EECS 490 DIGITAL IMAGE PROCESSING

December 2004

SESSION 3 Face Detection

Mike Adams	Face Recognition in Digital Images using Kmeans Clustering
Svend Johannsen	Face Recognition
Michael K. Lee	Face Recognition Using MATLAB
Deng-Hung Liu	Face Recognition by Color Segmentation and Morphological Image Processing
Iouri Petriaev	Face Recognition: Color Segmentation and Principal Component Analysis
Chris Roberts	Face Detection in Complex Scene Images using Color Segmentation and Morphological Techniques
Ira Ross	Face Detection and Localization in Images using Color Segmentation and Template Matching
Yu-Hong Yen	Face Detection

Face Recognition in Digital Images using Kmeans Clustering

Mike Adams

Department of Electrical Engineering and Computer Science,
Case Western Reserve University, Cleveland, OH, Email: mda10@case.edu

ABSTRACT

This paper presents a method of recognizing facial regions in digital images through the use of the Kmeans Clustering algorithm [1]. Training data used in clustering is extracted from color-segmented [2] images in which there are an arbitrary number of facial regions with arbitrary size, facial-pose, resolution and lighting conditions. Kmeans is one of many methods of classifying data for automated decision-making, but is shown here as one that is viable for use in face recognition.

KEYWORDS

Kmeans, face-recognition, digital image processing

INTRODUCTION

Face recognition is very important in many research areas from machine vision to complex security systems. One major difficulty in face recognition is the complexities inherent in characterizing a typical face. In an image there are countless ways that faces can be posed (looking up,down, straight ahead, etc.), rotated, shaded and lighted. These complexities are only compounded by the additionnal condisderation specific facial features that can be distorted depending on facial expression. The use of the Kmeans clustering algorithm is therefore motivated by its ability to classify or 'cluster' data based on any indicators that might be buried within a set of data, such as the features that describe facial regions in an image.

KMEANS CLUSTERING

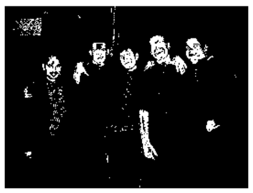
The goal of Kmeans is to take a set of training data points in N-dimensional space and group them into K separate clusters for the purpose of then deciding in which clusters a set of novel data points belong. The data points in N-dimensional space can be thought of as vectors, each with N components. The assignment of each of these vectors to a cluster is then decided based on the vectors' Euclidean distance to the center of a cluster. A vector gets assigned to the cluster closest in N-dimensional space to itself. Each vector has an attribute, thus each cluster has an average attribute. These attributes are the defining characteristics of a vector and a cluster (i.e. Cluster 1 has an average attribute value of .7, so its vector members each correspond to a facial region and any new vector member assigned to Cluster 1 must therefore correspond to a facial region) [1].

ALGORITHM [1]

1) Scale training data to be between 0 and 1.
2) Create the clusters by randomly choosing input patterns and assigning each by itself to a cluster for the desired number of clusters. Each cluster then has only one member vector (pattern) and that pattern is the cluster centroid.
3) Process every remaining input pattern by assigning each to the closest cluster (Euclidean Distance in N-space).
4) Update the clusters by re-computing the cluster centroids and average cluster attributes (from the addition of new patterns to each cluster, the centroid of each cluster will change, as will the average attribute of each cluster).
5) Update every single patter vector in N-space. Because at this point, all cluster centroids have changed, a pattern's distance to its own cluster may be greater than its distance to another cluster. Therefore, reassign that pattern to the closest cluster.
6) Continue for a desired number of passes through the input pattern population or until the system settles down to the point that on a particular pass there are no pattern reassignments.

TRAINING SET COMPILATION

The question then is what will the training set of input vectors consist of? Later questions will follow in the implementation of the algorithm, such as the number of clusters to be used and the size of the training set. As a first step, digital images containing faces may be color segmented by choosing one face (in a fully automated version, the color segmentation would be based on an average facial color vector, taken over many faces, not a face chosen by the user). The result is a black and white image where many non-face regions have been eliminated, and facial regions are left in predominance. See the example here.



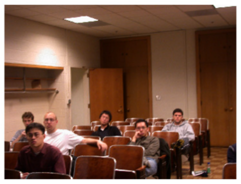
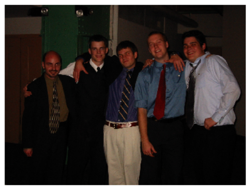
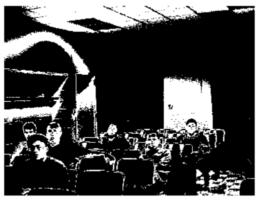
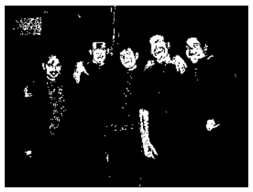
Nine such images were processed in this way, providing a good variety of segmentation results, facial poses and different white-to-black region ratios. Pixel regions (100x100) were extracted from these segmented images resulting in over 7600 input patterns, 71 of which were designated as 'face' regions.

Some different features were then extracted from the 100x100 pixel regions to compile the set of training input pattern vectors. In the first case the mean and variance of each column in a region were extracted, resulting in a set of pattern vector each with 200 features (200-D data points in 200-D space) and one attribute (1 corresponding to a face region, 0 corresponding to non-face). In the second case, the eigenvalues of each 100x100 pixel region were extracted, resulting in 100-D pattern vectors. In either case the final result is a training set consisting of 7644 possible 100- or 200-D pattern vectors, 71 of which correspond to facial regions.

Preprocessing of the image to be evaluated follows in the same manner, the instructions for which are given in the Proj1.m Matlab script file printed as the Appendix. The final step then is to run Proj.1m and follow the instructions given. Kmeans will output a set of indices, each corresponding to the top left corner of a 100x100 pixel region that should contain a face or part of a face in the evaluation image. The user is then instructed on displaying the final result which is the original image containing 100x100 white squares marking the Kmeans facial locations. Note that a single white square does not necessarily indicate the location of an entire single face, but rather the location of a region where face-designated pattern features are found.

RESULTS AND DISCUSSION

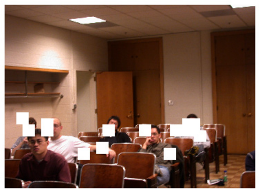
The two images shown below were used to evaluate the effectiveness of this method and with each image, experimentation with different numbers of clusters and different size training sets was done. In changing the training sets, only the number of non-face designated patterns was decreased while all available face patterns were kept in the training sets.

It was immediately clear that the set of pattern features corresponding to variances and means was the wrong sort of data to use in describing regions containing faces. Kmeans was returning too many indices corresponding to faces or none at all. The rest of the experimentation was therefore conducted using the training patterns corresponding to the eigenvalues of each 100x100 pixel region. This information seemed to better describe the 100x100 regions.

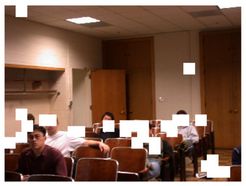
CLASSROOM PHOTO

Trained w/3500 Patterns



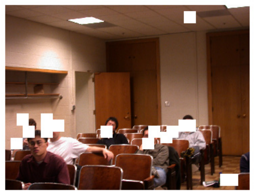
1000 Clusters

Trained w/ 2500 Patterns

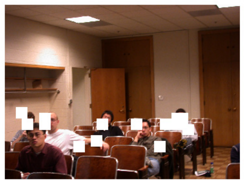


1000 Clusters

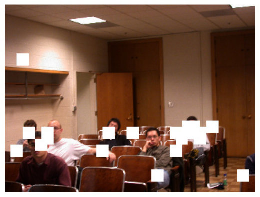
Trained w/3500 Patterns



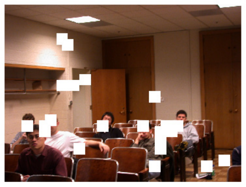
500 Clusters



1000 Clusters



1500 Clusters



2000 Clusters

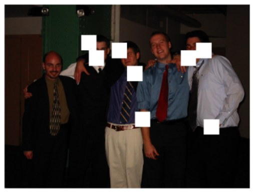
The first two images above illustrate the different results obtained in with different size training sets. Training with 3500 patterns yields the best results in that the same number of faces are found with fewer false positive identifications.

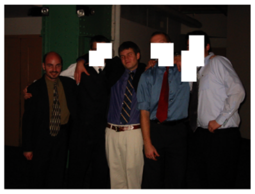
The next four images are all trained with 3500 input patterns but differing numbers of clusters. Clearly, the are fewer false positive region identifications in the 500 and 1000 clusters cases, but the 1000-cluster case seems to have the fewest false positives while correctly identifying 5 out of 6 face regions. Using more than 1000 clusters seems to yield more false positive identifications as the number of clusters rises.

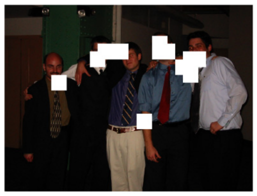
SHIRT 'N TIE PHOTO

This is a smaller photo than the previous one so it makes sense to train with a much smaller set and thus start with much fewer clusters.

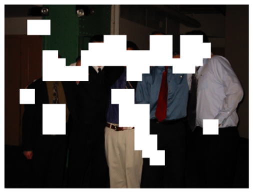
Trained w/ 1000 Patterns


200 Clusters

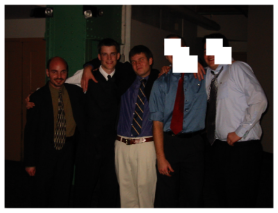

500 Clusters


800 Clusters

Trained w/ 200 Patterns


100 Clusters

Trained w/ 2500 Patterns


1200 Clusters

The best result above is for the case of training with 1000 patterns and using 800 clusters. 5 faces are correctly identified with a 6th that could go either way on the far left. Also the false positive identifications are no more than for the cases with lesser clusters. To motivate careful decision of training set size, the last two results are shown. Notice that for the case of training with only 200 patterns and using 100 clusters, all skin regions are identified along with a few other false positives. This result is an example of 'undertraining' , because in using only 200 training patterns, there are not enough examples of non-face regions or examples of differences between face pattern vectors and non-face pattern vectors. The last image is an example of 'overtraining' in that there are two many examples of non-face regions, therefore the number of face regions returned by Kmeans is quite low, even yielding too many false negative identifications.

In general, this second evaluation image gets much fewer false positive identifications than the Class Photo. The reason may be that this second image lends itself much better to color segmentation in terms of the singled out face regions and hardly any other white regions showing up in the segmented result. This proves to be a much better head start for the Kmeans algorithm to do its work.

SUMMARY

Kmeans is a viable method or at least a good start in face recognition. The method presented here though yields too many false positives to be useful in any meaningful application. Further work can be done in choosing pattern features that more aptly capture the differences between face regions and non-face regions in an image. If such pattern features are found, Kmeans will produce much better results.

ACKNOWLEDGMENTS

Prof. Newman wrote most all the C++ code for the Kmeans algorithm for the purpose of an EECS 484 assignment. The training set of images was obtained from Prof. Merat's EECS 490 website.

REFERENCES

[1] Yoh-Han Pao, Wyatt S. Newman, "A Primer for the Practice of Computational Intelligence".

[2] R. C. Gonzalez, Richard E. Woods, "Digital Image Processing, "2nd Edition, Prentice Hall, Upper Saddle River, NJ, 2002.

Appendix

```
clc;clear;close all;

colhisteq; %histogram equalization using an average color histogram
colseg; %segmentation performed based on Facial ROI
bw(1:size(I,1),1:size(I,2)) = I(1:size(I,1),1:size(I,2),1);
P = getMasks(bw);
F = getPatFeats(P);
```

```
csvwrite('Evaluation Data.csv',F);

%in Evaluation Data.csv insert as the first column text cells with
%at least 4 chars in them and insert in the last column just a column off
%zeros (for purposes of the C++ code, ease of programming for me)
%save as tab delimited .txt, run Kmeans.exe,train with
%Eigen Faces Train2_3500.txt or other .txt training file options that
%differ only in number of training patterns but make sure that training
%file contains all available face-designated patterns,
%copy the face indices from the program window, read in
%original image,convert image to type double, run translateIndices, run showFaces

%i = indices copied from program window
%image = imread('bros.jpg');
%image = im2double(image);
%[m,n] = translateIndices(indices,bw);
%showFaces(indices,m,n,image)
```

Face Recognition

Svend Johannsen

Department of Electrical Engineering and Computer Science,

Case Western Reserve University, Cleveland, OH, Email: skj7@cwru.edu

Abstract

This paper presents a design to perform face recognition in an image. The design is focused on isolating the regions of an image corresponding to peoples faces; it does not recognize specific faces. Image processing techniques are used to perform the said operation.

The RGB color of an average is face is acquired and this color is used to perform color segmentation [1]. The result of the color segmentation is a binary image where all regions that potentially could be a face are colored white. The smaller regions in this image represent things we are not interested in such as people hands and objects around the people that happened to have roughly the same color as skin. These things will be referred to, throughout the paper, as noise. A combination of dilation [2] and erosion [3] is used to keep only the largest of these regions, thereby getting rid of a lot of the noise. To improve the result, an opening filter [4] is applied; an opening filter is basically erosion followed by dilation using the same structuring element for both operations.

The final result is a binary mask which when applied to the original image, isolates the faces. Depending which faces you choose to represent an average face, the design will isolate either all or all but one face. The number of objects that are incorrectly classified as being faces is usually between one and three, again depending on which faces are chosen, to represent an average face.

KEYWORDS

Face recognition, image processing, color segmentation, dilation filter, erosion filter, opening.

INTRODUCTION

When using color segmentation to isolate the faces, the main problem is that faces are not uniformly colored. The region around the eyes, for instance, is significantly darker. To combat this problem a dilation filter is applied to the color segmented image. Hopefully the gaps in the color segmented image representing eyes, nose and mouth are so small that a relatively weak dilation filter can cover them. We are not interested in using a strong dilation filter because it will enhance noise as well.

An erosion filter is used to clean up the noise, such as hands. The erosion should be as strong as it possible; i.e. it should remove as much noise as it can without removing the faces completely. After applying the erosion filter the regions are sort of rugged and there is still some noise left, so an opening filter is used to smooth out the face regions and remove more of the noise.

