# Metadata of the chapter that will be visualized in SpringerLink

| | |
|---|---|
| Book Title | Advances in IoT and Security with Computational Intelligence |
| Series Title | |
| Chapter Title | GCD Thresholding Function Applied on an Image with Global Thresholding |
| Copyright Year | 2023 |
| Copyright HolderName | The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. |


| | | |
|---|---|---|
| Corresponding Author | Family Name | **Manasi** |
| | Particle | |
| | Given Name | **Hussain Kaide Johar** |
| | Prefix | |
| | Suffix | |
| | Role | |
| | Division | |
| | Organization | Maulana Azad National Institute of Technology |
| | Address | Bhopal, India |
| | Email | hussainjmanasi@gmail.com |
| Author | Family Name | **Bharti** |
| | Particle | |
| | Given Name | **Jyoti** |
| | Prefix | |
| | Suffix | |
| | Role | |
| | Division | |
| | Organization | Maulana Azad National Institute of Technology |
| | Address | Bhopal, India |
| | Email | jyotibharti@manit.ac.in |
| | ORCID | http://orcid.org/0000-0003-0237-9029 |



Abstract

In this paper, we have developed and applied a new threshold function over an image globally and found the results to be quite promising. The method utilizes the feature of calculating the greatest common divisor (GCD) of the pixels within blocks formed in the image. The results obtained when compared with other standard thresholding techniques show us further insight with regard to the robust quality and performance of the newly devised thresholding function. In this paper, 3 progressive algorithms are presented with their particular challenges and shortcomings. Of these, the third algorithm is the main successful implementation of this thresholding technique.



Keywords (separated by '-'): GCD thresholding - Computer vision - Global thresholding - Digital image processing

# GCD Thresholding Function Applied on an Image with Global Thresholding

**Hussain Kaide Johar Manasi and Jyoti Bharti**

**Abstract** In this paper, we have developed and applied a new threshold function over an image globally and found the results to be quite promising. The method utilizes the feature of calculating the greatest common divisor (GCD) of the pixels within blocks formed in the image. The results obtained when compared with other standard thresholding techniques show us further insight with regard to the robust quality and performance of the newly devised thresholding function. In this paper, 3 progressive algorithms are presented with their particular challenges and shortcomings. Of these, the third algorithm is the main successful implementation of this thresholding technique.

**Keywords** GCD thresholding · Computer vision · Global thresholding · Digital image processing

## 1 Introduction

Up until now, many thresholding techniques have been devised and the main concept of thresholding has been preserved to this date and that is to separate the foreground of any image from its background. Also called segmentation it allows us to bring into focus the main objective of the image, and we can then go ahead and perform further descriptive analysis on the obtained part of the image. The obtained part of the image depending on the specific application can refer to written text, targets, defective materials, etc.

Otsu Thresholding has by far proven to be a reliable thresholding method albeit quite a computation-intensive process depending on the range in the image [10]. Others like the p-tile method, several entropic methods, and even before that we

H. K. J. Manasi (✉) · J. Bharti
Maulana Azad National Institute of Technology, Bhopal, India
e-mail: hussainjmanasi@gmail.com

J. Bharti
e-mail: jyotibharti@manit.ac.in

 
A. Mishra et al. (eds.), *Advances in IoT and Security with Computational Intelligence*, Lecture Notes in Networks and Systems 755,
https://doi.org/10.1007/978-981-99-5085-0_8



604726_1_En_8_Chapter ✓ TYPESET DISK LE ✓ CP Disp.:1/8/2023 Pages: xxx Layout: T1-Standard

had seen the initial development of fuzzy clustering-based methods [12] try to make the thresholding more optimized. Also devised were the histogram transformations which took into consideration more than just the isolated pixel, but the majority and minority proportionalities of the available pixels in the overall image [1–3, 13].

