Robot Vision [06-25024]
Summative Assignment 1

March 10, 2023

Submission deadline:	**18:00pm (BST), Friday, 17 March 2023**
Instructor:	Dr. Hyung Jin Chang
	Dr. Jianbo Jiao
Total marks:	100
Contribution to overall module mark:	25%
Submission Method:	This assignment must be submitted through Canvas.

Important note on plagiarism

Plagiarism can have very serious potential consequences. Please see `https://canvas.bham.ac.uk/courses/64039/pages/university-legislation?module_item_id=2667061` for further information. Plagiarism includes failure to provide proper attribution for ideas originating from external sources and copying from other students or external sources. Changing the wording of the copied text is still considered plagiarism. It is acceptable and even encouraged, to discuss course content and assignments with your fellow students. However, you must write your answers to all assignments individually and you must not share those answers with other students.

Instructions

Your answer must be submitted to Canvas before the deadline in the form of **a single zip archive file** containing:

1. Your answers to the questions in prose and diagrams. This should take the form of a single PDF document with the answers to each question. You may choose any editor to create the PDF document, but it is highly recommended to use the provided LaTeX template.

2. Your code and any accompanying files necessary to execute the code for any programming questions as specified in the LaTeX template.

3. Do not use chatGPT or any other automated writing tool for this assignment.

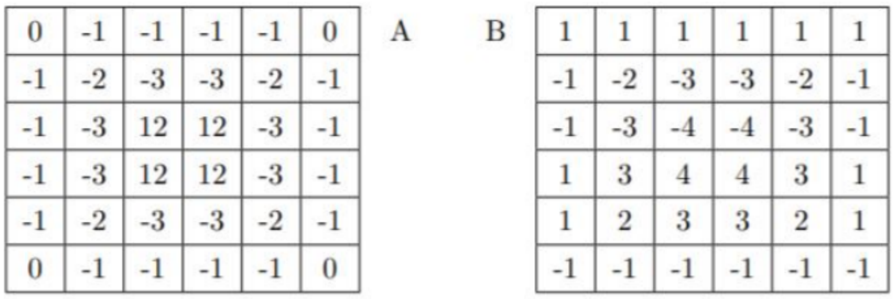
and **a separate PDF document** with the answers (two files in total; one zip file and one PDF file). Some or all of the text of each question is emphasised using *italics*. This emphasis indicates a question that must be explicitly answered or a task that must be completed.

Part 1

It is required to submit your code as username_assignment1_part1.m. In the absence of the source code this part will not be graded. The image files that will be used in this part are `Malards.jpg` and `Earth.jpg`.

Question 1.1 [**7 marks**] *Create a binary edge image of* ***Malards.jpg*** *using only the red channel of the image. Obtain the binary image with all edges that are stronger than 50 using Roberts edge detection. Show the binary image.*

Question 1.2 [**7 marks**] *A pair of filter kernels are shown in Figure 1. Produce the resulting convolved images for the image* ***Malards.jpg*** *with two kernels separately. Are the filters isotropic or anisotropic? Which filter would you use for finding edges? What kind of edges would you use it to find?*



A

0	-1	-1	-1	-1	0
-1	-2	-3	-3	-2	-1
-1	-3	12	12	-3	-1
-1	-3	12	12	-3	-1
-1	-2	-3	-3	-2	-1
0	-1	-1	-1	-1	0

B

1	1	1	1	1	1
-1	-2	-3	-3	-2	-1
-1	-3	-4	-4	-3	-1
1	3	4	4	3	1
1	2	3	3	2	1
-1	-1	-1	-1	-1	-1

Figure 1: An Example Image and Kernel

Question 1.3 [**8 marks**] *Create a function* ***DiffGaus****, which is capable of applying custom difference of Gaussian filter to the image* ***Earth.jpg****. It should be able to give the size of the filter in pixels, and the parameters of the two Gaussian, as inputs to the function. You may use existing MATLAB functions for filtering but must write your own code to generate difference of Gaussian kernels. Implement your function to approximate the Laplacian of the Gaussian with the two* 15×15 *Gaussian* ($\sigma_1 = 1$, $\sigma_2 = 2\sqrt{2}$) *and apply this filter to the image* ***Earth.jpg*** *and show your results.*

Question 1.4 [**8 marks**] *Implement the* ***DiffGaus*** *function you created in* ***Question 1.3*** *to apply difference of Gaussian filters to the image* ***Earth.jpg*** *at different scales to produce a set of filtered images forming a scale-space pyramid. Apply your function to the image* ***Earth.jpg*** *and show the different scales produced by your function.*

Part 2

It is required to submit your code as username_assignment1_part2.m. In the absence of the source code this part will not be graded.

Question 2.1 [**8 marks**] *Load the* **Old_house.jpg** *image. Convert them to grayscale. Write a function,* ***convolve****, which takes a filter, F, and an image, I, as input and calculates the 2D convolutions of I with F via the use of 'for' loops. Zero-pad the image within the function to ensure that the filtered image is the same size as the original image. You may assume that F always has an odd value for its height and width. DO NOT USE the existing conv, conv2, convn, filter, or filter2 functions. Apply a 15×15* ***Gaussian filter*** *with a standard deviation of 5 to both images using the* ***convolve*** *function. Show the results.*

Hint: The built-in function ***padarray*** *is useful for zero-padding.*

Question 2.2 [**6 marks**] *Apply a 15×15* ***mean filter*** *to the loaded images using the* ***convolve*** *function. Show the results.*

Question 2.3 [**8 marks**] *Implement a nonlinear filter — a **median filter**. While the Gaussian filter works by computing locally-weighted values of the signal, the median filter first sorts the signal values within a window and then takes the middle value (i.e., the median).*

