Proposed solutions to the exercises in the first exercise set in Matlab

Here you can find the proposed solutions for the first exercise set, written in Matlab.

```
close all % closes all figure windows

img1 = imread('football.jpg');
img3 = rgb2gray(img1);

h1 = ones(5,5) / 25;
img4 = imfilter(img3,h1);

figure, imagesc(img3), colormap gray, title('Original image');
figure, imagesc(img4), colormap gray, title('Filtered image');

%% Exercise 1

% a)
[Nx, Ny] = size(img3);
img4_indexed = img4(3:(Nx-2),3:(Ny-2));

figure, imagesc(img4_indexed), colormap gray, title('Borders removed, indexing technique

% b)
img4_conv2 = conv2(double(img3),h1,'valid');
img4_filter2 = filter2(h1,double(img3),'valid');

figure, imagesc(img4_conv2), colormap gray, title('Borders removed, using valid in conv2
figure, imagesc(img4_filter2), colormap gray, title('Borders removed, using valid in fil

% c)
img4_imfilter = imfilter(img3,h1,'same','symmetric');

figure, imagesc(img4_imfilter), colormap gray, title('Same-sized filtered image using im

%% Exercise 2

% see iimhist.m

hist_ = iimhist(img3);
hist_matlab = imhist(img3);

% See how iimhist performs compared to Matlab's imhist:
abs_err = abs(hist_ - hist_matlab);
max(abs_err)

bins = 1:256 ;

figure()

bar(bins, hist_, 'FaceAlpha', 0.5); % Facealpha is used to make the bars transparent.
hold on % Makes it possible to plot two plots on top of each other
bar(bins, hist_matlab, 'FaceAlpha', 0.5)

title('Histograms');
legend('iimhist','imhist')

%% Exercise 3

img2 = imread('coins.png');

% a)
thresholded1 = zeros(size(img2)); % Make a matrix of the same size as img2
thresholded1(img2 > 100) = 1;
```

```
thresholded2 = zeros(size(img2));
thresholded2(img2 < 100) = 1;

thresholded3 = zeros(size(img2));
thresholded3(img2 >= 120) = 1;

thresholded4 = zeros(size(img2));
thresholded4(img2 <= 120) = 1;

% Make subplots of the thresholded images
figure()

% The two first numbers represents the number of plots along the vertical
% axis and the horizontal axis, respectively. The last number indicates at
% which plot we are considering.
subplot(2,2,1)
imagesc(thresholded1)
colormap gray
title('Pixel values from img2 greater than 100')

subplot(2,2,2)
imagesc(thresholded2)
colormap gray
title('Pixel values from img2 less than 100')

subplot(2,2,3)
imagesc(thresholded3)
colormap gray
title('Pixel values from img2 greater or equal to 120')

subplot(2,2,4)
imagesc(thresholded4)
colormap gray
title('Pixel values from img2 less or equal to 120')

% b)
thr_otsu = graythresh(img2);

img2_otsu = im2bw(img2, thr_otsu); % or imbinarize

figure, imagesc(img2_otsu), colormap gray, title('img2 thresholded using Otsu`s method')

% c)

figure()

subplot(2,2,1)
imagesc(abs(thresholded1-img2_otsu))
colormap gray
title({'Absolute difference between thresholded1 and image','thresholded using Otsu`s me

subplot(2,2,2)
imagesc(abs(thresholded2-img2_otsu))
colormap gray
title({'Absolute difference between thresholded2 and image','thresholded using Otsu`s me


subplot(2,2,3)
imagesc(abs(thresholded3-img2_otsu))
colormap gray
```

```
title({'Absolute difference between thresholded3 and image','thresholded using Otsu`s me

subplot(2,2,4)
imagesc(abs(thresholded4-img2_otsu))
colormap gray
title({'Absolute difference between thresholded4 and image','thresholded using Otsu`s me

%% Exercise 4

h2x = [-1 -2 -1 ;  0  0  0 ;  1  2  1];
h2y = [-1  0  1 ; -2  0  2 ; -1  0  1];
resX = conv2(double(img3), h2x);
resY = conv2(double(img3), h2y);
resXY = sqrt(resX.^2 + resY.^2);

% Trick in matlab; use mat2gray to normalize the matrix between 0 and 1.
% Multiply then the matrix with 255.
resXY_norm = mat2gray(resXY).*255;

% See if the matrix has been correctly normalized:
max(resXY_norm(:)) % Gives 255
min(resXY_norm(:)) % Gives 0

% Threshold the image
T = 100;
thresholded = zeros(size(resXY));
thresholded(resXY_norm > T) = 1;

figure()
imagesc(thresholded)
colormap gray
title('Normalized img2 thresholded with T = 100')

% One could also do the normalization 'manually'.
% The normalization could be seen as a linear transform of the pixel
% intensities: normalized_img = a*img + b
% To determine the coefficients a and b, the following equations can
% be solved:
%
% 0 = a*min(img(:)) + b
% 255 = a*max(img(:)) + b
%
% which yields
% a = 255/( max(img(:)) - min(img(:)) )
% b = -a*min(img(:))
%   = -( 255 * min(img(:)) )/( max(img(:)) - min(img(:)) )
%
% and gives the following expression for the normalization
%
% normalized_img = 255/( max(img(:)) - min(img(:)) )*( img -  min(img(:)) )

resXY_norm2 = 255/( max(resXY(:)) - min(resXY(:)) )*( resXY -  min(resXY(:)) );

% Find the max error between the 'manual' normalization and normalization
% using mat2gray

diff = abs(resXY_norm - resXY_norm2);
max(diff(:)) % The maximum error is of order 10^{-14}, which practically is zero to nume
```

The program is available here (https://github.com/krisbhei/IN5520-IN9520/blob/master/1/proposed_solutions.m) (right click and press "save link as").