Optimization of Watershed Segmentation Method

[1] Ji-Ting Wu (巫季庭) [2] Jian-Jiun Ding (丁建均) [3] Chiou-Shann Fuh (傅楸善)
[1, 2] Graduate Institute of Communication Engineering,
[3]Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan
E-mail: r07942045@ntu.edu.tw jjding@ntu.edu.tw fuh@csie.ntu.edu.tw

ABSTRACT

Watershed is an algorithm that can perform segmentation according to the gradient of an image. However, the watershed algorithm requires much computation time and always suffers from the over-segmentation problem. In this paper, we try to improve the watershed algorithm and solve these problems. First, a more flexible way to initialize and adjust the threshold is proposed. Second, instead of comparing the four adjacent pixels each time, which is time consuming and requires lots of iterations, a more efficient way to label the pixels with the same quantization level is proposed. With these techniques, the over-segmentation problem can be solved and the computation time is reduced, which make the watershed algorithm more practical in image processing applications.

Keywords: *watershed, segmentation*

1. INTRODUCTION

Watershed is segmentation algorithm which segments image based on gradient of image. First, we should get horizontal and vertical gradient of image. Then, we get the horizontal and vertical gradients:

$$g = \sqrt{g_x^{\,2} + g_y^{\,2}}\,. \tag{1}$$

Then, the gradient is quantized with a quantization factor of Q by (2):

$$\text{gradmag} = \text{round}\left(\frac{g}{Q}\right). \tag{2}$$

Then, we set original threshold to zero, and merge adjacent pixels as long as their gradients are lower than original threshold in Fig. 1.

Even though in original method, we must set original method to zero, however, there are no zero adjacent to another zero, so it is difficult to observe merging process.

Accordingly, let us give another example of setting original threshold. If we choose 3 as the original threshold, we would get result in Fig. 2. The adjacent pixels whose quantized gradient is lower and equal to 3 is merged into the same region.

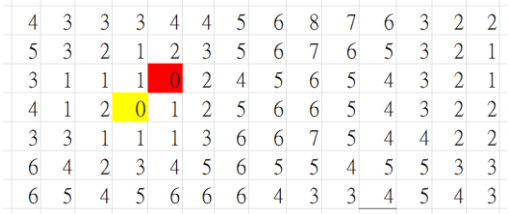

Fig. 1: The original threshold is 0. The numbers in the entries are quantized gradients. Adjacent pixels lower than or equal to 0 are merged into the same region.

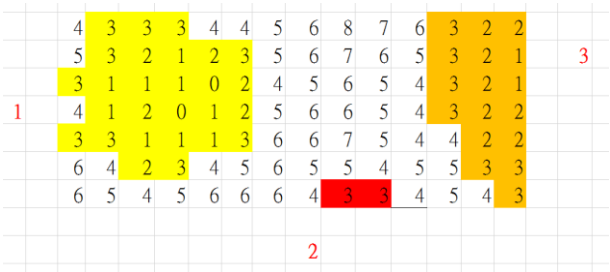

Fig. 2: The original threshold is 3. The numbers in the entries are quantized gradients. Adjacent pixels lower than or equal 3 are merged into the same region.

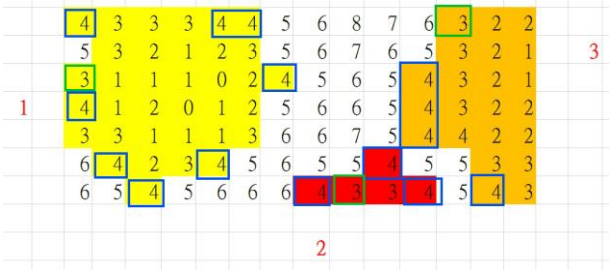

Fig. 3: Add one to threshold, merge pixels with adjacent region as long as their values equal to new threshold.

Finally, in the original watershed algorithm, we add one to the original threshold. We merge every pixel into adjacent region if its quantized gradient equals new threshold, as the pixels circled by blue squares in Fig. 3.

1	1	1	1	1	1	0	0	0	0	0	1	1	1
0	1	1	1	1	1	0	0	0	0	0	1	1	1
1	1	1	1	1	1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0	0	1	1	1	1
0	1	1	1	1	0	0	0	0	1	0	0	1	1
0	0	1	0	0	0	0	1	1	1	1	0	1	1

Fig. 4: Set a pixel to 1 if its value is lower than or equal to threshold. Otherwise, the pixel is set to 0.

