Face Beautification and Color Enhancement with Scene Mode Detection

Da-Yuan Huang
Dept. of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
r97022@csie.ntu.edu.tw

Chiou-Shann Fuh
Dept. of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
fuh@csie.ntu.edu.tw

***Abstract*—Auto photo beautification algorithms play important roles of promoting the quality of photograph because of the popularity of digital image production such as digital camera and printer.**

In this thesis, we develop a pipeline which combines automatic human face beautification and automatic color enhancement. The goal of this thesis is to beautify human faces and enhance the detail and color of the background of photos. The enhancement methods must produce steady and harmonious results quickly, so that our methods can be implemented on printer and used by users.

Finally, we will select the best-match scene mode based on the color distribution of input image. The printer can use the corresponding profile with the information of the scene; it will make the print-out photos more colorful and vivid.

Keywords-face; color; beautification; enhancement

I. INTRODUCTION

Nowadays, digital cameras replace film cameras gradually. Users can not only capture scenes but also modify it easily. People use digital still camera and other image recorder to capture scenes and convert to digital image information. Printers or monitors on the other hand, are used to display the image information. In order to display pleasing effect for printers or monitors, image enhancement is essential for users.

Humans use above digital instrument to record their activity and daily life, but they also want to make their photographs or image data can be displayed with beauty and elegance. Due to the progress of hardware, today we can get clear image information easily, and advanced image enhancement method such as human face beautification and image color enhancement become more and more important because they can help users get pleasing image from the raw image information.

We will transform input image from RGB color space to HSV color space and use Value channel as the information of luminance. The following methods of enhancement transform the luminance into luminosity, and will process the Value channel only.

Figure 1. presents the pipeline of our proposed method.

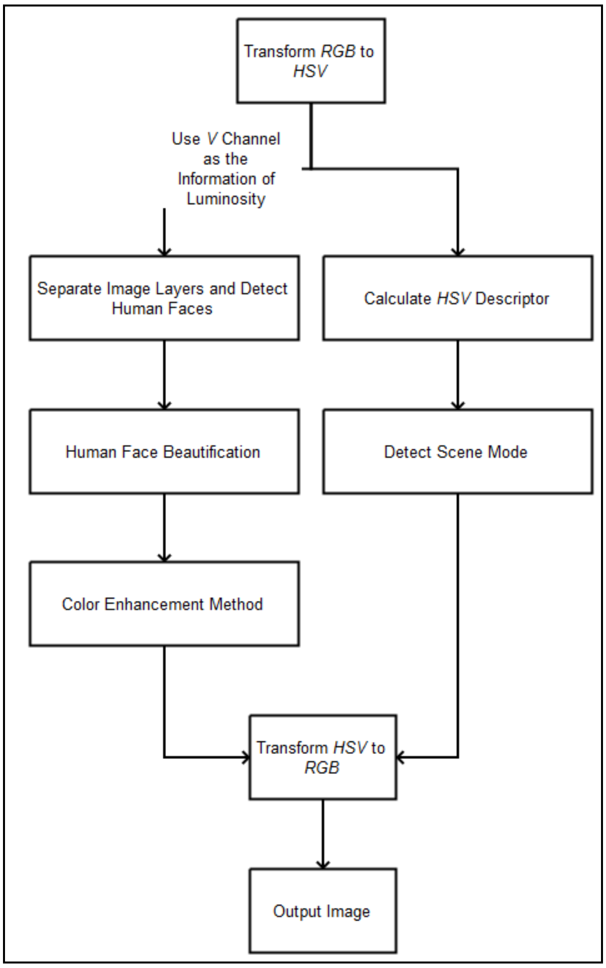

Transform *RGB* to *HSV*
Use *V* Channel as the Information of Luminosity
Separate Image Layers and Detect Human Faces
Calculate *HSV* Descriptor
Human Face Beautification
Detect Scene Mode
Color Enhancement Method
Transform *HSV* to *RGB*
Output Image

Figure 1. Our pipeline of our proposed method.

We will transform input image from RGB color space to HSV color space and use Value channel as the information of luminance. The following methods of enhancement transform the luminance into luminosity, and will process the Value channel only.

II. Related Works

A. Logarithm Curve

Logarithm curve obeys the Weber-Fechner law of JND response in human vision. The following is the formula of a simple logarithm curve form.

$$I'(x,y) = \frac{\log\,(I(x,y) \times (\beta - 1) + 1)}{\log\,(\beta)} \tag{1}$$

where $I'(x, y)$ is processed image; $I(x, y)$ is original image; and is the parameter which controls the brighten level.

B. Bilateral Filter

A bilateral filter is an edge-preserving smoothing filter. Whereas many filters are convolutions in the image domain, a bilateral filter also operates in the image's range—pixel values. Rather than simply replacing a pixel's value with a weighted average of its neighbors, as for instance the Gaussian filter does, the bilateral filter replaces a pixel's value with a weighted average of its neighbors in both space and range (pixel value). This preserves sharp edges by systematically excluding pixels across discontinuities from consideration [14].

