IMAGE PIPELINE ALGORITHMS FOR STANDARD MOBILE IMAGING ARCHITECTURE SENSORS

Bo-Ching Huang (黃柏青) and Chiou-Shann Fuh (傅楸善)
Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan
r92121@csie.ntu.edu.tw and fuh@csie.ntu.edu.tw

ABSTRACT

With rapid growth of DSC and camera phones, image pipeline becomes more and more important. Besides, new standard between camera module and mobile host called SMIA (Standard Mobile Imaging Architecture) are proposed last year. We will briefly introduce the standard in this paper. Image pipeline is a technique to transform sensor's raw data to final compressed image. The images are processing through the pipeline: Optical Black Clamp, White Balance, Color Interpolation, Color Correction, Gamma Correction, Edge Enhancement, and Image Compression. This thesis contains original thought in some steps of the image pipeline. In White Balance, we propose a new method called Modified Gray World Assumption (MGWA). In Color Interpolation, we combine gradient and Laplacian edge detectors to do Edge-Sensitive Interpolation. In Edge Enhancement, we proposed a new method called Dynamic High-Boost Filtering. Also, we bring up an idea that Saturation Adjustment can be added in the image pipeline algorithm.

1. OVERVIEW OF IMAGE PIPELINE

The main purpose of image pipeline is to process the image from raw data to final compressed image that fits human perception. Figure 1 shows our image pipeline steps.

First, we read raw data from the sensor. In order to reduce the thermal noises from the sensor, Optical Black Clamp is done first to prevent the noises going through the pipeline and leading to poor image. Furthermore, because the sensor's sensitivity is different from human perception, we must add more steps such as White Balance, Color Correction, and Gamma Correction to get preferred image. Color Interpolation transfers raw data to RGB data by interpolating the other two missing channels. Edge Enhancement makes the image perceptually sharper. Image Compression uses JPEG (Joint Photographic Experts Group) to reduce the size of the image.

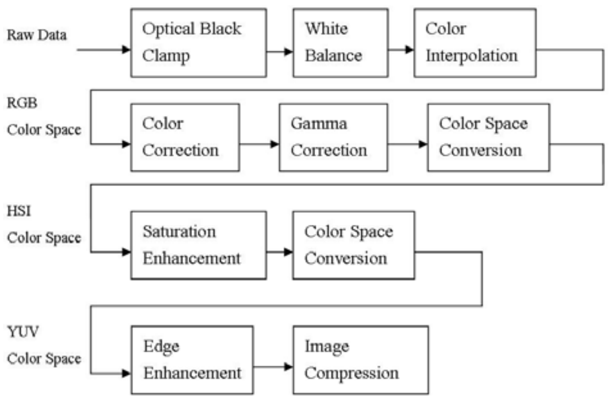


Figure 1 The flow of image pipeline.

2. OPTICAL BLACK CLAMP

As long as we are not taking pictures at 0oK, there is thermal noise on our circuit. We use an optical black to block light into the circuit. Ideally, there is no current in the optical black area, but thermal noise still exists on our circuit. Therefore, we subtract the average value of optical black area from all pixels to reduce the thermal circuit noise (Figure 2).

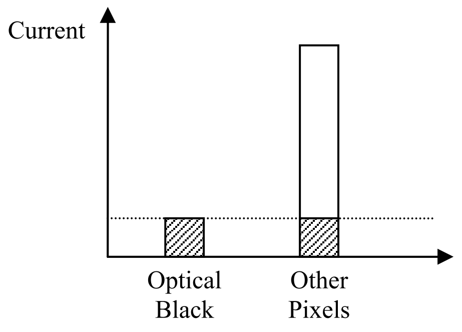
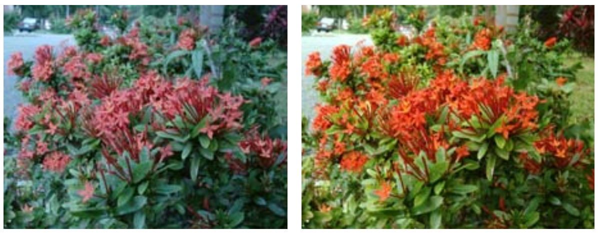


Figure 2 Optical black clamp.

3. WHITE BALANCE

Our white balance method is based on GWA with little enhancement. In GWA, we assume all pixels should meet a gray value. In our method, we assume pixels with $lower\_bound \le Y \le upper\_bound$ and $|Cb - Cr| < threshold$ should meet a gray value. That is, we filter out of the dark pixels, bright pixels, and color-biased pixels. For example, if we take photos on flowers shown in Figure 3, the result of GWA will be bluish (a).



(a) GWA. (b) Our method.