FACE RECOGNITION USING IMAGE PROCESSING

The image `Class Photo1.jpg` is selected as the image we wish to isolate the faces in. it is displayed in Figure 1.

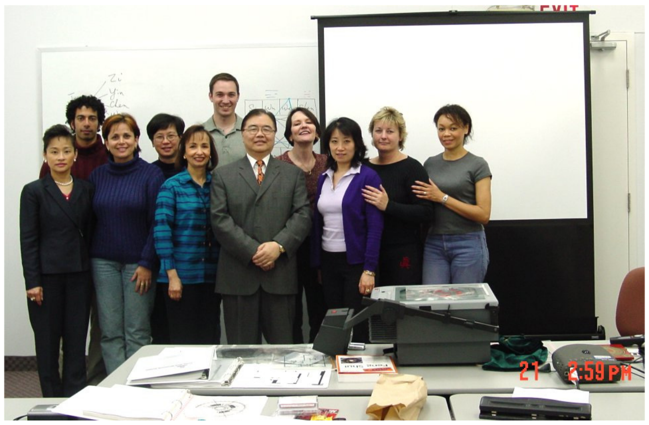


Figure 1: The image used for face recognition.

The MatLab function `roipoly` is used to select a region corresponding to a face in the image. All RGB color values in this region are then extracted and their mean and standard deviation is calculated using the MatLab functions `mean` and `std`. This procedure is similar to the one applied to the cookie image when extracting the average color value of a chocolate chip (midterm project).

The procedure is repeated four times, giving us an average color value and standard deviation of 4 different faces. Next the average of these 4 color values is computed along with an average of the 4 standard deviations. This gives us **the color of an average face** as well as the standard deviation.

The average face color is used in the color segmentation [1]. The result of the color segmentation is a binary image where all pixels, in the original image, with a color reasonably close to the average face color is given the value of 1 and all other pixels are 0. This is given by the following:

$$S_{ij} = \begin{cases} 1 & \text{if } R_{ij} \in \mu_R \pm 1.25 \cdot \sigma_R \wedge G_{ij} \in \mu_G \pm 1.25 \cdot \sigma_G \wedge B_{ij} \in \mu_B \pm 1.25 \cdot \sigma_B \\ 0 & \text{otherwise} \end{cases}$$

The equation defines a cube in color space. All pixels in the original image with a color value within this cube are given a 1 in the binary image; pixels with a color value outside the cube are given a 0. The result is displayed in Figure 2.

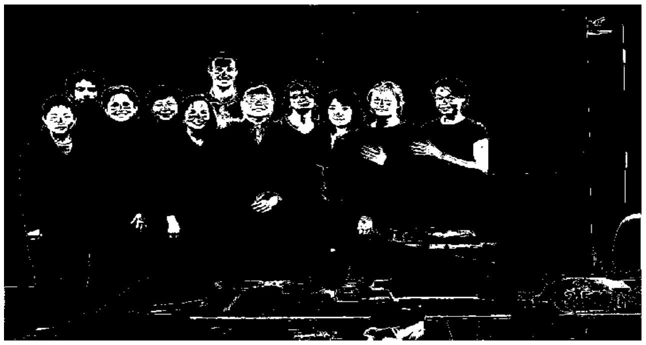


Figure 2: Result of a color segmentation using the average color of a face.

The first challenge is to create uniform regions from the sparsely outlined faces. A dilation filter [2] is chosen for this purpose. A dilation filter is given as:

$$A \oplus B = \left\{ z \mid \left[(\hat{B})_z \cap A \right] \subseteq A \right\}$$

The dilation is applied to the image A using the structuring element B. In this case B is a disc with a small radius. A small radius is used to avoid enhancing the noise to the left and towards the bottom of Figure 2 too much.

The dilation filter creates an edge with a width equal to the radius of the structuring element B around all white pixels in Figure 2. The result of the slight dilation is shown in Figure 3.

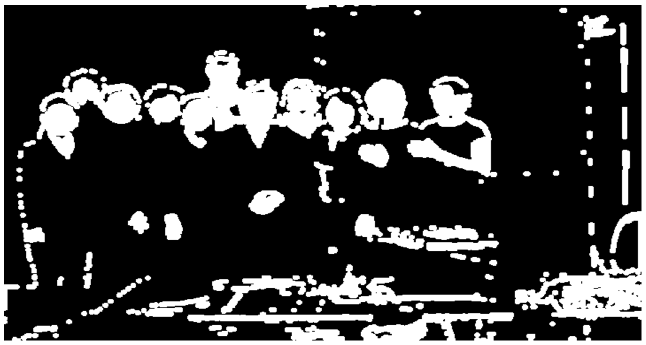


Figure 3: Slight dilation is used to create solid regions in the face areas.

The faces are now for the most part uniform regions. In order to get rid of all the noise as well as the smaller skin colored regions, such as hands, an erosion filter [3] is applied. Since the faces are the largest white regions in Figure 3 we can apply quite heavy erosion without loosing our primary information. An erosion filter can be thought of as an inverse dilation filter and is given by:

$$A \Theta B = \left\{ z \mid (B)_z \subseteq A \right\}$$

The structuring element B is now a disc with a larger radius than what was used for the dilation filter. The erosion filter removes the edge around all white objects in the image, this means small objects, such as the noise, will completely disappear. The result of the erosion is displayed in Figure 4.

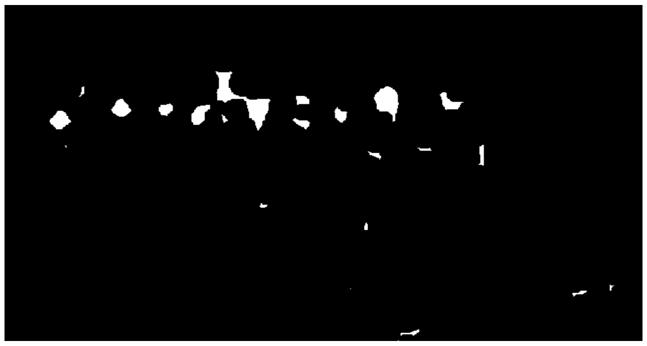


Figure 4: Heavy erosion is used to clean up the noise.

The noise at the bottom of Figure 4 is not completely gone, and some of the hands still show as well. Unfortunately one cannot apply heavier erosion without starting to loose faces.

In order to improve the result in Figure 4 an opening filter [4] is applied. An opening filter is just an erosion of the image A by B followed by a dilation of the intermediate result by B. Again a disc is used as the structuring element; notice that the radius of this disc is the same during both the erosion as well as the dilation process. The result after applying the opening filter is displayed in Figure 5.

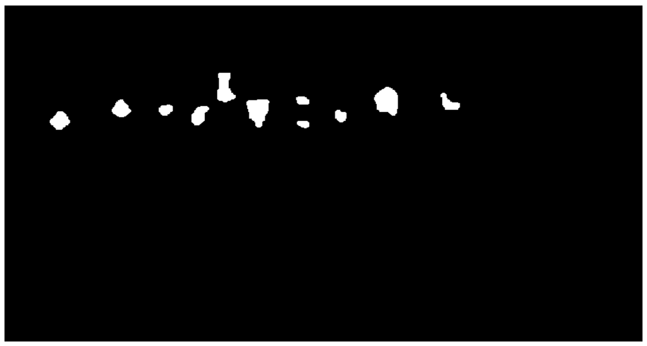


Figure 5: An opening filter is used to smooth the shape of the regions.

The image displayed in Figure 5 can be used as a mask to isolate the faces in the original image.

RESULTS AND DISCUSSION

The dilation filter uses a structuring element with a radius of 5. This is a large enough radius to even out the facial regions without enhancing the noise to the point where it also becomes uniform regions.

When applying heavy erosion, a filter with radius 13 is used, this is a rather high value for an erosion filter on an image of this size. One should notice that the face of the second person from the left is almost gone in Figure 4. At this point one cannot clean up the image further without removing the face of the second person to the left.

Applying the opening filter will sometimes result in the loss of one face, as the example in Figure 5 shows; it depends on which 4 faces are used to represent an average face. It will however always make the remaining regions more smooth, and clear up additional noise. The opening is performed using a radius of 5; that is an erosion filter is applied with a disc shaped structuring element of radius 5, followed by a dilation filter with the exact same structuring element.

Using the filter displayed in Figure 5 as a mask on the original image the faces can be extracted.

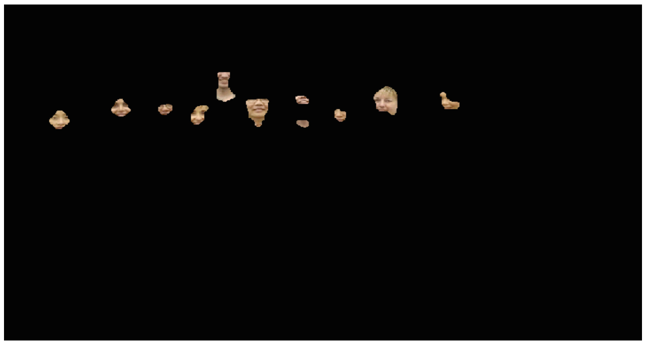


Figure 6: The obtained mask is used to isolate the faces in the original image.

The result is that 10 out of 11 faces were extracted and apart from person 8's neck, no noise is present in the image.

Faces 1, 5, 6 and 7, from the left, were chosen to represent an average face, had we instead choosen face number 1, 2, 5 and 6 to represent an average face the result of the face recognition would be as shown in Figure 7.

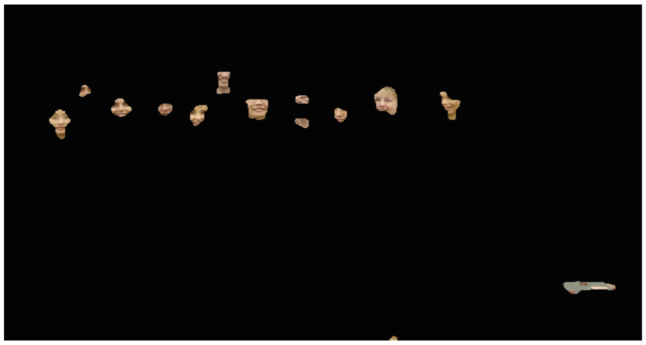


Figure 7: Using 4 different faces to represent an average face.

One should notice that all 11 faces are present in Figure 7 but two objects have incorrectly been classified as being faces. The reason for this result is that face number 2 is darker than the other faces, and therefore increase the standard deviation of the average face. A higher standard deviation will increase the chance of successfully isolating all faces; on the other hand a higher standard deviation also means more noise will be included in the final result.

SUMMARY

Image processing techniques have been applied to the selected image in order to isolate the faces in the image. The technique used involves finding the average color of 4 different faces and using these 4 colors to obtain the color of an average face. The color of an average face is then used to color segment the image.

The result from the color segmentation is a binary image showing all the faces as well as hands and items in the bottom and towards the left of image that happened to have the same color as the average face. A dilation filter is applied to make the regions representing the faces more uniform. Once this is done the face regions are the largest white regions in the image and an erosion filter can therefore be used to get rid of most of the noise.

To clean up the rest of the noise and smooth out the face regions an opening filter is applied, the resulting mask is applied to the original image to isolate the faces. 10 or 11 out of a total of 11 faces are isolated, depending on the choice of average face. This is a reasonably good result.

REFERENCES

[1] Raphael C. Gonzalez, Richard E. Woods, "Digital Image Processing" 2nd edition, pp. 331-339, 2002.
[2] Raphael C. Gonzalez, Richard E. Woods, "Digital Image Processing" 2nd edition, pp. 519-525, 2002.
[3] Raphael C. Gonzalez, Richard E. Woods, "Digital Image Processing" 2nd edition, pp. 525-528, 2002.
[4] Raphael C. Gonzalez, Richard E. Woods, "Digital Image Processing" 2nd edition, pp. 528-532, 2002.

Face Recognition Using Matlab

Michael K. Lee

Department of Electrical Engineering and Computer Science,

Case Western Reserve University, Cleveland, OH, Email: mkl7@cwru.edu

ABSTRACT
This paper presents an algorithm that detects a person's face on a static image using color segmentation and statistical data.

KEYWORDS
Face Recognition, Color Segmentation

INTRODUCTION
From an image, we can extract a lot of morphological information, such as finding which object is in the front most, or identifying what the object is next to the left wall. Another information we can look is to count how many people are in the image and locate their position. If we assume that everybody was looking at the camera when they took the picture, we could count the faces to find out how many people are there.

THE SKIN-TONE APPROACH
First I manually loaded the sample face images, and segmented the skin area. Then I used Matlab to calculate the average skin tone color, which were RGB values (152, 86, 74). Using this statistical information, I tried to segment the image with RGB values (152±d, 86±d, 74±d) where d is the offset. If I chose a small d, the algorithm will segment out only a portion of the faces; if I chose a large d, the algorithm may include the wooden chairs and doors as human skin. I tried to adjust the value of d to get the most optimized result, but it seemed to have more problems than I thought. If the environment such as the lights were changed, the skin tones changed drastically, and the average skin tone would have to be evaluated all the way from the beginning. Also, even in the same image, the lighting was illuminating the object in a different manner.

