
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, 18.12.2016

Table of Contents

List of Figures	4
Bibliography	5
1 Problem description	5
1.1 Overview	5
1.2 Face Detection	5
1.3 Color based face detection	6
2 Literature Analysis	8
2.1 Approach	8
2.2 LR color based face detection	9
2.2.1 General Information about Face detection/Face recog- nition	9
2.2.2 Color based face detection	9
3 Test scenario	10
3.1 Target	10
3.2 Implementation steps	10
3.2.1 Color based face detection on a picture	10
3.2.2 Color based face detection on a video	10
3.2.3 Color based face detection on a Raspberry pi	11
4 Implementation issue	12
4.1 Color based face detection on a picture	12
4.2 Color based face detection on a video	13
4.3 Color based face detection on a Raspberry Pi	13
4.3.1 Simulink Model - Face detection on an image	13
4.3.2 Simulink Model deployed on Raspberry Pi with I/O connection to the host PC	13
4.3.3 Simulink Model as Standalone Application	14

5	Results	16
5.1	Findings	16
5.2	Improvements	16
A	Appendix	17
A.1	YCbCr-Color-Space	17
A.2	Color threshold	20
B	MATLAB scripts	24
B.1	Show pictures in YCbCr space	24
C	Simulink Models	27
C.1	Simulink model to detect a face on an image	27
C.2	Simulink model to run on Raspberry Pi	29

List of Figures

1	Face Recognition Progress	5
2	Face Detection divided into approaches (more detailed from ?).	6
3	Literature reasarech table - 30.10.2016.	8
4	Simulink model - Face detection on an image	13
5	Simulink model setup - With I/O connection to the host . . .	14
6	Simulink model - With I/O connection to the host	14
7	Simulink model setup - Standalone application	15
8	YCbCr - without Thresholding - MEM3 student	18
9	YCbCr - with Thresholding - MEM3 student	18
10	YCbCr - comparison - full colored and thresholded	19
11	Test pictures	19
12	YCbCr - comparison - MEM3 student, Chan and Obama . . .	19
13	colorThresholder - Barrack Obama - without thresholding . .	21
14	colorThresholder - Barrack Obama - with thresholds	21
15	colorThresholder - Barrack Obama - with thresholds - Binary	21
16	Binary Picture - Barrack Obama	22
17	Chackie Jan	23
18	MEM student (private picture)	23
19	Nelson Mandela	23
20	Simulink model - Face detection on an image	27
21	Simulink model - Run on Raspberry Pi	29

1. Problem description

1.1 Overview

According to the article *Face recognition: A literature survey* from [?], face recognition can be segmented into three key steps, shown in figure 1.

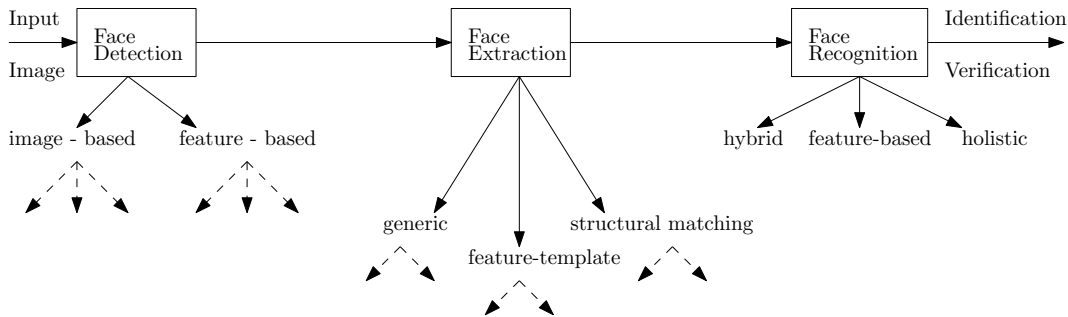


Figure 1: Face Recognition Progress

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 [?] gives a good overview of the topic face detection. The figure 2 (out of [?]) represents the different approaches to detect faces in a picture.

