AUTOMATIC FACE COLOR ENHANCEMENT

Da-Yuan Huang (*黃大源*), *Chiou-Shan Fuh* (*傅楸善*)

Dept. of Computer Science and Information Engineering, National Taiwan University
E-mail: r97022@cise.ntu.edu.tw

ABSTRACT

Face color enhancement is a very important issue in digital camera applications. However, to adjust photos appropriately is hard work to most light users. This paper suggests a general and automatic method to enhance the human face colors.

1. INTRODUCTION

In this paper, we introduce a pipeline for automatic skin color enhancement. We also combine some contrast and photo color enhancement to beautify the whole scenes of an input image.

Digital camera technique is very popular now. Unlike traditional photograph, digital images are able to be processed and adjusted again by computer software. Using the information of raw data image and producing a high quality photograph is a major issue traditional image pipeline.

Skin and face detection is a popular research area nowadays because of its wide range of applications, such as digital camera and video phone, and sing graphics software. In order to beautify human skin in photos this technique is an important step for image processor.

To fulfill the requirement of normal digital camera users who are not familiar with digital image processing and graphic software, how to design an automatic image processing pipeline and output a good-looking photograph is an important and interesting question.

In typical digital image processing, automatic human skin color enhancement is manual and atypical. Preferable-looking human skin color is usually processed by graphics software, or other image processing tools. Although such software is powerful, it is not user-friendly or automatic to most camera users. Therefore it is still an important issue to build an automatic skin color enhancement pipeline. In our method, we try to implement algorithms to beautify the skin color automatically.

Human skin does not always look well because of unhealthy condition, such as furrow and acne. The final output may be unsatisfactory for the camera user. In this situation, we must amend or hide those blemishes on human skin without break non-skin details in the photo. Skin Detection method becomes an important work in our image processing method. And we apply some methods to balance the skin color.

Fig. 1 is our pipeline of Automatic Face Color Enhancement Method.

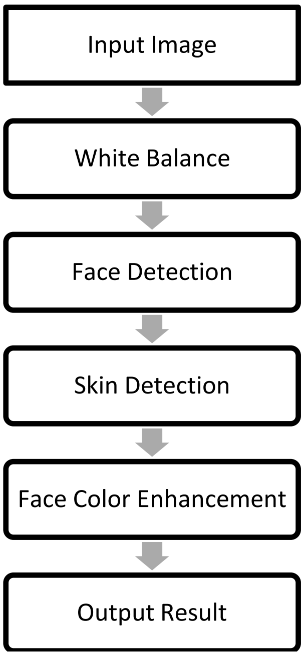


Fig. 1: Our pipeline of Automatic Skin Color Enhancement

2. EXTRACT FACE IMAGES

We assumed that user want to beautify human faces without effecting background in photograph.

Extracting human faces can let us justify human faces only. In this experiment, we simply use adaboost method in OpenCV library.

After set the face regions, we can define some rough sub-regions such as eye region and mouth region. These regions are useful to help as define skin color enhancement ratios later. To set sub-regions, we simply use fixed patterns to define it..

Fig. 2 is an example which detect human face and extract faces from an image.

Fig. 2: An example of finding human faces with fixed pattern

3. WHITE BALANCE

An image pipeline is an important step that transforms sensor's raw data to final compressed image. It balances the color temperature where image becomes bluish under high color temperature and becomes reddish under low color temperature[2]. A good white balance method can greatly improve the quality of final digital image. The example is as Figure 3.

For detecting the natural skin color and removing the effect of color temperature, White Balance is a necessary step. The thrust of our image pipeline focus on comparing the skin-beautify processing to other graphics software. We simply use white balance tools from Adobe Photoshop CS, pre-process our experiment images.

4. SKIN DETECTION

4.1 Skin Detection with Color Spaces

After extracting face images, we can promote the accurate detection by using different color space. Kumar and Bindu[1] suggests using YCbCr color space and LUX color space to find skin regions.
In our experiment shows that using YCbCr and LUX color spaces to detect skin region can lead good results .

