Questions may continue on the back. Please write clearly. What I cannot read, I will not grade.

1. The two files skin.txt and nonskin.txt accessible through the homework web page contain histograms of the red, green, and blue color components of a large number of pixels. These files were made available by the authors of the following paper


The pixels in skin.txt are from image regions that depict human skin, and those in nonskin.txt are from image regions without skin. In this assignment, assume that 20 percent of all pixels in an image collection are skin pixels.

The function in the file readHistogram.m can be used to read the two files above with the following calling syntax:

```matlab
sigma = 1.5;
rgbS = readHistogram('skin.txt', sigma);
rgbN = readHistogram('nonskin.txt', sigma);
```

Let \( h \) represent either \( rgbS \) or \( rgbN \). Then the value \( h(r, g, b) \) is the number of pixels with color components

\[
\text{Red} = C(r); \quad \text{Green} = C(g); \quad \text{Blue} = C(b);
\]

where

\[
C(r) = \text{linspace}(0, 1, H);
\]

and \( H = 32 \) (search the Matlab Help facility for \linspace\ if you are not familiar with this function, or just try it out).

This means that all pixel values are assumed to be between 0 and 1, and that the histograms bin these values into 32 bins in each of the three color components.

It is difficult to visualize three-dimensional distributions, so we will also refer to the marginal histograms

\[
\text{rgS} = \text{squeeze}(\text{sum}(\text{rgbS}, 3));
\text{rgN} = \text{squeeze}(\text{sum}(\text{rgbN}, 3));
\]

obtained by integrating out the blue channel.

Since you will work with distributions in either two and three dimensions, it is best for your code to be general in this respect. Code that works only in two or three dimensions is acceptable as well (but not necessarily easier to write, and definitely harder to test and debug).

(a) Write a function with header

```matlab
function p = joint(hp, hn, pp)
```

that takes a histogram \( hp \) for “positive” pixels (that is, skin pixels), a histogram \( hn \) for “negative” (non-skin) pixels, and the prior probability \( pp \) for the “positive” category, and returns a properly normalized, joint probability distribution \( p \) on observations and classes. Your function should work at least for two- and three-dimensional histograms, so that both following calls will work (with \( ps = 0.2 \)).

```matlab
p2 = joint(rgS, rgN, ps);
p3 = joint(rgbS, rgbN, ps);
```

The array \( p2 \) will have dimensions \( H \times H \times 2 \), and the array \( p3 \) will have dimensions \( H \times H \times H \times 2 \). Hand in your code and the plots of \( \text{squeeze}(p2(:, :, 1)) \) and \( \text{squeeze}(p2(:, :, 2)) \). Check that normalization is correct.

(b) Write a function with header

```matlab
function px = compose(p, x)
```

that takes a measure \( p \) (such as slice \( \text{squeeze}(p2(:, :, k)) \) or \( \text{squeeze}(p3(:, :, :, k)) \)) and an array of values \( x \) with \( \text{size}(x, 1) == d \), the dimensionality of observation space (2 or 3 in our examples). The result in \( px \) contains the result of evaluating \( p \) at all points in \( x \). If \( \text{xs} = \text{size}(x) \) then \( \text{size}(px) \) is \( \text{size}(x)(2:end) \). As before, all values of all entries in \( x \) are assumed to be between 0 and 1.

A slight programming challenge here is that we only know that the first dimension of \( x \) is \( d \). Leaving the other dimensions of \( x \) as unknown affords generality, so we can call \( \text{compose} \) on a single vector \( x \), an image, a table of values, and so forth. [Hint: use linear indexing. An array of any number of dimensions can be indexed with a single scalar integer. For instance, \( a(n) \) is the \( n \)-th item in \( a \), in the order in which items are stored in memory (first index changes fastest). So \( x(1:d) \) is the first point in \( x \), \( x((d+1):(2*d)) \) is the second, and so forth.]

Hand in your code, and, as a sanity check for me to use, the result of
compose([1 2 3; 4 5 6], [0.1 0.7; 0.3 0.2; 0.6 0.3; 0.9 0.9]')

(you should obtain a column vector of four numbers).

(c) Write a function with header

```matlab
function [label loss] = classify(x, L, p)
```

that takes an array `x` of observations as in the previous question, a $K \times K$ loss matrix `L` (your code should work at least for $K = 2$), and a probability density `p` such as the ones returned by your function `joint`. The output `label` should contain the results of classifying the observations in `x` with the classifier that yields the minimum expected loss, as described in Section 1.5.2 in Bishop. The output array `loss` should contain the expected loss for each value of `x`, that is, the quantity described by formula (1.81) in Bishop, but with posterior probabilities $p(C_k | x)$ replaced with joint probabilities $p(x, C_k)$.

Just as for `px` in the previous problem, if `xsize = size(x)`, then both `size(label)` and `size(loss)` are equal to `xsize(2:end)`.

Hand in your code, and plots of `label` and `loss` for the joint distributions obtained in the two-dimensional case (that is, starting with the histograms `rgS` and `rgN`), and with the loss matrix

```
L = [0 1; 1 0].
```

To obtain these plots, create a grid `x` (of size $2 \times H \times H$) with all the values in the red-green color space as follows:

```matlab
psize = size(p);
d = ndims(p) - 1;
[u v] = ndgrid(linspace(0, 1, psize(1)), linspace(0, 1, psize(2)));
x = zeros([d psize(1:(end-1))]);
x(1, :, :) = u;
x(2, :, :) = v;
```

Then classify the resulting values:

```matlab
[label loss] = classify(x, L, p);
```

(d) The file `normalizedRead.m` available through the homework web page contains the definition of a function that reads an image from a file and normalizes its pixel values to $[0, 1]$. Use this function to read the images made available on the homework web page.

Classify the pixels in each image with your function `classify` and the loss matrix given earlier. For each image, you should do this with the red-green and also with the red-green-blue distributions. Display each classification result as a new image that is black (or white, if you prefer) where the classifier found non-skin pixels. The color of each skin pixel in the result image should be that from the original image.

Hand in the results as a table of images: each row should have the input image, the classification result with the red-green distribution, and the classification result with the red-green-blue distribution.

(e) Comment on your results: Is your classifier usable for real applications (if so, which applications)? Can it be improved substantially (if so, how, and how far)?