THRESHOLDING TECHNIQUES IN COMPUTER VISION APPLICATIONS

By

RIYAZ MOHAMMED M. * SABIBULLAH M. **

*-** Department of Computer Science, Jamal Mohamed College (Autonomous), Tiruchirappalli, Tamil Nadu, India.

https://doi.org/10.26634/jip.11.2.21001

Date Received: 16/07/2024 Date Revised: 22/07/2024 Date Accepted: 04/08/2024

ABSTRACT

Thresholding techniques are key pillars of image processing, especially for distinguishing objects in complex environments. This paper examines four types of thresholding strategies, each based on different theories, practical, popular, and advanced. Through a thorough literature review, the paper explains the thresholding techniques, thresholding operations, evaluation metrics, image processing techniques, and Python code for ROI of binary images in an understandable manner. The findings underscore the significance of thresholding in various applications, from object recognition to medical imaging, and highlight the importance of selecting appropriate thresholding methods based on image characteristics.

Keywords: Thresholding, Computer Vision, Image Segmentation, Binary Image, Threshold Selection, Python Code on ROI.

INTRODUCTION

In the field of computer vision, the ability to recognize objects of a desired class amidst a cluttered background is vital. Image segmentation, which involves dividing an image into parts and classifying them, is one of the most important steps in any computer vision task. Thresholding methods are discrete elements of this process. This paper presents a comprehensive assessment of thresholding techniques in computer imaging, from basic concepts to practical applications.

In the broad domain of computer vision, segmenting objects from images is one of the most critical tasks. This technique, essential to a wide variety of applications ranging from object recognition to medical image analysis, heavily depends on the effectiveness of segmentation strategies. Among these strategies, thresholding plays a pivotal role, offering a simple yet powerful method for distinguishing objects from the background based on their intensity levels. The importance of thresholding techniques in computer vision cannot be overstated. Essentially, thresholding defines a threshold value and categorizes pixels based on whether their intensity data falls below or above that threshold. This decision-making process is labor-intensive and serves as a precursor to the analysis and interpretation of imaging data.

The simplicity and effectiveness of thresholding techniques have led to their widespread use in various fields. Thresholding serves as a foundation for tasks such as document analysis, industrial inspections, and satellite imagery, enabling the extraction of meaningful information. However, the success of thresholding depends on certain conditions, including image quality, lighting conditions, and the nature of the objects and background.

Conventional thresholding methods, which are primarily based on global thresholding, provide a straightforward solution by applying a single threshold value uniformly across the entire image (Baird, 1992). This approach works

well when lighting is uniform and there is a clear contrast between the foreground and background, but it fails in cases of uneven lighting or complex backgrounds. Consequently, adaptive thresholding methods have gained attention, where the threshold value is dynamically adjusted according to local image conditions. Among the advancements in thresholding, Otsu's method stands out as an important technique, which maximizes the between-class variance of pixel intensities. This method is used because it can self-adjust the threshold without any prior knowledge of the image content.

Another approach to thresholding is histogram-based, which uses the histogram of pixel intensities to determine the threshold value. Histogram-based methods offer flexibility in handling a wide range of image properties and are effective in environments with varying lighting conditions by analyzing the distribution of pixel intensities.

1. Literature Survey

Early thresholding techniques, including global thresholding, involve selecting a single threshold value to partition an image into foreground and background areas. While effective in certain conditions, global thresholding may fail in the presence of non-uniform illumination or complex backgrounds (Gonzalez & Woods, 2008). To address this limitation, adaptive thresholding techniques dynamically adjust the threshold value based on local image characteristics. Otsu's method, a classic thresholding approach, automatically computes the most reliable threshold by maximizing the between-class variance of pixel intensities (Otsu, 1979). Similarly, histogram-based thresholding techniques utilize the histogram of pixel intensities to determine the most efficient threshold value, offering flexibility in handling various image properties (Sahoo et al., 1988).

Studies have seen the emergence of machine learning techniques, particularly deep learning models, taking over data thresholding roles. Convolutional Neural Networks (CNNs) directly learn the discriminative features of the data, enabling more precise and robust segmentation (He et al., 2017). Furthermore, multilevel thresholding techniques have been developed to capture complex scenes in images with multiple intensity levels, thus allowing for more accurate scene segmentation (Sezgin & Sankur, 2004).