*Implement a function, **simpleMedian**, which takes as input an image, I, and the dimensions of the median filter, W and H, and returns a median filtered image. Zero-pad the image within the function to ensure that the filtered image is the same size as the original image.*

*Apply a 12×12 **median filter** to the loaded images. Show the results.*

Question 2.4 [**8 marks**] *Load the **Butterfly.jpg** image from part 2. Convert it to grayscale. Corrupt it with:*

i. Salt and Pepper Noise (noise density = 0.09):

ii. Gaussian Noise (mean = 0.1, variance = 0.2):

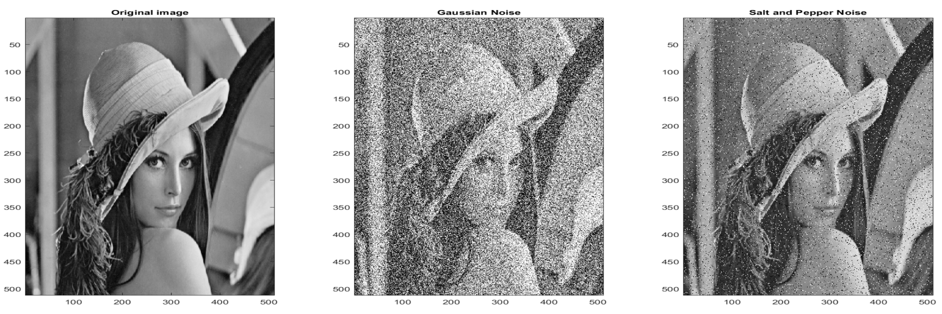


Figure 2: Examples of Noise

Examples of the two types of noise are shown in Figure 2 Then examine the effects of your Gaussian, Box, and Median filters in turn on the two types of noise. Display the results in a 2 × 4 figure with subplots, with the noise type changing between the rows and the columns displaying the filter used (no filter, Gaussian, Mean, and Median). The layout is shown in Figure 3.

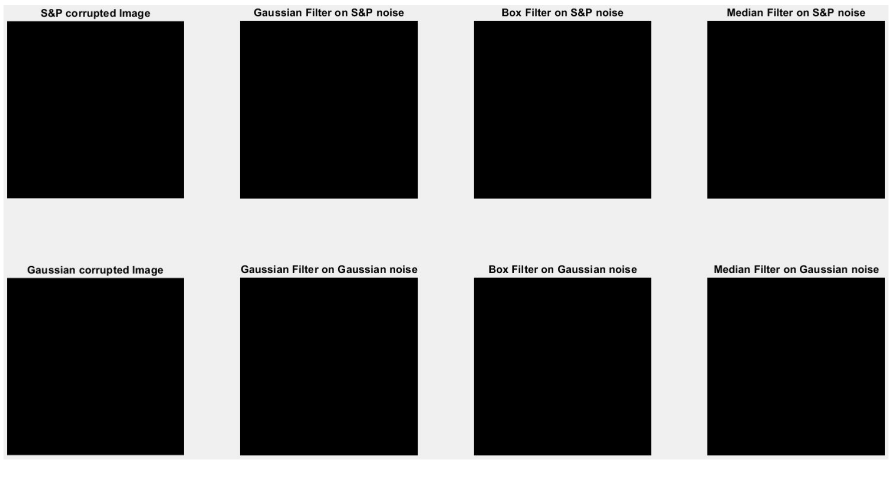

S&P corrupted Image
Gaussian Filter on S&P noise
Box Filter on S&P noise
Median Filter on S&P noise
Gaussian corrupted Image
Gaussian Filter on Gaussian noise
Box Filter on Gaussian noise
Median Filter on Gaussian noise

Figure 3: Example Layout for 3.4

Which filter performs the best on each noise and why? Explain what you see.

*Hint: The built-in function **imnoise** is useful for adding noise.*

Part 3

It is required to submit your code as username_assignment1_part3.m. In the absence of the source code this part will not be graded.

Question 3.1 [20 marks] *Use the Hough transformation for line and circle detection on the image. Your task is to implement a code that detects as many lines and circles in the given image* ***Dice.jpg*** *as possible. The example result is shown in Figure 4.*
Hint: Using the matlab built-in function like ***hough*** *and* ***hougpeaks*** *is allowed.*
Use ***hold*** *on and* ***plot*** *functions to display the detected circles on the image.*

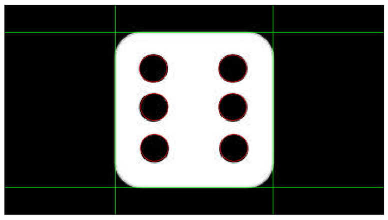


Figure 4: Example Result

Part 4

It is required to submit your code as username_assignment1_part4.m. In the absence of the source code this part will not be graded.

Question 4.1 [20 marks] *There are various algorithms which detect different kinds of features. An image of the University Square of UoB is provided. Plot on a 2×3 sub-plotted figure, the 100 strongest features when using the Minimum Eigenvalue, SURF, KAZE, FAST, ORB and the Harris-Stephens algorithms (all the above algorithms allow the use of* ***built-in functions*** *provided by MATLAB). Include this sub-figure in your report and ensure that when the MATLAB file executes, it appears as a sub-figure. Save and include the MATLAB sub-figure in your submission.*

Summary of what needs to be answered:

1. *Code for part 1.2 (**username_assignment1_part1_Q4_1.m**)*
2. *The generated sub-plotted figure (report in PDF)*
3. *The generated sub-plotted figure (.fig file)*
4. *All images needed for the code to function*