International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

A Multi-spectral Component Extraction Approach Using Radial Basis Neural Network

Shasha Yu, Wenwei Wang, Meijuan Li

School of Electronic Information
Wuhan University, 129 Luoyun Road, Wuhan, Hubei 430079, P.R. China

Abstract: This paper presents an approach based on radial basis function neural network to extract multi-spectral components from the true-color image. According to the color composition principle of chromatics, the multi-spectral components can be approximated from the three primitive components (RGB) of the true-color image. The radial basis function neural network is used as the multivariate interpolator. To train the kernel function centers of the radial basis function neural network, a method to combine the nearest cluster algorithm and the dynamic mean cluster algorithm is implemented. Experiments have been carried out and the results show that the virtual multi-spectral images created by the proposed network are consistent with the real multi-spectral images acquired by hardware equipments.

Keywords: Radial basis function neural network; Multi-spectral image; Color image; Chromatics

1. Introduction

Multi-spectral images, which can provide more information than color images and grayscale images, have been widely applied in remote sensing domain. In recent years, researchers have successfully applied the multi-spectral imaging technology to the medical diagnosis domain. However, it needs large and expensive hardware equipments to obtain the multi-spectral images [1,2], such as precise light filter, which mostly limits the promotion and application of the multi-spectral imaging technology. Therefore, how to obtain multi-spectral images in a cheap, convenient and rapid way will be the research hotspot in the future.

With the development of computer technology, it is in high demand to design a software light filter, which is of low cost, high precision and high efficiency. The implementation of this idea can bring the following advantages:

- Reduce the cost and complexity of extracting multi-spectral information: Using software light filter which is based on the computer technology to replace the complicated hardware equipment, such as the traditional expensive light filter.
- Extract multi-spectral images in demand: The kernel of the software light filter is the extracting algorithm and it's easy to be implemented in general purposed computers.
- Easy for storage and transmit of information: A multi-spectral image is composed of a sequence of grayscale images, the volume needed to store the n ($n \geq 3$) wave bands multi-spectral image is n/3 times more than the color image. It needs large bandwidth to transmit multi-spectral images in the network. If we have successfully devised the software light filter, senders can transmit color images in the network while receivers can use the software light filter to extract the multi- spectral images at anytime.

The three-primitive-color theory and color composition principle of chromatics point out that any visible color can be produced by commixing the primitive Red, Green and Blue components at a

specific proportion. This paper deals with the implementation of extracting multi-spectral images from normal true-color images composed of RGB components. Instead of using an expensive hardware light filter and corresponding cameras, we propose a software light filter approach, which uses a radial basis function neural network (RBFNN) as the multivariate interpolator, to extract multi-spectral images from color images. The multi-spectral images extracted by the hardware equipment are named as *real multi-spectral images* and the ones extracted by the software method are named as *virtual multi-spectral images*. The experimental results show that using the RBFNN to extract multi-spectral images form color images is effective.

