Course Number: EE 458

Title: Image Processing Final Project: Dr. Morales-Dr. Wolpert Classifier

Date: 5/1/2020

Students: Matthew Capuano

Professor: Dr. Aldo Morales

**Introduction:**

Students face many problems in the electrical engineering program at Penn State Harrisburg. However, a student’s greatest problem is arguably determining if they look more like Dr. Morales or Dr. Wolpert.

This project explores the use of image processing processes in order to detect faces in an image. It then sends the processed images into a neural network to classify the face as either Dr. Morales or Dr. Wolpert. Face detection is a useful image processing technique since it functions as a precursory step to facial recognition, which usually is implemented through machine learning techniques such as neural networks. In fact, both face detection and face recognition can and usually are implemented solely through neural networks, with minimal image processing techniques used. However, image processing techniques can sometimes be used when factoring in computational time, accuracy, and other constraints.

This project is an experimentation and adaptation of a set of processes and algorithms used in Thu-Thao Nguyen’s masters project “Real Time Face Detection and Tracking” from Cornell University. I take no credit for the sequences of preprocessing techniques used, and I am simply exploring them for educational purposes. Her project can be found in full detail in the sources section of the report.

**Methods:**

Color scheme conversion:

The first step in face recognition is skin detection. Human skin falls within a certain range of values on the RGB color scale. If one imagines the RGB color scale fitting on an XYZ axis, there is a certain volume of coordinate values that human skin will fall under. This volume is smaller if converting from RGB to YUV color schemes. YUV color uses luminance (Y), blue projection (U), and red projection (V). The conversion used is:

Thresholding and Segmentation:

There is now a range of U and V values that skin color falls under, and a segmentation algorithm can be applied as follows:

Morphological Filter:

Skin has now been detected in the image. Noise also exists in the image that prevents clean segmentation. This can be improved using MATLAB filters imerode and imfill. Imerode functions to remove groups of pixels that are smaller than a predetermined value. This helps to denoise the image and make the segmentation clearer. Imfill is then used to fill in what connected regions exist, further removing noise and creating a clean segmentation process. It essentially just sees if there are any black regions entirely covered by white pixels, and if so, turns the black regions white.

Labeling and Area Calculation:

Other objects with colors that fall under the same range as skin is picked up in the segmentation, and skin relating to regions other than the face is picked up as well. In order to discern the face from these other objects, the detected objects must be identified, labeled, and compared. In MATLAB, bwlabel is used to find connected objects, and regionprops is used to create a data structure of properties relating to each object. One of the properties is area, which will be used narrow down which identified objects could be a face. It is then assumed that the face area contains one of the largest areas compared to other skin areas, and ismember is used to only show segmented areas that are with 26% of the object with the max area.

Centroid and Border Calculation

At this point, perceived faces should only be left in the image. A centroid computation can then be used for each face in order to find its center location. From this centroid, an algorithm is used to move up, down, left, and right until a black pixel is reached. These are the border column and row values. A red line is drawn that then moves up and down from the left border column value to the top and bottom border values. A red line is drawn that moves from the bottom row border value to the left and right border column values. This process is repeated for the right and top border as well.

Mapping and Rescaling

From here, we can format our boxed area to be fed into a neural network. The boxed area is assigned to a new variable, and then the variable is rescaled to 50x50 pixels using imresize. The RGB color plane is then mapped to 1 plane, over a 50x150 image size. This picture can then be fed into a neural network for training and testing.

Training of Network

The 50x150 image is converted into a vector, and is assigned a desired output of 1,0 for Dr. Morales, or 0,1 for Dr. Wolpert. This image and 3 others are processed in this method and fed into the network for training.

**Results:**

Each part of methods section was implemented with figures 1 through 7 showing how each filter or technique effects the image.



