FH Vorarlberg
University of Applied Sciences

Color based face detection

Digital Signal Processing Project

University of Applied Sciences Vorarlberg
Master in Mechatronics

Submitted to

Prof. (FH) DI Dr. Reinhard Schneider

Handed in by

Burtscher Tobias
Stark Stefan

Dornbirn, 20.12.2016

Table of Contents

List of Figures

Bibliography

Amman, R. (2016). Image and signal processing - color. *FHV*, page 20.

Hjelmas, E. (2001). Face detection: A survey. *Computer Vision and Image Understanding*, 83(3):236–247.

Kumar, P. and M, S. (2014). Real time detection and tracking of human face using skin color segmentation and region properties. *International Journal of Image, Graphics and Signal Processing (IJIGSP)*, 6(8):40–46.

Marius, D., Pennathur, S., and Rose, K. (2000). Face detection using color thresholding, and eigenimage template matching. *Stanford University*.

Singh, S. K., Chauhan, D. S., Vatsa, M., and Singh, R. (2003). A robust skin color based face detection algorithm. *Tamkang Journal of Science and Engineering*, 6(4):227–234.

ZHAO, W., CHELLAPPA, R., P.J.Phillips, and Rosenfeld, A. (2003). Face recognition: A literature survey. *ACM Computing Surveys*, 35(4):339–458.

1. Problem description

1.1 Overview

According to the articel *Face recognition: A literature survey* from ZHAO et al. (2003), face recognition can be segmented into three key steps, shown in figure 1.

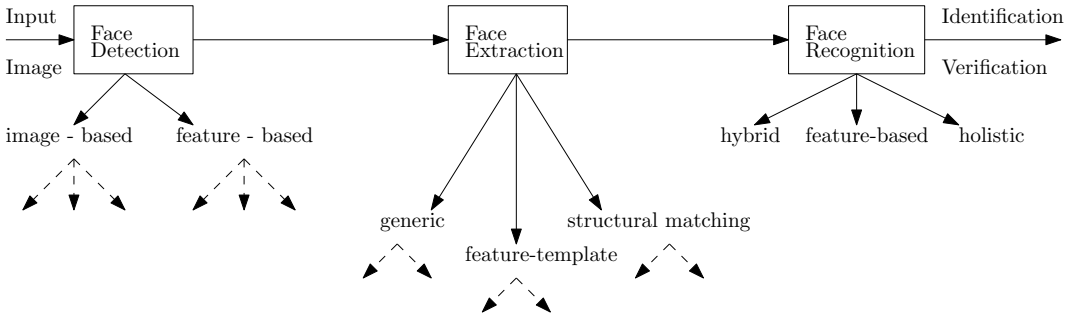


Figure 1: Face Recognition Progress

Face Detection is responsible for a rough normalization (like face tracking) and use for this task different approaches.
Face Extraction generates a more accurate normalization (like human emotions). The different approaches to get this emotions are shown in figure 1. Face detection and face extraction approaches can use the same feature-based-method (like informations out of color, Motion, ...) so they can perform simultaneous.
Face Recognition is the last step to identify/verify a picture. For a verification/identification several methods are available.

1.2 Face Detection

We decided to have a closer look on the face detection process because for the processes afterwards we need a detected face, which is not available without any effort.
To find an approach which we can study, implement and test we made further researches in this segment. The article *Face detection: A survey* from Hjelmas (2001) gives a good overview of the topic face detection. The figure 2 (out of Hjelmas (2001)) represents the different approaches to detect faces in a picture.

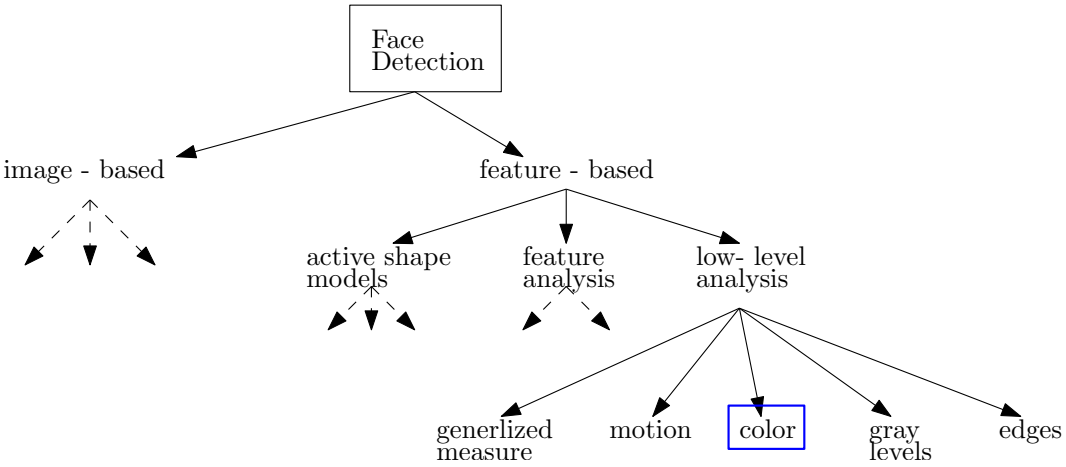


Figure 2: Face Detection divided into approaches (more detailed from Hjelmas (2001)).

According to Hjelmas (2001) are **Image-based approaches** the moste robust techniques for gray images, but on the other side they need a lot of computation time by multiresolution window scanning.
The **feature-based approaches** were the first attemps in the face detection history. They are built up simple and so they need less computation time, this enables these approaches access to real-time applications.

The most interesting approach for us was *Face detection based on color likelihood* approach (in figure 2 marked as *Color*). Argumentation for this algorithm can be found in the section 1.3;

1.3 Color based face detection

According to the article *A Robust Skin Color Based Face Detection Algorithm* of Singh et al. (2003) following points argue for the color based case detection

- Color processing is much faster than other facial features
- Color based algorithm is orientation invariant, that means that a motion estimation is much easier.
- Color based algorithm is often the first step for detection, this algorithm is popular.