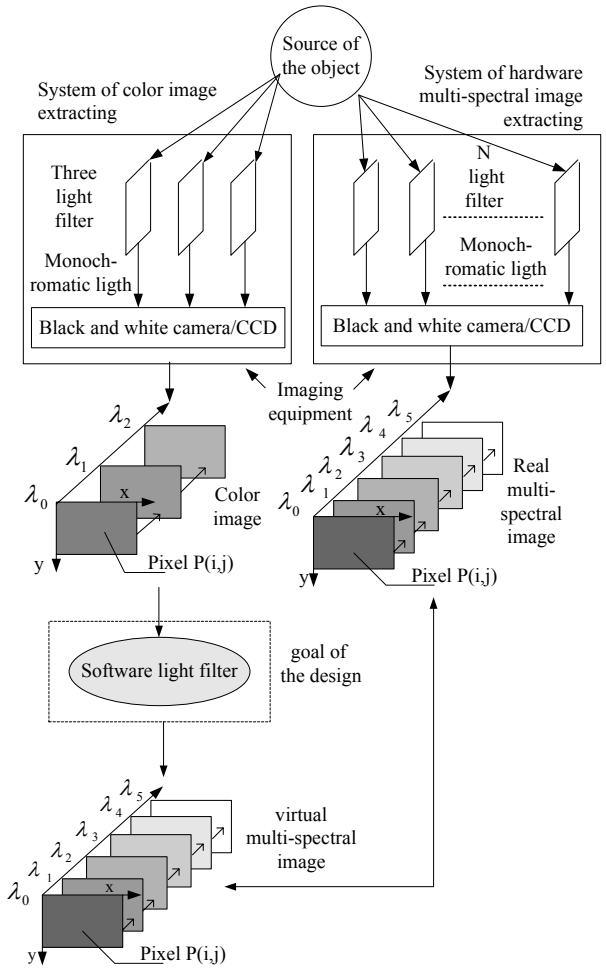


Figure.1 The schema of virtual light filter.

2. Procedures of extracting virtual multi-spectral images

The procedures of extracting virtual multi-spectral image are shown in Figure 1. The kernel of the whole flow is the design of the software light filter model, which has three-input and multi-output function, and its main function is to extract the virtual multi-spectral image from the color image and approximate the real multi-spectral image.

3. Principle of software light filter

The wave length of the three standard primitive colors prescribed by CIE is 700 nm (red), 546.1nm (green) and 435.8nm (blue). The spectrum curve of tristimulus values by matching three standard primitive colors is shown in Figure 2, where y-axis denotes tristimulus values by matching voluntariness colors and the x-axis denotes the wave length.

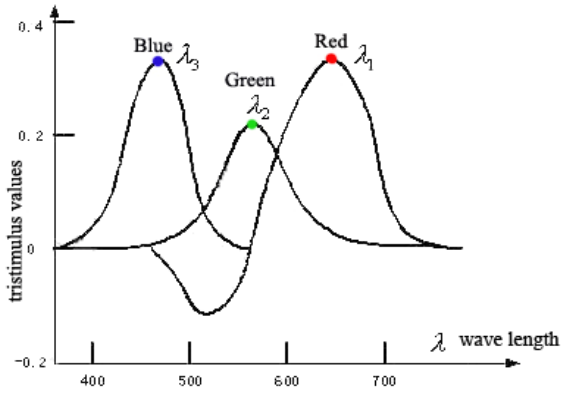


Figure.2 Spectrum curve of tristimulus values by matching three standard primitive colors.

The three-primitive-color theory and color composition principle of chromatics point out that any color in the visible light can be produced if we commix the R, G and B at a specific proportion, thus the match of the light color can be described by the following equation:

$$c = rR + gG + bB \tag{1}$$

Where *r*, *g* and *b* stand for the relative quantity of R, G and B respectively, and they are also known as tristimulus values, *c* is the real-color image.

And then, extracting virtual multi-spectral images from color images can be regarded as get other values in the curve from the three already knew values, which means *c* equals to *c'* as it's shown in equation (2) and (3), where *c'* is the multi-spectral image, C_1, C_2……C_n denote the wavelength, *n* is the number of the wave bands, in the real-color image, *n* equals to three, that is R, G and B. The mathematics essence of the extracting approach is function approximation [3] and multivariate interpolation [4,5].

$$c' = w_1 C_1 + w_2 C_2 +, ..., + w_n C_n \tag{2}$$

$$w_n = \mathrm{F}(r, g, b) \qquad (3)$$

Here w_n denotes the weight that we need to figure out. Our target is to find a function **F** to calculate w_n from tristimulus values. Generally, the relationship of tristimulus values and w_n is not linear, thus we devise a RBFNN to simulate the function **F**.

4. RBF neural network

4.1. Principium of the RBF neural network

The essence of neural network is nonlinear so that it can give a uniform mathematic model to any nonlinear problems which are hard to be described by a regular (or accurate) mathematic model. Neural network has been widely applied in function approximation algorithm in recent years [6]. The RBF neural network is a multi-layers feed-forward model consisting of input layer, hidden layer and output layer. The kernel layer provides an array of nonlinear radial basis functions, which are usually selected with Gaussian functions [7,8]. The structure of the neural network is shown in Figure 3.

