ISSN 1213-6972 (print)
ISSN 1213-6964 (online)
Journal of WSCG
http://www.wscg.eu
Vol.33, No-1-2, 2025

A multi-level thresholding algorithm for threshold count and values identification based on dynamic programming

Eslam Hegazy
German University in Cairo
11835, Cairo, Egypt
eslam.sabry@guc.edu.eg

Mohamed Gabr
German University in Cairo
11835, Cairo, Egypt
mohamed.kamel-gabr@guc.edu.eg

ABSTRACT

Multilevel image thresholding is a simple and efficient segmentation technique. Thresholding criteria such as Otsu and Kapur objective functions are extensively used in the literature. They are effective techniques but suffer from poor computational complexity. Thus, methods such as dynamic programming for exact optimization or metaheuristic algorithms for approximate optimization are applied to improve runtime. However, most of these algorithms take the count of thresholds as input. Hence, a novel input-less algorithm that can identify the count and values of thresholds simultaneously is proposed. The proposed method is then compared to state of the art methods to assess its efficiency and effectiveness.

Keywords

Multilevel Thresholding, Clustering, Image Segmentation, Dynamic Programming, Minimum error thresholding, Exact Optimization, Unsupervised Classification

1 INTRODUCTION

Image segmentation is an important task of computer vision and image processing tasks. It is the task of partitioning a digital image into a set of objects, where each pixel of the image is labeled by one of the members of this set. Image segmentation is usually applied in various domains such as analysis of medical images [RKS+21], analysis of satellite images [RS21] and defect detection in industrial parts [MWSG21].

Various segmentation techniques with different compromises have evolved over the years since the dawn of computer science. One of these is image thresholding. Image thresholding works by segmenting the image pixels with respect to some threshold(s) value(s). All pixels with an intensity value less than a threshold belong to a class different from the class(es) of those pixels with intensity values greater than that threshold. The speed and simplicity of thresholding comes with an obvious limitation which is ignoring spatial information related to each pixel as it only considers the intensity value of the pixel. It is also limited when applied to an image with various lighting conditions throughout the image regions. Therefore, thresholding is used more commonly as a pre-processing step in image processing applications [ARM24]. Image thresholding can be classified into two main categories: bi-level thresholding and multi-level thresholding. Bi-level thresholding segments an image into two segments/classes; typically an object/foreground and a background. Multilevel thresholding segments an image to more than two segments.

Throughout years, different techniques for bi-level thresholding have been proposed and they were extended to multilevel thresholding as well. Most of these thresholding methods follow a common method of having an evaluation function that assigns a score to every candidate solution and then finds the best solution by searching for the one that gives the best score. Examples of such objective functions include Otsu's between-class variance [Ots79], minimum error thresholding [KI86], Kapur's entropy [KSW85], automatic thresholding criterion [YCC95], generalized histogram thresholding [Bar20], Tsallis entropy [Tsa88], Renyi entropy [SWY97] and fuzzy entropy [DRD+19]. Therefore, the multilevel thresholding problem is considered an optimization problem. However, in the case of significantly increasing the number of thresholds, the runtime grows exponentially, deeming it non practical [Tub14]. Recently, metaheuristic optimization algorithms have been widely applied to multilevel thresholding. Metaheuristic optimization algorithms are not guaranteed to find an optimal solution to the subject problem. However, they make a compromise between reducing the runtime and maintaining the solution optimality. Meaning that,

they can reach a suboptimal solution in a much faster time than an exact method that would reach an optimal solution but in a longer time [VTM23].

It has been an ongoing trend to experiment on combining metaheuristic optimization algorithm with different objective function in order to to reach a faster runtime while keeping a good segmentation quality. Variants of Whale Optimization algorithm (WOA) [SG19, ALH21, AIA21, MY22, ABMAA22], Corona Virus Optimization algorithm [AEMH23, HKHM23], Chimp Optimization Algorithm [EHR+23] and Particle Swarm Optimzation (PSO) [Aka13, SL17, AMR+19, GZB+20, NLZ24] are some commonly used optimization algorithms in multilevel thresholding.

Another direction of research focuses on exploiting the decomposability property of objective functions. This leads to the possibility of using the dynamic programming technique. Dynamic programming is an exact optimization technique, which means that it does not compromise the optimality of the solutions found to execute faster. Luessi *et al.* proposed a hybrid technique combining dynamic programming and Otsu's between class variance to solve the multilevel thresholding problem [LESK06, LESK09]. The same method has been revisited and a full proof of correctness has been provided by Merzban and Elbayoumi [ME19]. Kurita *et al.* proposed a dynamic programming approach that utilizes the minimum error thresholding criterion [KOA92]. Lei *et al.* utilized the same dynamic programming framework in conjunction with Masi entropy [LLWY24].

Image thresholding can be considered a special clustering problem. In clustering methods, determining the appropriate number of clusters or segments suitable for an input beforehand is not a straightforward problem [XQZ+16]. In clustering, one solution to this problem is testing different numbers of clusters and comparing them in order to determine which is the most suitable. However, applying that in multilevel image thresholding defies the purpose of using thresholding as a segmentation technique which is simplicity and speed. Therefore, efficient techniques for determining a suitable number of classes needed for an image before or during the thresholding process are crucial.

Hence, this paper proposes a novel method to find a suitable count of thresholds and to find the values of these thresholds simultaneously. This novel method composes a dynamic programming technique that compares different counts of thresholds, with a modified version of the minimum error threshold method. The minimum error threshold method was modified in order for the recurrence used by the dynamic programming to compare different counts of thresholds. This method proves efficiency compared to traditional methods that do not automatically find the count of thresholds, while giving better or similar results in terms of quality.

