Implementation of an edge detector on faces

Ruifeng Chen

ruifeng2@ualberta.ca

Abstract

Edges of human faces are important for facial landmarks detection or tracking points initialization. This paper will give several quick ways to detect edges on a face.
Codes are available at :https://github.com/RAYFC/edges-detection-on-a-face

Keywords: face detection,edge detection

1 Introduction

Face detection and edge detection are 2 very important parts of computer vision. As for face detection, there are many ways like CNN cascade[3] and skin color model [1][2][5], which both give good results. For edge detection, a large number of approaches exist like Roberts, Sobel,Prewitt,LOG,etc. Also, combining face detection and edge detection could be very interesting and in this paper, I focused more on implementing skin color extraction model and different kinds of edge detection including Roberts, Sobel, Prewitt, LOG.

2 Face detection

I implemented 2 ways of face detection : detecting faces by a deep model and by skin color model.To test them, I have the input image Figure 1.

Figure 1: input image

2.1 Deep model

I used the model which offered by the Matlab which uses the Viola-Jones detection algorithm and a trained classification model for detection.

2.2 Skin color model

- Color Spaces

  1. RGB
     An RGB color space is any additive color space based on the RGB color model.[4]The RGB color space is an additive color model in which the primary colors red, green, and blue light are added together in various ways to reproduce a broad array of colors. The name comes from the initials of the three colors Red, Green, and Blue. [2]

Figure 2: RGB

  2. YCbCr
     Y, which is luminance, means that light intensity and Cb and Cr are the blue-difference and red-difference chroma components.The transformation used to convert from RGB to YCbCr color space is shown in the equation:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & -74.203 & 112 \\ 112 & -93.786 & -18.214 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

  3. HSV
     H parameter describes the angle around the wheel, S represents saturation and V represents value of lightness.

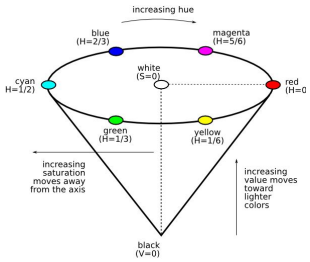

Figure 3: HSV

In my implementation, I transferred the RGB image into YCbCr and HSV to detach the human face from the background since RGB will be affected by the lighting and YCbCr and HSV focus more on colors.

- Procedure

  1. Apply the Gaussian filter to reduce the noise (Gaussian filter: In image processing, Gaussian filter is widely used by a 2-D Gaussian smoothing kernel).
  2. Transfer the RGB image to HSV space and YCbCr space separately.
  3. Binarize the YCbCr image by a mask:
     $80 \leq\ Y, 77 \leq\ Cb \leq 127, 133 \leq\ Cr \leq 174$
     $Let\ N\ = \frac{(x-ec_x)^2}{a^2} + \frac{(y-ec_y)^2}{b^2}$, where $\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} Cb - c_x \\ Cr - c_y \end{bmatrix}$
     and $c_x\ =\ 109.8, c_y\ =\ 152.02, \theta\ =\ 2.53\ ec_x\ =\ 1.60, ec_y\ =\ 2.41$
     $a = 25.39, b = 14.03$
  4. Binarize the YCbCr image by a HSV mask:
     $50 degrees \leq\ H, 0.23 \leq\ S \leq 0.68$
  5. Check labeled non-black connected components in 8 directions to form boxes.(These boxes are potential desired region.) (Hint: Matlab has built-in function bilabel to deal with it.)
  6. Check all potential regions with following constraints:

     (a) The area of the box can't be too small(i.e in my setting, the width and height can not be less than 20 pixels).
     (b) The ratio of width and height should be in a range.(I set $0.5 < height \div width < 2$).
     (c) The ratio of the squared circumference and area should be reasonable.
     (d) There should be at least black connected components inside the white region (2 eyes).
  7. Return all possible regions.
  8. Insert the location of regions to the original graph.
  9. Crop the desired region from that image.

2.3 Results

- Deep model

Figure 4: The region given by deep model

Figure 5: The cropped image given by deep model

- Skin color model

Figure 6: The binary image after the HSV and YCbCr mask

Figure 7: The region given by skin color model

Figure 8: The cropped image given by skin color model

2.4 Discussion

1. To have a better binary image, I combined the setting of 3 different papers[1][2][5] and adjusted some thresholds.
2. Pros of skin color model:
   (a) It doesn't rely on trained model, which is convenient.
   (b) It's fast and can detect multiple faces in one shot.(My implementation doesn't support this but the algorithm/model itself is capable for multiple faces detecting).
   (c) Lighting won't affect the result a lot.

3. Cons of skin color model:

   (a) It's hard to filter the neck since it's connecting to the head(they would be regarded as a whole part).

   (b) The algorithm is unable to detect small faces in the image.(It's part of my filter mechanism).

   (c) The color of background(or clothes) can not be too similar to the skin.

3 edge detection

There are many existing edge detection algorithm and Roberts, Sobel,Prewitt, LOG and Canny are commonly used and built-in in Matlab. However, to learn more about edge detection, I implemented my version of Roberts,Sobel,Prewitt and LOG.