Thus, RBFNN can be defined as:

$$y_j = \mathrm{f}(\boldsymbol{x}) = \sum_{i=1}^{l} w_{ij} \Phi(\|\mathbf{x} - \mathbf{c}_i\|) + \text{bias} \quad (4)$$

In which, Φ is called as the radial basis function. In this paper, we choose the Gaussian function:

$$\Phi(\mathbf{x}) = \exp(-\mathbf{x}^2 / 2\sigma^2) \qquad (5)$$

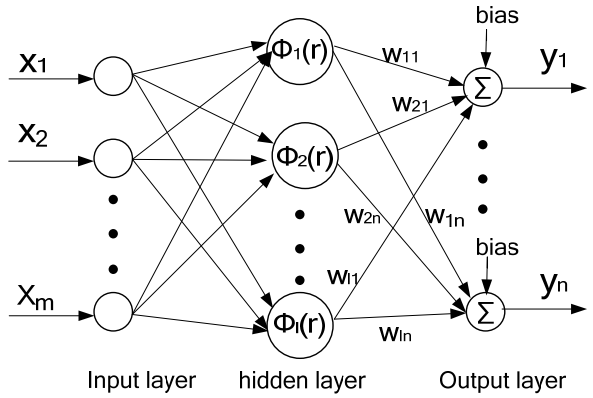


Figure.3 Structure of RBFNN.

Where $\mathbf{c}_i$ represents the vector related to the center of the basis function which is associated with the kernel unit *i*, w_{ij} is the weight from the *i*th kernel unit to the *j*th output unit. In the output layer, values of the units are decided via a linear combination of the nonlinear outputs from the hidden layer. In this paper, *r*, *g*, *b* is the input $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, while the output $y_1, y_2 \ldots y_n$ denote the $w_1, w_2, \ldots w_n$ in the equation (3).

4.2. Design of the RBF neural network

4.2.1. Preprocessing of the input data

The input of the RBF neural network in this paper is the value of R, G and B of a single pixel, thus the number of the input node m in Figure 3 is three. If there are big differences among the components of the input vectors, the learning accuracy and speed of the network will come down. In order to obtain good astringency when training samples, it needs to normalize the sample data.

Here we suppose that $\mathbf{X}^f$ is the input set:

$$\mathbf{X}^f = [\mathbf{x}_1^f, \mathbf{x}_2^f, \cdots, \mathbf{x}_k^f, \cdots, \mathbf{x}_v^f] \qquad (6)$$

Where $\mathbf{x}_k^f = (x_{1k}^f, x_{2k}^f, \cdots, x_{mk}^f)^T$ denotes the row vector of the *k*th input sample. There are *v* samples in total (*k=1，2,...,v*) and *v* input vectors make up the input matrix $\mathbf{X}^f$.

After mapping (preprocessing), the input matrix of the training sample set is:

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k, \cdots, \mathbf{x}_v] \qquad (7)$$

Where $\mathbf{x}_k = (x_{1k}, x_{2k}, \cdots, x_{mk})^T$ denotes the row vector of the *k*th input sample after mapping (*k=1，2,...,v*) and *v* input vectors make up the input matrix $\boldsymbol{X}$.

The target of normalization is to map the component data of the input sample vectors to [0, 1], that is:

$$x_{jk} \in [0,1] \quad j = 1, 2, \cdots, m;\ k = 1, 2, \cdots, v \quad (8)$$

Here x_{jk} is the *j*th component value of the *k*th input sample after mapping.

