doi:10.3772/j.issn.1006-6748.2021.01.012

Study on the de-watermark algorithm based on grayscale text①

Huang Guoquan(黄国权)*, Chen Zhipeng*, Sun Xiaocui②* **
(* Department of Computer Science, Guangdong Pharmaceutical University, Guangzhou 510006, P. R. China)
(** Medicinal Information & Real World Engineering Technology Center, Guangdong Pharmaceutical University, Guangzhou 510006, P. R. China)

Abstract

When using the current popular text recognition algorithms such as optical character recognition (OCR) algorithm for text images, the presence of watermarks in text images interferes with algorithm recognition to the extent of fuzzy font, which is not conducive to the improvement of the recognition rate. In order to pursue fast and high recognition rate, watermark removal has become a critical problem to be solved. This work studies the watermarking algorithm based on morphological algorithm set and classic image algorithm in computer images. It can not only remove the watermark in a short time, but also keep the form and clarity of the text in the image. The algorithm also meets the requirements that the higher the clarity of image and text, the better the processing effect. It can process the Chinese characters with complex structure, complicated radicals or other characters well. In addition, the algorithm can basically process ordinary size images in 1s, the efficiency is relatively high.

Key words: de-watermark, text recognition, character recognition, optical character recognition (OCR) application

0 Introduction

With development of text recognition, the higher the quality of text images in the algorithm of text recognition, the easier the identification of text recognition[1]. At present, optical character recognition (OCR) technology is widely used in character recognition applications[2-5]. OCR technology uses electronic devices, such as scanners or digital cameras, to examine characters printed on paper to determine their shape by detecting dark and bright patterns, then uses the shape to identify and pair with the corresponding font to translate its shape into computer text. Compared with the classic OCR, OCR is no longer limited to scanners or other electronic devices, but a generic image can also be used through pixel-level scanning to complete the text recognition of the image with OCR technology[6-7]. At this stage, OCR is widely used in text recognition in all walks of life, but the recognition rate of OCR technology is limited. Therefore, many researchers use different schemes to improve the recognition rate of OCR, such as OCR recognition method based on context semantic lexicon[8-9], which mainly uses common text to modify the unusual text in the recognized string with the context to improve the recognition rate. This method effectively solves the disadvantage of the image type text.

Sometimes there are fuzzy watermarks in the original images that need to be recognized[10]. These watermarks will affect the accuracy and speed of the whole character recognition. Especially in the Chinese character recognition, when some Chinese radicals are disturbed, they will completely affect the meaning of the characters. So in the process of Chinese text recognition, it is better to remove the interference caused by these watermarks.

Currently, the watermark technology is also improving. In the study of digital image watermarking technology, in order to improve the anti-attack of watermarking, many fields use watermarking technology such as discrete cosine transform (DCT), and digital watermarking (DWM)[11-13]. According to the transparency, watermark is divided into invisible watermark and visible watermark. However, the invisible watermark does not have a great impact on text recognition algorithm. This paper describes a new algorithm which uses the characteristics of watermark and font color

① Supported by the National Entrepreneurship Training Fund Project (No. 201810573007X), China Scholarship Council Project (No. 201708440547) and GDPU Higher Education Innovation Strong School Project (No. 2017GXJK079, 51359055).
② To whom correspondence should be addressed. E-mail: xiaocuisun1002@hotmail.com
Received on Apr. 10, 2020

separation to remove watermark, commonly known as image deletion attack.

1 Image binarization algorithm

Image binarization uses the classic threshold algorithm[14]. Threshold algorithm is the simplest segmentation method, which mainly completes the element level segmentation through threshold operation. The most common way is to change all pixels of an image to 0 or 255, that is, image binarization. Therefore, in many classic graphics tutorials, threshold binary algorithm will be introduced separately as a special independent threshold algorithm.

