Color Interpolation for Single CCD Color Camera

Yi-Ming Wu, Chiou-Shann Fuh, and Jui-Pin Hsu
Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan
Email: r88036@csie.ntu.edu.tw; fuh@csie.ntu.edu.tw; d92004@csie.ntu.edu.tw

Abstract

This paper proposes a method to reduce the problems of blurred-edge effects and color alias effects in the color interpolation for single charge-coupled device (CCD) color camera. We will first introduce the background, and review some traditional interpolation methods. Then we will explain our proposed methods as well as the results. Conclusions and future works will be addressed at the last.

1. Introduction

Presently, the digital still cameras (DSCs) and PC cameras are popular with consumers as a device to input digital images easily. Size reduction and improvement of the image quality are the current trends in developing DSCs. In order to reduce the cost and size, most DSCs use a single charge-coupled device (CCD), instead of using three CCDs, to acquire color image.

However, the single CCD does not provide sufficient color resolutions. The solution for most DSC designers is to cover the sensor's surface with a mosaic of colored filters. This kind of filters is called a color filter array (CFA). An example is Bayer pattern (Fig. 1), a popular arrangement of CFA. Since there is only one color array element in each pixel in CFA, the other two missing color elements must be estimated. Thus, the recovery of full color images from CFA-based sensors requires a process of estimating values of missing color elements at each pixel by its adjacent pixels. This process is commonly called color interpolation or color demosaicking.

We will explain our proposed interpolation methods using Bayer pattern. It should be easily applied to other patterns.

2. Background

Some interpolation techniques are reviewed here:

1. Nearest neighbor interpolation: Just use the value of the nearest neighbor as the interpolated value.
2. Bilinear interpolation: Use a linear plane with least square error (against the known neighbor pixels) to interpolate a missing value.
3. Cubic B-spline interpolation [1] [2] [3] [4] [5]: Use a higher-order polynomial sphere with least square error (against the known neighbor pixels) to interpolate a missing value
4. Cubic convolution interpolation [6] [7] [8] [9]: Similar to cubic B-spline interpolation.
5. Simple edge filter: We adopt this simple gradiant and Laplacian edge-fileter, and will explain it later.
6. Weighting-based edge filter [10] [11] [12]: We adopt this method and will explain it later.
7. Color correlation [13] [14] [15] [16] [17]: We adopt a modified version of this method, i.e. color difference, and will explain it later in detail.
8. Color difference: We thought of this idea independent of the paper [18]. We submitted our patent [19] on Dec. 31, 2001, and the master's thesis of the primary author was published on May 25, 2001, which is a little bit later than paper [18]. However, our contributions are more than the color difference.

The first four traditional techniques share the same two side effects: the blurred edge effects and the color alias effects.

3. The proposed method

We will first introduce an edge-sensitive interpolator and review the other. Both can reduce the blurred-edge effects, but the former is faster, and the later [2] developed by S. Carrato et al. has better performance. Then we will review the concept of color correlation [4, 5, 6, 7], which copes with the color alias effects. At last we will propose a composite method, combining either one of the edge sensitive interpolators and our modified version of the color correlation interpolator, i.e. the color difference.

3.1 An edge-sensitive interpolator: bilinear interpolator with simple edge detector

We use the gradient edge detector and Laplacian edge detector to determine if a horizontal or vertical edge exists. The horizontal edge response $\Delta\tilde{H}$ and the vertical edge response $\Delta\tilde{V}$ can be expressed as:

$\Delta\tilde{H} = \left|G_{x+1,y} - G_{x-1,y}\right| + \left|2R_{x,y} - R_{x-1,y} - R_{x+1,y}\right|$, and

$\Delta\tilde{V} = \left|G_{x,y+1} - G_{x,y-1}\right| + \left|2R_{x,y} - R_{x,y-1} - R_{x,y+1}\right|$

respectively.

With a predefined threshold *T*, we interpolate the missing pixel by:

(Case 1) $\Delta\tilde{H} > T$, and $\Delta\tilde{V} \le T$:

A vertical edge is detected, and the interpolation along the horizontal direction is not desired. Thus we can do bilinear interpolation along the vertical direction only (two-way average).