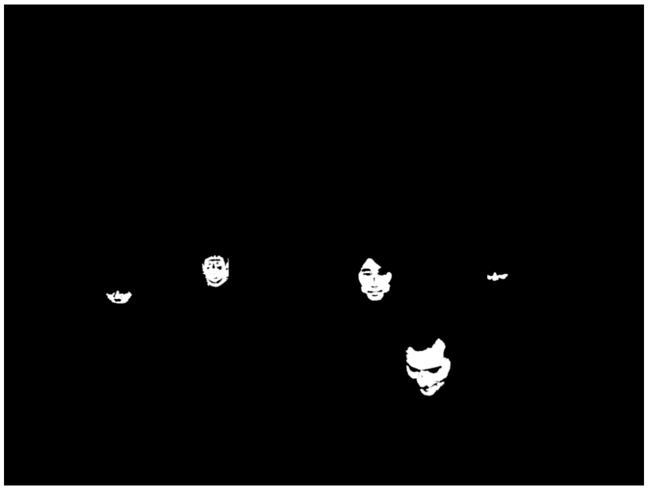


Figure 1. Segmenting Skin-toned Areas with Offset d = 15

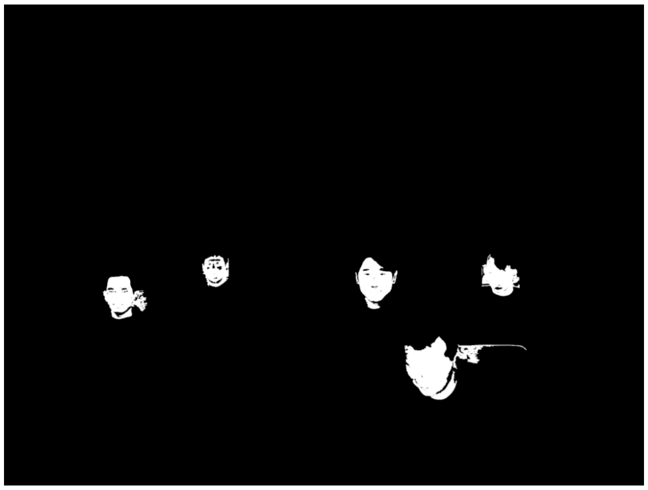


Figure 2. Segmenting Skin-toned Areas with Offset d = 23

Once I segmented the image, I had to cluster the pixels into groups so I could "count" the number of faces – if I just count the number of "regions" of skin-toned areas, there would be a discrepancy regarding when a person is right behind someone else, so two faces which are right next to each other can be counted as one. Since a face is roughly round or oval-shaped I made a function that groups a cluster of pixels into a "face". If a skin-toned area looks like a rotated 8 shape (∞, or the infinity symbol), or too long along the horizontal axis, I divided the region and count as two faces. After a few experiments, I found the ratio of the length along the vertical and horizontal axis would be optimal around 0.65 – if a skin toned region has height of 183 pixels and width of 248 pixels, I could say that the region has two faces located right next each other. The ratio is somewhat different from the actual face (0.65 means the height is almost twice the width) because the skin tone area also includes the neck, so the area becomes longer in the vertical direction. Knowing the ratio of a typical face oval, the same approach was used for identifying faces that are connected in the vertical/diagonal directions.

However, there were so many cases that the algorithm won't work because the effect of the hair. Although most people have similar skin tones, they have almost distinct hair colors. People also and wear a hat, dye in some other color and always can change the hairstyle that conceal the face, which became a variable that is too big to handle.

One another case was with bright light. Not only the skin tone was different from the average color, faces in a bright light had lot of contrasts – the shadows of the eyes, nose and mouth were distinct enough that the segmentation won't include them as a face, and the algorithm couldn't recognize the shape as a whole face.

THE TWINKLE APPROACH[1]

The human eye is very spherical and reflective. Locating the twinkle, a bright specular reflecton from the cornea due to the lighting, would be another approach to recognize faces. One advantage of this approach is that, most people have dark-colored pupil, and the twinkle is very bright. I could apply any edge detection algorithm, or a differential filter to find which point has large change in graylevel intensity. Not surprisingly, any small shiny objects such as buttons, glasses and even teeth were identified as twinkles. So, for the next algorithm, I added another condition for finding the twinkle – for any high contrast point in the image, try to look for another twinkle which is around 40~100 pixels apart in the horizontal axis. Again, the number 40~100 pixels comes from examining a set of pictures and averaging the distance between a person's eyes. Also, it has to be surrounded by some skin-toned area.

The algorithm seemed to work well with the images, especially after adding another condition. However, this approach was insufficient to recognize faces in low-light conditions where the twinkles of the eyes weren't bright enough. Also, if the picture was blurred, or the either the object or the camera moved while taking the picture, the twinkle approach won't work at all.

COMBINING THE TWO

After experimenting with both algorithms, I new thought came up – what happens if I try both algorithms at the same image? The skin tone approach was fairly good, but it needed a offset d small enough to segment the image containing only the faces. The twinkle approach would only work in bright conditions. So what would happen if I set the skin tone for a darker region and a small d, and apply both methods?

The result was promising than I expected. The skin tone approach handled the case where there were low light – I set the RGB values as (102, 53, 49), which was the average of skin tones in low light conditions. I also tried to make d as small as possible so it won't recognize any background as a person's skin. The only thing is that this won't include the faces that are in the front, that are usually brighter. However, it seemed that the one in the front would be recognized by the twinkle approach – people that are in the near to the camera or a light source would have two twinkles. Of course, I set the "surrounding skin tone" higher than before, where faces in the low light conditions would be handled with the skin tone approach.

Another addition to the algorithm was, I could break down the levels of skin tones with smaller offset d's. I applied the skin tone approach with a near-dark skin tone, repeating the algorithm with a little brighter skin tone, until I could find all of the faces. After a number of experiments, I realized repeatedly applying 4 skin-tone algorithm would find most of the faces.

Also I had to figure out how to identify which face in one algorithm is the same one on the other algorithm. A simple but effective solution was to find the size and location of the face, compare if the characteristics are similar; if two are far away I can say that they are faced that were not recognized on the other algorithm; if they are placed in a similar location and resemble in size, those two are the same face.

RESULTS

I tried a number of pictures and ran the matlab functions to recognize faces. Although I was to locate the faces, but the number of faces would be a rules-of-thumbs of evaluating the efficacy of an algorithm.

Number of Faces	Skin Tone Approach	Twinkle Approach	Combined Algorithm
6	4	2	6
6	5	3	5
5	5	1	5
4	3	2	4
4	4	3	4
3	2	3	3
0	1	1	0

SUMMARY

The face recognition was done in two different approaches which complement each other. The skin tone approach could recognize faces in low-resolution or blurry images but only could identify faces with skin tones in a narrow range of RGB color. The twinkle approach could accurately locate the faces with ease, but it would only work in high-resolution images taken in a bright environment. Combining the two would give a nice result. However, the algorithm here won't recognize the ones that are not facing the camera, or people with long hair screening the face.

REFERENCES

[1] Computer operation via face orientation by P. Ballard and G.C. Stockman; Pattern Recognition, 1992 . Vol.1. Proceedings 11th IAPR International Conference on Computer Vision and Applications, 30 Aug.-3 Sept. 1992 Pages:407 - 410

[2] Detecting faces in images: a survey by Ming-Hsuan Yang; D.J. Kriegman, and N. Ahuja; IEEE Transactions on Pattern Analysis and Machine Intelligence,Volume: 24 , Issue: 1 , Jan. 2002. Pages:34 - 58

[3] Low-dimensional procedure for the characterization of human faces by L.Sirovich and M. Kirby, in J. Optical Society of America, Vol. 4, No. 3, Page 519-524, March 1987.

[4] Rafael C. Gonzalez, Richard E. Woods, "Digital Image Processing" Second Edition, Prentice Hall, 2002.

[5] Kenneth R. Castleman, "Digital Image Processing", Prentice Hall, 1995

Face Detection by Color Segmentation and Morphological Image Processing

Deng-Hung Liu
Department of Electrical Engineering and Computer Science,
EECS 490 Digital Image Processing, Midterm Project
Case Western Reserve University, Cleveland, OH, Email: dxl74@cwru.edu

Abstract

Face detection has been a fascinating problem for image processing area during the last ten years because of many important applications such as video face recognition at some public area and security check point, digital image archiving, etc. In this report, I attempt to detect faces in a digital image using some basic image processing skills such as skin color segmentation, and morphological processing. Reasonable results were obtained with color segmentation, morphological image processing.

Keywords

Face detection, color segmentation, morphological

Introduction

We always take detection faces for granted because of we have a pair of eyes. Once if we want to create a computer program to perform the same task turns out to be a very difficult problem for which more effective and more efficient algorithms continue to surface. I simply use color segmentation and various morphological image processing to implement face detection.

While doing the face detection, I am wondering what is the definition of " face detection" . Finally, I found a paper which has a detail definition:

As Dr.Ming-Hsuan Yang and Dr. David J.Kreigman defined face detection : Given as arbitrary image, the goal of face detection is to determine whether or not there are any faces in the image and, if present, return the image location and extent of each face. The challenges associated with face detection can be attributed to the following factors:

- Pose. The images of a face vary due to the relative camera-face pose (frontal, 45degree, profile, upside down), and some facial features such as an eye or the nose may become partially or wholly occluded.
- Presence or absence of structural components. Facial features such as beards, mustaches, and glasses may or may not be present and there is great deal of variability among these components including shape, color and size.
- Facial expression. The appearance of faces are directly affected by a person's facial expression.
- Occlusion. Faces may be partially occluded by other objects. In an image with a group of people, some faces may partially occlude other faces.
- Image orientation. Face images directly vary for different rotations about the camera's optical axis.

Brief introduction of other detection method

Face detection is a hot subject that has already been widely studied all over the world. So it's fairly easy to get some information.

1. Face detection using Haudorff distance: First, a face model is built taking the average image of a training set containing face images, which have passed through an edge detector. Then all the pictures to be processed will also go through that same edge detector. Second, a sliding window scans the image, and the Hausdorff distance between the face model and the window is computed.
2. Eigenfaces: This method based on KLT and on PCA is of moderate complexity but is very powerful. First of all, e can apply the scheme on color faces and use color information to improve detection. Another advantage is that building eigenfaces does not require a big training set using Sirovich and Kirby method: 30 face images are sufficient.
3. Neural-Network based face detection: It first scans the image at different scales and applies a set of neural network-based filters to each block of pixels. Once this is done we have a list of possible face locations in the images but some of them correspond to the same face or are false positives: an arbitration program then corrects these artifacts.

Overview

In this section, I outline the main points of the face detection algorithm. The images are always taken as very high resolution due to the fancy digital camera. Though face detection can be done at much lower resolution. The input image is downsampled to reduce computational complexity. Then I do the color segmentation, which I can eliminate the non-skin part. I implement this part in HIS color space and I tune the threshold according to the picture. After skin color segmentation, I use various morphological image processing method to refine the image, like eliminate noise,and the part I don't want, etc. At last, I got the black and white logical image. Each white spot means a face. I sketch the block diagram below.

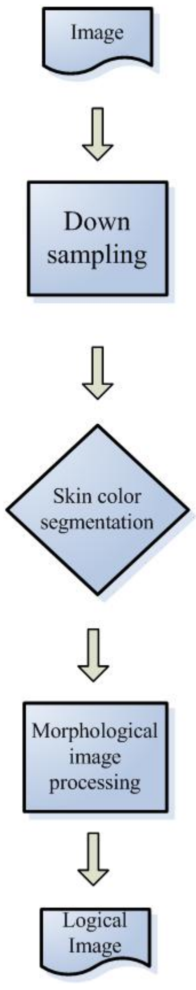


Figure 1.Block diagram

Skin color segmentation

The first step of doing face detection is skin color detection. And I have to decide which color model we want to implement. I think the HSI (Hue, saturation, intensity) space is the best color space for color segmentation because color is conveniently represented in the hue image. To accomplish the task of separating skin pixels form non-skin pixels ,I have to know the histogram of skin color in HIS space. Histograms were created in each of the three coordinates of HIS space using each of the two inputs (skin and non-skin). Figure 1 below shows the results of these histograms.

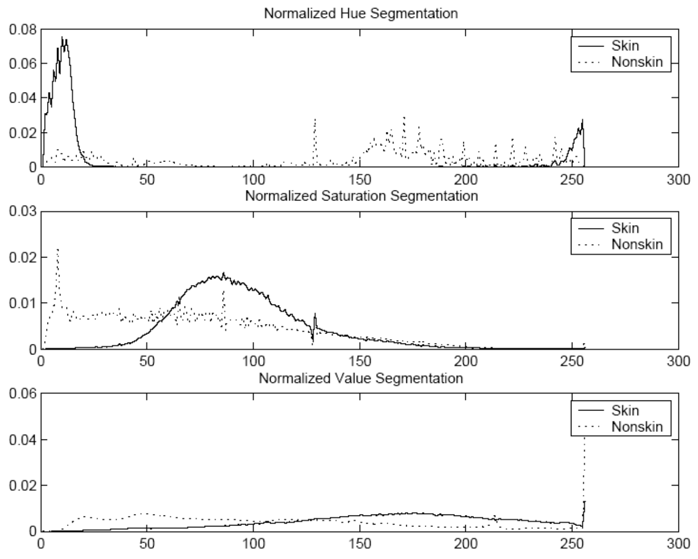


Fig 2.Skin color histogram in HIS space.

As the histogram shows, we can the skin color location in Hue domain.

Skin color (Hue)<20 and >230 ,

Figure 2 and 3 show the results of skin segmentation. I use the image which professor posted on website.

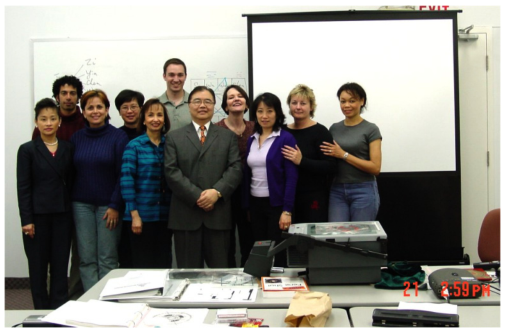


Figure 3. Original color image

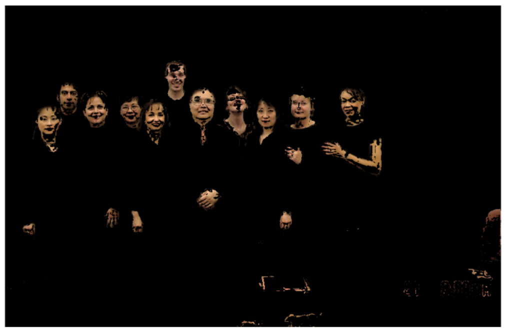


Figure 4.The skin color image.

As we can see above, we also detect some region we don't want, like arms, hands, legs, etc. So we should remove these parts from the skin color image.

Morphological Image Processing

We can see Figure 3 shows that skin color segmentation did a very good job of rejecting non-skin colors from the input image. However, the resulting image has quite a lot of noise and some parts we have to eliminate. A series of morphological operations are performed to cleanup the image and eliminate some parts we don't want.

- Intensity thresholding : We can eliminate some background by thresholding and also break up dark regions into many smaller regions so that they can be clean up by morphological opening. But we need to covert the image to gray scale first.