The mapping operator for $\boldsymbol{X}^f \rightarrow \boldsymbol{X}$ is:

$$x_{jk} = \frac{x_{jk}^f - x_{\min}^f}{x_{\max}^f - x_{\min}^f} \qquad (9)$$

$$\left\{ \begin{array}{l} x_{\min}^f = \min(x_{j1}^f, x_{j2}^f, \cdots, x_{jk}^f, \cdots, x_{jv}^f) \\ k = 1, 2, \cdots, v \\ x_{\max}^f = \max(x_{j1}^f, x_{j2}^f, \cdots, x_{jk}^f, \cdots, x_{jv}^f) \\ j = 1, 2, \cdots, m \end{array} \right\} (10)$$

In our method, we set $x_{\min}^f$ and $x_{\max}^f$ of all the samples as follows:

$$x_{\min}^f = 0, \qquad x_{\max}^f = 255 \qquad (11)$$

4.2.2. Training process of the RBF neural network

There are three main learning parameters of RBFNN: center, variance and weight, where the center and variance are defined in the hidden layer. The structure of the network and the ability of classifying will be totally different if the number of hidden layer units is different. Therefore, it's important to choose an appropriate number of hidden layer units, centers, variances and the kernel function.

In this paper, we present a methodology, which associated with the nearest cluster algorithms [9] and dynamic mean cluster algorithms, to train the center of the hidden layer units and the initial weight from the hidden layer units to the output layer units. This is an auto-adapt no intendance learning algorithm which needn't have to define the number of the hidden layer units in advance.

The details of the algorithm are shown below:

Step 1: Choose an appropriate distance threshold r; define a vector $\boldsymbol{A}$ to store the sum of actual output vectors which belong to every center; define a vector $\boldsymbol{B}$ to count the samples belonging to every hidden layer center.

Step 2: Start with the first data couple (x_1, y_1), set a center based on x_1, set $c_1 = x_1$, $\boldsymbol{A}(1) = y_1$, $\boldsymbol{B}(1) = 1$. Now there is only one hidden unit in this RBF network, and c_1 is the center of this hidden unit, $w_1 = \boldsymbol{A}(1)/\boldsymbol{B}(1)$ is the weight from the hidden unit to the output unit.

Step 3: Think about the second data couple (x_2, y_2), figure out the distance from x_2 to this clustering center, namely $\|x_2-c_1\|$.

If $\|x_2-c_1\| \le r$, c_1 is the closest clustering, set $\boldsymbol{A}(1)=y_1+y_2$, $\boldsymbol{B}(1)=2$, $w_1=\boldsymbol{A}(1)/\boldsymbol{B}(1)$; if $\|x_2-c_1\| > r$, set x_2 as a new clustering center, $c_2=x_2$, $\boldsymbol{A}(2)=y_2$, $\boldsymbol{B}(2)=1$, and add a new hidden unit to the RBF network that has set up and set $w_2=\boldsymbol{A}(2)/\boldsymbol{B}(2)$ as the weight from this new hidden unit to the output.

Step 4: Suppose now we are thinking about the kth data couple (x_k, y_k), where $k=3, 4,\ldots,N$ and there are M clustering centers, the centers are $c_1, c_2,\ldots,c_M$ and there are M hidden units in the RBF network. Now we figure out the distance from x_k to each center, namely $\|x_k-c_i\|$, i=1, 2,…, M. If $\|x_k-c_j\|$ is the minimum distance, we set c_j as the closest clustering of x_k.

(1) If $\|x_k-c_j\|>r$, set x_k as a new clustering center and set $c_m+1=x_k$, $M=M+1$, $\boldsymbol{A}(M)=y_k$, $\boldsymbol{B}(M)=1$, keep the value of $\boldsymbol{A}(i)$, $\boldsymbol{B}(i)$ no change, where $i=1,2,\ldots,M-1$ and add a new hidden unit to the RBF network.

(2) If $\|x_k-c_j\| \le r$, that is to say the input vector is in the error allowance range and belongs to the jth clustering. We apply the idea of k-means clustering principle , figure out the mean of the whole member vectors to update the center, namely

$$c_j = \frac{1}{N_{S_j}+1}\sum_{x\in S_j} X_i \qquad (12)$$

where S_j is the sum of the total input vectors of jth clustering, $x_k \in S_j$. Set $\boldsymbol{A}(j)=\boldsymbol{A}(j)+y_k$, $\boldsymbol{B}(j)=\boldsymbol{B}(j)+1$. If $i\neq j$, $i=1,2,\ldots,M$, keep the value of $\boldsymbol{A}(i)$, $\boldsymbol{B}(i)$ no change, and set $w(i)=\boldsymbol{A}(i)/\boldsymbol{B}(i)$ as the weight from hidden unit to output, where $i=1,2,\ldots,M$.