3.1 Methods

- Roberts
  Roberts Cross Edge Detector will apply 2 2x2 convolution kernel Gx: $\begin{array}{|c|c|}\hline +1 & 0 \\ \hline 0 & -1 \\ \hline \end{array}$ and Gy: $\begin{array}{|c|c|}\hline 0 & +1 \\ \hline -1 & 0 \\ \hline \end{array}$. Combine them together to find the gradient. $|G| = \sqrt{Gx^2 + Gy^2}$ or $|G| = |Gx| + |Gy|$. Finally, we can set a threshold to binarize the image.
  example: $\begin{array}{|c|c|}\hline P_1 & P_2 \\ \hline P_3 & P_4 \\ \hline \end{array}$
  $|G| = |P_1 - P_4| + |P_2 - P_3|$

- Sobel
  Sobel Edge Detector will apply 2 3x3convolution kernel Gx: $\begin{array}{|c|c|c|}\hline -1 & 0 & +1 \\ \hline -2 & 0 & +2 \\ \hline -1 & 0 & +1 \\ \hline \end{array}$
  and Gy: $\begin{array}{|c|c|c|}\hline +1 & +2 & +1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$. Combine them together to find the gradient. $|G| = \sqrt{Gx^2 + Gy^2}$ or $|G| = |Gx| + |Gy|$. Finally, we can set a threshold to binarize the image.

- Prewitt
  Prewitt Edge Detector will apply 2 3x3convolution kernel Gx: $\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$
  and Gy: $\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$. Combine them together to find the gradient.

$|G| = \sqrt{Gx^2 + Gy^2}$ or $|G| = |Gx| + |Gy|$. Finally, we can set a threshold to binarize the image.

- LOG(Laplacian of Gaussian) and Zero-cross
  Laplacian is:$L(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$ However, since it's the 2nd order derivative, it's sensitive to the noise. As a result, to counter this, the image is often Gaussian smoothed before applying the Laplacian filter. i.e $LoG(x, y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$ where σ is the standard deviation we can define. Also, we can choose the size.
  Zero-Cross: After filtering the image by LOG function(which can be derived from fspecial("Log") in matlab). For each pixel, we can find the nearby 4 or 8 neighbors(I used 4-direction version). If there is at least one neighbor which is different sign from the center pixel, we set that pixel to white,otherwise,black will be set.(Also, we can set a threshold here. i.e: The absolute difference of the center pixel and the neighbor has to be larger than the threshold to be set as white, otherwise, it would be set as black).

3.2 Results

The Figure was generated by Matlab built-in approaches with default thresholds,sigma and kernel size. (If you want to have a look at the effect of thresholds, you can follow the github link in Abstract. There are results for different algorithms with different thresholds. Also, they were generated by my own implementation.).

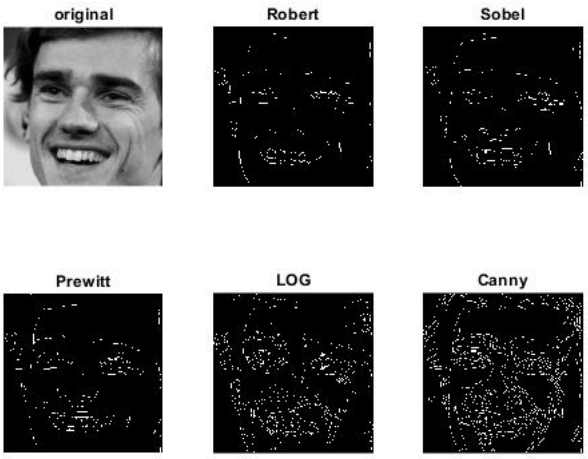

Figure 9: The edges of 5 different methods

3.3 Discussion

Algorithms considering the 2nd order derivative would probably return better results.(We can notice that the results of LOG and Canny contain more information). However, they are sensitive to noise. As a result, when we have a noisy image, we can try to filter the noise first.

4 Conclusion and future work

Both deep model and skin color model will give good results for face detection.However, compared to deep model, skin color model has more limitations. Canny and LOG edge detection can illustrate the edges on faces (For example, eyes, nose,lips,etc.) The information could be used for facial landmarks detection or tracking points initialization.
Future work:

1. Change my implementation to detect multiple faces(The algorithm supported multiple faces detection but my implementation didn't support that temporarily).

2. Add a new filter to separate the neck and face in skin color model.

3. Deep learning also does a good job on edge detection, I can do more research on that.

References

[1] J.Sun Q.Liao D.Zhang, B.Wu. A face detection method based on skin color model. https://download.atlantis-press.com/article/1725.pdf.

[2] G.Perez. J.Basilio, G.Torres. Explicit image detection using ycbcr space color model as skin detection. http://www.wseas.us/e-library/conferences/2011/Mexico/CEMATH/CEMATH-20.pdf.

[3] Spitsyn V.G. Kalinovskii I.A. Compact convolutional neural network cascade for face detection. https://arxiv.org/ftp/arxiv/papers/1508/1508.01292.pdf.

[4] Danny Pascale. A review of rgb color spaces...from xyy to r'g'b. http://www.babelcolor.com/index$_h tm_f iles/A$2003.

[5] P. Shimpi C. Bapat S. Kolkur, D. Kalbande and J. Jatakia. Human skin detection using rgb, hsv and ycbcr color models. https://arxiv.org/ftp/arxiv/papers/1708/1708.02694.pdf.