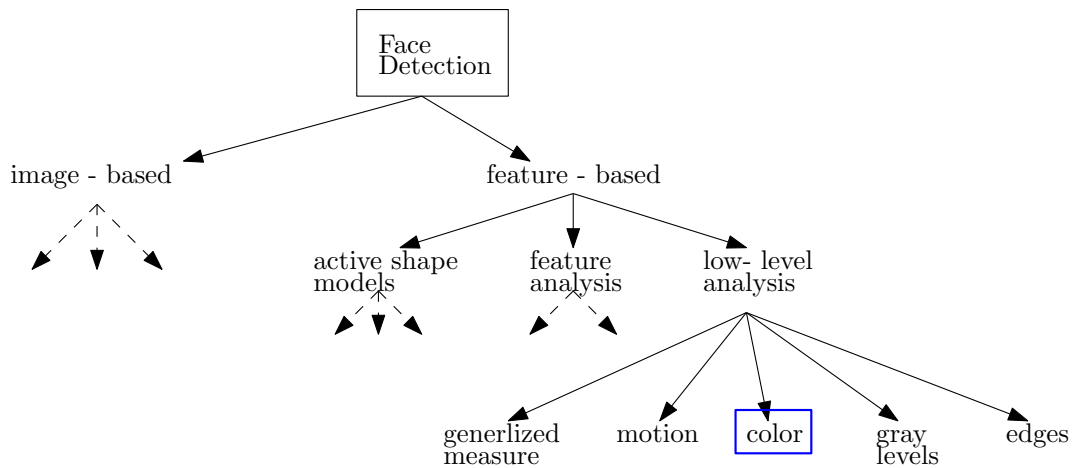


Figure 2: Face Detection divided into approaches (more detailed from [?]).

According to [?] are **Image-based approaches** the most 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 attempts 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 [?] 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 artichel *Face recognition: A literature survey* from ?

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 too 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	comment
1	22.10.2016	general	olav: face recognition database	ScienceDirect - On internal representations in face recognition systems analysis of face recognition systems; mentioned databases: FERET and Face database info MIT
2	27.10.2016	general	google: face recognition 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 detection	olav: face detection	ScienceDirect - Computer Vision and Image Understanding - Face Detection: A Survey Good overview about different approaches to detect faces
4	27.10.2016	Face detection - color	olav: face detection	ScienceDirect - Pattern Recognition - Face detection based on skin color likelihood Face detection based on color likelihood - approach of: Face detection -> Feature-based approaches -> low level analysis -> color

Figure 3: Literature research 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 ?
- *Face Recognition: A Literature Survey* from ?

2.2.2 Color based face detection

The article *A Robust Skin Color Based Face Detection Algorithm* from ? 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 the article *Face Detection Using Color Thresholding, and Eigenimage Template Matching* from ?. In this article thresholds for the C'b 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 ?. 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 (?). 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

```
1 % Thresholding -> binary
2 thresh.cb = cb > 105 & cb < 120;    % thresholding for cb values
3 thresh.cr = cr > 140 & cr < 165;    % thresholding for cr values
4 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 ?, for this steps MATLAB-Functions are available which looks like listening 4.2.

Listing 4.2: Rejection of non Face Skin Region

