BEELD: Project: Face Detection

Ralph Bisschops, Dennis Cardinaels

January 3, 2016

1 Preface

This a report on the application that implements face detection in MATLAB. It is based on 'Face Detection in Color Images' [1]. We use a color image and try to detect human faces in the image. This is done by using the color components of the image to extrapolate where the faces are and to search for eyes and mouths.

2 Use cases

This implementation has many use cases but is mainly useful for academic usage or pictures. It might also be used for video but the processing of the frames is computationally intensive and doesn't take advantage of other techniques that could be applied for video.

2.1 General Face Detection

This algorithm had default parameters and can be used without any knowledge about the image. The only assumption is that the image is a color picture with a relative accurate white balance.

The code can be used as follows:

```
main('../test/img/1/1.jpg','../results/1.jpg');
```

Where the first parameter is the input image and the 2nd parameter is the output image. This will load the image and calculate multiple faces in the image and draw red dots on the eyes and a green dot for the mouth. It will also draw a white shape around the face itself. You can see this in figure 1. It is also possible to use the test.sh script as:

```
$ ./test.sh
```

This script will run matlab without the UI and execute the algorithm on multiple images.

2.2 Advanced Face Detection

If the default parameters don't result in a satisfying image you can also change the parameters if you please. Here we will list some parameters you can change in order to satisfy your needs. You can change them by editing the top part of the 'main.m' and by changing:

```
useDefaultProperties = 0;
```

debug Changes if you want to print more images as debug output.

LightingCompensation Can disable or change methode of LightingCompensation

Threshold Grouping with x% of image size

MinSizeCluster Min size cluster

MinSumSizeCluster Min size cluster after grouping

Sigma $min(W, H)/X$ where X is value. Bigger means smaller blobs. Smaller means bigger blobs and slower

SigmaRange limits blob min/max size

MaxImageSize Maximum size image in height and width (improves speed and some times result)

LongN Ratio of average C_r^2 to average C_r/C_b.

MaxEyes Maximum amount of eyes (if more found, stop searching)

MaxMouths Maximum amount of mouths (if more found, stop searching)

MultipleFaces If set it searches for multiple faces

EyeColor RGB color value eyes

MouthColor RGB color value mouth

FaceBoundaryColor RGB color value face boundary

You still use the function defined in the general case to execute.

2.3 Limitations

Because this is an algorithm mainly in spatial domain of the image it has its limitations. The biggest limitation is that the image has to be a color image and cannot be heavily edited. The background of the image is also important, if the background is skin colored or contains skinish colors the algorithm might result in false positives. We also assume that the eyes and mouth are visible and not obscured by other objects like sunglasses.

3 Implementation

Here we are going a little bit more in depth into the algorithm itself. We will also point out some differences in our implementations from the paper[1].

3.1 Lighting Compensation

Under different lighting conditions, skin colors can be harder to detect. We use the approach found in the paper by basing our compensation technique on the average RGB-values found in the highest 5 percent of the luma ("reference white pixels"). If there are not enough of these pixels (<100, used in the paper), then the color bias can not be accurately calculated. If this is the case, no lighting compensation takes place. We supply two other lighting compensation techniques which use the RGB color model. The corrected image is then transformed to the YCbCr color space for further processing.

3.2 Skin Color Detection

In this function we use the corrected image and implement 'Appendix B' of the paper and try to detect as much skin color as possible. This will create an image that shows all the position of all the skin tones found. The only thing we did not fully understand and so we made a slight change compared to the paper was $\overline{C_i}(K_h)$ which we changed to $\overline{C_b}(K_h) = 108$ and $\overline{C_r}(K_h) = 154$.

3.3 Variance Based Segmentation

This follows from the previous section and uses the image generated there to set a threshold and generate a mask. We did not differ from the paper here.

3.4 Connected Component Grouping

Now that we have generated a mask. We want to smooth it out because there are still a lot of separate pixels that are not grouped. We first fill all the holes in the image, this will fill everything inside an already enclosed space (like noses and mouths).

We then look at all the clusters of pixels and exclude all the groups that are smaller then 20 pixels (or defined otherwise). We filter the small cluster out because there might be a lot of them and they do not change the result much in most cases. If we should leave them in we would have to do more computations. For the remainder of the clusters we calculate the properties of the cluster as defined in 'Appendix A' of the paper.

After we have calculated all the properties it is time to compare and group all the clusters together that are closer then 2% (or defined otherwise) of the largest dimension of the image. We mark these clusters as merged, but we do not recalculate their properties. Once we have checked all the clusters with each other we make a new cluster map where they are merged. The mask is not much

different at this stage, only the smaller clusters are excluded. But we marked clusters as 'connected' if they met the threshold value.

Then we calculate the convex hull around the grouped clusters and return that list. We will use each cluster as a facemask.

Now that we have all the facemasks we will start with recognizing facial features for each of them.

3.5 Eye Detection

Eye Detection is not much different from the algorithm defined in the paper. Only the scale parameter σ was not defined in the paper so we used $\lceil \frac{min(W,H)}{26} \rceil$ where W and H are the width and height of the facemask respectively. σ can also be altered.

3.6 Mouth Detection

We found that the formula for the estimated ratio average C_r^2 to C_r/C_b did not give us the best result. After some testing we found that 0.5 gave us the best result. We use the same σ as for the eyes.

3.7 Feature Pyramid

We found the paper lacking some clarity in this part. We did not figure out why they wanted to use the pyramid decomposition to approximate the location of the eyes/mouth. We implemented this but later excluded this from the code. Instead we used the iterative thresholding and binary morphological closing to find the eyes and mouths. The σ defined in previous sections will change whether the blobs will be merged or not. This might alter the position of the eye/mouth a bit. We then use the center of the blob as the position of the eye or mouth.

3.8 Face Boundary Map

Here we took a different approach, due to problems calculating the Hough transform for getting an elliptical shape. Our implementation does not use a Hough transformation to get a face boundary, but the canny edge technique. We still calculate σ based on the formula (14) described in the paper, but we take the average of all possible eye-mouth-triangles because it yielded great results. At the end of the process, when we have the best eye-mouth-triangle as calculated in the next section, we remove small lines and dots from the face boundary. The results are great if there is a circle-like shape in the face boundary. If not, the results can be unpredictable and are not always correct, but this does not happen too often.

3.9 Weighting Triangles

To calculate which eye-mouth-triangle most accurately describes a human face, we use the same formulas found in the paper for the eye-mouth-weight and face-orientation-weight. Because we did not use the Hough transformation, we created a new weight. This weight is based on the ratio of the distance between the two eyes and the distance of the mouth to the midpoint of the two eyes. When a person faces the camera directly, we observed that this ratio was between 0.85 and 0.95 in most of the cases. If the ratio is higher than 1, then the eyes are unnaturally close to the mouth which results in a weight of 0 so that these false faces are discarded. When the eyes are too far from the eyes, it will result in a lower total weight. This approach yielded valid faces for most of the images.

4 Conclusion

In most images the the results are very good. As you can see in the results in figure 1.

References

[1] Rein-Lien Hsu, M. Abdel-Mottaleb, and A.K. Jain. Face detection in color images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):696–706, May 2002.

Figure 1: Output of the algorithm