Algorithm of Fingerprint Extraction and Implementation Based on OpenCV

Yue Yaru, Zhu Jialin
Beijing Information Science and Technology University
BISTU
Beijing, China
e-mail: 9834904@qq.com, jlzhu@bistu.edu.cn

***Abstract*—At present, OpenCV function library is applied more and more widely, and it is used in digital image processing to solve some problems of image processing, and it can improve the effectiveness of image processing. Commonly it is easy to cause the loss of image detail for the Otsu method, in order to solve these problems the Otsu method is improved. In the case of uneven illumination and blurred image, it can segment the target, and the result is accurate, simple and shorter running time. What's more, it can reduce the amount of computation and storage space. In general, it is a fast and effective and good real-time image threshold segmentation algorithm. This paper uses the OpenCV functions to achieve a fingerprint extraction algorithm. The algorithm uses the Otsu algorithm improved to get the best threshold that it can segment image. Simulation experiment is carried out by using object-oriented Vc++6.0 programming tools, and it proves that the fingerprint extraction algorithm based on OpenCV function library is effective. It can improve the accuracy of fingerprint extraction, and image likes real image. Give some code.**

Keywords-opencv; otsu; fingerprint extraction; optimal threshold; Vc++6.0

I. INTRODUCTION

Lines of the fingerprint randomly appear all sorts of breakpoints and intersections because the fingerprint is affected by genetic and maternal environment, causing obvious differences of all fingerprints. That is to say, each person is unique. Rely on this only, you can put a person together with his fingerprint, according to one's fingerprints and prestored fingerprint comparison, you can verify the true identity of this person. In practice, fingerprint identification technology applies to security check, criminal investigation, Information matching and so on. Although this technology has been applied in many areas, and it is more mature, the technology isn't open because of the influence of commercial interests or other factors [1]-[4]. In the process of learning the OpenCV, I want to use this function library to study the fingerprint extraction, and it also has some theoretical significance and practical value.

In this paper, algorithm of fingerprint extraction based on OpenCV [5]-[10] open source visual function library is proposed, which can run on Linux, Windows, MacOs and other operating systems, and which has the advantages of fast computation and real-time. The algorithm of fingerprint extraction is used in Otsu algorithm in image segmentation mainly, which is a classical algorithm. And it can obtain adaptive threshold of the gray image to binarization. In the end, it can segment the target and background. However, it is found that the general gray transform will affect the details of images and even lost part of the image before using the Otsu algorithm due to read the relevant articles, which leads to the result that accuracy of processing image isn't high. We want to extract the details of fingerprint lines clearly. Then according to the image that its details demand higher and its illumination is light and its three components are uneven, it is proposed that using the weight conducts gray-scale transformation for fingerprint image. The method makes use of the human eyes that are the most sensitive to green, to improve the weight of the G channel [11]. The method meets human physiology. The algorithm can preserve the details of the image and Close to the content of the original image. And the process and results of the algorithm are given.

II. IDEA OF ALGORITHM

Usually, algorithms of adaptive threshold include Otsu algorithm, Iterative threshold method and Secondary fixed value method.

OTSU algorithm sometimes called Maximum between class difference method, and it was proposed by Otsu about in 1980, and it was derived based on the principle of least square method. Usually, it is considered to be the best algorithm of selecting the threshold in image segmentation because the OTSU algorithm that its calculation method is simple isn't affected by brightness and contrast of images [12], [13]. It is widely used in digital image processing, which is the practical foundation for the paper. So this method regards as the main body of the algorithm of fingerprint extraction. However, this algorithm is also flawed, which is improved in the process of programming.

The basic idea of Otsu algorithm is: using a gray value of image histogram, image is divided into two groups, a group of gray levels corresponding to the target, which is a set of corresponding fingerprint lines, and a group of gray levels corresponding to the background. When variance between the two groups is maximum, we take this gray value as the best threshold, and the image is divided into the two parts of background and foreground. In the histogram, the size of variance represents the situation of gray distribution. And the greater the variance between the target and the background shows that between the two parts of the difference is bigger, which is the better to separate the two parts. So the effect of segmentation is the best. However, when some of the targets are divided into the background, or some of the background

978-1-5090-6238-6/17/$31.00 ©2017 IEEE

is divided into the target, it will lead to the two parts of fuzzy difference, which will affect the loss of fingerprint image details. So the maximum variance between classes means that the misclassification probability is the smallest. Therefore, the gray scale transformation of the weighted average method before using the Otsu method is used to solve the defect of the image.

