Face Detection

Gary Chern, Paul Gurney, and Jared Starman

1. Introduction

Automatic face detection is a complex problem in image processing. Many methods exist to solve this problem such as template matching, Fisher Linear Discriminant, Neural Networks, SVM, and MRC. Success has been achieved with each method to varying degrees and complexities.

The assignment given to us was to develop an algorithm capable of locating each face in a color image of the class. We were given seven training images along with the corresponding ground truth data to develop and train our algorithms on. The end result for our group was an algorithm capable of finding over 95% of the faces in all but one image in approximately 30 seconds. In addition, we are able to successfully locate one of the females in two test images.

2. Our Algorithm

Figure 1 shows the face detection algorithm that we developed.

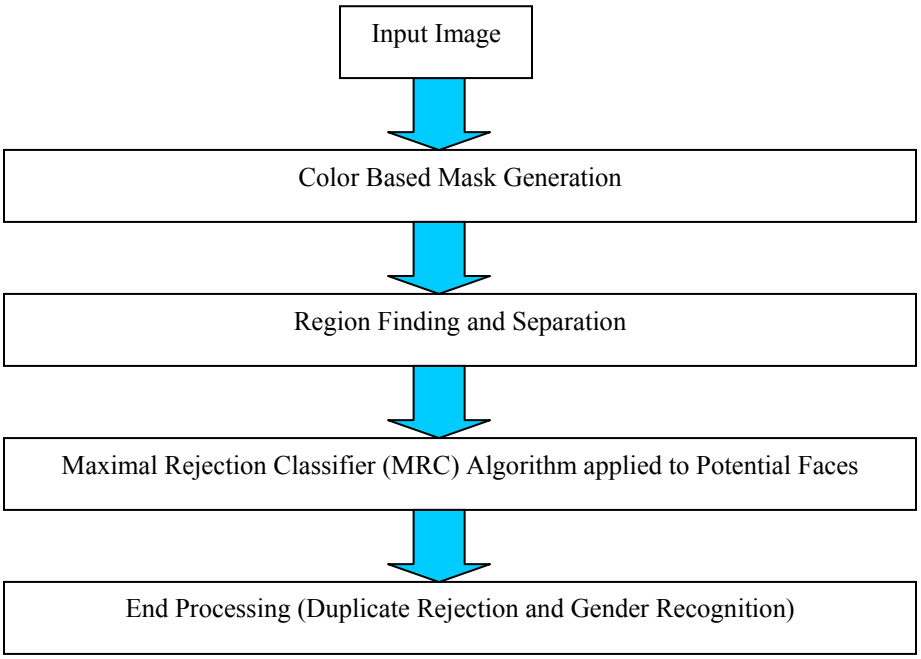

Figure 1: Block Diagram of our Face Detection Algorithm

The Color Based Mask Generation will be discussed in Section 3, Region Finding and Separation in Section 4, the MRC Algorithm in Section 5, and the End Processing in Section 6.

3. Color-based Mask Generation

We would like to reduce the number of locations in the image that need to be searched for faces. Color-based mask generation picks out the pixels which are most likely to be faces which can then be searched using more advanced techniques.

The first step in our algorithm is to assign the probability of being a face to every pixel in the image. We begin by using the training set to determine the distribution in RGB-space of the face pixels and the background pixels. Figure 2 shows the bounding regions for face pixels and background pixels based on the training images. (Note that the RGB values range from 0 to 63 in Figure to reduce the amount of memory required).