The flow chart of the algorithm is shown in Figure 4. The value of the distance threshold r decides the complexity degree of the dynamic auto-adaptive RBF network. The smaller the value of r is, the larger the number of the hidden layer units is, and also the amount of calculation.

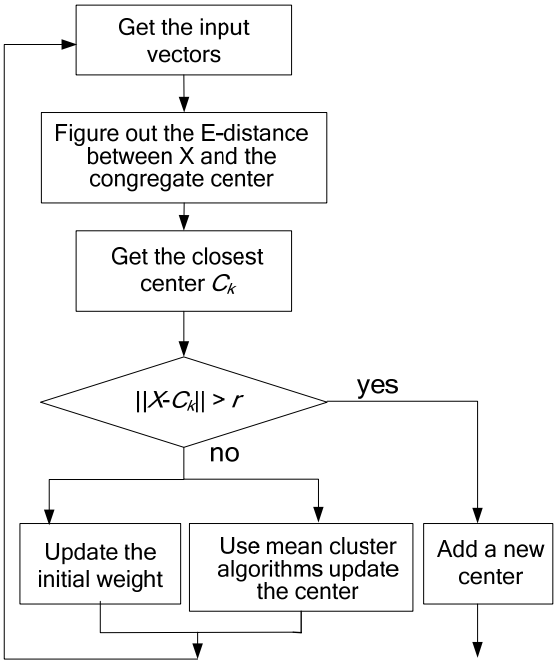
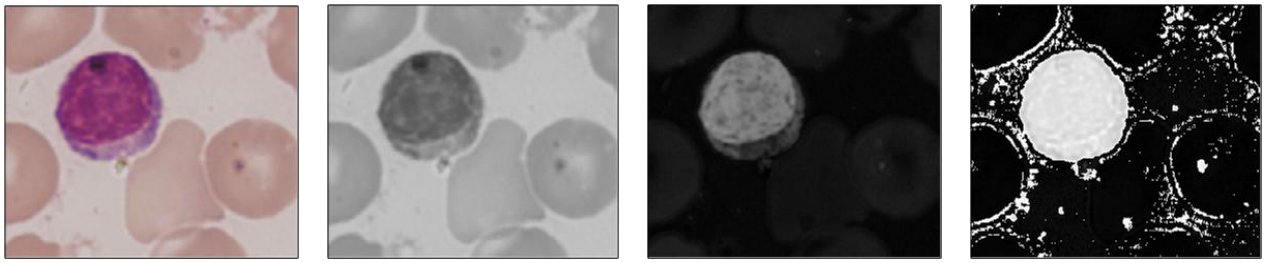


Figure 4. RBF center Training.

For r is a one-dimensional parameter, we can usually find an appropriate r by experiments and error information. As each input-output data couple may produce a new clustering, this dynamic auto-adaptive RBF network actually adjusts the parameter and the structure at the same time. Here we use the LMS (Least Mean Square) algorithm to adjust the weight of the output.



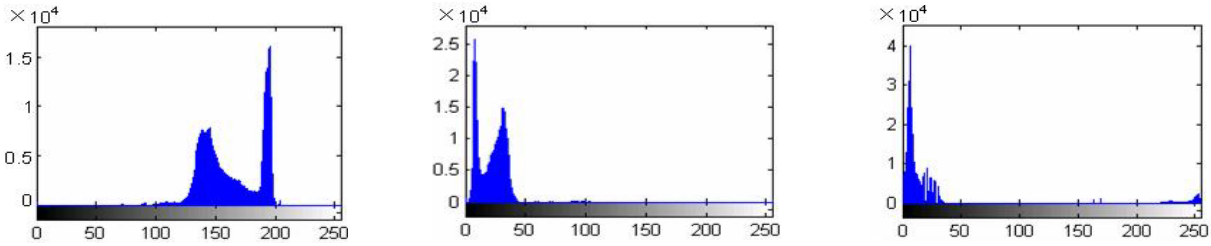
(a) The original white cell color image in the RGB space.

