Image Processing Final Project

University of Burgundy

Master's in Computer Vision and Robotics

(VIBOT & MScV)

Pixel-based Skin Color Detection Techniques

Group members:
Hassa ZAAL
AbdelRahman ABUBAKR

Supervisor:
Prof. Desire Sedibe

Contents

1 Introduction

Human skin color has been used and proved to be an effective feature in many applications such as human face detection, hand tracking, image content filtering, content-aware video compression and many more. In this project we implemented the algorithms mentioned in the paper "A Survey on Pixel-Based Skin Color Detection Techniques" by Vladimir Vezhnevets, Vassili Sazonov, and Alla Andreeva.

1.1 UCI skin segmentation data set

The Authors of the paper mentioned using the Compaq skin database for comparing the results of all algorithms. However, we could not find any source for the Compaq database, therefore we decided to use the UCI skin segmentation dataset, provided by Center for Machine Learning and Intelligent Systems at University of California, Irvine. the dataset can be downloaded from here: https://archive.ics.uci.edu/ml/datasets/Skin+Segmentation

According to the dataset website, the skin dataset is collected by randomly sampling B,G,R values from face images of various age groups (young, middle, and old), race groups (white, black, and asian), and genders obtained from FERET database and PAL database. Total learning sample size is 245057; out of which 50859 is the skin samples and 194198 is non-skin samples.

This dataset is of the dimension 245057 * 4 where first three columns are B,G,R values and fourth column is of the class labels (decision variable y), in the dataset file Skin colors are labeled as number 1 and Nonskin elements as 2.

1.2 Dataset visualization

First step to start any project with big amount of data, is to visualize this data to see its distribution and roughly expect the behavior of your algorithms. for our data set, we wrote a simple code Plot-Dataset.py that plots skin data in blue, and nonskin in red, the 2 axes are the Cb and Cr components of the colors. Figure shows the visualization of the dataset.

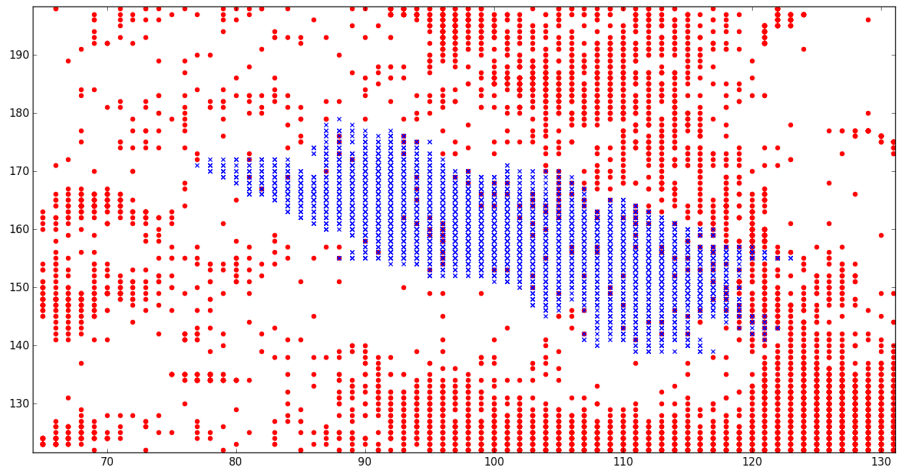

Figure 1: UCI Data set visualization

1.3 How to run ?

For the code implementation we used Python 3.5, with some help from Scikit-Learn 0.18.1 library when needed. We put all the algorithms in different files, and different directories, so for running any code, you can use any python IDE, or by Linux terminal change the directory for the code directory, then write:

Python FileName.py

All source code is available on Github Repository: https://github.com/abdelrahman-gaber/Pixel-based-Skin-Color-Detection

2 Explicitly defined skin region

One method to build a skin classifier is to define explicitly (through a number of rules) the boundaries skin cluster in some color space. Any (R, G, B) pixel values given to the algorithm is classified as skin if:

$$R > 95 \; and \; G > 40 \; and \; B > 20 \; and$$

$$max(R, G, B) - min(R, G, B) > 15 \; and$$

$$|RG| > 15 \; and \; R > G \; and \; R > B$$

This algorithm is implemented in the file defined-skin-region.py, and the results was as follows:

$$\text{True positive rate} = 100\%$$
$$\text{False Positive rate} = 3.7\%$$

We notice here the big value of True positive rate, but this is expected as the data set used has an obvious boundary for the skin region.