$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|p - q\|)\, G_{\sigma_r}(\|I_p - I_q\|)\, I_q \tag{2}$$

where is the intensity of processed pixel; is the neighbor intensities of pixels in domain *S*; *p*, *q* are the positions of processed pixel and neighbor pixels; measures the spatial closeness between *p* and *q*; determines the intensity closeness between *p* and *q*; and is the normalization term which makes sure that the total sum of the weight of all pixels in domain *S* equals one.

The fast versions of bilateral filter are many, in this thesis, we implement [10]'s version of fast bilateral filter to speed up our processing time.

C. *β* Map

S. C. Tai et al [6] used the information of edge and logarithmic curve to enhance LDR images. The following equations are Step 1 of their proposed method for LDR image input:

$$v(x,y) = \frac{\log\,(w(x,y) \times (\beta - 1) + 1)}{\log\,(\beta)} \tag{3}$$

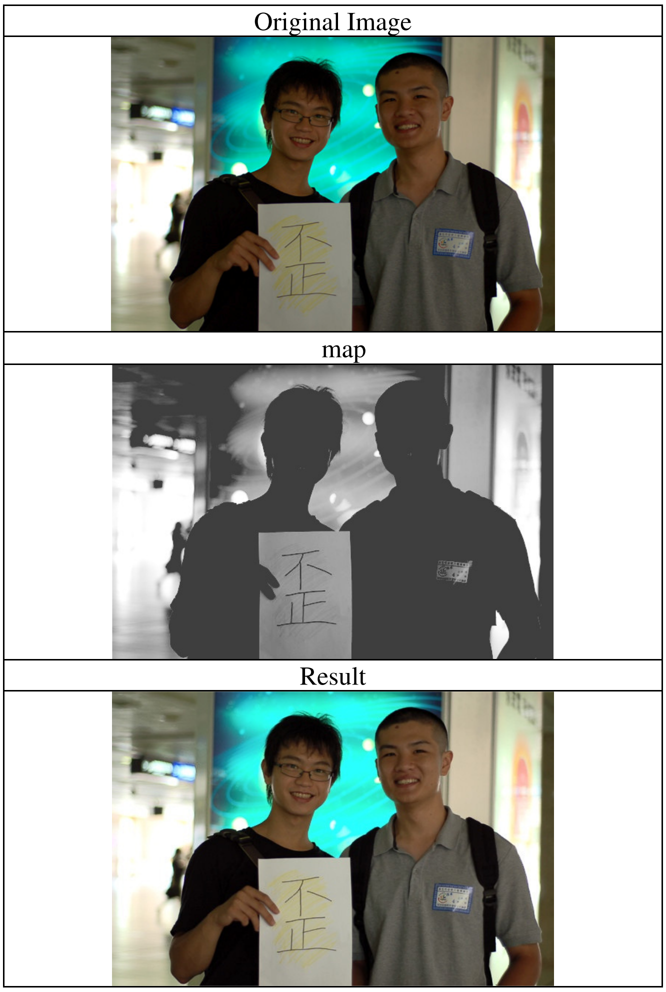
$$\beta = \begin{cases} 10 \times \dfrac{w(x,y) - mean}{w(x,y)} + 2, & w(x,y) \geq mean \text{ and } Sobel(x,y) < th \\ 2, & w(x,y) \geq mean \text{ and } Sobel(x,y) < th \\ undefined, & \text{otherwise} \end{cases} \tag{4}$$

where is the processed pixel data; is the original pixel data; *mean* is the average of intensity; *Sobel*(*x*, *y*) is the Sobel edges calculated with Sobel filter; *th* is the threshold of the value of Sobel edges; and is the parameter used by logarithmic function.

In Step 2, the *undefined* is a pixel where *Sobel*(*x*, *y*) is larger than *th*. In the second step, if the *β* value of edge pixel is undefined, a mask will be centered around the pixel, and the neighbors of center pixel will be found. We will set *β* value to the center pixel from one of *th* neighbor pixels whose luminance value is the closest to the center pixel.

After Steps 1 and 2, "*β* map" is set. We can use the map and logarithmic curve to complete the local contrast enhancement.

Figure 2. is an example of creating a *β* map of an image and the enhancement result.



Original Image

map

Result

Figure 2. An example of creating a *β* map of an image and the enhancement result.

III. Our Proposed Method

Our algorithm aims to reach three goals. First is to beautify human faces in photographs, reduce their wrinkles, pores, and other blemishes without causing obvious artifacts.

Second is to enhance image contrast without losing details of image. Although some contrast enhancement methods can produce strong effect on local contrast enhancement, they are not suitable for most cases.

Sometimes the contrast becomes too strong and loses details of result image and looks unnatural. Such methods are unsuitable for printer or monitor to use, because they cannot always provide improved results. We want our algorithm suitable for most cases, enhancing image with harmony.

