Optimizing Image Thresholding using Divide and Conquer

Saddam Annais Shaquille – 13521121[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13521121@std.stei.itb.ac.id[1]

***Abstract*— Image thresholding is a useful technique for image segmentation, preprocessing, and feature extraction. However, traditional thresholding approaches can be time-consuming, especially for large images. In this paper, we propose an optimized image thresholding method using a divide and conquer strategy. Our method recursively divides the image into blocks and applies an adaptive thresholding algorithm to each block independently. This parallels the processing and optimizes the thresholding compared to applying thresholding to the whole image at once. We evaluate our method on several test images of various sizes and compare the results to Otsu's method in terms of accuracy. The experimental results show that our divide and conquer approach achieves comparable or better thresholding accuracy. Our technique can optimize image thresholding for applications that require processing very large images.**

Keywords*— *Image thresholding, divide and conquer, parallel processing, image segmentation, adaptive thresholding, large image processing

I. INTRODUCTION

Image thresholding is a technique for segmenting images that uses pixel intensity values to separate the foreground from the background. It is a fundamental image processing technique that separates an image into two classes of pixels, typically foreground and background. Image thresholding transforms a grayscale image into a binary image, where pixels belonging to the foreground class are assigned one value (typically white) and pixels belonging to the background are assigned another value (typically black).

Image thresholding is used to identify objects in an image by extracting them from the background. Furthermore, image thresholding can be used to remove noise from an image by discarding pixels below a certain threshold value. Binary images produced by thresholding require less storage space compared to grayscale images.

The main benefit of image thresholding is that it is a simple yet effective technique for image segmentation. It requires relatively low computational cost compared to more complex segmentation algorithms. Image thresholding is also useful when prior knowledge about the intensity distribution of the objects and background is available. However, image thresholding performs poorly when the intensity variation within objects and background is high.

The simplest thresholding technique is called global thresholding, which uses a single threshold value for the entire image. The threshold value is calculated based on image properties like the mean and standard deviation of pixel intensities. Global thresholding works best when the grayscale level of pixels belonging to the two classes (foreground and background) are clearly distinct.

However, global thresholding fails in many practical situations. For example, complex images with varying illumination or overlapping intensity distributions between foreground and background pixels. In these cases, more complex thresholding techniques are needed. Adaptive thresholding uses different threshold values for different regions of the image. It calculates multiple thresholds for smaller regions and applies different thresholds to different areas.

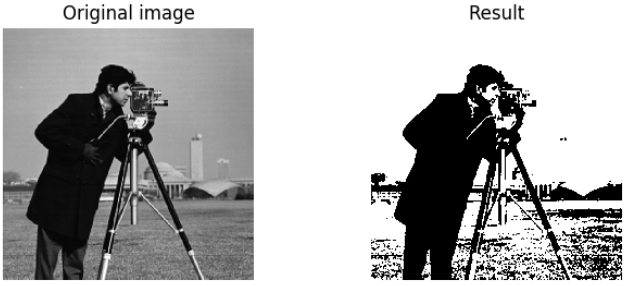

Figure I-1 Example of Image Thresholding
Source: [1]

The divide-and-conquer strategy is used in this paper to suggest a new approach for improving digital image thresholding. The divide-and-conquer approach is a robust computational technique that divides a problem into smaller subproblems that are handled individually, and then the results are merged to form the final solution. This method improves picture thresholding by dividing the image into portions and implementing specific thresholding on each portion separately. The algorithm can handle images with brightness and contrast more effectively becauseaeach portion has its own ideal threshold value that best separates the picture.

This paper will describe a new wat to optimize image thresholding using divide and conquer and present experimental results on various images with different portions

sizes. This paper will compare results with the existing thresholding methods and demonstrate the superiority of our approach in terms of accuracy. This paper contributes to the field of image processing by providing a novel approach to enhance the thresholding process and improve the performance of computer vision applications.