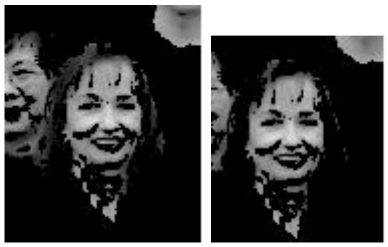


Fig 5. Remove hair by thresholding.

- Dilation and erosion: We can simply say the dilation can "thinner" the object and erosion can "thicker" the object. When combining these two operations, we can remove irrelevant part in terms of size.
- Closing and opening: Morphological opening is performed to remove very small objects from the image while preserving the shape and size of larger objects in the image. The definition of a morphological opening of an image is an erosion followed by a dilation, using the same structuring element for both operations. Closing also tends to smooth sections of contours but, as opposed to opening, it generally fuses narrow breaks and long thin gulfs, eliminates small holes, and fills gaps in the contour.
- Hole filling is done to keep the faces as single connected regions in anticipation of a second much larger morphological opening. I always use hole filling accompany with closing.
- Shrink: This operation will shrink objects with no holes to points; shrink objects with holes to rings. I use this operation to remove parts like arms.

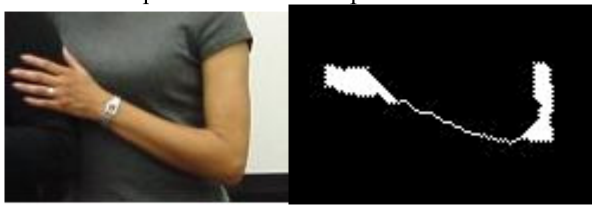


Fig 6. Shrinking the arm and use opening to remove

Special events

In this section, I mention some problem when I did the face detection.

1. Hands : Hands show the most often in common picture. Especially when the pictures include some "small" face and "large" hands. The picture one in results show the problem.

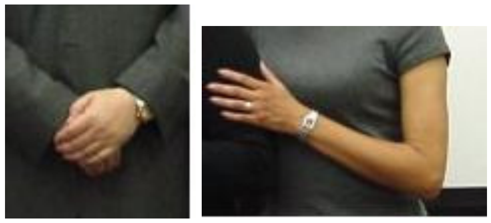


Figure 7.Hands and arms

2. Arms: Arm is along and thin part when compare with faces. But it cost huge area in the skin color image. When you do some morphological image processing, it will often connect to the noise or the part you don't want to select.
3. Occlude faces: It not easy to separate the occlude faces, especially only appear half or less on the picture. And it has to be very careful when doing some morphological processing , like closing.

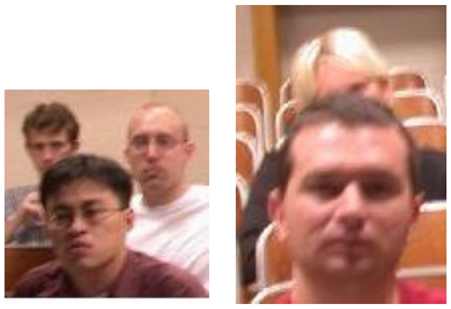


Figure 8. Occlude faces.

4. Color issues: When something has exactly the same color space as human skin, it cause problem. Sometimes are caused by flashlight (like classroom chairs), and sometimes are caused by the weather, like sun and cloud or maybe reflection by water or glass.

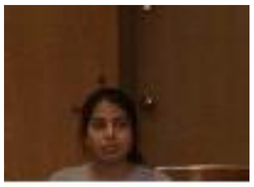
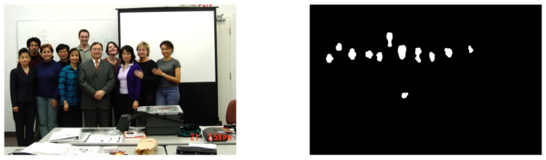


Figure 9. Skin color at the background.

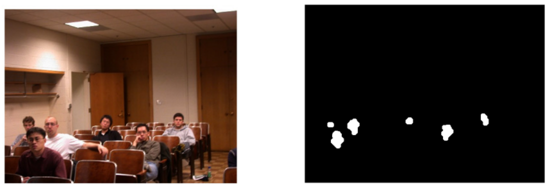
Results

Here are the results of the training pictures set. I think they look reasonable.

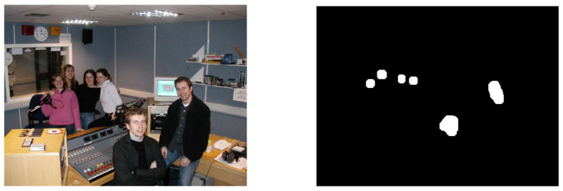
PICTURE 1



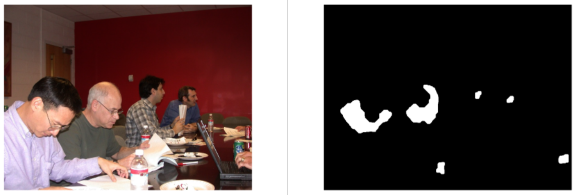
PICTURE 2



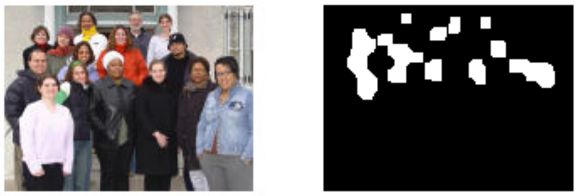
PICTURE 3



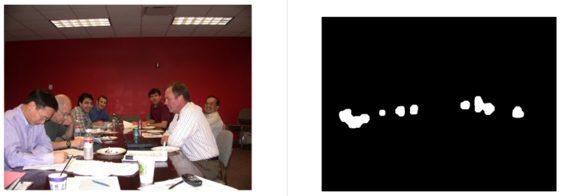
PICTURE 4



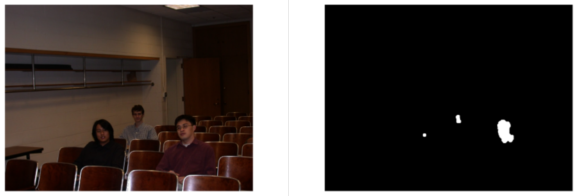
PICTURE 5



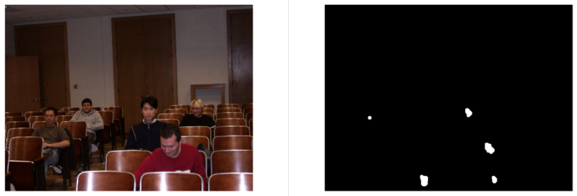
PICTURE 6



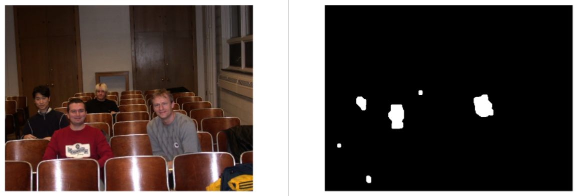
PICTURE 7



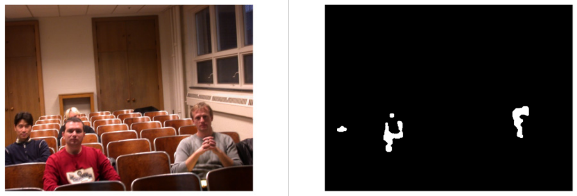
PICTURE 8



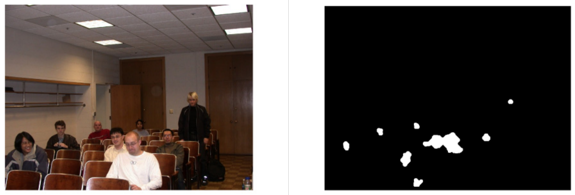
PICTURE 9



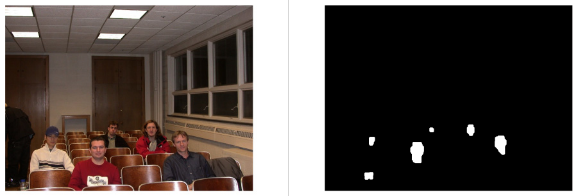
PICTURE 10



PICTURE 11



PICTURE 12



PICTURE	ACTUAL	DETECTED	ERROR

	FACES	FACES	
1	11	11	1(HANDS)
2	6	6	0
3	6	6	0
4	4	4	2(HANDS)
5	14	N/A	N/A
6	7	7	0
7	3	3	0
8	5	3	2(CHAIRS)
9	4	4	2(CHAIRS)
10	4	4	0
11	8	7	1(CHAIRS)
12	5	5	1(CHAIRS)
TOTAL	63	60	9

Picture 5 is a very low resolution picture, I can't do all the processing due to the low resolution. But I have already detected the face area.

Table.1 The results of the processing.

According the results, I can detect 95% faces in the image but I also has 1/7 errors. Most errors are due to the classroom chairs. Because when professor use the flashlight, the chairs' color and intensity looks exactly the same as skin. Also the size is similar to the face. It cause lots of errors.

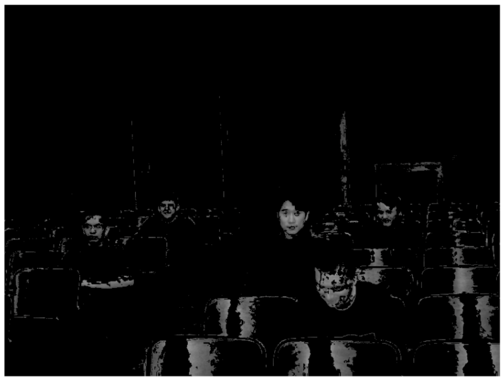


Figure10. The chairs affect the outcome.

Summery

As the initial step of the algorithm, skin color segmentation was by far the most effective means of eliminating non-face regions from consideration. For the other steps, I just implement various morphological operations. But I think if I have more time, I am willing to learn some sophisticated approaches. Due to this project, I found many interesting algorithm in digital image processing area. This is a very interesting project for a graduate student. But I think the loading for this project may be a little bit heavy. At least, I finished.

Reference

[1] Rafael C. Gonzalez, Richard E. Woods, "Digital Image Processing".

[2] Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins, "Digital Image Processing using Matlab".

[3] Computer operation via face orientation by P. Ballard and G.C. Stockman; Pattern Recognition, 1992 . Vol.1. Proceedings 11th IAPR International Conference on Computer Vision and Applications, 30 Aug.-3 Sept. 1992 Pages:407 - 410

[4] Detecting faces in images: a survey by Ming-Hsuan Yang; D.J. Kriegman, and N. Ahuja; IEEE Transactions on Pattern Analysis and Machine Intelligence,Volume: 24 , Issue: 1 , Jan. 2002. Pages:34 - 58

[5] Low-dimensional procedure for the characterization of human faces by L.Sirovich and M. Kirby, in J. Optical Society of America, Vol. 4, No. 3, Page 519-524, March 1987.

Face Recognition
Color segmentation and Principal Component analysis

Iouri Petriaev

Department of Electrical Engineering and Computer Science,

Case Western Reserve University, Cleveland, OH, Email: iap4@cwru.edu

Abstract

This paper presents the design details of an algorithm, which will locate faces in a given mage containing one or more faces. The goal of the designed algorithm is not to recognize the person, but simply to locate faces in an image. Common techniques used in this discussion include color segmentation and principal component analysis. Weaknesses and possible improvements to the resulting solution are also analyzed.

KEYWORDS

Color Segmentation, Object Segmentation, Principal Component, Correlation, IPT, Fourier transform, frequency domain, RGB

INTRODUCTION

Segmentation is one area in which better results generally are obtained by using RGB color vectors. Objective of using color segmentation is to classify each RGB pixel in a given image as having a color in the specified range or not. Performing pixel comparison, it is necessary to have a measure of similarity. There are three popular ways of performing this measurement: solid sphere, elliptical, and bounding box. First two methods make a good use of Euclidean distance between the RGB vector of average color 'a' and an arbitrary point in RGB space 'z'.

$$D(z,m) = \| z-m \| = [(z-m)^T (z-m)]^{1/2} =$$
$$[(z_r - m_r)^2 + (z_G - m_G)^2 + (z_B - m_B)^2]^{1/2}$$

Although, mathematically convenient, the implementation of these first two methods is computationally expensive for images of practical size. The third method is a compromise in which the box is centered on 'a', and its dimensions along each of the color axes is chosen proportional to the standard deviation of the samples along each of the axis. The color segmentation is achieved by determining whether or not the arbitrary color point is on the surface or inside the box.

A generalization of the distance equation mentioned above is a Mahalanobis distance measure of the form

$$D(z,m) = [(z-m)^T C^{-1} (z-m)]^{1/2}$$

where C is the covariance matrix of the samples representative of the color we wish to segment.

Assume that n denotes number of MxN images located on top of each other. There are n pixels for any given pair of coordinates (i, j), one pixel at that location for each image.

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$$

These pixels form vector

There are total of MN vectors containing all the pixels in the given images.

The mean vector is obtained by sample average:

$$m_x = \frac{1}{MN} \sum_{k=1}^{MN} x_k$$

The covariance matrix obtained by

$$C_x = \frac{1}{MN-1} \sum_{k=1}^{MN} (x_k - m_x)(x_k - m_x)^T$$

(MN-1) is used to obtain an unbiased estimate of covariance matrix C for the samples. Covariance matrix C is symmetric and real.

Principal Components used as the basis for describing sets of images that are registered spatially, but whose corresponding pixel values are different. It is used to describe boundaries and regions in a single image as well. One of the most important Hotelling transform properties deals with the reconstruction of x from y, where x is the group (vector) of n corresponding pixels of the n original images, and y is a vector of mapped x-vectors

$$y = A(x - m_x)$$

where elements of y are uncorrelated, and the covariance matrix C is diagonal.

DESIGN OF THE ALGORITHM

To improve performance, all images loaded for testing were scaled down to 700x700 by this algorithm. Every image was a color image containing multiple faces. Faces shown on the pictures had different level of shades overcastting them. Most of them had distinct skin-tone. There was a number of faces that were not fully visible or located upright.