```
1 %label all the connected components in the image
2 bw=bwlabel(close_binary_pic,8);
3
4 %image blob analysis - we get a set of properties for each labeled region
5 area=regionprops(bw, 'Area');
6 eulernumber=regionprops(bw, 'EulerNumber');
7 eccentricity=regionprops(bw, 'Eccentricity');
8 centroid=regionprops(bw, 'Centroid');
9 boundingbox=regionprops(bw, 'BoundingBox');
```

The whole script can be found in the attachment xx.

4.2 Color based face detection on a video

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 imager (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 4) 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 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).

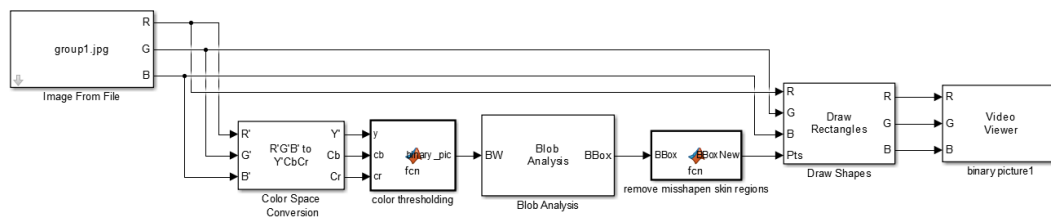


Figure 4: Simulink model - Face detection on an image

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 7. 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.

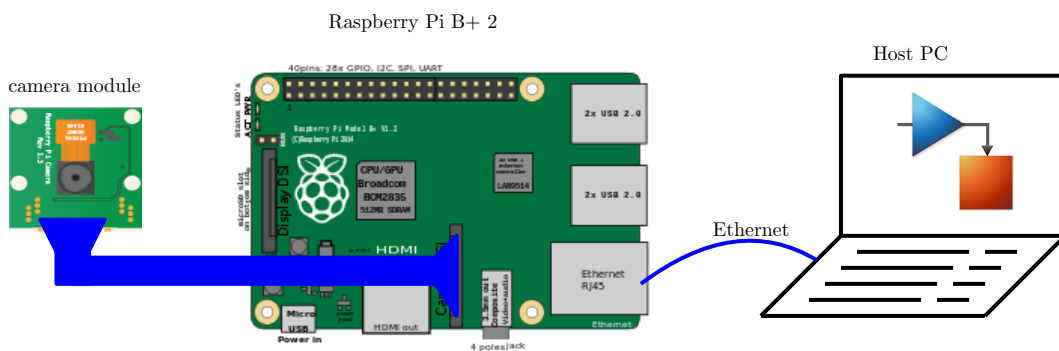


Figure 5: Simulink model setup - With I/O connection to the host

The Simulink model itself was built up similar with the model in figure 4, see figure ??.

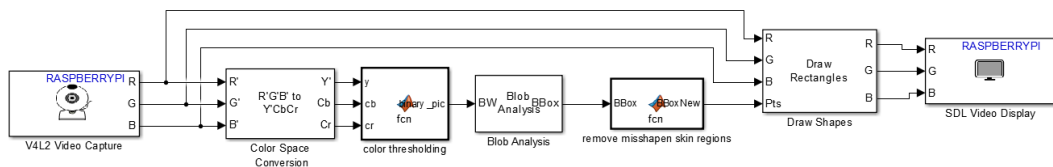


Figure 6: Simulink model - With I/O connection to the host

4.3.3 Simulink Model as Standalone Application

The final step is to deploy the on the hardware. As model the model of figure ?? is used. To deploy the model on the hardware the instruction Run Model as Standalone Application is useful.

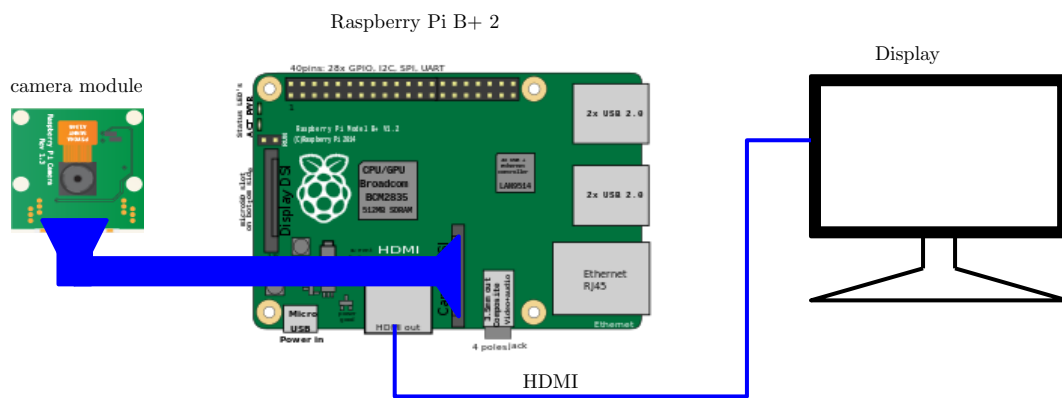


Figure 7: Simulink model setup - Standalone application