Figure 1: original image

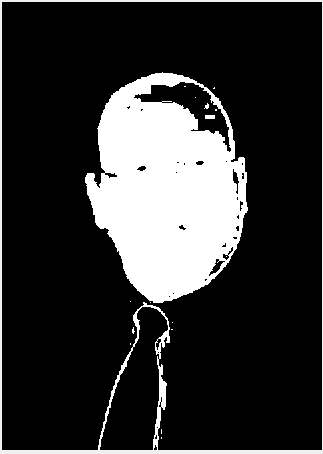


Figure 2: Image after applying thresholding and segmentation

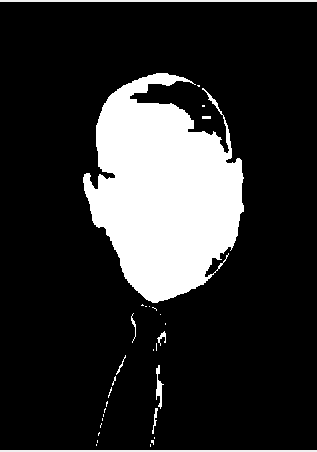


Figure 3: Image after applying morphological filters, notice areas such as eyes and glasses are removed



Figure 4: Image after applying area calculation to remove white areas not within 26% of the max area. This removes the tie.

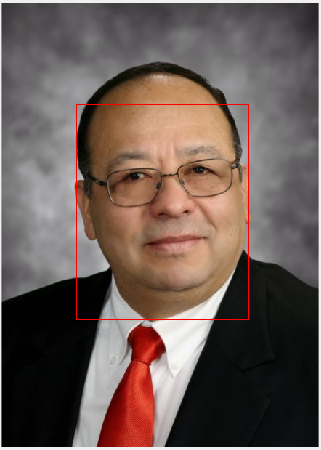
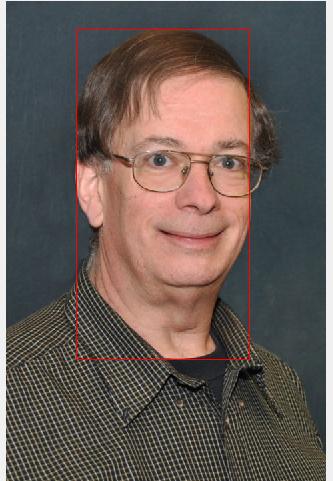
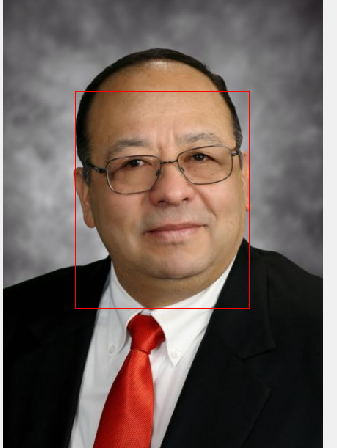


Figure 6: After centroid calculated from figure 5 image, border is applied to original image



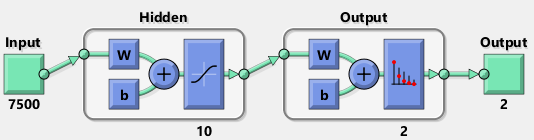
Figure 7: Image is cropped and rescaled to be fed into neural network for training. This image along with 1 other of Dr. Morales and 2 of Dr. Wolpert is fed in a neural in MATLAB and trained. This process is then repeated with a test image, and then fed into the trained network.

Below are the test input images:

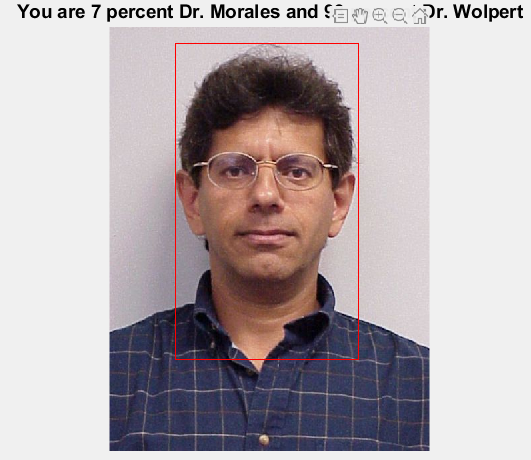


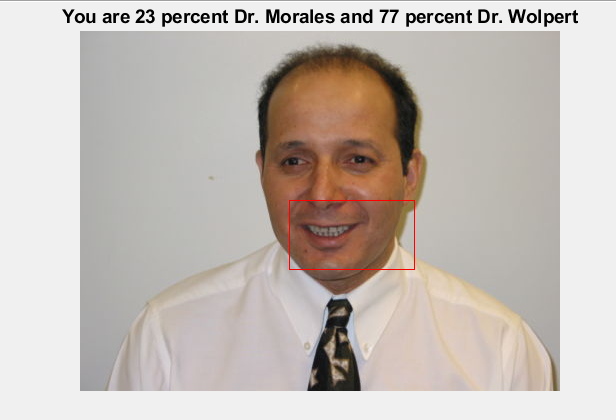
The images on the left are the original image with face detected. The images on the right are the image fed into the network for training.