The next step normally pursued after selecting and employing an appropriate thresholding function is to perform segmentation on the image. Here, global thresholding is the simplest as applying a single standard across all pixels of the image is pretty straightforward as depicted in Eq. 1. Conversely, there are also adaptive thresholding techniques wherein a threshold is recalculated and applied separately for different parts of the image. This is useful when the image has a high amount of variance from one part of the image to another but within its disparate regions, the frequency is quite low. Every image performs differently for different algorithms, and there is no perfect algorithm that suits all images perfectly yet [4–9]:

$$\text{Pixel, } p = \begin{cases} 255, & p > T \\ 0, & p \leq T \end{cases} \text{ where, } T \text{ is Threshold value} \tag{1}$$

The obtained part of the image in question can be anything from the foreground, i.e. a person in an image like in the lena.bmp standard. Or in several other situations, users may want to perform edge detection which is possible by the Sobel, Prewitts, and Roberts operators. Contour-based line detection is also a well-in-demand problem statement achievable by Hough transforms and Hough lines. When it comes to whole objects, however, the 2 main techniques highlighted are region growing and region splitting and merging. More derivative segmentation methods that provide nuance to the obtained part as well are defined in the watershed method and gradient transform [11].

In lieu of this search for unique thresholding techniques, this paper aims to propose its own approach to thresholding and attempts to build upon a quite well-known concept in simple mathematics. The Greatest Common Divisor (GCD) is an effective function that gives good insight and understanding with regards to a couple or group of numbers, and utilizing this relationship of factors among pixel values poses an appropriate approach to building good threshold values that can target a wide range of images. The rest of the paper is structured to first deliver on the initial hindrances that are encountered when developing a novel thresholding technique in Sect. 2, after which all the possible implementations and their evolutions are explained in Sect. 3. Section 4 contains all the output tables and images, with brief descriptions of all of them. Observations and insights are made in Sect. 5 and finally, the paper is wrapped up in Sect. 6 with hints of future possible work on this topic.

## 2 Implementation Challenges

The steps applied to gain the threshold value of images were first done in a general fashion to calculate the GCD after which several modifications were performed to fix, enhance, and test the following parameters:

604726_1_En_8_Chapter ✓ TYPESET DISK LE ✓ CP Disp.:1/8/2023 Pages: xxx Layout: T1-Standard

- **Runtime of the algorithm**: At first, the runtime of the algorithm was observed to be acceptable for smaller images, particularly of size $256 \times 256$. But the developed algorithm did not perform well with respect to larger images in the range of $3068 \times 2457$ and such. For this reason, the usage of a larger block size within the image was proposed and implemented, but with that came the challenge of inflated and adulterated results. Additionally, the core problem of the algorithm having a high running time complexity did not seem to be tackled. However, fortunately, this problem was tackled using appropriate data structures and that solved other troubles fairly easily as well.
- **Robustness against block size**: It was imperative that the threshold value not vary with respect to block size as it may not be feasible to reiterate the image, again and again, using different thresholding block sizes. And even if it was possible, there seemed to be no appropriate method to decide the most correct block size for that particular image without active human involvement and selection. The varying results would also have been a fair consideration should the implementation have been similar to adaptive thresholding; however, that is not the case and such results are treated as undesirable when global thresholding is applied. This solution was achieved with a fairly simple implementation as we will see later in the paper.
- **Failure of the algorithm with extremely dark or bright images**: In the first few major constructions of the algorithms, it was observed that the algorithm performed significantly terribly with darker images which had a slightly less discernible foreground. This was later rectified by considering only the unique values in the image as at first glance it is easily identified that repeating values in an image can produce outliers to the calculation of the threshold value. However, for uniformly bright images, no working algorithm could be devised as the output received would be a completely black image.
- **Distinguishing factor**: Certain math applied in the upcoming pages can prove to look quite redundant, and the initial challenge of working on this was to find an algorithm that not only performs in a manner different from currently established standards but also delivers results that are equally promising. Whether this delivers convincingly innovative solutions is a decision best left to the reader.

Each of the above challenges was tackled in consequent iterations of the working program, and the solutions will be elaborated on in the following sections. However, there still does remain a certain problem that even standard thresholding techniques have failed to solve, and that is that this algorithm performs quite poorly with images that have a regularly high brightness throughout the image. That would be the recommended focus and target going forward if this approach of implementing GCD proves to truly be an effective approach.

Additionally, modern digital image processing has entered the age of computer vision. Incorporating this method of thresholding with some degree of convolutional neural network layer processing during the segmentation and object detection stage would be truly insightful as to the future usability of the algorithms.

# 3 Implementations

## *3.1 Algorithm 1*

The first approach was fairly simple with limited considerations taken, for example, whether the pixel value was even or odd, and whether the cumulative sum of the GCDs of the pixel pairs exceeded the mean value. This approach was admittedly naive as in almost all realistic scenarios the cumulative sum would almost always exceed the mean value of the pixels and hence, the threshold would be brought down to the mean value and the results obtained would be no different than the traditional average of the image. This approach was subject to the fourth challenge stated above, as the results were never particularly unique or useful in any way than if simple averaging were employed. In fact, certainly in certain situations, the simple averaging would have actually delivered better results. The steps of the algorithm are as below.

