Journal of Beijing Institute of Technology, 2021, Vol. 30, Suppl.1

Improved Chroma Resampling Algorithm Based on Interpolation Method

Shiwei Ren, Qiuchen Ji, Wei Gao and Yue Ma✉

(School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, China)

Abstract: An improved algorithm based on bilinear interpolation was proposed. It could deal with the shortcoming of chroma component information loss in the original video image chroma resampling algorithm. With the bilinear interpolation method, the chroma component matrix in the YCbCr color space was scaled in both the horizontal and vertical directions. It could reduce information loss while meeting the corresponding standard data format. The new algorithm could retain more information from the source image's color, which improved image recovery. In addition, the paper proposed a new format with a higher compression ratio on the basis of ensuring visual performance. Experimental results showed that, compared with the original chroma resampling algorithm, the processing result of the improved algorithm had a significant improvement in the performance of parameters such as composite peak signal noise ratio (CPSNR).

Key words: YCbCr color space; chroma resampling; interpolation algorithm; image compression

CLC number: TP391 **Document code:** A **Article ID:** 1004-0579(2021)S1-0011-07

Digital video technology is in a stage of rapid development and widespread usage. People need not only higher video image quality, but also lower data transmission bandwidth and smaller data storage space in digital video systems. Therefore, data compression is a key technology for digital video transmission and storage. The chroma resampling algorithm is a video data compression method based on the study of the human visual system, which can compress the amount of data while ensuring image quality[1].

The chroma resampling algorithm is a simple but effective video compression algorithm. It samples the chroma components of the YCbCr color space in different ways to get new resampling formats. In this way, it can compress or recover the volume of video data. According to the resampling rate, the original chroma resampling algorithm selects the data of the operating points during compression, and copies the adjacent data during recovery. A greater loss of chroma component information occurs during this process; image information suffers a great loss after multiple compressions and recoveries.

To deal with the above problem, this paper proposes an improved algorithm based on bilinear interpolation. The new algorithm processes the target pixels and related pixels around them with bilinear interpolation. This way, the improvement reduces information loss during the transformation process in the original algorithm. The optimized algorithm has better simulation parameter results compared with the original one.

1 Video Chroma Resampling Algorithm

The video chroma resampling algorithm is used to realize the conversion between various data formats in the YCbCr color space. Each pixel in the video image can be composed of three components Y, Cb, and Cr. Y is the luma component, Cb is the blue chroma component,

Received 2020-06-16
Supported by the National Natural Science Foundation of China (61801024)
✉ Author for correspondence, lecturer, Ph.D.
E-mail: bitmayue@bit.edu.cn
DOI: 10.15918/j.jbit1004-0579.20120

and Cr is the red chroma component. The human eye is more sensitive to luma information compared to chroma information according to research on the human visual system. That is the reason why video systems can sample Cb and Cr components at a lower rate. This will reduce the volume of data in video storage and transmission while keep similar image quality.

Data formats include YCbCr4:4:4, YCbCr4:2:2, YCbCr4:2:0, YCbCr4:1:1, and so on. We can resample the chroma component at different rates to perform the conversion between the formats above. The numbers in these data formats indicate the numbers of Y, Cb, and Cr components in 4 adjacent pixels in an image. For example, YCbCr4:2:2 indicates that there are 4 Y components, 2 Cb components, and 2 Cr components in 4 adjacent pixels. A smaller number means that the number of components is smaller, the sampling rate is lower, and the compression rate is higher.

The diagrams of 4 data formats above are shown in Fig.1. Each square indicates 1 pixel in the video image. The "Y", "Cb" and "Cr" indicate the sampling situation of each component in the pixel[2].

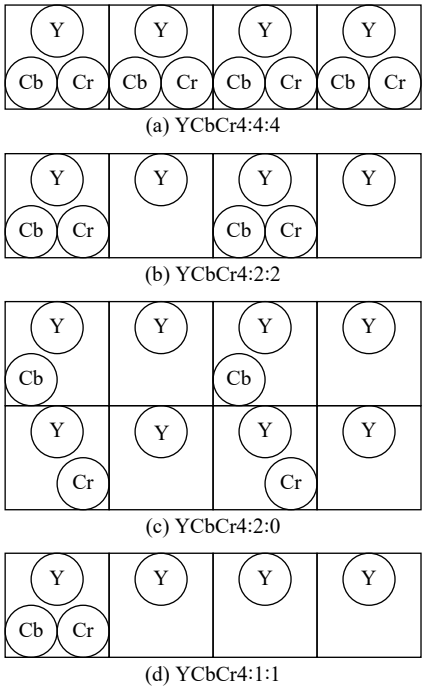

Fig. 1 Pixels diagrams of 4 commonly used YCbCr data formats

The RGB format is always used in image displaying. It can be converted to the YCbCr4:4:4 format[3]. (The conversion process is not described here.) Because of this, the chroma resampling algorithm is mostly used to perform conversions between YCbCr4:4:4 and the other formats. The specific algorithms are:

① Conversion between YCbCr4:4:4 and YCbCr4:2:2

During compression from YCbCr4:4:4 to YCbCr4:2:2, the original algorithm keeps the sampling rate of the luma component Y unchanged (the same is true of the following conversions, although omitted), and samples chroma components every 2 pixels in the horizontal direction. It only takes 1 Cb and 1 Cr component from 2 adjacent pixels in the same row, and discards the remaining chroma component data.

During recovery from YCbCr4:2:2 to YCbCr4:4:4, the original algorithm interpolates chroma components every 2 pixels in the horizontal direction. It copies the only Cb and Cr components from 2 adjacent pixels in the same row, and shares the data across the 2 pixels.

② Conversion between YCbCr4:4:4 and YCbCr4:2:0

During compression from YCbCr4:4:4 to YCbCr4:2:0, the original algorithm samples chroma components every 2 pixels in the horizontal direction, and then in the vertical direction. It only takes 1 Cb and 1 Cr component from each 2×2 section of adjacent pixels, and discards the remaining chroma component data.

During recovery from YCbCr4:2:0 to YCbCr4:4:4, the original algorithm interpolates chroma components every 2 pixels in the vertical direction, and then in the horizontal direction. It copies the only Cb and Cr components from each 2×2 section of adjacent pixels, and shares the data across the 4 pixels[4].

③Conversion between YCbCr4:4:4 and YCbCr4:1:1

During compressing from YCbCr4:4:4 to YCbCr4:1:1, the original algorithm samples chroma components every 4 pixels in the hori-

zontal direction. It only takes 1 Cb and 1 Cr component from 4 adjacent pixels in the same row, and discards the remaining chroma component data.

During recovery from YCbCr4:1:1 to YCbCr4:4:4, the original algorithm interpolates chroma components every 4 pixels in the horizontal direction. It copies the only Cb and Cr components from 4 adjacent pixels in the same row, and shares the data across the 4 pixels.

The diagrams of each conversion are shown in Fig.2.

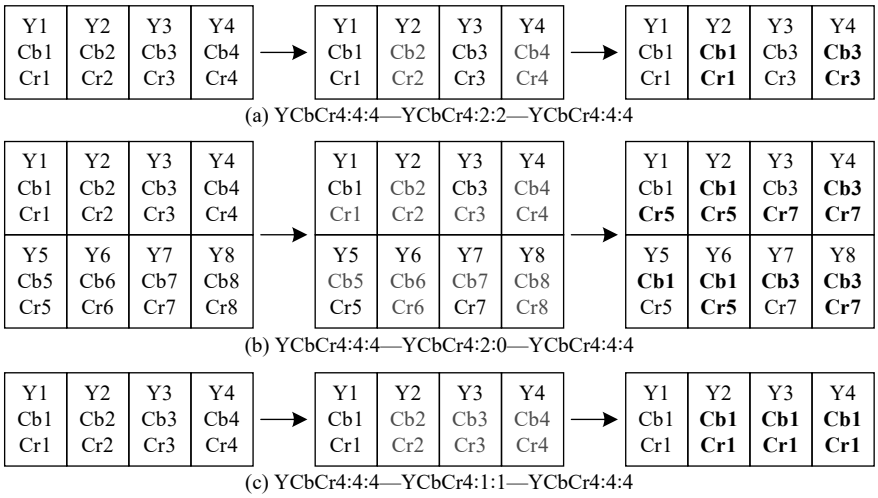

Fig. 2 Schematic diagrams of the data conversion with original chroma resampling algorithm

2 Interpolation Method and Principle

Bilinear interpolation is always used in image scaling. Each point P' to be inserted in the target image corresponds a position P in the source image. The value of point P' can be calculated with adjacent points around position P by an interpolation function[5].