Figure 3 Comparison of GWA and our method.

4. COLOR INTERPOLATION

In this step, we use an Edge-Sensitive Interpolation method [11]. First, we calculate the gradient and Laplacian edge filters in horizontal and vertical directions. Then, we combine the two edge filters defined as composite edge filters: $\Delta\tilde{H}$ and $\Delta\tilde{V}$. Figure 4 shows the equations to calculate the edge filters. Finally, we can define the composite edge filters as follows:

(1) If $\Delta\tilde{H} \le threshold$ and $\Delta\tilde{V} \le threshold$, then no edges exist in this 3 x 3 region. Use two-dimensional interpolation.

(2) If $\Delta\tilde{H} \le threshold$ and $\Delta\tilde{V} > threshold$, then horizontal edge exists in this 3 x 3 region. Use one-dimensional interpolation along horizontal direction.

(3) If $\Delta\tilde{H} > threshold$ and $\Delta\tilde{V} \le threshold$, then vertical edge exists in this 3 x 3 region. Use one-dimensional interpolation along vertical direction.

(4) Otherwise, if $\Delta\tilde{H} > threshold$ and $\Delta\tilde{V} > threshold$, then other directional edge, neither horizontal nor vertical, exists in this 3 x 3 鋌region. Use two-dimensional interpolation.

Gradient:

$$\Delta H = |G_{x+1,y} - G_{x-1,y}|$$
$$\Delta V = |G_{x,y+1} - G_{x,y-1}|$$

Laplacian:

$$\Delta\hat{H} = |2R_{x,y} - R_{x-2,y} - R_{x+2,y}|$$
$$\Delta\hat{V} = |2R_{x,y} - R_{x,y-2} - R_{x,y+2}|$$

Composite edge

$$\Delta\tilde{H} = \Delta H + \Delta\hat{H}$$
$$\Delta\tilde{V} = \Delta V + \Delta\hat{V}$$

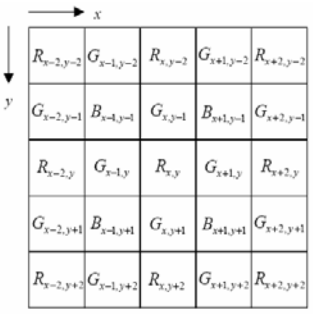

x
y
$R_{x-2,y-2}$ $G_{x-1,y-2}$ $R_{x,y-2}$ $G_{x+1,y-2}$ $R_{x+2,y-2}$
$G_{x-2,y-1}$ $B_{x-4,y-1}$ $G_{x,y-1}$ $B_{x+1,y-1}$ $G_{x+2,y-1}$
$R_{x-2,y}$ $G_{x-1,y}$ $R_{x,y}$ $G_{x+1,y}$ $R_{x+2,y}$
$G_{x-2,y+1}$ $B_{x-4,y+1}$ $G_{x,y+1}$ $B_{x+1,y+1}$ $G_{x+2,y+1}$
$R_{x-2,y+2}$ $G_{x-1,y+2}$ $R_{x,y+2}$ $G_{x+1,y+2}$ $R_{x+2,y+2}$

Figure 4 Equations to calculate edge filters.

5. COLOR AND GAMMA CORRECTION

5.1. Color Correction

The sensor's color sensitivity is different from human eyes. In order to match the perception of human eyes, we multiply a 3x3 matrix to each pixel of the image. How do we get the 3x3 matrix? First, we take photos on standard target like GretagMacbeth ColorChecker (Figure 5), then we can get the average RGB value of each block of the photographed ColorChecker. Besides, we know the ideal RGB value of the ColorChecker (Table 1). Thus, we multiply the color rotation matrix M to each block of photographed ColorChecker (Equation 1). Finally, we can get the color rotation matrix by using Equation 2. However, the color rotation matrix is an approximate matrix because Equation 2 is an over-determined equation.

$$\begin{bmatrix} R_1 & G_1 & B_1 \\ R_2 & G_2 & B_2 \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ R_{24} & G_{24} & B_{24} \end{bmatrix}_P \times M = \begin{bmatrix} R_1 & G_1 & B_1 \\ R_2 & G_2 & B_2 \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ R_{24} & G_{24} & B_{24} \end{bmatrix}_C \qquad \text{Equation 1}$$

$$M = \begin{bmatrix} R_1 & G_1 & B_1 \\ R_2 & G_2 & B_2 \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ R_{24} & G_{24} & B_{24} \end{bmatrix}_C \times \begin{bmatrix} R_1 & G_1 & B_1 \\ R_2 & G_2 & B_2 \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ R_{24} & G_{24} & B_{24} \end{bmatrix}_P^T \qquad \text{Equation 2}$$

where M is the color rotation matrix; P refers to the photographed values; and C refers to ColorChecker ideal values passing an inverse gamma function.