The threshold binary algorithm processes all pixels in the input image. If the value of the pixels is greater than the given threshold, it is changed to the value *maxVal*, while the other ones less than or equal to the threshold is changed to the value 0. Until all pixels are traversed successfully, the image will be modified to a binary image. In graphics, the threshold is a boundary value, which is typically used to represent a boundary value in a image algorithm. Eq. (1) is a mathematical representation of the thresholding binary algorithm, where the dst function returns pixels of the output image based on coordinates, and the src function returns pixels in the input image based on coordinates. *maxVal* is generally represented as the maximum value that pixels can represent, and on hardware it is white. 0 is generally represented as the minimum value of pixels, which corresponds to black on the hardware. *thresh* is a threshold to set.

$$\mathrm{dst}(x, y) = \begin{cases} maxVal & \mathrm{src}(x, y) > thresh \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Fig. 1 and Fig. 2 demonstrate an example. The pixel value distribution in Fig. 1 is the pixel distribution of the input image. The pixel distribution in the output image after the threshold binary algorithm processing is shown in Fig. 2. All pixel values become only 0 and *maxVal*, where the line is the set threshold. The pixel distribution graph is as follows: all values greater than the line are set to *maxVal*, and others are set to 0.

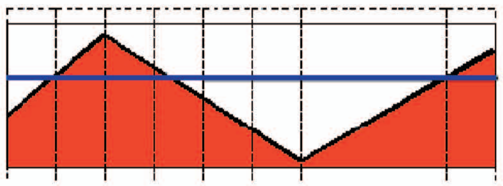


Fig. 1 Pixel distribution of input image

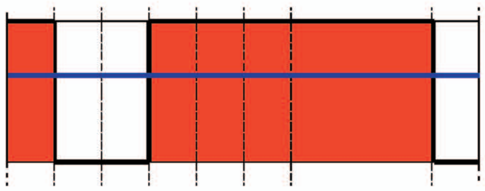
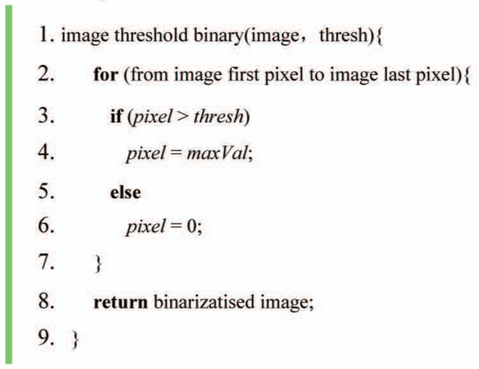


Fig. 2 Pixel distribution of output image after thresholding binary processing

Threshold binary can be defined as a function: image threshold binary (image, thresh), which is used to binary the input image according to the input threshold and return a processed image. In the de-watermark algorithm, binarization will change the required gray scale image into only 0 and *maxVal*, which is the key role of the de-watermark algorithm. It is mainly convenient to generate the selected area and complete more refined image clipping. Its pseudo code is shown as Fig. 3.

Take Fig. 4 as an example, and perform threshold entry on it (threshold is set to 128). After the threshold binary processing, it will be converted to the appearance as Fig. 5.



```
image threshold binary(image, thresh){
    for (from image first pixel to image last pixel){
        if ($pixel > thresh$)
            $pixel = maxVal$;
        else
            $pixel = 0$;
    }
    return binarizatised image;
}
```

Fig. 3 Pseudo code for threshold binary

Fig. 4 Lena

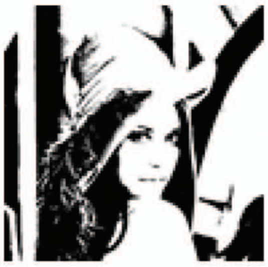


Fig. 5 Lena after threshold binary

2 Image color reduction algorithm

The image color reduction algorithm uses the classic algorithm look up table (LUT)[15]. The main purpose of the algorithm is to find the corresponding value of the pixel value according to a specified array. One of the basic starting points is to standardize the pixel points of the image on the basis of preserving the large area details of the image. For example, Eq. (2) is the formula to search for the pixel, I_{old} is the pixel to be input, and I_{new} is the value of the new pixel. Then traverse all the pixels in the image once, then all the pixels in the image will become multiples of n, and the color will be reduced as a whole.