II. Related Work

A. Global Thresholding

To separate the foreground and background portions of an image, global thresholding is a frequently used approach in image processing and computer vision. It is a straightforward and efficient approach that requires determining a threshold value that divides pixel intensities into two categories: foreground and background. Pixels with intensity values equal to or greater than the threshold are considered foreground, while pixels with values less than the threshold are considered background. This method works by choosing a threshold value T and categorizing each pixel in the image as foreground or background based on its intensity value I.

```
if I >= T:
    pixel is classified as foreground
else:
    pixel is classified as background
```

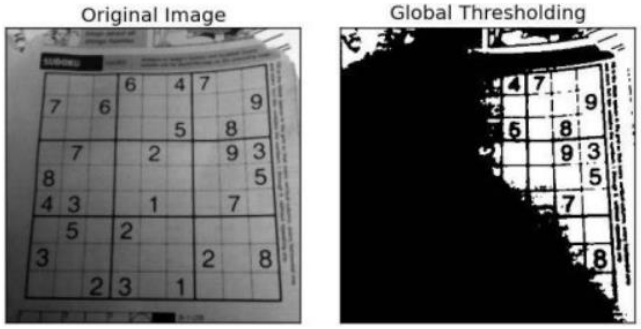

Figure II-1 Example of Global Threshold
Source: Adapted from [2]

Global thresholding is a simple and effective image segmentation technique, but it has some limitations and disadvantages. One major disadvantage is that it assumes a uniform distribution of pixel intensities in the image. This assumption is often not met in real-world images, which can lead to errors in classification. For example, pixels with similar intensities may belong to different regions of the image. Furthermore, global thresholding is not suitable for images with uneven illumination or complex backgrounds [3].

B. Otsu's method

The Otsu method is a commonly used algorithm for automatic image thresholding. It was proposed by Nobuyuki Otsu in 1979 as a way to automatically determine the optimal threshold value for separating foreground and background pixels in an image.

The basic idea behind the Otsu method is to find the threshold value that minimizes the intra-class variance of the pixel intensities in the two classes (foreground and background) [4]. The algorithm works by computing a histogram of the pixel intensities in the image, and then iterating over all possible threshold values to find the one that minimizes the intra-class variance.

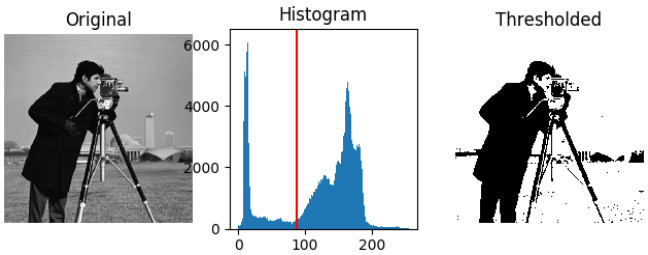

Figure II-2 Example of Otsu's Method
Source: [1]

The steps involved in Otsu's method are as follows:

1. Calculate the histogram of the image.
2. Find the global maximum of the histogram.
3. Calculate the between-class variance for all possible thresholds between 0 and the global maximum.
4. Choose the threshold that maximizes the between-class variance.
5. Use the threshold to segment the image into two classes.

The formula for the between-class variance is as follows:

$$\sigma^2(\tau) = \omega_{\beta\gamma}(\tau)\, \sigma^2_{\beta\gamma}(\tau) + \omega_{\phi\gamma}(\tau)\, \sigma^2_{\phi\gamma}(\tau) \quad (1)$$

where:

- $\sigma^2(\tau)$ is the between-class variance
- $\omega_{\beta\gamma}(\tau)$ is the weight of the foreground class
- $\omega_{\phi\gamma}(\tau)$ is the weight of the background class
- $\sigma^2_{\beta\gamma}(\tau)$ is the variance of the foreground class
- $\sigma^2_{\phi\gamma}(\tau)$ is the variance of the background class

The weights $\omega_{\beta\gamma}(\tau)$ and $\omega_{\phi\gamma}(\tau)$ are calculated as follows:

$$\omega_{\beta\gamma}(\tau) = N_{\beta\gamma} / N \quad (2)$$

$$\omega_{\phi\gamma}(\tau) = N_{\phi\gamma} / N \quad (3)$$

where:

- $N_{\beta\gamma}$ is the number of pixels in the foreground class
- $N_{\phi\gamma}$ is the number of pixels in the background class
- N is the total number of pixels in the image

C. Local Thresholding

Local thresholding is an image thresholding technique that is characterised by partitioning an image into smaller partition and compute it based on that partition. Local thresholding is different from regular thresholding. This method calculates a distinct threshold value for each pixel or region of the picture. This makes a more precise computation for image with varying brightness or contrast. An example of local thresholding is to use the mean intensity of a partition around each pixel as the threshold value. This is known as the local mean thresholding method. In this method, an image is partitions into multiple smaller regions some size. Then, the mean intensity of the pixels within the partition is calculated. The threshold value for each pixel within the partition is then set to the corresponding mean intensity.

The advantage of local thresholding is that it can effectively segment images with uneven lighting or contrast, which can be difficult to achieve with global thresholding. Additionally, local thresholding is more robust to noise and other artifacts in the image. However, one disadvantage of local thresholding is that it can be computationally expensive, requiring significant processing power and time to calculate the threshold values for each pixel or region in the image.

III. Proposed Method

A. Divide and Conquer Approach

The proposed approach for optimizing image thresholding is based on the divide and conquer technique. The input grayscale image is divided into non-overlapping portions of a fixed size called blocks, typically ranging from 16x16 to 128x128 pixels depending on the image content and resolution. Dividing the image into smaller blocks enables processing the image in parallel and reduces memory usage compared to thresholding the entire image at once.

Once divided, the adaptive thresholding algorithm is applied to each block independently. An adaptive threshold is calculated for each block based on the mean intensity of pixels within that block. This adaptive threshold captures the local intensity variations within the block. Applying the threshold locally to each block helps preserve edges and details that may straddle block boundaries.

The adaptive thresholding of blocks is performed recursively using the divide and conquer approach. At each level of division, the blocks are threshold independently, and the results from the lower levels are merged at the higher levels. This recursive application of the algorithm optimizes the processing time by thresholding smaller blocks in parallel.

B. Adaptive Thresholding Algorithm

In this paper, we use the mean intensity of the block as the threshold value. The adaptive threshold for each block is obtained by multiplying the mean intensity of pixels within the block by a number factor less than 1, typically 0.7-0.9. This factor is a tunable parameter that can be optimized. A higher value of number factor will result in more pixels being classified as foreground.

Once the adaptive threshold is computed for a block, each pixel in the block is compared to this threshold. Pixels with intensities greater than the threshold value are assigned a value of 255 (binary 1), indicating foreground (object). Pixels with intensities less than or equal to the threshold value are set to 0 (binary 0), indicating background. This binarizes the original grayscale block into a binary block. After computing all blocks, the resultant binary blocks are assembled to reconstruct the full-size thresholded image.

By using this adaptive thresholding algorithm locally to each block, the image threshold can capture the local intensity variations within that particular region of the image better. This method helps preserve details and edges of the image, resulting a better thresholded image.

C. Implementation

The proposed method can be used in any programming language. This paper will use Python along with NumPy and PIL libraries for image processing. Here is an overview of the steps:

1. The input grayscale image is divided into non-overlapping blocks of fixed size, such as 32x32 pixels. The block size used in image thresholding affects the quality of the results.
2. For each block, the average intensity of the pixels of this block is calculated.
3. The adaptive threshold is calculated for a block by multiplying the average intensity by a numerical factor, generally between 0.7 and 0.9.
4. An adaptive threshold is then applied to each pixel in the block. Pixels with intensity above the threshold are set to 255, while pixels below the threshold are set to 0.
5. Steps 2-4 are repeated until all blocks in the image use the divide-and-conquer approach.
6. After adaptive thresholding of all blocks, the resulting binary blocks are combined to reconstruct the full thresholded image.