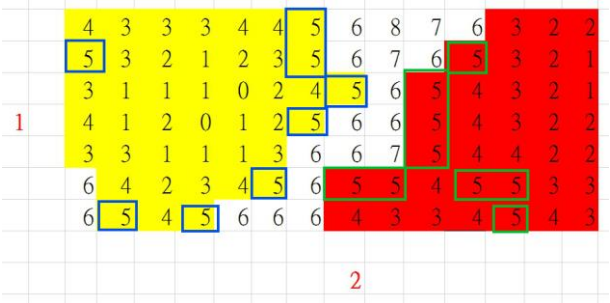

Fig. 5: When threshold is set to 5, the second and the third region in Fig. 3 are merged into the second region in Fig. 5.

However, in this original algorithm, we should compare 4 adjacent pixels with pixel whose value equals threshold, even though only one region near some of pixels, as the pixels circled with blue squares in Fig. 3. We could decrease time for calculating, if we merge them together directly.

Another problem is that we must spend much time to iterate in order to make sure all pixels merged with adjacent region. In the original algorithm, the original threshold we must choose zero. It would cause over-segmentation problem which means divided into too many regions.

Then, because it increases the complexity, we need to use more time to execute.

Owing to over-segmentation problem, amounts of regions became a large number of data, so when we use original algorithm to help compression, efficiency of compression would not be good.

Accordingly, in our proposed method, we would correct the fifth step of original watershed algorithm.

2. PROPOSED METHOD

2.1. Check Merged Pixel with "<=" Instead of "="

After adding one to the threshold in the final step of original algorithm, we just need to turn every pixel into 1, if its value is lower than or equal to threshold, otherwise into 0 in Fig. 4.

Then, merge the adjacent pixels into the same region as long as their value is 1. Consequently, we can get the same result in Fig. 3.

Fig. 6: Only one region next to the green pixel circled with blue pen.

Fig. 7: The green pixel circled with blue pen in Fig. 6 is combined with region next to it.

However, if we always use "<=" in watershed algorithm, some of region would be merged together.

In this example, when threshold is set to 5, the second region and the third region in Fig. 3 are merged together in Fig. 5.

2.2. Record Representative Pixel of Region

Consequently, before adjusting threshold, we have to record one pixel's place in every region in Fig. 3 to represent these regions, place of these 3 pixels circled with green pen in Fig. 3 need to be recorded, and check whether these pixels should be merged into the same region after threshold is added one.

If a region is not merged with another region after threshold added one (we could check whether its representative pixel and representative pixel of another region are merged into the same region after threshold added one.), just like the blue pixels in Fig.5, we do not have to deal with them temporarily.

2.3. Merge with Adjacent Region

Otherwise, just like the green pixels in Fig. 5, we have to check whether these pixels belong to which region, we merge a pixel with adjacent region if only one region is next to it, just like the green pixel circled with blue pen in Fig. 6.

After merging this pixel, the result is Fig. 7. We continue scanning (from top to bottom, and from left to right) green pixel in Fig. 7, and check continually whether these pixels belong to which region, if we see Figs. 6-10, we could understand this step more easily.

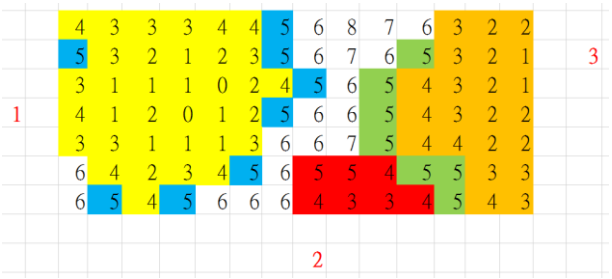

Fig. 8: The green pixel in Fig. 7 is combined with region next to it.

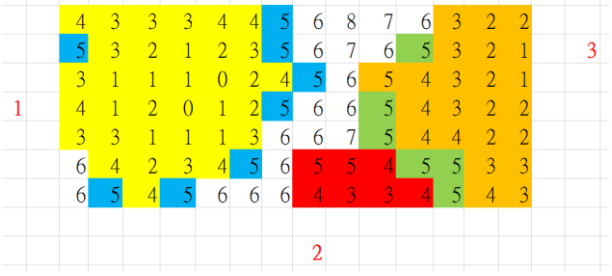

Fig. 9: The green pixel in Fig. 8 is combined with region next to it.