The network is trained using a simple 10 layer feed forward architecture with sigmoid activation functions.



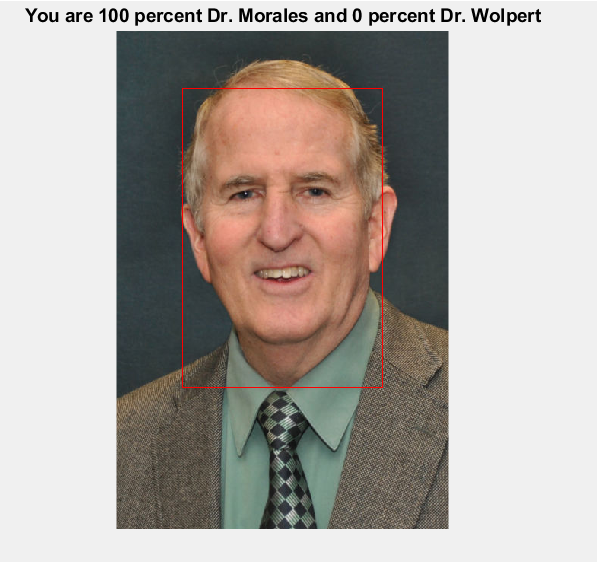
Once, trained, test images are fed through the system:

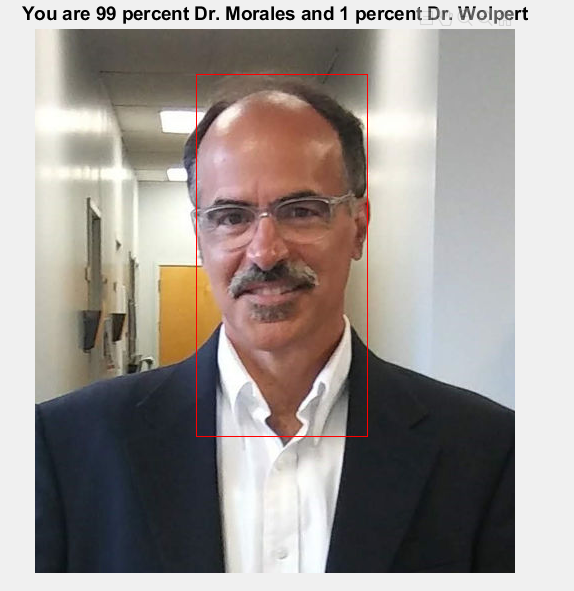


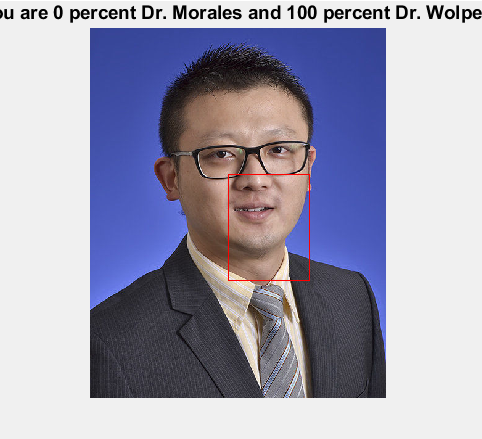


Black areas inside of face region caused border to cutoff early











The MATLAB code for implementing this algorithm is provided below.

function f = custom\_detect(n)

rgb\_img = n;

R = rgb\_img(:,:,1);

G = rgb\_img(:,:,2);

B = rgb\_img(:,:,3);

R = double(R);

G = double(G);

B = double(B);

img\_dimension = size(R);

img\_rows = img\_dimension(1);

img\_cols = img\_dimension(2);

Y = (R + 2\*G + B) / (4); %convert to YUV

U = R - G;

V = B - G;