3 Non-parametric skin distribution modeling

The final goal of skin color detection is to build a decision rule, that will discriminate between skin and non-skin pixels. This is usually accomplished by introducing a metric, which measures distance (in general sense) of the pixel color to skin tone. The type of this metric is defined by the skin color modeling method. the paper differentiated between many types of modeling, the most important two of them are Non-parametric and parametric skin modeling techniques.

The key idea of the non-parametric skin modelling methods is to es- timate skin color distribution from the training data without deriving an explicit model of the skin color. The result of these methods sometimes is referred to as construction of Skin Probability Map (SPM).

3.1 Bayes Classifier

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes theorem with the naive assumption of independence between every pair of features. Given the vectors of the RGB colors with the class labels of them, we used the Scikit-Learn Naive Bayes classifier to train and test our data.

We used half of the elements of both skin and nonskin data for the training, and the other half for the test.The Naive Bayes classifier has many different types, the different classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i|y)$. For our case, as we know that the data is mainly a gaussian type, so we used the GaussianNB() function, which implements the Gaussian Naive Bayes algorithm for classification. In this case, the likelihood of the features is assumed to be Gaussian.

The result of Naive bayes classifier for our labeled data is as follows:

True positive rate = 99.032%
False Positive rate = 0.967%

4 Parametric skin distribution modeling

The need for more compact skin model representation for certain applications along with ability to generalize and interpolate the training data stimulates the development of parametric skin distribution models.

All implemented parametric methods operate in colorspace chrominance plane, ignoring the luminance information. For our code we convert the RGB color data to YCbCr first, then work on Cb and Cr values.

4.1 Single Gaussian model

Skin color distribution can be modeled by an elliptical Gaussian joint probability density function (pdf), for this case we implemented the joint pdf between Cb and Cr components, and used half of the skin elements to train the model and find the parameters. Figure 2 shows a visualization of our Gaussian model, where the X axis is the Cb component, and Y axis is the Cr component.

The mean values and the covariance matrix of our model are as follows:

$$Mean = \begin{bmatrix} 100.48 & 161.55 \end{bmatrix}$$

$$Covariance = \begin{bmatrix} 73.87 & -43.02 \\ -43.02 & 39.52 \end{bmatrix}$$

After training the model and getting the parameters, we set a threshold to the probability of any new color input. for each input from the test set, if the probability is bigger than the threshold, the color input is considered as skin, otherwise, it is non-skin pixel. for our case we chosed a probability threshold = 0.0001, and the results of the testing set are as follows:

True positive rate = 98.50%
False Positive rate = 0.16%

We notice the high value of True poisitve, and low value of False Positive rates. This can be explained from the visualization of the original data set, from which it is obvious that the skin and non-skin data are not overlapped, and there is an obvious boundry between them.

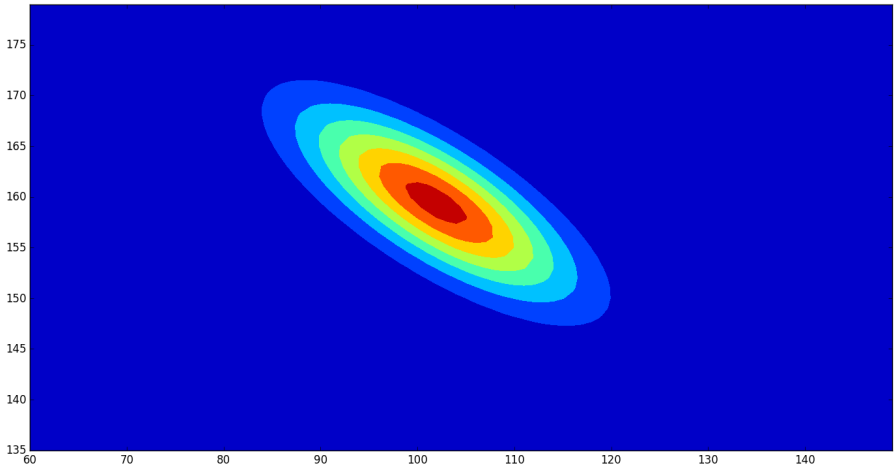

Figure 2: The Gaussian Model of the skin data

4.2 Gaussian Mixture Model (GMM)

A more sophisticated model, capable of describing complex-shaped distributions is the Gaussian mixture model. It is the generalization of the single Gaussian, the pdf in this case is just a sum of many weighted Gaussian each with its own mean and covariance. The difficulty of this model arises when trying to find the parameters of each Gaussian, and the weight of each of them. In our case we used the Expectation Maximization algorithm to find all these parameters.