(Case 2) $\Delta\tilde{H} \le T$, and $\Delta\tilde{V} > T$:

Similarly, a horizontal edge is detected, and we do bilinear interpolation along the horizontal direction only (two-way average).

(Case 3) Otherwise:

We will do the four-way average interpolation (two-dimensional bilinear interpolation).

3.2 Review of another edge-sensitive interpolator: weighting-based interpolator

The previous edge-sensitive interpolator can only detect horizontal or vertical edges. For edges in other directions, it may not work well. Thus a weight-based interpolator [2] comes.

Considering a location *(x, y)* with missing green value, we define $DG(s,t,u,v,p) = (G_{s,t} - G_{u,v}) * p$, and assign the four weights as:

$$w_{i,j} = \left(\begin{array}{l} 1 + DG\left(x-i, y-j, x+i, y+j, \frac{1}{2}\right)^2 \\ + DG\left(x-i, y-j, x+3i, y+3j, \frac{1}{4}\right)^2 \end{array} \right)^{-\frac{1}{2}}, \text{ or}$$

alternatively, we can save some computation by simplifying the weights as:

$$w_{i,j} = \left(\begin{array}{l} 1 + \left|DG\left(x-i, y-j, x+i, y+j, \frac{1}{2}\right)\right| \\ + \left|DG\left(x-i, y-j, x+3i, y+3j, \frac{1}{4}\right)\right| \end{array} \right)^{-1}, \text{ where}$$

$(i, j) \in \{(1,0), (-1,0), (0,1), (0,-1)\}$. Then, we can interpolate $G_{x,y}$ by $G_{x,y} = \dfrac{\sum_{(i,j)\in Neighbors(x,y)} G_{i,j} w_{i-x,j-y}}{\sum_{(i,j)\in Neighbors(x,y)} w_{i-x,j-y}}$,

where *Neighbors(x,y) = {(x-1,y), (x+1,y), (x,y-1), (x,y+1)}* is the pixel set of the diamond-shaped neighborhood of position *(x,y).*

For positions with defined green value, the weighting-based interpolations of $R_{x,y}$ and $B_{x,y}$ can be similarly derived.

3.3 Color correlation

To overcome or reduce color-alias effects, some original interpolation schemes [3, 4] were proposed. Among those schemes, the color correlation is frequently mentioned. Kuno and Sugiura [4] proposed a new interpolation method using discriminated color correlation for digital still cameras. In practice, however, their proposed scheme uses a quotient equation, which sometimes fails to reconstruct stably when the denominator is small.

To avoid the quotient, we adopt another image model developed by Adams [5, 6, 7], with the assumption that green and red, or green and blue values are perfectly correlated to each other within a simple offset in a small region, that is, $G_{x,y} = R_{x,y} + K_R$, and $G_{x,y} = B_{x,y} + K_B$, where K_R and K_B are called the difference domains of the red channel, and blue channel respectively. We will call it color difference. Color difference works quite well, since K_R and K_B channels are quite flat in practice.

We will not give an explicit interpolation method here, instead, we will give a composite method using the above two equations directly.

3.4 The proposed composite method

We will focus on the weight-based edge-sensitive interpolator plus the color correlation. Combining the gradient and Laplacian edge-sensitive interpolator plus the color correlation is similar.

The whole task is to interpolate the unknown pixel values. First, we will estimate the missing *G* values.

For position (x, y) with missing G value but known red value, we defined the four K values as:

$K_{i,j} = G_{x+i,y+j} - (R_{x+2i,y+2j} + R_{x,y}) / 2$ ....**(Def. 1)**

where $(i, j) \in \{(-1,0), (1,0), (0,1), (0,-1)\}$.

Note that in the equation, the averages of the two *R* values are actually an estimation of the middle *R* values, i.e. $R_{x+i,y+j}$. Thus these four *K* values are actually estimations of *G* – *R* values for each pixels in *Neighbors(x,y)*. Note that all four *K* values are well-defined since all those *G* and *R* values on the right-hand-side are known values. We further defined:

$K_{0,0} = G_{x,y} - R_{x,y}$ ....**(Eq. 1)**

All these five K values are said to be the value of R in the color difference domain.