Fig. 10: The green pixel in Fig. 9 is combined with region next to it.

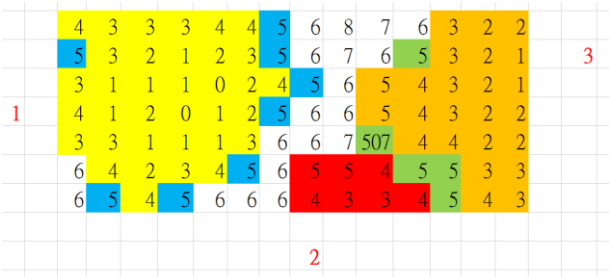

Fig. 11: The value of green pixel circled with blue pen in Fig. 10 is set to 512-threshold.

2.4. Set Watershed

If the green pixel is near over one region, like the green pixel circled with blue pen in Fig. 10, we let it be watershed, and turn its value into 512-threshold in order to make it like a wall, when threshold keeps increasing, the wall would separate region next to it. We could see Fig. 10. Another advantage is if we want to recover original quantized gradient, we just use 512 to subtract value which is over 256.

After setting watershed, the result is Fig. 12.

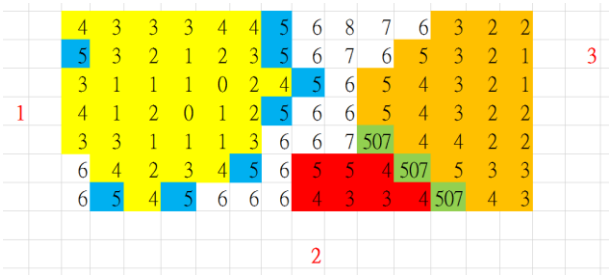

Fig. 12: After setting watershed when threshold is 5.

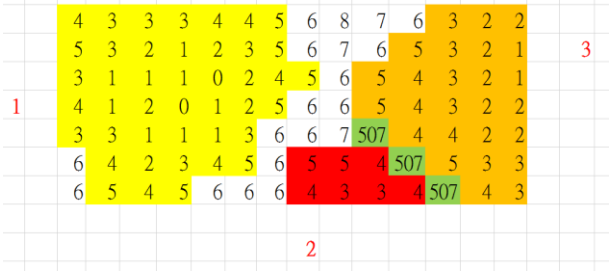

Fig. 13: Result after correcting with watershed.

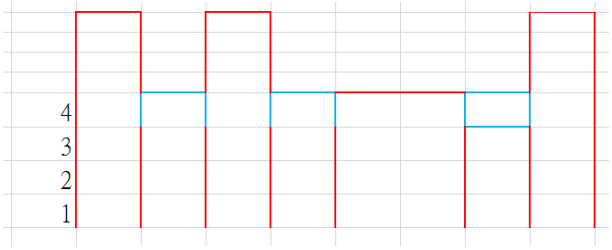

Fig. 14: Original water level is 4.

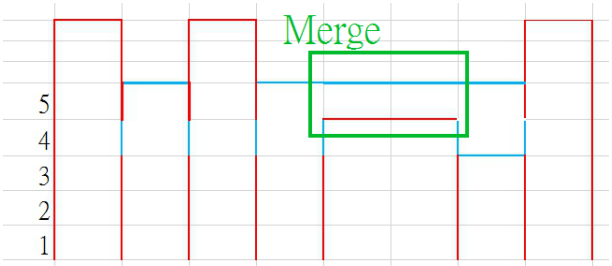

Fig. 15: Merging problem happen when water level becomes 5.

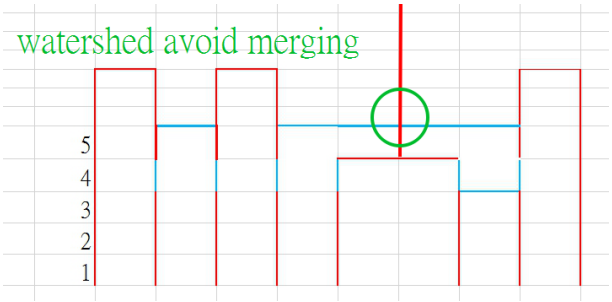

Fig. 16: After setting watershed, increase water level to 5 again, and different regions would not be merged thanks to watershed.