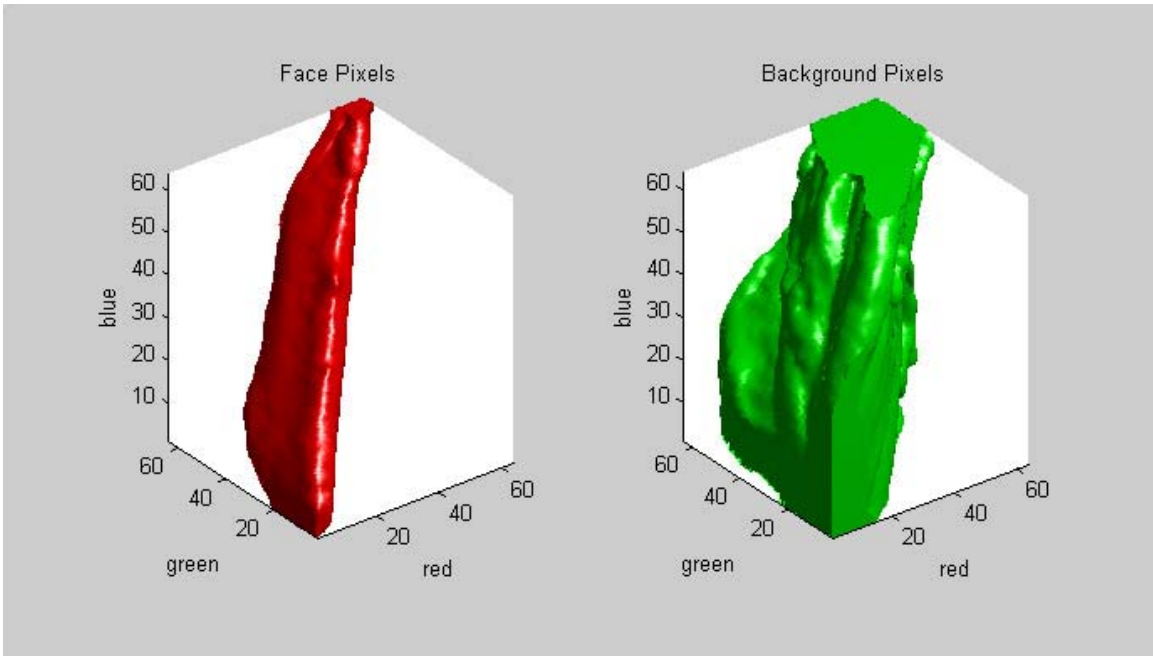

Figure 2: 3-D space spanned by Face and Non-face RGB color vectors

The simplest option for face detection would be to find only those pixels which are contained in the bounding region for face pixels. However, there is a noticeable overlap between the face pixels region and the background pixels region.

If we take transverse slices of the 3d plot shown above, we get Figure3 which shows the distribution of face pixels (red), background pixels (green), and where they overlap (yellow). We would like to assign a high probability of being a face pixel to pixels which reside in the red region, a medium probability to those which reside in the yellow region, and a low probability to those which lie in the green/black regions.

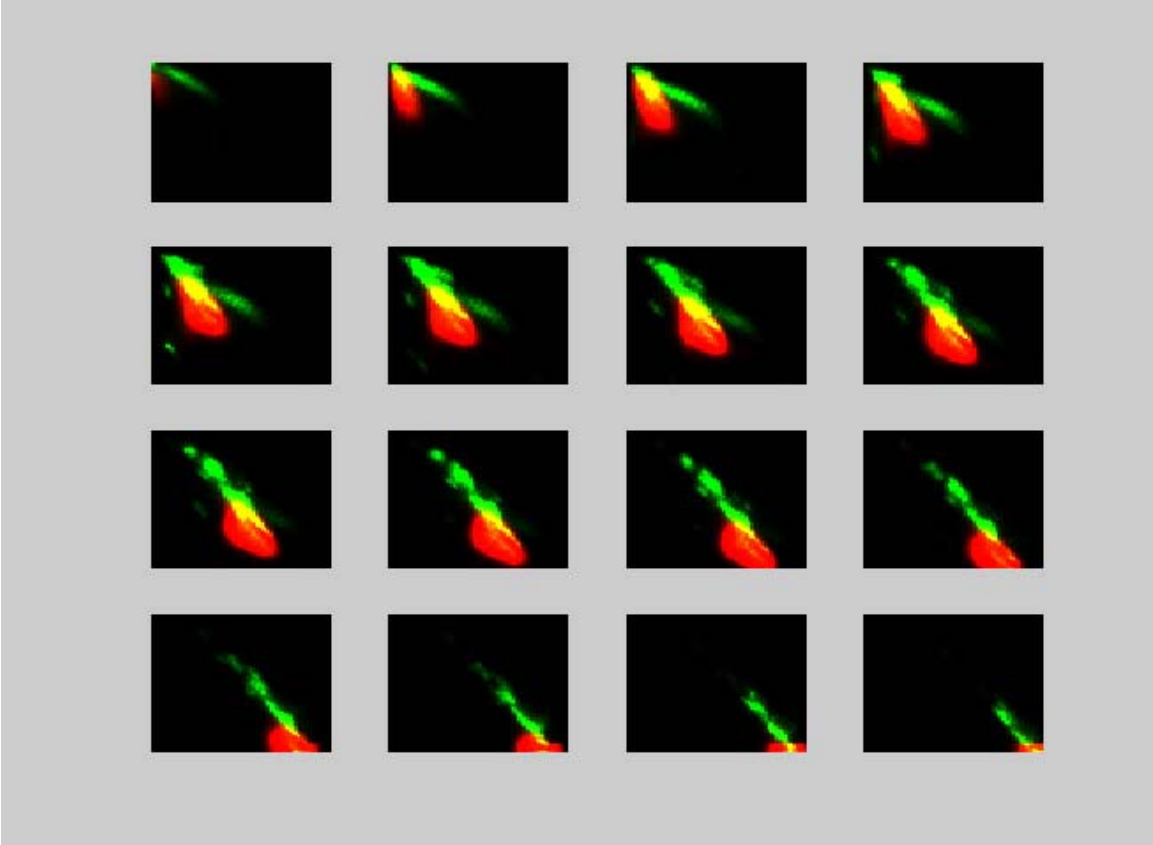

Figure 3: Transverse slices through 3-D Color RGB color space show in Fig. 2

For each location in RGB space, we use the following formula to calculate the associated probability of being a face:

Probability = (# of face pixels) / (# of face pixels + (# of background pixels) ^ weight)

The weight is used to compensate for the fact that there are many more background pixels than face pixels in the training images. It also provides a simple way to bias the function towards finding or rejecting background pixels. For example, a low weight will assign a higher probability to face pixels also in the background region, but will also cause pixels in the background to have a higher probability. Weight values of 0.6 to 0.8 were found to work quite well.

Figure 4 shows transverse slices through the resulting 3-dimensional probability function. For every RGB value, we can assign a probability between 0 and 1. This pdf is then smoothed by a 3-d box kernel in order to reduce sensitivity to specific lighting conditions. Fortunately, this pdf can be computed once and then just loaded from a file.

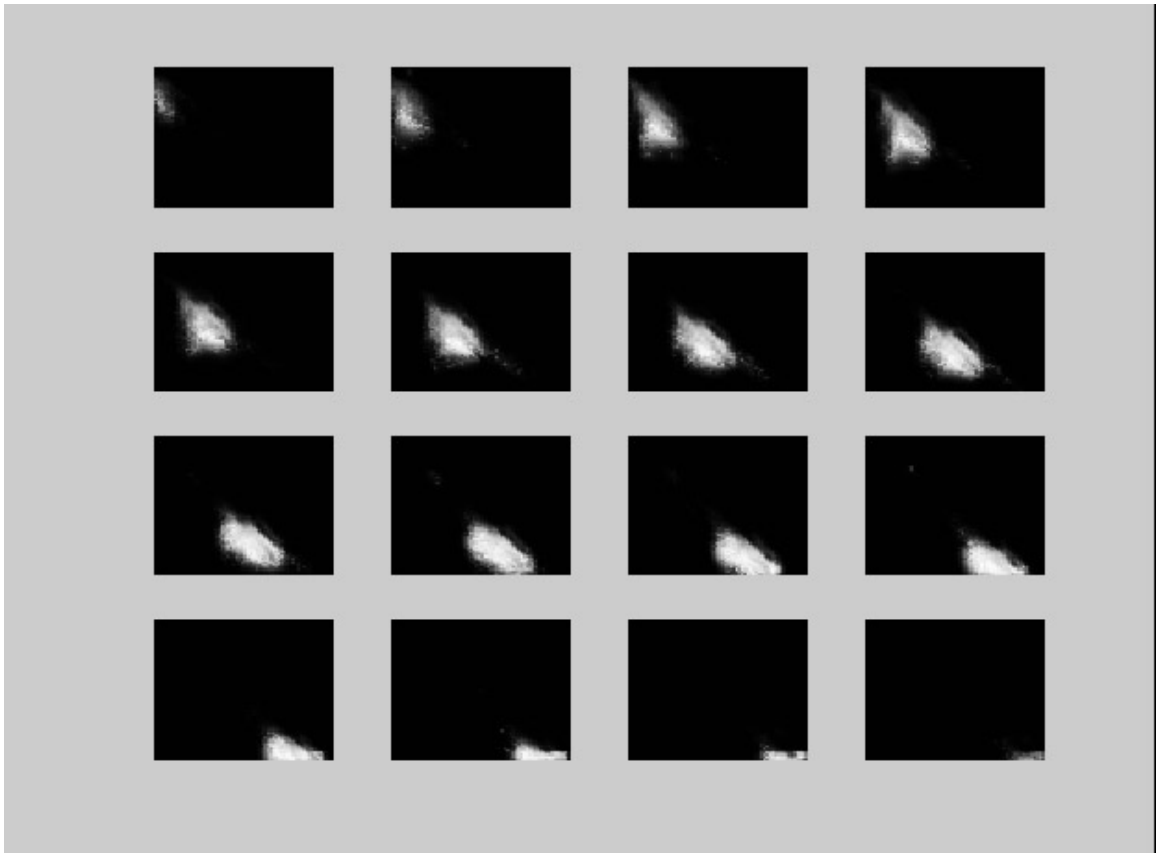

Figure 4: Transverse slices through 3-D probability function for a given specific image

Using this probability function, we look up the probability value for every pixel in the given image. Figure 5 shows the results of this operation:

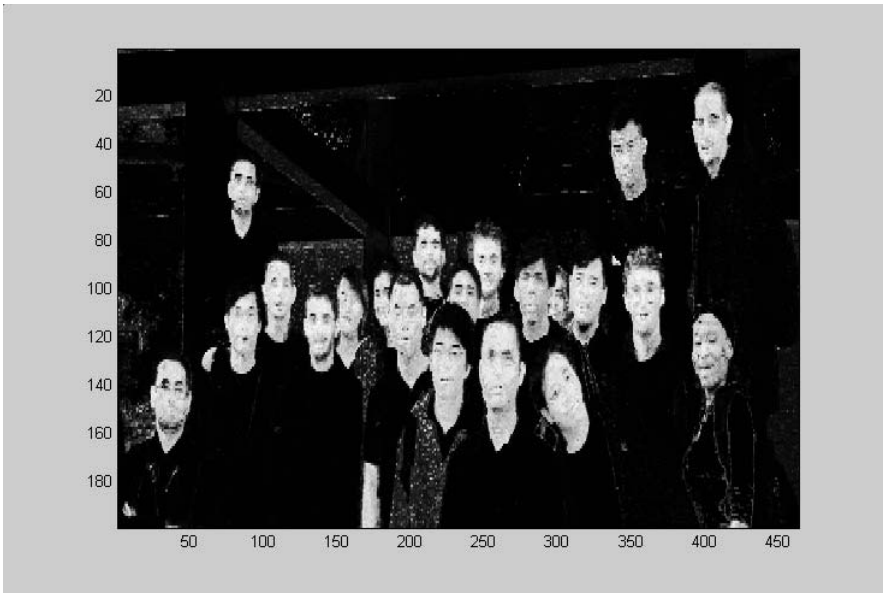

Figure 5: Image showing the probability that a given pixel is a Face-Pixel. White is higher probability.

Clearly, the faces are assigned high probabilities, while the background is assigned medium to low probabilities. We can now exploit the high spatial coherence of the faces, by convolving with ovals approximately the same size of a face. Convolving with an oval followed by thresholding results in the mask shown in Figure 6.

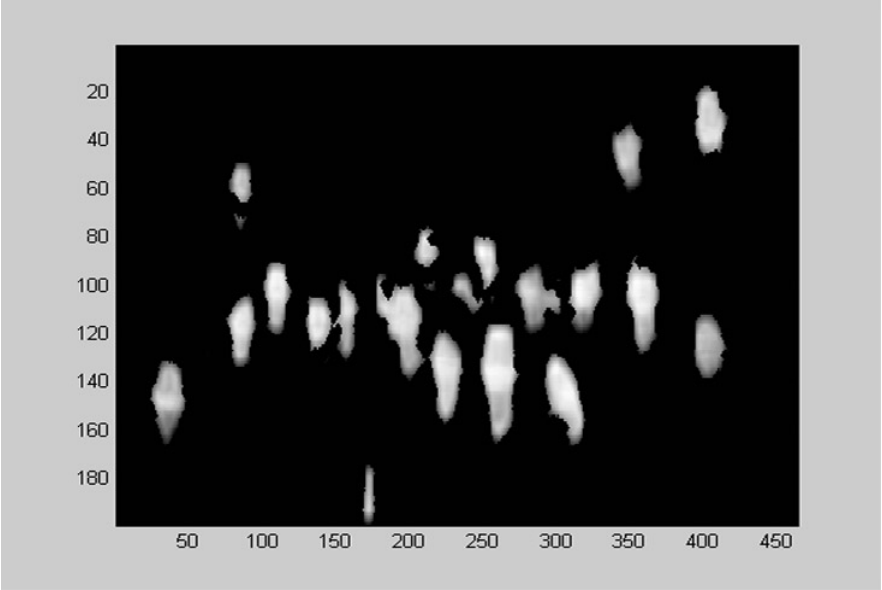

Figure 6: Mask that results from filtering probability image in Figure 5 with an oval. The result is then thresholded to get the above mask.

At this point, we have created a mask which reduces the number of pixels which must be searched for faces by at least an order of magnitude.

4. Region Finding and Separation

4.1 Region Finding:

Once we have a mask showing potential faces, we need to split this mask up into regions which can be searched for faces. After some very basic erosion and hole-filling steps, most of the faces are nicely contained in a single contiguous set of pixels. These sets can easily be found and labeled. Unfortunately, some contiguous regions contain more than one face. Ideally, these regions could be separated so that the more advanced algorithms need only output a yes/no answer rather than counting the number of faces in a region.

Figure 7 shows an example of such a region which requires separation. While it may seem possible to separate these regions by further erosion, over-erosion causes some of the smaller or obstructed faces to disppear entirely. Therefore, we need a better way of separating connected regions into single face-shaped regions.

