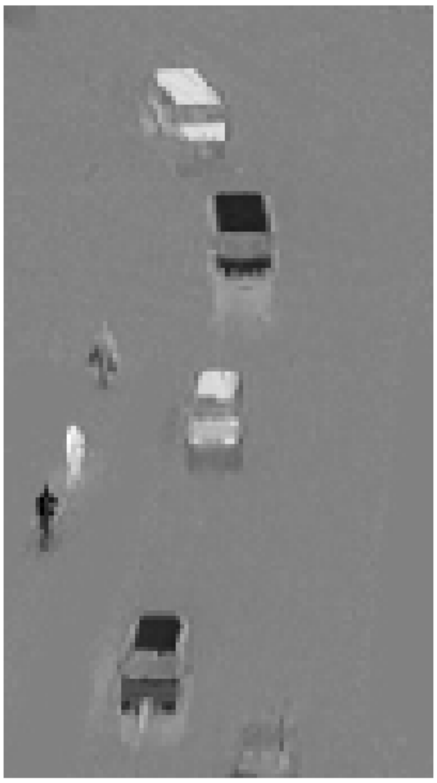
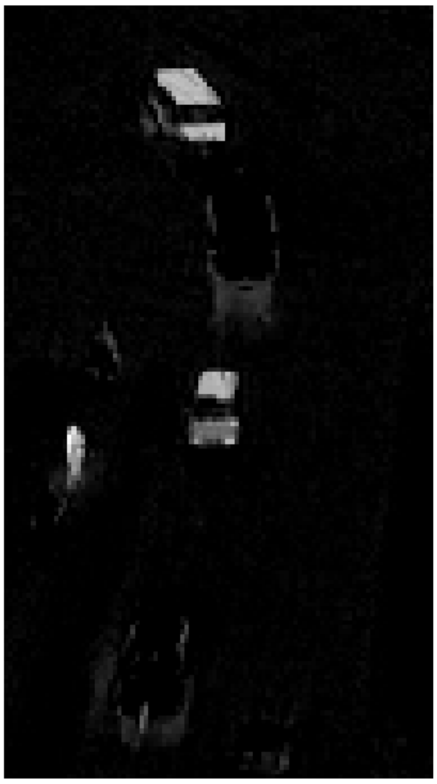
COMPUTER VISION

Answers to Exercise 1

Exercise 1

Sample output:





Exercise 2

%% Line 1: substrat the min vaule in the image

processed_im = im-min(min(im));

%% Line 2: divide the image by the max value

processed_im = processed_im./max(max(processed_im));

%% Line 1 and Line 2 together normalize the image, wherein all pixels are of value [0 1];

%% Line 3: apply a non-linear correction to the normalized image, which can be considered as soft thresholdin

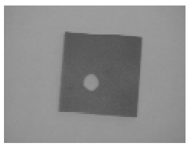
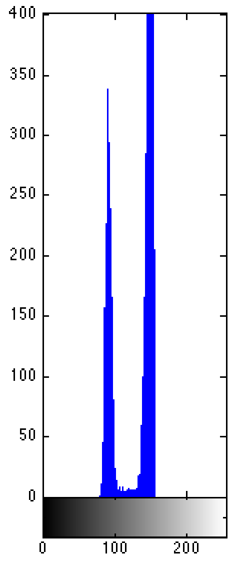
processed_im = 1./(1 + exp(g*(c-processed_im))); % Apply Sigmoid function

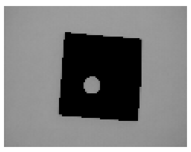
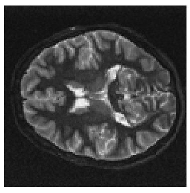
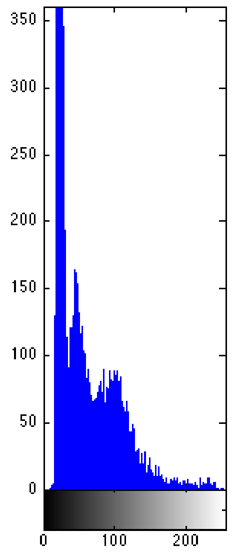
Exercise 3

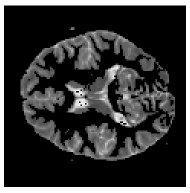
Hints for the threshold image function:

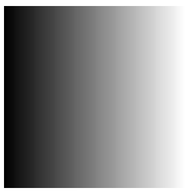
```
output_image = (input_image >= Low_thres) .* (input_image <= High_thres) .* input_image;
```

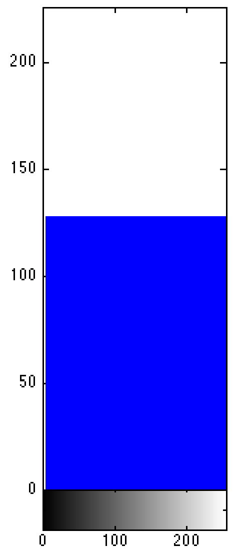
Sample outputs:

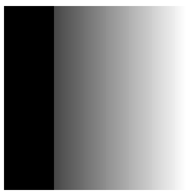









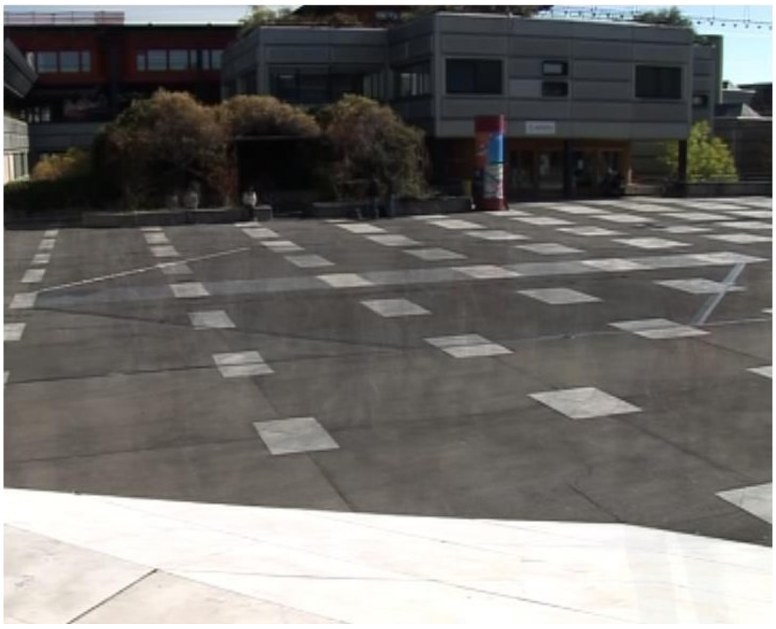


Exercise 4

1. We first get a list of all the images in the sequence1 directory.

```
img_folder = 'sequence1/';
img_ext = '*.jpg';
list_images = dir([img_folder img_ext]);
```

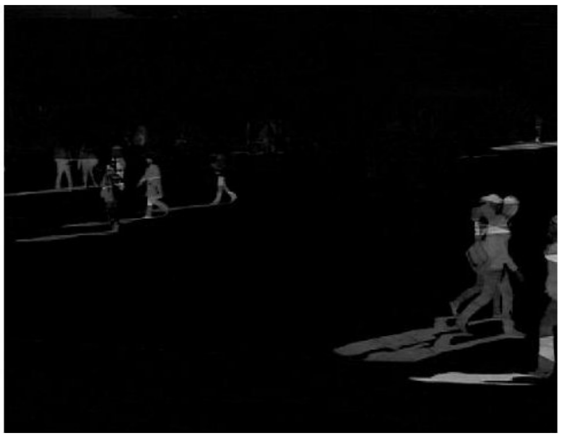
You can then load all the images in list_images using imread and average them. Here is the expected averaged i



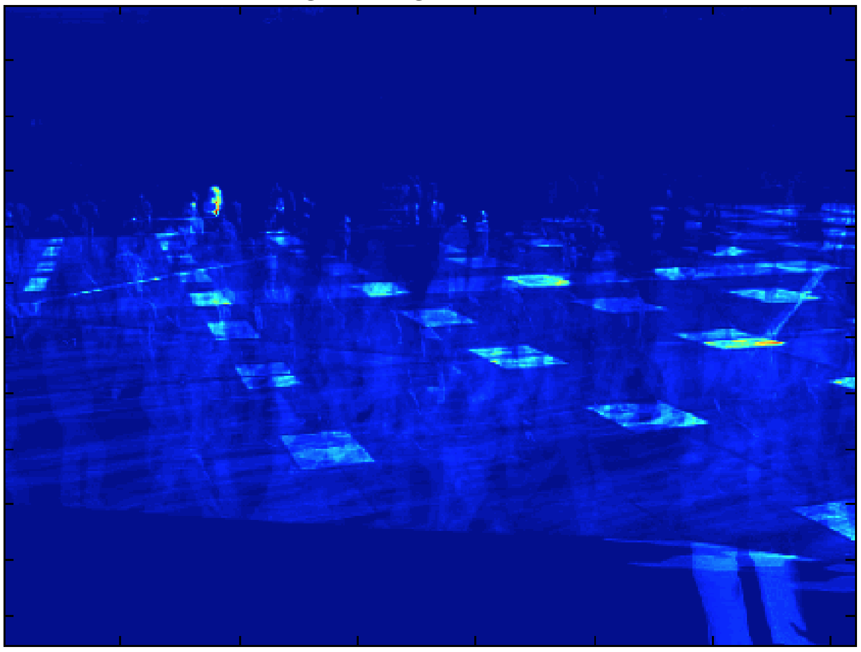
Once you have the background image, you can just subtract a given image from it. We then average the image o

```
diff_img = background_image - img;
average_difference = mean(diff_img, 3);
imshow(uint8(average_difference));
```