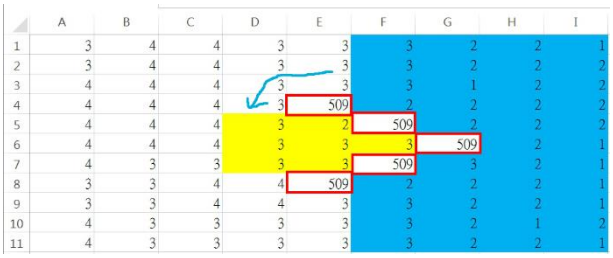

Fig. 17: Yellow region and blue region would still be merged together from the top left.

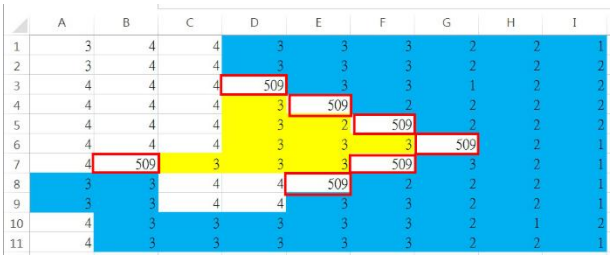

Fig. 18: Scan Fig. 17 from bottom to top, from right to left again.

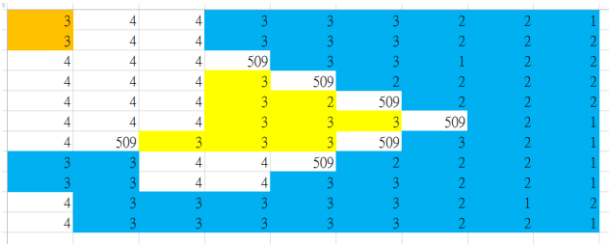

Fig. 19: Different regions are kept from merging.

2.5. Merge Pixel Smaller Than Threshold again after Setting Watershed

After setting watershed, we could use the same method in Section 2.1 again, pixels with lower than or equal to threshold need to be set to one, otherwise set to zero. Then, merge the adjacent pixels into the same region as long as their value is 1, so that we could get result in Fig. 13.

It seems like original water level is 4 in Fig. 14. However, different regions mix together when water level is 5 in Fig. 15. Consequently, we put a watershed between regions, and try again when water level is 5, and merging problem would not happen in Fig. 16.

2.6. Correction of Section 2.3

We sometimes need to scan from bottom to top, from right to left again after Section 2.4. Fig. 16 is another case after Section 2.4. However, although we set watershed to keep yellow region and blue region from merging together, they would still be merged together from the top left in Fig. 17.

Accordingly, we also scan from bottom to top, from right to left in Fig. 18, and we could keep them from merging together. The result is shown as in Fig. 18 and an example is shown in Fig. 20.

Table 1: Compare of the original and our proposed methods.

Original Method	Proposed Method.
Scan pixel in whole image, and merge with adjacent region.	Region merging with adjacent pixel in one time.
Check every pixel merged with which adjacent region iteratively.	Region merging with adjacent pixel in one time.
Much execution time.	Less execution time.
Over-segmentation problem.	Segment more appropriately.
No flexibility.	More adjusting parameters in application like recognition and compression.

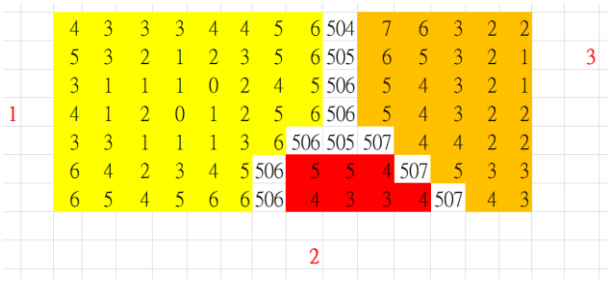

Fig. 20: Result of example in introduction.

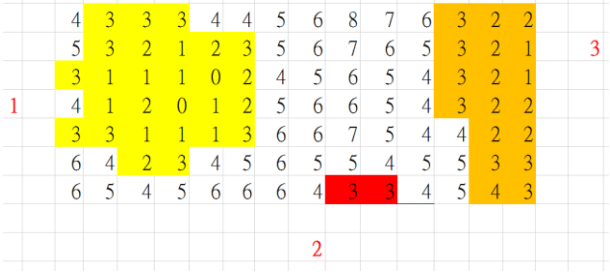

Fig. 21: Result of allocated area when threshold is 3.