An application for the simple and real-time capable algorithmus face detection with color can be found in the articel *Face recognition: A literature survey* from ZHAO et al. (2003)

> *In video conferencing systems, there is a need to automatically control the camera in such a way that the current speaker always has the focus. One simple approach to this is to guide the camera based on sound or simple cues such as motion and skin color.*

2. Literature Analysis

The literature analysis began with the topic selection (see chapter 1). The supervisor told us that the initial chosen topic *Face Detection* is to big to treat within one semester, so the first literature research was done to find a specific topic to handle.
The second literature research was done to find information about the chosen topic.

2.1 Approach

All interesting literature which were found and marked as interesting (by scanning the abstract) were saved in a list on the Ilias project space. This articles were read in a more detail afterwards.
The structure of the table (see figure 3) make additional sorting (by exporting/copying into an EXCEL) possible and the implementation on ILIAS makes it possible to get access easily to the actual table.

Literature Research (LR)

ID	Date	Topic	Source	site	comment
1	22.10.2016	general	olav: face recogintion database	Science direct - On internal representations in face recognition systems	analysis of face recognition systems; mentoined databases: FERET and Face database info MIT
2	27.10.2016	general	google: face recoginition overview	Face Recognition: A Literature Survey	nice overview about face recognition (split into Detection, extraction and Recognition) -> Useful to search a more detailed topic.
3	27.10.2016	Face detecion	olav: face detecion	ScienceDirect - Computer Vision and Image Understanding - Face Detection: A Survey	Good overview about different approaches to detect faces
4	27.10.2016	Face detecion - color	olav: face detecion	ScienceDirect - Pattern Recogintion - Face detection based on skin color likelihood	Face detection based on color likelihood - approach of: Face detection -> Feature-based approaces -> low level analysis -> color

Figure 3: Literature reasarech table - 30.10.2016.

All literatures which were mentioned in this document are also listed in the bibliography.

2.2 LR color based face detection

2.2.1 General Information about Face detection/Face recognition

The articles in subsections gives an overview of the topic face recognition and face detection. These articles were used to define the chosen topic: "Color based face detection".

- *Face Detection: A Survey* from Hjelmas (2001)
- *Face Recognition: A Literature Survey* from ZHAO et al. (2003)

2.2.2 Color based face detection

The article *A Robust Skin Color Based Face Detection Algorithm* from Singh et al. (2003) compares the three different color models (RGC, YCbCr and HSI). According to this article the color model YCbCr is widely used in the digital video domain and is more accurate (in detecting faces) than the other to color models. These two arguments are the reason why the color model YCbCr was chosen for this project.

A article which describes step by step how an approach of the color based face detection can be implemented can be found in *Face Detection Using Color Thresholding, and Eigenimage Template Matching* from Marius et al. (2000). In this article thresholds for the Cb and Cr are defined.

A different approach to morphological operations can be found in the article *Real Time Detection and Tracking of Human Face using Skin Color Segmentation and Region Properties* from Kumar and M (2014). This article defines also thresholds for the Cb and Cr.

3. Test scenario

3.1 Target

The target of the implementation is to use the video from a web cam to test the implemented color based face detection. The implemented code should as simple as possible, run the code on a Raspberry Pi.

3.2 Implementation steps

Following steps will be done to test if the implemented solution is real-time capable.

3.2.1 Color based face detection on a picture

The first step is to test the algorithms on different pictures (private pictures and pictures from WIKIMEDIA COMMONS). For this following steps are scheduled:

1. Transform picture into the YCbCr color space.
2. Find suitable threshold ranges for the YCbCr.
3. Make Thresholding on the YCbCr to get a binary picture.
4. Detecting faces out of skin regions.
5. Draw boxes to identify the faces on the picture.

3.2.2 Color based face detection on a video

Use a web cam and test the implemented color based algorithm.

3.2.3 Color based face detection on a Raspberry pi

In this project a Raspberry Pi Model B2 and the Raspberry Pi camera module V2 will be used.

Simulink Model Image

The first step is to transform the developed algorithm from MATLAB to Simulink. The *Image Processing Toolbox* provides boxes which covers some needed features. This boxes will be used to achieve a high performance (assumption is that the toolbox from Mathworks is as efficient as possible).

Simulink Model on host PC and Hardware from Raspberry Pi

In the next step a Simulink model should be created which can be deployed on the Raspberry Pi and check the results (original camera video frames and face detected video frames) with figures/Displays on the host PC.

Simulink Model on Raspberry Pi

The final step is to use the developed Simulink model to run it in a Standalone version on the Raspberry Pi. To see the result the modified video (including face detection) will be streamed over the internal HDMI port to a display or a beamer.

4. Implementation issue

4.1 Color based face detection on a picture

To load an image into the workspace and find suitable thresholds in the YCbCr space the informations from the lecture *Color* in the course *Image and Signal Processing* (Amman (2016)). A detailed description to this procedure can be found in the attachment A.2

The thresholding itself can be done by logical operations like in the listening 4.1.

Listing 4.1: Color thresholding

```
% Thresholding -> binary
thresh_cb = cb > 105 & cb < 120;     % thresholding for cb values
thresh_cr = cr > 140 & cr < 165;     % thresholding for cr values
binary_pic = thresh_cb&thresh_cr;    % create binary picture
```

To detect faces out of the binary image (to reject areas/blobs which are no faces) a few process steps are necessary. The process is described in the article Kumar and M (2014). To verify that the found skin region represents a face, it must satisfy every of the following conditions:

Small Area Skin regions which have less than a specified number of pixels are rejected. This number highly depends on the resolution of the image and has to be defined individually for each resolution.

Euler Number Human faces contain some holes like eyes, eyebrows, a mouth etc. If a skin region does not contain any holes, this region is discarded. This is done using the Euler Number: $E = C–H$. Where C is the number of connected components and H is the number of holes in a region. If the Euler Number is greater than zero, the region is discarded.

Eccentricity The oval shape of a face can be approximated by an ellipse. If a skin region has an eccentricity greater than 0.91, it gets discarded. An ellipse whose eccentricity is 0 is a circle, while an ellipse whose eccentricity is 1 represents a line segment.

Bounding Box Properties If height to width ratio of a skin region is greater than $\frac{1}{2}$, the skin region is discarded.