The remaining parts of this paper are structured as follows. Section 2 illustrates the proposed algorithm and the algorithms it is compared with, whereas Section 3 presents the experimental results and discussion. Section 4 concludes the paper and suggests possible directions for future research.

2 METHODS

In this section, the modified minimum error thresholding objective function is presented. Moreover, its reformulation as a recurrence relation in order to be used by dynamic programming is derived. Furthermore, three techniques to search for globally optimal thresholds for this objective function are illustrated. These techniques are exhaustive search, count-known dynamic programming and count-unknown dynamic programming.

2.1 Minimum Error Thresholding (MET)

The method of minimum error thresholding, proposed by Kittler and Illingworth [KI86], relies on the assumption that each class forms a normal distribution. Then, a candidate set of thresholds is evaluated based on the function r defined by Equation (1). The function r is to be minimized in order to find the best set of thresholds.

$$r(T_1,...,T_n) = 1 + 2 * \sum_{i=1}^{n+1} w_i * (\log \sigma_i - \log w_i) \quad (1)$$

where

$$w_i = \sum_{g=T_{i-1}}^{T_i-1} h(g) \quad (2)$$

$$\sigma_i^2 = \sum_{g=T_{i-1}}^{T_i-1} h(g) * (g-\mu_i)^2 \quad (3)$$

where

$$\mu_i = \frac{\sum_{g=T_{i-1}}^{T_i-1} h(g) * g}{w_i} \quad (4)$$

For n thresholds $(0 < T_1 < T_2 < ... < T_n < L)$, there are $n+1$ classes. We assume $T_0 = 0$ and $T_{n+1} = L = 256$ for a grayscale image where L is the maximum intensity level. h is the normalized histogram of the grayscale input image. It can be seen that minimizing the function r in Equation (1) is equivalent to minimizing the function f given by the equation:

$$f(T_1,...,T_n) = \sum_{i=1}^{n+1} w_i * (\log \sigma_i - \log w_i) \quad (5)$$

We can rewrite Equation (5) to be

$$f(T_1,...,T_n) = \sum_{i=1}^{n+1} Q(T_{i-1}, T_i - 1) \quad (6)$$

where Q is the class cost function, given by the equation:

$$Q(a,b) = \sum_{g=a}^{b} h(g) * (A(a,b,g) - B(a,b,g)) \quad (7)$$

$$A(a,b,g) = \log \sqrt{\sum_{g=a}^{b} h(g) * (g - \frac{\sum_{g=a}^{b} h(g) * g}{\sum_{g=a}^{b} h(g)})^2} \quad (8)$$

$$B(a,b,g) = \log \sum_{g=a}^{b} h(g) \quad (9)$$

2.2 Exhaustive Search Approach

Given a number of thresholds, n, the best n thresholds of an input image can be found by minimizing the function $r(T_1,...,T_i,T_n)$ given by Equation (1). The traditional method is to perform exhaustive search over all the possible values of T_i for every $1 \leq i \leq n$. However, the runtime of such an approach grows exponentially as n increases [Tub14].

2.3 Dynamic Programming count-known Approach

We show a multi-level thresholding approach based on MET method and dynamic programming where the count of thresholds is fed to the algorithm as an input. This algorithm is inspired by other techniques intertwining dynamic programming with Otsu's between class variance [LESK06, ME19] or Masi entropy [LLWY24]. Algorithm 1 combines dynamic programming with MET objective function to solve the multi-level thresholding problem. The algorithm takes as input the number of thresholds, $N\_Thresholds$, and outputs the best values of these thresholds that minimize the minimum error threshold (MET) criterion.

2.4 Dynamic Programming Approach for threshold count identification

Algorithm 2 is designed to operate without taking the number of thresholds, $N\_Thresholds$, as input. However, it finds that number and the values of these thresholds in the same process. This is achievable due to the fact that the values of the objective function of minimum error thresholding (MET) on different numbers of thresholds are comparable. The recurrence relation implemented by this algorithm is given by the equation:

$$J(i,j) = \min(Q(i,j), \min_{k=i+1}^{j} \{Q(i,k) + J(k,j)\}) \quad (10)$$

Algorithm 1 MET-DP-N Algorithm

```
dp[N_Thresholds, L] initialized with INF
next[N_Thresholds, L] initialized with −1
for t = 0 : L − 1 do
    dp[0,t] ← score(0,t)
end for
for i = 1 : N_Thresholds − 1 do
    for T_j = i + 1 : (L − 1) − N_Thresholds + i do
        for T_k = T_j − 1 down to i do
            if dp[i − 1][T_k] + score[T_k][T_j] < dp[i][T_j] then
                dp[i][T_j] ← dp[i − 1][T_k] + score[T_k][T_j]
                next[i][T_j] ← T_k
            end if
        end for
    end for
end for
T_n ← −1
Best ← INF
for i = 0 : L − 1 do
    if dp[N_Thresholds − 1][i] + score[i][L − 1] < Best then
        Best ← dp[N_Thresholds − 1][i] + score[i][L − 1]
        T_n ← i
    end if
end for
Thresholds ← []
t ← T_n
Thresholds.push(t)
for i = N_Thresholds − 2 down to 1 do
    t ← next[i][t]
    Thresholds.push(t)
end for
```