%Note: v\_thresh has little significance to accuracy and may be left out to increase computation

U\_thresh = zeros(img\_dimension); %apply threshold

V\_thresh = zeros(img\_dimension);

for i = 1:1:img\_rows

for j = 1:1:img\_cols

if U(i,j) > 10 && U(i,j) < 74

U\_thresh(i,j) = 1;

else

U\_thresh(i,j) = 0;

end

end

end

for i = 1:1:img\_rows

for j = 1:1:img\_cols

if V(i,j) > -40 && V(i,j) < 11

V\_thresh(i,j) = 1;

else

V\_thresh(i,j) = 0;

end

end

end

skin\_seg = zeros(img\_dimension); %apply segmentation

for i = 1:1:img\_rows

for j = 1:1:img\_cols

if U\_thresh(i,j) == 1 && V\_thresh(i,j) == 1

skin\_seg(i,j) = 1;

else

skin\_seg(i,j) = 0;

end

end

end

%%%%%%%%%%%%%%%%%%%%%%%%%: Morph Filter ->connected components -> area calc -> area filter -> centroid calc

%apply morphological filters

skin\_seg1 = imerode(skin\_seg,strel('square',3)); %erode/remove small groups of pixels

skin\_seg1 = imfill(skin\_seg1, 'holes'); %fill holes in the face (if black region exists where entirely surrounded by white, make it white

%label regions and compute their area

[L, n] = bwlabel(skin\_seg1);

face\_region = regionprops(L, 'Area');

face\_area = [face\_region.Area];

%find and show only the regions with at least 26% area of max area

face\_idx = find(face\_area > (.26)\*max(face\_area));

face\_shown = ismember(L, face\_idx);

%5/1 centroid and box calcuation

s = regionprops(face\_shown, 'Centroid');

centroids = cat(1, s.Centroid); %cat puts all the s.Centroid values into amatrix

%store edge values of face for box, only calculates 1 centroid and box for

%now

ii = round(centroids(1));

jj = round(centroids(2));

while face\_shown(jj, ii) == 1

jj = jj + 1;

end

border\_bottom = jj;

jj = round(centroids(2));

while face\_shown(jj, ii) == 1

jj = jj - 1;

end

border\_top = jj;

jj = round(centroids(2));

while face\_shown(jj,ii) == 1

ii = ii + 1;

end

border\_right = ii;

ii = round(centroids(1));

while face\_shown(jj,ii) == 1

ii = ii - 1;

end

border\_left = ii;

ii = round(centroids(1));

%find left border and right

for i = jj:-1:border\_top

rgb\_img(i, border\_left, 1) = 255;

rgb\_img(i, border\_left, 2) = 0;

rgb\_img(i, border\_left, 3) = 0;

rgb\_img(i, border\_right, 1) = 255;

rgb\_img(i, border\_right, 2) = 0;

rgb\_img(i, border\_right, 3) = 0;

end

for i = jj:1:border\_bottom

rgb\_img(i, border\_left, 1) = 255;

rgb\_img(i, border\_left, 2) = 0;

rgb\_img(i, border\_left, 3) = 0;

rgb\_img(i, border\_right, 1) = 255;

rgb\_img(i, border\_right, 2) = 0;

rgb\_img(i, border\_right, 3) = 0;

end

%find bottom border and top

for i = ii:-1:border\_left

rgb\_img(border\_bottom, i, 1) =255;

rgb\_img(border\_bottom, i, 2) = 0;

rgb\_img(border\_bottom, i, 3) = 0;

rgb\_img(border\_top, i, 1) = 255;

rgb\_img(border\_top, i, 2) = 0;

rgb\_img(border\_top, i, 3) = 0;

end

for i = ii:1:border\_right

rgb\_img(border\_bottom, i, 1) = 255;

rgb\_img(border\_bottom, i, 2) = 0;

rgb\_img(border\_bottom, i, 3) = 0;

rgb\_img(border\_top, i, 1) = 255;

rgb\_img(border\_top, i, 2) = 0;

rgb\_img(border\_top, i, 3) = 0;

end

k =1;

z =1;

zzz = rgb\_img;

imshow(zzz);

for i = border\_top:1:border\_bottom %map boxed info into new picture

k = k + 1;

