





[Dashboard](#)  [My courses](#)  [7CCSMCVI & 6CCS3COV 20-21](#)  [Week 3: Low-Level Vision \(Artificial\)](#)  [Coursework 3 \(worth 8% of module mark\)](#)

Started on	Monday, 12 October 2020, 9:05 AM
State	Finished
Completed on	Thursday, 15 October 2020, 9:31 AM
Time taken	3 days
Grade	8.00 out of 8.00 (100%)

Feedback

Well done!

You have successfully passed this quiz.

Information

Introduction

This coursework provides the opportunity to experiment with a number of image processing techniques used for low-level computer vision.

Remember that commands which you can type at the MATLAB prompt are indicated by text in the following typeface:

```
matlab_function(parameter1, parameter2);
```

You will work with the rooster, elephant and boxes images. Copy the rooster.jpg, elephant.png and boxes.pgm files from the module's KEATS webpage to the directory in which you are running MATLAB, and load these images into MATLAB by using the following commands:

```
Ia=imread('rooster.jpg');
```

```
Ib=imread('elephant.png');
```

```
Ic=imread('boxes.pgm');
```

Once you load these images into MATLAB, convert colour images to greyscale (i.e. intensity) images, and convert all images from uint8 format to double format. Use these converted images throughout the rest of this coursework. For example, when the instructions tell you to use the rooster image, it means that you should use the grayscale floating-point version of this image.

This coursework will make extensive use of convolution.

In MATLAB, 2-D convolution is performed using the function `conv2(I,H,shape)`, where `I` is a 2-dimensional matrix representing the image, `H` is a matrix representing the mask, and `shape` is a parameter defining the size of the output. To perform convolution on a colour image we need to convert the image to grayscale (or apply the convolution to a single channel of the image).

Execute the command:

```
help conv2
```

Determine what values of parameter "shape" can be used.

MATLAB provides a command `fspecial` that can generate some masks commonly used in image processing.

Execute the command:

```
help fspecial
```

Determine what parameter values are required to generate a box (or average) mask and a Gaussian mask.

Question 1

Correct

Mark 1.00 out of 1.00

Smoothing using Box Masks

A box, or average, mask is just an array of equal numbers that sum to one. Such a mask when convolved with an image provides smoothing.

Use the command `ones` or the command `fspecial` to create a 5x5 box mask and a 25x25 box mask. Ensure these are correctly normalised.

Convolve both the rooster (converted to grayscale) and boxes images with each of the box masks so that the resulting outputs are the same size as the original images.

Display the output of each convolution as a subplot in the same window. Use a "gray" colormap and provide a colorbar for each image.

EXERCISE:

Briefly explain the results you have obtained.

Report the the pixel intensity values for the following images and locations (correct to 2 decimal places):

Pixel intensity at row=301, column=6 of rooster image convolved with 5x5 box mask: ✓

Pixel intensity at row=301, column=6 of rooster image convolved with 25x25 box mask: ✓

Pixel intensity at row=46, column=35 of boxes image convolved with 5x5 box mask: ✓

Pixel intensity at row=46, column=35 of boxes image convolved with 25x25 box mask: ✓

You should have written code equivalent to the following:

```
box5=ones(5,5)./(5^2);
```

```
box25=ones(25,25)./(25^2);
```

```
Iabox5=conv2(Ia,box5,'same');
```

```
Iabox25=conv2(Ia,box25,'same');
```

```
Icbox5=conv2(Ic,box5,'same');
```

```
Icbox25=conv2(Ic,box25,'same');
```

```
figure(1), clf
```