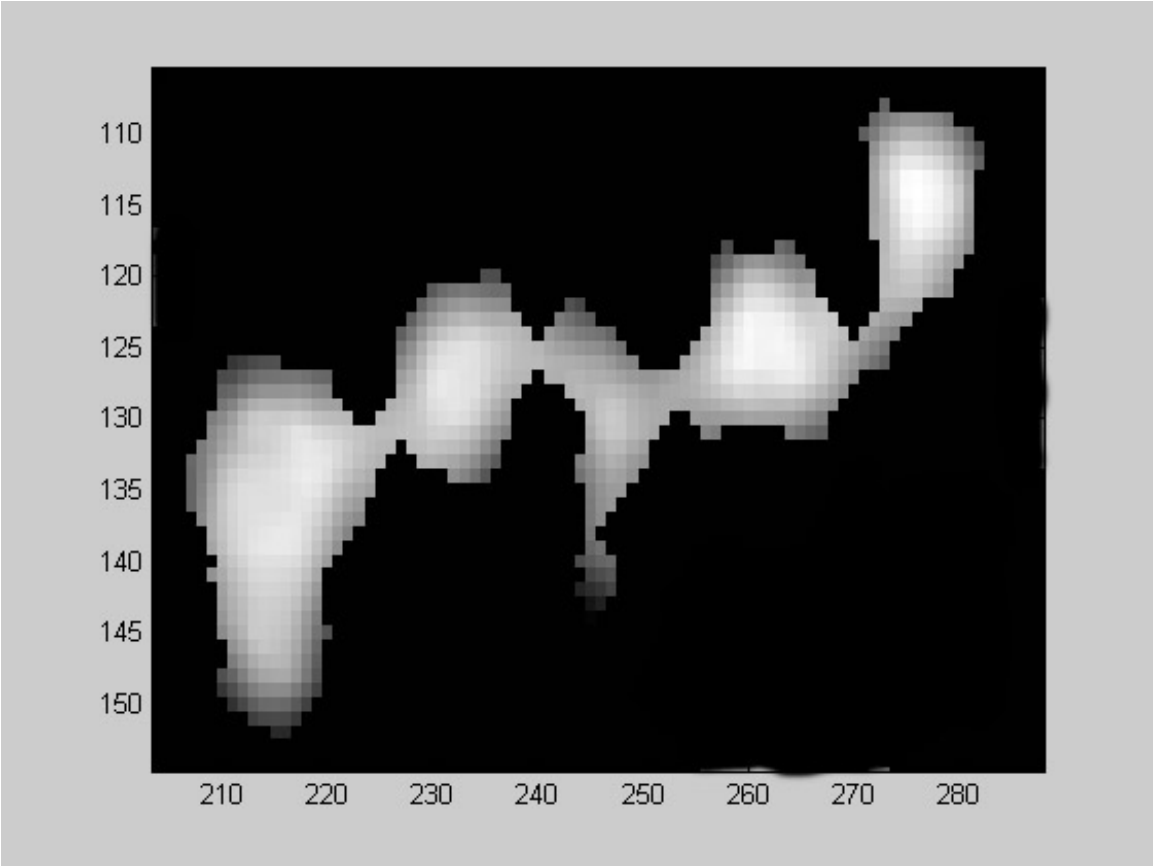

Figure 7: Example of several connected faces in the mask image, even after performing erosion and dilation to separate.

4.2 Region Separating

The fundamental problem is that the faces come in such different sizes. Some faces are very large and include necks, while other are obstructed or are very small.

We use the following algorithm to separate the regions:

1. Convolve the mask with a large head-and-neck shaped template (examples are shown in Figure 8)

2. Find the peak value resulting from the convolution, and subtract a dilated version of the template from the location of the peak value.
3. Repeat steps 1 and 2 until the peak value falls below a certain threshold
4. Repeat steps 1 to 3 with smaller and smaller head-and-neck shaped templates.

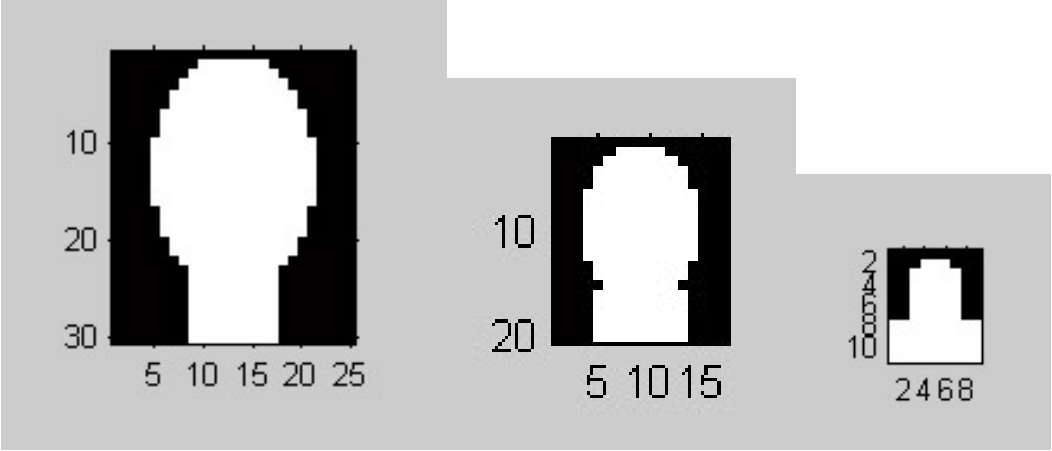

Figure 8: Three different sizes of our "head-and-neck" shaped template consist of an oval as the head and a square as the neck.