The used values for the eccentricity and the bounding box condition are obtained by trial and error. For the implementation of these conditions ready-made MATLAB-Functions are available. The code can be seen in listing 4.2.

Listing 4.2: Rejection of non Face Skin Region

```
%label all the connected components in the image
bw=bwlabel(close_binary_pic,8);

%image blob analysis - we get a set of properties for each labeled region
area=regionprops(bw,'Area')
eulernumber=regionprops(bw,'EulerNumber');
eccentricity=regionprops(bw,'Eccentricity');
centroid=regionprops(bw,'Centroid');
boundingbox=regionprops(bw,'BoundingBox');
```

After applying the above mentioned conditions to the binary image, rectangles are drawn around the remaining skin regions. This can be seen in figure 4.

Computation time The computation time of the face detection algorithm highly depends on the size of the image and the detected skin area. The image shown in figure 4 for example has a size of 1548x2048 pixels and needs a computation time of 1.241 seconds using the MATLAB stopwatch timer. Considering the fact that the MATLAB plot functions need a lot of computation time, the measurement was repeated without plotting. The algorithm then needs 0.923 seconds computation time.

The whole script can be found in the attachment B.2.

4.2 Color based face detection on a video

After implementing face detection on images, the algorithm is now enhanced to detect faces in a webcam video stream.

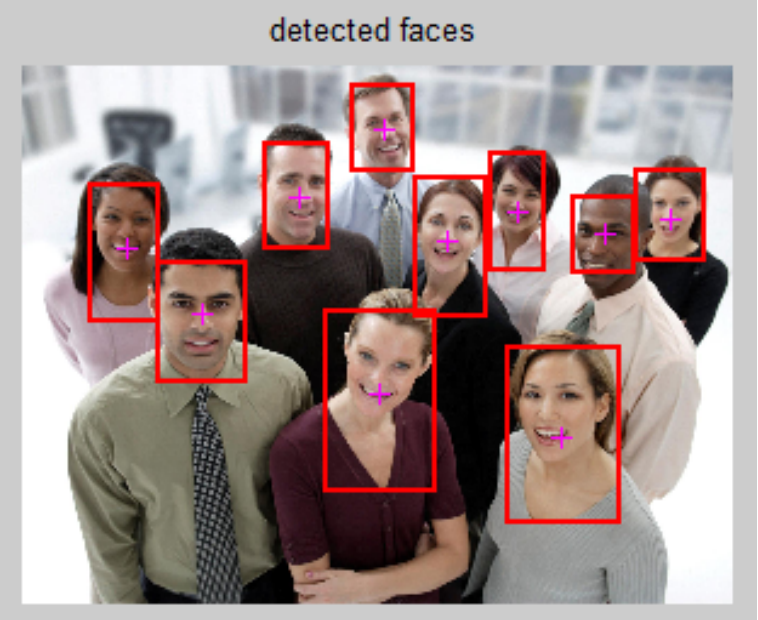


Figure 4: Detected faces

The main idea is to take snapshots of the incoming video stream to let them run through the former explained algorithm for face detection in images. Every time the acquired image is fully processed, a new snapshot is taken.

Knowing that the computation time of the algorithm highly depends on the size of the image, the resolution is set to 320x240 pixel. Thus it can be guaranteed that the algorithm is fast enough for real time applications.

Calculation time The calculation time for processing a snapshot and plotting the image with rectangles around faces lies in between 0.04 and 0.09 seconds. Taking the snapshot takes another 0.08 - 0.1 seconds. Corresponding to this values, the algorithm has a worst-case computation time of 0.19 seconds and a best-case computation time of 0.12 seconds for each snapshot. In other words the frames per second rate lies in between 5.26 and 8.33 (see table 1).

ID	Best case	Worst case
Computation time for on snapshot	0.12 s	0.19 s
Frames per second	8.33	5.26

Table 1: Performance comparison

The whole script for the face detection on a video using MATLAB can

be found in appendix B.3.

4.3 Color based face detection on a Raspberry Pi

4.3.1 Simulink Model - Face detection on an image

The *Image Processing Toolbox* from Simulink provides the function to load an image from a file, convert this image (from RGB) into the YCbCr space, make the blob analysis, draw the rectangular (which represents a detected face) on the image and display the image on a figure.

To threshold the YCbCr image a MATLAB-function block (*color thresholding* in figure 5) was inserted. The MATLAB-function block *remove misshapen skin regions* is used to remove rectangles which are wider than height (with this functions skin regions from vertical hands can be rejected). This function block was necessary because the Blob Analysis block didn't support all necessary functions (for example the eulernumber is missing).

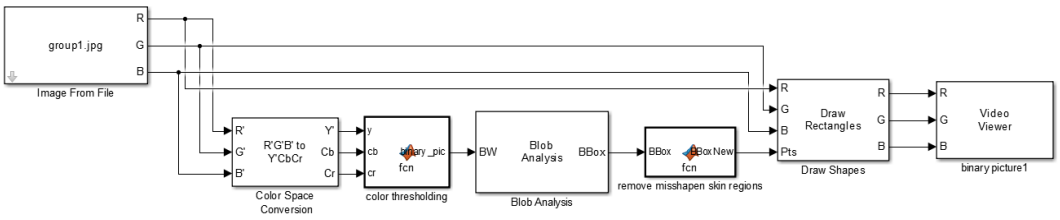


Figure 5: Simulink model - Face detection on an image

The code of the Matlab function blocks can be found in the attachment C.1.

4.3.2 Simulink Model deployed on Raspberry Pi with I/O connection to the host PC

To run a Simulink model on a Raspberry Pi the hardware support package Run models on Raspberry Pi is necessary. The package installation hints and Examples can be found on Raspberry Pi Support from Simulink.

Hints to run a Simulink block on the Raspberry Pi:

- Use a standalone licence or a network licence without VPN. By using a licence over a VPN it is not possible to communicate with the Raspberry Pi.

- Before using the Raspberry Pi camera module the steps from the instructions Use Camera Board with V4L2 Video Capture Block must be done. Otherwise the Simulink block will not find the camera module.

In a first step the setup looks like figure 6. The idea is to get familiar with the Raspberry Pi camera module (the resolution, brightness, ...) and use the data for adjustments (especially for find suitable thresholds). In this case the model should run in the external mode, the instructions Run Model in External Mode provides all necessary informations.