Hypothesis gray level of gray image is L, then the gray level range from 0 to [L-1], to calculate the optimal threshold of fingerprint image by the Otsu algorithm. Formula is as follow.

$$t = Max[w_0(t)*(u_0(t)-u)^2 + w_1(t)*(u_1(t)-u)^2] \quad (1)$$

In the formula, t represents the threshold of image segmentation; w_0 represents background ratio; u_0 represents background mean; w_1 represents proportion of foreground; u_1 represents foreground mean; u represents mean for the whole image. The t making formula (1) obtain the maximum value is the best threshold.

The Otsu algorithm regards as the main algorithm based on the OpenCV library, roughly algorithm of fingerprint extraction is: Firstly, the RGB three channels are decomposed, and the image of RGB three channels is obtained. Secondly, the weight of the G channel is improved to obtain the grayscale. Again, use the Otsu method to obtain the optimal threshold. Finally, according to the best threshold value, the image is segmented, and the fingerprint is extracted from it. Specific part is as follows.

III. ALGORITHM PROCESS AND PROCEDURE

Fingerprint extraction steps: acquisition of fingerprint images, image preprocessing (denoising, gray processing, enhancement), fingerprint extraction (segmentation). Fingerprint extraction flow chart as shown in Figure 1.

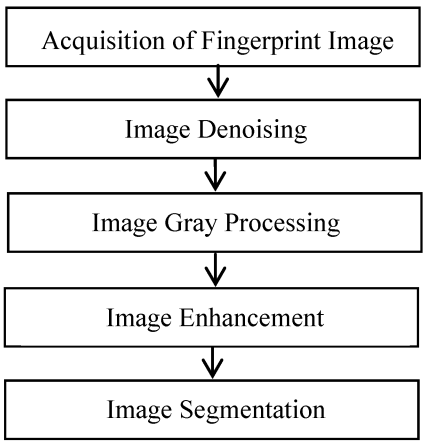

Figure 1. Fingerprint extraction flow chart.

What is said above, the process of fingerprint extraction can be used in fingerprint identification system to collect training samples. According to the characteristics of the fingerprint image, we select the appropriate feature extraction method that is as follow to achieve the purpose of extracting feature. Finally, the fingerprint features are stored as feature library. This process takes a lot of time, manpower and financial resources to complete. In the recognition phase, the extraction method based on OpenCV is used to extract the fingerprint features, and then the appropriate pattern matching algorithm is selected to match the characteristics of the training samples in the feature database. And finally output the recognition process.

A. *Image Denoising*

Although the Otsu method has the advantages of simple calculation and strong adaptive ability, it is very sensitive to the noise. So the fingerprint images collected carry out denoising processing [14], [15]. In this paper, the two-dimensional median filter is used to suppress the background noise. Because the median filtering method is compared with the neighborhood average method, it can eliminate the noise and preserve the details of the image, and prevent the bluring edge [16], which meets the needs of this paper. Median filter is a nonlinear method to remove noise. And its basic principle is to put a point in a digital image with the median value of each sample point in the field of a place. It is easy to be extended to two-dimensional that its windows are various. 3×3 smoothing filter template is selected. It can be seen from the final results, it has a good effect in image denoising.

B. *Image Gray Processing*

Convert the color image into gray image. As we know, in RGB model, if R=G=B, then the color represents a gray color, which is called the gray value that ranges from 0 to 255. And the greater the value is, the brighter the point is. The smaller the value is, the darker the point is. This is the basic idea of gray-scale transformation.

There are three methods of gray level transformation: the maximum method, the average method and the weighted average method.

In order to retain the details of the image, this paper uses the weighted average method. And to R, G, B gives different weight coefficients. And count weighted sum. Get the gray value $Gray$.

Conversion relationship:

$$Gray(i,j) = 0.11R(i,j) + 0.59G(i,j) + 0.3B(i,j)$$

Implementation steps: get the data area pointer; each pixel of the image is circulated, and R, G, B three component values of each pixel is calculated; the gray value $Gray$ is calculated according to the formula; R, G, B three component values of the corresponding pixels set as the same gray value.