where $J(a,b)$ is the minimum value of the minimum error thresholding (MET) in the histogram subregion $[a,b]$ and $Q(a,b)$ is the error value of the histogram subregion $[a,b]$ and is given by Equation (7). Note that the intensity level k is considered twice in the expression $Q(i,k) + J(k,j)$ instead of just once as in Equations (1) and (6). That modifies the minimized function to be the one in Equation (11) instead of Equation (6). This results in each class being defined as the closed interval $[T_{i-1}, T_i]$ rather than the half-open interval $[T_{i-1}, T_i)$, thus including the upper bound T_i in the class. This adds a number of samples with intensity T_i to each of these classes. Hence, the weight, w_i, of each class will increase. Likewise, the standard deviation σ_i increases because occurrences of T_i are added to the class, introducing a new value not previously present. This has been observed to cause an increase in the expression $w_i * (\log \sigma_i - \log w_i)$. This increase reduces the tendency to divide the class, imposing a penalty that discourages some unnecessary splittings.

$$f(T_1,...,T_n) = \sum_{i=1}^{n+1} Q(T_{i-1}, T_i) \quad (11)$$

Algorithm 2 uses a 2D array *mem* of size $L \times L$, where $mem[i][j]$ stores the best (minimum) score possible for the histogram region from i to j. This best score may either correspond to treating the interval $[i,j]$ as a single class or to splitting it into multiple classes. Additionally, *next* is a 1D array of size L, where $next[i]$ stores a pointer to the next threshold location that, together with subsequent decisions, achieves the best score stored in $mem[i][j]$. *score* is assumed to be a 2D array of size

Algorithm 2 MET-DP Algorithm

```
1:  mem[L][L] initialized with INF
2:  next[L] initialized with −1
3:  for i = L−1 down to 0 do
4:      for j = i+1 : L−1 do
5:          mem[i, j] ← scores[i, j]
6:          next[i] ← j
7:          for k = i+1 : j-1 do
8:              if scores[i][k] + mem[k][j] < mem[i, j] then
9:                  mem[i, j] ← scores[i][k] + mem[k][j]
10:                 next[i] ← k
11:             end if
12:         end for
13:     end for
14: end for
15: t ← 0
16: Thresholds ← []
17: while t ≠ −1 do
18:     t ← next[t]
19:     Thresholds.push(t)
20: end while
```

$L \times L$ that contains the precomputed values of the class cost function $Q(i, j)$, defined by Eq. (7) for every $i < j$. The loop in lines 3–14 operates by iterating the starting index i from the end of the histogram at $L-1$ down to 0. For each i, the loop iterates over all possible ending indices j in the range $i+1$ to $L-1$. For each window $[i, j]$, the algorithm considers two possibilities: whether $[i, j]$ is better treated as a single class, or it is better divided into two classes $[i,k]$ and $[k, j]$ for some $i < k < j$. This is implemented by initially setting $mem[i][j]$ to $score[i][j]$ where $score[i][j]$ is the cost of treating the region $[i, j]$ as a single class. Next, the algorithm iterates over all possible split points k such that $i < k < j$, and evaluates whether splitting the region improves (decreases) the cost. In such case, $mem[i][j]$ and $next[i]$ are updated. Note that $mem[k][j]$ must have been computed in an earlier iteration since $k > i$ and the outer loop progresses with i decreasing from $L-1$ to 0. After all iterations are complete, the *next* array can be used to reconstruct the optimal thresholding: if a class starts at index u, it ends at index $v = next[u]$, and the process repeats from v onward.

3 EXPERIMENTAL RESULTS

This section describes the experimental setup followed by the evaluation metrics used for quality and efficiency. Then, results and their interpretations are presented. We compare our proposed method MET_DP with MET_DP_N and Otsu_DP.

3.1 Setup

The tested algorithms were implemented in Python 3. An i7-10750H CPU and 16GB of RAM on a Microsoft Windows 10 operating system were used. A group of 8 standard images was used for testing and visual analysis. These images are retrieved from Berkeley Segmentation Data Set BSDS500 [AMFM11] and USC SIPI Image database according to Tab. 1.

Image	Source	Name
1	BSDS500	test/2018.jpg
2	USC SIPI	4.1.05 (House)
3	BSDS500	test/223060
4	BSDS500	test/28083.jpg
5	USC SIPI	4.2.06 (Sailboat on lake)
6	BSDS500	test/106005.jpg
7	Standard	lighthouse.bmp
8	BSDS500	test/250047.jpg

Table 1: Sources of the used test images

3.2 Metrics

The segmentation quality is assessed using structural similarity index (SSIM), and peak signal-to-noise ratio (PSNR). An algorithm efficiency is evaluated by measuring the CPU time taken by it. For comparing between the quality of automatic detection of the count of thresholds, the SSIM and PSNR values are inspected. Moreover, visual inspection of the histogram and segmented images is performed.

3.3 Quality

The first two approaches, were run on counts of thresholds ranging from 1 to 5 inclusively. The third approach, as explained, doesn't take the thresholds count as an input, but finds the suitable count while it is running. Thus, we show the result of the third approach only on the corresponding found count of thresholds and 'N/A' is written for all other counts of thresholds.

Tables 2 to 5 summarize the results of running the three approaches on the eight test images. Table 2 shows the obtained values of thresholds for each of the three approaches, together with the objective function value obtained. As explained earlier, the first algorithm utilizes Otsu objective function while the other two utilize the MET objective function. Thus, the fitness values obtained by the first algorithm and the other two can't be compared together. Table 4 shows the obtained SSIM values.