Fig. 22: Scan pixel in whole image, and merge with adjacent region. Yellow pixel circled with green pen is pixel being scanned.

Fig. 23: Scan pixel in whole image, and merge with adjacent region. Yellow pixel circled with green pen is pixel being scanned.

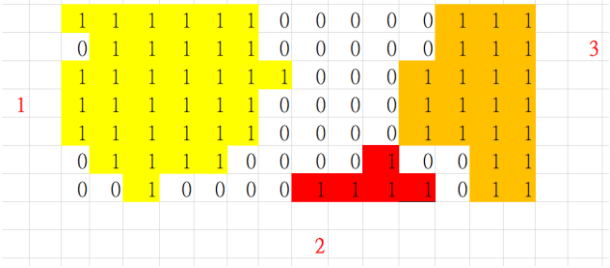

Fig. 24: Result of proposed method.

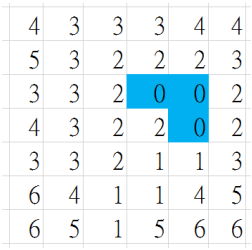

Fig. 25: Result of allocate area when threshold is 0.

3. SIMULATION & COMPARISON

In this section, we could see comparison between original method and our proposed method in Table 1.

3.1. Merging with Adjacent Region

The original threshold is 3, Fig. 21 is result of allocated area when threshold is 3. Then, we add the threshold by one and the threshold is 4 now.

3.1.1. Original Method

In original method, we should scan pixel in whole image, and merge with adjacent region, for example yellow pixel circled with green pen in Figs. 22 and 23.

3.1.2. Proposed Method

In proposed algorithm, we do not have to scan pixel in the whole image and compare with adjacent region, we just turn pixel lower and equal threshold into one, otherwise into zero, and we use the binary segmentation method. Accordingly, we get result in Fig. 24 directly, and we do not have to scan pixel in the whole image.

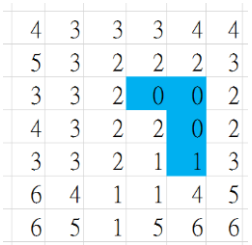

Fig. 26: Scan quantized gradient and merge pixel with adjacent region, if its value is 1.

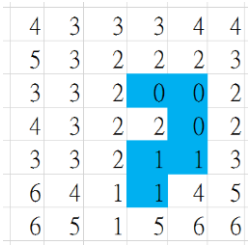

Fig. 27: Scan quantized gradient and merge pixel with adjacent region, if its value is 1.

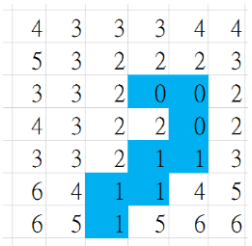

Fig. 28: Scan quantized gradient and merge pixel with adjacent region, if its value is 1.

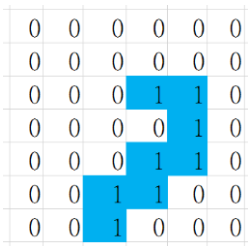

Fig. 29: Result of proposed method when threshold is one.

3.2. Proposed Iteration Method

The original threshold is 0. Fig. 25 is the result of area allocation when threshold is 0.

Then, we add threshold and one, so threshold is 1 now. In the original method, we should check every pixel merge with which adjacent region iteratively in Figs. 26-28.

However, in the proposed algorithm, we just turn pixel lower and equal threshold into one, otherwise into zero, and we use the binary segmentation method. Accordingly, we get result in Fig. 29 directly, and we do not have to scan pixel iteratively.