The importance of thresholding in medical imaging cannot be overstated, as it facilitates tasks such as tumor segmentation, tissue segmentation, and other related imaging applications. For example, in Magnetic Resonance Imaging (MRI), thresholding is used to identify areas of interest and detect abnormalities (Styner et al., 2003). Similarly, in histopathological image analysis, thresholding methods assist in the segmentation of nuclei and the identification of pathological features (Veta et al., 2013).

In document analysis and optical character recognition (OCR), thresholding methods have been developed for text extraction and document segmentation. Text-background thresholding is particularly useful for separating text from background regions, thereby improving the accuracy of OCR techniques (Baird, 1992). Additionally, in industrial inspection and quality control, thresholding techniques are employed to detect defects and anomalies in manufactured products (Lu et al., 2018). Thresholding methods are also applied in remote sensing and satellite imagery analysis, as shown in Table 1. In land cover classification, thresholding is used to differentiate between land cover types based on spectral signatures (Jensen, 2005). Similarly, in urban area analysis, thresholding helps outline built-up areas, vegetation, and water bodies (Foody, 2002).

Despite their effectiveness, thresholding algorithms are not without challenges. These include sensitivity to noise, discrepancies in intensity maps, and disjointed intensity

Technique	Best Use Case	Limitations
Global Thresholding	Uniform lighting and simple images	Poor performance under variable light
Adaptive Thresholding	Non-uniform lighting conditions	More computationally intensive
Otsu's Method	Bimodal histograms	Not suitable for multi-modal images
Multi-level Thresholding	Images with multiple objects	Requires defining multiple thresholds

Table 1. Comparison of Thresholding Techniques

distributions, which can affect accuracy. Additionally, selecting the optimal threshold value is a critical issue that requires adjustment or the development of optimal parameters.

Thresholding techniques play a crucial role in a wide range of computer vision applications, including medical imaging, text analysis, and remote sensing. The integration of thresholding methods with advanced machine learning technologies continues to drive progress in the field of computer vision. Future research may focus on addressing real-world challenges and enhancing the robustness and scalability of thresholding methods.

2. Methodology

Implementing thresholding techniques in computer vision involves a systematic methodology that encompasses several key steps (Otsu, 1979).

2.1 Image Preprocessing

The image preprocessing step is a very essential one to be done prior to applying thresholding procedures. This is done with the objective of improving the quality of the input image and ensuring that the accuracy of the subsequent segmentation is optimal.

Contrast Enhancement: Increasing the contrast between the image and the background enhances the appearance of objects and details, making them easier to distinguish from the background. To achieve this, tools such as histogram equalization and adaptive histogram equalization can be used.

Histogram Equalization Formula: G(i,j) $= \frac{255}{M * N} \sum_{k=0}^{i} n_k$

Where,

G(i,j) G(i,j) is the equalized pixel value at location (i,j) (i, j) in the output image.

*M*M and *N*N are the dimensions of the image.

*nk*nk is the number of pixels with intensity *k*k in the input image.

Noise Reduction: In the event of title image transmittance, noise is introduced, leading to sound pollution. However, media filtering or blurring techniques, such as median filtering or Gaussian smoothing, can be applied to the image to visualize both parts of the string (Gonzalez & Woods, 2008).

Median Filtering Formula: G(i,j) = median (F(i+a, j+b))G(i,j)=median(F(i+a,j+b))

Where,

(F(i+a, j+b) F (i+a, j+b) represents the pixel values in a neighbourhood around pixel (i,j).

G(i,j) G(i,j) is the output pixel value after applying the median filter.

Intensity Normalization: Contrasting various hot spots throughout the picture might reduce the effectiveness of thresholding techniques. Intensity normalization algorithms are designed to adjust the pixel intensities across the image to ensure uniform illumination.

Histogram Stretching Formula:

$$G(i,j) = \frac{255}{I_{max} - I_{min}} * (F(i,j) - I_{min})$$

Where,

G(i,j) G(i,j) is the normalized pixel value at location *(i,j)*(i,j) in the output image.