Table 3 shows the obtained PSNR values. At image 1, the value obtained by MET_DP is the highest among the five results of MET_DP_N and higher than all of Otsu_DP_N. Moreover, in Otsu_DP_N, the highest PSNR value happens at the count of 4, which suggests that even with Otsu, which is a different criterion from MET, the count of thresholds of 4 turns out to give the best segmentation quality. At all other images (except 1 and 2), PSNR values of Otsu_DP_N increase with the increase of N as a higher number of thresholds allow for more segmentation levels, which results in less difference between the original and segmented image which makes PSNR generally higher. The same pattern can be observed with PSNR values of MET_DP_N results.

Figures 1 and 2 shows a visualization of these results for the 8 test images. It can be seen that the identified classes had various variances and weights. There were

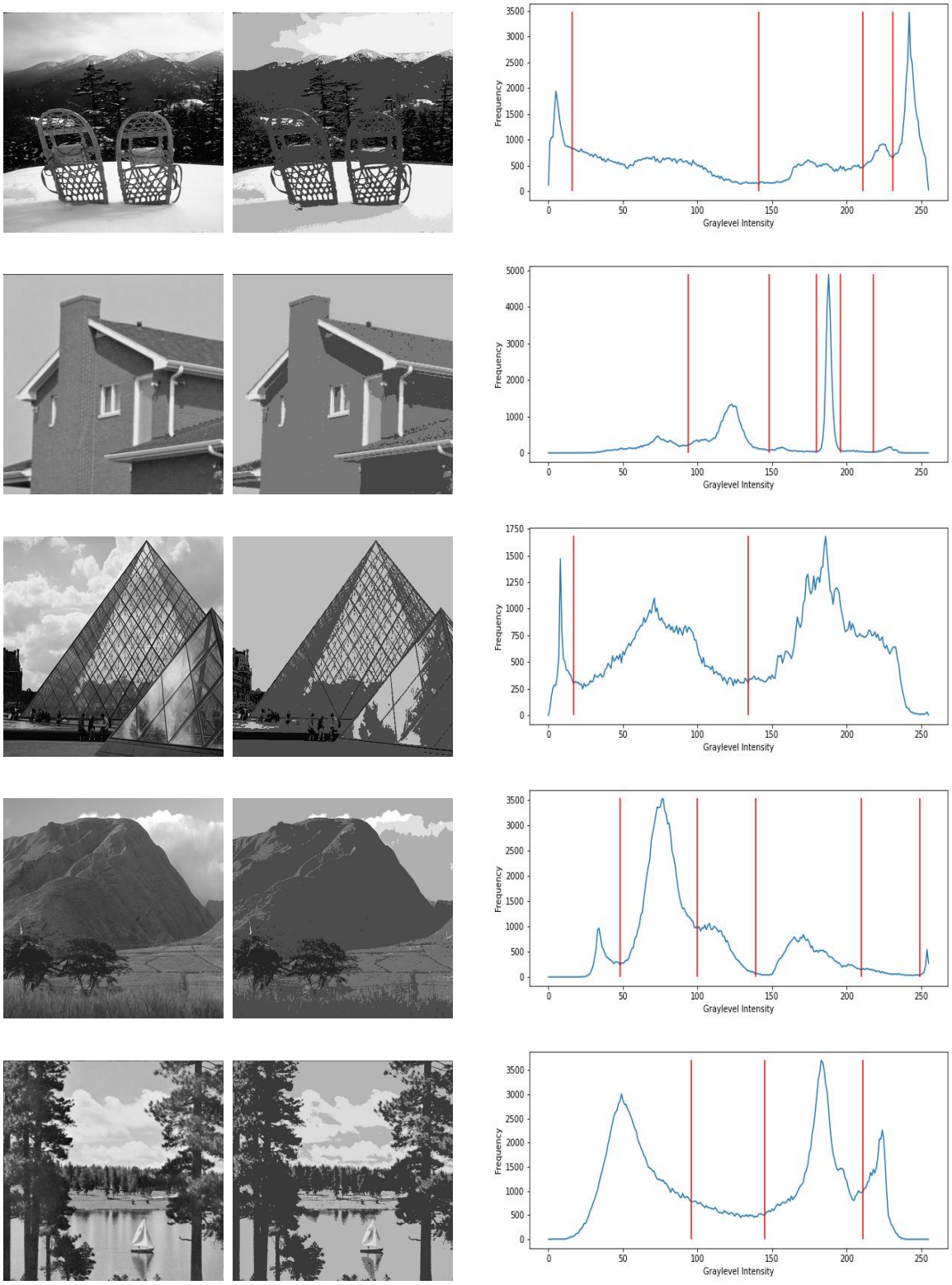

Figure 1: From top to bottom: the original image (on the left), the segmented image (on the middle) and its corresponding histogram (on the right) of test images 1, 2, 3, 4& 5. The threshold positions are highlighted by vertical red lines. The shown results are based on count and values of thresholds produced by the third approach