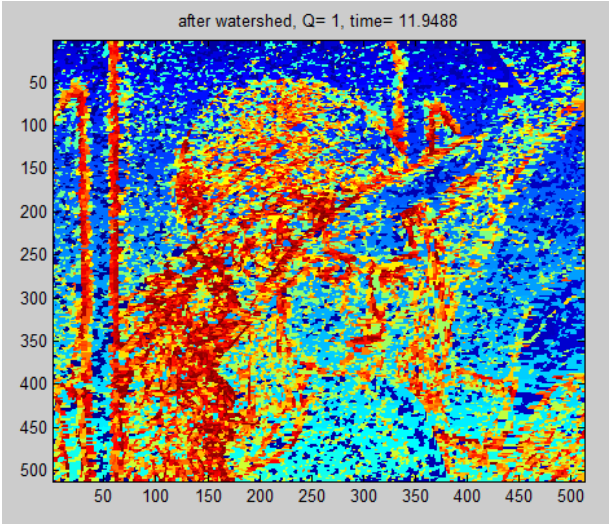

Fig. 30: If pixels are allocated to the same region, the color of them would be totally the same. We use 1 as quantized number Q, then, we should spend 11.9488 seconds to segment Lena image.

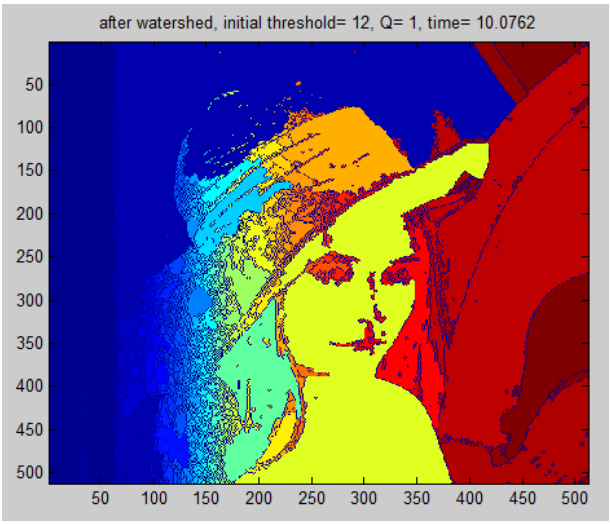

Fig. 31: We set original threshold as 12; quantized number Q is still 1; and we use our proposed method

3.3. Time Delay Result from Over-Segmentation Problem

In this section, we decrease computation time result from over-segmentation problem, and we tried on many cases. The results are in Table 2.

3.3.1. Original Method

In the original method, if we segment Lena image, and use 1 as quantized number Q, we should spend 11.9488 seconds. We get result in Fig. 30 and Fig. 32.

3.3.2. Proposed Method

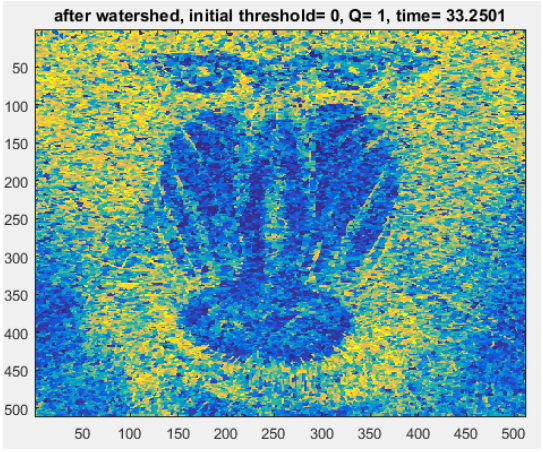

Fig. 32: If pixels are allocated to the same region, the color of them would be totally the same. We use 1 as quantized number Q, then, we should spend 33.2501 seconds to segment Baboon image.

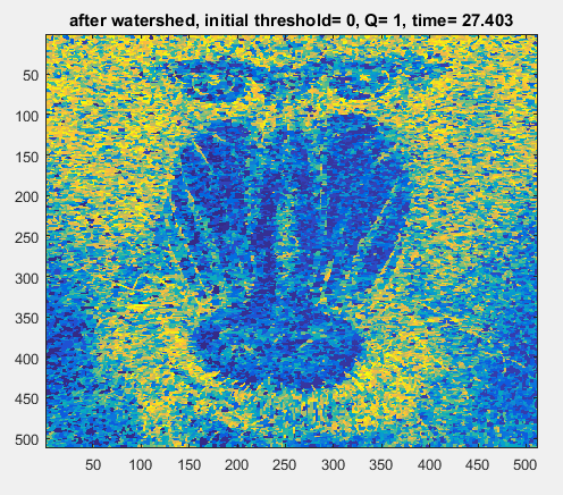

Fig. 33: We set original threshold also as 0; quantized number Q is still 1; and we use our proposed method when threshold equals 0 and 1.

