(2) Histogram Equalization

(a) HIST_EQ.m function created to equalize an input image. This was done in a manner in which the CDF values [0 1] of the histogram replace the original image pixel intensity values. This is then normalized to the number of bins input to the function and again to a full [0 255] grayscale intensity map. Code is shown in Appendix A.

(b) Images and Histograms are shown in Figure 1 and Figure 2.

(c) HIST_EQ.m was performed on 32x32 blocks within the jetplane image. The M-file blockproc.m was not available my current version of MATLAB so I wrote a simple looping function to accomplish the same. The resulting image is shown in Figure 3.

Figure 1: Histogram equalized images using various number of intensity bins.

Figure 2: Histograms of equalized images (Original histogram, 256, 128, and 64 equalized bins).

Figure 3: Jetplane image using histogram equalization on [32x32] blocks.
(3) Basic Morphology

(a) By cropping the faces of players from the pictures provided, I converted the small face image to HSV. I then took a histogram of the hue and saturation of the cropped image to find a collection of ranges for “skin color” thresholds. I then compared each pixel to this threshold to decide whether or not it was a skin pixel. This was transferred to the binary images shown in Figure 1.

Results showed that detection was affected by variations in skin tone as well as lighting and anything that would affect the overall coloring of the face. Also, it was difficult to discern between face and other exposed skin area, for instance the futbol players' knees and arms were often detected as faces.

(b) To clean the detected image, bwmorph.m was used. Various orders and repetitions of morphology were attempted until I found what seemed to be the best for skin tone detection. The resulting images are shown in Figure 4.

(c) The cleaned images used bwlabel.m to label the different objects (areas of skin). Next, regionprops.m was used to find the area and boundaries of the objects. From this data, rectangles were drawn around the objects, as shown below. The first image detected all 11 faces but had 9 extra detections on body parts. The second image detected 8 faces (one of which was only partial) and added 11 additional detections on body parts. Code is shown in Appendix B.

Figure 4: (a) Threshold images of “skin color” pixels. (b) Cleaned images using morphology for object detection and labeling. (c) The output image with bounding boxes around faces.
(4) Filtering

(a) Noise was added to the image using imnoise.m. The resulting noisy image was filtered with box filters and median filters ([3x3] and [7x7]). The quantitative results are shown in Table 1. Visually, it seems the [3x3] size did well in both filters, though the median filter seem to maintain crisper edges than the averaging filter.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>242.07</td>
<td>439.45</td>
<td>300.59</td>
<td>453.97</td>
</tr>
<tr>
<td>PSNR (dB)</td>
<td>24.33</td>
<td>21.74</td>
<td>23.38</td>
<td>21.59</td>
</tr>
</tbody>
</table>

Table 1: Results of filtering gaussian white noise.

(b) Next, salt & pepper noise was added and the filtering repeated. Results were similar. The quantitative results are shown in Table 2. Visually, it seems the [3x3] size did well in both filters, though the median filter seem to maintain crisper edges than the averaging filter.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>242.07</td>
<td>439.45</td>
<td>300.95</td>
<td>453.97</td>
</tr>
<tr>
<td>PSNR (dB)</td>
<td>24.33</td>
<td>21.74</td>
<td>23.38</td>
<td>21.59</td>
</tr>
</tbody>
</table>

Table 2: Results of filtering salt and pepper noise.

(c) Running the filtering with the image at a higher ISO had similar results. The time it took to do the [3x3] median filter (0.1070 s) took nearly twice as long as the [3x3] box filter (0.0638 s). Code is shown in Appendix C.
Appendix A:

MATLAB Code for problem 1.2

```matlab
% HIST_EQ: Returns histogram equalized image

% EQ_image = hist_eq(im,levels)
% IM = grayscale intensity image
% levels = number of gray level value bins

function EQ_image = hist_eq(IM,levels)

% number of intensity levels of original image
bins = 255;

values = reshape(IM,[],1);
% histogram of grayscale intensity values
H = hist(values,[0:bins]);
% conversion to probability [0 1]
prob = H/numel(IM);
% calculate cumulative distribution
cumDF = cumsum(prob);
% replace original image values with those from CDF on scale [0 1]
EQ_image = cumDF(IM+1);
% normalize to number of bins
EQ_image = floor(EQ_image*levels);
% normalize to [0 bins] grayscale
EQ_image = EQ_image*(bins/levels);

EQ_image = uint8(EQ_image);
```
Appendix B:

MATLAB Code for problem 1.3

% HW1 Prob 3

clear all; close all; clc

% RGB = imread('SJEarthquakesteampic.jpg');
RGB = imread('barcelona-team.jpg');
HSV = rgb2hsv(RGB);

[r c depth] = size(HSV);

faceDet = zeros(r,c);

for i = 1:1:r
    for j = 1:1:c
        H = HSV(i,j,1);
        S = HSV(i,j,2);
        if H > 0.01 & H < 0.12
            if S > 0.1 & S < 0.6
                faceDet(i,j) = 1;
            end
        end
    end
end

M = faceDet;
M = bwmorph(M,'erode',2);
M = bwmorph(M,'majority',3);
M = bwmorph(M,'erode',2);
M = bwmorph(M,'dilate',12);

figure
imshow(faceDet)
figure
imshow(M)
lbl = bwlabel(M);
stats = regionprops(lbl);

figure
imshow(RGB)
hold on

for i = 1:length(stats)
    pos = stats(i).BoundingBox;
    rectangle('Position',pos,'EdgeColor','r','LineWidth',2)
end

disp('number of faces detected = ');
disp(length(stats))
Appendix C:

MATLAB Code for problem 1.4

% HW1 Prob 4

clear all; close all; clc

% read in image
%IM = imread('DSCN0479-001.jpg');
IM = imread('DSCN0482-001.jpg');

% add noise
%noiseIM = imnoise(IM,'gaussian',0,0.005);
noiseIM = imnoise(IM,'salt & pepper',0.05);

% create filter
filtSize = [3 7];
box1 = (1./(filtSize(1).^2)).*ones(filtSize(1));
box2 = (1./(filtSize(2).^2)).*ones(filtSize(2));

t = zeros(1,4);

% filter image with created filter
for i = 1:1:3
    tic;
    outImage1(:,:,i) = filter2(box1,IM(:,:,i));
t(1) = t(1)+toc;
tic;
    outImage2(:,:,i) = filter2(box2,IM(:,:,i));
t(2) = t(2)+toc;
tic;
    outImage3(:,:,i) = medfilt2(IM(:,:,i),[3 3]);
t(3) = t(3)+toc;
tic;
    outImage4(:,:,i) = medfilt2(IM(:,:,i),[7 7]);
t(4) = t(4)+toc;
end

outImage1 = uint8(outImage1);   % [3x3] box filter
outImage2 = uint8(outImage2);   % [7x7] box filter
outImage3 = uint8(outImage3);   % [3x3] median filter
outImage4 = uint8(outImage4);   % [7x7] median filter

% calculate MSE
err1 = double(IM) - double(outImage1);
MSE1 = sum(sum(sum(err1.^2)))/numel(IM);
err2 = double(IM) - double(outImage2);
MSE2 = sum(sum(sum(err2.^2)))/numel(IM);
err3 = double(IM) - double(outImage3);
MSE3 = sum(sum(sum(err3.^2)))/numel(IM);
err4 = double(IM) - double(outImage4);
MSE4 = sum(sum(sum(err4.^2)))/numel(IM);

% calculate PSNR
PSNR1 = psnr(IM,outImage1);
PSNR2 = psnr(IM,outImage2);
PSNR3 = psnr(IM,outImage3);
PSNR4 = psnr(IM,outImage4);
% display results
fprintf('MSE: \n');
fprintf('\n\t'); disp(MSE1);
fprintf('\n\t'); disp(MSE2);
fprintf('\n\t'); disp(MSE3);
fprintf('\n\t'); disp(MSE4);
fprintf('PSNR: \n');
fprintf('\n\t'); disp(PSNR1);
fprintf('\n\t'); disp(PSNR2);
fprintf('\n\t'); disp(PSNR3);
fprintf('\n\t'); disp(PSNR4);
fprintf('Time: \n');
fprintf('\n\t'); disp(t(1));
fprintf('\n\t'); disp(t(2));
fprintf('\n\t'); disp(t(3));
fprintf('\n\t'); disp(t(4));
figure
subplot(3,2,1)
imshow(IM)
title('Original Image')
subplot(3,2,2)
imshow(noiseIM)
title('Noisy Image')
subplot(3,2,3)
imshow(outImage1)
title('Smoothed [3x3] Box Filter')
subplot(3,2,4)
imshow(outImage2)
title('Smoothed [7x7] Box Filter')
subplot(3,2,5)
imshow(outImage3)
title('Smoothed [3x3] Median Filter')
subplot(3,2,6)
imshow(outImage4)
title('Smoothed [7x7] Median Filter')