Bilinear interpolation needs 4 adjacent pixels a_{00}, a_{01}, a_{10}, and a_{11} around the position P in the source image, as shown in Fig.3. We can do linear interpolation in the x and y directions with 4 surrounding adjacent pixels.

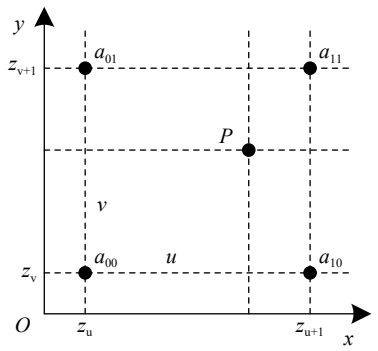

Fig. 3 Bilinear interpolation diagram

The size of the source image is $m \times n$, and the zoom factors are w and l. The size of the target image is $(w \times m) \times (l \times n)$. Assume the distance between 2 adjacent points in the source image is 1, and we can calculate the step-length ratios s_u and s_v in the x and y directions before and after scaling as

$$\begin{cases} s_u = \dfrac{m-1}{wm-1} \\ s_v = \dfrac{n-1}{ln-1} \end{cases} \tag{1}$$

Use $\boldsymbol{A}$ and $\boldsymbol{B}$ to indicate the source image matrix and the target image matrix, respectively. The point P' in the target image can be expressed as $\boldsymbol{B}(i, j)$. The coordinates of position P in source image can be expressed as $\boldsymbol{A}(z_u + u, z_v + v)$. z_u and z_v are the integers of the coordinates of P, which are equal to the coordinates of a_{00}. u and v indicate the decimals of the coordinates of P, which are equal to the vector distance between P and a_{00}. We can calculate these parameters as

$$\begin{cases} z_{\mathrm{u}} = \lfloor (i-1)\, s_{\mathrm{u}} + 1 \rfloor \\ u = (i-1)\, s_{\mathrm{u}} + 1 - z_{\mathrm{u}} \\ z_{\mathrm{v}} = \lfloor (j-1)\, s_{\mathrm{v}} + 1 \rfloor \\ v = (j-1)\, s_{\mathrm{v}} + 1 - z_{\mathrm{v}} \end{cases} \tag{2}$$

The value of position P can be calculated by linear interpolation in 2 directions as follows. The value of position P is equal to the value of $\boldsymbol{B}(i,j)$ in the target image[6].

$$\begin{aligned} \boldsymbol{B}(i,j) = & \boldsymbol{A}(z_{\mathrm{u}} + u, z_{\mathrm{v}} + v) = \\ & (1-u)(1-v)A(z_{\mathrm{u}}, z_{\mathrm{v}}) + (1-u)v\times \\ & A(z_{\mathrm{u}}, z_{\mathrm{v}} + 1) + u(1-v)A(z_{\mathrm{u}} + 1, z_{\mathrm{v}}) + \\ & uvA(z_{\mathrm{u}} + 1, z_{\mathrm{v}} + 1) \end{aligned} \tag{3}$$

3 Improved Chroma Resampling Algorithm

3.1 Shortcomings of the original algorithm

As described above, there will be chroma component information loss after compression and recovery with the original video image chroma resampling algorithm. The extent of information loss is proportional to the times of "compression-recovery". For example, when compressing from YCbCr4:4:4 to YCbCr4:2:2, the original algorithm discards 1/2 of the chroma component data directly. In addition, the original algorithm discards 3/4 of the chroma component data directly when compressing to higher compression rate formats such as YCbCr4:2:0 and YCbCr4:1:1.

This paper utilizes composite peak signal noise ratio (CPSNR) to evaluate the image quality before and after chroma resampling.[7] The CPSNR indicates the average deviation between the target image and source image on the red, green, and blue channel. A larger CPSNR parameter means a smaller deviation. Therefore, it can be used for measuring the effect of the algorithm on image quality. First, we convert the source image from RGB format to YCbCr4:4:4 format. Then, we use the original algorithm to compress and recover at different sampling rates. Finally, we convert the result back to RGB format. We can compare the CPSNR parameters of 2 images before and after conversion so that we can measure the effect on image quality of the original algorithm.