*F(i,j)*F(i,j) is the original pixel value at location *(i,j)*(i,j) in the input image.

*I*max*I*max and *I*min*I*min are the maximum and minimum intensity values in the input image, respectively.

2.2 Threshold Selection

The choice of the thresholding method, the percentage selected, and the basis on which it is chosen all significantly impact the segmentation outcome. Selecting an appropriate thresholding technique involves analyzing various characteristics of the input image:

Intensity Distribution: The identification of pixel intensity in the image is a crucial part of the process that helps determine the optimal thresholding method to use. Histogram analysis is employed to visualize the intensity distribution and to distinguish between the peaks corresponding to the foreground and background regions.

Contrast and Noise Levels: Pictures with high contrast and low noise levels are best suited for global thresholding

methods, where a single threshold value is applied uniformly across the image. On the other hand, images with non-uniform illumination or complex contexts require adaptive thresholding techniques that adjust the threshold based on the content of the image.

2.3 Thresholding Operation

After determining the thresholding method and threshold value, the thresholding operation is applied to the preprocessed image to generate a binary mask, as shown in Figure 1 (Sezgin & Sankur, 2004). This binary mask separates the foreground from the background of the image, creating domains where subsequent analysis or processing tasks can be based (Gonzalez & Woods, 2008).

The thresholding operation consists of two stages: the transfer function stage and the histogram equalization stage.

Threshold Application: The selected threshold value is applied to each pixel in the preprocessed image, classifying pixels as either foreground or background based on their intensity values. Pixels with intensity values above the threshold are assigned to the foreground class, while those below the threshold are assigned to the background class.

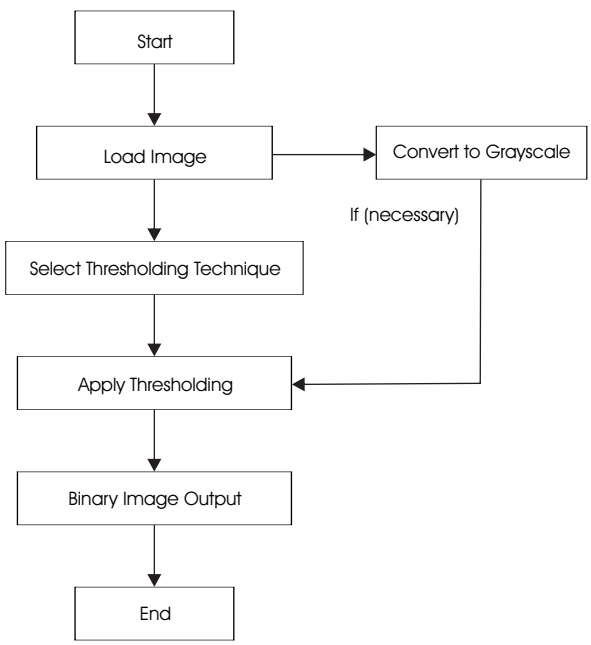

Figure 1. Flowchart for Thresholding Process

Binary Mask Generation: The thresholded image is transformed into a binary mask, where foreground pixels are represented as white (255) and background pixels as black (0). This binary mask serves as the basis for segmenting objects from the background and is used for further analysis or processing tasks.

Post-Processing: Depending on the application requirements, post-processing steps consisting of morphological operations can be applied to refine the segmentation results (Veta et al., 2013). Morphological operations, such as erosion, dilation, opening, and closing, help remove small noise artifacts, fill gaps in segmented objects, and smooth object boundaries, thereby enhancing the overall segmentation quality. A summary of thresholding methods is shown in Table 2.