Part of the program of image grayscale processing is as follows.

```
//The image isn't grayscale bitmap, which needs to convert
 if(!IsGrade())
 {
        //The space required for grayscale bitmap data is calculated; the space required for grayscale bitmap is calculated
```

```
        UINT uGradeBmpLineByte = (lWidth + 3) / 4 * 4;
          DWORD dwGradeBmpDataSize = uGradeBmpLineByte * lHeight;
          DWORD dwGradeBmpSize = sizeof(BITMAPINFOHEADER) + sizeof(RGBQUAD) * 256 + dwGradeBmpDataSize;
          //Allocate space for grayscale bitmap and initialize to 0
          LPBYTE lpGradeBmp = (LPBYTE)new BYTE[dwGradeBmpSize];
          memset(lpGradeBmp, 0, dwGradeBmpSize);
          //Set grayscale bitmap color table
          LPRGBQUAD lpGradeBmpRgbQuad = (LPRGBQUAD)(lpGradeBmp + sizeof(BITMAPINFOHEADER));
           //Initialize the palette of 8 bit grayscale images
           LPRGBQUAD lpRgbQuad;
           for(int k = 0; k < 256; k++)
          {
            lpRgbQuad = (LPRGBQUAD)(lpGradeBmpRgbQuad + k);
            lpRgbQuad->rgbBlue = k;
            lpRgbQuad->rgbGreen = k;
            lpRgbQuad->rgbRed = k;
            lpRgbQuad->rgbReserved = 0;
          }
          // Grayscale bitmap data processing
          BYTE r, g, b;
          LPBYTE lpGradeBmpData = (LPBYTE)(lpGradeBmp + sizeof(BITMAPINFOHEADER) + sizeof(RGBQUAD) * 256);
          // Color conversion
          for(int i = 0; i < lHeight; i++)
          {
            for(int j = 0; j < lWidth; j++)
            {
              b = m_lpData[i * uLineByte + 3 * j];
              g = m_lpData[i * uLineByte + 3 * j + 1];
              r = m_lpData[i * uLineByte + 3 * j + 2];
              lpGradeBmpData[i * uGradeBmpLineByte + j] = (BYTE)(0.299 * r + 0.587 * g + 0.114 * b);
            }
          }
      }
```

C. Image Enhancement

The image gray-scale processing, and then image enhancement processing. Histogram equalization is widely used in image enhancement processing, and it can produce an image that distribution of gray level possesses uniform probability density, which expands the dynamic range of pixel values. The basic idea of the histogram equalization method is to expand gray level with more the number of pixels in the image, and to reduce gray level with less the number of pixels, so as to achieve the purpose of clear images. Histogram equalization is very useful for converting images into a consistent format before image comparison and segmentation. Histogram describes the gray level and reflects status of the image. This method can be used to improve the quality of the image, which is clearer and much more lively than the original image.

D. Image Segmentation

The image enhancement processing, and then image segmentation. Image segmentation is to divide the image into several regions, and uses a variety of algorithms to determine the region of interest.

From the type of image, image segmentation types include gray image segmentation, color image segmentation and texture image segmentation. According to the definition of image segmentation, image segmentation algorithm can be divided into two categories. One is image segmentation algorithm based on the region, which takes advantage of regional similarity. The other is image segmentation algorithm based on the boundary, which makes use of regional discontinuity. According to the segmentation mechanism of algorithms, including watershed method, threshold method, edge detection, etc.

Threshold method is one of the most commonly segmentation algorithms based on region segmentation algorithm, which is based on a certain criterion to automatically find the optimal threshold, and then the pixels are classified according to the gray level to achieve segmentation. It can be seen that the method can well separate the background and the target area. However, this method's computation is relatively large, and is the long running time. Therefore, using OpenCV conducts treating processes, thereby to reduce the amount of calculation.

Selecting and determining the threshold is key for image segmentation. This paper uses Otsu algorithm in threshold method, and we improve Otsu algorithm to obtain the optimal threshold. And according to the best threshold value, the image is divided into two areas of the target and background. So the fingerprint information is extracted.

What is said above, the image is divided into two parts by T: more than T pixel group and less than T pixel group. This is the most special method for studying gray level transformation, which is called the binarization of the image. The value is to divide the image into two fields of the target and background.

Transform function expression:

$$f(x) = \begin{cases} 0 \text{ or } 255 & x < T \\ 255 \text{ or } 0 & x > T \end{cases}$$

In the environment of OpenCV function library, the OTSU algorithm is used to extract threshold of the fingerprint image. The formula is as follow:

$$g = w_0 * pow((u - u_0),2) + w_1 * pow((u - u_1),2) \qquad (2)$$

In the formula, w_0 represents the proportion of the background pixels to the whole image; u_0 represents average gray of w_0 ; w_1 represents the proportion of the

target pixels to the whole image; u_1 represents average gray of w_1 ; u represents the average gray level for the whole image.