4.2 YCbCr Color Space

The YCbCr video format is a linear combination of red, green, and blue components (RGB) used as TV standard. It is a family of color spaces used in video and digital photography systems. The prime on the Y is to distinguish the luma from luminance. YCbCr is sometimes abbreviated to YCC [9]. The formula of YCbCr color space is:

$$\begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.098 \\ 0.500 & -0.4187 & -0.0813 \\ -0.1687 & -0.3313 & 0.500 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 1 \\ 128 \\ 128 \end{bmatrix} \quad (1)$$

4.3 LUX Color Space

According to the paper, "Nonlinear Color Space and Spatiotemporal MRF for Hierarchical Segmentation of Face Features in Video", LUX color space is extended from LIP color space, For ***L***ogarithmic h***U***e e***X***tension [10].
The LUX color space is very sensitive in red component; it can extract human skin from background and from non-skin objects in photograph efficiently. The expression of LUX color components are the following:

$$L = (R+1)^{0.3}(G+1)^{0.6}(B+1)^{0.1} - 1$$
$$U = \begin{cases} \frac{M}{2}\left(\frac{R+1}{L+1}\right) & if\ R < L, \\ M - \frac{M}{2}\left(\frac{L+1}{R+1}\right) & otherwise. \end{cases} \quad (2)$$
$$X = \begin{cases} \frac{M}{2}\left(\frac{B+1}{L+1}\right) & if\ R < L, \\ M - \frac{M}{2}\left(\frac{L+1}{B+1}\right) & otherwise. \end{cases}$$

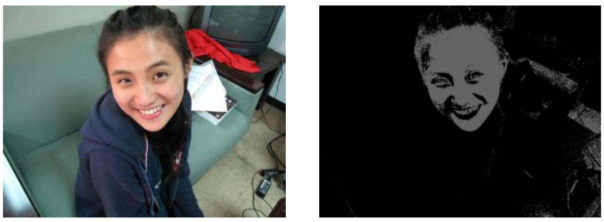
4.4 Definition of Skin Regions

Using Kumar's method, we can simply find skin region, because of the definition of face region, the skin region which not in the face region will not be changed.

In our method, we define "skin map" in following algorithm:

1. $U' = \begin{cases} 256 \times \frac{G}{R} & if\ \frac{R}{G} < 1.5\ and\ R > G > 0 \\ 255 & otherwise. \end{cases} \quad (3)$
2. If $77 \le Cr \le 127$ and $0 \le U \le 249$, the value of skin map is 1.0.
   Else skin map is U / Cr

Fig. 3 is an example of detecting skin region with YCbCr and LUX color space.



(a) (b)

Fig. 3: (a) is an input image, (b) is an example of skin region after skin detection. Gray pixels represent skin area.

5. FACE COLOR ENHANCEMENT

5.1 Split Base Layer and Detail Layer of Faces

We inspired by Bae's method in "Two-scale Tone Management for Photographic Look"[10], In our method, we thought that human faces can be split into base layer (perfect skin) and detail layer (acne or scar).

5.2 Fast Bilateral Filter

To accomplish the splitting, we implement bilateral filter, which is a filter used to blur image but contain edges of blurred image.

However, traditional bilateral filter can lead lots of computation and is not practical. To speed up our image pipeline, we implement fast bilateral filter.

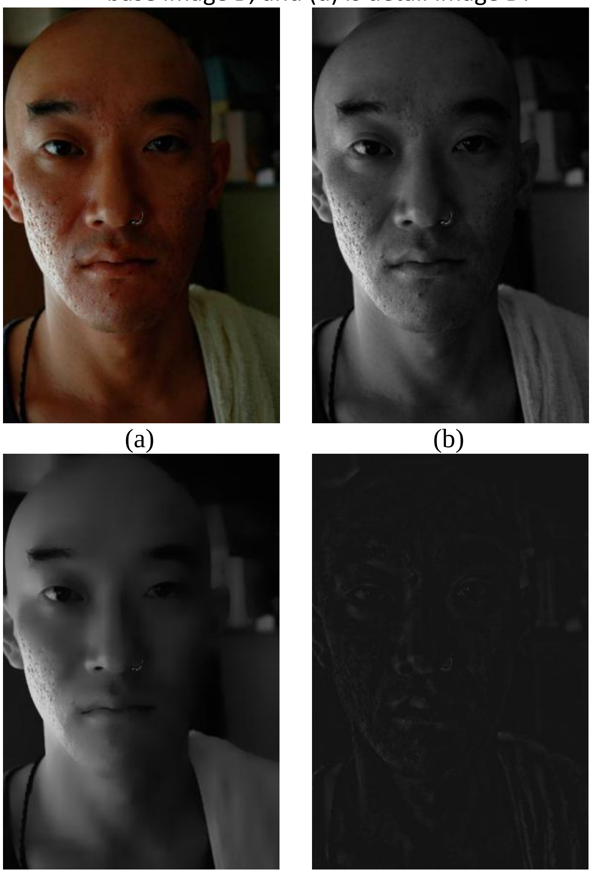
The following is our assumption in mathematic form, which F is original human face, B is base layer, and D is detail layer:

$B = fbs(F)$, where *fbs* is fast bilateral algorithm. (3)

$D = F – B$ (4)

Fig. 4 is an example of splitting a face into base layer and detail layer.