Figure 3. is the pseudo code of our whole proposed method, where *PhotoEnhancer* is the function containing all of our proposed method; *img* is input image; *RGB2HSV(hsv, img)* transforms *img* from RGB color space to HSV color space to *hsv*; *ColorDescriptor(hsv)* calculates HSV color histogram of *hsv*; *Split(hsv, h, s, v)* separates hsv into its corresponding channels *h, s, v*; *luminance* is the luminance information of *img*; *BilateralFilteringAndFaceFinding(luminance, base, detail, faces_pos)* applies bilateral filter on *luminance* and calculates base layer to *base* and detail layer to *detail*; *faces* restores the positions of human faces in *luminance; FaceBeautification(luminance, base detail, faces_pos)* beautifies faces of *luminance* with information *base* and *detail*; *ColorEnhancement(luminance, base, detail)* modifies *luminance* with information *base* and *detail*; *Merge(h, s, v)* merges *h*, *s*, *v* into *hsv*; *HSV2RGB(hsv)* transforms *hsv* from HSV color space to RGB color space; and *DetectSceneMode(descriptor)* finds the best match mode by *descriptor* and returns the result to *mode*.

```
PhotoEnhancer(img){
   hsv = RGB2HSV(img)
   Split(hsv, h, s, v)
   descriptor = ColorDescriptor(h, s, v)
   luminance = v
   BilateralFilteringAndFaceFinding(luminance, base,
detail, faces_pos)
   luminance = FaceBeautification(luminance, base, detail,
faces_pos)
   luminance = ColorEnhancement(luminance, base, detail)
   v = luminance
   hsv = Merge(h, s, v)
   HSV2RGB(hsv, img)
   mode = DetectSceneMode(descriptor)
   return (img, mode)
}
```

Figure 3. Pseudo code of our whole proposed method.

A. Human Face Beautification

1) Separate Image Layers and Detect Human Faces

We usually separate image into many layers. The goal of our enhancement methods is to enhance the quality of image without losing harmony. Inspired by S. Bae [3], we separate an image into base layer and detail layer by using bilateral filter [].

The following equation (5) and (6) define the meanings of layers, where presents the result after *Image* processed bilateral filter, and presents fast bilateral filter algorithm [], presents the result of subtraction between Image and *Base*

$$Base = BF(Image) \tag{5}$$

$$Detail = Image - Base \tag{6}$$

Lienhart et al. [15] proposed Haar-like features for object detection. Moreover, we apply Lienhart's frontal face classifier to set the positions of human faces.

2) Human Face Beautification

To beautify faces, we extract the faces found by Haar-like features. The next steps are finding skin map, smoothing skin, and blending faces back to luminance.

Figure 4. is the pseudo code of human face beautification where Extract extracts faces of image by using *faces_pos*, a set which records all of the positions of human faces; faces is the set which extracts faces from luminance; *faces_base* and *faces_detail* are the sets extracted from base layer and detail layer; *FindSkinMap*(*faces*[*i*], *th_x*, *th_y*) finds the skin pixels of one of faces and stores it to *skin_map*; *th_x* and *th_y* are the thresholds to find skin pixels; *Smoothing*(*faces*[*i*], *face_base*[*i*], *face_detail*[*i*], *skin_map*) smoothes *faces*[*i*] with *face_base*[*i*], *face_detail*[*i*] and *skin_map*; and Blending(*luminance*, *faces*[*i*]) blends beautified face *face*[*i*] back to *luminance*.

```
FaceBeautification(luminance, base, detail, faces_pos){
   faces = Extract(luminance, faces_pos)
   faces_base = Extract(base, faces_pos)
   faces_detail = Extract(detail, faces_pos)
   for(i=1, i<faces.number, ++i){
      skin_map = FindSkinMap(faces[i], th_x, th_y)
      Smoothing(faces[i],
faces_base[i],faces_detail[i],skin_map)
      Blending(luminance, faces[i], faces_pos[i],
faces_detail[i])
   }
   return luminance
}
```

Figure 4. The pseudo code of human face beautification.

3) Generating Skin Map:

After setting the positions of human faces, we have to select and modify possible skin pixels.

The pseudo code of generating Skin Map is in Figure 5. where Sobel is the function of applying Sobel filter on face in x and y directions; face is the facial region of luminance; sobel_x and sobel_y are the results after the face applied Sobel filter in x-direction and y-direction; th_x and th_y are the thresholds of smooth pixels; and skin_map records the result of our skin detection method. In our thesis, we set th_x = th_y = 15.0 by default. Figure 6. is the result after our skin detection method.

```
FindSkinMap(face, th_x, th_y){
   sobel_x = Sobel(face, x-direction)
   sobel_y = Sobel(face, y-direction)
   for(row=1; row <= face.height; ++row){
      for(col=1; col <= face.width; ++col){
         x_value = sobel_x(row, col)
         y_value = sobel_y(row, col)
```

```
        if(x_value<th_x & y_value<th_y & $x\_value^2+y\_value^2<th\_x^2+th\_y^2$)
            skin_map(row, col) = 1.0
        else
            skin_map(row, col) = 0.0
      }
   }
   return skin_map
}
```

Figure 5. The pseudo code of human face beautification.