To implement this algorithm and find the parameters from the training data, we used the Scikit-Learn mixture package, which implements Gaussian Mixture Model, and finds the parameters using EM algorithm. For our training data, we tried various number of Gaussians and discovered that the best number of them is 4, which increases the TP vs. FP values, and balance the weights of the Gaussians. Figure 3 shows the visualization of the GMM model with 4 components, we can observe that some Gaussians have more probability weight than others [Red color], these weights are also evaluated by the EM algorithm.

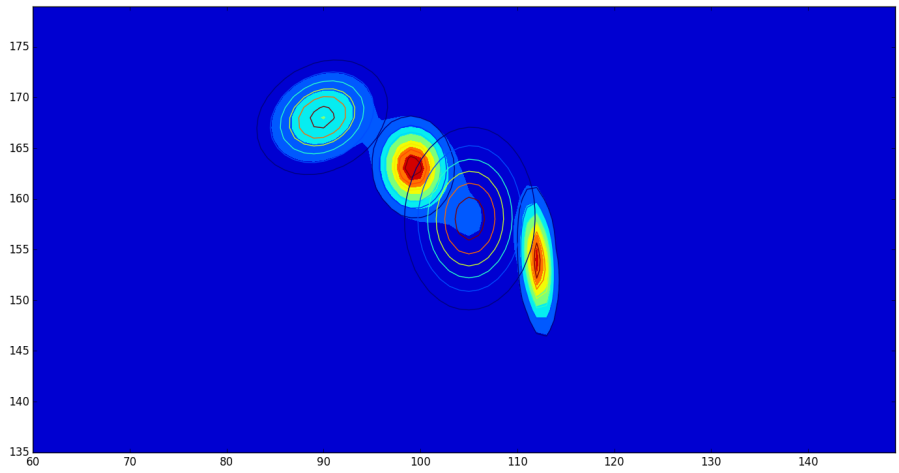

Figure 3: The Gaussian Mixture Model obtained by EM algorithm

Using the EM algorithm, we found the parameters of the Gaussians as follows: First component with weight = 0.230 and

$$Mean_1 = \begin{bmatrix} 112.10 & 153.82 \end{bmatrix}$$

$$Covariance_1 = \begin{bmatrix} 1.13 & -1.05 \\ -1.05 & 14.45 \end{bmatrix}$$

Second component with weight = 0.328 and

$$Mean_2 = \begin{bmatrix} 99.21 & 163.13 \end{bmatrix}$$

$$Covariance_2 = \begin{bmatrix} 4.32 & -0.27 \\ -0.27 & 6.10 \end{bmatrix}$$

Third component with weight = 0.159 and

$$Mean_2 = \begin{bmatrix} 105.09970179 & 158.0579959 \end{bmatrix}$$

$$Covariance_2 = \begin{bmatrix} 11.84 & 0.38 \\ 0.38 & 21.36 \end{bmatrix}$$

Fourth component with weight = 0.281 and

$$Mean_2 = \begin{bmatrix} 89.84 & 168.04 \end{bmatrix}$$

$$Covariance_2 = \begin{bmatrix} 12.11 & 2.18 \\ 2.18 & 8.42 \end{bmatrix}$$

An important thing to notice is that the sum of all weights must be = 1, as the model is a new pdf. For any new color vector, we can now compare its probability with a threshold, and it is considered as color if the probability is bigger than this threshold. For our code, we set the threshold to value = 0.000001. For this case the results of the testing set are as follows:

True positive rate = 92.48%
False Positive rate = 0.169%

4.3 Elliptic boundary model

By examining skin and non-skin distributions in several colorspaces, it has been concluded that skin color cluster, being approximately elliptic in shape is not well enough approximated by the single Gaussian model. Due to asymmetry of the skin cluster with respect to its density peak, usage of the symmetric Gaussian model leads to high false positives rate. An alternative called an elliptical boundary model is equally fast and simple in training and evaluation as the single Gaussian model, and avoid the complexity and time consumption taken by Gaussian Mixture Model.

The algorithm is implemented by the equations in section 3.34 in the paper, and all the parameters of the model are evaluated. Given threshold θ and input chrominance of a pixel X, the pixel is classified as skin if $\Phi(\mathrm{X}) < \theta$ and as non-skin otherwise. For our code, we found a threshold of value = 1.0 is good choice. the results of the testing set are as follows:

True positive rate = 98.682%
False Positive rate = 0.205%