5. Results

5.1 Findings

- The

5.2 Improvements

- The

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 8.

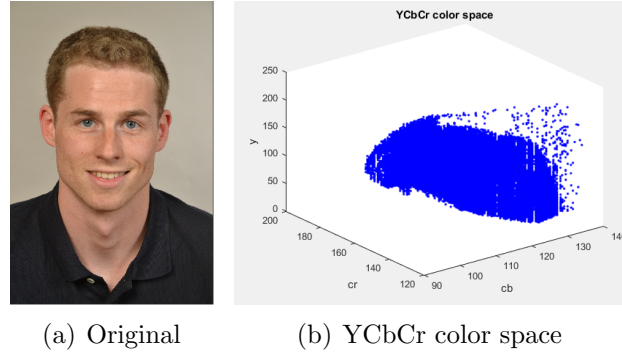


Figure 8: YCbCr - without Thresholding - MEM3 student

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 9).

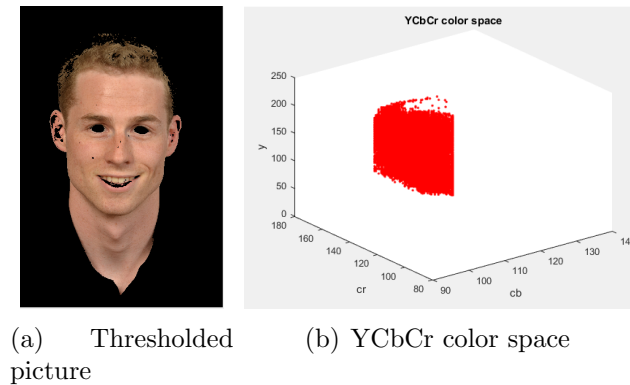


Figure 9: YCbCr - with Thresholding - MEM3 student

A comparison between the whole coloured picture and the threshold picture (see figure 10) 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 11 were thresholded and compared see figure yy.

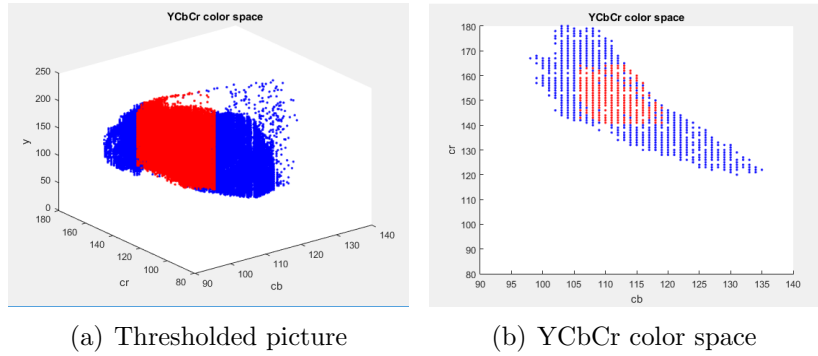


Figure 10: YCbCr - comparison between the full colored picture (blue dots) and the thresholded picture (red dots) - MEM3 student

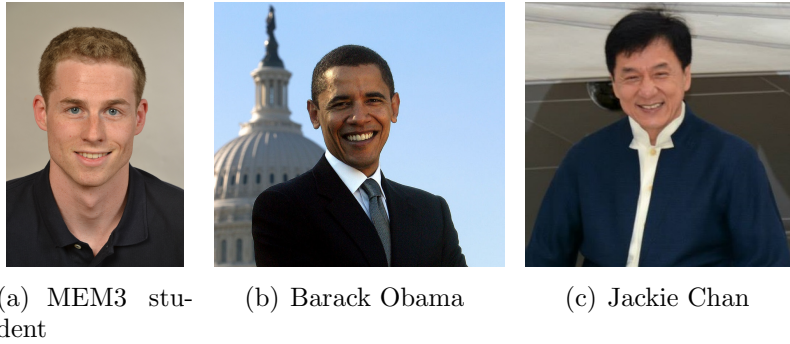


Figure 11: Test pictures

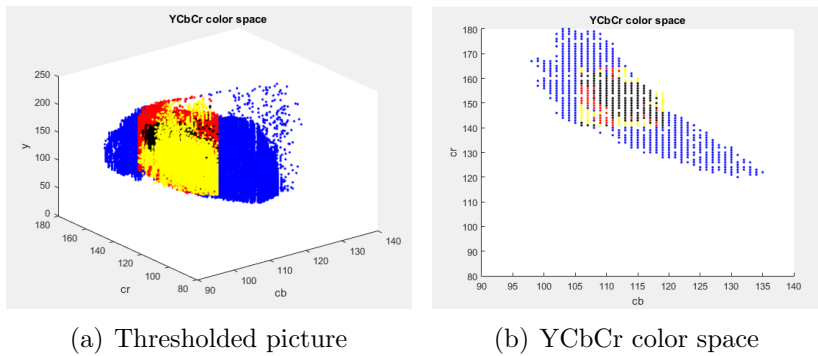


Figure 12: YCbCr - comparison between the full colored picture of MEM3 student (blue dots), thresholded MEM3 student (red dots), thresholded Jackie Chan (yellow dots) and thresholded Barack Obama (black dots).

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 ?, Face Detection Using Color Thresholding, and Eigenimage Template Matching ? and A Robust Skin Color Based Face Detection Algorithm ?) 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 13) the thresholds can be set (see figure 16).

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

- Cb: $105 > Cb < 120$
- Cr: $140 > Cr < 165$

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:

```
1 % Thresholding -> binary
2 thresh_cb = cb > 105 & cb < 120;    % thresholding for cb values
3 thresh_cr = cr > 140 & cr < 165;    % thresholding for cr values
4 binary_pic = thresh_cb&thresh_cr;    % create binary picture
```

The result looks like figure 15.

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.

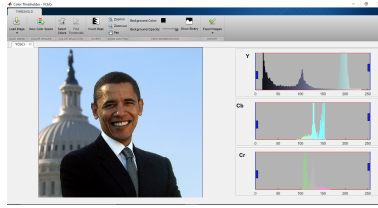


Figure 13: colorThresholder - Barack Obama - without thresholding

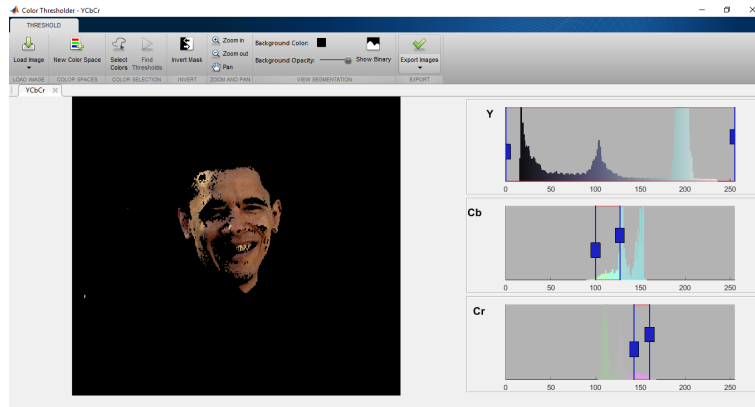
Figure 14: colorThresholder - Barack Obama - with thresholds : $105 > cb < 120$ and $140 > cr < 165$ 

Figure 15: colorThresholder - Barack Obama - with thresholds - Binary

A.2.2 MATLAB implementation

Listing A.1: Color thresholding