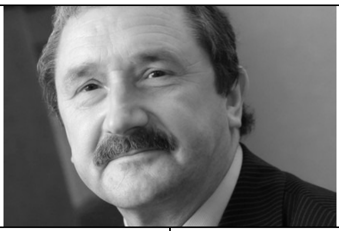
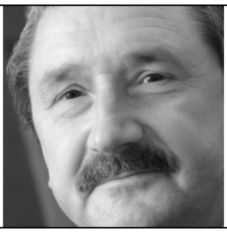
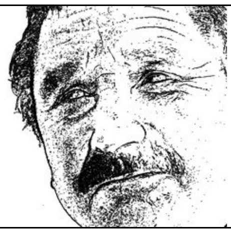
<table>
<tr><td colspan="2">Luminance Information</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Face Image after Haar Face Detection</td><td>Skin Map</td></tr>
<tr><td></td><td></td></tr>
</table>

Figure 6. The result after our skin detection method.

4) *Beautification and Blending:*

In our previous work [5] [8], we separate human faces into two layers: smooth skin layers and blemish layers. Therefore, we can assume that the base layer of human faces contains smooth skin pixels, while detail layer presents blemishes on human faces.

To reach the goal of human face beautification, we can reduce the magnitude of detail. The pseudo code of our smoothing method is in Figure 7. Moreover, Figure 8. is a result of this process where α is the ratio parameter. We set α = 0.3 by default in this thesis.

```
Smoothing(faces[i], face_base[i], face_detail[i], skin_map)
{
   for(row=0, row < face.height, ++row){
      for(row=0, row < face.width, ++col){
         if(skin_map(row, col) == 1)
            face(row, col) = face_base(row, col)+α*(face_detail, row, col)
         else
            /* remain the same*/
      }
   }
}
```

Figure 7. The result after our skin detection method.

Before Face Beautification	After Face Beautification

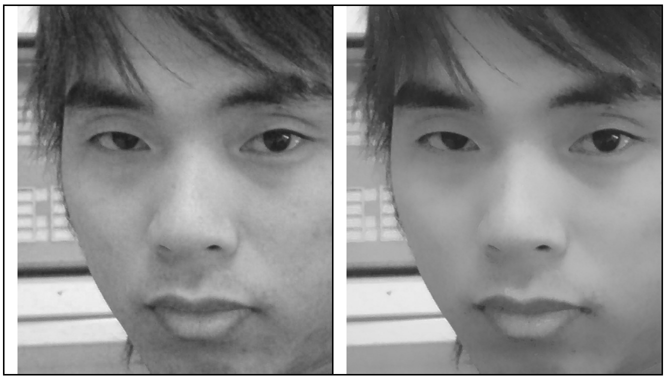


Figure 8 A result of face beautification.

Figure 9. is our pseudo code for blending where center records the central position of face; mask_map records the ratio of linear blending, our blending method prevents artifacts of the facial edge after blending beautified faces back to original image.

```
Blending(luminance, face, face_pos, , faces_detail){
   center = face.center
   for(row=face_pos.y, row<(face_pos.y+face.height), ++row){
      for(col=face_pos.x, row<(face_pos.x+face.width, ++col){
         mask_map(row, col) = 1.0-(row-center.y)/(face.height/2.0)*(col-center.x)/(face.width/2.0)
         luminance(row, col) = face(row-face_pos.y, col-face_pos.x)+(1.0- mask_map(row, col))*face_detail(row, col)
      }
   }
}
```

Figure 9 The pseudo code for blending.

B. *Image Color Enhancement*

Since we already know that separating image into layers is convenient for image enhancement, we try to use this spirit on generating map.

S. C. Tai [12] use Sobel edge pixels to achieve local contrast enhancement, which means its map separates image into edge pixel layer and non-edge pixel layer. We extended this approach by using base layer and detail layer. Since base layer already contains the information of neighbors due to Gaussian Blur of space and intensity of each pixel, we can find best match intensity value by checking value on base layer for detail pixels without checking all the neighbor pixels.