Assume the source image is $\boldsymbol{X}$ and the target image after conversion is $\boldsymbol{Y}$. We can calculate the CPSNR parameter as[8]

$$\begin{aligned} & \mathrm{CMSE} = \frac{1}{3WH} \sum_{k=1}^{3} \sum_{i=1}^{W} \sum_{j=1}^{H} [\boldsymbol{X}(i,j,k) - \boldsymbol{Y}(i,j,k)]^2 \\ & \mathrm{CPSNR} = \\ & 10\log_{10} \frac{(2^n - 1)^2}{\dfrac{1}{3WH} \displaystyle\sum_{k=1}^{3} \sum_{i=1}^{W} \sum_{j=1}^{H} [\boldsymbol{X}(i,j,k) - \boldsymbol{Y}(i,j,k)]^2} = \\ & 10\log_{10} \frac{(2^n - 1)^2}{\mathrm{CMSE}} \end{aligned} \tag{4}$$

CMSE indicates the "composite mean square error" between source image and target image. i is the number of rows. j is the number of columns. $k = 1, 2, 3$ indicates R, G, and B channels respectively. H and W represent image height and width. n is the number of bits of a pixel component (generally 8).

We take the test image "Lena" as an example, which is commonly used in image processing. We do chroma resampling compression and recovery with the original algorithm at different resampling rates. As shown in Tab.1, we can get the corresponding CPSNR parameters after each process. The CPSNR parameter decreases obviously after compression and recovery in 3 common formats, which indicates that the original algorithm has an effect on image quality.

Tab. 1 CPSNR parameters of the original algorithm

dB

Uncompressed	YCbCr4:2:2	YCbCr4:2:0	YCbCr4:1:1
322.6674	44.5477	42.8425	38.0020

3.2 Improved algorithm based on interpolation method

In the original algorithm, we only consider the closest pixels around the target pixel during the process of compression and recovery, which causes the loss of image quality. Bilinear interpol-

ation will consider the data of 4 adjacent points around the target pixel. Because of this, more information can be retained during the scaling process. Therefore, this paper proposes an improved chroma resampling algorithm based on bilinear interpolation. In the new method, we scale the Cb and Cr chroma component matrices with bilinear interpolation. We set the horizontal and vertical zoom ratios separately to match the chroma component form with the corresponding chroma resampling format form. We can replace the method of dropping or copying pixel data directly with the interpolation method.

The specific improvement algorithm is as follows.

① Assume the size of the source image before chroma resampling is (L, W), which is also the size of the luma (Y) component matrix.

② Keep the Y component matrix unchanged. Scale the Cb and Cr component matrix using bilinear interpolation with horizontal scaling ratio x and vertical scaling ratio y. The specific scaling ratios depend on the YCbCr formats of the conversion, as written in Tab.2.

Tab. 2 Scaling ratio of chroma component matrix in different resampling formats

YCbCr format before conversion	YCbCr format after conversion	Horizontal scaling ratio (x)	Vertical scaling ratio (y)
YCbCr4:4:4	YCbCr4:2:2	1/2	1
YCbCr4:2:2	YCbCr4:4:4	2	1
YCbCr4:4:4	YCbCr4:2:0	1/2	1/2
YCbCr4:2:0	YCbCr4:4:4	2	2
YCbCr4:4:4	YCbCr4:1:1	1/4	1
YCbCr4:1:1	YCbCr4:4:4	4	1
YCbCr4:4:4	YCbCr4:1:0	1/4	1/2
YCbCr4:1:0	YCbCr4:4:4	4	2

③According to the YCbCr format data structure, split the Cb and Cr component matrix which has already been scaled to size $(x \times l, y \times w)$, and combine with the Y component matrix whose size is (L, W).

This paper not only improves the conversion between common YCbCr formats, but also proposes a new YCbCr4:1:0 format. Fig.4 shows the YCbCr4:1:0 data structure. In the YCbCr4:1:0 format, there are 4 Y components and only 1 Cb component or 1 Cr component in every group of 4 pixels. Cb and Cr are sampled at intervals by column. When the YCbCr4:1:0 format is combined with the improved algorithm in this paper, the data compression efficiency is improved, while ensuring the quality of the image.