#	N	Otsu_DP_N	Fitness	MET_DP_N	Fitness	MET_DP	Fitness
1	1	134	6633.9316	141	6.0781	N/A	N/A
	2	58 155	7221.4748	141 231	5.9632	N/A	N/A
	3	58 150 242	7612.1279	16 141 231	5.9333	N/A	N/A
	4	51 129 204 242	7861.4903	16 141 211 231	5.9220	16 141 211 231	5.9220
	5	51 129 204 242 243	8075.5199	16 141 211 231 255	5.9220	N/A	N/A
2	1	148	1666.4725	167	5.2082	N/A	N/A
	2	97 156	1980.8985	181 196	4.6701	N/A	N/A
	3	97 152 188	2178.6707	181 196 218	4.6587	N/A	N/A
	4	97 152 188 189	2333.3539	94 148 180 196	4.6158	N/A	N/A
	5	97 152 187 188 189	2483.0315	94 148 180 196 218	4.6043	94 148 180 196 218	4.6043
3	1	126	3620.1978	134	5.8479	N/A	N/A
	2	71 146	3999.0115	17 134	5.8278	17 134	5.8278
	3	59 126 186	4233.7851	17 134 255	5.8278	N/A	N/A
	4	8 64 128 186	4391.7945	17 134 252 255	5.8293	N/A	N/A
	5	8 43 86 142 194	4496.7901	17 134 252 253 255	5.8304	N/A	N/A
4	1	133	1703.2676	139	5.1999	N/A	N/A
	2	87 145	1925.9237	139 249	5.1659	N/A	N/A
	3	87 137 186	2027.6683	143 210 249	5.1607	N/A	N/A
	4	66 94 140 190	2116.7351	41 143 210 249	5.1585	N/A	N/A
	5	66 94 140 186 254	2201.2226	48 100 139 210 249	5.1569	48 100 139 210 249	5.1569
5	1	125	3682.4636	125	5.6422	N/A	N/A
	2	83 153	3976.7739	89 134	5.6277	N/A	N/A
	3	71 134 187	4154.1965	89 145 211	5.6243	96 145 211	5.6244
	4	49 86 142 187	4262.0726	3 89 145 211	5.6247	N/A	N/A
	5	49 84 140 185 224	4357.7471	3 4 89 145 211	5.6250	N/A	N/A
6	1	143	2308.9727	63	5.8227	N/A	N/A
	2	86 162	2976.0846	70 151	5.7158	N/A	N/A
	3	83 144 203	3161.7949	70 151 251	5.6971	N/A	N/A
	4	83 144 203 255	3291.5089	70 151 175 251	5.6899	70 151 175 251	5.6899
	5	83 144 203 254 255	3398.6151	70 151 175 249 251	5.6959	N/A	N/A
7	1	133	2025.3460	186	5.6125	N/A	N/A
	2	99 166	2595.9818	103 167	5.6014	N/A	N/A
	3	67 106 169	2716.4402	27 99 167	5.5981	N/A	N/A
	4	63 103 158 207	2816.6491	9 27 99 167	5.5976	11 27 99 167	5.5977
	5	63 96 132 173 207	2876.3495	4 9 27 99 167	5.5978	N/A	N/A
8	1	96	2336.5958	96	5.4530	N/A	N/A
	2	41 109	2619.4860	25 88	5.4035	N/A	N/A
	3	36 90 152	2813.1837	25 88 249	5.3959	25 88 250	5.3960
	4	36 84 139 183	2891.7222	25 88 249 250	5.3969	N/A	N/A
	5	13 42 86 139 183	2958.1230	25 88 244 246 249	5.3977	N/A	N/A

Table 2: Output of the three approaches. The first column contains the test image number. The second column contains the number of thresholds used

classes with big weights (many pixels belonging to the class) and small variances such as the last class at histogram 1, the fourth class at histogram 2 and the first class at histogram 3. Moreover, there were classes with small weights and variances such as the fourth class at histogram 1, the last two classes at histogram 2, the last class at histogram 4 and the first two classes ath histogram 7. Regardless, most of these classes follow the bell-curve shape of a normal distribution, as by the core assumption of the MET method [KI86].

3.4 Efficiency

Table 5 shows the CPU time taken by each algorithm on each test image on each count of thresholds. It can be seen that the CPU time taken by MET_DP algorithm is longer than that taken by one run on one count of thresholds by either of the Otsu_DP_N or MET_DP_N methods. However and as previously stated in Section 2.4, the DP_MET method does not take the count of thresholds as input. Therefore, we assume that the correct count of thresholds for all of our test images is , for simplicity, between 1 and 5. Hence, the way to compare DP_MET with each of Otsu_DP_N and MET_DP_N is to compare the CPU time of DP_MET with the total CPU time of the five runs of Otsu_DP_N or MET_DP_N. Hence, we can see that MET_DP runs in much less time than the total runtime of the 5 runs of either Otsu_DP_N or MET_DP_N on each test image.