Figure 10. is our pseudo code of generating β map. We separate β map into four layers to observe:

- pixels with weak base and weak detail
- pixels with strong base and weak detail
- pixels with weak base and strong detail
- pixels with strong base and strong detail

where *avg* and *sdv* are the average and standard deviation of *luminance* calculated by function *CalAvgSdv*;

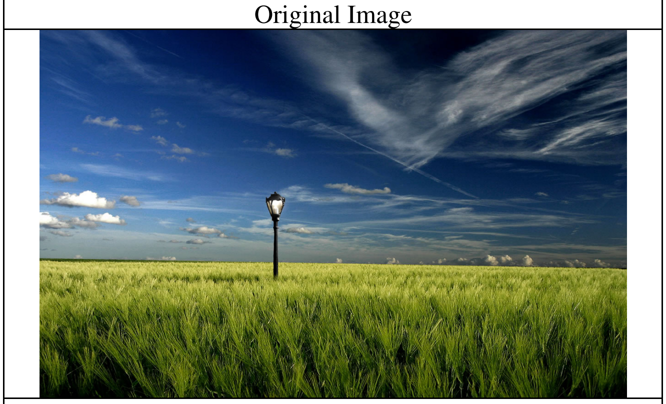
beta_default is the default ; *max_beta* and *min_beta* are the lower bound and the upper bond of β; *beta* is the final β value of β map; and *p1* and *p2* are the constants to adjust *min_beta* and *max_beta*. In this thesis, we set *p1* = 1.2, *p2* = 1.5 by default.

```
ColorEnhancement(luminance, base, detail){
   (avg, sdv) = CalcAvgSdv(luminance)
   min_beta = p1 + (sdv - avg);
   max_beta = min_beta + p2 * (1.0 + sdv - avg)
   for(row=0, row < luminance.height, ++row){
      for(row=0, row < luminance. width, ++col){
         if(base(row, col)<avg & abs(detail)<th)
            beta = luminance(row, col) * (max_beta–min_beta) + min_beta
         else if(base(row, col)>avg & abs(detail)<th)
            beta = luminance (row, col) * (max_beta – min_beta) + min_beta
         else if(base(row, col)<avg & abs(detail)>th)
            beta = base(row, col) * (max_beta – min_beta) + min_beta
         else //base(row, col)>avg & abs(detail)>th
            beta = base(row, col) * (max_beta – min_beta) + min_beta
         LogarithmicCurve(luminance, beta)
      }
   }
   return luminance
}
```

Figure 10. The pseudo code for color enhancement.

Figure 11. is an example of generating map. After generating map, we simply use logarithmic curve function to finish the color enhancement.

Original Image



Result Image

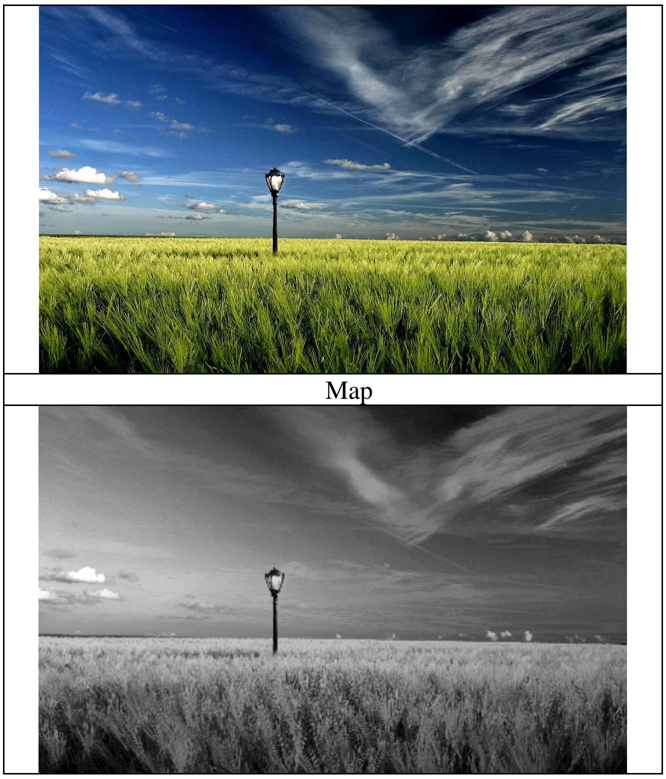


Figure 11. The pseudo code for color enhancement.

C. *Scene Classification*

We are inspired by the 'scene mode shooting' in many cameras. Users can select appropriate scene mode and take pictures, such as forest mode, sunset mode, ocean mode, and so on.

Our goal is to automatically detect the best-match scene mode for input image. After scene mode detection, displays such as monitors or printers can use corresponding scene modes or profiles to present the processed results.

The flow of scene mode detection contains two parts: training and classification.

1) *Color Descriptor*

To present color histogram of an image efficiently, J. R. Smith [11] provides a color descriptor by using the information of HSV color space of an image. We calculate 3D color histogram of HSV with 18 bins in H channel, 3 bins in S channel, and 3 bins in V channel.

