FACE DETECTION THROUGH TEMPLATE MATCHING AND COLOR SEGMENTATION

By

Scott Tan Yeh Ping (yptan@stanford.edu)
Chun Hui Weng (cwengc@stanford.edu)
Boonping Lau (bppower@stanford.edu)

(EE 368 Final Project)

FACE DETECTION THROUGH TEMPLATE MATCHING AND COLOR SEGMENTATION

Scott Tan Yeh Ping, Chun Hui Weng, Boonping Lau

ABSTRACT

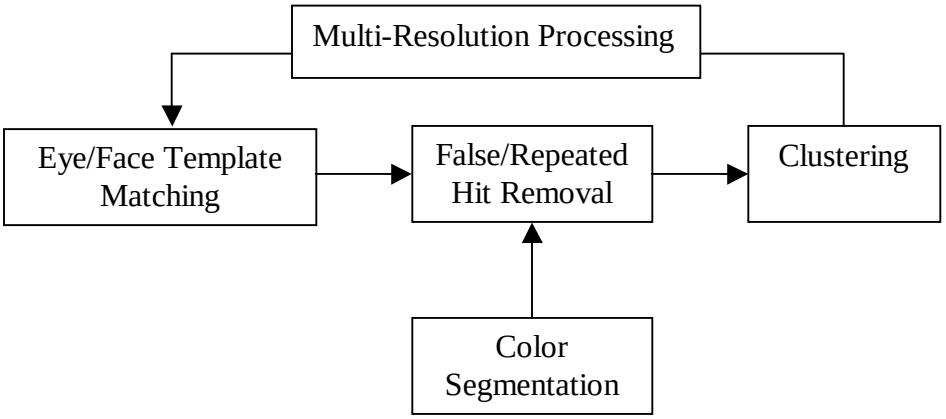
A process for face detection, which involves multi-resolution template matching, region clustering and color segmentation, works with high accuracy, and gives good statistical results with training images. Given the generality of the images and the templates used, the assumption would be that the implementation works well on other images, regardless of the scene lighting, size of faces or type of faces in the pictures.

INTRODUCTION

The increasing use of computer vision in security in place of humans led many to research the problem of face detection in images. The problem is not a trivial one as the classification of a human face proves to challenging. Despite the many variations of a human face, features can still be found, given a certain context, which will uniquely identify a face. In this project, we attempt to come up with a technique to give a high percentage of face detection based on correlation and post-processing. The first part of this paper will concentrate on the approach used in solving this problem, namely: correlation, false/repeated hits removal techniques, color segmentation and multi-resolution approach. The second section will go into the actual results of the implementation, where statistics taken after performing the algorithm on training images will be provided. A conclusion will then be given.

APPROACH

The algorithm used for face detection in this project is given below:



Template matching is performed first to find the regions of high correlation with the face and eyes templates. Subsequently, using a mask derived from color segmentation and cleaned by texture filtering and various binary operations, the false and repeated hits are removed from the template matching result. The output of this process is then passed to a clustering procedure, where points are within a certain euclidean distance from one another will be clustered into one point. The whole process will then be repeated at a different scale/resolution. The outputs from each resolution are then recombined into a single mask.

The individual processes will be explained in detail below.

COLOR SEGMENTATION

The aim of colorspace transformation is to increase the separability between skin and non-skin classes while decreasing the separability among skin tones. Hopefully it will bring robust performance under varying illumination conditions. However, there are many colorspaces to choose from and a large number of metrics to judge whether they are effective.

Some potential colorspaces that we were interested in :

- CIEXYZ
- CIEXYZ
- YCbCr
- YUV
- YIQ

A performance metric that others have used includes the computation of the separability of clusters of skin and non-skin pixels using scatter matrices. Another is to do a histogram comparison of the skin and non-skin pixels after colorspace transformation. The YCbCr colorspace was found to perform very well in 3 out of the 4 peformance metrics used[1], so we decided to use it in our color segmentation algorithm.

Here is the equation for transforming from RGB to YCbCr.

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & -74.203 & 112 \\ 112 & -93.786 & -18.214 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

To transform back…

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 0.00456621 & 0 & 0.00625893 \\ 0.00456621 & -0.00153632 & -0.00318811 \\ 0.00456621 & 0.00791071 & 0 \end{bmatrix} \left(\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} - \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \right)$$