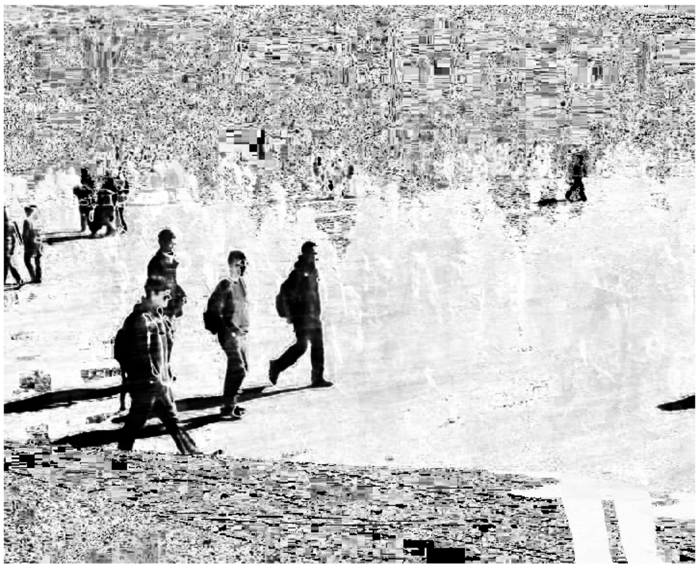
In the figure below, the left image is the original image while the right image is the result of the subtraction.



2. Constructing a Gaussian model for each pixel consists in looping over all the pixels and then computing the me
Here is the variance image averaged across channels.



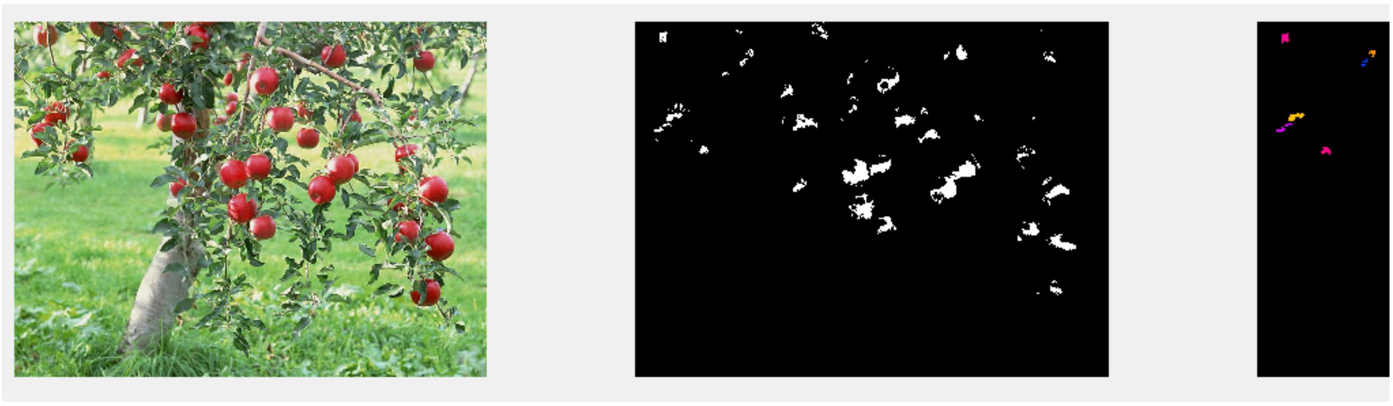
Expected subtracted image (averaged over channels)



Note that the results is very noisy because the Gaussian distribution is not a good pick for all pixels.

Exercise 5

The idea is to segment red pixels from the rest of the image. Of course, it is not enough to simply take pixels with that other channels have low values.
Here's a simple implementation and its output (counting 31 apples).



```
function Exe5()
    close all
    image = imread('apples.jpg');
    figure,
    for i = 1:3
        subplot(1,3,i)
        imshow(image(:,:,i),[])
    end
    imageRed = image(:,:,1);
    imageGreen = image(:,:,2);
    imageBlue = image(:,:,3);
    binaryImage = (imageRed > 150) .* (imageGreen< 80) .* (imageBlue< 80);
    % imageBinary = imageRed > 200
    elementsThreshold = 50
    [imageColor, nConnectedComponents] = CountConnectedComponents(binaryImage, elementsTl
    figure, subplot(1,3,1), imshow(image)
    subplot(1,3,2), imshow(binaryImage)
    subplot(1,3,3), imshow(imageColor)
end
```

Last modified: Thursday, 26 February 2015, 9:44 AM