Figure 12 is the pseudo code of creating Color Descriptor where (*h_pos*, *s_pos*, *v_pos*) is the coordinate of 3D histogram of HSV; *h_bin*, *s_bin*, and *v_bin* are the bins of 3D histogram; descriptor records the information of 3D histogram of HSV; and Scale normalizes the 3D histogram by dividing total pixel number of input image.

```
ColorDescriptor(h, s, v){
   for(row=0, row<hsv.height, ++row){
      for(col=0; col<hsv.width, ++col){
         h_pos = Floor(h(row, col) / 360.0 * h_bin)
         s_pos = Floor(s(row, col) / 1.0 * s_bin)
```

```
            v_pos = Floor(v(row, col) / 1.0 * v_bin)
            descriptor(h_pos, s_pos, v_pos) += 1.0
        }
    }
    Scale(descriptor, h.width * h.height)
    return descriptor
}
```

Figure 12 The pseudo code for color descriptor.

2) *SVM Training and Scene Mode Classification*

To classify an image scene with high accuracy, we simply use C. J. Lin's [4] SVM libraries to train our database of different scenes. We focus on three scenes: cool color mode, warm color mode, and forest color mode.

We download hundreds of images from flickr [17] for each main theme. We first resize images to 700 pixels by 700 pixels, then calculate their color descriptor as training data. Finally we use C. J. Lin's program [4] to finish training and complete its classification.

Database are collected from flickr. Each image will have a color descriptor, and each color descriptor will have a label. SVM machines will use these descriptors and labels to complete the training process.

IV. Experimental Results

Here we introduce our experiment environment:

CPU	Intel Core2 Duo
Memory	3.25 GB
Operating System	Windows 7 Enterprise Edition
Programming Language	Visual Studio 2008 + OpenCV 2.0

Our method takes 1.71 second to complete total image enhancement and scene mode detection on an image with resolution 2592 by 1944 pixels. The following figures are the comparison of original images and the results.

Figures. 13 to 14 are the comparison between original image and the results after our face color enhancement.

To compare our color enhancement method, we implement S. C. Tai's LDR contrast enhancement method [12]. We also download George DeWolfe's Photoshop Plug-in [6] called PercepTool and use its No-UI Filter to complete his color enhancement method. Figures. 15 to 16 are the comparison between original image, our color enhancement image, image after S. C. Tai's [12] contrast enhancement, and the result by using Percept Tool of George Dewolfe.

The test data are from database of Hiti Digital [7] and flickr. Our correct rate of scene mode detection is about 87.10% from test database with 30 images totally.

V. Conclusion and Future Work

A. *Conclusion*

Our proposed method provides faster and more effective method than post-production.

There are three major parts of our proposed method: human face beautification, color enhancement, and scene mode detection.

We beautify human faces in the picture and enhance its luminance information to luminosity. Finally we automatically select scene mode for printer that can be printed out by corresponding profile.

After our proposed method, blemish on human faces will be reduced, skin pixels on human faces become smooth and natural. On the other hand, the contrast of luminance and details will be enhanced. After all the proposed methods are done, input photograph will make good result both on human faces and luminosity efficiently.

B. *Future Work*

The most important thing in our future work is to provide a more precise face detection algorithm and blending method.

Furthermore, the accuracy of scene mode detection is not satisfactory for practical use. We have to find more effective color descriptor or classifier to improve the accuracy for scene mode selection.

VI. Reference

[1] Adobe Labs, "Photoshop CS3," http://labs.adobe.com/technologies/photoshopcs3/, 2010.

[2] Anthropics Technology, "Portrait Professional," http://www.portraitprofessional.com/, 2010.

[3] S. Bae, S. Paris, F. Durand, "Two-Scale Tone Management," http://people.csail.mit.edu/soonmin/photolook/, 2010.

[4] C. C. Chang, C. J. Lin, "LIBSVM -- A Library for Support Vector Machines", http://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html, 2010.

[5] C. W. Chen, D. Y. Huang, and C. S. Fuh, "Automatic Skin Color Beautification", *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, p. 157, 2009.

[6] G. Dewolf, "The Digital Fine Print," http://www.georgedewolfe.com/dfp.html, 2010.

[7] Hiti Digital, "Hiti," http://www.hiti.com/us/, 2010.

[8] D. Y. Huang and C. S. Fuh, "Automatic Face Color Enhancement," *Proceedings of IPPR Conference on Computer Vision, Graphics, and Image Processing*, Shitou, Taiwan, D6-8, p. 175, 2009.

[9] Literate Programs, "RGB to HSV Color Space Conversion (C)," http://en.literateprograms.org/RGB_to_HSV_color_space_conversion_(C)#chunk def:compute hue, 2010.

[10] S. Paris, P. Kornprobst, J. Tumblin, and F. Durand, "A Gentle Introduction

to Bilateral Filtering and its Applications," http://people.csail.mit.edu/sparis/bf_course/, 2010.

[11] J. R. Smith, "Chapter 11: Color for Image Retrieval," *Image Databases: Search and Retrieval of Digital Imagery*, Wiley-Interscience, New York, 2001.

[12] S. C. Tai, N. C. Wang, Y. Y. Chang, and Y. C. Lu, "A Two-Stage Contrast Enhancement Algorithm for Digital Images," *Proceedings of the IEEE Congress on Image and Signal Processing*, Sanya, Hainan, China, Vol. 3 - Volume 03, pp. 256-260, 2008.