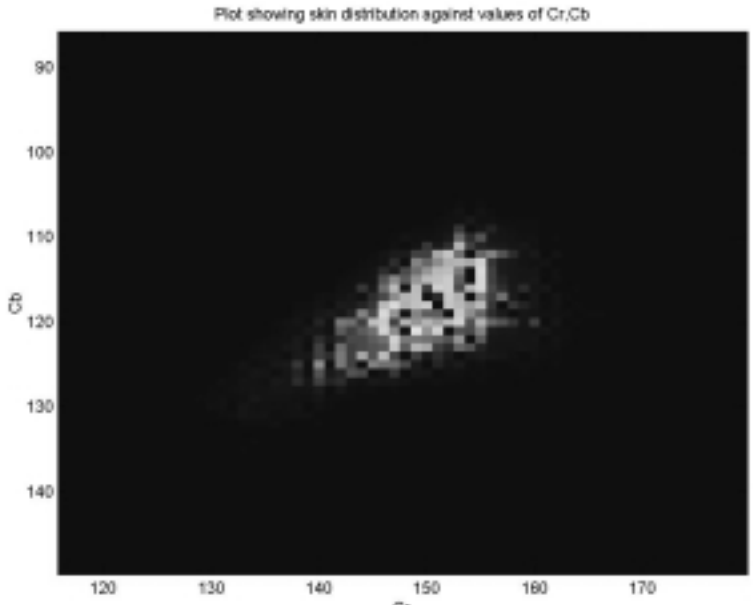


Fig 1. Skin color distribution histogram in Cb-Cr plane based on pixels under ref.png*

[1] http://marathon.csee.usf.edu/~tsap/papers/wacv02.pdf

This is a plot of the skin distribution against Cr and Cb in the YCbCr colorspace. If the sample size is large enough, it will be a probability distribution for the human skin. However, our data was restricted to the training pictures we had and we did our best to find the best skin filter. Attempts were made to retrieve online databases for reference to generalize our skin filter, however it was found that the values that work best were still those derived from the training data at the loss of generalization.

Many studies have come to the conclusion that pixels belonging to skin region have similar Cb and Cr values. It is also conjectured that the perceived difference in skin color cannot be differentiated from the chrominance information of that region. The fairness or darkness of the skin is characterized by the difference in the brightness of the color, which is determined by Y. Therefore solely using Cb and Cr will generalize the filter for people of all skin colors.

Several approaches were tried to define a boundary for skin and non-skin pixels. The first one we tried was an elliptical approach. Basically, it was to try to find the best fitting ellipses to enclose the non-zero region we have above.

Another approach we tried was to use a histogram model. Basically we built a 256x256 mapping function from (Cr,Cb) to the decision function of skin or non-skin. By thresholding, we have a set S of tuples of Cr, Cb values that will be considered as skin. If the number of skin pixels falling into that particular bin with a certain (Cr, Cb) value exceeds a certain threshold, we will consider that a skin color.

$$f(Cr,Cb) = \begin{cases} 1 & \text{if } (Cr,Cb) \in S \\ 0 & \text{if } (Cr,Cb) \notin S \end{cases}$$

This approach is better than the elliptical approach in a few aspects. Basically, we can have a better bounding shape than an ellipse. However, it could overtrain the model we are trying to build. It is also computationally simpler than an ellipse since it just involves looking up in the map to see if the current pixel should be classified as skin.

The output from the histogram model has to undergo a morphological operation. For this purpose, MATLAB's *imfill* was used. It is a function that will change zeros to ones when the zeros are fully surrounded by ones.

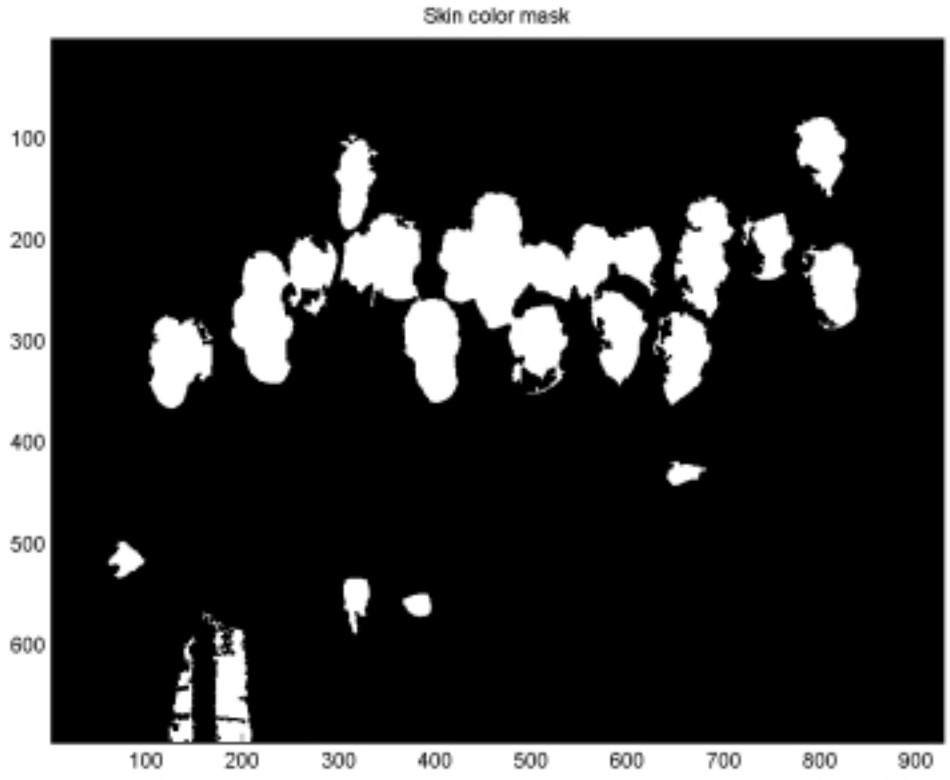


Fig 2 An example of a mask produced by color segmentation, then processed by texture filtering and various binary operations.