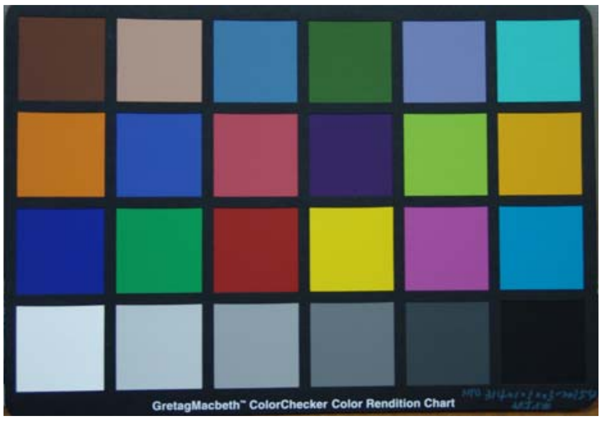


Figure 5 GretagMacbeth ColorChecker.

No	Name	R	G	B
1	Dark skin	94	28	13
2	Light skin	241	149	108
3	Blue sky	97	119	171
4	Foliage	90	103	39
5	Blue flower	164	131	196
6	Bluish	140	253	153

7	Orange	255	116	21
8	Purplish	7	47	122
9	Moderate	222	29	42
10	Purple	69	0	68
11	Yellow	187	255	19
12	Orange	255	142	0
13	Blue	0	0	142
14	Green	64	173	38
15	Red	203	0	0
16	Yellow	255	217	0
17	Magenta	207	3	124
18	Cyan	0	148	189
19	White	255	255	255
20	Neutral 8	249	249	249
21	Neutral 6.5	180	180	180
22	Neutral 5	117	117	117
23	Neutral 3.5	53	53	53
24	Black	0	0	0

Table 1 The ideal RGB values of the GretagMacbeth ColorChecker. (*optical density)

5.2 Gamma Correction

Sensors are linear; human eyes are nonlinear. To compensate the difference of sensors and human eyes, we adjust the original linear curve to nonlinear curve to fit human eyes' perception. But, human vision varies. Some may feel a gamma-corrected image brighter; others may think the image just fits the scene.

There are many ways to do gamma correction. A common approach is applying gamma 0.45 curve shown at Equations 3 and 4. The former is 10-bit to 8-bit conversion; the later is 8-bit to 8-bit conversion.

$$f(x) = 255 * ((x+1) / 1024)^n \qquad \text{Equation 3}$$
$$f(x) = 255 * ((x+1) / 256)^n \qquad \text{Equation 4}$$

where *n = 0.45* in gamma 0.45 curve; *x* is the value of each pixel.

For efficient computing, we always construct a gamma lookup table in gamma correction step. Therefore, the gamma correction step is also called the gamma mapping step. However, sensor manufacturers provide their own gamma lookup table according to the sensor sensitivity.

6. SATURATION ADJUSTMENT

6.1 Color Space Conversion from RGB to HSI

Saturation Adjustment processes in HSI color space, so we need to convert the color space from RGB to HSI. Equation 3.4.1 shows the transformation formula.

Hue: $$H = \begin{cases} \delta & , \text{if } B \le G \\ 360^o - \delta & , \text{if } B > G \end{cases}$$

where $$\delta = \cos^{-1} \frac{\frac{(R-G)+(R-B)}{2}}{\sqrt{(R-G)^2 + (R-B)(G-B)}} \text{ in } [0, 180^o)$$

Saturation: $$S = 1 - 3 \times \frac{\min\{R, G, B\}}{R+G+B}$$

Intensity: $$I = \frac{R+G+B}{3} \text{ for } R, G, B \in [0,1]$$

Equation 5.

After conversion, we can easily use the H, S, and I components to tune our images. For example, if we want to present a sepia effect, we can easily set a specific angle on the H component and a fixed value on the S component.

To convert color space from HSI to RGB is more complex. We use different conversion methods according to the H component as shown in Figure 6

.

