COMPUTER VISION

Exercise Session 1

Matlab tutorial

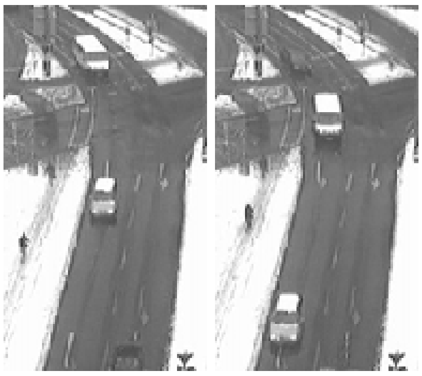
We'll be using a matlab tutorial from Stephan Roth publicly available at

http://www.cs.brown.edu/courses/cs143/docs/matlab-tutorial/

If you are already familiar with matlab and matrix operations, you can go directly to section 8 of the tutorial ("Working with gray level images"). For those of you beginning with matlab, we recommend that you spend enough time to make sure you understand all the concepts introduced in the tutorial. Try to copy-paste the code and run it step by step.

1. Extracting a moving object

Load the "street1.gif" and the "street2.gif" images.



- Transform the 8-bit images into double images (don´t forget to check variable types and cast them if necessary when manipulating images in matlab). Then, subtract the second image from the first one using basic matrix arithmetic operations. What is the result ? Why? (Save this result for future work).

- Use the imsubtract function of MATLAB to check your result.

2. An easy program

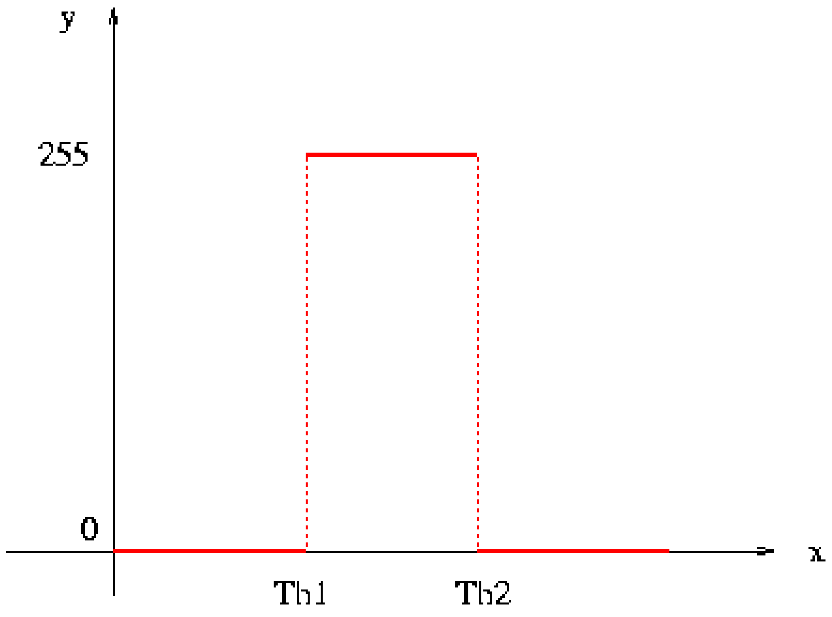
- Download the m file process_im.m and try to understand it by tuning the parameters. What could be the use of this function?

3. Segmenting an image

In many vision applications, it is useful to separate out the regions of the image corresponding to objects in which we are interested from the regions of the image that correspond to the background. Thresholding often provides an easy and convenient way to perform this segmentation on the basis of the different intensities or colours in the foreground and background regions of an image.

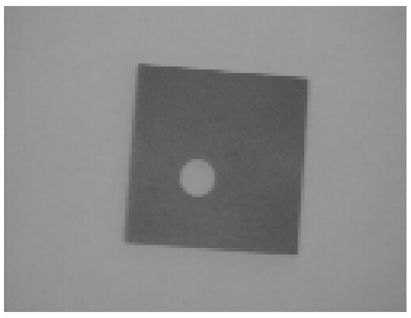
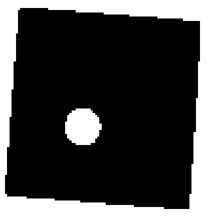
The input to a thresholding operation is typically a greyscale or colour image. In the simplest implementation, the output is a binary image representing the segmentation. Black pixels correspond to background and white pixels correspond to foreground (or vice versa). Multiple thresholds can be specified, so that a band of intensity values can be set to white while everything else is set to black.

If it is possible to separate out the foreground of an image on the basis of pixel intensity, then the intensity of pixels within foreground objects must be distinctly different from the intensity of pixels within the background. In this case, we expect to see a distinct peak in the histogram corresponding to foreground objects such that thresholds can be chosen to isolate this peak accordingly. If such a peak does not exist, then it is unlikely that simple thresholding will produce a good segmentation.