As can be seen from above, most of the skin pixels have been correctly classified. Background that is close to skin color will have a chance of being wrongly classified too, but further processing by correlation should be able to iron out the false hits.

After extracting the pixels based on their color values, we performed the following binary operations on the color mask to remove noise:

- Close
- Fill holes
- Remove small areas
- Texture Filtering using variance thresholding

Aspect ratio criterions are found to be unreliable because the faces are too clustered, and may give false aspect ratios on regions. Binary operations, when applied in excess, will cause us to lose generality of the mask and give good results only for a specific image. Thus we discarded the former and used the latter sparingly, especially erode and dilate operations since it was impossible to justify the structuring elements used and the number of operations performed convincingly.

TEMPLATE MATCHING

For a primary face detection method, we had many possible choices like neural nets, PCA etc. We knew that neural nets are simple to implement but tend to be over-trained on such a limited training sample (especially if we need to split the images into a disjoint test set and training set. Partially occluded faces also may cause a problem, and it is difficult to predict the response of a neural net in these circumstances. PCA involves lowering the dimensionality of an image and may face the same problem of over-specification of the face-subspace due to the very limited training set.

Ultimately we chose template matching as the primary face detection method due to its conceptual simplicity and our confidence in its inherent extensibility to more general images and predictable response in the case of partially occluded faces, even with such a limited training set. This effectively pre-empts the problem of obtaining a fair and big sample of training faces from other sources. Another consideration was the ability to combine feature detection with face detection very easily, with only the extra step of creating more templates.

Although template matching is basically the two-dimensional cross-correlation of a grayscale image with a grayscale template, hence estimating the degree of similarity between the two, there are numerous details involved that would have drastic influence on the overall performance of the system:

Choice/Quality of Template

Evidently we need to choose a suitable template that is in effect a 'general' face that has all the features of a face without being too specific. These conflicting requirements determine the performance of the template matching stage, and with exactly the same code, detection rate varied tremendously with different templates.

A study of the extreme cases bears out this fact. Our first attempt was to simply pick a 'normal-looking' face and lowpass it to blur out the specific details and use that as a template. It turned out to be capable of detecting about 50% of the faces in any image as expected, illustrating over-specifying of the face template.

On our 2nd attempt at using a linear combination of eigenfaces, we generated the eigenfaces using preprocessing code to automatically rotate training face images back to the vertical orientation and scale them to the same size. The resultant eigenfaces were obtained very quickly but has a very blurry quality due to imprecise rotational orientations and improperly centered images. This gives a very general template that could be anybody's face, over-generalized to an oval shaped object. Consequently the result from the template matching was comparable to simply using a white oval template with a black background.

(However, note that this degradation to ‘shape’ matching could be useful if we had a close-to-perfect result from a color segmentation stage with which to combine it. We would be able to locate the centroid of the oval-shaped regions accurately. This warrants further investigation, but we unfortunately had to leave it aside and focus on what we set out to do given the timespan of the project.)

This ruled out automatic preprocessing of training faces for use in building a template, and we used Photoshop to manually rotate, crop, and center faces from the 7 training images before superposing them to create an average-face template. To overcome the problems of face shape variations, we limited the template to a square region within the face cropped from the eyebrows to the mouth vertically and to the outer edge of the eyes horizontally (see Fig.3b This also has the potential of performing much better in the case of occluded faces since it focuses on the most central part of the face with the most features. (In effect, if this template region is greatly occluded, even human perception becomes difficult in a noisy background.)

This choice of face region as a template distinctly kept the relative spatial location of the facial features yet allowed a great degree of generalization to different shaped faces, and as such is a good compromise suitable for the conditions of the project.

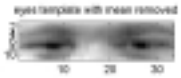


Fig 3a) General eye template used.

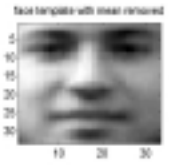


Fig 3b) General face template used

Choice of Threshold

To classify each image pixel as a face or non-face from the correlation result, we need to choose a good threshold value. This is a difficult choice and risk degrading into playing with a heuristics and losing performance on general images.

Fixing a hard threshold is difficult as the correlation result must be properly normalized to remove the effects of image size. This problem will be compounded when we try to combine the results from different resolutions in the multi-resolution extension. Moreover this is greatly dependent on the degree of clutter in the image and lighting intensities. In fact, we are not even sure that there is going to be a face in any general image, thus we cannot be tempted to use correlation maximas as the sole criteria.

We decided to use the mean plus an arbitrary number of standard deviations as the correlation threshold value. Thus the threshold is adaptive and adjusts to different images, lighting conditions, degree of clutter etc. Each face is equally likely to exceed the threshold while background will be suppressed as low correlation relative to the mean correlation. By playing with the number of standard deviations above the mean (fixed at 3 based on empirical results), we are setting a confidence level of our template matching.