$$I_{new} = \lfloor \frac{I_{old}}{n} \rceil \times n \qquad (2)$$

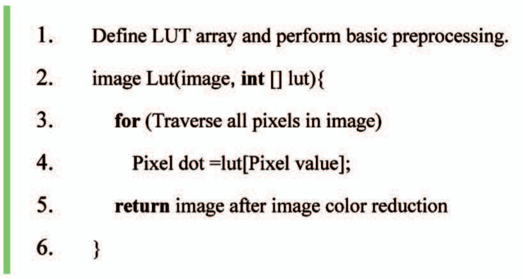
However, there are thousands of pixels in an image, so it will cost a lot of resources to call thousands of division and multiplication. To do this, it can be built a LUT. If the matrix element stores a single channel pixel, and uses the C or C++ unsigned character type, then the pixel can have 256 different values. After that an array with a length of 256 can be built and indexes of all the possible values are calculated in advance by the corresponding I_{new}. When these values are needed, they can be assigned directly by using the lookup table. The next traversal does not need to call multiplication and division, while only a simple array reading is needed. Lookup table is a one-dimensional or multi-dimensional array, which stores the output values corresponding to different input values. Its advantage is that it only needs to read and does not need to do calculation.

Considering a pixel in the source image with value 230, the new value 200 will be obtained by calculation according to the above formula. It seems that there is a big difference between 230 and 200. In fact, in many common algorithms, they basically belong to the same color and do not need to be so refined. However, if a picture meets the required values of the algorithm, discarding those useless values can greatly improve the algorithm. For example, in text recognition, in the dewatermark algorithm, only three colors (white, gray and black) need to be considered. For this algorithm, when a picture is introduced into RGB or a picture with fine pixels, it will become a burden. Some definitions of LUT in OpenCV is shown in Eq. (3).

$$\mathrm{dst}(I) \leftarrow \mathrm{lut}(\mathrm{src}(I) + d)$$
$$d = \begin{cases} 0 & \text{if scr has depth } CV\_8U \\ 128 & \text{if scr has depth CV\_8}S \end{cases} \qquad (3)$$

LUT can be defined as a function: image Lut (image, int[ ] lut), where Lut is an array that has been calculated outside the function, and its length should meet *maxVal*. If it is an unsigned uchar, its length should be 256, so that the value of the new pixel can be exactly equal to the value in the LUT array index corresponding to the old pixel value. The function of color reduction is to subdivide the difference between watermark and text. Try to subdivide and separate the white watermark and black font in the picture at the pixel level. Its pseudo code is shown in Fig. 6.

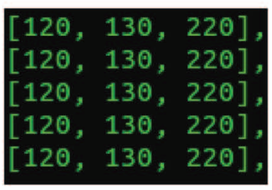
To solve this problem, we only need to find an int array Lut with appropriate picture changes, and then it can be used Lut function to properly handle the function. Take Fig. 4 as an example, initialize Lut with $n = 10$ in Eq. (2) for Fig. 4, and process function Lut (Fig. 4, Lut). Fig. 7 shows the result of Fig. 4 after the processing. By careful comparison, the naked eye can't see the difference between Fig. 4 and Fig. 7, but in the program, all the pixels in Fig. 7 have been adjusted to a multiple of $n = 10$ (Fig. 8). However, due to the micro changes in pixels, almost all the details of the image are preserved.



```
1. Define LUT array and perform basic preprocessing.
2. image Lut(image, int [] lut){
3.     for (Traverse all pixels in image)
4.         Pixel dot =lut[Pixel value];
5.     return image after image color reduction
6. }
```

Fig. 6 Pseudo code for image Lut

Fig. 7 Image processed by Lut



```
[120, 130, 220],
[120, 130, 220],
[120, 130, 220],
[120, 130, 220],
[120, 130, 220],
```

Fig. 8 Partial pixel value of Fig. 7

3 Image corrosion and expansion algorithm

Image dilation and eroding are common and simple morphological algorithms in computer graphics, they are widely used in graphics. The most important applications are noise elimination, sketching edges, expanding edges and so on.

Dilation superposes an image A and a kernel B with any shape and size, but usually rectangular and circular. Kernel B has an anchor point which is generally the kernel center. Then B will scan along picture A and calculate the maximum pixel value after superposing with B. For brighter areas it will get larger, so this is called dilation. Image comparison before and after general operation is shown as Fig. 9 and Fig. 10.

