Image Binarization

Juan Antonio Pérez Juárez
University of Guadalajara
Guadalajara, Jalisco
juan.perez0996@alumnos.udg.mx

I. INTRODUCTION

Image binarization is a fundamental preprocessing step in various image processing applications, particularly in the fields of computer vision and pattern recognition. The primary objective of binarization is to convert a grayscale image into a binary image, where each pixel is represented by either a 0 or a 1. This transformation simplifies the image data, making it easier to analyze and process while retaining essential structural information.

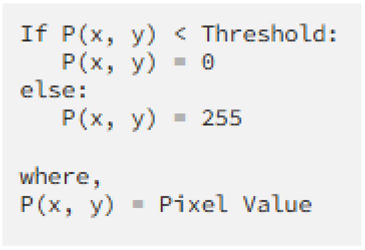
In the context of image analysis, binarization plays a crucial role in enhancing the visibility of objects of interest against the background. By reducing the complexity of the image, it facilitates subsequent tasks such as feature extraction, object detection, and image segmentation. Various methods exist for achieving binarization, ranging from simple thresholding techniques to more advanced adaptive algorithms that consider local pixel intensities.

This article focuses on a straightforward yet effective binarization method, which serves as a foundation for understanding more complex techniques. We will explore the underlying principles of this method, its implementation, and its applicability across different scenarios. Additionally, we will discuss the advantages and limitations of using a simple binarization approach in comparison to more sophisticated methods, providing a comprehensive overview of its role in image processing workflows. Through this exploration, we aim to highlight the significance of binarization in enhancing image analysis outcomes and its relevance in contemporary research and applications.

II. THRESHOLDING

Image Binarization, also known as Image Thresholding, is a technique to create a binary image from a grayscale or RGB image that can be used to separate the image's foreground from its background. Image thresholding is one of the most fundamental ways to extract useful information from a given image or segment of a region of interest.

Image thresholding is usually performed on grayscale images. Once the image is converted to a grayscale image, each pixel value is compared with a threshold value where, if the pixel value is less than the threshold; the pixel value is set to zero; else, it is set to the maximum possible value (255).



```
If P(x, y) < Threshold:
   P(x, y) = 0
else:
   P(x, y) = 255

where,
P(x, y) = Pixel Value
```

Fig 1. The simplest image thresholding

III. GLOBAL THRESHOLDING

A fixed threshold value is applied to every pixel, assuming that the image has a bimodal histogram separating the background from the foreground. The global thresholding algorithm compares each pixel intensity in the source image with the predefined threshold value and reassigns different intensity values if the intensity is larger or smaller than the threshold.

The most basic global thresholding technique is binary thresholding. OpenCV offers different types of thresholding, which can be provided as a parameter in the cv2.threshold function. The available thresholding types are as follows.

- cv2.THRESH_BINARY: If pixel intensity is greater than the threshold, sets the value to 255; else, sets it to 0.
- cv2.THRESH_BINARY_INV: Inversion of cv2.THRESH_BINARY.
- cv.THRESH_TRUNC: If pixel intensity is greater than the threshold, it is set to be the same as the threshold(truncated); else, the value remains the same.
- cv.THRESH_TOZERO: For all the pixels with intensity less than the threshold, the intensity value is set to 0
- cv.THRESH_TOZERO_INV: Inversion of cv2.THRESH_TOZERO.

In addition to the thresholding type, the following parameters should also be passed when calling the cv2.threshold function.

cv2.threshold(source, thresh, maxVal, type)
source — Input image in grayscale
thresh — Thresholding value
maxVal — Maximum intensity value of a pixel
type — Type of thresholding

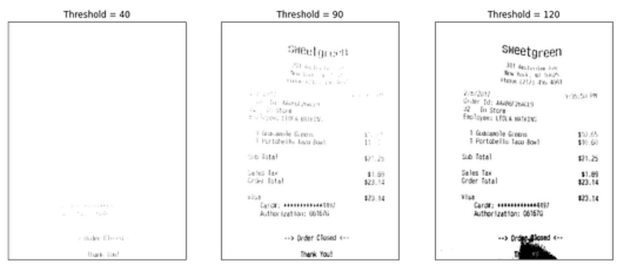


Fig 2. Global Thresholding with different values.

IV. ADAPTATIVE THRESHOLDING

A. Adaptive thresholding

A local thresholding method helps overcome the above drawback by calculating different thresholding values for different image regions.

In OpenCV the cv2.adaptiveThreshold function can be used to perform adaptive thresholding on a given image. There are two adaptive thresholding algorithms/methods provided by OpenCV that calculates the threshold value using different mechanisms.