Pre-processing/Post-processing before Template Matching

To adjust for different lighting conditions and allow for proper correlation, we use a template with its mean subtracted, and we remove the local mean over a template-sized region of the grayscale image. The latter is difficult to implement efficiently as we are reluctant to pass a window over the entire image calculating local mean values. Instead, we use a lowpass approximation to the mean i.e. we simply subtract a suitable lowpass filtered version of the image (a simple filter of dimensions related to template size is used) from itself to remove the mean and also slow variations of lighting intensity over the image. This allows for a fair correlation invariant to the lighting condition (see Fig 4 below).

After correlation, we discard hits too near the edges of the image as these are unreliable due to edge distortions (template window partially off the image).

Fig 4. Imsge with local mean removed

Processing Time

A two-dimensional correlation calculation over a large image is very time-consuming, and to overcome this, we did the correlation in the frequency domain by multiplying the two-dimensional FFTs the Hermitian template and the image, then took the inverse FFT to obtain the output image. This significantly reduces the processing time by a few orders of magnitude.

As an additional gain, we chose to subsample the input image by a 2:1 factor, adjusting the template sizes accordingly for proper detection. This does not seem to degrade correlation results at all, but should not be done if the image is already small enough.

Enhancements of template matching stage:

The purpose of this template matching stage is to detect all the faces, in this respect we have achieved a good result with **162/164 faces detected** (using 4 resolutions, see section on **multi-resolution**), including very small occluded faces.

However, this leads to multiple false and repeated hits, and in order to eliminate a large proportion of these, we decided to use a secondary stage of **eyes template matching**, considering an overall hit only if both eyes and face were detected. This has an additional advantage as it allows us to relax the individual thresholds to get all the possible hits, especially the smaller occluded faces, before eliminating some of the false and repeated hits. This combines the strength of feature detection to detect partial faces and the reliability of face detection for full frontal shots.

Another advantage is that by dilating the hit pixels from eye and face template matching separately before combining them, we allow an extra degree of limited rotational invariance ie. an eyes hit need not be detected directly above a face hit (face hit is marked on nose, eyes hit is marked between the eyes) but can be a slightly shifted. This dilation also allows better detection of elongated faces by relaxing the distance constraint between the spot between the eyes and the tip of the nose in any face.

In any case, there are other following stages designed specifically to overcome the problems of false and repeated hits as developed later.

FALSE/REPEATED HITS REMOVAL

The face detection process will very often generate repeated hits on a face after applying the color mask to the correlation result., due to a relaxed correlation threshold to detect more faces and the effects of combining results from multiple resolutions.

This is due to the fact that we choose to apply the color mask after correlation instead of otherwise. This implementation will give non-zero correlation values outside the face regions, which will affect the outcome of the correlation. False hits will also be produced because certain regions have features that resemble that of a face when template matched. However, we believe that this will give better performance in detecting very small, occluded faces, as observed in our results. Also, it is less dependent on the quality of the color segmentation mask.

To remove the false/repeated hits, we will make use of the color mask, since the color mask gives us an approximate indication of where the skin part of the picture lies. Not only does the color mask provide that information, it also shows the approximate shape of the area of pixels deemed to be skin. We want to be able to remove hits that are on the edge of faces or hits that belong to skin pixels that are not from the face i.e. not in the areas of a face shape. In the latter case, this method also attempts to remove false hits.

The algorithm to do that involves another form of template matching, where a binary mask, of a certain size, bearing the approximate shape of a face is applied (elemental multiplication) to the correlation-cum-color-mask output at the centroid of every area of high correlation. The number of pixels that is of value 1 is that calculated and divided by the number of 1's in the original binary mask used. This will give us a percentage value that can be checked against a threshold percentage value. Below the threshold, the correlation is rejected, removing repeated hits.

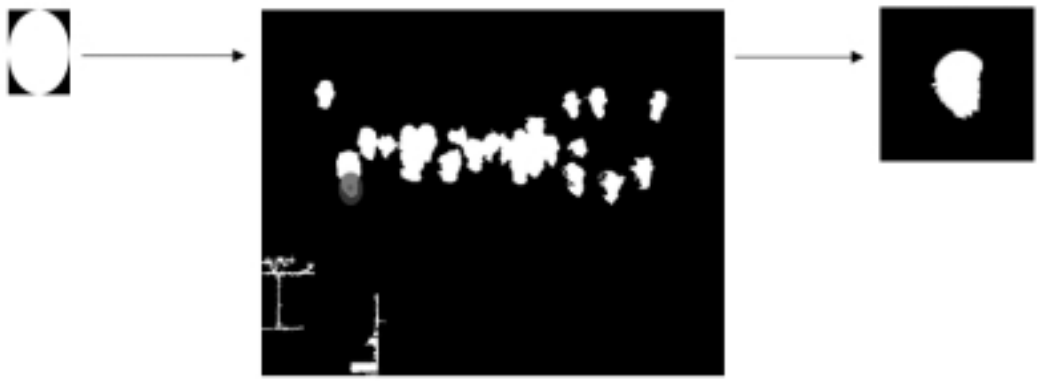


Fig 5 a) Series of pictures showing the rejection of a point close to the edge of a face with 85% acceptance rate.