[13] Wikipedia, "HSL and HSV," http://en.wikipedia.org/wiki/HSL_and_HSV, 2010

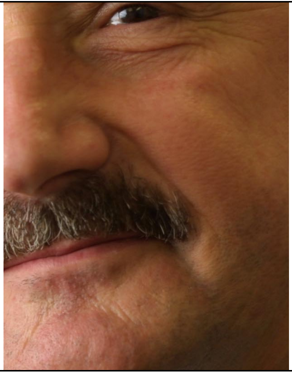
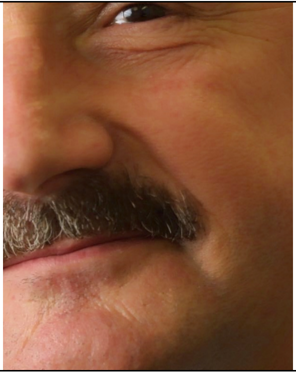
[14] Wikipedia, "Bilateral Filter," http://en.wikipedia.org/wiki/Bilateral_filter, 2010.

[15] Wikipedia, "Haar-like features," http://en.wikipedia.org/wiki/Haar-like_features, 2010.

[16] Wikipedia, "Histogram Equalization," http://en.wikipedia.org/wiki/Histogram_equalization, 2010.

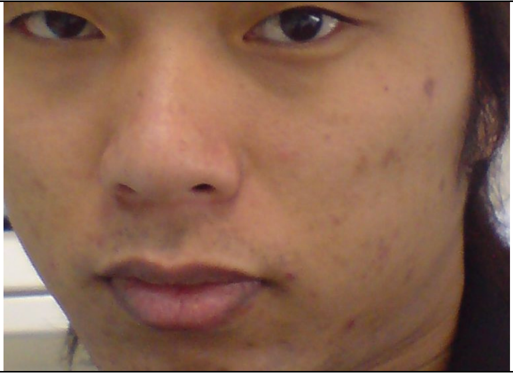
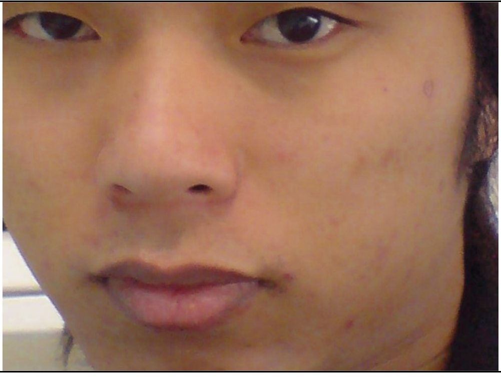
[17] Wikipedia, "Adaptive Histogram Equalization," http://en.wikipedia.org/wiki/Adaptive_histogram_equalization, 2010.

[18] Wikipedia, "Flickr," http://zh.wikipedia.org/zh-tw/Flickr, 2010.

Original Image (0 vote)	Our Result Image (21 votes)
	

Warm Color Mode

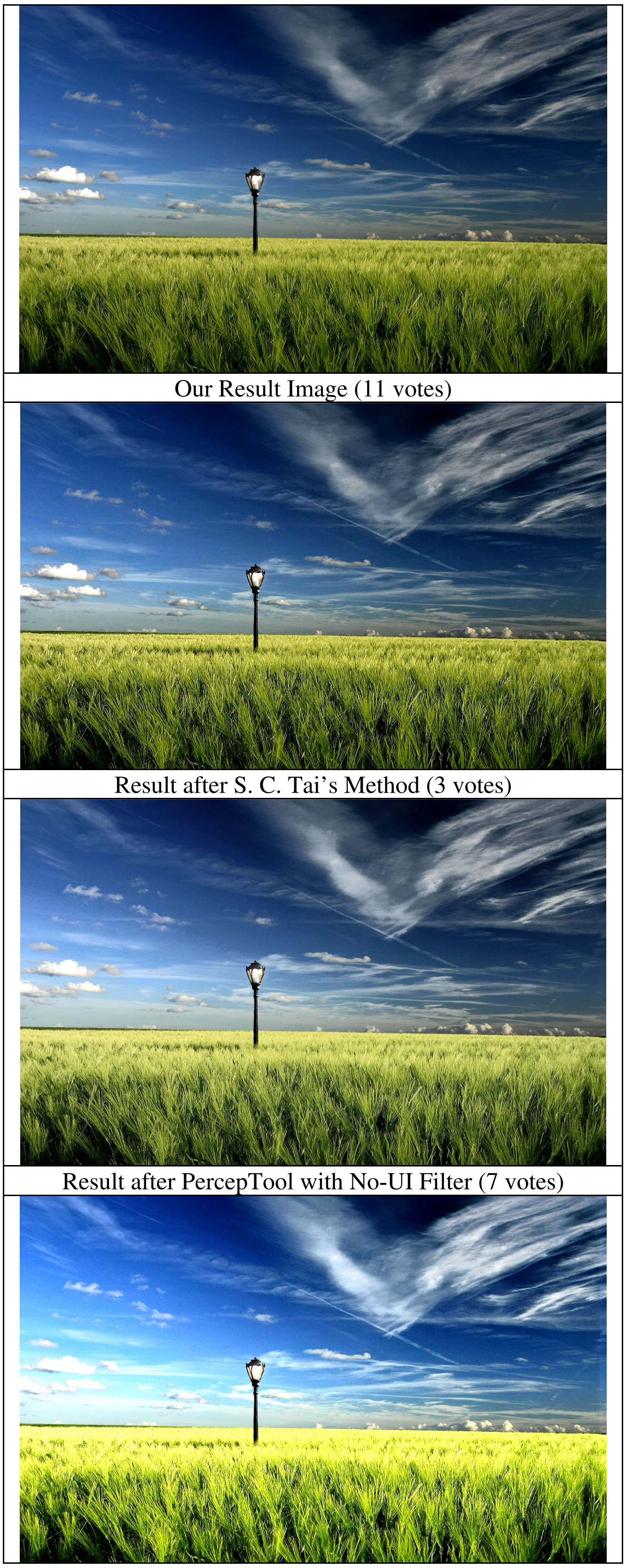
Figure 13. Our method smooth human wrinkles.