1. Calculate the mean of the pixels of the image and initialize the current running GCD threshold value ($currgcd$) of the image as the first pixel in the image.
2. Loop through the pixel values in the image and perform operations depending on which case is encountered
   (a) **Case 1**: The pixel value is even; then calculate GCD like normal between pixel value and $currgcd$ and store in $pgcd$.
   (b) **Case 2**: The pixel value is odd, then add 1 to pixel value and then calculate GCD like normal between pixel value and $currgcd$ and store in $pgcd$.
3. For each $pgcd$, add it to $currgcd$ and check if it is greater than the mean, and if it is then $currgcd$ is brought down to equal the mean value.

## *3.2 Algorithm 2*

The second approach is significantly more valuable for consideration. Here, a standard initial block size is selected as 4, and pixels are taken in blocks of 16 values. For 16 values, they are all first considered in pairs. That corresponds to $\frac{16*15}{2} = 120$ pairs. This calculation is important as it shows a degree of how long the algorithm has to run when considering block sizes.

$Block\_size = 3 \rightarrow 9 \rightarrow 36$ pairs
$Block\_size = 5 \rightarrow 25 \rightarrow 300$ pairs
$Block\_size = 6 \rightarrow 36 \rightarrow 630$ pairs

The pairs are increasing in a quite steep fashion, and this means the complexity of the algorithm increases proportionally too. However, the larger the block size the faster the entire image gets processed, so having a larger block size also corresponds to the image being processed quickly. Hence, one can safely devise Eq. 2:

$$Time\_Complexity(O) \propto \frac{Image\_size}{Block\_size} \tag{2}$$

Without further digressions, the steps performed in Algorithm 2:

1. First take all pixel values in that block and reshape them into a 1-D array of just normal values.
2. Take the GCD of all the pairs of values in the formed 1-D array.
3. Take the summation of the elements in the blocks and store them in another array that stores the sums of the different blocks.
4. Find the mean (%255) of all the elements in the list of sums of the blocks. That is the threshold value that will be applied.

The main concept while developing this algorithm was to harness the regional differences in the image while combining them at the same time. The summation of each block gives us an idea of the variations and GCDs of the pixels within that block, while averaging all the blocks gives the image a chance to balance out the regions of high brightness and deep darkness. We still, in this method, encounter a number of difficulties particularly.

1. **Higher thresholds**: This wouldn't normally be a problem if it actually reflected the characteristics of the image. The results obtained here are significantly more generous, as anything that isn't matching the gradient of a bright background gets caught in the threshold value and is labeled as foreground.
2. **High execution time**: This corresponds to the first challenge described earlier. Running time for larger images of sizes of up to $3068 \times 2457$ can take upwards of 15 min to completely execute. This is simply too high and had to be worked on as the time complexity of this method also is $O(n^2)$. AQ4
3. **Increasing block sizes failure**: An attempt to increase block size can on paper seem like a great option to speed up execution but, in reality, it didn't change execution time by much. Moreover, the acquired threshold values were simply too high to be appropriate. This corresponds to the second challenge described above. The thresholding function is sensitive to changing block sizes, and this is not a good phenomenon as this means certain block sizes would work better for different images, and this as of now has not yet been clearly discerned.

These challenges were however overcome in the next and final iteration of building this algorithm.

## *3.3 Algorithm 3*

Finally, this devised algorithm is the working success of this paper. The primary difference here is that the 1-D array holding all the pixel values gets substituted for a dictionary holding a count of all the pixel values appearing in the block. Initially, all the pixel values and their GCDs were considered but that had to be modified to consider the fact that the threshold values seemed to be stagnating at $\approx$127. Considering only unique values resulted in more dynamic threshold values being obtained due to lesser repeating outliers of GCD calculation like having multiple repeating



604726_1_En_8_Chapter ✓ TYPESET DISK LE ✓ CP Disp.:1/8/2023 Pages: xxx Layout: T1-Standard

255 pixels in a single image. Further, the block size that performed best was size 6. This contrast to the earlier selected value of 4 performing best can be explained by Eq. 3

$$Block\_size = 6 \rightarrow 36 \rightarrow 630\%255 = 120\ pairs \tag{3}$$

Regardless, the steps and algorithm for this process are elaborated below

1. Initialize a dictionary of 256 keys and initialize their values to 0.
2. Loop over each pixel for each block and increment value of corresponding keys in the dictionary.
3. Then iterate over the dictionary to calculate the GCDs of all the key pairs with non-zero values. For pixel values that repeat more than once, their GCDs among the other pixels are simply calculated by factoring in their dictionary values.
4. Do this for each block while accommodating the leftover pixels that wouldn't fit perfectly into a block.
5. Finally find the mean of the obtained resulting list of sums of all blocks.