for j = border\_left:1:border\_right

morales0(k,z,1) = rgb\_img(i,j,1);

morales0(k,z,2) = rgb\_img(i,j,2);

morales0(k,z,3) = rgb\_img(i,j,3);

z = z + 1;

end

z = 1;

end

morales0 = imresize(morales0, [50 50]); %resize the picture

for i=1:1:50 %map rgb values to 1 color plane

for j = 1:1:50

y(i,j) = morales0(i,j,1);

end

end

for i = 1:1:50

for j = 51:1:100

y(i,j) = morales0(i, j - 50, 2);

end

end

for i = 1:1:50

for j = 101:1:150

y(i,j) = morales0(i, j-100, 3);

end

end

%convert to vector

y = y(:);

%output data

f = y;

end

Figure 1: Function for image processing

close all

clear all

rgb\_img = imread('morales.jpg');

morales = custom\_detect(rgb\_img);

morales = double(morales);

rgb\_img1 = imread('morales0.jpg');

morales1 = custom\_detect(rgb\_img1);

morales1 = double(morales1);

rgb\_img2 = imread('wolpert.jpg');

wolpert = custom\_detect(rgb\_img2);

wolpert = double(wolpert);

rgb\_img3 = imread('wolpert1.jpg');

wolpert1 = custom\_detect(rgb\_img3);

wolpert1 =double(wolpert1);

%run morales, morales\_1, wolpert, wolpert\_1 into NN for training

%concatenate vectors

NN\_in(:,1) = morales;

NN\_in(:,2) = morales1;

NN\_in(:,3) = wolpert;

NN\_in(:,4) = wolpert1;

NN\_desired(:,1) = [1 0];

NN\_desired(:,2) = [1,0];

NN\_desired(:,3) = [0,1];

NN\_desired(:,4) = [0,1];

net = patternnet(10);

net = train(net,NN\_in,NN\_desired);

q = view(net);

rgb\_img5 = imread('morales.jpg');

elaraby = custom\_detect(rgb\_img5);

elaraby = double(elaraby);

output=sim(net,elaraby)

percent\_morales = round(output(1) \* 100);

percent\_wolpert = round(output(2) \* 100);

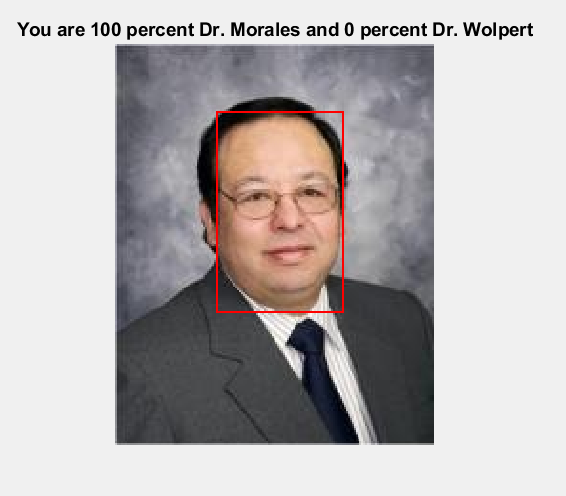
caption = sprintf('You are %d percent Dr. Morales and %d percent Dr. Wolpert', percent\_morales, percent\_wolpert);

title(caption, 'FontSize', 14);

Figure 2: Main function that calls the function in Figure 1 and then trains and runs the neural network

**Conclusion:**

This was an interesting and somewhat successful method for face detection in images. Most literature regarding face detection uses neural networks for every step of the process instead of image processing methods, so this is definitely a unique process for unique applications. It can easily be scaled for real time applications for microcontrollers and FGPA’s as well. I would like to implement the neural network portion of this project by hand sometime in the future if I ever have time, and then try to implement it in a real time processing system. It should be noted that there are many limitations to using this method, and it is inaccurate for many different types of pictures. It is also only programmed to perform centroid and box calculations for one face, but it can easily be scaled up for more if there are multiple faces in the picture. I am also not sure if the neural network I trained actually works since I am not sure it can make sense of my input information since I am just feeding the entire face image into the network, and it only trained off of 4 input images.



References: <http://people.ece.cornell.edu/land/courses/eceprojectsland/STUDENTPROJ/2012to2013/tnn7/tnn7_report_201212141110.pdf>