Aston University

CS3330: Image and Video Processing

Unit 7: Assessed Lab 2

**Instructions:**

Submission is required for this Unit. Please fill in your answers and submit the completed document as “Lab Assessment 2” in the “Lab Assessment Submission” folder of the “Labs” section of the Blackboard module.

You have until 13:55 to complete this assessment. It is your responsibility to ensure that your work is submitted on time and 2 marks will be subtracted from your total for every part minute over this deadline that you submit. After submission, please ensure that your work has been uploaded by choosing the “my submissions” option from the top of the assignment submission page and then selecting your file’s name next to your submission. If the document that you submitted is displayed, then your assignment has been submitted correctly.

**Task 1 (8 marks):**

In this task, you will write a simple function, detect\_horizontal\_edges, which aims to detect horizontal edges in a grayscale image. It should take a single argument, a grayscale image, and return the result of:

1. filtering the image with whichever one of the two Sobel operators (denoted and in the unit 5 lecture notes) is more appropriate for detecting horizontal edges,
   * Note that measures gradient in the horizontal direction, while measures gradient in the vertical direction. Does change in a horizontal edge happen in the horizontal or in the vertical direction?
2. taking the absolute value of result of step 1,
3. thresholding the image, setting the threshold to the 90th percentile of the pixel values resulting from step 2.
   * Matlab has a function,
   * (x,90) returns the 90th percentile of the values of x. prctile, which calculates percentiles of a data set.
   * If x is a vector in Matlab, then prctile

The function should return a matrix which is the same size as the original image, containing 1 (alternatively, true) where an edge has been detected and 0 (alternatively, false) elsewhere.

To check the result of your function, you may want to compare it to Matlab’s implementation which can be applied to an image stored in Matlab variable img using the following command:

edge(img,'Sobel','horizontal','nothinning');

Note that the results will not be identical as Matlab uses a different method for calculating an appropriate threshold.

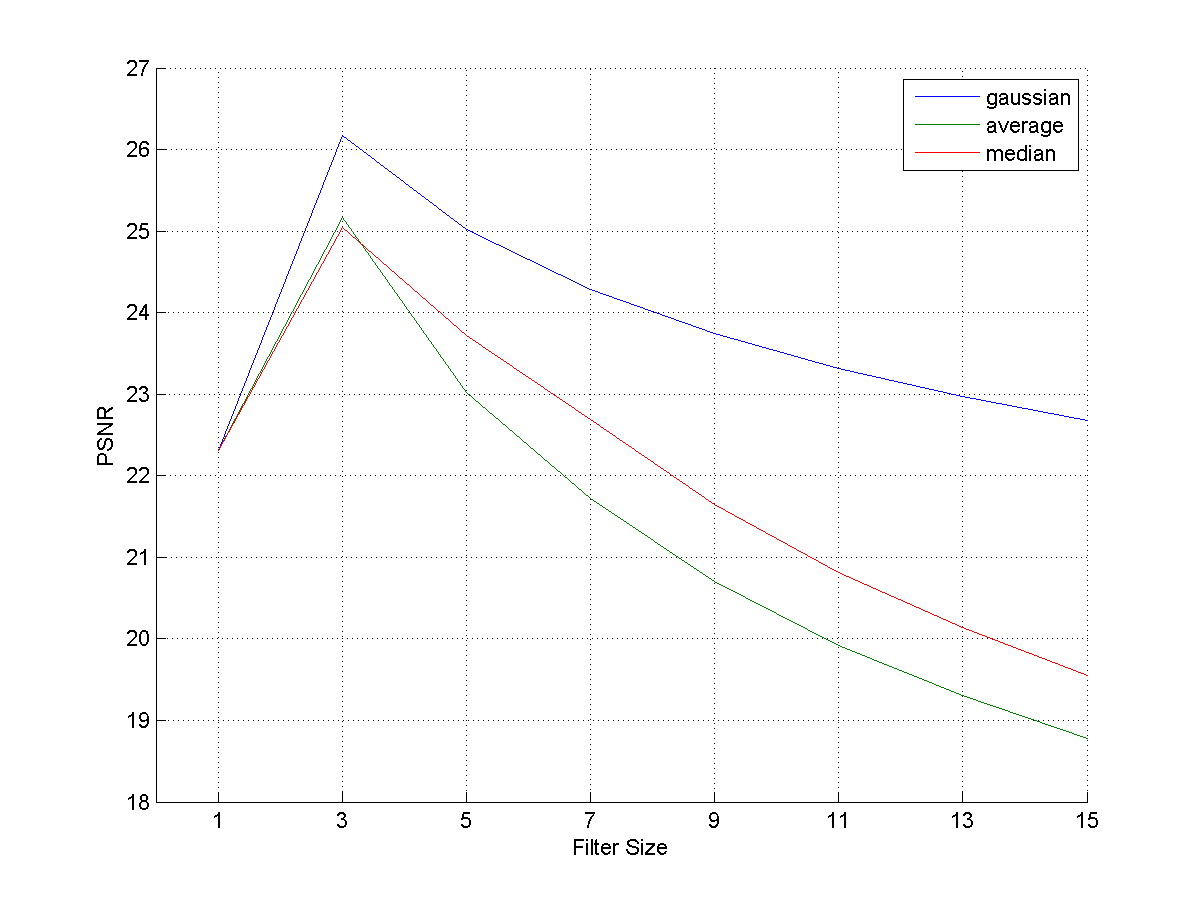
When called, your function should not produce any output in the command window. Copy the contents of your function into the answer box below, indicating the name of the file.