**Algorithm 1:** Best results for GCD thresholding by below algorithm

```
Initialize: 1. A dictionary with 256 keys all with value 0; dict
2. An empty list that stores cumulative calculated values of each block; blockgcd
for i = h0, h0+1, ..., h0 + block_size do
    end
    for j = w0, w0+1, ..., w0 + block_size do
    end
    dict[img[i,j] += 1
for k1,v1 in dict.items() do
    end
    for k2, v2 in dict.items() do
    end
    if k1 == k2 or v1 == 0 or v2 == 0 then
    end
    exit()
GCDsum = GCDsum + GCD(k1,k2) + v1*v2
return GCDsum                                  ▷ Calculated threshold value
```

## 4 Results

The results obtained are first tabulated as a progression of threshold values obtained throughout the process of development by different algorithms in Table 1.

Table 2 then provides insights with respect to comparisons between our algorithm and results obtained from Otsu's thresholding.



604726_1_En_8_Chapter ✓ TYPESET DISK LE ✓ CP Disp.:1/8/2023 Pages: xxx Layout: T1-Standard

**Table 1** Progression of thresholds in our algorithms

| Image | Algorithm 1 | Algorithm 2 | Algorithm 3 |
|---|---|---|---|
| Lena | 72.0 | 84.4 | 100.2 |
| Shapes1 | 183.8 | 219.7 | 88 |
| Shapes2 | 216.5 | 149.0 | 21.2 |
| Scan1 | 88 | 166.6 | 126.1 |
| Scan2 | 58.8 | 58.4 | 105.0 |
| Dark | 26.3 | 36.8 | 53.8 |

**Table 2** Comparing our algorithm with that of Otsu's

| Image Name | Algorithm 3 | Otsu Method |
|---|---|---|
| Lena | 100.2 | 71.0 |
| Shapes1 | 88 | 173.0 |
| Shapes2 | 21.2 | 115.0 |
| Scan1 | 126.1 | 168.0 |
| Scan2 | 105.0 | 87.0 |
| Dark | 53.8 | 46.0 |


**Fig. 1** Shapes1 through Algorithm 1


**Fig. 2** Shapes1 through Algorithm 2

Figures 1, 2, and 5 are all different outputs of the same image consisting of different shapes with different outlines and colors. Figures 4 and 7 are results obtained from implementing our algorithms on medical scans. Standard image processing images like the lena.bmp are used in Figs. 3 and 6. Lastly, Fig. 5 is an example of the 3rd algorithm's performance on darker images (Fig. 8). AQ5


**Fig. 3** Lena through Algorithm 2

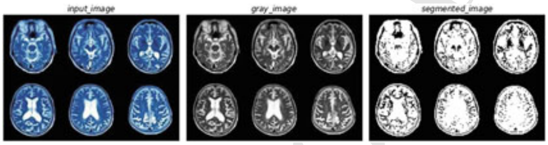


**Fig. 4** Medical scans through Algorithm 2

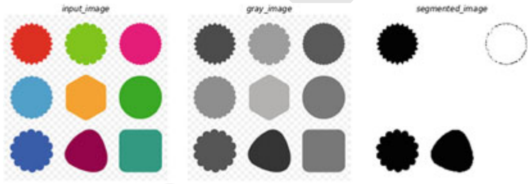


**Fig. 5** Shapes1 through Algorithm 3


**Fig. 6** Lena through Algorithm 3

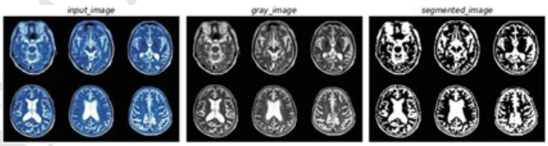


**Fig. 7** Medical scans through Algorithm 3


**Fig. 8** Dark images through Algorithm 3



## 5 Discussions

The purpose of discussing is not just the final observations but the process of developing the algorithm as well as to highlight the exact complications encountered when developing a thresholding function. Modifying the program bit by bit brings us to a conclusion that looks completely unrecognizable from the original intent and implementation. Regardless, an obvious observation from the finally devised algorithm is that it manages to perform quite well even in low lighting conditions but fails terribly when uniformly bright images are considered, and that too unexpectedly the problem seems to be that the threshold comes out too high for the regularly bright images with no clear background and foreground.