Concrete realization method: The histogram after grey level transformation is calculated, and cycling all the gray value of the histogram ranges from 0 to 255, and each gray value regards as a threshold. The threshold is used to segment the image into two groups. Average and variance of each group is calculated. When a gray value is divided between the two groups, the maximum gray value is the best threshold value of the two groups.

Programming the algorithm based on the Vc++6.0 integrated development environment platform is the use of the CVAdaptiveThreshold () function that is provided by OpenCV. Computer CPU is Intel-CORE-i5. Pictures use BMP format that information is rich and structure is simple. The following is the main body of this algorithm.

```
void CVAdaptiveThreshold::OtusThreshold(void)
{
   int i, j;
   LPBYTE p_data;
   p_data = m_pDib->GetData();
   // The number of bytes per pixel in an image
   int nLineByte = m_pDib->GetLineByte();
  // Image width
   int nWidth = m_pDib->GetWidth();
   // Image height
   int nHeight = m_pDib->GetHeight();
  // Gray histogram array, and initialization
   int nGrayHistogram[256];
   memset(nGrayHistogram,0,sizeof(nGrayHistogram));
  // Counting every gray level corresponding to the number of pixels, and storing in the gray histogram array
   int nPixel;
    for (j = 0; j < nHeight; j ++)
       for (i = 0; i < nWidth; i ++)
       {
            // Get the gray value of the current pixel
            nPixel = p_data[nLineByte * j + i];
           // Counting amount of gray value
            nGrayHistogram[nPixel] ++;
       }
  // Mean of the whole histogram; mean of C0 group and Mean of C1 group; the probability of C0 group and C1 group; variance and maximum variance
   float u, u0, u1, w0, w1, fVaria, fMaxVaria;
   // Total number of pixels in C0 group; threshold and optimal threshold
   int nCount0, nT, nBestT;
   // Counting histogram of the total number of pixels, and storing in the nSum
   int nSum=0;
   for(i = 0; i < 256; i ++)
      nSum += nGrayHistogram[i];
   // Enable threshold nT to traverse from 0 to 255
   for(nT = 0; nT < 256; nT ++)
   {
      // Mean of the whole histogram
      u = 0;
      nCount = 0;
      for(i = 0; i <256; i++)
      {
         u += i * nGrayHistogram[i];
         nCount += nGrayHistogram[i];
      }
      u /= nCount;
      // When the threshold is nT, the mean value and probability of C0 group are calculated
      u0 = 0;
      nCount0 = 0;
      for(i = 0; i <= nT; i++)
      {
         u0 += i * nGrayHistogram[i];
         nCount0 += nGrayHistogram[i];
      }
      u0 /= nCount0;
      w0 = (float) nCount0 / nSum;
      // When the threshold is nT, the mean value and probability of C1 group are calculated
      u1 = 0;
      for(i = nT+1; i < 256; i ++)
      {
          u1 += i * nGrayHistogram[i];
      }
      u1 /= (nSum - nCount0);
      w1 = 1 - w0;
      // Calculate the variance between the two groups
      fVaria = w0 * (u - u0) ^2 + w1 * (u - u1)^2;
      // Maximum variance and optimal threshold
      if(fVaria > fMaxVaria)
      {
         fMaxVaria = fVaria;
         nBestT = nT;
      }
   }
   // The optimal threshold is used to segment the original image
   for(j = 0; j < nHeight; j ++)
      for(i = 0; i < nWidth; i ++)
      {
         if(p_data[j * nLineByte + i] < nBestT)
            p_data[j * nLineByte + i] = 0;
         else
            p_data[j * nLineByte + i] = 255;
      }
}
```

IV. Simulation Results and Analysis

Based on the OpenCV function library, simulation experiments run in the Vc++ software as shown in Figure 2-Figure 5. Among them, Figure 2 and Figure 3 are a set of figures, Figure 4 and Figure 5 are a set of figures. In a dark

environment, image is collected as shown in Figure 2. From Figure 3, we can still be seen clearly extracted the details of the original fingerprint image, and the edge is more clear, and the background noise is small, and it is close to the original fingerprint. On the whole, the effect is very good. But it exists some highlighted points at some places, and then it needs to be improved. The original fingerprint collected is not very clear as shown in Figure 4. But it passes through image segmentation using the Otsu method improved, it can be seen from Figure 5 that the extracted fingerprint image can extract the fingerprint line primely, and keep the local details, and be close to the original fingerprint. In short, the effect is good.