(b) The I component of the original color image.

(c) The S component of the original color image.

(d) The H component of the original color image.

Figure 5. Transformation from the RGB space into the HSI space.



(a) The histogram of the component I in the HSI space.

(b) The histogram of the component S in the HSI space.

(c) The histogram of the component H in the HSI space.

Figure 6. Histogram of the original white cell color image in the HSI space.

5. Experiments

5.1. Preprocessing of the color images

In this paper, we use white cell color images as the samples. In order to get a high compute speed and raise the quality of the multi-spectral images extracted, it need to take preprocessing to the true-color images of white cells. At first, we transfer the image in the RGB space into the HSI space, after which the information is more compact, the components are more independent and the color information is less lost. And then, we apply a threshold method to segment the image to remove background, noise and the red cells. The transform equation from RGB space to HSI space is shown bellow [10]:

$$I = \frac{(R+G+B)}{3} \qquad (13)$$

$$S = 1 - \frac{\min(R,G,B)}{I} \qquad (14)$$

$$H = \begin{cases} \varphi & G \geq B \\ 2\pi - \varphi & G < B \end{cases} \qquad (15)$$

$$\varphi = \cos^{-1} \frac{1/2[(R-G)+(R-B)]}{\sqrt{(R-G)^2 + (R-B)(G-B)}} \qquad (16)$$

The result of the transformation is shown in Figure 5.

According to the histogram of the I, S, H component in Figure 6, we find that the intensity (I) in the background is larger than others evidently and the hue (H) between white cells and red cells has conspicuous differences. Thus we use this information to remove most of the background and red cells. The image after preprocessing is shown below:

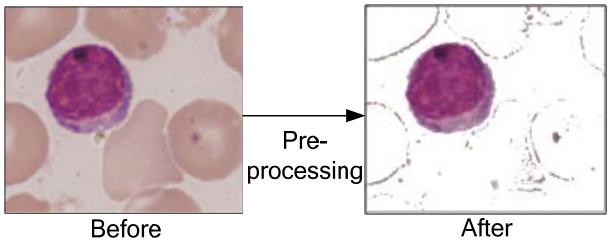


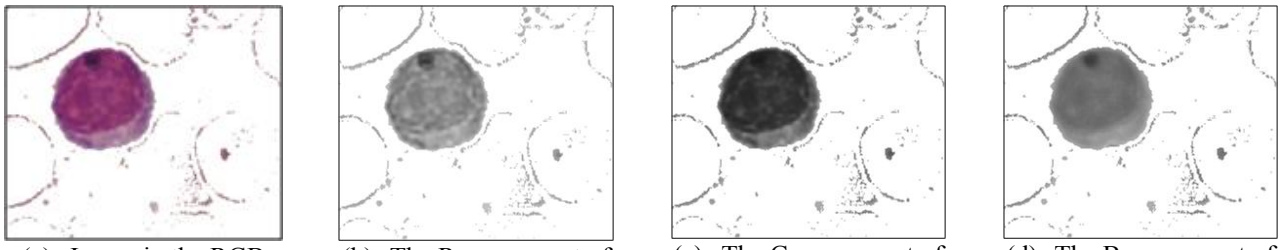
Figure 7. The white cell color image after preprocessing.

5.2. Implementing the multi-spectral component extraction

In this paper, we use the method discussed above to design the RBFNN, accordingly to realize the

function of software light filter. The images in the experiment are all after preprocessing. The number of input units of the RBFNN is three and the output is twelve. We use the Gauss function as the hidden layer function; associate the nearest cluster algorithms and dynamic mean cluster algorithms to train the center of the hidden layer unit; set an appropriate distance threshold r=0.1; apply the LMS to figure out the weight of the output layer and set the mean squared deviation σ=1.