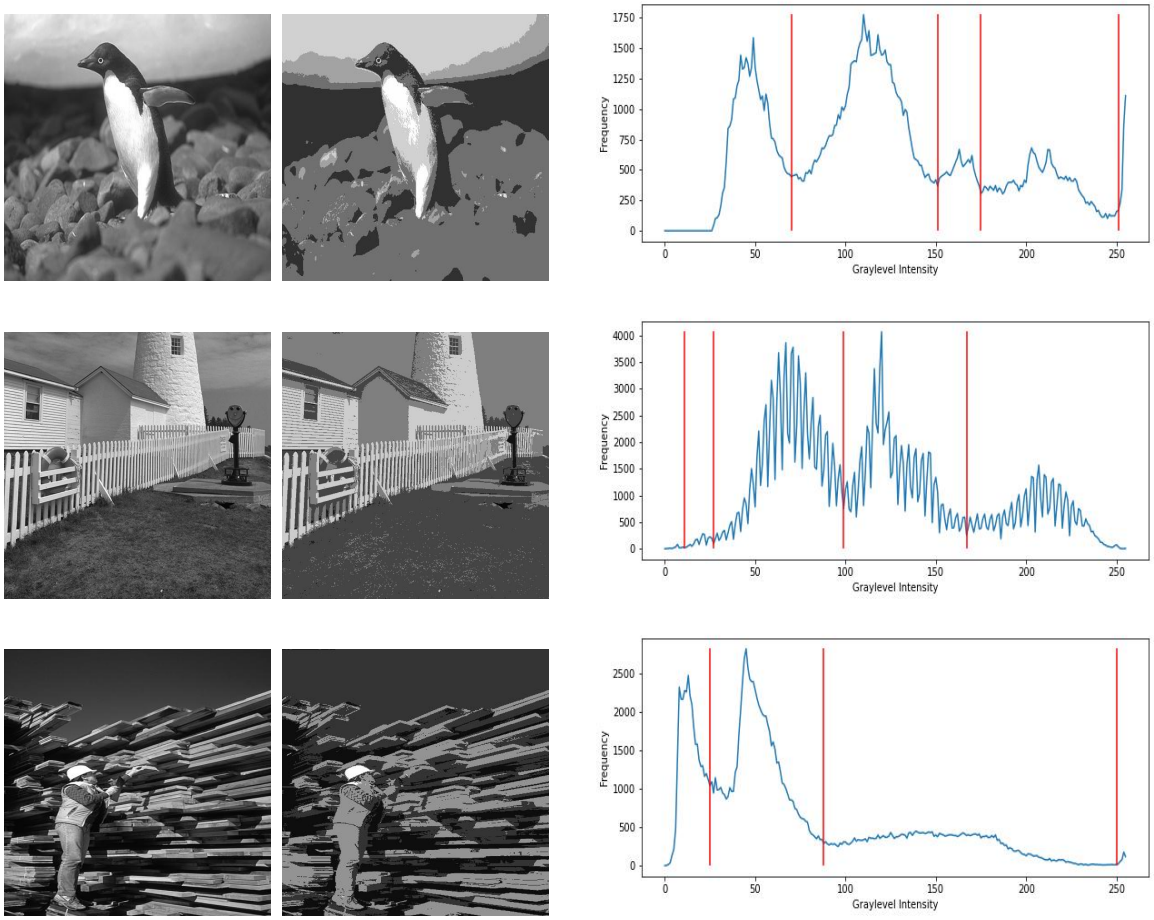

Figure 2: From top to bottom: the original image (on the left), the segmented image (on the middle) and its corresponding histogram (on the right) of test images 6, 7& 8. The threshold positions are highlighted by vertical red lines. The shown results are based on count and values of thresholds produced by the third approach

4 CONCLUSION

In this paper, a novel, exact and input-less multilevel thresholding method was proposed. The proposed method was compared against other recent and standard methods. It was able to determine the number of thresholds correctly. Moreover, it could reach values of these thresholds comparable to other methods. Additionally, comparing the runtime of the proposed method to others, it reaches the appropriate count and values of thresholds faster than the other methods. Furthermore, the proposed method is an exact and optimal method. Exploring the performance of thresholding criteria other than MET in conjunction with the proposed dynamic programming framework is one of our future works. Also, how the objective function of minimum error thresholding may be tuned to different types of images or problem domains is to be examined. In addition, extending the experimental analysis to a larger and more diverse image dataset is planned to better assess the robustness of the method. Finally, a thorough study to analyze where the proposed method achieves good results and where it does not is to be carried out.

5 REFERENCES

[ABMAA22] Mohamed Abdel-Basset, Reda Mohamed, Nabil M. AbdelAziz, and Mohamed Abouhawwash. Hwoa: A hybrid whale optimization algorithm with a novel local minima avoidance method for multi-level thresholding color image segmentation. *Expert Systems with Applications*, 190:116145, 2022.

[AEMH23] Yousef S. Alsahafi, Doaa S. Elshora, Ehab R. Mohamed, and Khalid M. Hosny. Multilevel threshold segmentation of skin lesions in color images using coronavirus optimization algorithm. *Diagnostics*, 13(1818):2958, January 2023.

[AIA21] J. Anitha, S. Immanuel Alex Pandian, and S. Akila Agnes. An efficient mul-

Image	N	Otsu_DP_N	MET_DP_N	MET_DP
1	1	17.514	17.488	N/A
	2	20.471	18.633	N/A
	3	21.653	20.457	N/A
	4	24.434	21.256	21.256
	5	24.432	21.256	N/A
2	1	21.475	21.129	N/A
	2	26.271	21.099	N/A
	3	26.618	21.140	N/A
	4	26.379	28.277	N/A
	5	26.160	28.494	28.494
3	1	18.894	18.863	N/A
	2	21.293	19.931	19.931
	3	23.667	19.931	N/A
	4	23.862	19.940	N/A
	5	25.707	19.940	N/A
4	1	21.244	21.222	N/A
	2	23.883	21.614	N/A
	3	25.430	22.016	N/A
	4	27.811	23.193	N/A
	5	28.107	27.311	27.311
5	1	20.266	20.266	N/A
	2	23.003	22.393	N/A
	3	25.142	24.792	24.465
	4	26.396	24.793	N/A
	5	26.748	24.793	N/A
6	1	17.952	15.386	N/A
	2	22.303	21.827	N/A
	3	24.637	22.460	N/A
	4	24.637	24.064	24.064
	5	24.887	24.118	N/A
7	1	18.830	17.436	N/A
	2	23.562	23.510	N/A
	3	25.131	23.909	N/A
	4	26.348	23.910	23.911
	5	28.225	23.910	N/A
8	1	20.025	20.025	N/A
	2	22.444	21.596	N/A
	3	25.266	21.939	21.931
	4	26.597	21.939	N/A
	5	26.974	21.978	N/A