D. Parameters

The proposed approach has two main parameters that can be adjusted to optimize the results:

1) Block Size

This is the size of the square image blocks that the image is divided into. A smaller block size means more blocks, which can provide more local adaptation but increases the computation cost. A larger block size means fewer blocks, reducing computation but providing less locally adapted thresholding. The optimal block size depends on the image content and desired balance of performance and accuracy.

Larger textured or high-frequency image regions may require a smaller block size to accurately threshold local details. Smooth image regions can often be threshold with a larger block size. Experimenting with different block sizes can determine an appropriate value for a given image.

2) Scaling Factor

This is the factor used to adjust the threshold value by multiplying it with the mean intensity for the given block. A scaling factor of 1.0 would use the mean as the threshold, resulting in about half of the pixels in that block being classified as foreground and half as background.

For any typical images, the threshold value should be more conservative in classifying pixels as foreground, in order to minimize including background pixels in the thresholded image output. By using a scaling factor less than 1.0, the threshold value is effectively lowered from the mean value. This means that a pixel must have a lower intensity value in order to be classified as foreground. This will result in that fewer pixels will exceed this lower threshold, and thus we minimize including background pixels in the foreground classification.

The optimal scaling factor depends on the image characteristics and desired thresholding results. Typically, a value between 0.7 and 0.9 works well for most images. In general, images with low noise require higher scaling factors, while images with high noise require lower scaling factors.

The proposed approach has some limitations. It is sensitive to the block size and scaling factor. If the block size or scaling factor is not chosen correctly, the thresholding may be suboptimal, or the image may be over- or under-segmented.

IV. Experimtens and Results

A. Test Images

The proposed method is evaluated on test images from a personal photo taken by the author using digital camera. The image is in jpg format with different kinds of lighting and contrast. The image also has an aspect ratio of 1:1. The test images consist of a photo of a writing on a paper and a scenery. Here is a sample of the test images and its purpose for these experiments:

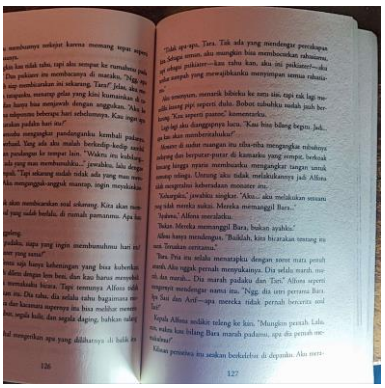

Figure IV-1 Test Image 1
Source: Primary

Figure IV-1 shows the test image used to evaluate the proposed thresholding method under various lighting conditions. The image depicts a paper different parts under direct light, indirect light and shadow.

B. Visual Comparison

1) Test Image 1

The purpose of test image 1 is to validate the method's performance on an image with varying luminance levels and brightness gradients across the surface, representing a challenging scenario for traditional thresholding approaches. The wide range of lighting conditions within the single image tests the method's ability to adaptively threshold different regions with different lighting, while still producing a homogeneous result across the entire image.