Here we use sixty white cell multi-spectral images as the experiment samples and choose fifteen representational images of them as training samples while the rest as testing samples. The twelve wave bands of the extracted multi-spectral images are 440, 450, 480, 490, 500, 520, 530, 540, 550, 580, 670 and 700nm. Figure 8 is the input of the RBFNN, (a) is the real-color image; (b), (c) and (d) are the grayscale images of R, G, B components of the color image. The three input units denote the value of the three wavebands for each pixel respectively. Figure 9 shows the result of extracting.

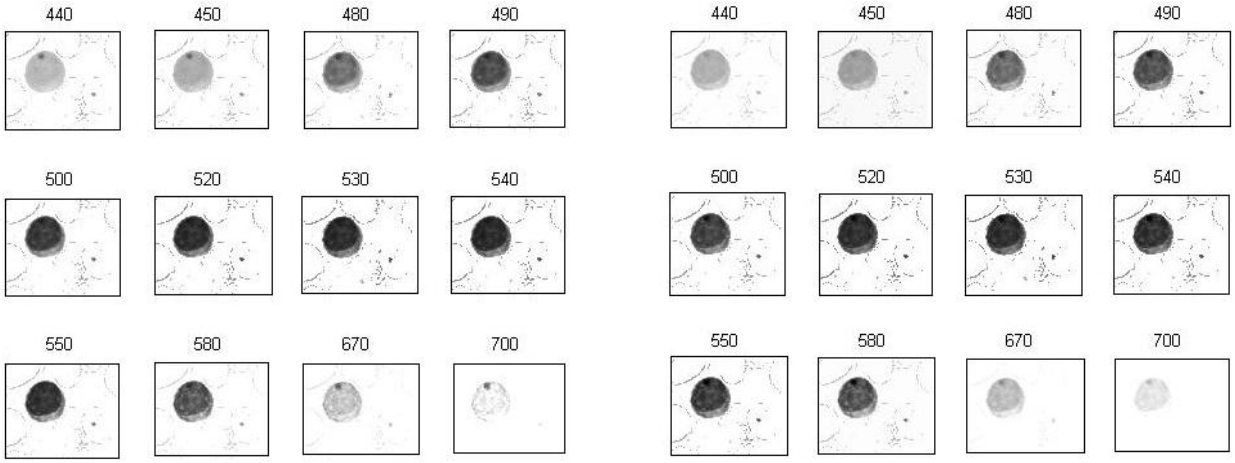


(a) Image in the RGB space. (b) The R component of the color image. (c) The G component of the color image. (d) The B component of the color image.

Figure 8. The color image and its R, G, B components.

From the Figure 9 (a) and (b), we can find that there are few differences between the real multi-spectral images and the virtual multi-spectral images. Set the real multi-spectral images extracted by hardware light filter as the statement, Figure10 explains the distribution of relative error for every pixel of every wave band in the real and virtual multi-spectral images.

As the results present, though the multi-spectral images extracted by the two methods have grayscale differences at a certain extent, most of pixels have few differences, 88.538% of the total pixels' error is between -5%~+5%, 94.046% is between -10%~+10% and 97.086% is between -15%~+15%. After preprocessing there mainly leukocyte karyon, leukocyte plasm and background exist in the cell image, so there are three curves in the spectrum curve chart. As it's shown in Figure11, virtual multi-spectral curve approaches to the real multi-spectral curve at a high degree. The shapes of the two curves are similitude.



(a) Real multi-spectral images extracted by hardware light filter. (b) Virtual multi-spectral images extracted by RBFNN software fight filter.

Figure 9. Contrast of real multi-spectral image and the virtual ones.