Now we can obtain an estimate $\hat{K}_{0,0}$ of $K_{0,0}$ by applying either of the edge-sensitive interpolator on the first four *K* values, i.e., $K_{-1,0}$, $K_{1,0}$, $K_{0,-1}$, and $K_{0,1}$, and then estimate $G_{x,y}$ from $\hat{K}_{0,0}$ and $R_{x,y}$ via **Eq. 1**. Note that to apply the weighting-based interpolator, just replace all *G*'s to *K*'s for the function *DG(.)*, and the last interpolation equation. Also, we now obtain *DK(.)* by replacing all *G*'s to *K*'s in *DG(.)* for later use.

To interpolate *G* values on positions with defined *B* values is symmetric.

Next, we will estimate the missing *R*, and *B* values at positions with originally known *G* values. For such a position *(x,y)* with known *R* values at its left and right, i.e. *(x-1,y)* and *(x+1,y)*, **Def. 1** becomes

$K_{i,0} = G_{x+i,y} - R_{x+i,y}$ .... **(Def. 2)**

Note that we already have those *G* values in previous estimations.

Since we only have two neighboring *R* values to interpolate the R value at position *(x,y)* from, we will estimate $\hat{K}_{0,0}$ of $K_{0,0}$ again, but by one-dimensional weighting-based interpolation, which we can just assign

$$w_{i,0} = \begin{pmatrix} 1 + DK\left(x-i, y, x+i, y, \frac{1}{2}\right)^2 \\ + DK\left(x, y, x+2i, y, \frac{1}{2}\right)^2 \end{pmatrix}^{-\frac{1}{2}}, \quad \text{or}$$

alternatively, $$w_{i,0} = \begin{pmatrix} 1 + \left| DK\left(x-i, y, x+i, y, \frac{1}{2}\right)\right| \\ + \left| DK\left(x, y, x+2i, y, \frac{1}{2}\right)\right| \end{pmatrix}^{-1},$$

where $i = \pm 1$ and assign *Neighbors(x,y)={(x-1,y), (x+1,y)}* in the calculation of weighting-based interpolation of $K_{0,0}$. Then again, using **Eq. 1**, but this time, we estimate $R_{x,y}$, using estimated $K_{0,0}$, and the previously estimated $G_{x,y}$

Similarly, other missing *R* and *B* values in positions with originally known *G* values can be done symmetrically.

Next, we will estimate *R* values at the positions with originally known *B* values. In the neighborhood of such a position *(x,y)*, the positions with known *R* values are at the four positions $(x \pm 1, y \pm 1)$. To interpolate Rfrom those known *R* values, we then define another four *K* values as $K_{i,j} = G_{x+i,y+j} - R_{x+i,y+j}$, where $i, j = \pm 1$. Then we define another four *w* values

$$w_{i,0} = \begin{pmatrix} 1 + DK\left(x-i, y-j, x+i, y+j, \frac{1}{2}\right)^2 \\ + DK\left(x, y, x+2i, y+2j, \frac{1}{2}\right)^2 \end{pmatrix}^{-\frac{1}{2}}, \quad \text{or}$$

alternatively,

$$w_{i,0} = \begin{pmatrix} 1 + \left| DK\left(x-i, y-j, x+i, y+j, \frac{1}{2}\right)\right| \\ + \left| DK\left(x, y, x+2i, y+2j, \frac{1}{2}\right)\right| \end{pmatrix}^{-1}, \quad \text{where}$$

$i = \pm 1$, and assign *Neighbors(x,y) = {(x-1,y-1),(x+1,y+1),(x-1,y+1),(x+1,y-1)}* and apply the weighting-based interpolation to obtain an estimation of $K_{0,0}$. Then again, we use Eq. 1 to estimate $R_{x,y}$.

Similarly, interpolating *B* values at positions with originally known *R* values can be done symmetrically. This concludes the interpolation process.

In fact, we choose G channel as the base to do the color difference, simply because green values are dominant (Half of the pixels have known green values).

4. Results and discussion

To simulate Bayer pattern, original images are down-sampled, to extract only one color band per pixel, according to Bayer pattern. Fig. 3 shows the images before and after down-sampling. Fig. 4 shows a original image with the interesting region in a bounded box. Fig. 5 through 7 shows some interpolation results.