This algorithm sequentially removes the head shapes from the mask until there are none remaining. It uses templates of many sizes and is therefore insensitive to size differences in the heads, and because it removes large heads first, it results in very few "double detections".

Adjusting the thresholds downward results in more false-detections (i.e. more hands, arms, etc…), but also results in better detection of obstructed faces. Ideally, algorithms later in the chain would be capable of rejecting the false-detections.

Figure 9 shows the results of this region separation algorithm. It detects both large and small heads, and in this particular case results in a perfect score, even before doing any actual face detection.

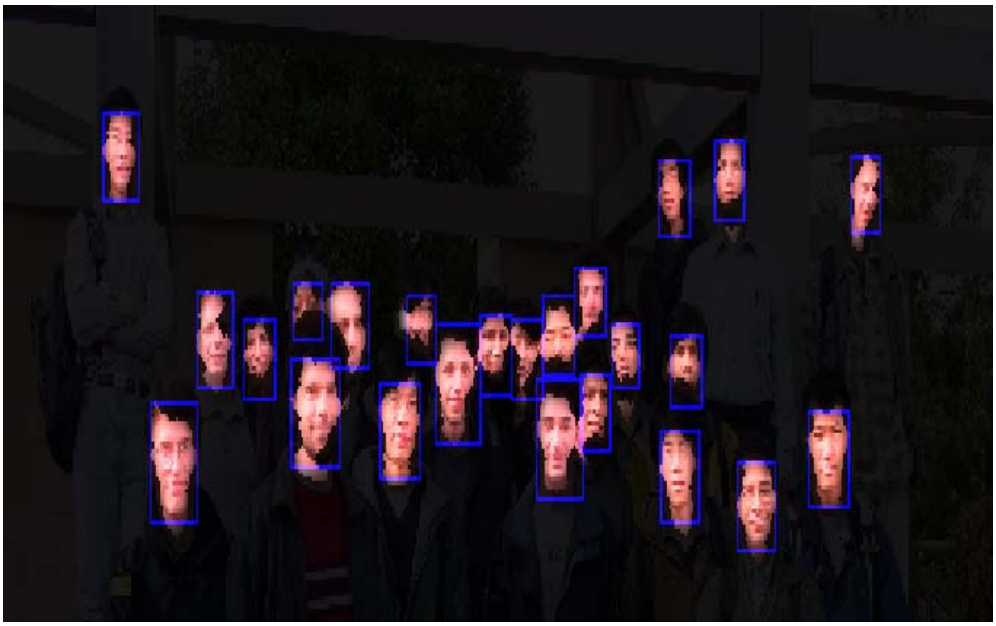

Figure 9: Results of the Region Separation Algorithm. Note for this example that all faces were properly segmented.

5. Maximum Rejection Classification

The reader is referred to [1] for the detailed theory behind MRC and its application to face detection. The lectures and slides given by Michael Elad in class [2] were also helpful to our understanding of MRC. In this section we will describe the basic details of MRC and our implementation of it for this project.

5.1 Theory-Training:

We are given two sets, the non-face image set {Yk} and the face image set {Xk}. The non-face image set is assumed to be much larger than the face set. The images are assumed to be gray-value images. The images can be projected onto a one-dimensional line by a kernel θ, which is easily found from the image set statistics The projection of the face set will occupy a certain range d1 and d2 on the line, while the projection of the non-face set will occupy another range (hopefully much larger). The goal of this projection is to minimize the number of non-face images projected onto the [d1,d2] range. This eliminates as many non-face images as possible while keeping all the face images. The remaining images (complete face set and non-face images that weren't rejected) are then used to find another θ to project onto, where a new [d1,d2] is found to reject as many non-faces as possible. The iterations are repeated and more θ's are found until practically all of the non-face images have been eliminated.