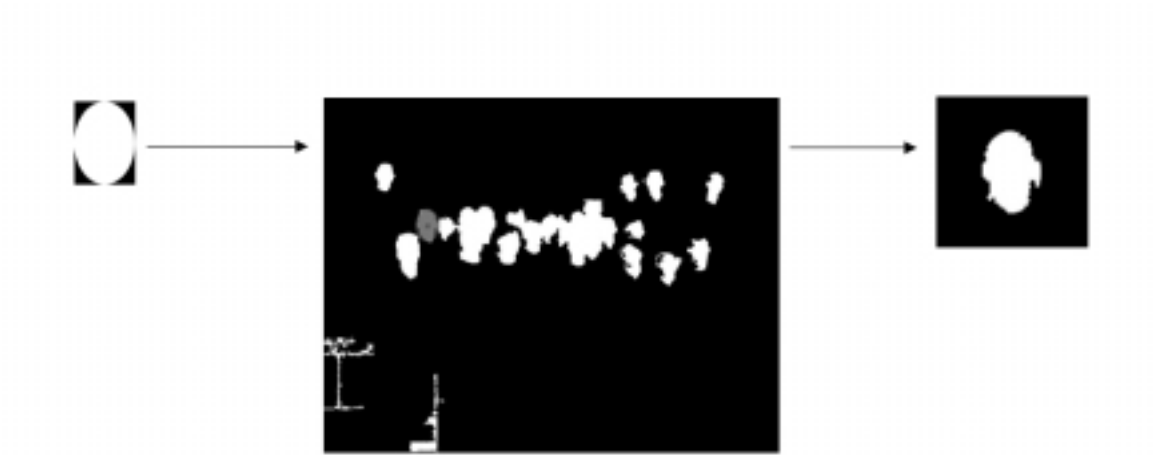


Fig 5 b) Series of pictures showing the acceptance of a point near the middle of the face with 85% acceptance rate.

For the first picture, note that the area of overlap is less than 85 percent of the actual template used, which is just an ellipse. Hence, that region of high correlation (as represented by the red patch) is rejected. In the second case however, the area of overlap is almost over the entire template, hence the region is accepted.

This technique of removing false/repeated hits works very well in general, but has to be done at each resolution separately as we need to adapt to the template size at each resolution.

CLUSTERING

After correlation with a face template, verification with a skin mask and false/repeated hits removal at each resolution, there might still be multiple hits on the same face. Furthermore, a hit might occupy area more than a pixel. Basically the next task is to cluster hits to reduce multiple detections. There are many algorithms to do clustering, but many of them require prior knowledge of how many clusters one would expect. However, if we were to make no assumptions about how many people are in the picture, we cannot tell how many clusters we will be expecting.

Nonetheless, we can make use of the knowledge that anything that is a face and correlates strongly with the face template must have about the same dimensions as the face. This led us to come up with a clustering algorithm that is similar to the K-means.

1. Use MATLAB's *bwlabel* to label each region.

2. Find the centroid of each region and calculate the area it occupies. This is sort of a weight for each hit.

3. Starting with the region that has the highest weight, for each point representing the centroid, determine if the distance between the pixel and the nearest point is < dimensions of template. If not, merge the pixel and its nearest to a point using their weights. $[x_3\ y_3] = (w_1 + w_2)^{-1} (w_1[x_1\ y_1] + w_2[x_2\ y_2])$

4. Keep repeating until no point violates the minimum distance condition anymore.

This is applied separately to the result from the previous false/repeated hits removing phase from each resolution.

The clustering algorithm is also the reason why the location of the hit will not be in exact center of the face. Although repeated hits in the same face tend to have lower weight than those on the nose, our centroid-weighting algorithm (see step 2 above) will possibly cause a slight shift. In most cases, the weight of the hit on the nose is so high as to make it negligible.

Another use of the clustering algorithm is to allow us to collapse the final hits from each resolution back into a single mask. To do this we simply call the clustering algorithm after using a logical OR operation to combine the multi-resolution masks together. Again this causes slight shifts of hits from the noses, which can be significant for profile faces etc.

MULTI-RESOLUTION PROCESSING

The above process is performed on images at various levels of subsampling. The purpose of subsampling i.e. the multi-resolution approach, is that some faces are smaller than other faces. Since the process of template matching relies on a template of fixed size, it is important that multi-resolution processing be applied to template matching. If not, a general result cannot be achieved as faces that are too small or too big, relative to the template, will not be detected in a correlation process. Hence, multi-resolution processing makes the face detection scale-invariant.

In this project, subsampling and upsampling is performed on the template instead of the image to be processed. This is an equivalent process to that described above. This is simpler in implementation, since the subsampling and upsampling can be done offline and stored in files, saving time during actual detection. True multi-resolution is achievable with a Gaussian pyramid, but was disabled to save runtime.