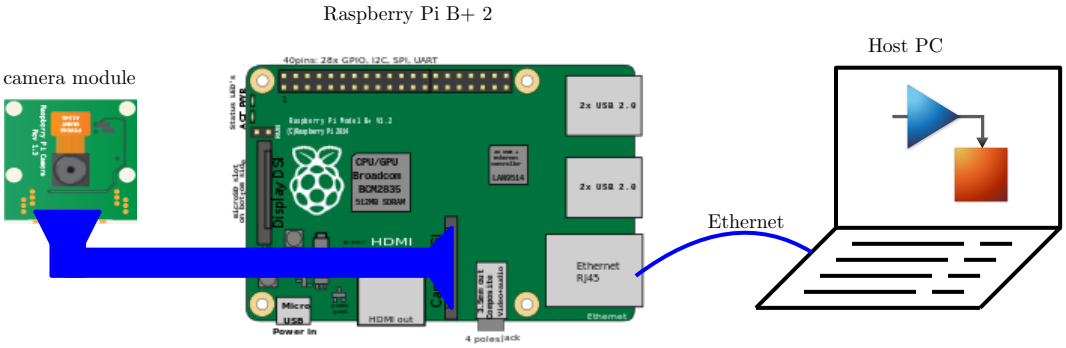


Figure 6: Simulink model setup - With I/O connection to the host

The Simulink model itself was built up similar with the model in figure 5, see figure 7.

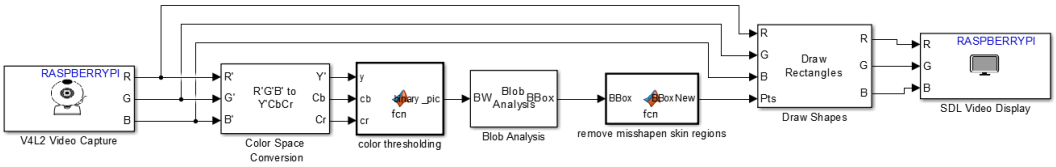


Figure 7: Simulink model - With I/O connection to the host

4.3.3 Simulink Model as Standalone Application

The final step is to deploy the on the hardware like in figure 8. As model the model of figure 7. To deploy the model on the hardware the instruction Run Model as Standalone Application is useful.

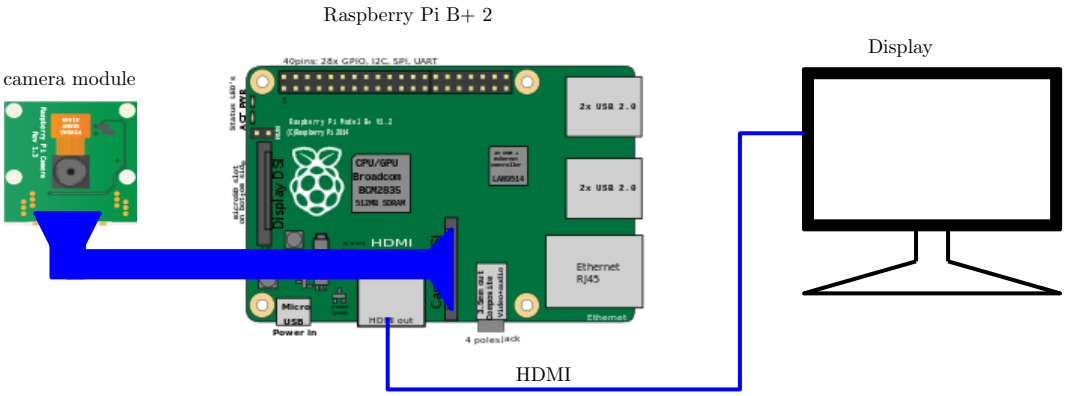


Figure 8: Simulink model setup - Standalone application

5. Results

5.1 Findings

- Resolution of the image/video must be defined, because the resolution has an influence to the amount of blob pixels (area property) and the computation time.
- Thresholding values are various for different camera types. For every camera the thresholds must be found (Webcam has other thresholds than the raspberry pi camera module).
- Lighting conditions can affect the algorithm – e.g. shadows (also depending on the used camera).
- Hands and other skin regions are often interpreted as faces, without any additional filtering (like concerning the ratio of the rectangular).
- Adjacent faces melt to one big rectangle.
- Simulink blocks don't support all region properties (e.g. no euler number)in contrast to the Matlab algorithms. For some applications/ situations additional morphological operations are necessary.

5.2 Improvements

- To increase the speed the Raspberry Pi 3, or other powerful embedded hardware, instead of the Raspberry Pi 2 can be used.
- To increase the speed, the algorithm should be programmed in PYTHON.
- To get rid of the resolution problem (different amount of blob analysis), a function which calculates the resolution depending parameters (e.g. Area) can be implemented.

A. Appendix

If not otherwise noted pictures are taken from *Wikicommons*. http://commons.wikimedia.org

A.1 YCbCr-Color-Space

A.1.1 Color Spaces

A color space is a mathematical model to represent color information out of a picture. There were several color models available:

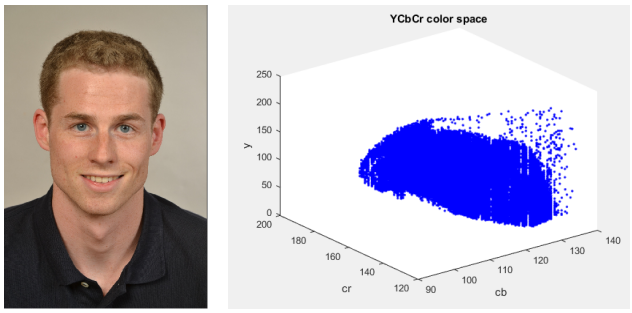
- RGB based color space (RGB, normalized RGB)
- Hue based color space (HSI, HSV and HSL)
- Luminance based color space (YCbCr, YIQ and YUV)
- Perceptually uniform color space (CIEXYZ, CIELAB, and CIELUV)