The calculation of θ is very simple and only requires finding the mean (Mx and My) and covariance matrices (Rx and Ry) of each image set:

$$Mx = \frac{1}{N_x}\sum_{i=1}^{Nx} X_k$$

$$My = \frac{1}{N_y}\sum_{i=1}^{Ny} Y_k$$

$$Rx = \frac{1}{N_x}\sum_{i=1}^{Nx} (X_k - M_x)(X_k - M_x)^T$$

$$Ry = \frac{1}{N_y}\sum_{i=1}^{Ny} (Y_k - M_y)(Y_k - M_y)^T$$

The objective function can be written as

$$f\{\theta\} = \frac{\theta^T \{[M_x - M_y][M_x - M_y]^T + Rx + Ry\}\theta}{\theta^T R_x \theta}$$

The goal is to maximize the above function, which has the effect of maximizing the distance between the PDFs of the face set and non-face set. Taking the derivative of the objective function and setting to zero we find:

$$\{[M_x - M_y][M_x - M_y]^T + Rx + Ry\}\theta = \lambda Rx\theta$$

This equation is of the form $R\theta = \lambda Q\theta$, which is a generalized eigenvalue problem. θ is then the eigenvector that corresponds to the largest eigenvalue.

5.2 Theory-Detecting Faces:

Once an adequate number of θ's and their corresponding [d1,d2] ranges have been found (for our algorithm we acquired 10 θ's), we apply the kernels to a test image to find faces. The basic idea is to go through the image taking consecutive blocks of a certain size (same as all the data set images). The first θ is applied to the block, and we check if the projection falls within the specified [d1,d2] range. If it does, then we apply the second θ and check if the projection falls within the corresponding [d1,d2]. As long as the block keeps satisfying the [d1,d2] range for each kernel, we apply the next θ. If we run through all the kernels and the

block satisfies every one, the block is classified as a face. If for any θ the block falls outside the "face" range, the block is discarded and classified as non-face. This is done for every block in the image. The key here is that we must work with multiple resolutions of the image. Since the training data is of a certain size (say 15x15), it is geared to find faces that are of the same size. In a given image, the faces could be of all different sizes, so faces will be found on different resolutions of the input image. Elad recommends working with resolution ratios of 1:1.2.

5.3 Implementation Details:

We chose a block size of 15x15 for our training data as recommended in [1]. We gathered the face set from the seven provided training images. We selected each face in Matlab using a program we wrote which had the user click on the face in the training set; the program would then cut the face off from just above the eyebrows to just above the chin and resize the face to 15x15. We assumed that the forehead and chin area probably wouldn't add much information to a face; most of the detail is in the eyes, nose and mouth. From the training images we gathered 164 faces. We then flipped each one across the vertical axis to generate 164 more faces for a total of 328 faces. In order to represent a larger range of faces, we performed a procedure called regularization, adding a value of $\sigma^2 I$ (with $\sigma = 20$) to the face covariance matrix [3].

We gathered our non-face images from Training Image 4 (masked with the ground truth data to get rid of the faces). We wrote a program that went through the image and took successive blocks as non-face images. We then went to a lower resolution and repeated the same process. We actually started with an image that was one-third resolution of the original and then went through five additional lower resolution versions to generate our data set. The reason we started with a lower resolution image is that faces in the training images were all at least 50x50 pixels, so when finding faces we probably wouldn't start looking for faces until we down-sampled the input image by at least three (since our face images are only of size 15x15). Thus we only care about non-face images that are on the same scale. We were able to generate 618,988 non-face images. Ideally we would want our non-face image set to be as large as possible to cover every possible case of a non-face. However, generating these non-face sets took a long time in Matlab, and an even bigger problem was storing the images. It wasn't practical to generate a matrix to store every image, so we kept track of blocks by their positions in the training image.

There were a couple of very helpful pre-processing procedures that we performed on the input blocks before classifying them as face/non-face. The first was removal of the mean from the block. This processing helped to deal with different skin colors and lighting on faces. The second procedure was masking out some of the pixels in the input block, namely the bottom left and right corners. This removed some of the background pixels that could be found in the face block. It turns out that these procedures can be applied to the kernel itself instead of each block, thereby saving innumerable calculations.

5.4 Advantages and Disadvantages

One of the advantages of MRC is that it uses simple (and therefore fast) classifiers. The classifiers are weak so we need to use many of them, but this isn't a problem because each runs through the algorithm quickly. MRC is simple to implement once the classifiers and thresholds have been acquired. It is also very accurate.

One problem with MRC is that if given a very large input image, traversing it with 15x15 blocks at several resolutions could take a long amount of time. Another logistical problem lies in combining the results from multiple resolutions. If you find a face on one resolution, then find it again on another resolution, how to know if you found a new face or if it's the same one? This problem is actually not difficult to solve. The biggest problem with MRC is that it obviously will have some false detections. However, when we combined MRC with the color segmentation/region finding algorithm we were able to severely cut down on these false detections

5.5 Integration of MRC and Color Segmentation/Region Finding:

We use color segmentation/region finding (from now on referred to as just color segmentation) to segment the input image into different blocks; the color segmentation algorithm "claims" each selected block as

having one face. We then take each of these selected blocks and perform MRC to determine if there really is a face in the block or maybe it's just a flesh-colored orange. The MRC works with the block in five different resolutions to find a face; once it finds a face in the block at a certain resolution, it stops and labels the block as a face. It then moves on to the next block selected by color segmentation and does the same thing until all the selected blocks have been processed.

