveillance cameras or different software applications using either the TCP/IP protocol or through the USB cable. The device will detect the edges in the image and send back the image obtained for further processing purposes.

REFERENCES