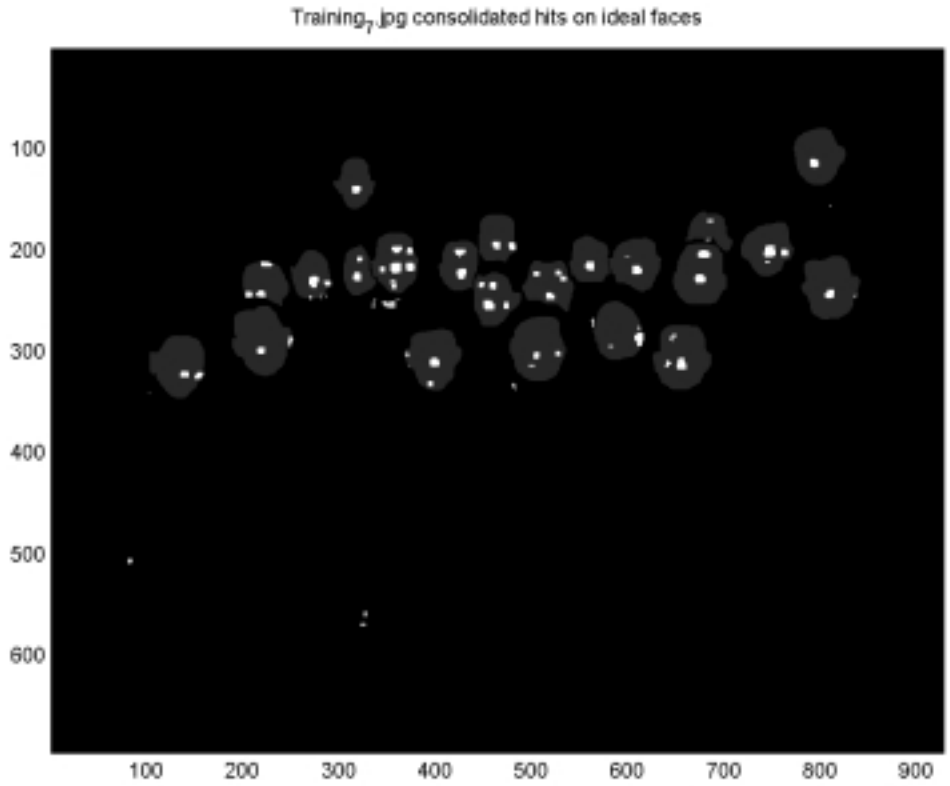


Fig 6. Example of false and repeated hits generated when all the multi-resolution results were combined. Hence, we need to do false/repeated hits removal and clustering to remove all the false hits.

RESULTS

In our tests, the test images are first subsampled by 2 to speed up the process (the excess resolution is not required). The multi-resolution process is then taken to 90%, 100%, 120% and 130% of the resultant image. The threshold used for color segmentation is 900, based on a sample of 1.5 million pixels obtained from the face pixels in the training data. The threshold used for false/repeated hits removal is 75%. The threshold used for clustering is 28 (roughly a face width). For a better color segmentation, small area rejection and texture filtering is used. The variance used for texture filtering is 1000. The area used for small area rejection is 500.

The following results are obtained:

Image	**detected-false-repeat/total**	**Results**	**Score**	
Training_1	20-1-0/21=19/21	90%	19	
Training_2	23-1-0/24=22/24	92%	22(24)	debatable result, occluded face, pixel actually on face if not occluded
Training_3	22-2-0/25=20/25	80%	20	
Training_4	21-2-0/24=19/24	79%	19	
Training_5	18-2-0/24=16/24	67%	16(20)	debatable result, miss by 1 pixel only!! Arbitrarily defined edge of face
Training_6	22-0-0/24=22/24	92%	22	
Training_7	22-0-0/22=22/22	100%	22	
Evaluate Total	**148-8-0/164=140/164**	**85%**	Based on result of *evaluate.m*	
Real Total	**151-5-0/164=146/164**	**89%**	Based on our interpretation	

Intermediate result images are generated all through the process and are shown below for a sample input image.