4.1 Peak signal to noise ratio (PSNR) results and comparisons

We now apply the PSNR to measure the similarity between the original images and the down-sampled image followed by various interpolation methods. The PSNR value is defined as follows: $PSNR(I_o, I_r) = 10 \times \log_{10}\left(\frac{255^2}{MSE(I_o, I_r)}\right)$, where

$$MSE(I_o, I_r) = \frac{1}{H \times W} \times \sum_{y=0}^{H-1}\sum_{x=0}^{W-1}\left(I_o(x,y) - I_r(x,y)\right)^2 ,$$

I_o is the original image, I_r is the resulting image, *MSE* is the mean square error, *H* is the height of the image, *W* is the width of the image, and 255 is the maximum value which a pixel can have in the 8-bits image.

In our experiments, we take 100 images shown in Fig. 2. The first 90 images are natural scenes in our lives and the last 10 images are artificial paints. We down-sampled these 100 images and then apply various interpolation methods to them. The interpolation methods which will be compared in this experiment are nearest neighbor, bilinear, cubic B-spline, cubic convolution, simple edge-filtered (Gradiant plus Laplacian), weighting-based edge-filtered, simple edge-filtered plus color-difference, weighting-based edge-filtered plus color-difference, and simplified weighting-based plus color-difference. The first four methods are traditional interpolation methods, the next two methods are edge-sensitive interpolation methods, and the last three methods are our proposed composite interpolation schemes. At last, the PSNR values are calculated.

For clearer comparison, we assign a score to every method according to the rank of the PSNR value. We assign score 1 to the minimum PSNR (worst) among all methods, score 2 to the next better place, and increasing scores to the others according to their places, the maximum PSNR (best) will get score 9. After scoring, we summed total scores per method and the result is shown in Table 4 and observed that the weighting color-difference interpolation method has the best score and its simplified version is slightly worse. Summarily, the composite interpolation methods have better scores, the edge-sensitive methods are next, and the traditional interpolation methods are worst. Besides, the weighting-based method is better than edge-filtered method. Among traditional interpolations, the cubic convolution interpolation is the best and the nearest neighbor interpolation is the worst.

5. References

[1] D. P. Mitchell and A. N. Netravali, "Reconstruction Filters in Computer Graphics", Computer Graphics, vol. 22, no. 4, pp. 221-228, 1988.

[2] H. S. Hou and H. C. Andrews, "Cubic Splines for Image Interpolation and Digital Filtering," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 26, pp. 508-517, 1978.

[3] C. Lee, M. Eden, and M. Unser, "High Quality Image Resizing Using Oblique Projection Operators," IEEE Transactions on Image Processing, vol. 7, no. 5, pp. 679-692, 1998.

[4] B. Lee, J. Kim, and C. Lee, "High Quality Image Interpolation for Color Filter Arrays," Proceedings of IEEE International Conference on System, Man, and Cybernetics, vol. 2, pp. 1547-1550, 2000.

[5] M. Unser, A. Aldroubi, and M. Eden, "B-spline Signal Processing: Part 1 – Theory," IEEE Transactions on Signal Processing, vol. 41, pp. 821-833, 1993.

[6] R. G. Keys, "Cubic Convolution Interpolation for Digital Image Processing," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 29, pp. 1153-1160, 1981.

[7] K. W. Simons, "Digital Image Reconstruction and Resampling for Geometric Manipulation," Proceedings of IEEE Symposium on Machine Processing of Remotely Sensed Data, pp. 3A-1-3A-11, 1975.

[8] D. P. Mitchell and A. N. Netravali, "Reconstruction Filters in Computer Graphics," Computer Graphics, vol. 22, no. 4, pp. 221-228, 1988.

[9] W. F. Schreiber and D. E. Troxel, "Transformation Between Continuous and Discrete Representations of Image: A Perceptual Approach," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 7, no. 2, pp. 178-186, 1985.

[10] S. Carrato, G. Ramponi, and S. Marsi, "A Simple Edge-Sensitive Image Interpolation Filter," Proceedings of International Conference on Image Processing, vol. 3, pp. 711-714, 1996.