2.4 Evaluation Metrics

Assessing the performance of thresholding strategies often involves quantitative assessment using metrics such as precision, recall, F1 score, and intersection over union (IoU) (Veta et al., 2013). These metrics provide quantifiable measures of segmentation accuracy and help in comparing different thresholding techniques under various conditions:

Precision: Precision measures the ratio of correctly segmented foreground pixels against all of those classified as foreground. That means it tells how accurate the positive predictions are.

$$Precision = \frac{\text{True Positives}}{\text{True Positives} + \text{FlasePositives}}$$

Recall: Recall is the ratio of efficiently segmented foreground pixels against all real foreground pixels inside the floor fact, being a measure for the completeness of predictions.

$$Recall = \frac{\text{True Positives}}{\text{True Positives} + \text{Flase Negative}}$$

Thresholding Method	Description
Global Thresholding	Selects a single threshold value for the entire image.
Adaptive Thresholding	Adjusts threshold value locally based on image content.
Otsu's Method	Automatically computes optimal threshold.
Histogram-based Thresholding	Utilizes intensity distribution for threshold selection.

Table 2. Summary of Thresholding Methods

F1-Score: The F1-score is the harmonic mean of precision and recall, providing a balanced measure of segmentation accuracy that considers both false positives and fake negatives.

$$F1 - score = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Intersection Over Union (IoU): IoU (He et al., 2017) quantifies the overlap of the segmented vicinity with the ground truth, hence providing a metric for segmentation consistency and spatial accuracy.

$$\text{IoU} = \frac{\text{Intersection}}{\text{Union}}$$

3. Thresholding Methods

Comparing thresholding strategies using these metrics for an overall assessment would facilitate a better evaluation of thresholding performance, allowing researchers to make informed decisions regarding the suitability of specific packages and image characteristics. This comprehensive methodology ensures the systematic application of thresholding methods in computer vision, from preprocessing an input image to objectively comparing the segmentation results. All conditions are meticulously designed for real-world images and will be applied with parameters that optimize the precision and robustness of the segmentation technique.

Methods for identifying regions of interest (ROI) from the background (Sahoo et al., 1988) assist in isolating the region of interest with greater accuracy, whether for further analysis or subsequent processing.

Thresholding is one of the simplest and most effective segmentation techniques, classifying pixels into two classes based on their intensity relative to a defined threshold value. The process for classifying pixels using thresholding is shown in Table 3.

Preprocessing Technique	Description
Contrast Enhancement	Improves visual contrast of the image
Noise Reduction	Removes or reduces noise from the image
Intensity Normalization	Adjusts pixel intensity values to a standard scale

Table 3. Common Image Preprocessing Techniques

- *Threshold Value Selection:* A threshold value is chosen such that pixels are divided into two categories: those above the threshold, called the region of interest (ROI) or forensic, and those below the threshold, called the background.
- *Convert the Image to Grayscale:* This can be done in the case of a colored image. This aids in simplifying the procedure, as only one intensity channel will be used.
- *Thresholding:* By comparing the chosen threshold value with the intensity value at each pixel, if a pixel's intensity is greater than or equal to the threshold, it is assigned a new intensity value, such as 255 for white, designating it as part of the ROI. Otherwise, it is assigned a different intensity, like 0 for black, representing the background.
- *Create Binary Image:* After thresholding, a binary image is produced in which pixels are classified into two categories with respect to the threshold value: foreground or background.

Here is an example of Python code that thresholds an image using the OpenCV library:

```
import cv2
Read the image
image=cv2.imread('input_image.jpg',cv2.IMREAD_GRAYSCALE)
Choose a threshold value
threshold_value = 127
Apply thresholding
_,binary_image=cv2.threshold(image,threshold_value,255,cv2.THRESH_BINARY)
Display the binary image
cv2.imshow('Binary Image',binary_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

This code snippet applies thresholding to a grayscale image using the cv2.threshold() function. The THRESH_BINARY flag indicates that all pixels with an intensity value above the threshold are set to 255 (white),

while those below are set to 0 (black). The binary image output is then displayed for visualization.

Clearly, the range of threshold values affects the classification of the pixels in the image. We may need to experiment with different threshold values to achieve the desired image segmentation.

In image processing and segmentation, pixels classified as part of the region of interest (ROI) due to thresholding are usually represented in white or another specified intensity value in the resulting binary image. A binary image produced this way is considered a mask, emphasizing the ROI by setting only the pixels above the threshold to a specified intensity value.

To easily identify and extract the ROI from the binary image, further processing steps such as contour detection, region labeling, or masking can be employed (Veta et al., 2013; Styner et al., 2003). These steps help identify the ROI pixels in a binary image.

- *Contour Detection:* Contour detection algorithms, such as OpenCV's cv2.findContours(), are implemented to identify connected components in the binary image. Contours represent the boundaries of regions with equal intensity values.
- *Filtering Contours:* Contours can be filtered based on area, shape, or other properties relevant to the ROI.
- *Contour Drawing or Mask Creation:* Contours can be either drawn on a blank image or used to create a mask by filling them with an intensity value.
- *Masking an Image:* The mask is then applied to extract the ROI from the original image by masking out the pixels that are not of interest.

Here is a Python code snippet that illustrates contour detection to enable the automatic identification and extraction of the ROI from a binary image.