```
1 I=imread(' ../pictures/obama.jpg'); % load image from workspace
2
3 % RGB -> YCbCr
4 YCBCR = rgb2ycbcr(I); % transform image into YCbCr space
5 y = YCBCR(:,:,1); % extract Y value out of matrix
6 cb = YCBCR(:,:,2); % extract Cb value out of matrix
7 cr = YCBCR(:,:,3); % extract Cr value out of matrix
8
9 % Thresholding -> binary
10 thresh_cb = cb > 105 & cb < 120; % thresholding for cb values
11 thresh_cr = cr > 140 & cr < 165; % thresholding for cr values
12 binary_pic = thresh_cb & thresh_cr; % create binary picture
13
14 % show binary picture:
15 figure
16 imshow(binary_pic);
```

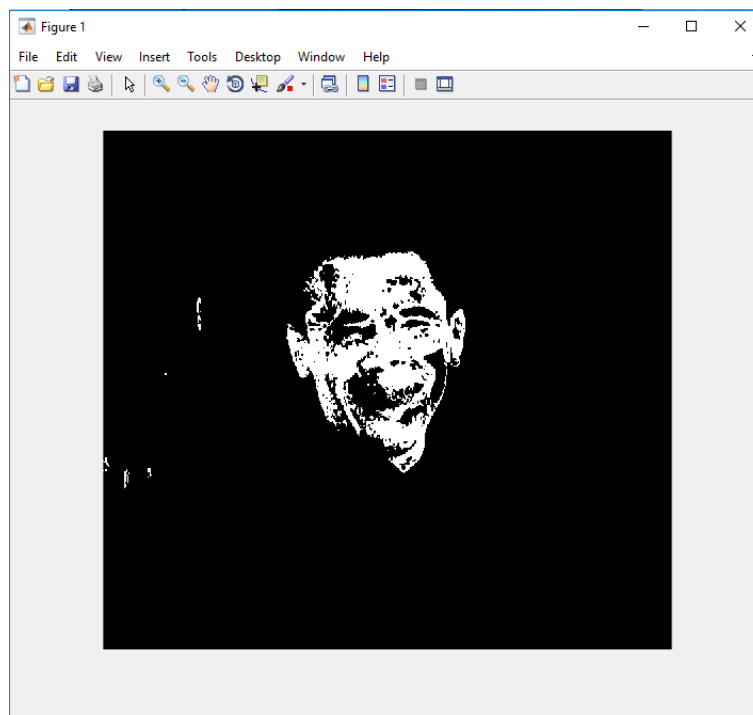


Figure 16: Binary Picture - Barack Obama - figure out of matlab code A.1

A.2.3 Examples

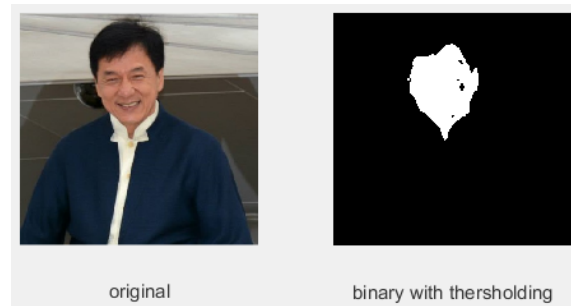


Figure 17: Chackie Jan

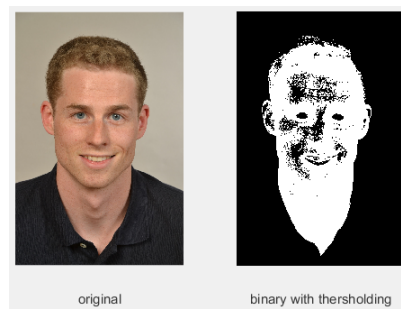


Figure 18: MEM student (private picture)



Figure 19: Nelson Mandela

B. MATLAB scripts

B.1 Show pictures in YCbCr space

B.1.1 Main script