Hence, it is safe to conclude that GCD thresholding is a lower bound thresholding method. The value for the threshold will under very specific unrealistic circumstances reach a value higher than 130. This is because the GCD of any values at the 200 range would never have a factor greater than 130. The taken and considered GCD of any real image without changing the image itself can never be higher than 130. Hence, brighter images miss out on this thresholding method.

## 6 Conclusion

In this paper, we have developed an algorithm that accommodates the GCD of the pixel values in the image. We first begin from a naive understanding of the problem statement and then evolve toward a fully developed solution tackling each of the encountered problems one by one. The performance of the algorithm improves as well and is documented throughout development and is then compared with results from performing Otsu thresholding on the images. We find that it is a promising method of thresholding that can further be developed with reasonable interest, especially with the new establishment of computer vision and CNNs processing images and videos at extremely high speeds.

## References


1. Kaur N, Kaur R (2011) A review on various methods of image thresholding. Int J Comput Sci Eng 3(10):3441
2. Sankur B, Sezgin M (2001) Image thresholding techniques: a survey over categories. Pattern Recogn 34(2):1573–1607
3. Guruprasad P (2020) Overview of different thresholding methods in image processing. In: TEQIP sponsored 3rd national conference on ETACC
4. Sahoo P, Soltani S, Wong A, Chen Y (1988) A survey of thresholding techniques. Comput. Vision Graph Image Process 41(2)
5. Cuevas E et al (2009) A novel multi-threshold segmentation approach based on artificial immune system optimization. In: Advances in computational intelligence. Springer, Berlin, Heidelberg, pp 309–317
6. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), vol 1. IEEE
7. Singh TR et al (2012) A new local adaptive thresholding technique in binarization. arXiv:1201.5227
8. Dhanachandra N, Manglem K, Chanu YJ (2015) Image segmentation using K-means clustering algorithm and subtractive clustering algorithm. Procedia Comput Sci 54:764–771
9. Wang Q, Chi Z, Zhao R (2002) Image thresholding by maximizing the index of nonfuzziness of the 2-D grayscale histogram. Comput Vis Image Underst 85(2):100–116

10. Bangare S, Dubal A, Bangare P, Patil S (2015) Reviewing Otsu's method for image thresholding. Int J Appl Eng Res. https://doi.org/10.37622/IJAER/10.9.2015.21777-21783
11. Priyadharshini KS, Singh T (2012) Research and analysis on segmentation and thresholding techniques. Int J Eng Res Technol (IJERT) 1(10):1–8
12. Henila M, Chithra P (2020) Segmentation using fuzzy cluster-based thresholding method for apple fruit sorting. IET Image Proc 14(16):4178–4187
13. Tan KS, Isa NAM (2011) Color image segmentation using histogram thresholding-Fuzzy C-means hybrid approach. Pattern Recogn 44(1):1–15

# Author Queries

**Chapter 8**

| Query Refs. | Details Required | Author's response |
|---|---|---|
| AQ1 | Please check and confirm if the author names and initials are correct. | |
| AQ2 | Please be aware that your name and affiliation and if applicable those of your co-author(s) will be published as presented in this proof. If you want to make any changes, please correct the details now. Please note that after publication corrections won't be possible. Due to data protection we standardly publish professional email addresses, but not private ones, even if they have been provided in the manuscript and are visible in this proof. If you or your co-author(s) have a different preference regarding the publication of your mail address(s) please indicate this clearly. If no changes are required for your name(s) and affiliation(s) and if you agree with the handling of email addresses, please respond with 'Ok'. | |
| AQ3 | Kindly note the text mismatched between Manuscript pdf and Tex file, we have followed Tex source. Please check and confirm. | |
| AQ4 | Please correct "take upwards of". | |
| AQ5 | Please check and confirm if the inserted citation of Fig. 8 is correct. If not, please suggest an alternate citation. Please note that figures should be cited sequentially in the text. | |