In the proposed method, because we not only segment image into fewer regions, but also decrease time of merging with adjacent pixel. And we solve the iteration problem, so the complexity is decreased, and then we could spend less time and get better result.

The results of the proposed algorithm are shown in Fig. 31 and Fig. 33. We set the original threshold as 12; quantization number Q still is 1; and we use our proposed method.

3.4. Over-segmentation Problem

3.4.1. Original Method

Over-segmentation problem happens in original method. It means that we divide into too many regions, so we get too many data, and we should use a large number of bits to save data. If we have too many data, our compression efficiency would not be good.

Table 2: Computation time when using the original and the proposed algorithms for image segmentation.

Test Images	Original Method	Proposed Method
Lena	19.1726	12.8798
Airplane	16.7737	12.5247
Baboon	20.4523	19.2475
Barbara	20.6602	16.5796
Boats	20.003	16.7301
bridge	19.8715	16.8298
Cameraman	2.0124	1.8985
Goldhill	17.9745	16.6462
House	1.5944	1.5216
Peppers	2.2101	1.784
Village	18.5827	15.0871
zelda	17.2917	12.2994

Table 3: Region numbers after segmentation with original or our proposed method.

Test Images	Original Method	Proposed Method
Lena	29,973	5,941
Airplane	25,885	5,143
Baboon	38,279	30,299
Barbara	34,366	19,112
Boats	28,748	7,650
bridge	32,667	25,057
Cameraman	7,077	24,22
Goldhill	29,704	7,582
House	6,654	701
Peppers	29,048	3,462
Village	29,704	7,582
zelda	25,538	2,250

3.4.2. Proposed Method

When applying the proposed algorithm, Lena image is divided into 5,941 regions, as in Fig. 31. By contrast, when suing the original algorithm, Lena image is divided into 29,973 regions, as in Fig. 30.

We could divide Lena image into fewer regions with our proposed methods, and we could allocate region with higher consistency. For example, in original method, we divide the face of Lena into many regions. But in our proposed method, we divide the face of Lena into only one region, so we could get better compression efficiency thanks to dividing region with higher consistency.

We also segment other images with original method and our proposed method. The result is in Table 3.

3.5. Flexibility of the Proposed Method

3.5.1. Recognition

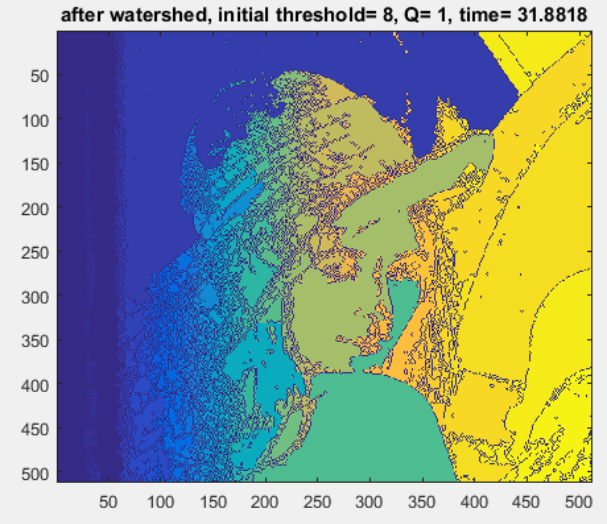

Fig. 34: If we set initial threshold as 8 and quantized number as 1, we could divide the person's eyes, nose, mouth, and cheek into different regions.

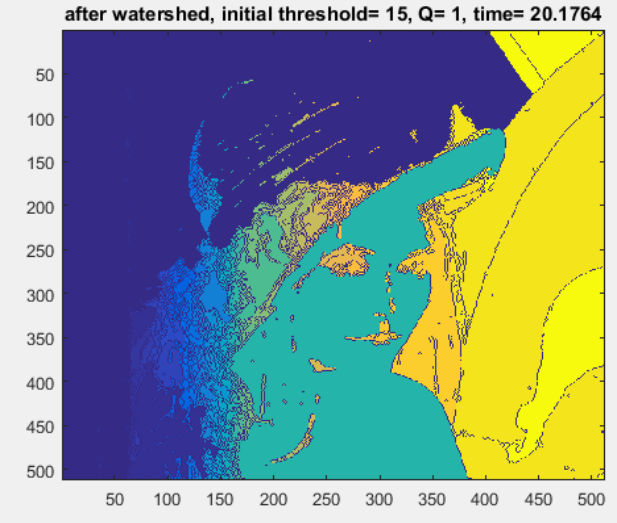

Fig. 35: If the initial threshold is set to 15 and the quantization factor is 1, we could divide the whole face into the same region.