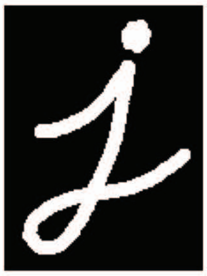


Fig. 9 Before dilation

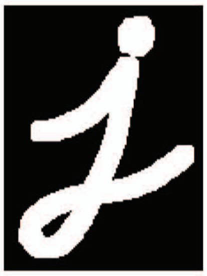


Fig. 10 After dilation

Eroding, the operation of eroding is similar to dilation. It is also the superposition of image A and a kernel B with any shape and size. However, after the superposition of eroding, the minimum pixel value is calculated, so the darker area for corrosion will become larger. These are the similarity and difference between dilation and eroding. An example is shown as Fig. 11 and Fig. 12.

Kernel is actually an array, which describes a binary mask of morphological operations such as dilation and eroding. Its anchor point describes the distance of movement. Therefore, generally, with the increase of the size of the kernel, the effects of dilation and eroding are increasingly obvious.

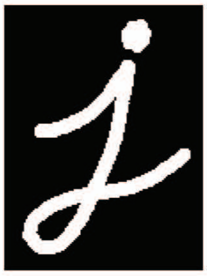


Fig. 11 Before eroding

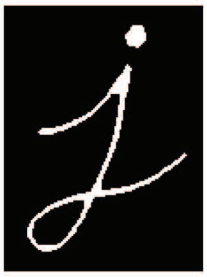


Fig. 12 After eroding

The dilation calculation formula is shown in Eq. (4), where the max function is used to find the maximum pixel value in src $(x + x', y + y')$ and change the points around the pixel into the pixel value according to the kernel.

$$\mathrm{dst}(x, y) = \max_{(x', y'):element(x', y') \neq 0} \mathrm{src}(x + x', y + y') \tag{4}$$

The eroding calculation formula is shown in Eq. (5), but the minimum value is found by min.

$$\mathrm{dst}(x, y) = \min_{(x', y'):element(x', y') \neq 0} \mathrm{src}(x + x', y + y') \tag{5}$$

In order to make use of this function directly, it will be named that image erod (image, kernel), where image represents the source image, kernel is the required kernel, and its return value is just a picture with modified pixels. The eroding algorithm is relatively complex, so the pseudo code is not expanded here. The selection is assisted by eroding algorithm. Because of the coarseness of binarization, when using the image binarization algorithm described in Section 1, for the edge of text, the sawtooth is very serious. In this way, the areas selected in images are not smooth, which will influence the selected effect. Eroding algorithm can help to complete this move, make the edge of the selected area smooth, so as to achieve the desired effect.

4 De-watermark algorithm

The de-watermark algorithm can be divided into four steps: color space reduction, text area selection, text selection area dilation and text selection. The original image should be a gray image or an RGB image that can be converted into gray image. The color of its watermark is significantly different from that of the font, with a difference of at least 16 pixels. In this paper, an image with white background and black font with gray watermark is used as the sample of this algorithm (Fig. 13).

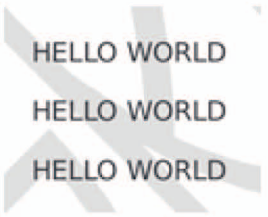


Fig. 13 Sample image with watermark

4.1 Color space reduction

This part uses LUT algorithm. In the process of pixel reduction, LUT algorithm does not cause the text position offset or shape distortion. Its function is mainly to increase the difference between text and watermark, that is to say, to make the color between text and watermark more distinct, so that the image is ready for the next step (text area selection). In the grayscale image, the value range of the image pixel is [0, 255]. Its color is shown as Fig. 14.