Fig. 4
(a) is original image, (b) is gray scale image, (c) is base image *B*, and (d) is detail image *D*.



(a) (b)

(c) (d)

Fig. 4: (a) is input image, convert to grayscale image as (b). (c) is base layer of (b) and (d) is detail layer of (b).

5.3 Skin Color Enhancement

Traditional method of smoothing skin is using user interface to blurred unsatisfied skin region step by step. However, over blurred is may be look cheerful, but can also lead the result look unreal.

In our enhancement method, we want to lower down the unsatisfied details, such as acne, scars…etc. Fortunately, although blemish is often unlike to neighbors, it is usually still in the skin region, based on this observation, we construct the following pseudo code in Fig.5.

Face Color Enhancement Algorithm:

1. Restore image I from input image.
2. Calculate face region map M_f.
3. Calculate skin region map M_s.
4. Calculate base layer B = $fbs(I)$.
5. Calculate detail layer $D = I – B$.
6. D = $\{d_1, d_2, d_3, …, d_n\}$, d_i is in pixel domain, $i \in \{1, …, n\}$.

   For each d_i,
   if (d_i is in skin region) and (d_i is in face region)
   then weaken the value of d_i and assign it to d_i'.

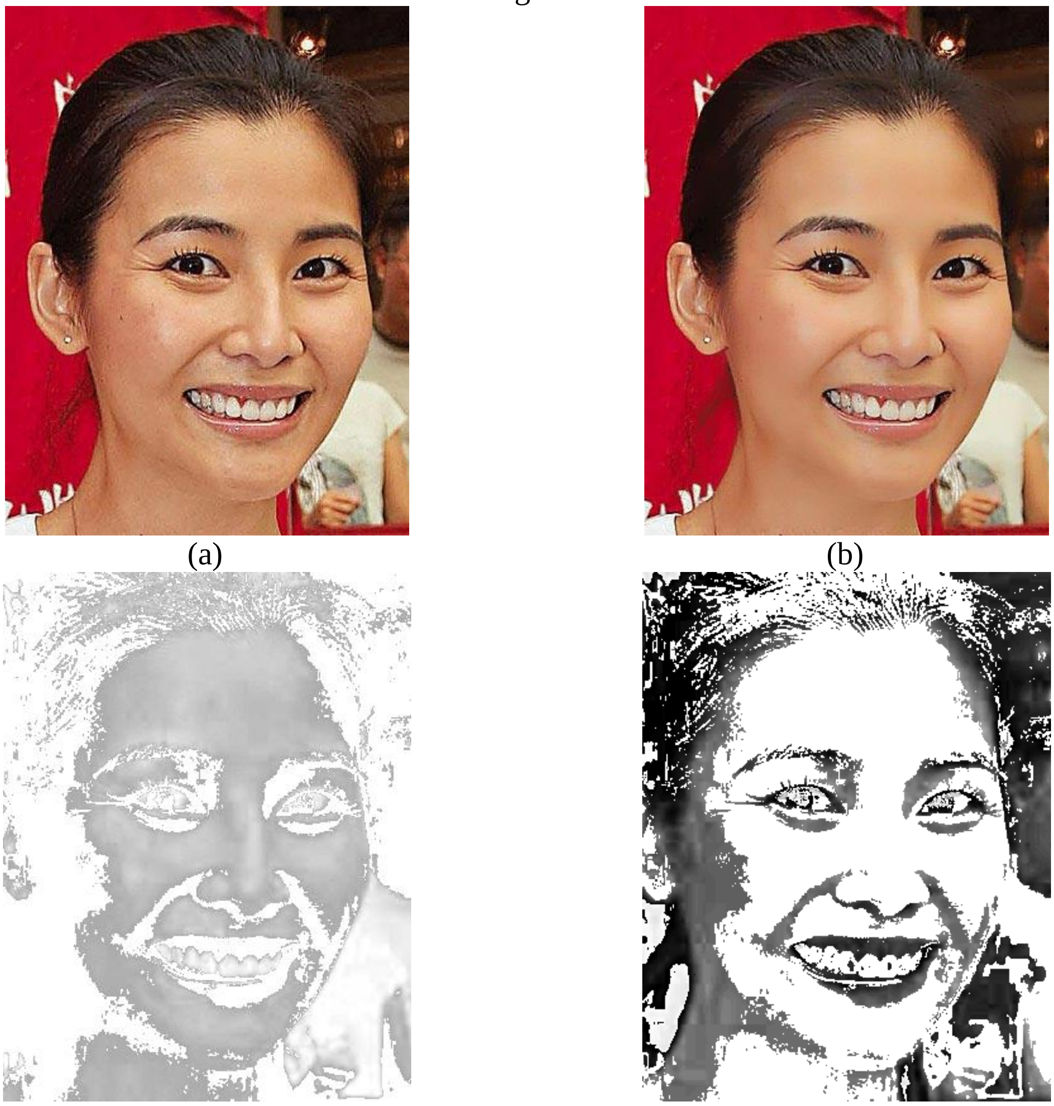
7. Set $D' = \{d_1', d_2', …, d_n'\}$. Ms = $\{s_1, s_2, …, s_n\}$.
   $d'_i = s_i * d_i$
8. Set $I' = B + D'$.
9. Return I'.