The proposed method is more flexible. For example, we could use segmentation to help recognition. If we want to recognize who the person is, we could choose original threshold, so the person's eyes, nose, mouth, and cheek would be allocated to different region, and the result is in Fig. 34.

However, if we just want to know how many people there are, we could set higher initial threshold, so the whole face would be allocated to the same region. Accordingly, if we need less information, we could increase initial threshold to decrease computation time in Fig. 35.

3.5.2. Compression

When we use segmentation to help compression, if we want to have better quality, we decrease initial threshold

like Fig. 34. However, if we want to have better compression efficiency, we could increase initial threshold, like Fig. 35.

4. CONCLUSION

In original algorithm, we should check 4 adjacent pixels with pixels whose value equals threshold. Values equal to threshold just like numbers circled in Fig. 3. Furthermore, initial threshold must be zero in original method, so it is not flexible, and it would cause over-segmentation problem.

By contrast, in our proposed method, we do not have to compare 4 adjacent numbers. We just have to do some operation in logic matrix, so we can increase speed of watershed algorithm, and get the same result. In addition, we can change initial threshold according to our requirement to get more flexibility, and over-segmentation problem would be solved with our proposed method.

ACKNOWLEDGE

The authors thank for the support of Ministry of Science and Technology, Taiwan, under the contracts of 107-2221-E-002 -120 -MY3

REFERENCES

[1] H.P. Ng, S.H. Ong, K.W.C. Foong, P.S. Goh, and W.L. Nowinski, “Medical Image Segmentation Using K-Means Clustering and Improved Watershed Algorithm,” *IEEE Southwest Symposium on Image Analysis and Interpretation*, pp. 61-65, 2006.

[2] A. P. Mangan and R. T. Whitaker, "Partitioning 3D Surface Meshes Using Watershed Segmentation," *IEEE Trans. Visualization and Computer Graphics*, Vol. 5, pp. 308-321, 1999.

[3] L. Shafarenko, M. Petrou, and J. Kittler, “Automatic Watershed Segmentation of Randomly Textured Color Images,” *IEEE Trans. Image Processing*, Vol. 6, No. 11, pp. 1530-1543, Nov. 1997.

[4] H.T. Nguyen, M. Worring, and R. van den Boomgaard, “Watersnakes: Energy-Driven Watershed Segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 25, No. 3, pp. 330–342, Mar. 2003.

[5] I. Patras, E. A. Hendriks, and R.L. Lagendijk. “Video Segmentation by MAP Labeling of Watershed Segments,” *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 23, No. 3, pp. 326-332, Mar. 2001.

[6] I. Levner and H. Zhang. “Classification-Driven Watershed Segmentation,” *IEEE Trans. Image Processing*, Vol. 16, No. 5, pp. 1437-1445, May 2007.

[7] Y.C. Lin, Y.P. Tsai, Y.P. Hung, and Z.C. Shih, “Comparison between Immersion-Based and Toboggan-Based Watershed Image Segmentation,” *IEEE Trans. Image Processing*, Vol. 15, No. 3, pp. 632–640, Mar. 2006.

[8] V. Grau, A. U. J. Mewes, M. Alcaniz, R. Kikinis, and S. K. Warfield, “Improved Watershed Transform for Medical Image Segmentation Using Prior Information,” *IEEE Trans. Medical Imaging*, Vol. 23, No. 4, pp. 447–458, 2004.

[9] Y. Tarabalka, J. Chanussot, and J.A. Benediktsson, “Segmentation and classification of hyperspectral images using watershed transformation,” *Pattern Recognition*, Vol. 43, No. 7, pp. 2367–2379, Jul. 2010.

[10] Y. Tarabalka, J. Chanussot, J. A. Benediktsson, J. Angulo, and M. Fauvel, “Segmentation and Classification of Hyperspectral Data Using Watershed,” in *Proc. IEEE IGARSS*, Vol. 3, pp. 652–655, 2008.