For every image loaded by the application, the mask of an arbitrary face was taken by using MATLAB function *roipoly.* This method was chosen so the user can have freedom to select any desirable part of an image. Since mask consists of ones within enclosed area selected by the user and zeros elsewhere, a fully colored representation of the mask was obtained by:

```
red = immultiply(mask, J(:,:,1));
green = immultiply(mask, J(:,:,2));
blue = immultiply(mask, J(:,:,3));
g = cat(3, red, green, blue);
```

Obtained mask was used to retrieve principal –components of the mask. At first, we would extract vectors from an image stack of mask using *imstack2vectors* function. The principal-component images were obtained by function *princomp(X, 3)*, where *X* is the vector population contained in the rows of *X*, and 3 indicates the number of eigenvectors used in constructing the components.

One of the obtained principal-components was passed for segmentation. In this solution, we tried to pass one of each component at first to determine which one of three would yelled the best result. The segmentation is performed at first by calling function *reshape* which returns m by n matrix whose elements are taken column wise from the component passed to it as first argument. Then non-zero elements found by function *find*. We calculate the mean vector *m* by calling function *covmatrix*. The actual color segmentation performed by *colorseg('euclidian',J,25,m)* method call where *J* is the actual image, 25 is the largest standard deviation, and *m* is described above mean vector.

Using principal component to obtain the description of the mask and then using that description to segment image left us with relatively large number of segments on the image. Quite often, majority of these segments were objects not related to the faces and the algorithm clearly needed some logic to distinct or filter faces from the rest of elements. An attempt to do so consisted of creation of Gaussian low pass filter and applying correlation operation to the image in hopes that small bits of noisy pixels would get averaged with the background.

$$H(u,v) = e^{-D^2(u,v)/2\sigma^2}$$

Threshold filter is applied as the last process in this algorithm.

RESULTS

Obtaining principal components of the mask of the first image yelled relatively clear results stating that the principal component of the mask corresponding to the largest eigenvalue provides better comparisons. This is not unexpected result, for the eigenvalues are the variances of the elements of the *y* vectors. The *Figure 1* on the next column shows original image followed by three images of the three principal components.

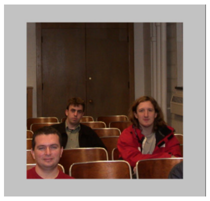


Figure 1. Original image

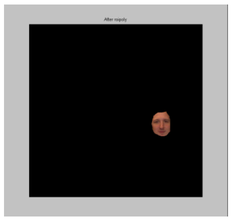


Figure 2. Mask

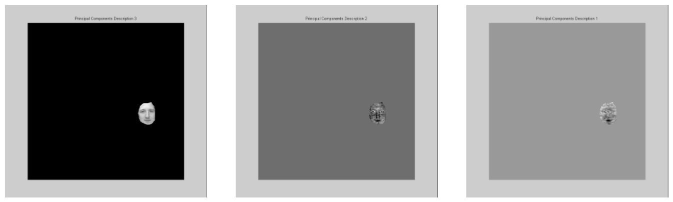


Figure 3. Principal components of the mask. First component corresponds to the highest eigenvalue.

Table 1 shows eigenvalues corresponding to the principal components above.

Table 1. Eigenvalues

λ_1	λ_2	λ_3
600.2772	0.5358	0.3014

Three images below corresponds respectively to the eigenvalues listed in the Table 1. These images are output of the algorithm and suppose to demonstrate recognized faces. Only first image that corresponds to the largest eigenvalue contains three faces. This result is not surprising. As was mentioned before, the eigenvalues are the variances of the elements of the *y* vectors, and the larger the variance the bigger the contrast of an image.

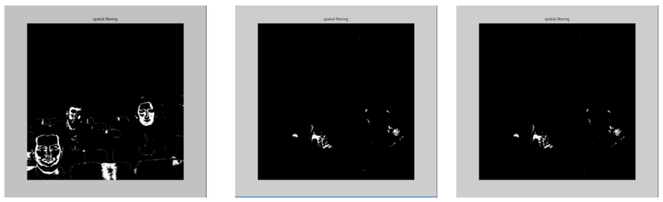


Figure 4. Images produced by the algorithm. Each image corresponds to the aigenvalue above. (see Table 1)

Beside faces on the first image a number of edges and bright spots can be seen as well. These elements of an image are shown for they may have very similar color spectrum to that of faces. The solution for this problem that was chosen in this design was described above. The following discussion will concentrate on three images and attempts to identify faces in them.
Following three images are the original images that will be discussed further.

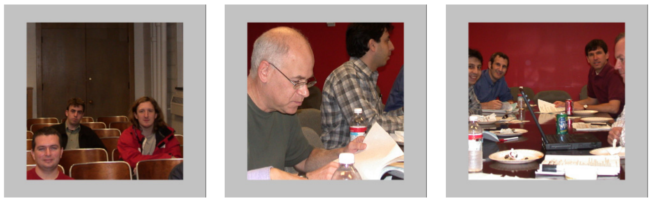


Figure 5. Original images.

Each of three original images contains multiple faces; moreover, each one of them has enough disturbances such as glare on the chairs (first image), discrepancy in color of faces (second image), partially visible faces (second and third images).
The following three images show masks selected from each original one by calling *roipoly* method during separate executions of algorithm.

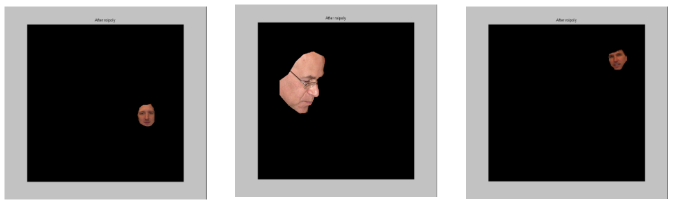
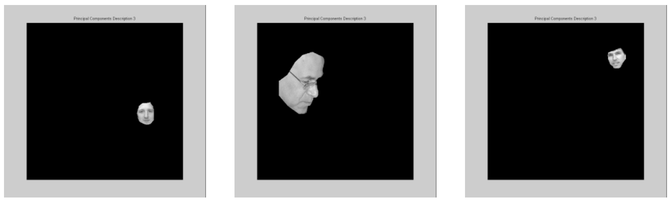
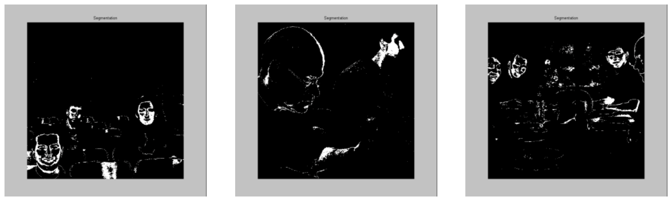


Figure 6. Masks selected for each separate image by using MATLAB function *roipoly*

Each mask contains a face presented under different angle then the other two. This will allow us to test how much position of the object on the image will effect the result produced by this algorithm.
Principal components that correspond to largest eigenvalue in each image were produced next. Due to the fact that principal components have relatively general description (some amount of image information is omitted by principal components), there is a good chance that the algorithm will be able to

Figure 7. Principal components generated from masks selected by user from the images.


The following collection of images shows segmented images with pixels matching the mask equal to 1 and others equal 0. Every image contains a number of unrelated and unwanted elements that were selected by the segmentation algorithm. In the first image, we can see light reflection coming from the chairs in the room. The glare coming from the first chair is particularly noticeable. The second image has highlighted visible part of the face located in the upper right corner. The face located on the upper left corner has only its edge selected. Last image has a great amount of reflection coming from the wall. The closest face to the camera just like the one on the second image has only its edges highlighted. This can caused by the difference in the color between the faces located closer to the camera and those located farther away from it.

Figure 8. Segmented images


Each image in the following collection of three images is modified by the low-pass Gaussian filter. Although not very noticeable on the images provided below, application of this filter using correlation caused to some unwanted elements of the image (noise) to fade into the background. Filter mask used for all three images was of 64 by 64 size, and had standard deviation σ=0.9. Any attempt to increase the mask and standard deviation would eliminate the edges and features of the faces visible on the images below.

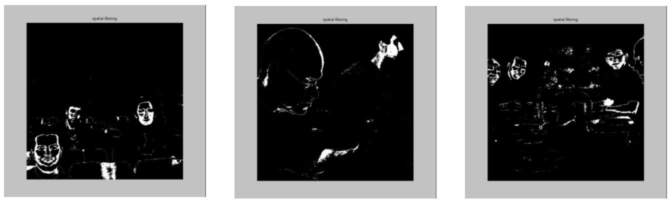
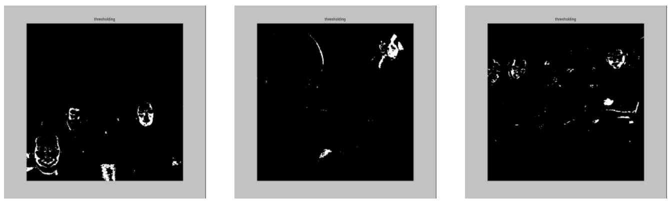
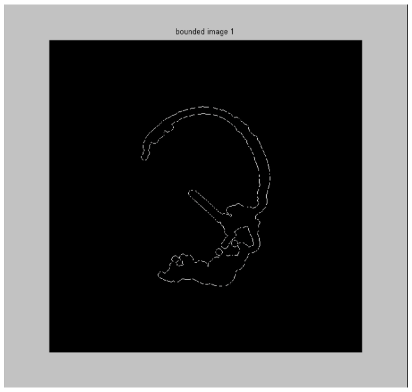


Figure 9. Gaussian low-pass filter applied to all three images.

To eliminate noise with faded intensity but not completely disappeared from the image all three images were filtered by threshold filter. Any pixel greater then 0.9 was set to 1, and to 0 otherwise. From the images below we can see that a good amount of noise was eliminated. Unfortunately, a god amount of data corresponding to the faces was eliminated as well.
Figure 10. Threshold (g>0.9) applied.


The attempt was made to use Hough transform and line detector to separate faces from the noise. The purpose of doing so was to preserve as much face data as possible and separate each face from the rest of the image by having distinct boundaries identified. The existence of boundaries of other objects in the image interconnected among each other and faces made it close to impossible. The image below shows one successful attempt of doing so.



SOURCE CODE
The source code of the main method is provided below.

```
%Load images
%f = imread('DSCN1126.jpeg');
%f=imread('DSCN1141.jpeg');
%f=imread('DSCN1118.jpeg');
f=imread('DSCN1120.jpeg');
f1 = f(:,:,1);
f2 = f(:,:,2);
f3 = f(:,:,3);

%calculate subimages located at the center of current image
%and 512x512 dimentions
offset = 700/2;
[row,cols]=size(f1);
center_x = row/2;
center_y = cols/2;

%select centered 700x700 subimages
f1 = f1((center_x - (offset-200)):(center_x + ((offset+200)-1)),(center_y - offset):(center_y + (offset-1)));
f2 = f2((center_x - (offset-200)):(center_x + ((offset+200)-1)),(center_y - offset):(center_y + (offset-1)));
f3 = f3((center_x - (offset-200)):(center_x + ((offset+200)-1)),(center_y - offset):(center_y + (offset-1)));

J = cat(3,f1,f2,f3);
mask = roipoly(J);

red = immultiply(mask, J(:,:,1));
green = immultiply(mask, J(:,:,2));
blue = immultiply(mask, J(:,:,3));
g = cat(3, red, green, blue);
figure,imshow(g), title('After roipoly');

%Principal components for description
%Oranize the stack into array X.
[X,R] = imstack2vectors(g);

%Obtain the principal-component images by using q=3.
P = princomp(X,3);

%Generate and display all component images.
g1 = P.Y(:, 1);
g1 = reshape(g1,700,700);
g1 = gscale(g1);
figure,imshow(g1),title('Principal Components Description 1');

g2 = P.Y(:,2);
g2 = reshape(g2,700,700);
g2 = gscale(g2);
figure,imshow(g2),title('Principal Components Description 2');

g3 = P.Y(:,3);
g3 = reshape(g3,700,700);
g3 = gscale(g3);
figure,imshow(g3),title('Principal Components Description 3');

%Color segmentation
[M,N,K] = size(g);
R = reshape(g, M*N, 3);
idx = find(g3);%mask);
R = double(R(idx, 1:3));
[C, m] = covmatrix(R);
e25 = colorseg('euclidean', J, 25, m);
figure, imshow(e25), title('Segmentation');

e25 = im2double(e25);
w = fspecial('gaussian', [64 64], 0.9);
e25 = imfilter(e25, w, 'corr', 'replicate');
figure, imshow(e25), title('spatial filtering');

%apply threshold
e25 = e25 > 0.9;
figure, imshow(e25),title('thresholding');
```

```
%try Hough Transform
e25 = houghtrans(e25);
figure,imshow(e25),title('hough transform');
```

Source code for Hough transform function.

```
function g3 = houghtrans(f)
%Uses Hough Transform to center text document
%in a digital image
%gtheta = pi/6;
g = real(f);

%Transform
deltax = 0;
deltay = 0;
s = 1;%0.5;

[m,n]= size(g);

%Get longest boundary.
b = boundaries(g);
d = cellfun('length',b);
[max_d, k] = max(d)
bound = b{k(1)};
d(k) = 0;

%Convert to image.
bim = bound2im(bound,m,n);
figure, imshow(bim), title('bounded image 1');

[max_d, k] = max(d)
bound = b{k(1)};
d(k) = 0;
%Convert to image.
bim = bound2im(bound,m,n);
figure, imshow(bim), title('bounded image 2');

[max_d, k] = max(d)
bound = b{k(1)};
d(k) = 0;
%Convert to image.
bim = bound2im(bound,m,n);
figure, imshow(bim), title('bounded image 3');

[max_d, k] = max(d)
bound = b{k(1)};
d(k) = 0;
%Convert to image.
bim = bound2im(bound,m,n);
figure, imshow(bim), title('bounded image 4');
```

SUMMARY

The design of the color segmentation and principal component analysis application for face recognition was demonstrated. The algorithm developed was certainly able to extract faces from images through using user's selection of the mask. This algorithm also exposed the complexity of the image segmentation. The technique proposed in this discussion was composed of color segmentation, principal component analysis, spatial and frequency filtering.

Main concentration of this solution was devoted to colors. Concentrating on a single property (feature) of an image would be rather risky attempt. As was shown in this discussion, issues such as segmentation of an object surrounded by noise may require additional techniques such as neural networks, feature extraction from an image based on geometric properties of elements of objects such as eyes and nose on a face.