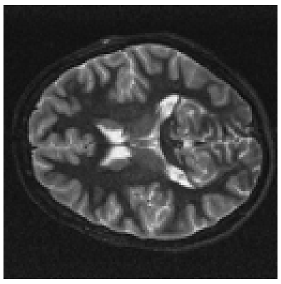


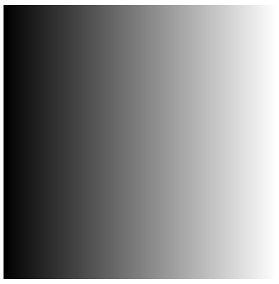
- Write a method to threshold a gray scale image by using two threshold values as shown above. The values must satisfy the following conditions: Th1<Th2, Th1>0, Th2<max.
- Take a look at the histogram (see **imhist** function) of the following image "*wdg.png*", choose the best threshold values and segment the image.

Input (8-bit)..Output

- Threshold the result image from the first exercise (the difference image). Play with the threshold values try to obtain an image of the cars and pedestrian of one image and the same for the other image. What are the results obtained if we apply the threshold directly to the original image?
- If time allows, try to segment the following images:



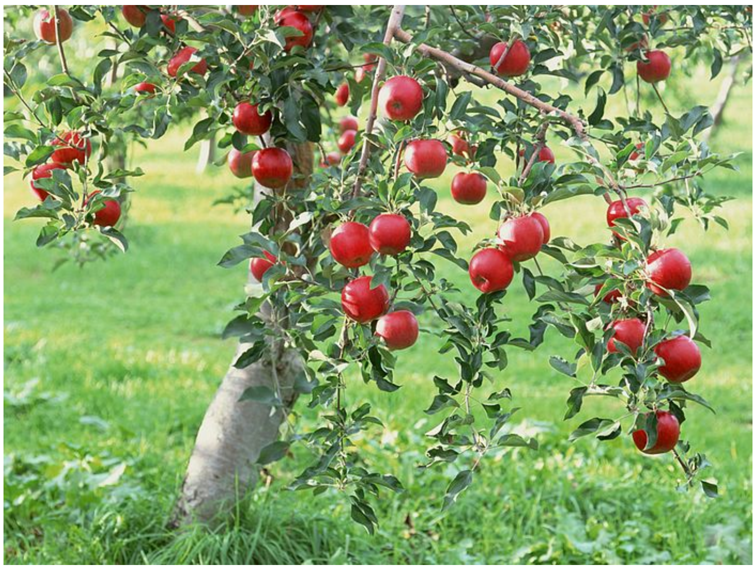


4. Background subtraction

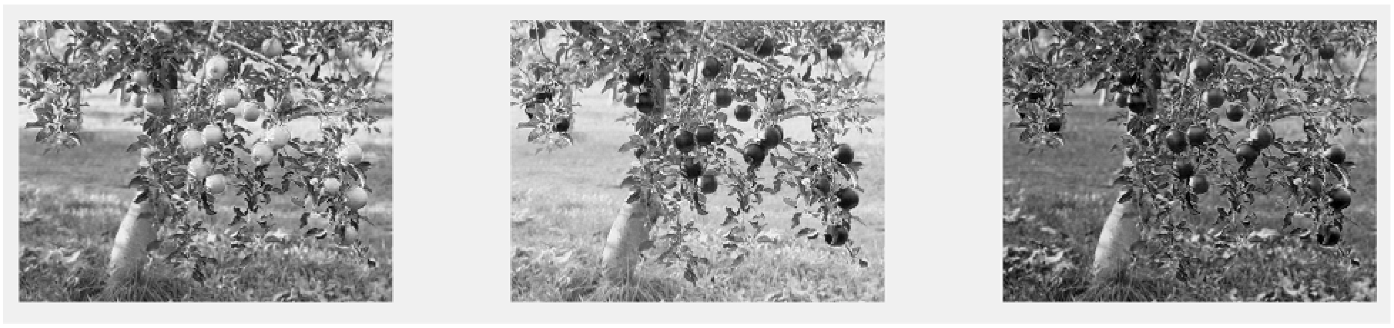
For this exercise, you are given a sequence of images (uncompress the sequence1.zip file) that contains pedestrians we wish to segment with a background subtraction algorithm.
1. First, you are asked to build a "background model" by averaging out the set of given images. The goal of this exercise is to detect pedestrians by subtracting the background model from the original images and applying the right threshold (Hint: use **dir** function to get the list of files).
2. We now want to create a more sophisticated background model where each pixel is modeled with a Gaussian distribution. The threshold will then have to be computed for each pixel according to the variance of the Gaussian distribution.

5. Color image segmentation for a real world application

Our goal is to count the number of apples in the picture below.



1. Download the Exe_1_5.zip package, containing the image and the CountConnectedComponents function.
2. Read the 3-channel image in matlab
3. Visualize the 3 channels (red, green, blue) separately, as it is shown on the figure below:



4. Try to obtain a binary image such that binary image == 1 for pixels representing apples, 0 otherwise.
5. Count the number of connected components in your binary image. For this, you can use the provided function

```
[imageColor, nConnectedComponents] = CountConnectedComponents(binaryImage, elementsThreshold)
```

6. Discuss with your neighbours and answer the following questions:
  - Could your algorithm count the correct number of apples ?
  - What are the main limitations of this approach ?
  - How could you improve your method ?
  - Could you use some other method?
  - Why is it so easy for a human to perform this task?

Last modified: Wednesday, 25 February 2015, 1:06 PM