```
1 % University of Applied Science Vorarlberg
2 % Master of Mechatronics
3 % -----
4 % Course:      Sensor Systems
5 % -----
6 % Author:      Tobias Burtscher and Stefan Stark
7 % Date:        07.12.2016
8 % Description: Script which shows picture (original and thresholded)
9 %              informaion in YCbCr space.
10
11 %%
12 clear all, close all, clc;          % clean up
13
14 %%
15 tLI = tic;                          % start a stopwatch timer
16 I = imread('pictures/mel_mod.jpg'); % load image
17 tLoadImage = toc(tLI)               % stop timer to see how much time is
18                                     % necessary to load an image
19
20 tTI = tic;                          % start a stopwatch timer
21 YBCR = rgb2ycbcr(I);               % transfare image into the ycbcr space
22 y = YBCR(:,:,1);                   % separete variable
23 cb = YBCR(:,:,2);                  % separete variable
24 cr = YBCR(:,:,3);                  % separete variable
25 tTransformImage = toc(tTI)          % stop timer
26
27 % Schow image in YBCR color space
28 figure
29 plot3(cb,cr,y,'b.')
30 hold on;                           % to draw the thresholded values into
```



```

31                                     % the same figure
32
33 %% draw into the same figure the thresholds of different pictures
34 % load image from disk and save rgb values as matrix
35 image1 = imread('pictures/mel_mod.jpg');
36 image2 = imread('pictures/JackieChan.jpg');
37 image3 = imread('pictures/obama.jpg');
38
39 % define colors
40 colo = ['r' 'y' 'k'];
41
42 % define thresholds
43 cb_low = 105; cb_high = 120;
44 cr_low = 140; cr_high = 165;
45
46 % run function which
47 %   calculates cb and cr values,
48 %   make thresholding and
49 %   create YCbCr plot;
50 createYCbCrPlot3(rgb2ycbcr(image1), cb_low, cb_high, cr_low, cr_high, colo(1))
51 createYCbCrPlot3(rgb2ycbcr(image2), cb_low, cb_high, cr_low, cr_high, colo(3))
52 createYCbCrPlot3(rgb2ycbcr(image3), cb_low, cb_high, cr_low, cr_high, colo(2))
53
54
55 hold off;
56 xlim([90 140]);ylim([80 180]);          % scale axis for better view
57 xlabel('cb');ylabel('cr');zlabel('y');    % label axis
58 title('YCbCr color space');              % create title of figure

```

B.1.2 functions

```

1 % University of Applied Science Vorarlberg
2 % Master of Mechatronics
3 % -----
4 % Course:          Sensor Systems
5 % -----
6 % Author:          Tobias Burtscher and Stefan Stark
7 % Date:            07.12.2016
8 % Description:     Function which separate Y, Cb and Cr values out of
9 %                  transformed picture; threshold the picture and show
10 %                 thresholded picture pixels in an existing figure
11
12 function createYCbCrPlot3( YCBCR,cb_low,cb_high,cr_low,cr_high,colo)
13     y = YCBCR(:, :, 1);          % separate variable
14     cb = YCBCR(:, :, 2);         % separate variable
15     cr = YCBCR(:, :, 3);         % separate variable
16     % Thresholding the image

```