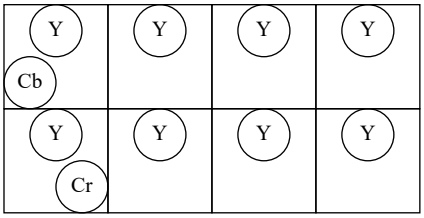

Fig. 4 YCbCr4:1:0 format pixels data diagram

4 Experimental Results and Analysis

We use a Matlab simulation to verify the effectiveness of the improved algorithm. Fig.5 shows the test process, which uses the compression of YCbCr4:2:2 as an example. We select 24 Kodak standard test images as the simulation object as shown in Fig.6, which are representative in terms of color and texture[9]. The research on the improved algorithm in this paper is based on these 24 standard test pictures.

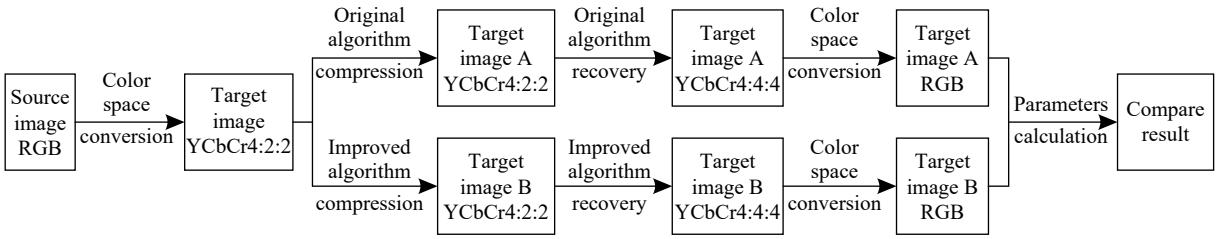

Fig. 5 Schematic diagram of algorithm testing process (YCbCr4:2:2 as an example)

Fig. 6 24 standard test images from Kodak

We use the CPSNR parameter to measure the quality of images. Tab.3 shows the parameter test results. The 24 test images Kodak01-Kodak24 are abbreviated as k01-k24. Algorithm A represents the original algorithm and algorithm B represents the improved algorithm in this paper.

Tab. 3 CPSNR results of chroma resampling performance of 2 algorithms

Image	Algorithm	CPSNR/dB				Image	Algorithm	CPSNR/dB			
		4:2:2	4:2:0	4:1:1	4:1:0			4:2:2	4:2:0	4:1:1	4:1:0
k01	A	44.561	42.663	40.334	39.617	k02	A	42.983	39.430	37.827	36.570
	B	50.039	48.232	44.530	44.077		B	48.450	44.928	42.069	41.058
k03	A	45.045	41.190	38.845	38.067	k04	A	44.603	41.366	39.605	38.456
	B	51.121	46.809	45.153	43.963		B	50.176	47.074	44.141	43.267
k05	A	41.871	38.318	35.797	35.000	k06	A	46.628	42.881	42.334	40.699
	B	48.108	44.509	41.635	40.759		B	50.509	46.866	46.129	44.768
k07	A	45.405	41.637	39.243	37.915	k08	A	42.066	40.001	36.807	36.091
	B	51.001	47.415	45.511	44.318		B	46.161	44.410	41.224	40.759
k09	A	45.609	43.135	39.459	38.803	k10	A	46.388	43.493	41.119	40.420
	B	51.594	48.809	45.484	44.791		B	50.889	47.981	45.706	44.879
k11	A	46.142	42.660	40.720	39.565	k12	A	46.522	44.206	41.147	40.502
	B	51.122	47.670	45.542	44.491		B	51.990	49.492	46.052	45.478
k13	A	43.399	40.137	38.527	37.426	k14	A	40.254	36.444	34.061	32.896
	B	46.912	43.820	42.524	41.629		B	46.458	42.252	39.747	38.567
k15	A	42.646	40.907	37.420	36.777	k16	A	50.537	46.546	45.259	43.789
	B	47.727	45.402	41.935	41.412		B	54.484	50.986	50.227	48.946
k17	A	45.999	43.645	41.384	40.620	k18	A	42.335	39.370	37.138	36.177
	B	49.320	47.314	45.336	44.962		B	46.567	44.018	41.809	41.183
k19	A	46.494	43.458	41.225	40.173	k20	A	44.832	40.881	39.764	38.307
	B	50.999	48.319	45.802	45.116		B	48.955	45.468	44.486	43.311
k21	A	46.266	42.235	41.043	39.575	k22	A	42.276	39.133	37.468	36.326
	B	50.371	46.538	45.824	44.517		B	46.370	43.759	41.464	40.687
k23	A	43.335	38.563	36.686	35.261	k24	A	39.245	36.762	35.109	34.388
	B	50.303	45.087	44.037	42.263		B	42.516	40.180	38.200	37.837

By comparing the parameter performance of the original algorithm and the improved algorithm, it can be seen that:

①For the commonly used YCbCr4:2:2, YCbCr4:2:0, and YCbCr4:1:1 compression methods, the CPSNR parameters of the improved algorithm in this paper are all better than the parameters of the original algorithm. This is because when calculating pixels in the chroma component, the improved algorithm considers the in-

fluence of the surrounding adjacent pixels.