[11] S. Carrato, G. Ramponi, and S. Marsi, "A Simple Edge-Sensitive Image Interpolation Filter," Proceedings of International Conference on Image Processing, vol. 3, pp. 711-714, 1996.

[12] Y. C. Lan, "Adaptive Digital Zoom Techniques Based on Hypothesized Boundary," Master Thesis, National Taiwan University, Department of Computer Science and Information Engineering, 1998.

[13] W. T. Freeman, "Median Filter for Reconstructing Missing Color Samples," United States Patent, 4724395, 1998.

[14] T. Kuno and H. Sugiura, "New Interpolation Method Using Discriminated Color Correlation for Digital Still Cameras," IEEE Transactions on Consumer Electronics, vol. 45, no. 1, pp. 259-267, 1999.

[15] J. E. Adams, Jr., "Design of Practical Color Filter Array Interpolation Algorithms for Digital Cameras," Proceedings of SPIE, vol. 3028, pp. 117-125, 1997.

[16] J. E. Adams, Jr., "Design of Practical Color Filter Array Interpolation Algorithms for Digital Cameras, Part 2," Proceedings of International Conference on Image Processing, vol. 1, pp. 488-492, 1998.

[17] J. Adams, K. Parulski, and K. Spaulding, "Color Processing in Digital Cameras," IEEE Micro, vol. 18, no. 6, pp. 20-30, 1998.
[18] S. C. Pei and I. K. Tam, "Effective Color Interpolation in CCD color filter array using signal correlation," in Proc IEEE International Conference on Image Processing, Vol. 3, pp. 488-491, 2000
[19] Yi-Ming Wu, Chiou-Shann Fuh, Pu-Hua Mei, "A color interpolation technique for digital images", Taiwan Patent, #90133340, December 2001

Method	NN	BI	CB	CC	SEF	WB	SEFCD	WBCD	SWBCD
Accumulated scores	100	325	217	452	453	560	730	874	788

Table 1: The total PSNR comparison scores, the higher the better. NN: nearest neighbor, BI: bilinear, CB: cubic B-spline, CC: cubic convolution, SEF: simple edge-filterd, WB:weighting-based filtered, SEFCD: simple edge-filtered + color difference, WBCD: weighting-based edge-filtered + color difference, SWBCD: simplified weighting-based + color difference

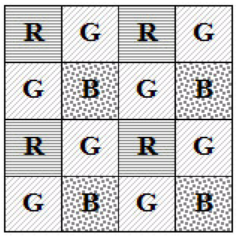


Figure 1: Bayer pattern for CFA.

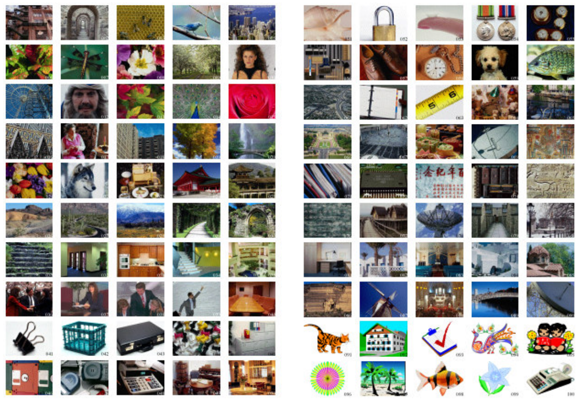
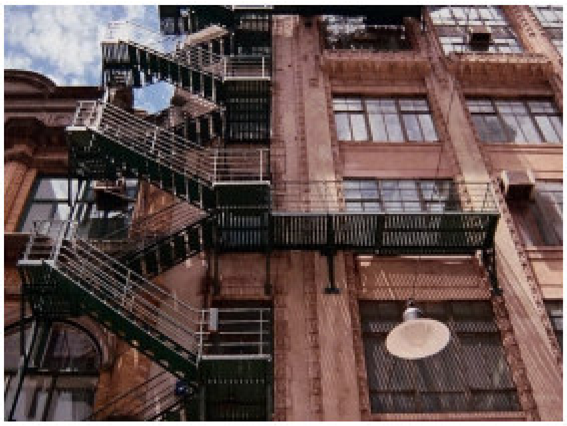


Figure 2: The 100 testing color images (640x480) to be measured.