```
import cv2
Read the binary image
binary_image = cv2.imread('binary_image.jpg',cv2.IMREAD_GRAYSCALE)
Find contours
contours,_ = cv2.findContours(binary_image,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
Filter contours based on area or other criteria
For simplicity, we will assume the largest contour represents the ROI
largest_contour = max(contours,key=cv2.contourArea)
Create a blank image for drawing contours
roi_mask = np.zeros_like(binary_image)
Draw the largest contour on the mask
cv2.drawContours(roi_mask,[largest_contour],-1,255,-1)
Apply the mask to extract the ROI from the original image
roi_image = cv2.bitwise_and(original_image,original_image, mask=roi_mask)
Display the extracted ROI
cv2.imshow('Region of Interest',roi_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In this piece of code:

- cv2.findContours() has been applied to the binary image to find the contours.
- The largest of the chosen contours corresponds to the ROI.
- *Mask Creation:* It means to trace out the contour selected in a blank image.

Basically, the original image is masked to obtain the ROI using a 'bitwise AND' operation. This process can be further tuned based on specific requirements and the nature of the ROIs in the images.

Contour detection with respect to an ROI in an image involves detecting the boundary or outline of that ROI. Contours are explained as curves formed by continuous points along the boundary of an object or region in an image.

In contour detection, the goal is to isolate and extract the contour representing the boundary of the region of interest. This stage is crucial in many image processing tasks, including object detection, shape recognition, and segmentation. The process can be outlined as:

- *Identify ROI:* The region of interest within the image is

marked where contour detection is intended.

- *Preprocess the image:* Certain features of the image may need adjustment before contour detection, such as converting the image to grayscale, applying thresholding, or smoothing.
- *Find Contours:* The actual contour detection is performed using OpenCV's findContours function.
- *Filter Contours:* The detected contours may need filtering and further refinement based on criteria like area, aspect ratio, or hierarchy.
- *Draw Contours:* The detected contours are then drawn on the original image to highlight the extracted boundary.