Fig. 14 Grayscale value ranges

As the pixel value increases, the color tends to be white. In the sample image, the text color is black, so its pixel range should be approximately [0, 128]. The color of the watermark is gray, so its pixel range should be approximately [128, 224]. The pixel value of background color should be 255 as it is white. In order to make the black more prominent and the watermark less obvious, LUT algorithm should meet the following requirements: the pixels in the font color range [0, 100] become smaller, while those of the watermark color range [128, 200] become larger and the white background remains unchanged. Therefore, Eq. (6) can be used to calculate every pixel of the image.

$$I_{new} = \begin{cases} \left(\frac{I_{old}}{32}\right) \times 32 & I_{old} < 128 \\ \left(\frac{I_{old} + 32}{32}\right) \times 32 - 1 & \text{otherwise} \end{cases} \quad (6)$$

After LUT's color reduction, the color distribution will become a color block with an interval of 16. Any pixel value less than 128 will be reduced to the value which is the multiple of 16 and nearest smaller than it, and any pixel value greater than 128 will be expanded to the value which is the multiple of 16 and nearest larger than it. And the formula also keeps the background color unchanged (Fig. 15).

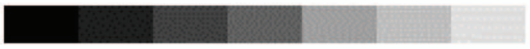


Fig. 15 Grayscale image value after LUT algorithm processing

At this time, the color value of the text will be {0, 32, 64, 96}, and the color value of the watermark will be {159, 191, 223, 255}. The difference between the maximum color value of the text and the minimum color value of the watermark is 63. Therefore, it is more obvious to separate the text from the watermark in the image, which is helpful to the construction of the text selection area.

4.2 Text area selection

The method to construct the selection area is to use thresholding algorithm to binarize the image. The most critical thing is to perform the binarization process on the image which is processed by the LUT algorithm in the previous step to enlarge the difference between the watermark and the text. This can accurately binarize and reduce the sawtooth. Because the selection area is completely constructed by the binarization in the source image, so some requirements should be followed.

(1) The selected area must be the same size as the source image.

(2) If there are only 0 and *maxVal* in the selection area, the part with pixel equal to 0 must be the content selected by the selection area, and the part with pixel equal to *maxVal* is the unselected part. In other words, the black part is selected by binary operation, while the white part is not selected.

Before directly binarizing the source image, the pixels of the image around the text in the source image and their edges can be observed, as shown in Fig. 16.

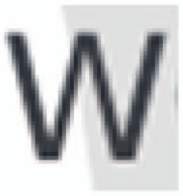


Fig. 16 Partial image

The pixels at the edge of the font are lighter than the font color and darker than the watermark color. If there is no LUT processing, during the binarization process, there may be a part of watermark be selected. At the same time, an acceptable threshold should also be established, which is thresh. After several attempts, it is found that when thresh is equal to 120, after binarization, the following selection will be obtained as Fig. 17.

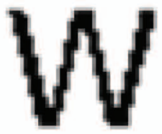


Fig. 17 Partial image after binarization

Next, we can select the text in the original image according to the selection, but the result will not be satisfactory at this stage. The reason is simple. The selection area is not smooth and there are many zigzag areas. If the selection area cannot fully cover the font, the font obtained must also have very serious zigzag. Therefore, in order to obtain more information or more perfect font, the font should be expanded select and the selection should be smoothed. Although the sawtooth still exist, but this has a better threshold, a more perfect selection of sawtooth repair version could be achieved through eroding.

4.3 Text selection area dilation

After the text selection in the previous step, a binary selection image is obtained. At this time, it can be used as the source image to select the text portion of the source image that needs to be intercepted. But a smooth selection is desired instead of one with obvious sawtooth to reach a selection area where you can get more information about the text of the source image.

For this, more black part should be selected in the image. We only need to perform eroding operation on the selected image. But it should be understood that the selection of the kernel in the eroding operation will affect the effect of the whole eroding. If there are more small fonts in the original image, the kernel effect is very crucial, that is to say, the greater the degree of black eroding, the easier to distinct the font from the watermark. Therefore, it is suggested that the kernel should not be too large or too small. If the selection area needs to be dilated, a better scheme should be chosen. For example, first enlarge the font, and then use it to do selection. Font style also affects the selection. If it is not bold, the kernel can be enlarged in some extend.

As the font in the example picture is not bold and thin, and the font is comparatively small in the image. The most suitable eroding kernel for this example is a 2 × 2 rectangle. The selection after eroding will be shown in Fig. 18.