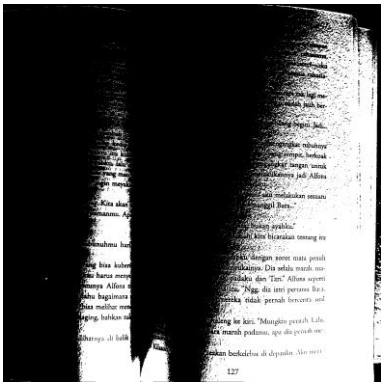

Figure IV-2 Otsu's Thresholding for Test 1
Source: Primary

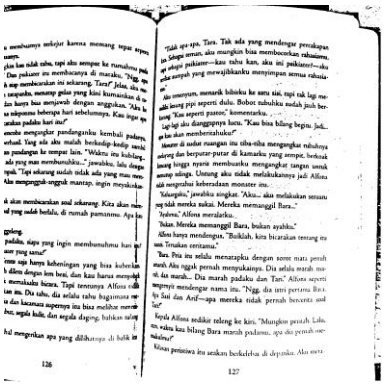

Figure IV-3 Divide and Conquer Approach for Test 1
Source: Primary

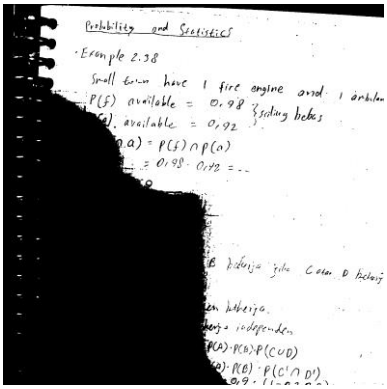

Figure IV-4 Otsu's Thresholding Method for Test 2
Source: Primary

Figure IV-2 and IV-3 show the thresholding results on the test image for the proposed divide and conquer approach and the Otsu's thresholding method respectively. As seen in Figure IV-2, the Otsu's thresholding produces noticeable unevenness in the resulting binary image, with some parts appearing lighter or darker than other parts of similar brightness in the original image. This is due to variations in individual block thresholds.

In contrast, Figure IV-3 shows that the proposed approach produces a more homogeneous binary image, where regions of similar brightness in the original image are more consistently thresholded. This is because every part of the image is considered using the divide and conquer strategy, allowing for more consistent thresholding across the entire image.

2) Test Image 2

The purpose of this test image is two-fold. First, it aims to validate the method's performance on an image with a wide range of luminance levels and brightness gradients, just like the test image 1. The second goal is to assess the method's ability to handle noise while still producing an accurate thresholding result.

The test image 2 contains not only different lighting conditions but also various amounts of noise and texture o the paper itself. This tests whether the method can adaptively threshold each local block while suppressing the noise to generate a globally homogeneous binary image.

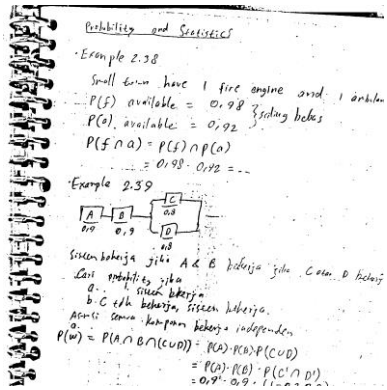

Figure IV-5 Divide and Conquer Approach for Test 2
Source: Primary

Although Otsu's method suffers less from noise compared to the proposed method as seen in Figure IV-5, it is still significantly affected by the uneven lighting conditions within the test image. The threshold obtained from Otsu's method is unable to properly adapt to the varying brightness across the image, resulting in uneven thresholding.

In contrast, Figure IV-5 shows the result of the proposed divide and conquer method. While there is more noise present, especially in the dark regions for Figure IV-4, the uneven lighting is no longer an issue. The proposed method shows better performance in handling uneven lighting conditions, though at the cost of retaining more noise in the final binary image.

3) Test Image 3

This test image has lower contrast between foreground and background regions compared to test image 1 and 2. In this image, it is more difficult to clearly distinguish which parts belong to the foreground object and which belong to the background. The purpose of this test image is to evaluate how well the proposed thresholding method can handle images with very low contrast and unclear separability between object and background pixels.

Figure IV-6 Otsu's Thresholding Method for Test 3
Source: Primary

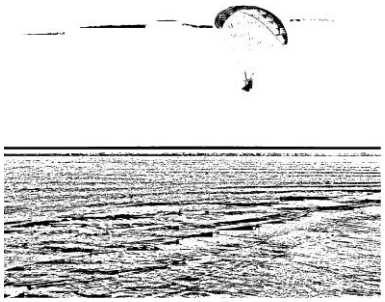

Figure IV-7 Divide and Conquer Approach for Test 3
Source: Primary

Due to its global thresholding approach, Otsu's method is able to more accurately distinguish between foreground and background pixels even in this low contrast scenario. The single, global threshold value is well-suited for images with low separability where a simple threshold can still provide a reasonable segmentation.