REFERENCES

[1] Rafael C. Gonzalez, Rechard E. Woods, Steven L. Eddins, "Digital Image Processing using MATLAB", Pearson *Prentice Hall*, 2004.

[2] Rafael C. Gonzalez. Richard E. Woods, "Digital Image Processing", Prentice-Hall, 2002

Face Detection in Complex Scene Images using Color Segmentation and Morphological Techniques

Chris Roberts, Frank Merat

Department of Electrical Engineering and Computer Science,

Case Western Reserve University, Cleveland, OH, Email: Robert.Roberts@case.edu

Abstract

This paper presents a method for the detection of faces in a complex scene image using color segmentation and morphological techniques. These techniques allow for the facial structures to be mostly isolated before the process of detecting the faces using template matching occurs. The use of these methods makes human facial structures stand out so that they can be detected. This algorithm has the potential for use in areas such as personnel identification, homeland security, and aiding the disabled in computer use [3].

KEYWORDS

Facial Recognition, Color Segmentation, Image Morphology, Template Matching

INTRODUCTION

With the invention of fast computing and inexpensive high resolution digital imaging devices there has been a steady interest in research centered on the detection and processing of faces in images. Facial recognition has become a large topic in recent times to help law enforcement in the identification of criminals and other persons of interest at our border, within our country, and throughout the world. Other uses that have been explored include the use of facial cues to control a computer for the physically challenged [3].

One basic facial recognition technique that can be employed is to use an average face image as a template, and perform template matching [4]. The average face is generated by combining many images into a common image that will likely resemble faces found in an image [8]. This template can then be convolved with the image where faces are to be detected. Given that the face in the image is scaled to match the template and is in full view, and that the face is very distinct in the picture, a maximum will be created where the template matches the face on the image. This maximum can then be processed to extract the face from the image for further processing.

Another method used for facial recognition in images is to apply mathematical transforms to the template image to create an eigenface [5]. This eigenface is a numerical representation of a face that can then be used to computer other faces in an image.

In real world conditions, faces are often obstructed or skewed in an image. Also, aside from idealized conditions such as photo studios, the backgrounds of most images are commonly filled with very complex and colorful items such as landscapes, furniture, buildings, etc. This “noise” in an image makes the task of identifying faces in that image much more difficult. A robust method must then be used to defeat this noise. Other robust methods include knowledge based methods, and appearance based methods [4]. Knowledge based methods rely on using the knowledge of what makes up a human face such as eyes, ears, a mouth and a nose. Appearance based methods use a variety of training images to “learn’ what a face looks like for the use of facial recognition. Neural Networks may be facilitated to aid in this learning process.

Color also plays an important role in the detection of faces in images. Skin color is often very different than the surrounding colors in an environment. The hue and chrominance of the skin can also set its color apart from similar colors in the environment, such as sand or wood, if the appropriate techniques are applied [1]. There are a variety of color spaces that can be used, such as RGB, CMYK, HSI, and a series of chrominance color spaces [2]. By converting an image into a different color space, detecting the face should become easier. The colors in an image are greatly affected by the lighting of the environment which can change the skin tones dramatically, and can also have a large affect on the identification process.

A common technique employed to isolate a color difference is to implement color segmentation of an image. Segmentation looks at the region of an image, such as a face, and uses the mean color and its deviation to extract only the similar colors in an image [7]. A binary mask can then be created that can eliminate portions of the image that fall outside of the desired color range. This can eliminate much of the noise in an image.

Morphological Techniques are another common image processing technique use to manipulate grayscale or binary images. Readily existing implementations of morphology can provide tasks such as removing small amounts of noise from an image mask, and reducing regions of interest to a single point [6]. These algorithms are extremely useful in face detection as they can help remove small regions of skin tone such as hands and feet, that color segmentation does not eliminate.

TECHNICAL APPROACH

The technical approach taken was to create a simple and relatively fast method to locate the faces on an image. The first step is to read in the image containing the faces to be detected. An original of this is stored, and a copy is used for processing.

The template image is then read into the program [8]. It is scaled to match the relative size of the faces in the image, and is cropped to use only the eyes and the nose for the template. This template is then padded with zeros to match the dimension of the image is will be compared with.

Both of these images are then converted from RGB space to HSI space to ready them for color segmentation and processing.

The image containing the faces is then used to select a region to be segmented. A face containing the average looking color of all of the faces in the image is used, and then the segmentation algorithm generates a binary valued image mask that isolates these color regions.

Noise in the mask such as hands is present, so the morphological technique of erosion is used to eliminate these small regions of noise. Dilation is then applied to grow the remaining regions of the mask.

This mask is then multiplied with the Intensity layer of the image to isolate only the facial regions. It is then convolved with the intensity layer of the padded template to find the correlation between the two images. This is done in the frequency domain as it is much faster.

The correlation image is then thresholded to become a binary image with only the highest regions of correlation remaining a non-zero value.

A morphological technique to collapse the remaining non-zero regions is applied so that the mask contains only single pixels; each where a face has been detected. The coordinates of these pixels are then used to place a mark on the original stored image to indicate where the algorithm detects an image.

RESULTS AND DISCUSSION

To initially develop this algorithm an image was chosen where all of the faces present were of similar size and were not obstructed much, as shown in Figure 1. This would allow for easier testing of the different techniques to assure that the algorithm works.

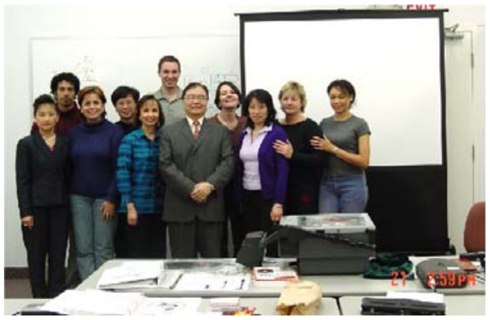


Figure 1 – The original image supplied, apparently obtained from Google Images.

Next, the average facial image was read in, scaled and cropped as shown in Figure 2. This image is very small, so after padding, the image was mostly zeros. Using different templates such as an entire head, a single eye, and a mouth were explored. All of these templates provided poorer results than using the eye and nose combination.

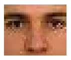


Figure 2 - Template Image

Using the color segmentation code, and carefully adjusting the parameters of the size of the erosion process, a color segmentation mask that eliminates most of the limbs and noise besides the faces was generated, as seen in Figure 3. Adjusting the morphology to retain all of the faces, while rejecting other features in the image proved very difficult and retaining all of the face data without leaving significant noise was a problem. Other techniques should be employed to aid with this in future experiments.

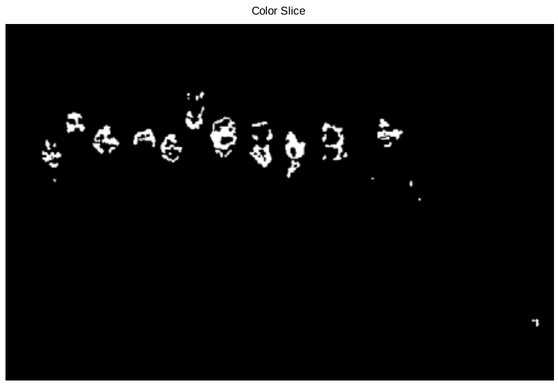


Figure 3 - The Color Segmentation Mask

This mask was then easily applied to the intensity layer of the image and the correlation image from the convolution was generated. Since some non-facial regions remained in this process, and the faces are very close together in this image, adjusting the threshold of this image was very important. If the threshold was set too low, noise would remain in the mask, and the faces would appear to be connected together. If the threshold value was set too high, some of the faces might be lost in the thresholding process. After much experimentation with the threshold level, a successful mask was created with all eleven of the faces being recognized as near single regions, and all other elements in the image were eliminated, as seen in Figure 4.

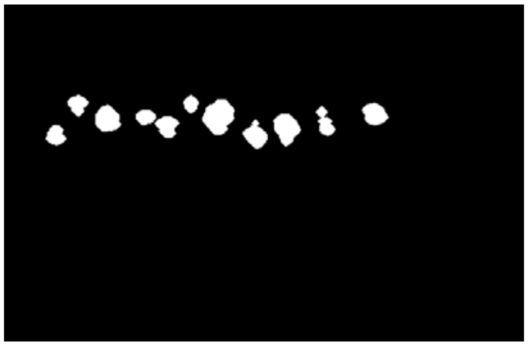


Figure 4 - The identified facial regions

To assure that each face region was made up of a single area, dilation was applied to this mask to expand the regions slightly. This mask was then shrunk down using morphological operations to have 11 sets of single pixels whose coordinates were applied to mark the faces on the original image as a white dot, shown in Figure 5.

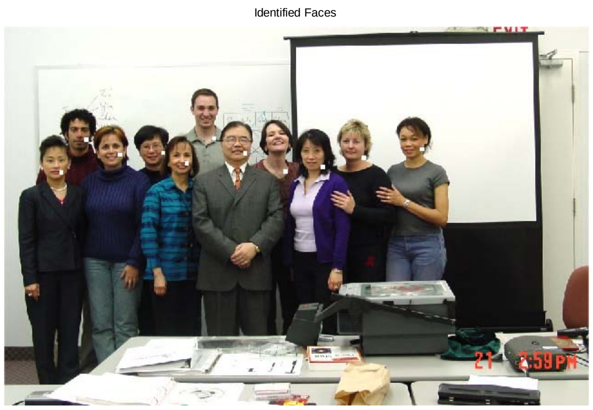


Figure 5 - Image showing the algorithms identification of all faces in a picture.

Looking closely at where the algorithm placed the marker, it is usually found in the lower left hand side of the face, not in the center by the eyes and nose where one would expect to see the highest correlation. Also, in a few of the cases, the mark is on the neck of the subject in the picture.

Exploration of this phenomenon revealed that the convolution flips a signal before correlating the two images. With this in mind, the template image was flipped and the faces located. This had little to no change on the location of the face markers.

Further exploration and a morphological lecture showed that the morphological operations operate from the upper right to the lower left corner of an image. This seems to indicate that as the operation shrunk each of the regions in the mask down to a pixel; it slowly shrank each region down to the lower left hand pixel, which would explain the markers location on the image. Another technique such as the hit or miss algorithm should be explored to find the center pixel of each region, to better indicate the face location.

Expanding the algorithm to images such as the classroom image below in Figure 6 provided interesting and important results.

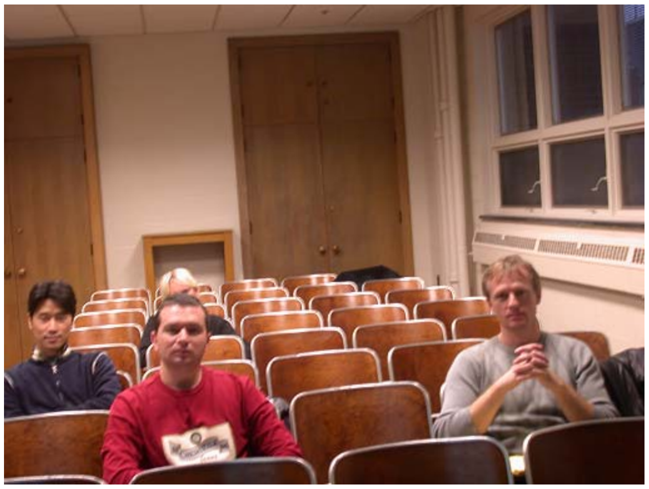


Figure 6 - Another sample image with obstructed faces and flesh colored backgrounds.

Figure 6 has an obstructed face, as well as chairs and a wall behind the subjects that share a very similar color to the faces in the image.

When applying the segmentation technique to this image, it was not possible to extract the facial data without extracting a large amount of the background. This rendered the morphological operations to dampen small amounts of noise useless, and the algorithm was highly ineffective at detecting faces.

Overall, exploration with this algorithm proved that it was very sensitive to the lighting conditions and color differences in an image. It would work very well at distinguishing faces in a controlled environment where there is a consistent lighting condition and a background that contrasts facial colors in an image.

The morphological operations are also very sensitive to the size and orientation of the regions they are operating on. Since the masks generated vary greatly from image to image, the parameters are very high maintenance to keep the algorithm working correctly.

One nice thing about this technique is that it finds the faces very quickly. The operation takes about 10 seconds on a 1.4 Ghz computer with 512MB of RAM. The algorithm could then be used to find the region close to the face, a bounding box could then be generated to extract the region containing the face, and more complex algorithms could be applied to identify the face or to find the identity of the person in the image.

This algorithm would also prove useful in other areas, such a quality control and item identification in an industrial setting. Since these applications offer a consistent background environment, a new mask could be created to identify another object, and the algorithm could be useful identifying this object in a controlled setting where the variables for the morphological operations and the threshold can remain constant. A sample application could use this algorithm for counting components on an assembly line.

SUMMARY

Facial recognition in images is a large field of research with strong industry backing. Facial recognition techniques are currently used for security, personnel identification, and other applications. A wide variety of methods must be employed to differentiate and extract facial regions from the complex environments that many images are created in. This pre-processing of images to eliminate as much extraneous information as possible proves a much larger technical hurdle than the actual correlation process of template matching the images.

The algorithm proposed and implemented in this paper was successful at identifying all faces in a sample image. The algorithm did not find the center of the faces due to the methods employed, and proved to be less than robust when it came to new lighting conditions that changed the color of the faces in the image. Background colors that matched the flesh tones of a face also proved too large of a challenge for the implemented algorithm.

This paper gives a basic technique of one method of identifying the faces in an image. Other techniques to eliminate extraneous information or new techniques to identify the faces themselves can be applied to increase the robustness of this basic facial recognition algorithm.

ACKNOWLEDGMENTS

This work was done as a midterm project for the EECS 490 class taught by Dr. Frank Merat. Special thanks to Dr. Merat for providing guidance for this project, as well as the test images used in its implementation.

REFERENCES

[1] L. Torres, J. Y. Reutter, and L. Lorente. "The Importance of Color Information in Face Recognition." *IEEE International Conference on Image Processing*, Kobe, Japan, October 25-28, 1999.

[2] J.C. Terrillon and S. Akamatsu, "Comparative Performance of Different Chrominance Spaces for Color Segmentation and Detection of Human Faces in Complex Scene Images." Proceedings *Vision Interface Conference 19-21 May, 1999.* pp. 180-188, 1999.