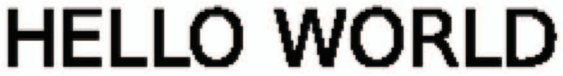


Fig. 18 Selection after eroding

The sawtooth is not so obvious. The most important thing is that the selected area is eroded outwards and the black part is more obvious than the previous step. This will be more conducive to the next step of text selection.

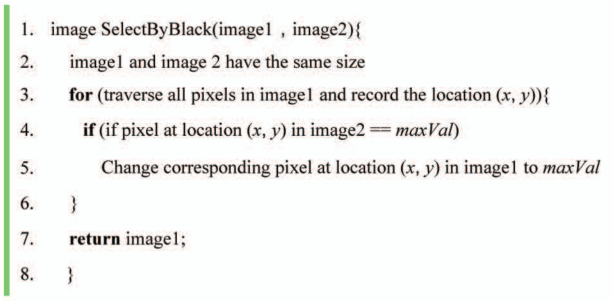
4.4 Text selection

Even if the text selection is expanded, the pixels in the selection image still remain in a binarized state, that is, their pixel values are still maintained as either 0 or *maxVal*. The size of the selected image is the same as that of the source image, and the black part in the selection, that is, the part with 0 pixels is the place where the text is to be selected from the source image.

So the process of selecting a picture is very simple. It is to traverse the whole image, keep the pixels with 0 pixels in the selection picture corresponding to the pixels in the original image, and replace the rest with a white background.

Define this step as a function SelectByBlack to represent the algorithm of the selected text: image SelectByBlack (image1, image2). The return value is a processed image, the parameter image1 is the source image, and the parameter image2 is the selected image. Its C pseudo-code is shown in Fig. 19.

Take Fig. 18 as the selected image and Fig. 13 as the source image and invoke the SelectByBlack function. The processed image is shown in Fig. 20.



```
image SelectByBlack(image1 , image2){
  image1 and image 2 have the same size
  for (traverse all pixels in image1 and record the location (x, y)){
    if (if pixel at location (x, y) in image2 == maxVal)
      Change corresponding pixel at location (x, y) in image1 to maxVal
  }
  return image1;
}
```

Fig. 19 Pseudo code for SelectByBlack

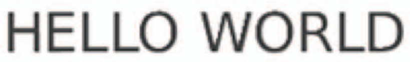


Fig. 20 Final image preview

The text in the original image is almost perfectly captured without excessive sawtooth. Of course, if the quality of the original image is not good enough and the blur is unclear, then the result obtained by the watermarking algorithm is not good, as its quality is the same as the source image. The size and resolution of the text are related.

4.5 Algorithm summary and flow chart

Input: a gray image with clear text.

Output: a gray image with watermark removed.

Algorithm flowchart is shown in Fig. 21.

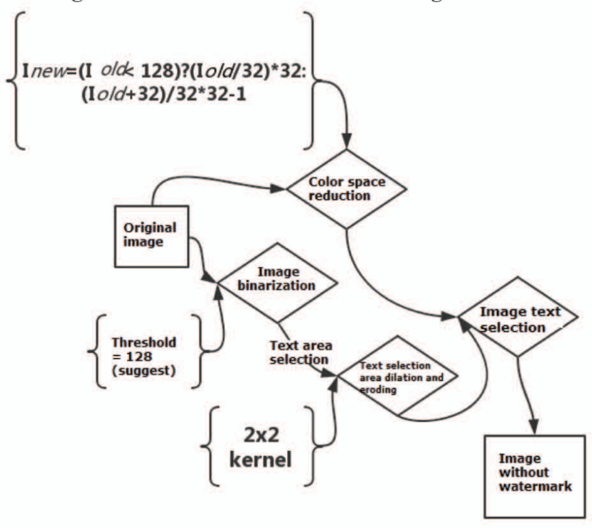


Fig. 21 Algorithm flowchart

4.6 Complete processing results

This section gives detailed description of the processing result of Fig. 13, which size is 306 × 246 pixels. The data generated below comes from ubuntu18.4 + OpenCV 4.4.0 environment. The code is written by C++, using GCC 7.5.0 compiler.