Figure 2. Fingerprint image.

Figure 3. Fingerprint extraction graph.

Figure 4. Fingerprint image.

Figure 5. Fingerprint extraction graph.

V. CONCLUSION

Algorithm of fingerprint extraction based on the OpenCV function library has been proposed. The simulation results show that compared with the ordinary Otsu image segmentation, this algorithm can make use of the gray information of the image itself to select the optimal threshold commendably. And fingerprint lines are clearer, and the background noise is small. What's more, it can well preserve the details of the image, and segmented image is close to the real image. In a word, the fingerprint extraction comes true. And using the OpenCV function library to write the code is simple and efficient. However, there are some deficiencies in the system, such as the line where there are prominent points. To study the effect of enhancement process is the next step to solve the problem.

REFERENCES

[1] Hu Chunfeng, "The Fingerprint Feature Extraction and Matching," National University of Defense Technology, 2013.

[2] Thi Hanh Nguyena, Yi Wanga, Renfa Lia, "An improved ridge features extraction algorithm for distorted fingerprints matching," Journal of Information Security and Applications, 18(4), pp. 206-214, 2013.

[3] H Choi, K Choi, J Kim, "Fingerprint Matching Incorporating Ridge Features With Minutiae," IEEE Transactions on Information Forensics & Security, 6(2), pp. 338-345,2011.

[4] Kai Cao, Xin Yang, Xunqiang Tao, Peng Li, Yali Zang, Jie Tian, "Combining features for distorted fingerprint matching," Journal of Network and Computer Applications, 33(3), pp.258-267, 2010.

[5] Xue Shengli, Cai Qizhong, Yang Hailin, Xu Xiaoyu, "The algorithm of train ticket identification based on OpenCV," Journal of Guangxi University of Science and Technology, 27, pp.46-51, 2016.

[6] Bradski G, Kaehler A, "Learning OpenCV: Computer vision with the OpenCV library," [S.l.]:O'Reilly Media, Inc, 2008.

[7] Wen HuanWu, Ying JunZhao, Yong FeiChe, "Research and Implementation of Face Detection Based on OpenCV," Advanced Materials Research, 971, pp.1710-1713, 2014.

[8] L Zhang, MY Yin, W Li, HL Liu, "Implementation of Camera Calibration Method Based on OpenCV," Applied Mechanics & Materials, 602, pp.3796-3799, 2014.

[9] Chen Kai, Song Ailing, Liu Yuanfu, Li Luoji, Li Zheng, "Detection and tracking of table tennis in table tennis match based on OpenCV Technology, " CACSS. SPIE, pp.13-18, 2013.

[10] Zhang Xiaoyu, Peng Siwei, "Moving Object Recognition Algorithm and Implement Based on OpenCV," Modern Electronics Technique, 22, pp.99-101, 2009.

[11] Bu Wenbin, You Fucheng, Li Quan, Wang Huihua, Duan Huaifen, "An Improved Image Segmentation Method Based on Otsu," Journal of Beijing Institute of Graphic Communication, 23(4), pp.76-78, 2015.

[12] Yuan Xinzhi, Jiang Hong, Chen Yunzhi, Wang Xiaoqin, "Extraction of Water Body Information Using Adaptive Threshold Value and OTSU Algorithm," Remote Sensing Information, 31(5), pp.36-42, 2016.

[13] Wang Rong, Hou Pengpeng, Zeng Zhaolong, "The Application of Face Detection and Tracking Method Based on OpenCV," Science Technology and Engineering, 24(14), pp.115-118, 2014.

[14] Bian Weixin, Ding Shifei, Xue Yu, "Combining weighted linear project analysis with orientation diffusion for ingerprint orientation field reconstruction," Information Science, 396, pp.55-71, 2017.

[15] Zhou Dapeng, "An Variable Coefficient Images Denoising Method," ICMIA 2013, 333-335, pp.832-835, 2013.

[16] Zeng Wanmei, Wu Qingxian, Jiang Changsheng, "A New Adaptive Threshold Method for Target Image Segmentation," Electronics Optics & Control, 16(5), pp.27-29, 2009.