Luminance based color space has the split the image into intensity (luminance) informations and color (chrominance) informations. The YCbCr space was chosen because several articles recommended this color space for video applications (because of its speed). The letters of YCbCr represents:

- Y: luminance
- Cb: blue-yellow chrominance
- Cr: red-green chrominance

A.1.2 YCbCr skin color

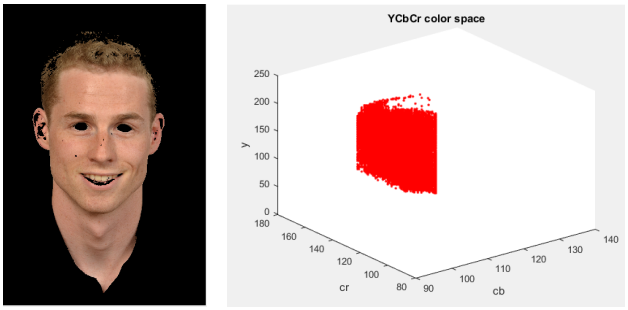
A color picture leads to an YCbCr color space like in figure 9.



(a) Original (b) YCbCr color space

Figure 9: YCbCr - without Thresholding - MEM3 student

By applying the determined thresholds(see section A.2) the skin pixels can be seen clustered in a region of the YCBCR space (see figure 10).

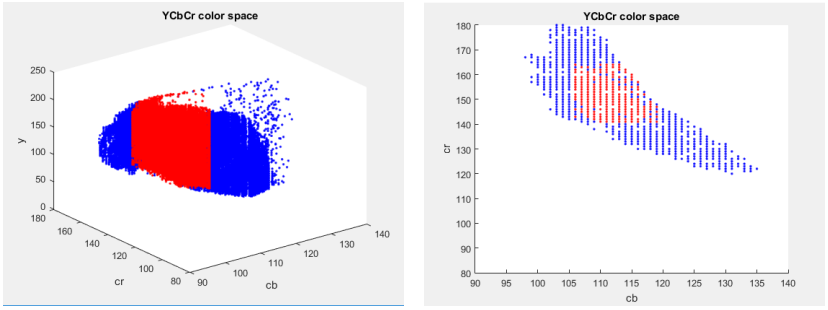


(a) Thresholded picture (b) YCbCr color space

Figure 10: YCbCr - with Thresholding - MEM3 student

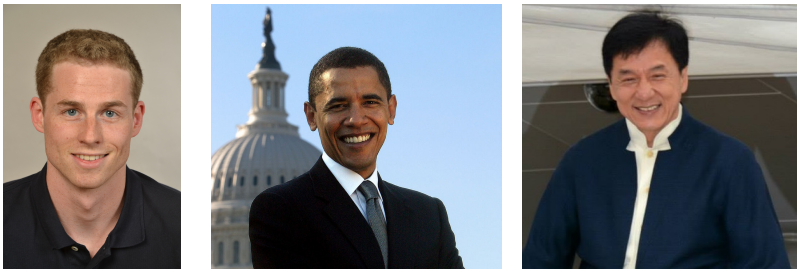
A comparison between the whole coloured picture and the threshold picture (see figure 11) shows that the relevant information of the skin color is in the chrominance (Cb and Cr) and independent of the luminance (Y).

Interesting is that independent of the skin type (white, black or yellow) the relevant skin pixels are always at the same region. To show this the pictures of figure 12 were thresholded and compared see figure yy.



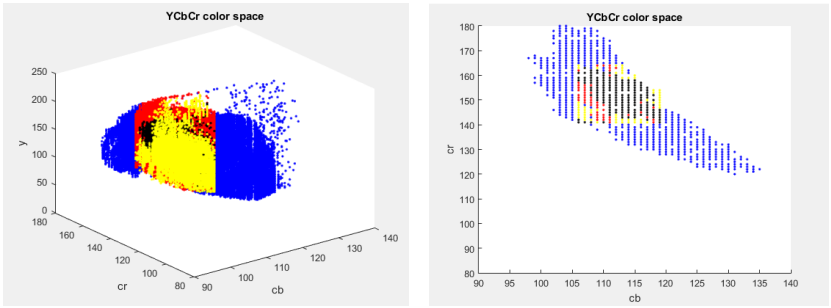
(a) Thresholded picture (b) YCbCr color space

Figure 11: YCbCr - comparison between the full colord picture (blue dots) and the thresholded picture (red dots) - MEM3 student



(a) MEM3 student (b) Barack Obama (c) Jackie Chan

Figure 12: Test pictures



(a) Thresholded picture (b) YCbCr color space

Figure 13: YCbCr - comparison between the full colord picture of MEM3 student (blue dots), thresholded MEM3 student (red dots), thresholded Jackie Chan (yellow dots) and thresholded Barack Obama (black dots).

A.2 Color threshold

A.2.1 Procedure

According to different literature (Real Time Detection and Tracking of Human Face using Skin Color Segmentation and Region Properties Kumar and M (2014), Face Detection Using Color Thresholding, and Eigenimage Template Matching Kumar and M (2014) and A Robust Skin Color Based Face Detection Algorithm Singh et al. (2003)) the thresholds must be found with an try and error procedure.

The three mentioned articles have chosen different thresholds, in this project the thresholds were chosen with the MATLAB application colorThresholder (from the Image Processing Toolbox).

To run this application the command *colorThresholder* must be entered into the command window of MATLAB.

After loading an image and choosing the color space (in this case the YCbCr space - see figure 14) the thresholds can be set (see figure 17).

The next step is to use the find values for Cb and Cr:

- Cb: $105 > \text{Cb} < 120$
- Cr: $140 > \text{Cr} < 165$

to threshold the image (values above the over limit and pixel values under the lower value will be set to black, all pixels within the limit will set to white). This can be done with the matlab commands:

```
% Thresholding -> binary
thresh_cb = cb > 105 & cb < 120;     % thresholding for cb values
thresh_cr = cr > 140 & cr < 165;     % thresholding for cr values
binary_pic = thresh_cb&thresh_cr;    % create binary picture
```

The result looks like figure 16.

The next steps is to modify the binary picture (by removing the small black pixels in the face and the small white pixels out of the face) to make face detection more efficient.