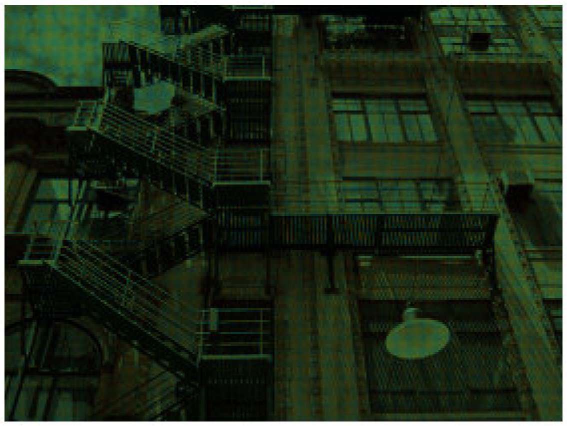


Figure 3:
(a) The original image.

(b) The result of the down-sampled (a).

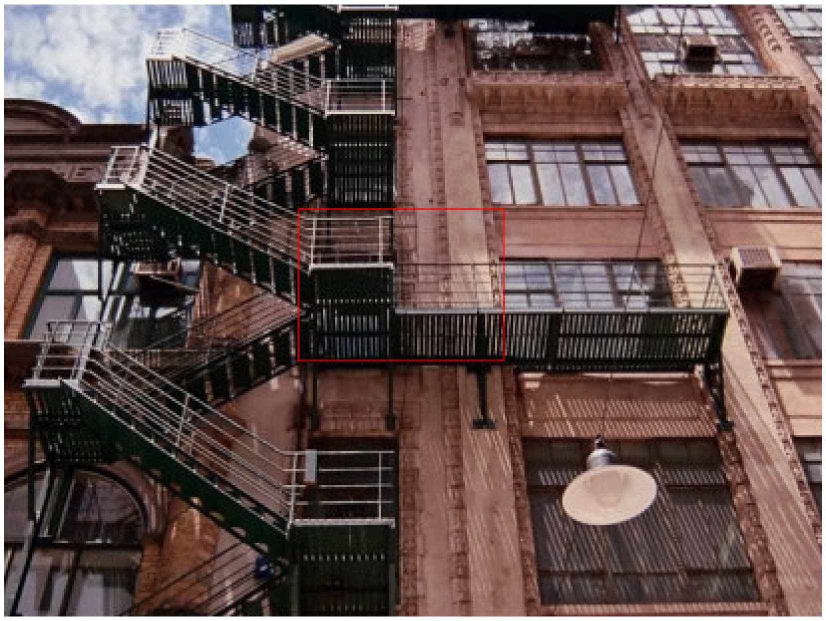


Figure 4: The original image for testing and the bounding box in the center is the region of interest.

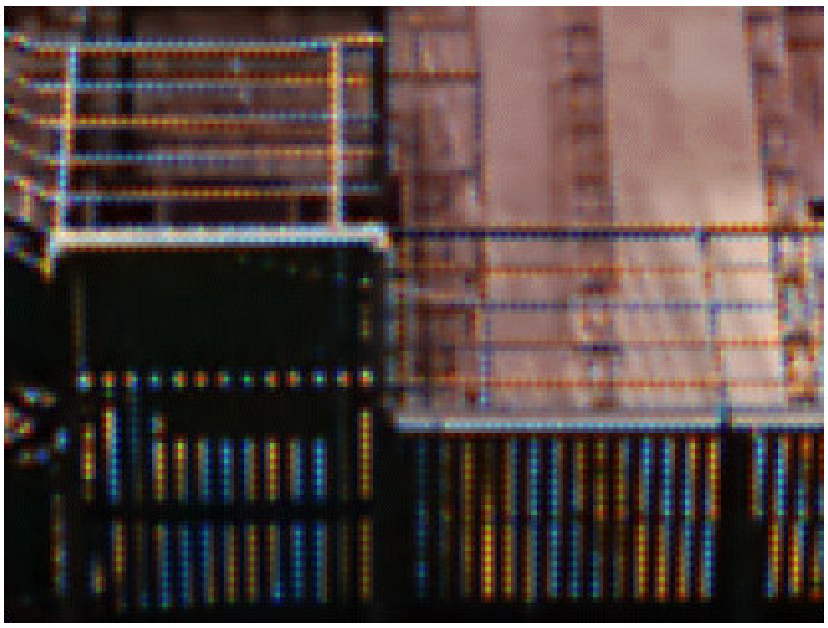


Figure 5: The interpolated image with bilinear interpolation, showing the scale-up image of the bounded box, where the average PSNR is 25.21 dB.