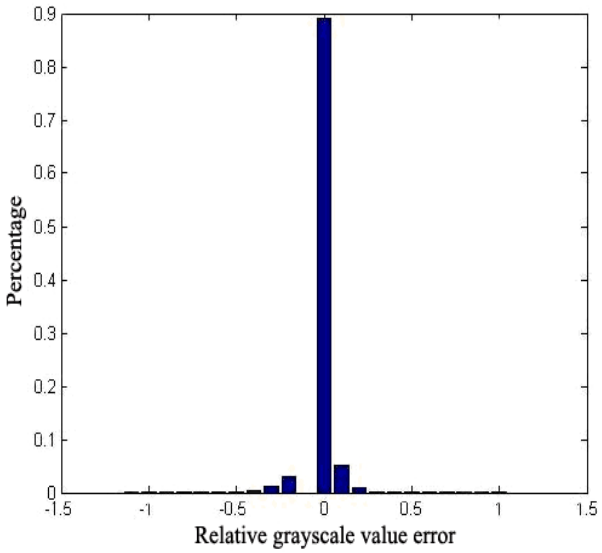


Figure 10. Relative gray error statistical comparison

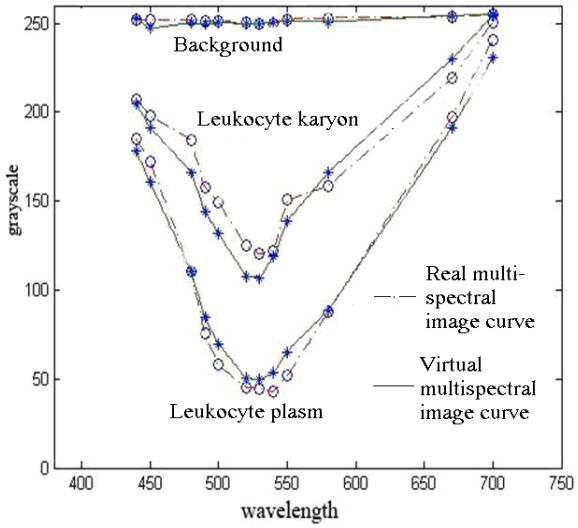


Figure 11. Comparison of spectrum curve of real multi-spectral images and the virtual ones.

6. Conclusion

This paper proposes an original method to extract multi-spectral image based on RBFNN, and then the nearest cluster algorithms and dynamic mean cluster algorithms are combined to train the hidden layer centers and define the weights from the hidden layer units to the output units. It is the first time to realize the extraction of multi-spectral images with software. At the same time, it solves some problems of hardware equipment, such as high cost, hard to operate and large size. It's also easy for multi-spectral images to store, process, analyze and transmit. The experiment results illustrate that the present approach is feasible.

References

[1] Xijian Gao, Libo Zeng, Qiongshui Wu and Diancheng Wang, "An auto-division method based on microgram of multi-spectral image of cervix cell" (in Chinese), *Data Collection and Procession*, Vol.19, No.4, pp.441-445, Apr.2004.

[2] Shulong Zhu, Zhanmu Zhang, "Obtain and Analysis of Remote Sensing Image" (in Chinese), *Science publishing company*, pp. 33-92, Apr. 2000.

[3] Park J, Sandberg IW, "Universal approximation using radial basis function networks", *Neural Computation*, 3: 246 -257, 1991.

[4] Broomhead DS, Lowe D, "Multivariable functional interpolation and adaptive networks", *Complex Systems*, 2:321-355, 1998.

[5] Powell MJD, "Radial basis function for multivariable interpolation", *Algorithms for Approximation*, 7:143-167, 1987.

[6] Simon Haykin, "Neural Network: A Comprehensive Foundation, 2nd Edition", New Jersey: Prentice-Hall/Pearson, pp.109-229, 1999.

[7] Hartman EJ, Keeler JD, Kowalski JM, "Layered neural networks with Gaussian hidden units as universal approximations", .*Neural Computation*, 2:210-215, 1990.

[8] Girosi F, Jones M, Poggio T, "Regularization theory and neural networks architectures", *AI Memo*, 1430: MIT, 1993.

[9] Wei Wang, Gengfeng Wu, Bofeng Zhang, Yuan Wang, "Radial Basis Function Neural Network and the Application" (in Chinese). *Earthquake*, Vol.25, No.2, pp.19-25, Apr.2005.

[10] Kim Won-Soon, Park Rae-Hong, "Color image palette construction based on the HSI color system for minimizing the reconstruction error", In *Proceeding of International Conference on Image Processing*, pp.1041-1044, Mar.1996.