[3] P. Ballard and G.C. Stockman, "Computer Operation via Face Orientation," *Proceedings 11th IAPR International Conference on Computer Vision and Applications, 30 Aug.-3 Sept. 199*2, vol. 1, pp. 407 - 410, 1992.

[4] Ming-Hsuan Yang; D.J. Kriegman, and N. Ahuja, "Detecting Faces in Images: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 24 , No 1. . pp. 34 - 58. Jan. 2002

[5] L.Sirovich and M. Kirby, "Low Dimensional Procedure for the Characterization of Human Faces," *J. Optical Society of America*, Vol. 4, No. 3 pp. 519-524, March 1987

[6] Rafael C. Gonzalez, Richard E. Woods and Steven L. Eddins, Digital Image Processing Using Matlab. Prentice-Hall, Upper Saddle River, new Jersey, 2nd Ed. 2004

[7] Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing. Prentice-Hall, Upper Saddle River, new Jersey, 2nd Ed. 2004

[8] Average Faces. Beauty Check. http://www.uni-regens-burg.de/Fakultaeten/phil_Fak_II/Psychologie/Psy_II/beautycheck/english/durchschnittsgesichter/durchschnittsgesichter.htm. Modified: 7 January 2002. Viewed 27th November 2004.

Face Detection and Localization in Images Using Color Segmentation and Template Matching

Ira Ross

Department of Electrical Engineering and Computer Science

Case Western Reserve University, Cleveland, OH, Email: ira.ross@case.edu

ABSTRACT

This paper presents a face detection and localization technique for intelligently identifying faces in color group photographs. The algorithm utilizes color segmentation to isolate human skin based on its chrominance properties, performs simple morphology on the mask, then passes the masked result through a correlation procedure with a defined average face. Highly correlated points indicate areas where the source image most represents facial features. These points are processed and extracted to produce a single set of coordinates at each instance of a face. With proper thresholding, this technique for face detection and localization is able to identify all faces in group photos of forward facing subjects.

KEYWORDS

Face detection, face localization, color segmentation, image morphology, template matching

INTRODUCTION

In the world of image processing, there are many motivations for the ability to autonomously detect and analyze the human face. For national security purposes, there has been a recent push towards facial and retinal recognition in order to cross-reference potentially threatening individuals against an image database of known terrorists. On another level, detection of facial features and expressions is being used by some computers to interactively gain information about a user's identify, state, and intent [3].

The first step of facial recognition is detecting the locations in an image where faces are present. Extensive research on the subject has produced many distinct approaches to the problem of face detection in an image. The basic strategies employed can be split into four categories – knowledge-based methods, feature invariant methods, template matching methods, and appearance-based methods [3]. Knowledge-based, top-down methods operate on a set of rules established for how a basic face is comprised, and check for the presence of these properties in the source image. Feature-based methods are bottom-up, and aim to define facial features that are invariant of lighting, angle, or pose. Template matching methods employ an image of an average face or facial features, and find the correlation between the template and the source image. Last, appearance-based methods learn the properties of a face from a set of representative face templates, and calculate facial candidates based on this information [3].

This project utilizes both bottom-up feature-based methods and template matching methods in order to accurately identify human faces. First, potential face candidates are extracted by segmenting the source image based on skin tone. Because skin can be uniquely represented independent of luminance in a small range of chrominance, the YCbCr color space is chosen in order to perform the segmentation [4]. The result is a mask that can be multiplied by the original image, but must first be enhanced using a morphological opening and closing to eliminate noisy patterns. Finally, the template matching method is employed by taking the correlation of the masked areas with a scaled image of an average face. The areas of highest correlation are computed, and used to find the coordinates of the faces.

COLOR SEGMENTATION

If incorporated properly, color segmentation based on skin tone can be a very powerful tool for facial detection. By initially narrowing the detection field to areas representing human skin, segmentation saves valuable time and increases the success rate of other methods. For this paper, the YCbCr color space is used, because research has found that skin can be accurately represented independent of luminance within a small range of chrominance (Figure 1) [4,5].

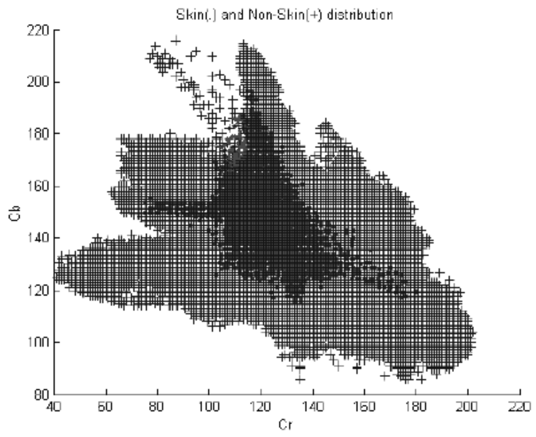


Figure 1. Distribution of skin (.) and non-skin (+) pixels in Cb vs. Cr chrominance plane [5]

Experimentally determining the range of chrominance for skin gives a rectangular region spanning 77 ≤ Cb ≤ 127, and 122 ≤ Cr ≤ 173 [5]. This is merely an estimate for segmenting skin tone based on chrominance, but it will work sufficiently for extracting important areas from the test image. The result of implementing this range for skin segmentation can be seen in Figure 2.

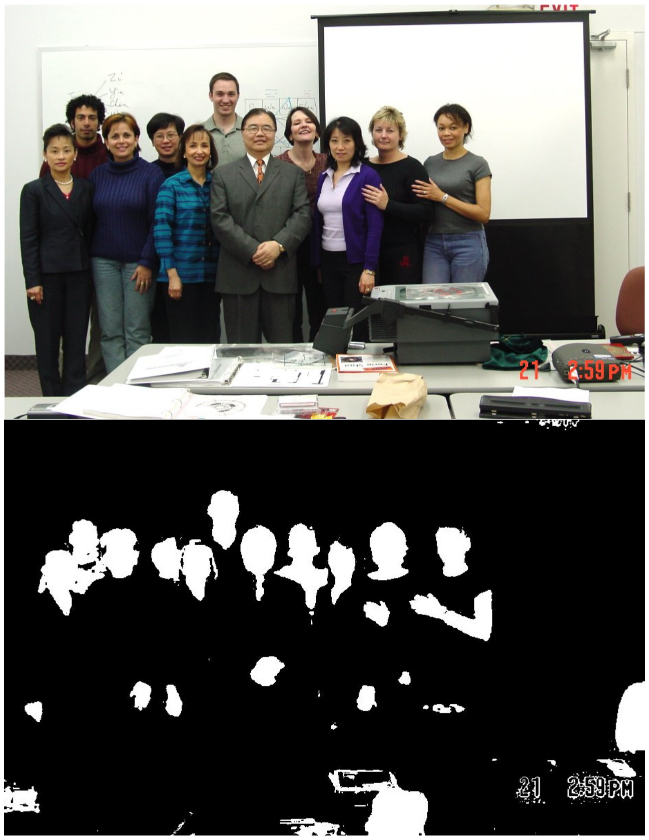


Figure 2. Original image before and after color segmentation based on skin tone chrominance

Segmentation produces a very useful mask, but it is desired to eliminate objects smaller than the size of a human head in order to facilitate a more accurate correlation procedure. It is necessary to perform some basic morphological image processing to the mask, which will be discussed in the next section.

MORPHOLOGICAL OPERATIONS

To get a more accurate result with the template matching detection method, further processing of the color segmentation mask is needed. The morphological operations of dilation and erosion are used in different combinations for the elimination of unwanted areas that do not represent the shape of a human face. Using a specified structuring element, dilation expands an image's borders in areas where the element overlaps its edges. Conversely, erosion contracts an image's borders to where the structuring element fits inside. By executing an erosion followed by a dilation, one can perform an image *opening* - *s*moothing the contour of an object, breaking narrow isthmuses, and eliminating thin protrusions. Similarly, an image *closing* is a dilation followed by an erosion, which will smooth contours, fuse narrow breaks, eliminate holes, and fill gaps [1].

Since color segmentation lets everything pass within a specified chrominance range, areas that do not resemble the basic shape of a face need to be removed. By using an image opening with a properly sized circular structuring element, it is possible to cut out these unwanted areas from the mask. This particular image requires a disk structuring element with a radius of approximately 16 pixels to achieve the desired opening (Figure 3).

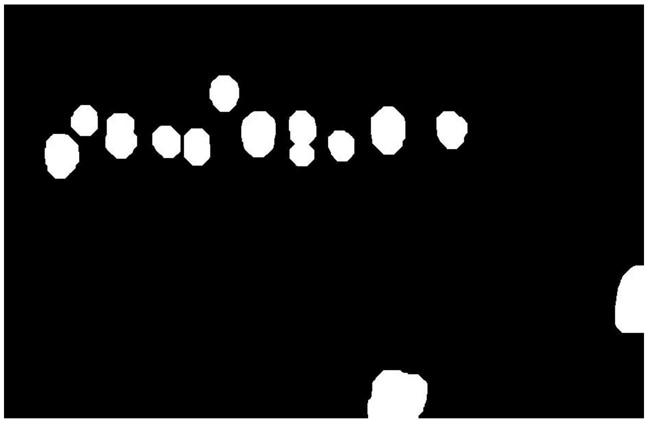


Figure 3. Skin color segmented mask processed by morphological opening with circular element

The new mask is now multiplied by a grayscale version of the original image (Figure 4) for use in the template matching procedure.

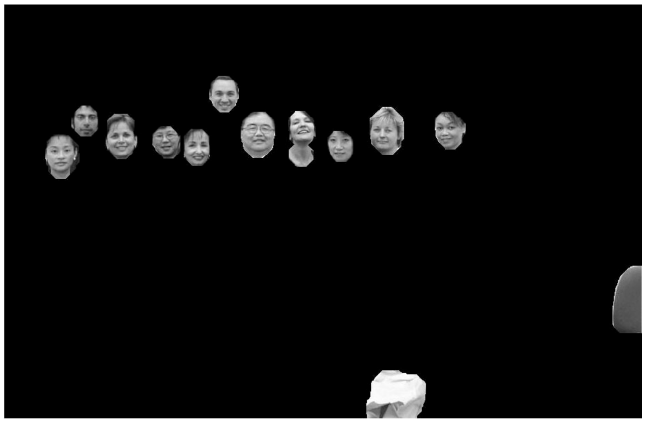


Figure 4. Morphed mask multiplied by grayscale of original group photograph

TEMPLATE MATCHING

The last step in this project's face detection procedure is to run the processed image through a template matching algorithm in order to locate points where the image is most correlated with an average face (Figure 5). First, a Sobel filter is applied to both the average face and the source image to extract edge information. Using edge filtered images will result in a more accurate correlation procedure [3].

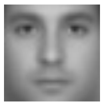


Figure 5. Image of an average face used in template matching correlation procedure

Instead of trying to perform a correlation in the spatial domain, it is both easier and faster to convert to the frequency domain. Simply take the 2-D FFT of the Sobel filtered source image, along with the 2-D padded FFT of the filtered average face. The conjugate of the face FFT multiplied times the FFT of the original image gives their correlation in the frequency domain. By taking the real portion of the inverse FFT of the product, the spatial correlation is found (Figure 6) [2].

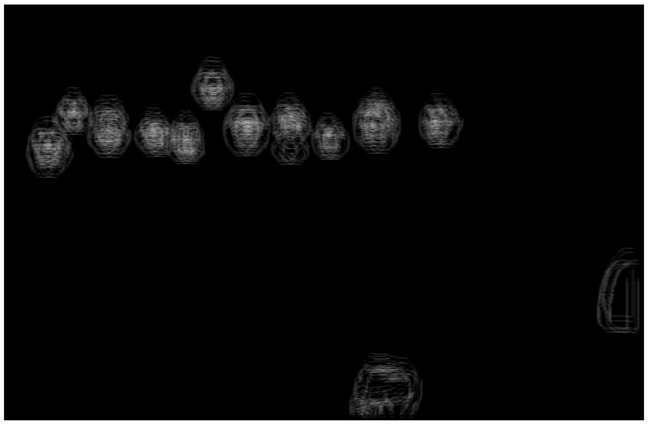


Figure 5. Correlation of original masked image with average face

The correlation output is passed through a threshold to eliminate pixels below a certain grayscale value. With this image, a threshold of 170 is used to eliminate all extraneous points, but to preserve at least one point from each face. Because some faces are more correlated to the average face than others, it is necessary to combine any set of points denoting a single face. A morphological dilation is performed to connect all points within a certain neighborhood, followed by the MATLAB function "bwmorph" to shrink all connected regions to a single point [2]. Once a pair of coordinates is found for each face in the image, a loop is implemented to increment through points and to draw a white square around the perimeter.

RESULTS AND DISCUSSION

Using the correct parameters, the facial detection algorithm worked remarkably well at a 100 percent detection rate in the first image with no false detections (Figure 6). The high success rate depended on choosing the proper size and type of structuring element used for the morphological operations on the skin tone mask. Another factor that attributed to success was that the image had very few objects with color close to human skin tone.