The benefits of integrating MRC and color segmentation are numerous. Doing color segmentation then MRC saves time because the MRC algorithm no longer has to go through every successive block in the input image. Instead it only has to go through the 30 or so blocks (assuming 20-25 faces in image) identified by the color segmentation. Another advantage is that MRC can get rid of false detections from the color segmenting. A flesh colored shirt that is segmented looks nothing like a face to MRC and is eliminated. We don't have to worry about MRC false detections since the color segmenting gets rid of so much of the input image, decreasing the chances of a false detection (that would have occurred if MRC had to traverse the entire image). Yet another benefit is that now we don't have to worry about finding the same face at multiple resolutions. We know that there's either one face or no faces in the selected block. So once we find a face, there's no need to continue processing at lower resolutions, so we can continue with the next color-segmented block. The input blocks into the MRC algorithm never contained more than one face for the test images we ran, so we don't have to worry about multiple faces per block.
The only real problem with this integration is that sometimes there really is a face in the block that color segmentation identifies, but MRC fails to find it. This often occurs if the face is partially obstructed or severely rotated. However, we were willing to not find these rare cases in order to avoid the false detections that MRC alone or color segmentation alone would have detected.

Figure 10 below shows the input color-segmented and region separated images that are input to the MRC algorithm. Figure 11 shows the output of the MRC algorithm. The missing images (at locations (2,3), (6,2), and (6,3) are not present in the output) are those that MRC did NOT classify those as faces. This was a perfect detection image.

Figure 10: Example of face candidates passed from Region Separating Algorithm to the MRC algorithm. Note the non-faces at positions (2,3), (6,2), and (6,3)

Figure 11: Output of our MRC Algorithm. Note the non-faces from Figure 10 have been removed, but nothing else has been. This is perfect detection.

6. End Processing

After the MRC algorithm makes the final face selection, we perform two more steps in our algorithm.

First, we take the results of the MRC and if two faces are too close together, we eliminate the second face. We found that the previous steps of our algorithm can sometimes double count certain faces, and that this technique successfully eliminated those extra faces.

Second, we tag each face that the MRC algorithm finds with its average Value (from HSV color space). Instead of doing gender recognition, we only search for one specific female in the set of detected faces. The face with the lowest Value from HSV space is labeled as a female. One of the three females in the class has much darker skin than anyone else, so she usually has the lowest average Value in HSV color space, depending on lighting conditions.

7. Other Methods Considered

Before settling on a final algorithm we also considered template matching as a simpler approach to the MRC algorithm. We created a template by averaging together all the faces across all seven training images. Prior to averaging the faces together we normalized the distance between the eyes so that each face was approximately the same size and features such as the eyes, nose, and mouth lined up. Our template can be seen in Figure 12 below.

Figure 12: Template made from averaging all faces in test images

Using a single template to try to match faces from the output of the region finding and separation step actually gave lower scores than doing nothing. To improve upon this we scaled the template to several different resolutions, which gave much better results than before. However, there were still several false positives and negatives in each image. Other improvements that we made included constraining the peak of the correlation to be near the center of the input image. Also, if template matching produced results that didn't have a well-defined peak but instead just a large high-valued area, we discarded those results too. High-valued correlations without a well-defined central peak were typically due to arms, necks, or other non-faces.

8. Results

Testing our algorithm on each of the training images provided to us gives the results as seen in Table 1.

Table1: Results from running each test image through our algorithm

Image #	#Faces Detected	#Faces in Image	Percentage Correct	# Repeated Faces and False Positives	Bonus
1	20	21	95%	0	1
2	23	24	96%	0	1
3	25	25	100%	0	0
4	23	24	96%	0	0
5	21	24	88%	0	0
6	23	24	96%	0	0
7	22	22	100%	0	0

As can be seen from the table, our algorithm performs quite well. Two images have a 100% face detection rate, while all but one image have above a 95% face detection rate. Also of note is that we have no repeated faces or false positives in any of the training images. Furthermore, we were able to correctly identify one female in two of the test images.

9. Acknowledgements

We would like to especially acknowledge Michael Elad for his help with the MRC algorithm

10. References

[1] Michael Elad, Yacov Hel-Or, Renato Keshet, “Rejection Based Classifier for Face Detection”, Hewlett Packard Laboratories, Israel, Pattern Recognition Letters, April 2001.
[2] Michael Elad, EE368 Lecture, 19 May 2003 and 21 May 2003.
[3] Frederic Sarrat, “Rejection-based Linear Classifier”, Spring 2002 EE 368 Project Report.

Other:
Gonzalez and Woods, *Digital Image Processing*, Prentice Hall, New Jersey, 2002.