Table 3: PSNR values

Image	N	Otsu_DP_N	MET_DP_N	MET_DP
1	1	0.615	0.606	N/A
	2	0.754	0.604	N/A
	3	0.729	0.705	N/A
	4	0.770	0.720	0.720
	5	0.769	0.720	N/A
2	1	0.733	0.719	N/A
	2	0.788	0.709	N/A
	3	0.745	0.711	N/A
	4	0.718	0.815	N/A
	5	0.736	0.817	0.817
3	1	0.600	0.586	N/A
	2	0.720	0.643	0.643
	3	0.744	0.643	N/A
	4	0.750	0.643	N/A
	5	0.805	0.643	N/A
4	1	0.590	0.583	N/A
	2	0.699	0.588	N/A
	3	0.699	0.586	N/A
	4	0.770	0.621	N/A
	5	0.767	0.758	0.758
5	1	0.627	0.627	N/A
	2	0.727	0.723	N/A
	3	0.730	0.736	0.726
	4	0.779	0.736	N/A
	5	0.772	0.736	N/A
6	1	0.622	0.628	N/A
	2	0.717	0.714	N/A
	3	0.739	0.713	N/A
	4	0.739	0.719	0.719
	5	0.736	0.720	N/A
7	1	0.504	0.524	N/A
	2	0.667	0.654	N/A
	3	0.789	0.679	N/A
	4	0.810	0.679	0.679
	5	0.834	0.679	N/A
8	1	0.668	0.668	N/A
	2	0.807	0.787	N/A
	3	0.867	0.789	0.789
	4	0.881	0.789	N/A
	5	0.887	0.790	N/A

Table 4: SSIM values

tilevel color image thresholding based on modified whale optimization algorithm. *Expert Systems with Applications*, 178:115003, 2021.

[Aka13] Bahriye Akay. A study on particle swarm optimization and artificial bee colony algorithms for multilevel thresholding. *Applied Soft Computing*, 13(6):3066–3091, June 2013.

[ALH21] Mohamed Abd Elaziz, Songfeng Lu, and Sibo He. A multi-leader whale optimization algorithm for global optimization and image segmentation. *Expert Systems with Applications*, 175:114841, 2021.

[AMFM11] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011.

[AMR+19] A. Ahilan, Gunasekaran Manogaran, C. Raja, Seifedine Kadry, S. N. Kumar, C. Agees Kumar, T. Jarin, Sujatha Kr-

Image	N	Otsu_DP_N	MET_DP_N	MET_DP
1	1	0.093	0.291	N/A
	2	0.386	0.582	N/A
	3	0.692	0.899	N/A
	4	1.017	1.242	2.799
	5	1.362	1.600	N/A
2	1	0.107	0.297	N/A
	2	0.374	0.526	N/A
	3	0.638	0.765	N/A
	4	0.943	1.099	N/A
	5	1.253	1.420	2.565
3	1	0.101	0.299	N/A
	2	0.369	0.592	2.662
	3	0.695	0.918	N/A
	4	1.037	1.255	N/A
	5	1.402	1.614	N/A
4	1	0.097	0.416	N/A
	2	0.393	0.818	N/A
	3	0.689	1.164	N/A
	4	1.008	1.521	N/A
	5	1.365	1.884	2.895
5	1	0.102	0.308	N/A
	2	0.520	0.675	N/A
	3	0.946	1.085	2.789
	4	1.413	1.622	N/A
	5	1.866	2.064	N/A
6	1	0.091	0.294	N/A
	2	0.361	0.585	N/A
	3	0.672	0.884	N/A
	4	0.992	1.222	2.709
	5	1.625	1.574	N/A
7	1	0.095	0.318	N/A
	2	0.475	0.713	N/A
	3	0.894	1.137	N/A
	4	1.318	1.570	2.691
	5	1.763	2.063	N/A
8	1	0.090	0.293	N/A
	2	0.358	0.567	N/A
	3	0.658	0.859	2.751
	4	0.986	1.204	N/A
	5	1.314	1.541	N/A

Table 5: CPU Time

ishnamoorthy, Priyan Malarvizhi Kumar, Gokulnath Chandra Babu, N. Senthil Murugan, and Parthasarathy. Segmentation by fractional order darwinian particle swarm optimization based multilevel thresholding and improved lossless prediction based compression algorithm for medical images. *IEEE Access*, 7:89570–89580, 2019.

[ARM24] Mohammad Amiriebrahimabadi, Zhina Rouhi, and Najme Mansouri. A comprehensive survey of multi-level thresholding segmentation methods for image processing. *Archives of Computational Methods in Engineering*, 31(6):3647–3697, August 2024.

[Bar20] Jonathan T. Barron. A generalization of otsu's method and minimum error thresholding. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 455–470, Cham, 2020. Springer International Publishing.

[DRD+19] Krishna Gopal Dhal, Swarnajit Ray, Arunita Das, Jorge Galvez, and Sanjoy Das. Fuzzy multi-level color satellite image segmentation using nature-inspired optimizers: A comparative study. *Journal of the Indian Society of Remote Sensing*, 47, 06 2019.