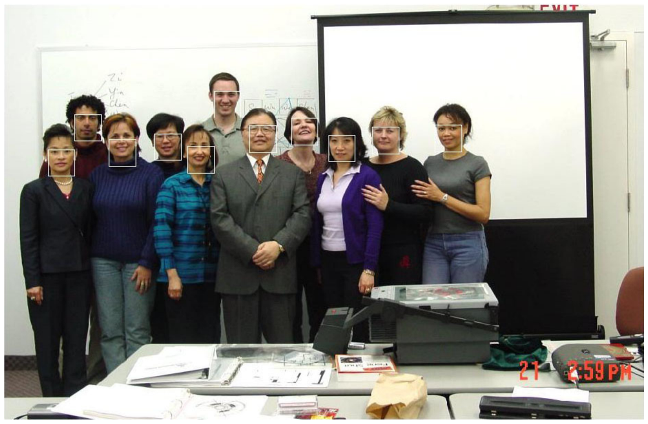
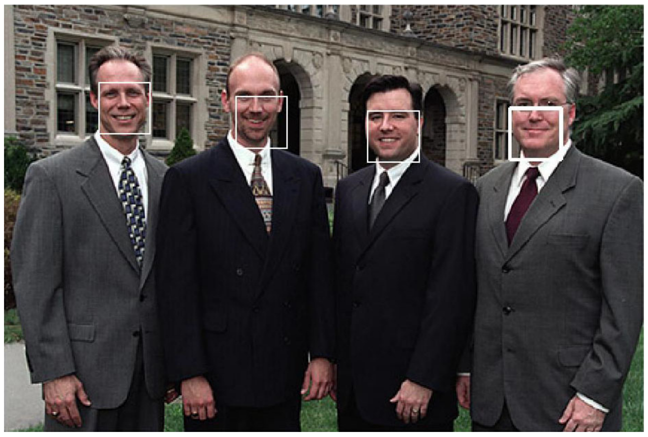


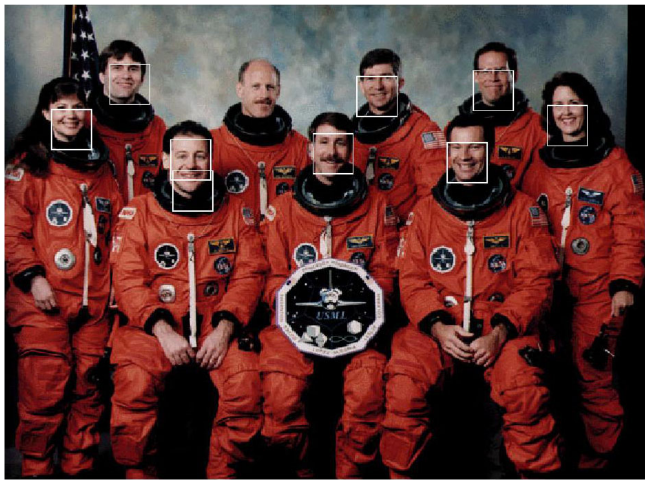
Figure 6. Final result of face detection procedure – detected faces are outlined with a white box

One of the largest disadvantages to using template matching is that the template must be approximate in size to the faces in the image. Therefore, any images with multiple face sizes should be passed with several scaled average faces, followed by taking the intersection of the results to get an accurate correlation. Although it would be enhanced by this addition, the method presented in this paper has shown to be best suited for detecting faces in a fixed environment where face size is held relatively constant and objects close to skin tone are not present.

When tested with other images, the detection program performs quite well. As shown in Figure 7(a), it is able to accurately locate the faces of four men against a background that could possibly cause problems some types of color segmentation. With this image, it was necessary to adjust the size of the morphological structuring element so that it did not eliminate some faces. Using the same parameters set for Figure 7(a), the result shown in Figure 7(b) was achieved with a NASA team photograph. Only eight out of nine faces were detected, most likely due to the fact that the missed face has no eyebrows, a mustache, and is not completely forward-facing. Overall, out of 24 total faces in the three photographs, the face detection program was able to accurately locate 23 faces with no false positives. These results are highly dependant upon the size and type of photograph chosen to pass into the face detection software.



(a)



(b)

Figure 7. Additional testing of face detection method

To create a truly bottom-up face detection program that does not depend as much on face size and background color, it is necessary to implement a feature-invariant method instead of template matching. A well written neural net will have much more success in recognizing faces and locating finer facial details. The skin chrominance model used in this project was also a rough estimate of the actual Cb-Cr distribution. Using finer modeling based on Gaussians would most likely give a more accurate segmentation that does not pick up as much background information.

ACKNOWLEDGEMENTS

This work was inspired by Professor Frank Merat, who has done a superb job teaching Digital Image Processing for the fall semester of 2004 at Case Western Reserve University.

SUMMARY

A method for face detection and localization based on color segmentation and template matching is presented in this paper. From a set of three properly scaled images, the method was able to detect 23 out of 24 faces for a success rate of 96 percent. By combining multiple image processing techniques, the program effectively eliminated any cases of false identification.

Future work on the method should employ a more detailed model for skin chrominance, as well as another feature-invariant detection method to help classify faces of different angle, size, and pose. Face detection is the initial step for employing an effective facial recognition procedure where finer facial details need to be located and analyzed.

REFERENCES

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed., Prentice-Hall, 2002.

[2] R. C. Gonzalez, R. E. Woods, and S. L. Eddins. *Digital Image Processing Using MATLAB*, Prentice-Hall, 2002.

[3] M . Yang, D. J. Kriegman, and N. Ahuja, "Detecting Faces in Images: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, Jan. 2002.

[4] H. Wang and S.F. Chang, "A Highly Efficient System for Automatic Face Region Detection in MPEG Video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 4, Aug. 1997.

[5] P. H. Lee, V. Srinivasan, and A. Sundararajan, "Face Detection," Stanford University.

Face Detection

Yu-Hong Yen

Department of Electrical Engineering and Computer Science,

Case Western Reserve University, Cleveland, OH, Email: yxy61@cwru.edu

Abstract

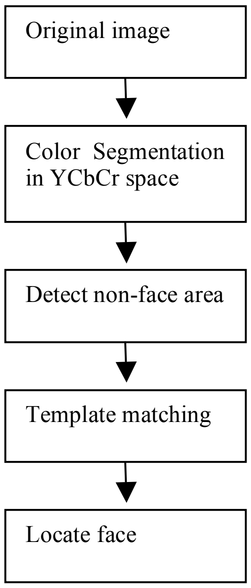
The purpose of this project is to detect faces in various images. There are many different applications in which a face detection program could be used. The querying of image databases is one possible application that would use face detection. Also, face detection is the first step in the process of face recognition. Many surveillance companies could make use of programs that can reliably scan a surveillance photo, and recognize certain individuals. In order to recognize a person in an image, it is first necessary to find the face of each person in that image. This type of program is especially useful in places such as airports to find criminals.

KEYWORDS

Color segmentation, Morphological image processing, Face Detection

METHOD

The flowchart of this project is as following:



The original image is as figure 1:

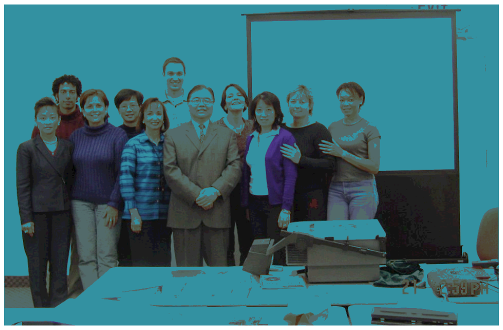


Figure 1

The first step of the program is to modify the original image in which we are intending to detect faces. In order to eliminate lighting effects (luminance), it is necessary to take the original color image and convert the colors into chromatic color ("pure color") space. The format of the RGB space (original image) includes luminance, which makes it difficult to characterize skin colors because lighting effects can change the appearance of the skin. The chromatic color space will eliminate the luminance component. To convert an image from RGB to chromatic colors you simply compute:

$r = R/(R+G+B)$
$b = B/(R+G+B)$

The value of the g component is the same as the r and b values and r+b+g=1. To convert from RGB to chromatic color space in Matlab, the function "rgb2ycbcr" is used. The following is the original image, and the original image converted to chromatic color space:

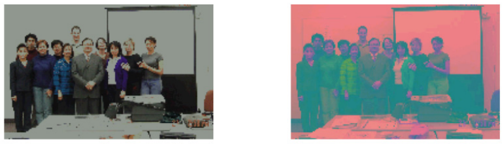


Figure 2

The next step is to create a skin model in chromatic color space. To create the skin model, it was necessary to use several images with people of varying skin colors. I decided to use 5 different images of people with different skin colors from the original image. Using these different images of people, I cropped small portions of skin from each image and created new images with only the cropped portions of skin. After reading in each of the skin color images, they were then converted to chromatic color space as previously described. It will generate a filter that can filter the non-skin color area. The result is as following:

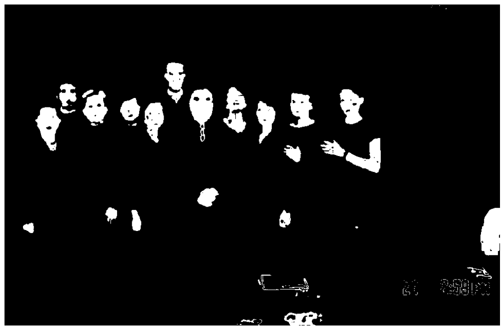


Figure 3

It is now necessary to evaluate each segmented skin region to determine if it is a face. The current segmented image shows any skin regions such as arms, legs or any other skin area. Since faces consist of eyes, noses and a mouth, it is safe to say that a face would consist of at least one hole in the segmented image. To determine the number of holes in a region, the following equation is used:

E = C – H

In this equation, E is the Euler number, C is the number of connected components and H is equal to the number of holes. Since we are analyzing only one segmented region at a time, the C is equal to one.

H = 1 – E

The Euler number is determined by the “bweuler” function in Matlab. If an area with at least one hole is found, we then continue to find some statistics about the region to be used in the template-matching portion of the code. Area is found by using the “size” function in Matlab. The center of mass is determined by:

$$\bar{x} = \frac{1}{A}\sum_{i=1}^{n}\sum_{j=1}^{m} jB[i,j]$$

$$\bar{y} = \frac{1}{A}\sum_{i=1}^{n}\sum_{j=1}^{m} iB[i,j]$$

The orientation angle is found by:

$$\theta = 1/2 \, \mathrm{atan} \frac{b}{a-c}$$

$$a = \sum_{i=1}^{n}\sum_{j=1}^{m}(x'_{ij})^2 B[i,j] \quad b = 2\sum_{i=1}^{n}\sum_{j=1}^{m} x'_{ij}x'_{ij}B[i,j] \quad c = \sum_{i=1}^{n}\sum_{j=1}^{m}(y'_{ij})^2 B[i,j]$$

$$x' = x - \bar{x}$$
$$y' = y - \bar{y}$$

Finally the Width and Height are determined. A ratio of height to width is then determined. Since faces normally have a ratio of about 1, this parameter can be used to determine if the segmented area is a face. To be safe, I used a range of values from 0.6 to 1.2 in order for the segmented region to continue to be evaluated for face characteristics. Similar the part of the program that finds the number of holes, if the value is not within this range, the area is determined to not be a face and the next segmented area is then evaluated.

The next step is template matching. The basic idea of template matching is to convolve the image with another image (template) that is representative of faces. Finding an appropriate template is a challenge since ideally the template (or group of templates) should match any given face irrespective of the size and exact features. The following template face was made by Principal Components Process from 8 different faces in figure 5.

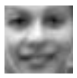


FIGURE 4

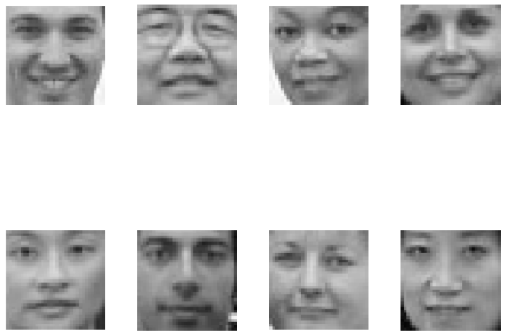


FIGURE 5

The template face is first resized according to the measurements taken on the segmented image. Based on the height and width of the segmented skin region, the template face is then converted to these dimensions so that it can later be placed in the segmented region. The theta of the segmented region is then used to rotate the template face to the same angle. The center of mass of the segmented skin region is used to place the template face directly in the center of the segmented image. This process will completely fill the segmented area with the image of the template face.

Once the template face is placed inside the segmented image, it is necessary to see how “well” the template fits inside the region. A way to determine this value is to use a correlation, which computes the two-dimensional correlation coefficient between two matrices. To find the correlation of these two matrices, we use the “corr2” function in Matlab. This function operates on the
following algorithm:

$$r = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{\left(\sum_m \sum_n (A_{mn} - \bar{A})^2\right)\left(\sum_m \sum_n (B_{mn} - \bar{B})^2\right)}}$$

It was found that a good correlation value between the two matrices is close to 0.6. If the correlation between the test face matrix and the segmented region matrix is 0.6 or higher, the original image is shown with the template face replacing this region and is shown as a grayscale image.

The same process is repeated by testing each segmented region for a height to width ratio between 0.6 and 1.2 and a correlation greater than or equal to 0.6. Once every region has been evaluated, the final product of the original color image is displayed with rectangles showing each of the detected faces in the image. Once we have successfully determined that the segmented area is a human face, a rectangle is placed around the face showing the program has detected a face in the image. Using the coordinates determined from the height and width based on the center of mass, the “rectangle” function in Matlab was used to create the boxes. The final result is as following:

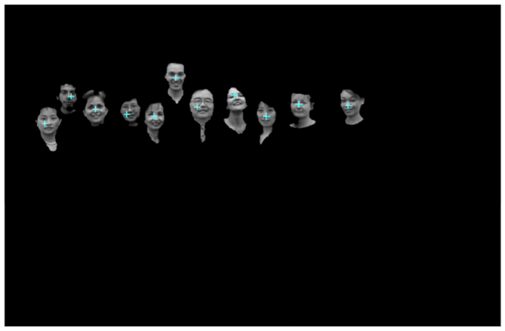


Figure 6

CONCLUTION

The task of face detection in a digital image is a well established problem. There are many approaches which all try to achieve the same end result: efficiently detecting all human faces in a given image and rejecting everything that is not a face. The illumination corrected template matching yields near perfect results.

Another area that this project could be expanded would be to add side face detection as well. This would be relatively simple to achieve, the only extra work would be to create an average side view of a person and repeat the process used in this program. Although the program is not perfect, for most applications, 83% accuracy would be sufficient. This type of program would work best for taking the first step in face recognition. For example, if a door was to be opened by a new security system, this program could be used. This program would be implemented by having the person stand in front of the camera (preferably with a solid background) next to the door and take a frontal picture of the persons face. This program would detect the positioning of the person’s face in the image and then the face recognition process could begin.

REFERENCES

[1] Jie Yang and Alex Waibel, "A Real-Time Face Tracker", CMU CS Technical Report.

[2] Rafael C. Gonzalez and Richard E. Woods, *Digital Image Processing, 2nd Edition*. Prentice-Hall.

[3] Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins: “Digital Image Processing Using MATLAB,” Prentice Hall; 1st edition

[4]http://ise.stanford.edu/2003projects/ee368/Project/reports/ee368group06.pdf

[5] Wei-min Huang and Robert Mariani. *Face Detection and Precise Eyes Location* IEEE 2000.