Fig. 5: The pseudo code of our image pipeline.

6. EXPERIMENT RESULT

Fig.5, Fig.6, Fig.7 and Fig.8 are the experiment results. For each figure, (a) is original image, (b) is output image, (c) is *U* channel map, and (d) is skin map.

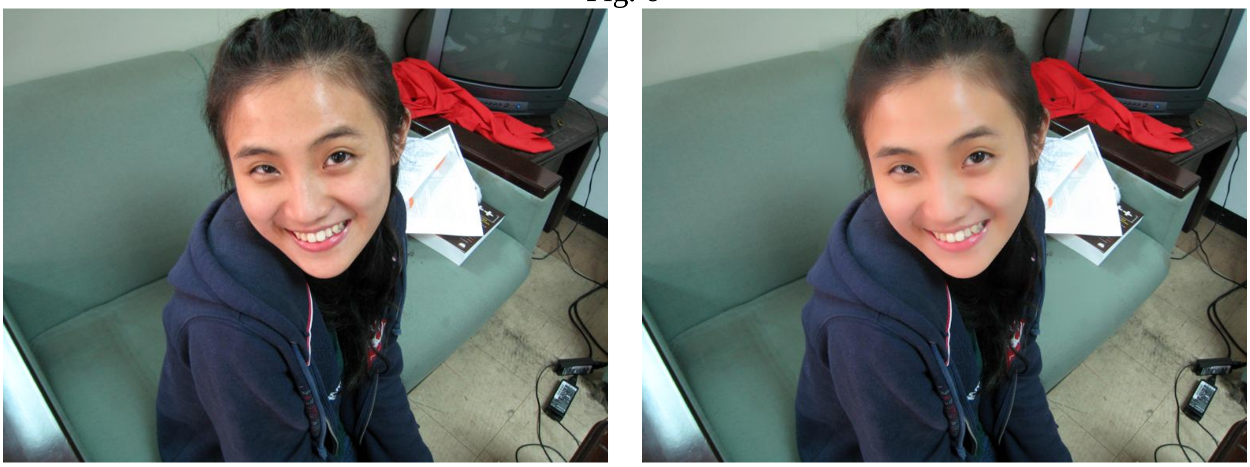
Fig. 5



(a) (b)

(c) (d)

Fig. 6



(a) (b)

(c) (d)

Fig. 7

(a) (b)

(c)

(d)

Fig. 8

(a)

(b)

(c)

(d)

REFERENCES

[1] C. N. R. Kumar and A. Bindu, “An Efficient Skin Illumination Compensation Model for Efficient Face Detection,” http://www.facedetect.741.com/, 2008.

[2] P. M. Wang and C. S. Fuh, “Automatic White Balance with Color Temperature Estimation,” Master Thesis, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, 2006.

[3] R. L. Hsu, M. Abdel-Mottaleb, and A. K. Jain, “Face Detection in Color Images,” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, No. 5, pp. 696-706, 2002.

[4] Wikipedia, “Gaussian Blur,” http://en.wikipedia.org/wiki/Gaussian_blur, 2008.

[5] Wikipedia, “Normal Distribution,” http://en.wikipedia.org/wiki/Normal_distribution, 2008.

[6] Wikipedia, “Scale Space Implementation,” http://en.wikipedia.org/wiki/Scale-space_implementation, 2008.

[7] Wikipedia, “RGB Color Space,” http://en.wikipedia.org/wiki/RGB_color_space

[8] Wikipedia, “YCbCr Color Space,” http://en.wikipedia.org/wiki/YCbCr

[9] Marc Liévin and Franck Luthon, Member, “Nonlinear Color Space and Spatiotemporal MRF for Hierarchical Segmentation of Face Features in Video,” IEEE, Marc Liévin and Franck Luthon, Member.

[10] BAE S., PARIS S., DURAND F. “Two-scale tone management for photographic look”. In SIGGRAPH ’06: ACM SIGGRAPH 2006 Papers (New York, NY, USA, 2006), ACM Press, pp. 637–645.