Original Image (0 vote)	Our Result Image (21 votes)
	

Warm Color Mode

Figure 14. Our method reduces blemishes on human face.

Original Image (0 votes)



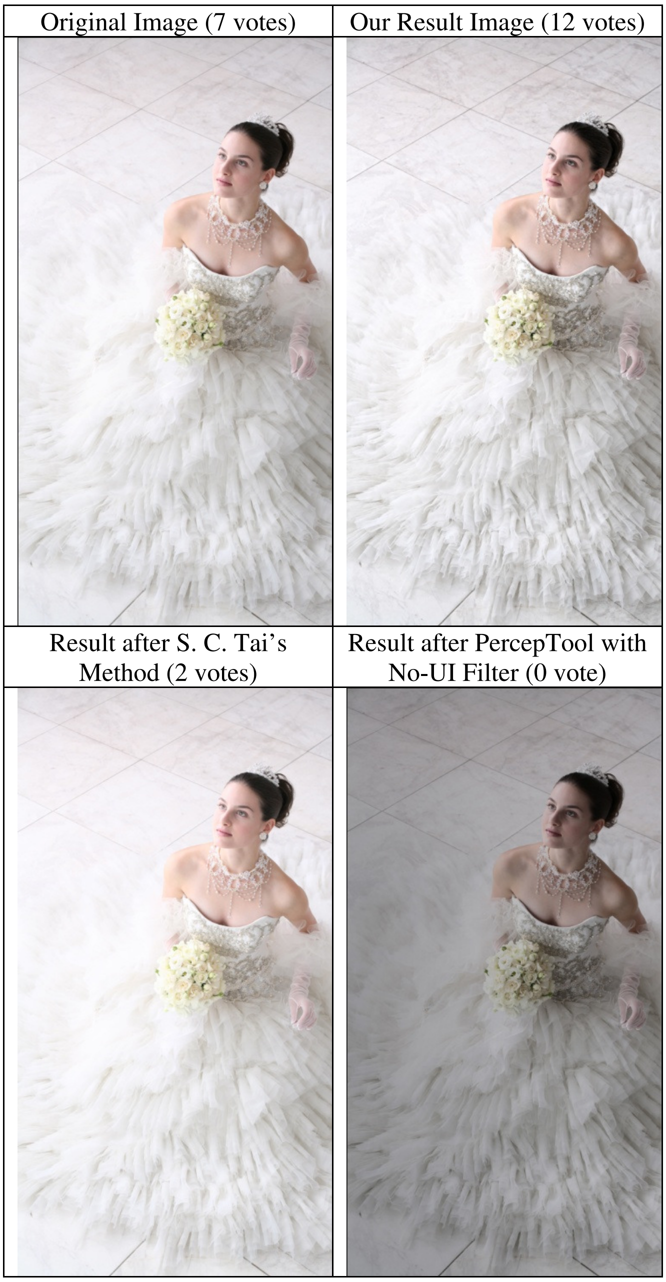
Our Result Image (11 votes)

Result after S. C. Tai’s Method (3 votes)

Result after PercepTool with No-UI Filter (7 votes)

Cool Color Mode

Figure 15. Our color enhancement method keeps the details of sky and grass, but S. C. Tai’s method and PercepTool will lose some details of sky and grass.



Original Image (7 votes)

Our Result Image (12 votes)

Result after S. C. Tai’s Method (2 votes)

Result after PercepTool with No-UI Filter (0 vote)

Figure 16. Our proposed method enhances local contrast of details while others might reduce the contrast of details.