```
17     thresh_cb = cb > cb_low & cb < cb_high;
18     thresh_cr = cr > cr_low & cr < cr_high;
19
20     %define color
21     colo = sprintf('%s.', colo);
22     % plot thresholded picture into YCbCr figure
23     plot3(cb.*(uint8(thresh_cb)), cr.*(uint8(thresh_cr)), y, colo);
24
25 end
```

C. Simulink Models

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

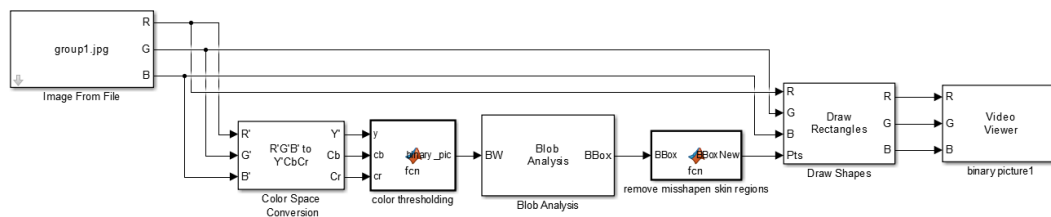


Figure 20: Simulink model - Face detection on an image

Listing C.1: MATLAB function block - color thresholding

```

1  % University of Applied Science Vorarlberg
2  % Master of Mechatronics
3  % -----
4  % Course: Sensor Systems
5  % -----
6  % Author:      Tobias Burtscher and Stefan Stark
7  % Date:        10.12.2016
8  % Description: Function which thresholds the YCbCr picture for face
9  %               detection
10
11 function binary_pic = fcn(y,cb,cr)
12     % Thresholding -> binary
13     thresh_cb = cb > 76 & cb < 125;    % thresholding for cb values
14     thresh_cr = cr > 140 & cr < 165;    % thresholding for cr values
15     binary_pic = thresh_cb & thresh_cr; % create binary picture
16
17     y_th = y;                          % set output
18     cb_th = cb.*(uint8(thresh_cb));    % set output cb

```

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

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

```
1  % University of Applied Science Vorarlberg
2  % Master of Mechatronics
3  % -----
4  % Course: Sensor Systems
5  % -----
6  % Author:      Tobias Burtscher and Stefan Stark
7  % Date:        10.12.2016
8  % Description: Function to reject boxes which are wider than high.
9  %              Remove misshapen boxes.
10
11 function BBoxNew = fcn(BBox)
12     [row col] = size(BBox);    % extract amount of rows from BBox
13     count = int32(1);          % initialize a count variable
14
15     for i=1:row
16         width = BBox(i,3);     % extract width out of BBox, for each blob
17         height = BBox(i,4);    % extract height out of BBox, for each blob
18         ratio = width/height;  % calculate ratio
19
20         if ratio > 1.5          % if box is wider than height
21             BBox(i,:) = 0;     % set entrie to zero
22         else                    % ratio is good -> possible face
23             BBox(count,:) = BBox(i,:);
24             count = count+1;    % count is necessary to be sure that the
25                                 % first entries of BBox are boxes with the
26                                 % right ratio.
27         end
28     end
29     BBoxNew = BBox(1:10,:);    % print the first 10 boxes
30 end
```

C.2 Simulink model to run on Raspberry Pi

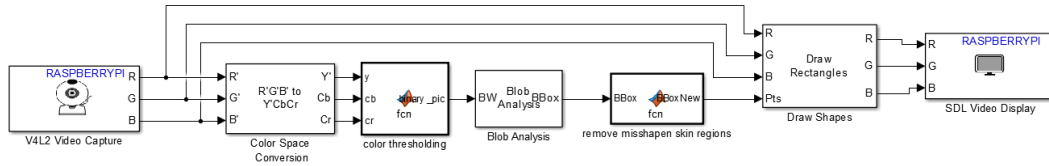


Figure 21: Simulink model - Run on Raspberry Pi