|  |
| --- |
| function [ threshold\_edge ] = detect\_horizontal\_edge( grayscale\_image )  sobel\_mask = [-1 0 1;-2 0 2;-1 0 1];  sobel\_filter = filter2(sobel\_mask, grayscale\_image);  absolute\_value = abs(sobel\_filter);  threshold\_edge = prctile(absolute\_value,90);  end |

**Task 2 (8 marks):**

For this task you will need to download “noisy\_image.bmp” to your Matlab current working directory. The image is available on Blackboard in the same location as this lab sheet. The file contains a corrupted image.

The graph below shows the **peak signal to noise ratio (PSNR)** of “noisy\_image.bmp” compared to the original image (i.e. the image before noise was added) after filtering by three filter types (Gaussian, box and median) at different (square) filter sizes.



Use the information in the graph to decide on the type and size of filter which, when applied to “noisy\_image.bmp”, provides the best reconstruction of the original image. Write down the filter size and type in the answer box below.

|  |
| --- |
| Guassian Filter with filter size 3 |

In Matlab write a script which:

* loads the noisy image,
* filters the image with the filter identified in your answer above,
* attempts to detect horizontal edges in the resulting image using your function from task 1,
* saves the result to your hard drive as a **png** image with name “image\_edges.png”.

Your script should not produce any output in the command window and, after running, should not leave any extra variables in the workspace.

Copy the contents of the script file into the answer box below.

|  |
| --- |
| noisy\_image = imread('noisy\_image.bmp');  gaussian\_mask = fspecial('gaussian',3,0.84932);  gaussian\_img = filter2(gaussian\_mask, noisy\_image);  edge\_detected = detect\_horizontal\_edge(gaussian\_img);  imwrite(edge\_detected,'image\_edges.png'); |

**Task 3 (4 marks):**

For this task you will need to download “unsharp\_mask.mat” to your Matlab current working directory. The file is available on Blackboard in the same location as this lab sheet. The file is a Matlab data file containing a variable unsharp\_mask, a precalculated unsharp mask.

Write a script which loads the data file and displays the unsharp mask, scaled such that it can be displayed as an 8-bit grayscale image. You may make use of the function scale\_matrix from lab 6 in your solution. An implementation of scale\_matrix is available under “Lab Slides and Solutions” on Blackboard.

Your script should not produce any output in the command window and, after running, should not leave any extra variables in the workspace.

Copy the contents of the script file into the answer box below.

|  |
| --- |
| matrix = load('unsharp\_mask.mat');  minValue = min(matrix(:));  newMatrix = matrix - minValue;  maxValue = max(newMatrix(:));  productMatrix = newMatrix.\*(255/maxValue);  figure,image(productMatrix),axis off,colormap gray(256);  clear; |