Performing 10 000 color space reduction operations on Fig. 13 takes an average of 0.4064 s. The image after color space reduction is shown in Fig. 22.

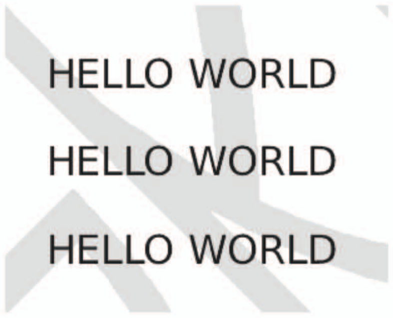
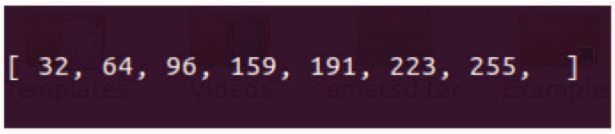


Fig. 22 Full image after color space reduction

In terms of vision, the image does not change too much, but in terms of image pixel value, Fig. 22 is composed of a finite set of pixel values as shown in Fig. 23.



```
[ 32, 64, 96, 159, 191, 223, 255,  ]
```

Fig. 23 Image pixel value

Taking Fig. 22 as input, 10 000 times of text area selection is constructed, with an average time of 0.0377 s. The image after text area selection is shown in Fig. 24.

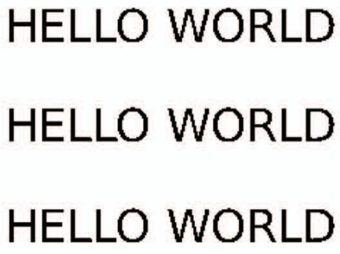


Fig. 24 Full image after text area selection

Taking Fig. 24 as input, 10 000 times of expanding image text selection takes 0.2242 s on average, and final full image after processing is shown in Fig. 25.

Among the processing step, the most time-consuming step is the color space reduction operation, accounting for 61% of the total time, which is closely related to the size of the image. If it can be predicted in

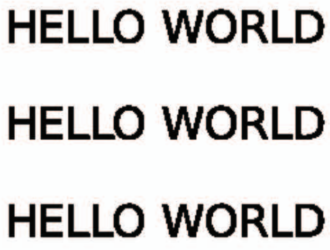


Fig. 25 Final full image

advance that the gray difference between the watermark and the font is large, we can ignore the color space reduction step to reduce the time consumed, and directly use a more radical threshold algorithm. Taking Fig. 13 as an example, if it is carried out directly and aggressively, the watermark problem can be solved in a very short time, but at the same time, the effect of the sawtooth will be more obvious, and the effective set represented by the pixel value will be expanded more complex.

In a specific case, if the grayscale value of the watermark of the grayscale image is completely consistent and has a huge difference from the text, direct threshold segmentation may obtain good results. However, in most cases, the above conditions could not be met, so the watermark removal may be incomplete. In this case, the shallower part of the watermark may be removed, but the deeper part could not be removed completely.

4.7 Algorithm evaluation

The speed of the algorithm is related to the size of the image. By using C++ in the OpenCV library, it can call the relevant morphological functions to evaluate the different size of the pictures, and collect the following tables, where the number of pixels is represented by the size of the pictures.

The speed of the algorithm is related to the size of the image. Using C++ to call some related morphological functions in the OpenCV library to perform different rate evaluations on images of different sizes. The following Table 1 is collected, where the number of pixels is represented by the image size, the unit of time taken is seconds.

The speed of the algorithm is considerable. For common-size images, the watermarking operation can be completed in less than 0.1 s. In practical applications, it can not only reduce or eliminate the watermark in the image in a short time, but also greatly reduce the time for the future text recognition algorithm to recognize the text. This is worth the image in entering the text recognition algorithm to a watermark algorithm. It is worth to perform image watermarking algorithm before text recognition algorithm.