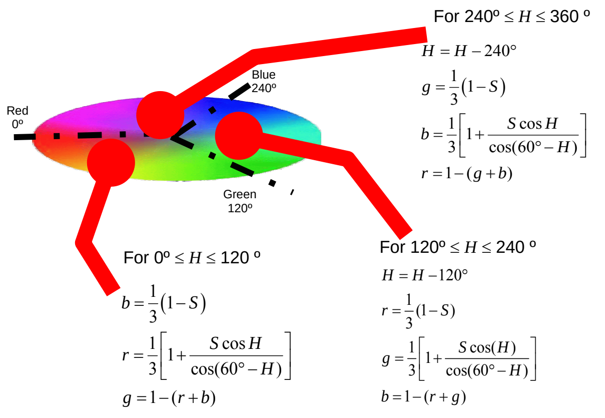

For $240^o \le H \le 360^o$
$H = H - 240^o$
$g = \frac{1}{3}(1-S)$
$b = \frac{1}{3}\left[1 + \frac{S\cos H}{\cos(60^o - H)}\right]$
$r = 1-(g+b)$
Red 0°
Blue 240°
Green 120°
For $0^o \le H \le 120^o$
$b = \frac{1}{3}(1-S)$
$r = \frac{1}{3}\left[1 + \frac{S\cos H}{\cos(60^o - H)}\right]$
$g = 1-(r+b)$
For $120^o \le H \le 240^o$
$H = H - 120^o$
$r = \frac{1}{3}(1-S)$
$g = \frac{1}{3}\left[1 + \frac{S\cos(H)}{\cos(60^o - H)}\right]$
$b = 1-(r+g)$

Figure 6 Color space conversion from HSI to RGB [5].

6.2 Saturation Adjustment

After color space conversion, we get the information of hue, saturation, and intensity. The saturation is a floating number from 0 to 1 where "0" represents no saturation (gray) and "1" represents full saturation (vivid color). Thus, we use a gamma function shown in Equation 6 to enhance the saturation channel.

$$S = S^{\beta} \qquad \text{Equation 6}$$

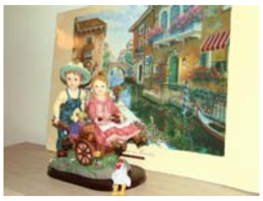
Figure 7 shows saturation adjustments with different β values. Figure 8 shows corresponding β mapping functions where *x*-axis is the input saturation value and *y*-axis is the output saturation value. From our statistics, the image with β *=0.6* is the favorite one.

(a) $\beta = 1$

(b) $\beta = 0.8$

(c) $\beta = 0.7$

(d) $\beta = 0.6$


(e) $\beta = 0.5$

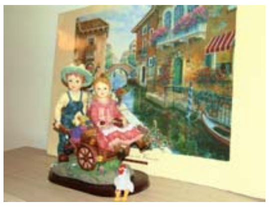
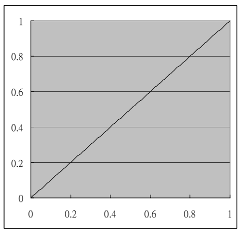
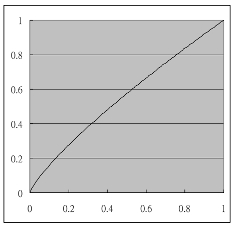

(f) $\beta = 0.45$

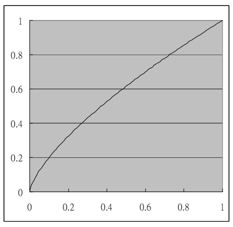
Figure 7 Saturation adjustments with different β values.



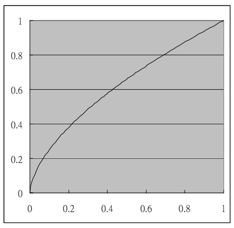
(a) $\beta = 1$



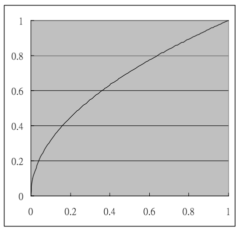
(b) $\beta = 0.8$



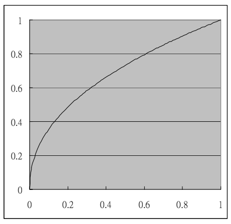
(c) $\beta = 0.7$



(d) $\beta = 0.6$



(e) $\beta = 0.5$



(f) $\beta = 0.45$

Figure 8 Different β mapping functions.

7. EDGE ENHANCEMENT

Our edge enhancement method is based on high-boost filtering mentioned in Section 2.8 because most people like brighter images. However, it is hard to determine a constant value of *A* that would satisfy all images. For example, Figure 9 and Figure 10 show high-boost filtering using different *A* value. Most people prefer the image with *A=1.2* or *A=1.3* in Figure 3.6.1 because the color of the flower is very attractive. Moreover, most people prefer the image with *A=1.1* in Figure 3.6.2, because the clouds are over-saturated in the images with *A>1.2* and the image with *A=1.0* is too dim. Therefore, we propose dynamic high-boost filtering to dynamically choose the best *A* value. We take the Gray Value (*GV*) obtained in White Balance, *LV* (Light Value), and flash into consideration. The *LV* is obtained from the header of raw data with Equation 7.

$$EV = \log_2 \frac{F \times F}{S}$$
$$LV = EV + \log_2 \frac{ISO}{100} \qquad \text{Equation 7}$$

where *EV* is exposure value; *F* is f-number; *S* is shutter speed; *ISO* is ISO speed.