Here is a Python code example that shows a simple implementation of contour detection within a region of interest using OpenCV.

```
import cv2
Read the image
image = cv2.imread('image.jpg')
Define the ROI coordinates (x,y,width,height)
roi = (100,100,200,200)
Extract the ROI from the image
x,y,w,h = roi
roi_image = image[y:y+h,x:x+w]
Convert ROI image to grayscale
gray_roi = cv2.cvtColor(roi_image,cv2.COLOR_BGR2GRAY)
Apply thresholding or other preprocessing steps as needed
Find contours in the ROI
contours,_ = cv2.findContours(gray_roi,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
Draw contours on the ROI image
cv2.drawContours(roi_image,contours,-1,(0,255,0),2)
Display the ROI image with contours
cv2.imshow('ROI with Contours',roi_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Conclusion

Thresholding methods are fundamental tools in computer vision, enabling image segmentation and evaluation across various applications. Theoretically based on existing literature, this paper provides a critical survey of the theoretical fundamentals, practical applications, and recent advancements in thresholding methods. The evolution has progressed from global thresholding to advanced adaptive and machine learning-based strategies as studies continue to innovate to overcome challenges such as varying illumination conditions and complex backgrounds. This study highlights the key steps for implementing thresholding algorithms, aiding practitioners in their practical application. As time goes on, thresholding will remain a cornerstone in the development of robust and efficient computer vision systems. Insights on image processing techniques and their thresholding procedures are discussed.

References

[1]. Baird, H. S. (1992). Document image defect models. In *Structured Document Image Analysis* (pp. 546-556). Springer Berlin Heidelberg, Berlin, Heidelberg.

https://doi.org/10.1007/978-3-642-77281-8_26

[2]. Foody, G. M. (2002). Status of land cover classification accuracy assessment. *Remote Sensing of Environment*, 80(1), 185-201.

https://doi.org/10.1016/S0034-4257(01)00295-4

[3]. Gonzalez, R. C., & Woods, R. E. (2008). *Digital Image Processing*. Pearson Education.

[4]. He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2961-2969).

[5]. Jensen, J. R. (2005). *Introductory Digital Image Processing: A Remote Sensing Perspective*. Pearson Education.

[6]. Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1), 62-66.

https://doi.org/10.1109/TSMC.1979.4310076

[7]. Lu, R., Wu, A., Zhang, T., & Wang, Y. (2018). Review on

automated optical (visual) inspection and its applications in defect detection. *Acta Optica Sinica*, 38(8), 0815002.

http://doi.org/10.3788/AOS201838.0815002

[8]. Sahoo, P. K., Soltani, S. A. K. C., & Wong, A. K. (1988). A survey of thresholding techniques. *Computer Vision, Graphics, and Image Processing*, 41(2), 233-260.

https://doi.org/10.1016/0734-189X(88)90022-9

[9]. Sezgin, M., & Sankur, B. L. (2004). Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1), 146-168.

https://doi.org/10.1117/1.1631315

[10]. Styner, M., Gerig, G., Lieberman, J., Jones, D., & Weinberger, D. (2003). Statistical shape analysis of neuroanatomical structures based on medial models. *Medical Image Analysis*, 7(3), 207-220.

https://doi.org/10.1016/S1361-8415(02)00110-X

[11]. Veta, M., Van Diest, P. J., Kornegoor, R., Huisman, A., Viergever, M. A., & Pluim, J. P. (2013). Automatic nuclei segmentation in H&E stained breast cancer histopathology images. *PloS One*, 8(7), e70221.

https://doi.org/10.1371/journal.pone.0070221

ABOUT THE AUTHORS

M. Riyaz Mohammed is currently working as an Assistant Professor in the PG & Research Department of Computer Science at Jamal Mohamed College (Autonomous), Tiruchirappalli, Tamil Nadu, India. He has over 17 years of rich academic experience, specializing in Image Processing, Database Concepts, Operating Systems, and Python Programming, among other areas. He has presented research papers and articles, and has attended various National Level Technical Symposiums, Seminars, Workshops, Webinars, International Conferences, FDPs, SDPs, and more. He has also acted as a resource person, delivering special lectures on leading computer topics. He has participated in several Orientation Courses conducted by leading institutes and has acquired academic certificates from reputable agencies. As evidence of his academic caliber, he has prepared numerous study materials and video lectures on computer science for the benefit of the student community. Additionally, he has published a book titled "Programming in Python" and has supervised many innovative projects as part of PG programs.

Dr. M. Sabibullah is currently working as an Associate Professor in the PG & Research Department of Computer Science at Jamal Mohamed College (Autonomous), Trichy, Tamil Nadu, India. He has over 25 years of rich academic experience and more than 15 years of research experience, specializing in Machine Learning, Biomedical Big Data, Cloud Computing, and Health Care applications. He has published numerous research papers and articles, organized national-level technical symposiums, seminars, and workshops, attended international conferences, and delivered radio talks on All India Radio. He is currently guiding four Ph.D. research scholars and has successfully supervised more than five M.Phil. research scholars with notable academic outcomes. Dr. Sabibullah serves as a reviewer for many international conferences and has chaired numerous national-level conferences. He is particularly interested in advancing research in hot areas of the Computer Science domain, such as IoT, Big Data, Cloud Computing, Health Care Predictive Analytics, and Data Classification algorithms. Additionally, he is guiding both Ph.D. and M.Phil. research scholars in the field of Computer Science and is a life member of various professional bodies.