Table 1 Time-consuming of different size images

Size of image/pixel	233 × 385	451 × 736	510 × 1079	1169 × 826
Time consuming/s	0.0072	0.0305	0.0480	0.0819

5 Conclusions

In order to increase the OCR text recognition rate, an algorithm is designed to efficiently remove watermarks in text images. In text images with watermarks, if the watermark is lighter than the text and does not cause large-scale occlusion of the text, this algorithm can effectively remove the watermark in such images. The algorithm of removing watermarks is mainly achieved by reducing pixels and performing eroding on the basis of image binarization. The proposed algorithm uses some algorithms of graphics morphology to achieve the accurate elimination of watermark in the text, so as to retain the details of the font to the greatest extent. It can deal with Chinese characters with complex structure or complicated radicals or other characters well. In addition, the efficiency of this algorithm is relatively high, as it can basically process ordinary size images within 1 s.

In the future, the algorithm should be improved to more friendly to RGB images and the atypical white watermark to enhance the accuracy of the algorithm.

References

[1] Breuel T M. High performance text recognition using a hybrid convolutional-LSTM implementation [C] // The 14th IAPR International Conference on Document Analysis and Recognition, Kyoto, Japan, 2017: 11-16

[2] Nikbakht P, Mahdavi M. Targeted dewatermarking of two non-blind SVD-based image watermarking schemes [C] // 2015 5th International Conference on Computer and Knowledge Engineering, Mashhad, Iran, 2015: 80-86

[3] Naseer A, Zafar K. Meta features-based scale invariant OCR decision making using LSTM-RNN [J]. *Computational and Mathematical Organization Theory*, 2019, 25: 165-183

[4] Kaothanthong N, Theeramunkong T, Chun J. Improving Thai optical character recognition using circular-scan histogram [C] // The 14th IAPR International Conference on Document Analysis and Recognition, Kyoto, Japan, 2017: 567-572

[5] Zhang Z, Wang H, Liu S, et al. Deep contextual stroke pooling for scene character recognition [J]. *IEEE Access*, 2018, 99: 1-1

[6] Chan K, Yeung D. Recognizing on-line handwritten alphanumeric characters through flexible structural matching [J]. *Pattern Recognition*, 1999, 32(7): 1099-1114

[7] Tkachenko I, Gomez-Krämer P. Robustness of character recognition techniques to double print-and-scan [C] // The 14th IAPR International Conference on Document Analysis and Recognition, Japan, Kyoto, 2017: 27-32

[8] Fu Q, Ding X, Liu C. Modified AdaBoost algorithm for handwritten Chinese character recognition [J]. *Chinese High Technology Letters*, 2009, 19(4): 5-10 (In Chinese)

[9] Cox I J, Kilian J, Leighton T, et al. A secure, robust watermark for multimedia [C] // International Workshop on Information Hiding, Berlin, Germang, 1996: 185-206

[10] Wu J, Shi H, Zhang S, Leighton T, et al. De-mark GAN: removing dense watermark with generative adversarial network [C] // 2018 International Conference on Biometrics, Gold Coast, Australia, 2018: 69-74

[11] Nikolaidis A, Pitas I. Region-based image watermarking [J]. *IEEE Transactions on Image Processing*, 2001, 10 (11): 1726-174

[12] Satyanarayana M P, Rajesh K P. Towards robust reference image watermarking using DWT- SVD and edge detection [J]. *International Journal of Computer Applications*, 2014, 68(9): 10-15

[13] Rey C, Doerr G, Csurka G, et al. Toward generic image dewatermarking? [C] // Proceeding of International Conference on Image Processing, New York, USA, 2002: 633-636

[14] Si B, Gao W, Lu H, et al. Image retrieval method based on regions of interest [J]. *Chinese High Technology Letters*, 2003, 13(5): 13-18 (In Chinese)

[15] Arsalan M, Malik S A, Khan A. Intelligent threshold selection for reversible watermarking of medical images [C] // The 12th Annual Conference Companion on Genetic and Evolutionary Computation, Portland, USA, 2010: 1909-1914

Huang Guoquan, born in 1974. He received his M. S. degree in system analysis and integration from Hubei University in 2006. He also received his B. S. degree in computer and application from Hubei University in 1997. His main research interests include software system architecture, artificial intelligence, knowledge map and medical information processing.