The cases are divided into two categories: flash-on and flash-off, because *LV* is incorrect while flash-on. For flash-off category, we construct a table shown in Table 2 by experiment. It shows that images with higher *LV* should have higher gray values. Also, higher *A* values on lower gray value images get more preferences and cause the dim image to become brighter. For flash-on category, the *LV* is not considered such that we assume that *LV=7~9* and imitate the flash-off category.

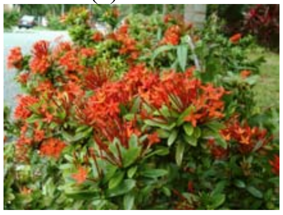
GV \ *LV*	0~7	7~9	9~11	12~
7000~	1	1	1	1
6000~7000	1	1.1	1.1	1.1
5000~6000	1	1.1	1.1	1.1
3000~5000	1.1	1.2	1.2	1.3
1000~3000	1.2	1.3	1.4	1.4
0~1000	1.4	1.4	1.4	1.5

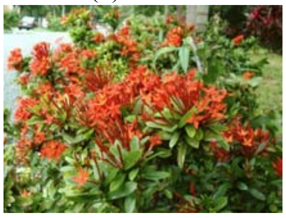
Table 2 Dynamic high-boost filtering with flash-off.

(*GV*: Gray Value in 14 bits, *LV*: Light Value)

(a) *A=1.0*

(b) *A=1.1*


(c) *A=1.2*


(d) *A=1.3*

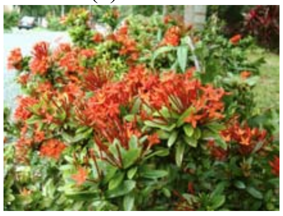
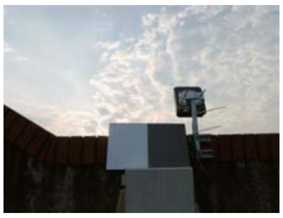
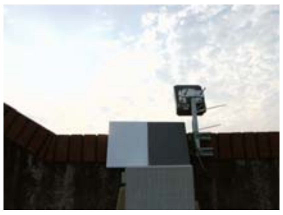
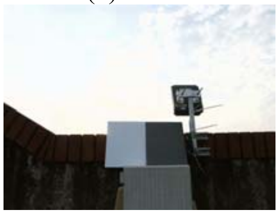

(e) *A=1.4*

Figure 9 High-boost filtering with different *A* values (*LV*=8, *GV*=3235, flash: off).


(a) *A=1.0*

(b) *A=1.1*


(c) *A=1.2*


(d) *A=1.3*

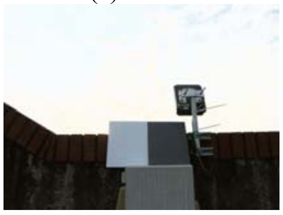
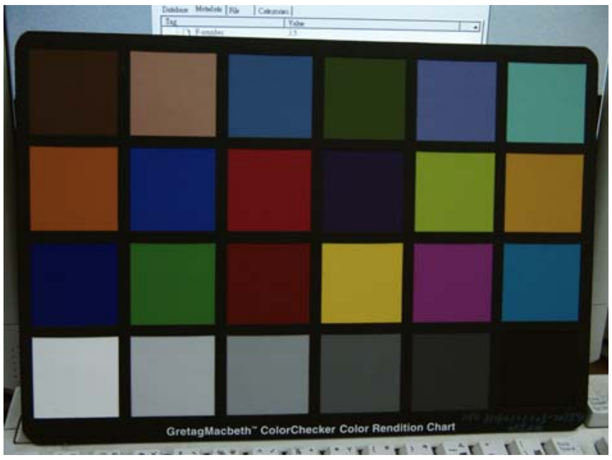

(e) *A=1.4*

Figure 10 High-boost filtering with different *A* values (*LV*=13, *GV*=6372, flash: off).