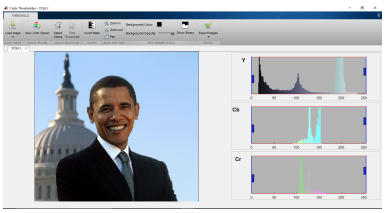


Figure 14: colorThresholder - Barrack Obama - without thresholding

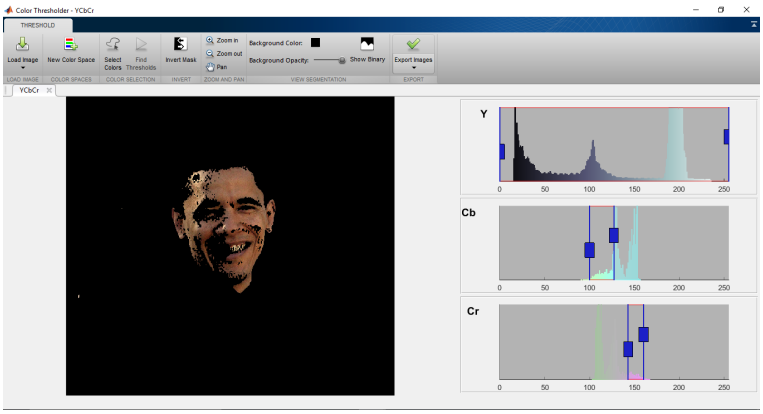


Figure 15: colorThresholder - Barrack Obama - with thresholds : $105 > cb < 120$ and $140 > cr < 165$

Figure 16: colorThresholder - Barrack Obama - with thresholds - Binary

A.2.2 MATLAB implementation

Listing A.1: Color thresholding

```
I=imread('../pictures/obama.jpg');  % load image from workspace

% RGB -> YCbCr
YCBCR = rgb2ycbcr(I);               % transform image into YCbCr space
y = YCBCR(:,:,1);                   % extract Y value out of matrix
cb = YCBCR(:,:,2);                  % extract Cb value out of matrix
cr = YCBCR(:,:,3);                  % extract Cr value out of matrix

% Thresholding -> binary
thresh_cb = cb > 105 & cb < 120;    % thresholding for cb values
thresh_cr = cr > 140 & cr < 165;    % thresholding for cr values
binary_pic = thresh_cb&thresh_cr;   % create binary picture

% show binary picture:
figure
imshow(binary_pic);
```

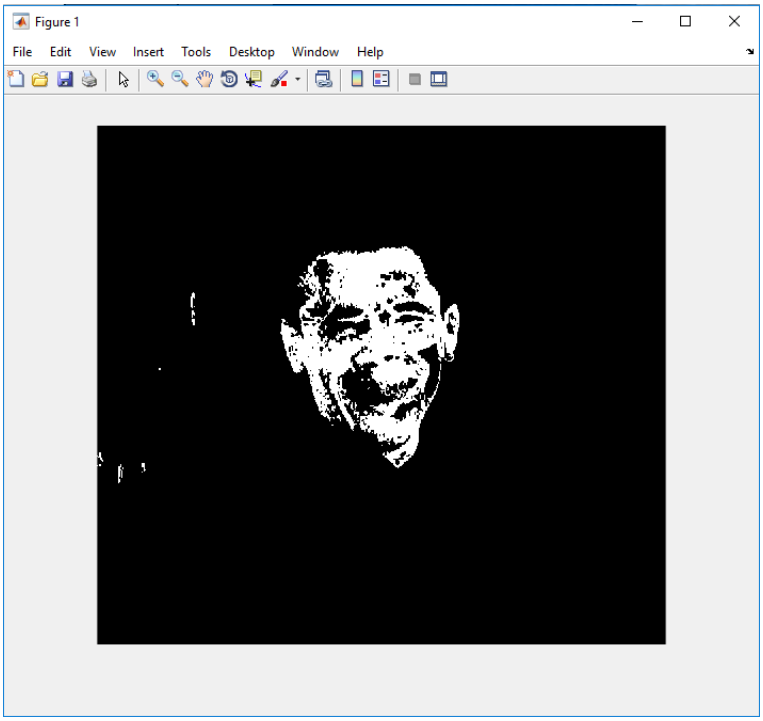


Figure 17: Binary Picture - Barrack Obama - figure out of matlab code A.1

A.2.3 Examples

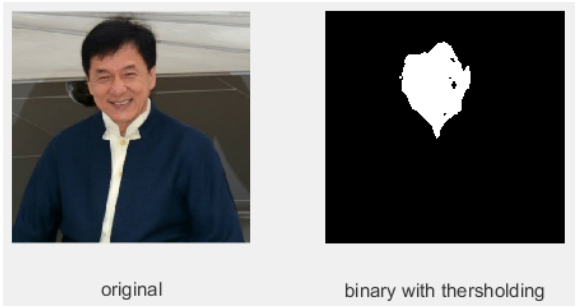


Figure 18: Chackie Jan

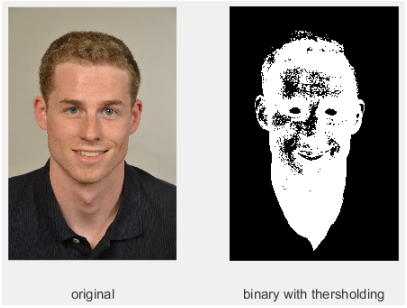


Figure 19: MEM student (private picture)

