4. Implementation

4.1 Color based face detection on a picture using MATLAB

As mentioned in chapter 3.2.1, the first implementation step is to detect faces out of different pictures using the software MATLAB. Therefore an image is loaded into the MATLAB workspace and converted into the YCbCr color space. To separate skin pixels from non-skin pixels a thresholding for the Cb and Cr component is done. The chosen threshold values and the MATLAB code for this purpose can be seen in listing 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

The resulting binary image contains white areas which represent skin pixels (also called blobs), all the other pixels are converted into black. An example can be seen in figure XY. A detailed description of the color-thresholding can be found in the appendix A.2.



The next step is to decide if the found skin region really represents a face. As shown in figure XY the detected area also could be a hand or a leg. For this purpose a few steps, which are also described in the article Kumar and M (2014) are necessary.

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 ½, 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

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')

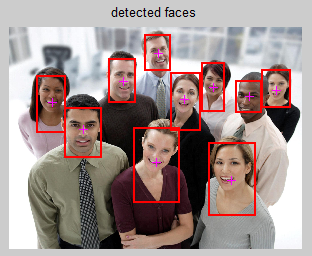
6 eulernumber=regionprops(bw,'EulerNumber');

7 eccentricity=regionprops(bw,'Eccentricity');

8 centroid=regionprops(bw,'Centroid');

9 boundingbox=regionprops(bw,'BoundingBox');

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



**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 XY 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 for the face detection on a picture using MATLAB can be found in appendix XY.

4.2 Color based face detection in a video stream using MATLAB

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

|  |  |  |
| --- | --- | --- |
|  | Best case | Worst case |
| Computation time for one snapshot | 0.12 | 0.19 |
| Frames per second | 8.33 | 5.26 |

The whole script for the face detection on a video using MATLAB can be found in appendix XY.

Findings

* Resolution of the image/video must be defined -> amount of blob pixels (area property) and computation time depend on it
* Thresholding values are different for different cameras
* Lighting conditions can affect the algorithm – e.g. shaddows
* Hands and other skin regions are often interpreted as faces
* 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 morphological operations are used