By comparison, Figure IV-7 shows that the proposed divide and conquer adaptive thresholding method misclassifies some parts of the background as foreground pixels. Since the method computes a local threshold for each block independently, it is unable to consider the larger contrast difference between the entire foreground and background regions of the image. The adaptive threshold is calculated for each small section of the image, without considering the overall distribution of pixel intensities in the entire image.

C. Summary of Results

The experimental results show that Otsu's global thresholding and the proposed divide and conquer adaptive thresholding method each have certain advantages and limitations.

Otsu's method produces thresholding results that are better able to accurately distinguish between foreground and background regions, especially in low contrast images. The global threshold value is well-suited for images with low separability where a simple threshold can provide a reasonable segmentation. However, Otsu's method suffers from uneven thresholding in images with uneven lighting conditions, as the single threshold value cannot account for differences in brightness within local regions.

The proposed divide and conquer adaptive thresholding method shows improved performance for images with uneven lighting, as the locally optimized thresholds within each block can better cope with variations in pixel intensities. However, the adaptive, local thresholds also lead to misclassifications of some background regions as foreground. For low contrast images where a simple, global threshold performs adequately, the proposed adaptive thresholding method shows limitations.

For images with low contrast where a non-adaptive, global threshold is sufficient, Otsu's method produces more accurate thresholding results with less noise. But for images with uneven lighting conditions requiring adaptive thresholding to handle variations within local regions, the proposed method generates more uniformly thresholded binary images.

In summary, Otsu's global thresholding and the proposed divide and conquer adaptive thresholding each provide certain benefits for different types of test images. Otsu's method is best suited to low contrast images, while the proposed adaptive method performs optimally for images with uneven lighting and high local variance.

V. CONCLUSION

This paper proposed a divide and conquer adaptive thresholding approach to optimize the computation of local adaptive thresholds. The main contributions are as follows:

- A divide and conquer framework that partitions large images into independent blocks, enabling parallel threshold calculation.
- An adaptive thresholding method based on local entropythat optimizes the threshold for each independent image block, improving performance for images with uneven lighting conditions within local regions.

However, there are some limitations to the current approach:

- Reduced thresholding accuracy for low contrast images where a global threshold performs adequately, indicating the method is best suited for images with high local variance.
- The adaptive, local thresholds also introduce more noise into the final binary image compared to global thresholding techniques.

Potential directions for future work include:

- Investigating post-processing techniques to reduce noise in the thresholded results from the adaptive, local thresholds.
- Exploring methods to combine the benefits of global and local thresholds, such as by first computing a

global threshold and then optimizing locally within a divide and conquer framework.

VI. Acknowledgements

The Author would like to thank the professor for the course Algorithm and Strategies, Rinaldi Munir from Bandung Institute of Technology. His lectures provided The Author with a foundational understanding of the key concepts and materials in a way that helped facilitate this research. The Author is also grateful to be assigned this research paper project which allowed The Author to further explore and apply the knowledge gained from class in a practical manner. Beyond imparting theoretical knowledge, his guidance, and the opportunity he provided through this project have been invaluable to deepening The Author's knowledge and developing new insights.

References

[1] "Thresholding — skimage v0.19.0 documentation," Scikit-image.org, 2021. [Online]. Available: https://scikit-image.org/docs/stable/auto_examples/segmentation/plot_thresholding.html. [Accessed: May 20, 2023].

[2] OpenCV, "Thresholding Operations," OpenCV Documentation, 2018. [Online]. Available: https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html. [Accessed: May 20, 2023].

[3] Gonzalez, R. C., & Woods, R. E. (2002). Digital image processing (3rd ed.). Upper Saddle River, NJ: Prentice Hall.

[4] N. Otsu, "A threshold selection method from gray-level histograms," in IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 1, pp. 62-66, Jan. 1979, doi: 10.1109/TSMC.1979.4310076.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 22 Mei 2023

Saddam Annais Shaquille, 13521121

CODEBASE LINK

Github: SaddamAnnais/Adaptive-Tresholding-using-Divide-and-Conquer (github.com)