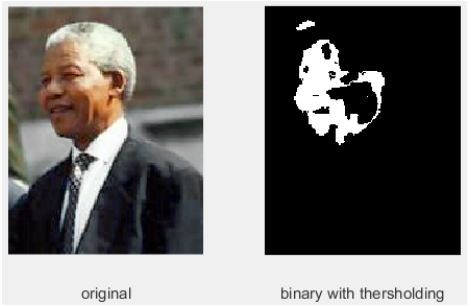


Figure 20: Nelson Mandela

B. MATLAB scripts

B.1 Show pictures in YCbCr space

B.1.1 Main script

```
% University of Applied Science Vorarlberg
% Master of Mechatronics
% ------------------------------------------------------------------------
% Course:       Sensor Systems
% ------------------------------------------------------------------------
% Author:       Tobias Burtscher and Stefan Stark
% Date:         07.12.2016
% Description:  Script which shows picture (original and thresholded)
%               informaion in YCbCr space.

%%
clear all, close all, clc;          % clean up

%%
tLI = tic;                          % start a stopwatch timer
I = imread('pictures/me1_mod.jpg'); % load image
tLoadImage = toc(tLI)               % stop timer to see how much time is
                                    % necessary to load an image

tTI = tic;                          % start a stopwatch timer
YCBCR = rgb2ycbcr(I);               % transfare image into the ycbcr space
y = YCBCR(:,:,1);                   % seperate variable
cb = YCBCR(:,:,2);                  % seperate variable
cr = YCBCR(:,:,3);                  % seperate variable
tTransformImage = toc(tTI)          % stop timer

% Schow image in YCBCR color space
figure
plot3(cb,cr,y,'b.')
hold on;                            % to draw the thresholded values into
```

```
                                    % the same figure

%% draw into the same figure the thresholds of different pictures
% load image from disk and save rgb values as matrix
image1 = imread('pictures/me1_mod.jpg');
image2 = imread('pictures/JackieChan.jpg');
image3 = imread('pictures/obama.jpg');

% define colors
colo = ['r' 'y' 'k'];

% define thresholds
cb_low = 105; cb_high = 120;
cr_low = 140; cr_high = 165;

% run function which
%   calculates cb and cr values,
%   make thresholding and
%   create YCbCr plot;
createYCbCrPlot3(rgb2ycbcr(image1),cb_low,cb_high,cr_low,cr_high,colo(1))
createYCbCrPlot3(rgb2ycbcr(image2),cb_low,cb_high,cr_low,cr_high,colo(3))
createYCbCrPlot3(rgb2ycbcr(image3),cb_low,cb_high,cr_low,cr_high,colo(2))


hold off;
xlim([90 140]);ylim([80 180]);          % scale axis for better view
xlabel('cb');ylabel('cr');zlabel('y');  % label axis
title('YCbCr color space');             % create title of figure
```

B.1.2 functions

```
% University of Applied Science Vorarlberg
% Master of Mechatronics
% ------------------------------------------------------------------------
% Course:       Sensor Systems
% ------------------------------------------------------------------------
% Author:       Tobias Burtscher and Stefan Stark
% Date:         07.12.2016
% Description:  Function which seperate Y, Cb and Cr values out of
%               transformed picture; threshold the picture and show
%               thresholded picture pixels in an existing figure

function createYCbCrPlot3( YCBCR,cb_low,cb_high,cr_low,cr_high,colo)
    y = YCBCR(:,:,1);                   % seperate variable
    cb = YCBCR(:,:,2);                  % seperate variable
    cr = YCBCR(:,:,3);                  % seperate variable
    % Thresholding the image
```

```
    thresh_cb = cb > cb_low & cb < cb_high;
    thresh_cr = cr > cr_low & cr < cr_high;

    %define color
    colo = sprintf('%s.',colo);
    % plot thresholded picture into YCbCr figure
    plot3(cb.*(uint8(thresh_cb)),cr.*(uint8(thresh_cr)),y,colo);

end
```

B.2 Color based face detection on a picture

```
% University of Applied Sciences Vorarlberg
% Master of Mechatronics
% ------------------------------------------------------------------------
% Course:       Sensor Systems
% ------------------------------------------------------------------------
% Author:       Tobias Burtscher and Stefan Stark
% Date:         18.12.2016
% Description:  Face Detection in images.

%%
clear all, close all, clc;     % Clean up
I = imread('group1.jpg');      % Load image

YCBCR = rgb2ycbcr(I);         % Transfer image into YCbCr color space
y = YCBCR(:,:,1);             % Extract y,cb and cr information
cb = YCBCR(:,:,2);
cr = YCBCR(:,:,3);

% Threshold cb and cr values to get a binary image
thresh_cb = cb > 76 & cb < 125;
thresh_cr = cr > 140 & cr < 165;
binary_pic = thresh_cb&thresh_cr;

% Label all the connected components (blobs) in the image
bw=bwlabel(binary_pic,8);

% Image blob analysis - we get a set of properties for each labeled region
area=regionprops(bw,'Area');
eulernumber=regionprops(bw,'EulerNumber');
eccentricity=regionprops(bw,'Eccentricity');
centroid=regionprops(bw,'Centroid');
boundingbox=regionprops(bw,'BoundingBox');

subplot(2,2,3); imshow(I);  % Show image
```

```
title('detected faces');    % Create title of figure
nr_faces = 0;               % Initialize variable for counting faces
hold on                     % Hold image to add rectangle

% Loop every blob
for i=1:length(area)

    % Decide if skin area is a face
    if eccentricity(i).Eccentricity<0.91 && area(i).Area>2000 ...
        && eulernumber(i).EulerNumber<=0

        bb=boundingbox(i).BoundingBox;  % save bounding box values in bb
        width = bb(3);                  % width of bounding box
        height = bb(4);                 % height of bounding box
        ratio = width/height;           % height to width ratio of box

        % Discard blob if height to width ratio is greater than 1.2
        if ratio<1.2

        bc=centroid(i).Centroid;    % Coordinates for rectangle center
        rectangle('Position',bb,'EdgeColor','r','LineWidth',2)  % Draw rectangle
        plot(bc(1),bc(2),'-m+')     % Draw rectangle center
        nr_faces = nr_faces+1;      % Increment counter
        end
    end
end

fprintf('\n  Detected number of faces: %i\n',nr_faces); % print number of faces

subplot(2,2,1);
imshow(I)                           % Show the original image
title('original')                   % Create title of figure

subplot(2,2,2);
imshow(binary_pic)                  % Show the thresholded binary image
title('binary with thresholding')   % Create title of figure
```