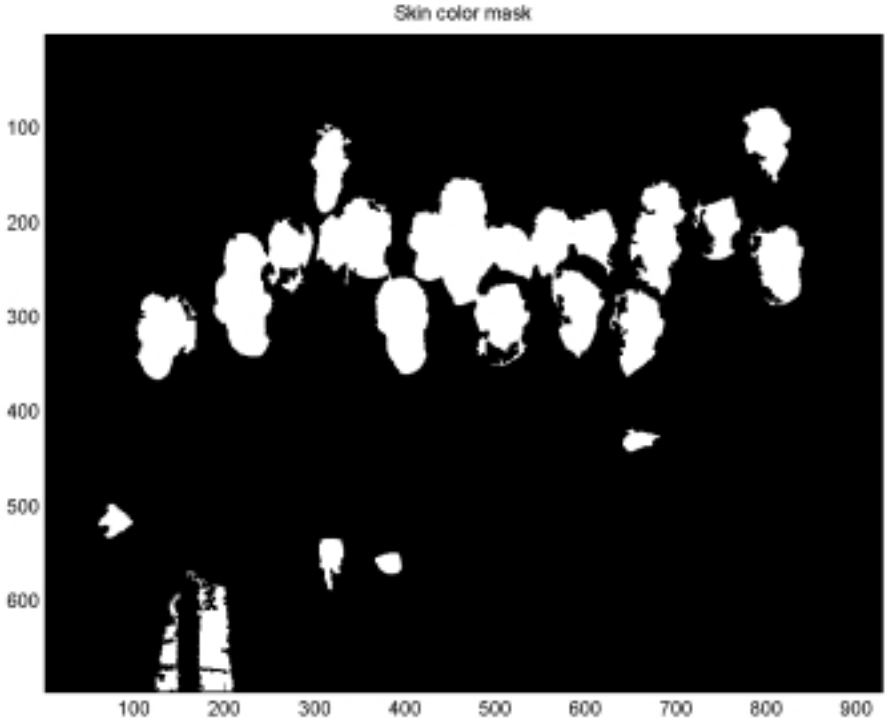


Fig 7 a) Mask resulting from color segmentation and various binary operations

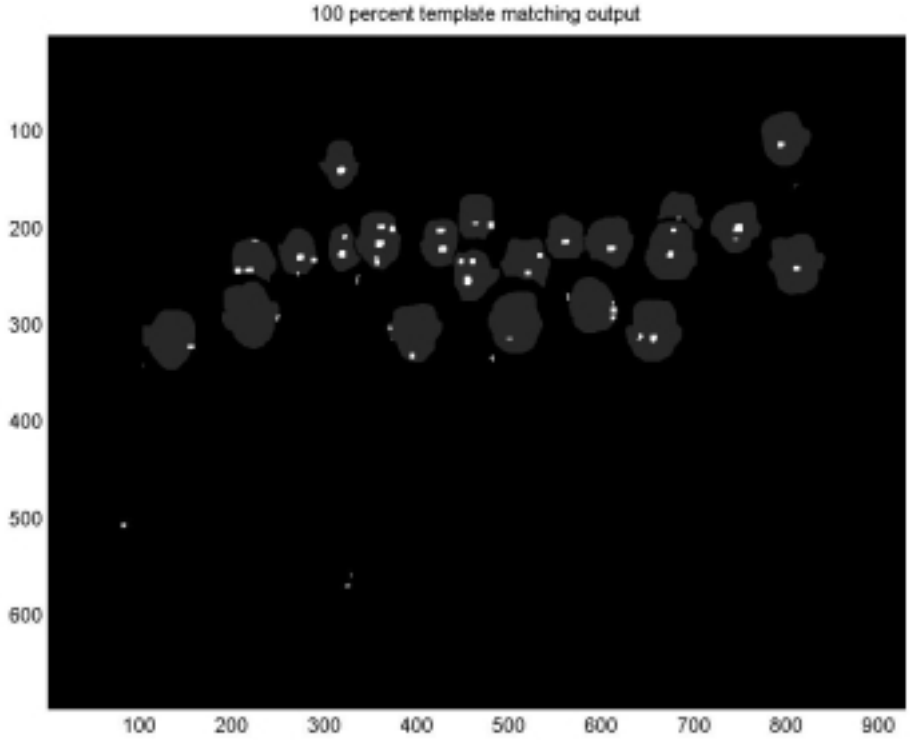


Fig 7 b) Template matching results using eye template and face template at 100% resolution