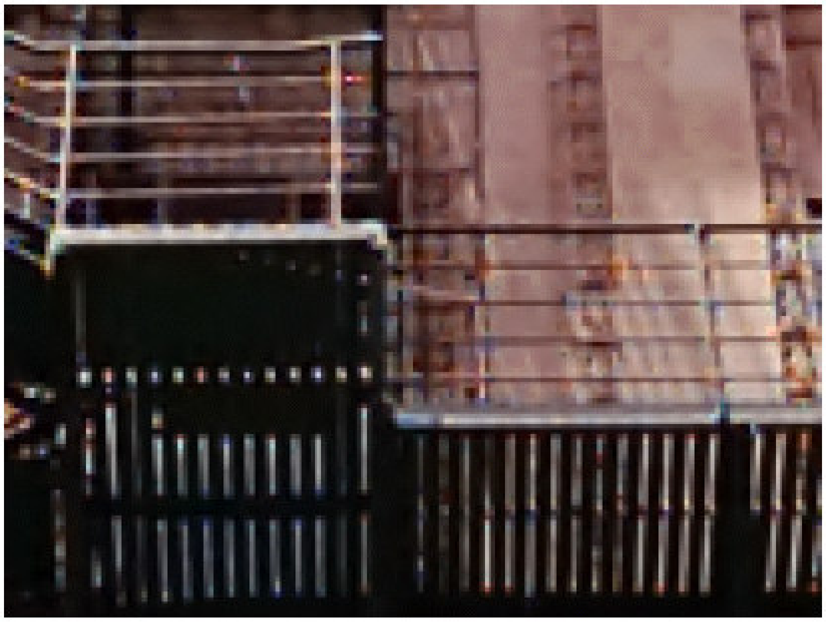


Figure 6: The interpolated image with edge-filtered color-difference interpolation, showing the scale-up image of the bounding box, where the average PSNR is 34.52 dB.

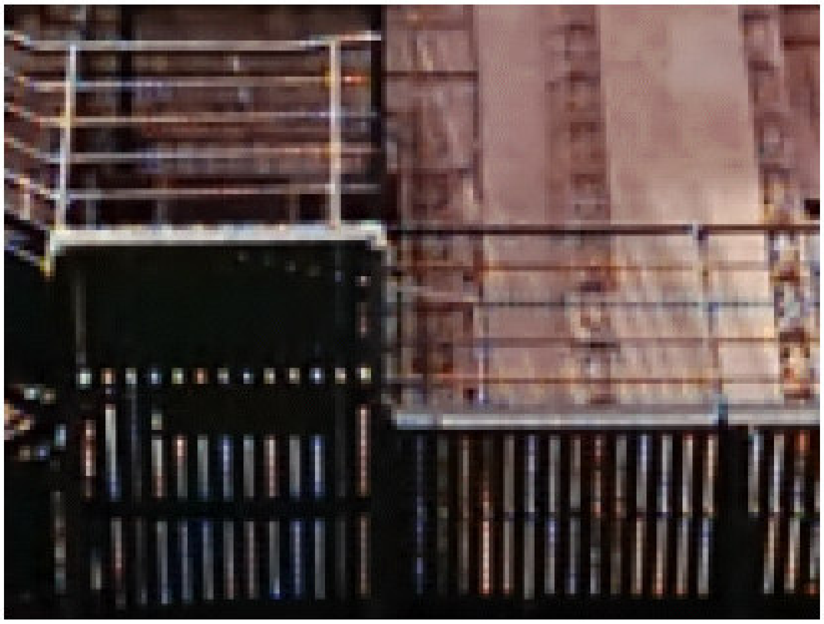


Figure 7: The interpolated image with weighting color-difference interpolation, showing the scale-up image of the bounding box, where the average PSNR