This method calculates the threshold value for a given pixel by calculating the mean intensity of the neighboring pixels and subtracting a constant C from it. In OpenCV, Adaptive Mean Thresholding is defined by cv.ADAPTIVE_THRESH_MEAN_C and the operation can be mathematically represented as follows:

$$Threshold = Mean_{neighbour\,hood} - constant$$

B. Adaptative Gaussian method

This method calculates the threshold value for a given pixel by calculating the Gaussian weighted sum of the neighboring pixels and subtracting a constant C from it. In OpenCV, Adaptive Gaussian Thresholding is defined by cv2.ADAPTIVE_THRESH_GAUSSIAN_C and the operation can be mathematically represented as follows,

$$Threshold = GaussianWeightedSum_{neighbourhood} - constant$$

In addition to the adaptive thresholding algorithm, the following parameters should also be passed when calling the cv2.adaptiveThreshold function.

cv2.adaptiveThreshold(src, maxValue, adaptiveMethod, thresholdType, blocksize, C), where:

1. src — Input image.
2. maxValue — Maximum intensity value of a pixel.
3. adaptiveMethod — Algorithm that calculates the threshold value.
4. thresholdType — Type of thresholding (binary, binary inverse, etc.)
5. blocksize — Size of the pixel neighborhood used to calculate the threshold.
6. C — Constant to subtract from the mean or gaussian weighted sum of the neighborhood pixels.

V. OTSU THRESHOLDING

is a Global Binarization technique where the threshold value is selected automatically, assuming that the image's foreground and background create a bimodal distribution with 2 peaks (representing 2 classes). The optimal value between the two histogram peak values is usually considered the threshold value in this method.

Otsu Thresholding is a variance-based thresholding technique. The key idea is to find the threshold value where the weighted variance between the background and foreground pixels (2 classes) of the image is the minimum (minimizing weighted within-class variance/maximum weighted between-class variance). The algorithm iterates through all possible threshold values and measures the distribution of background and foreground pixels to find the threshold value where the spread is minimum.

The weighted within-class variance can be represented as follows:

$$\sigma_w^2(t) = w_1(t)\sigma_1^2(t) + w_2(t)\sigma_2^2(t) - - - - - -(1)$$

where w1(t) and w2(t) are the probabilities of the two intensity classes divided by a threshold t, with the value within the range 0–255. The terms of the equation can be expressed as:

$$w_1(t) = \sum_{i=1}^{t} P(i) \text{ and } w_2(t) = \sum_{i=t+1}^{I} P(i)$$

The overall Otsu Thresholding algorithm can be represented as:

1. Calculate the intensity histogram and intensity level probabilities.
2. Initialize $w_i(0)$ and $\sigma_i^2(0)$.
3. Iterate over all possible thresholds: t = 0, ... , max_intensity .
    a. Update the values of w_i, and σ_i^2, where w_i, is the probability and σ_i^2,, is the variance of class i.
    b. Calculate the within-class variance $\sigma_w^2(t)$
4. The final threshold is the minimum $\sigma_w^2(t)$.

In OpenCV the cv2.threshold function can be used to perform Otsu Thresholding on a given image by passing cv2.THRESH_OTSU as an extra flag. The algorithm then finds the optimal threshold value automatically and returns it as the first output argument. It is also necessary to pass the parameters below when calling the threshold function for Otsu thresholding.

cv2.threshold(src, thresh, maxVal, thresholdType), where:

- src: Input image.
- thresh: Arbitrary threshold value(this will be omitted because of the THRESH_OTSU flag).
- maxVal: Maximum intensity value of a pixel.
- thresholdType: This should be set to cv2.THRESH_BINARY+cv2.THRESH_OTSU to perform Otsu Thresholding.

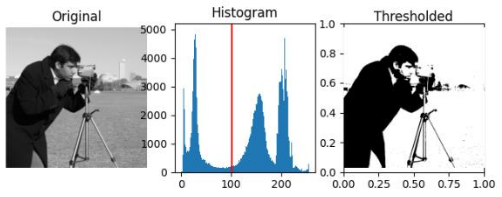


Fig 3. Otsu Thresholding example, with histogram.

VI. CONCLUSION

In conclusion, image binarization is a vital technique that significantly enhances the efficiency and effectiveness of image processing tasks. By transforming grayscale images into binary formats, this method simplifies the data while preserving critical information needed for further analysis. The simple binarization approach discussed in this article demonstrates how foundational techniques can still yield effective results in various applications, from document analysis to object recognition.

While more advanced methods exist, the simplicity and ease of implementation of basic binarization techniques make them valuable tools, especially in scenarios where computational resources are limited or rapid processing is required. This study emphasizes that even basic methods can provide substantial benefits when applied appropriately

Future research may explore the integration of simple binarization techniques with more sophisticated algorithms to enhance performance further. By combining the strengths of both approaches, we can develop more robust image processing systems that address a broader range of challenges. Ultimately, understanding and utilizing image binarization is essential for researchers and practitioners aiming to improve the accuracy and efficiency of image analysis in a variety of fields.

REFERENCES

[1] N. Otsu, "A threshold selection method from gray level histograms," IEEE Trans. Syst. Man Cybern. SMC-9, 62–66 (1979).

[2] W. Niblack, An Introduction to Image Processing, pp. 115–116, Prentice-Hall, Englewood Cliffs, NJ (1986)

[3] M. K. Yanni and E. Horne, "A new approach to dynamic thresholding," EUSIPCO'94: 9th European Conf. Sig. Process. 1, 34–44 (1994)

[4] Sezgin, Mehmet, and Bülent Sankur. "Survey over image thresholding techniques and quantitative performance evaluation." Journal of Electronic imaging 13.1 (2004): 146–165.

[5] “OpenCV: Image thresholding.” https://docs.opencv.org/4.5.2/d7/d4d/tutorial_py_thresholding.html