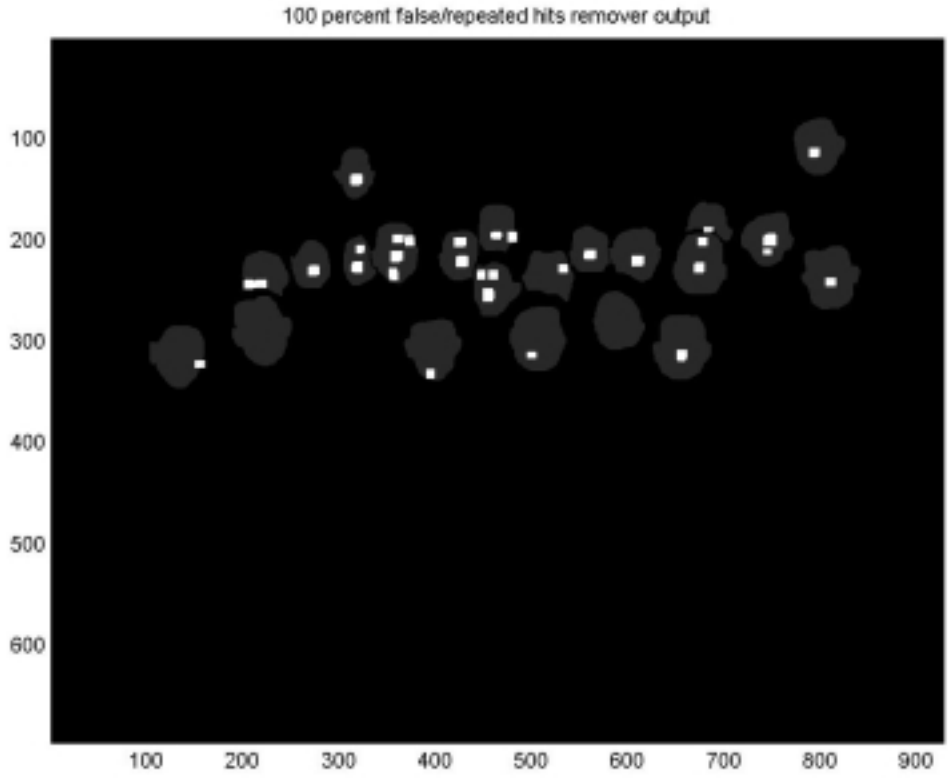


Fig 7 c) Output of false/repeated hits removal. Note the removal of hits near the edges of the face.

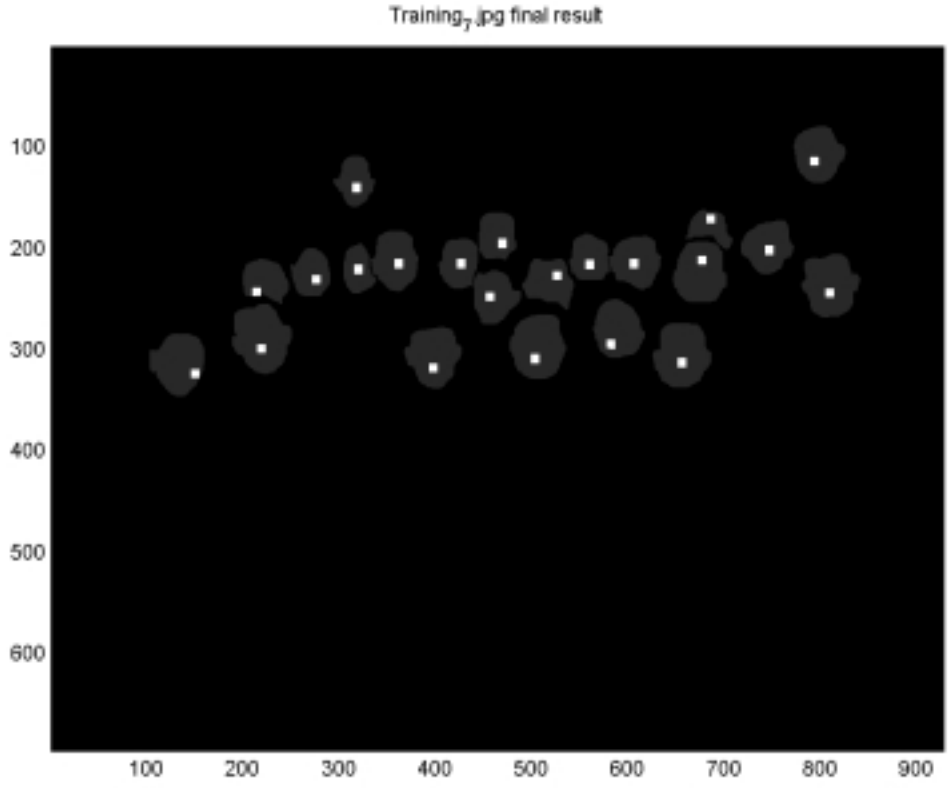


Fig 7 d) Hits consolidated from all the resolutions.

Fig 7 e) Final result. Faces are marked with red dots.

Fig 7 f) Another example. Faces are marked with red dots.

Given the generality of the images and the templates used, the assumption would be that the implementation would work well on other images, regardless of the scene lighting, size of faces or type of faces in the pictures. The only problem is the color segmentation mask which was done on pixels under the ref*.png masks. This includes a lot of non-skin pixels and is a non-general sample base. With a better color mask built using an identical method on a much wider skin pixel sample base, the same results will be generally achievable on any image.

CONCLUSION

Given the favorable statistics of results of this implementation, it is concluded that the method, which involves multi-resolution template matching, region clustering and color segmentation works with high accuracy with the training images which actually reflect worse-than-average face-detection conditions due to the high clustering factor of the faces, presence of profile faces, rotated faces etc. These results are therefore excellent.

BIBLIOGRAPHY

[1] Jae Y. Lee and Suk I. Yoo, "An Elliptical Boundary Model for Skin Color Detection"
[2] L. Fan and K.K. Sung, "Face Detection and Pose Alignment Using Color, Shape and Texture Information", *Proc. Visual Surveillance, 2000*
[3] J.-C. Terrillon, M.N. Shirazi, H. Fukamachi, and S. Akamatsu "Comparative Performance of Different Skin Chrominance Models and Chrominance Spaces for the Automatic detection of Human Faces in Color Images", *Proc. Automatic Face and Gesture Recognition, 2000*
[4] B. Menser and F. Muller, "Face Detection in Color Images Using Principal Components Analysis