B.3 Color based face detection on a video

```
% University of Applied Sciences Vorarlberg
% Master of Mechatronics
% ------------------------------------------------------------------------
% Course:       Sensor Systems
% ------------------------------------------------------------------------
% Author:       Tobias Burtscher and Stefan Stark
% Date:         18.12.2016
```

```
% Description:  Face Detection in a webcam video stream.

%%
clear all, close all, clc;              % clean up

% Get the webcam-video input in the desired resolution
vid = videoinput('winvideo', 1,'YUY2_320x240');

set(vid,'FramesPerTrigger',Inf);        % Keep acquiring frames until end condition
set(vid,'ReturnedColorSpace','YCbCr'); % Specify returned color space
vid.FrameGrabInterval = 5;              % Acquire only every 5th frame from video
                                        % stream

start(vid);                             % Start video stream

while(vid.FramesAcquired<=100)          % Stop loop after 100 acquired frames
    tic
    data=getsnapshot(vid);              % Take a snapshot of video stream
    toc

    % Extract y,cb and cr information
    y = data(:,:,1);
    cb = data(:,:,2);
    cr = data(:,:,3);

    % Threshold cb and cr values to get a binary image
    thresh_cb = cb > 76 & cb < 125;
    thresh_cr = cr > 135 & cr < 165;
    binary_pic = thresh_cb&thresh_cr;

    % Label all the connected components (blobs) in the image
    bw=bwlabel(binary_pic,8);

    % Image blob analysis - we get a set of properties for each labeled region
    area=regionprops(bw,'Area');
    eulernumber=regionprops(bw,'EulerNumber');
    eccentricity=regionprops(bw,'Eccentricity');
    centroid=regionprops(bw,'Centroid');
    boundingbox=regionprops(bw,'BoundingBox');

    combine_layers=cat(3,y,cb,cr);          % Combine the y,cb and cr layer
    color_image=ycbcr2rgb(combine_layers);  % Convert snapshot to rgb space
    imshow(color_image);                    % Show snapshot
    hold on                                 % Hold image to add rectangle

    % Loop every blob
    for i=1:length(area)

        % Decide if skin area is a face
```

```
        if eccentricity(i).Eccentricity<0.91 && eulernumber(i).EulerNumber<=0 ...
            && area(i).Area>2000

            bb=boundingbox(i).BoundingBox;  % Save bounding box values in bb
            width = bb(3);                  % Width of bounding box
            height = bb(4);                 % Height of bounding box
            ratio = width/height;           % Height to width ratio of box

            % Discard blob if height to width ratio is greater than 1.2
            if ratio<1.2

                % Draw rectangle around face
                rectangle('Position',bb,'EdgeColor','r','LineWidth',2)
            end
        end
    end
    hold off
end

stop(vid);    % Stop video stream after end codition
```

C. Simulink Models

C.1 Simulink model to detect a face on an image

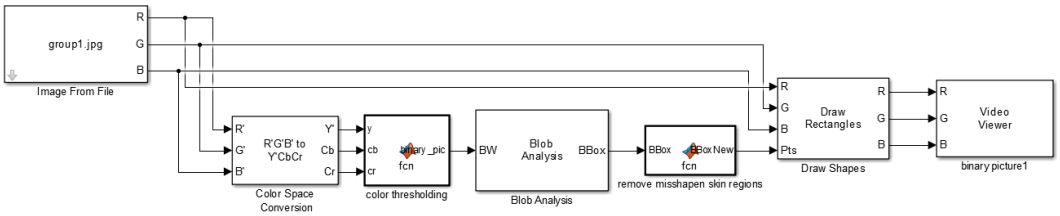


Figure 21: Simulink model - Face detection on an image

Listing C.1: MATLAB function block - color thresholding

```
% University of Applied Science Vorarlberg
% Master of Mechatronics
% ------------------------------------------------------------------------
% Course: Sensor Systems
% ------------------------------------------------------------------------
% Author:       Tobias Burtscher and Stefan Stark
% Date:         10.12.2016
% Description:  Function which thresholds the YCbCr picture for face
%               detection

function binary_pic  = fcn(y,cb,cr)
    % Thresholding -> binary
    thresh_cb = cb > 76 & cb < 125;     % thresholding for cb values
    thresh_cr = cr > 140 & cr < 165;    % thresholding for cr values
    binary_pic = thresh_cb&thresh_cr;   % create binary picture

    y_th = y;                           % set output
    cb_th = cb.*(uint8(thresh_cb));     % set output cb
```

```
    cr_th = cr.*(uint8(thresh_cr));     % set output cr
                                        % typecast is necessary to create
                                        % a number out of the boolean
                                        % thresh_cb
end
```

Listing C.2: MATLAB function block - remove misshapen skin regions

```
% University of Applied Science Vorarlberg
% Master of Mechatronics
% ------------------------------------------------------------------------
% Course: Sensor Systems
% ------------------------------------------------------------------------
% Author:       Tobias Burtscher and Stefan Stark
% Date:         10.12.2016
% Description:  Function to reject boxes which are wider than high.
%               Remove misshapen boxes.

function BBoxNew = fcn(BBox)
    [row col] = size(BBox);     % extract amount of rows from BBox
    count = int32(1);           % initialize a count varible

    for i=1:row
        width = BBox(i,3);      % extract width out of BBox, for each blob
        height = BBox(i,4);     % extract height out of BBox, for each blob
        ratio = width/height;   % calculate ratio

        if ratio > 1.5          % if box is wider than height
            BBox(i,:) = 0;      % set entrie to zero
        else                    % ratio is good -> possible face
            BBox(count,:) = BBox(i,:);
            count = count+1;    % count is necessary to be sure that the
                                % first entries of BBox are boxes with the
                                % right ratio.
        end
    end
    BBoxNew = BBox(1:10,:);     % print the first 10 boxes
end
```

C.2 Simulink model to run on Raspberry Pi

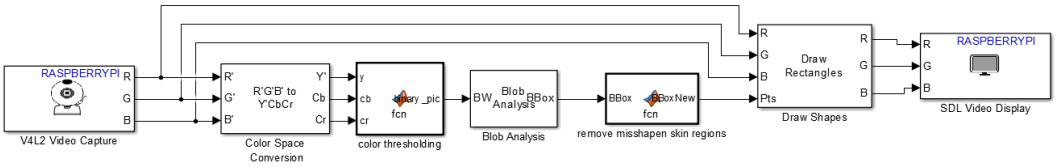


Figure 22: Simulink model - Run on Raspberry Pi