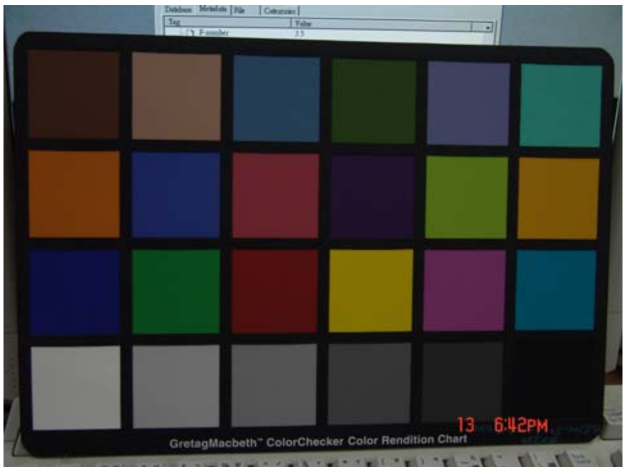
8. EXPERIMENTS AND RESULTS

In this chapter, we will briefly explain the experiment environment and show our image pipeline results. We simulate with a C++ program and apply our image pipeline algorithm on raw data taken from Sony DSC-F828. The way to judge which image is better is very objective if many voters vote on two images side-by-side.

8.1 ColorChecker

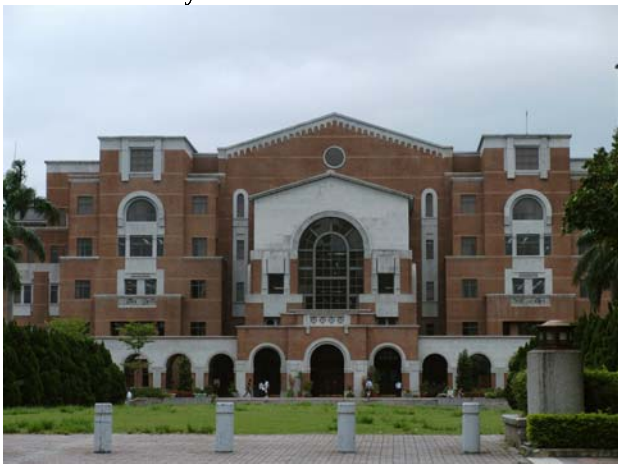


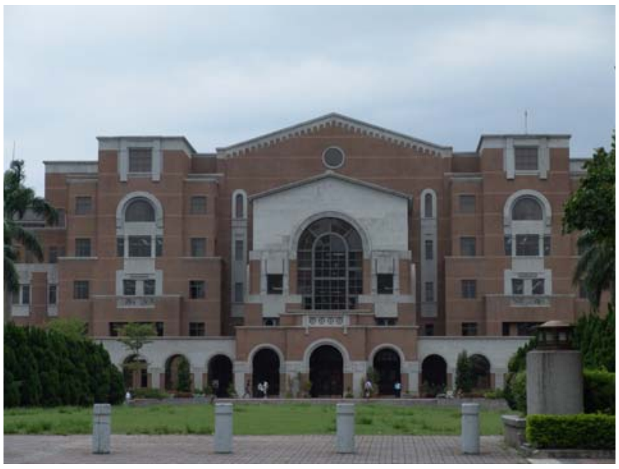
Our method: 17 votes.



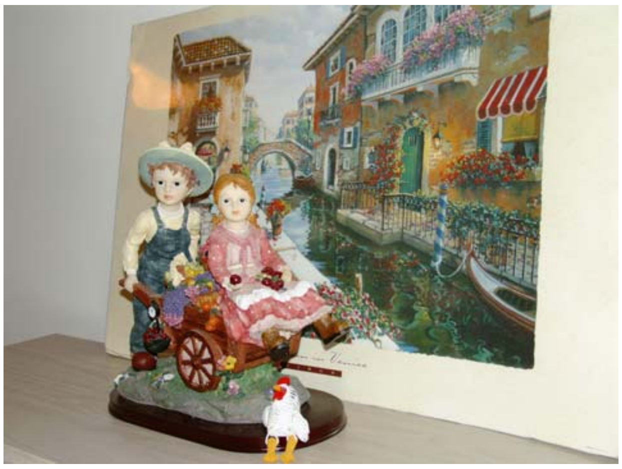
Sony DSC-F828: 4 votes.

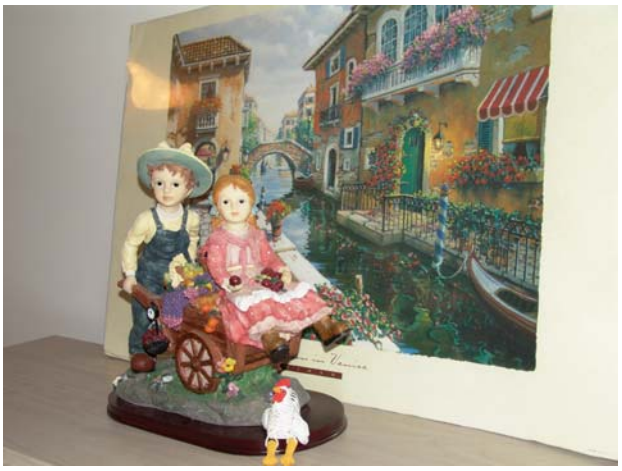
8.2 NTU Library


Our method: 19 votes.