[EHR+23] Zubayer Kabir Eisham, Md. Monzurul Haque, Md. Samiur Rahman, Mirza Muntasir Nishat, Fahim Faisal, and Mohammad Rakibul Islam. Chimp optimization algorithm in multilevel image thresholding and image clustering. *Evolving Systems*, 14(4):605–648, August 2023.

[GZB+20] Qiyong Gong, Xin Zhao, Congyong Bi, Lei Chen, Xin Nie, Pengzhi Wang, Jun Zhan, Qian Li, and Wei Gao. Maximum entropy multi-threshold image segmentation based on improved particle swarm optimization. *Journal of Physics: Conference Series*, 1678(1):012098, nov 2020.

[HKHM23] Khalid M. Hosny, Asmaa M. Khalid, Hanaa M. Hamza, and Seyedali Mirjalili. Multilevel thresholding satellite image segmentation using chaotic coronavirus optimization algorithm with hybrid fitness function. *Neural Computing and Applications*, 35(1):855–886, January 2023.

[KI86] J. Kittler and J. Illingworth. Minimum error thresholding. *Pattern Recognition*, 19(1):41–47, 1986.

[KOA92] T. Kurita, N. Otsu, and N. Abdelmalek. Maximum likelihood thresholding based on population mixture models. *Pattern Recognition*, 25(10):1231–1240, October 1992.

[KSW85] J.N. Kapur, P.K. Sahoo, and A.K.C. Wong. A new method for gray-level picture thresholding using the entropy of

the histogram. *Computer Vision, Graphics, and Image Processing*, 29(3):273–285, 1985.

[LESK06] M. Luessi, M. Eichmann, G. M. Schuster, and Aggelos K. Katsaggelos. New results on efficient optimal multilevel image thresholding. In *2006 International Conference on Image Processing*, pages 773–776, 2006.

[LESK09] M. Luessi, M. Eichmann, G.M. Schuster, and A.K. Katsaggelos. Framework for efficient optimal multilevel image thresholding. *Journal of Electronic Imaging*, 18(1), 2009.

[LLWY24] Bo Lei, Jinming Li, Ningning Wang, and Haiyan Yu. An efficient adaptive masi entropy multilevel thresholding algorithm based on dynamic programming. *J. Vis. Comun. Image Represent.*, 98(C), may 2024.

[ME19] Mohamed H. Merzban and Mahmoud Elbayoumi. Efficient solution of otsu multilevel image thresholding: A comparative study. *Expert Systems with Applications*, 116:299–309, 2019.

[MWSG21] Sebastian Meister, Mahdieu A. M. Wermes, Jan Stuve, and Roger M. Groves. Review of image segmentation techniques for layup defect detection in the automated fiber placement process, May 2021.

[MY22] Guoyuan Ma and Xiaofeng Yue. An improved whale optimization algorithm based on multilevel threshold image segmentation using the otsu method. *Engineering Applications of Artificial Intelligence*, 113:104960, 2022.

[NLZ24] Fangyan Nie, Mengzhu Liu, and Pingfeng Zhang. Multilevel thresholding with divergence measure and improved particle swarm optimization algorithm for crack image segmentation. *Scientific Reports*, 14(1):7642, April 2024.

[Ots79] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[RKS+21] K.K.D. Ramesh, G. Kiran Kumar, K. Swapna, Debabrata Datta, and S. Suman Rajest. A review of medical image segmentation algorithms. *EAI Endorsed Transactions on Pervasive Health and Technology*, 7(27), 4 2021.

[RS21] Jarjish Rahaman and Mihir Sing. An efficient multilevel thresholding based satellite image segmentation approach using a new adaptive cuckoo search algorithm. *Expert Systems with Applications*, 174:114633, 2021.

[SG19] Basu Dev Shivahare and S.K. Gupta. Multilevel thresholding based image segmentation using whale optimization algorithm. *International Journal of Innovative Technology and Exploring Engineering*, 8(12):4602–4613, Oct 2019.

[SL17] Shilpa Suresh and Shyam Lal. Multilevel thresholding based on chaotic darwinian particle swarm optimization for segmentation of satellite images. *Applied Soft Computing*, 55:503–522, 2017.

[SWY97] Prasanna Sahoo, Carrye Wilkins, and Jerry Yeager. Threshold selection using renyi's entropy. *Pattern Recognition*, 30(1):71–84, 1997.

[Tsa88] Constantino Tsallis. Possible Generalization of Boltzmann-Gibbs Statistics. *J. Statist. Phys.*, 52:479–487, 1988.

[Tub14] M. Tuba. Multilevel image thresholding by nature-inspired algorithms - a short review. *Comput. Sci. J. Moldova*, November 2014.

[VTM23] Nguyen Van Thieu and Seyedali Mirjalili. Mealpy: An open-source library for latest meta-heuristic algorithms in python. *Journal of Systems Architecture*, 139:102871, June 2023.

[XQZ+16] Shuo Xu, Xiaodong Qiao, Lijun Zhu, Yunliang Zhang, Chunxiang Xue, and Lin Li. Reviews on determining the number of clusters. *Applied Mathematics & Information Sciences*, 10:1493–1512, 07 2016.

[YCC95] Jui-Cheng Yen, Fu-Juay Chang, and Shyang Chang. A new criterion for automatic multilevel thresholding. *IEEE Transactions on Image Processing*, 4(3):370–378, March 1995.