②For the new YCbCr4:1:0 format using the improved algorithm, the CPSNR parameter performance is better than YCbCr4:2:0 and YCbCr4:1:1 using the original algorithm. At the same time, the new YCbCr4:1:0 format has a smaller data volume. Every 4 adjacent pixels only contain 4 luma components and 1 chroma component. The data volume of each format is counted in Tab.4 below. Tab.4 contains the number of luma and chroma components in every 4 adjacent pixels, the number of components on average in each pixel, and the data volume compression efficiency compared with RGB or YCbCr4:4:4 formats[10]. Thus, the new YCbCr4:1:0 format further improves the compression efficiency of video data.

Tab. 4 Compression efficiency of each YCbCr formats

YCbCr format	Luma components	Chroma components	Average	Compression efficiency
YCbCr4:4:4	4	8	3	00.00%
YCbCr4:2:2	4	4	2	33.33%
YCbCr4:2:0	4	2	1.5	50.00%
YCbCr4:1:1	4	2	1.5	50.00%
YCbCr4:1:0	4	1	1.25	58.33%

5 Conclusion

To deal with the shortcomings of chroma component information loss during the compression and recovery process in the original video image chroma resampling algorithm, this paper proposed an improved algorithm based on bilinear interpolation, and a new compression format. The interpolation method has the advantage of considering the influence of adjacent pixels around the target pixel. So we use the interpolation method to improve the calculation of chroma component data in the original algorithm. Therefore, we can improve the performance of objective parameters after image compression and recovery, and ensure the subjective visual quality at the same time. The experimental results show that, when compressing and recovering the video image with chroma resampling, the effect of the improved algorithm proposed in this paper is better than that of the original algorithm.

References:

[1] Livingstone M. Vision and Art: The Biology of Seeing[M]. New York: Abrams, 2008.

[2] Dargham J A, Chekima A, Moung E G, et al. A comparison of the YCBCR color space with gray scale for face recognition for surveillance applications [J]. Advances in Distributed Computing and Artificial Intelligence Journal, 2018, 72: 43-52.

[3] Shi Kaijie, Cao Hu, Tian Li, et al. A small-area and low-power circuit implementation of color space conversion from RGB to YCbCr[C]// 2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology. Hangzhou: IEEE Press, 2016: 1316-1318.

[4] Zhang Yukai, Shi Longzhao. Variable size and format YCbCr to RGB fast conversion module design based on FPGA [J]. Electronic Technology, 2014(5): 70-74.

[5] Zhang Xuejing, Chen He, Ma Long, et al. Fast image registration algorithm based on sub-image features[J]. Transactions of Beijing Institute of Technology, 2015, 35(7): 744-749. (in Chinese)

[6] Wang Ping, Quan Jicheng, Zhao Boyu. Realization of image Zooming in GPU based on bilinear interpolation [J]. Microelectronics & Computer, 2016，33(11): 129-132.

[7] Shan Baotang, Shen Tingzhi, Yan Xuemei, et al. Presentation and application of an adaptive proportional combination demosaicking algorithm[J]. Transactions of Beijing Institute of Technology, 2007, 27(11): 1013-1016. (in Chinese)

[8] Wu Chunming, Zheng Hongkuo, Wang Yanjiao, et al. A image compression algorithm based on multinode cooperation in wireless multimedia sensor networks[J]. Transactions of Beijing Institute of Technology, 2018, 38(5): 545-550. (in Chinese)

[9] Shan Baotang. The research on the key technologies of the multiple channel CMOS image sampling system[D]. Beijing: Beijing Institute of Technology, 2007. (in Chinese)

[10] Zhou Lisha, Guo Jian, Chen Yixing. Area optimization design of LCOS driver chip based on YCbCr [J]. Microcontrollers & Embedded Systems, 2019, 19(3): 20-22.