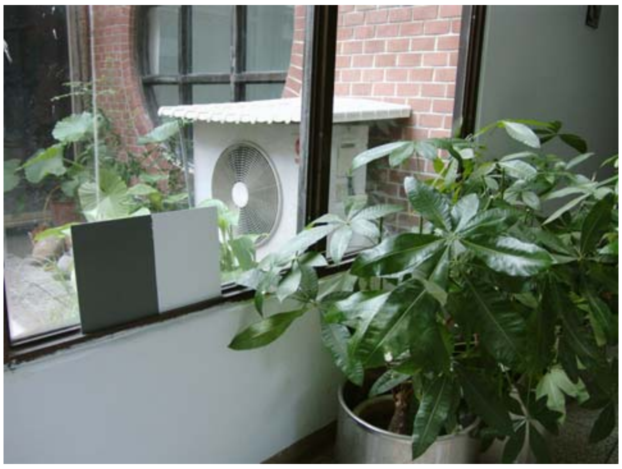

Sony DSC-F828: 2 votes.

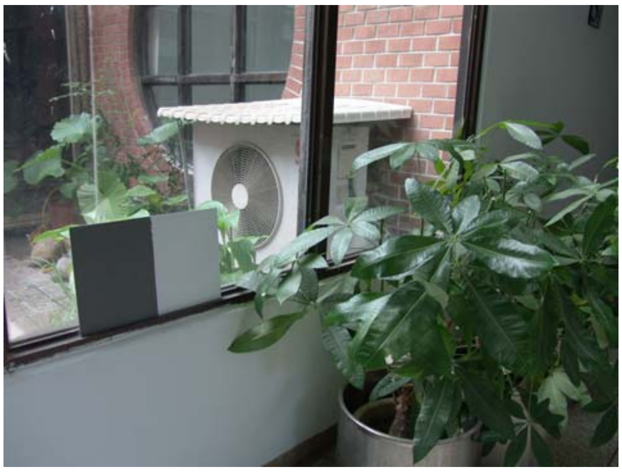
8.3 Dolls


Our method: 12 votes.

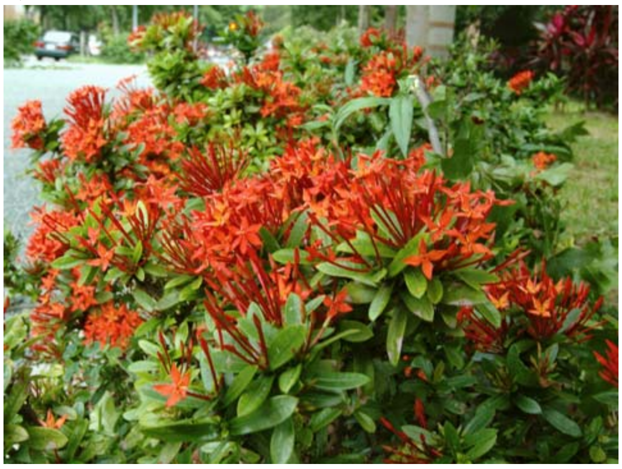

Sony DSC-F828: 9 votes.

8.4 Plants

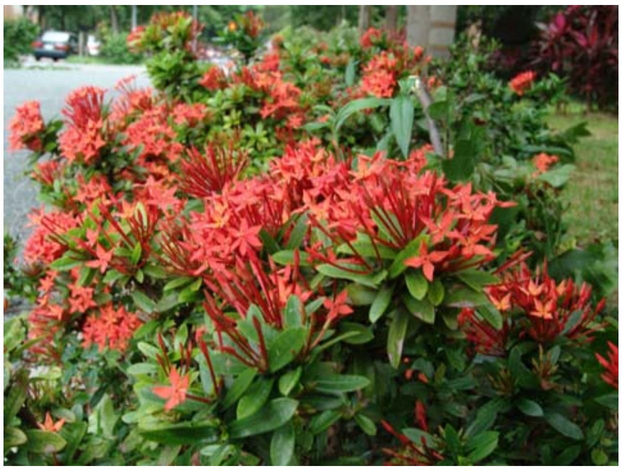

Our method: 19 votes.


Sony DSC-F828: 2 votes.

8.4 Flower



Our method: 13 votes.

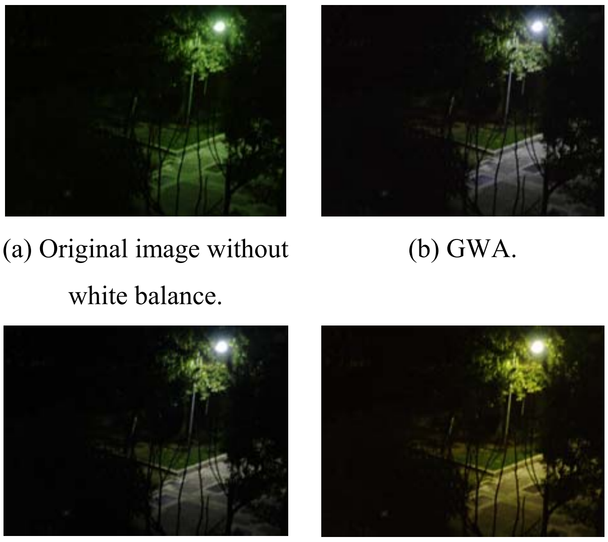


Sony DSC-F828: 8 votes.

9. CONCLUSION AND FUTURE WORK

9.1 Conclusion

In this thesis, we review each step of the traditional image pipeline algorithm and propose our image pipeline algorithm. Besides, we simulate the image pipeline algorithm with a C++ program and compare the results with Sony's images. Most of our images get more votes than Sony's images. However, some of our images are terrible that our images get 0 votes as shown in Figure 11.



(a) Original image without white balance. (b) GWA.

(c) Sony's image. (d) Our method.

Figure 5.1.1 Comparison of GWA, Sony's image, and our method.

9.2 Problems of Our Image Pipeline Method

Greenish Image:

Some of our images may become greenish because our white balance method is wrong under some situations. As shown in Figure 5.1.1, our white balance discards all pixels except the streetlight. That is, the streetlight should meet a gray value. Thus, the image is greenish as the original image is greenish.

Time Complexity:

Compared with the traditional image pipeline, our image pipeline algorithm involves three color spaces that would need more computational time.

Parameter Adjustment:

We use many parameters in our image pipeline algorithm such as *upper_bound*, *lower_bound*, *threshold* in White Balance, *threshold* in Color Interpolation, β in Saturation Adjustment, and A in Edge Enhancement. We use 14-bit raw images on our experiments. If the platform changes, we may need to adjust our parameters.

9.3 Future Work

There are many steps of image pipeline that can be further improved such as White Balance, Color Interpolation, and Edge Enhancement. Besides, we can add more steps in image pipeline such as Tone Mapping and Contrast Enhancement. Above all, we should try to find more ways to improve the image.

We will implement the image pipeline algorithm on SMIA mobile host to test and verify our algorithm performance. Otherwise, we will modify the algorithm accordingly.

10. ACKNOWLEDGEMENT

This research was supported by the National Science Council of Taiwan, R.O.C., under Grants NSC 94-2213-E-002-032 and NSC 93-2213-E-002-073, by the EeRise Corporation, EeVision Corporation, Machvision, Tekom Technologies, IAC, ATM Electronic, Primax Electronics, Scance, Lite-on and Liteonit.

11. REFERENCES

[1] F. Al-Turkait, “Specular Reflection,” http://ise.stanford.edu/class/psych221/projects/99/fahad/Spec_Ref_Intro.html, 1999.
[2] B. E. Bayer, “Color Image Array,” U.S. Patent 3971065, 1976.
[3] Y. C. Cheng, W. H. Chan, and Y.Q. Chen, “Automatic White Balance for Digital Still Camera,” *IEEE Transactions on Consumer Electronics*, Vol. 41, pp. 460-466, 1995.
[4] J. Chiang, “Gray World Assumption,” http://ise.stanford.edu/class/psych221/projects/99/jchiang/intro2.html, 1999.
[5] C. Y. Foo, “Introduction to Color Space,” Digital Camera and Computer Vision Laboratory, Department of Computer Science and Information Engineering, National Taiwan University, 2003.
[6] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Prentice-Hall, New Jersey, 2002.
[7] P. H. Mei, “Introduction to Digital Signal Processor,” Digital Camera and Computer Vision Laboratory, Department of Computer Science and Information Engineering, National Taiwan University, 2004.
[8] M. Peterson, “White Balance,” http://www.nikondigital.org/articles/white_balance.htm, 2004.
[9] SMIA, “Standard Mobile Imaging Architecture,” http://www.smia-forum.org, 2004.
[10] J. Waltman, “Filter: Custom Convolution,” http://www.jasonwaltman.com/thesis/filter-convolution.html, 2001.
[11] Y. M. Wu, “Color Interpolation for Single-CCD Color Camera,” Digital Camera and Computer Vision Laboratory, Department of Computer Science and Information Engineering, National Taiwan University, 2001.