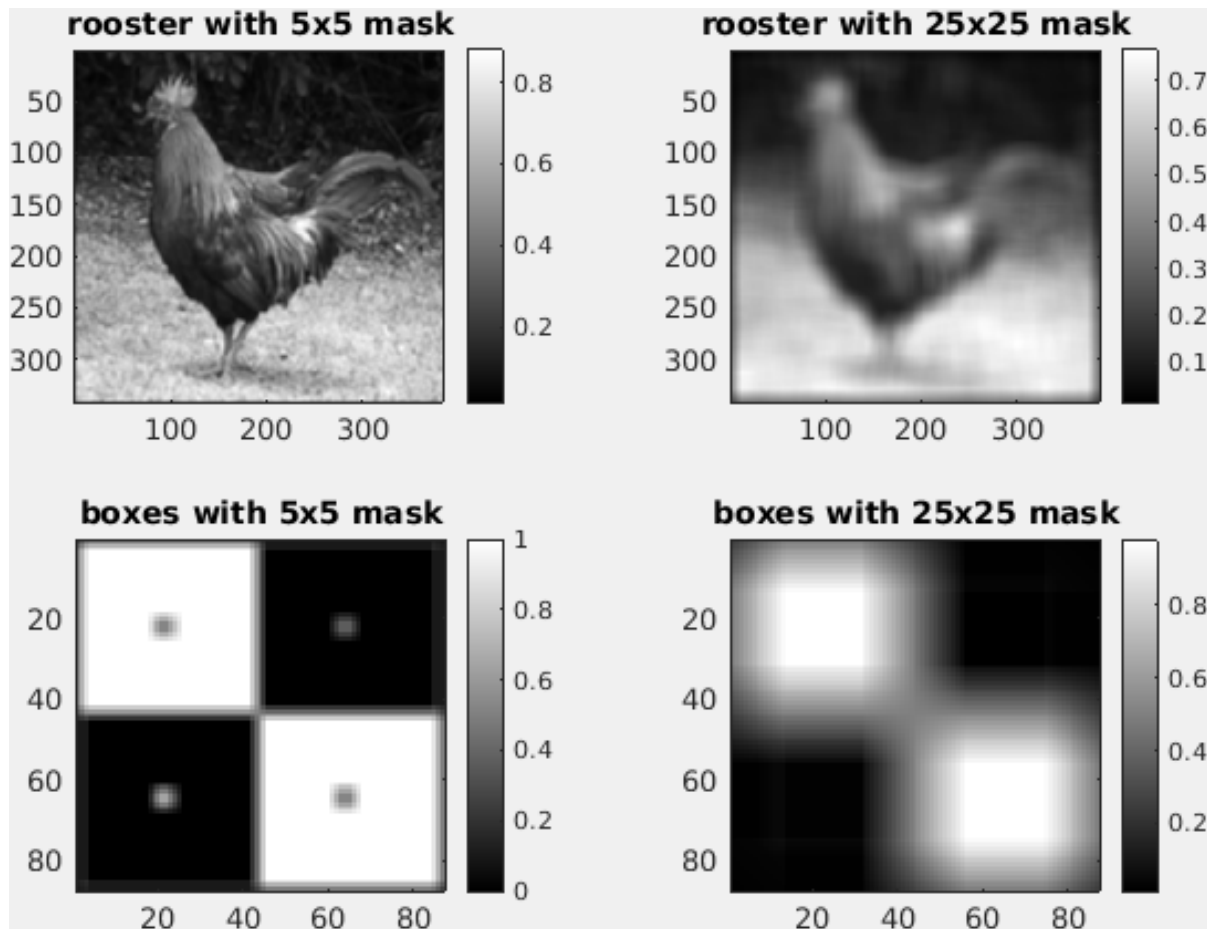
```
subplot(2,2,1), imagesc(Iabox5), colorbar, title('rooster with 5x5 mask');
```

```
subplot(2,2,2), imagesc(Iabox25), colorbar, title('rooster with 25x25 mask');
```

```
subplot(2,2,3), imagesc(Icbox5), colorbar, title('boxes with 5x5 mask');
```

```
subplot(2,2,4), imagesc(Icbox25), colorbar, title('boxes with 25x25 mask');
```

And produced a figure like this:



Observe that all pictures are more blurred than the originals (equivalently, the images contain less high spatial frequency components than the originals, or are smoothed). Blurring is due to each pixel in the original image being replaced by a pixel with an intensity value that is the average of the original pixel and its neighbours. When the mask is larger, the blurring (reduction in high spatial frequency) is greater as more pixels are included in the average. This can be seen in the boxes image where the small squares in each quadrant disappear when the image is convolved with the larger box mask.

You should have obtained the numerical values, as follows (changing the variable names to match those used in your code):

Pixel intensity at row=301, column=6 of Iabox5: `Iabox5(301,6)`

Pixel intensity at row=301, column=6 of Iabox25: Iabox25(301,6)

Pixel intensity at row=46, column=35 of Icbox5: Icbox5(46,35)

Pixel intensity at row=46, column=35 of Icbox25: Icbox25(46,35)

Question 2

Correct

Mark 1.00 out of 1.00

Smoothing using Gaussian Masks

Repeat the procedure described in the preceding section using two Gaussian masks (generated using the `fspecial` command) with standard deviations 1.5 and 10. Ensure the size of each mask is sufficient to accurately represent the Gaussian.

EXERCISE:

Compare these results with those obtained using the box masks. Briefly explain the results you have obtained.

Report the the pixel intensity values for the following images and locations.

Pixel intensity at row=337, column=208 of rooster image convolved with Gaussian with std=1.5: 0.7901 ✓

Pixel intensity at row=337, column=208 of rooster image convolved with Gaussian with std=10: 0.4484 ✓

Pixel intensity at row=83, column=56 of boxes image convolved with Gaussian with std=1.5: 0.9962 ✓

Pixel intensity at row=83, column=56 of boxes image convolved with Gaussian with std=10: 0.5755 ✓

You should have written code equivalent to the following:

```
g1=fspecial('gaussian',9,1.5);
```

```
g2=fspecial('gaussian',60,10);
```

```
Iag1=conv2(Ia,g1,'same');
```

```
Iag2=conv2(Ia,g2,'same');
```

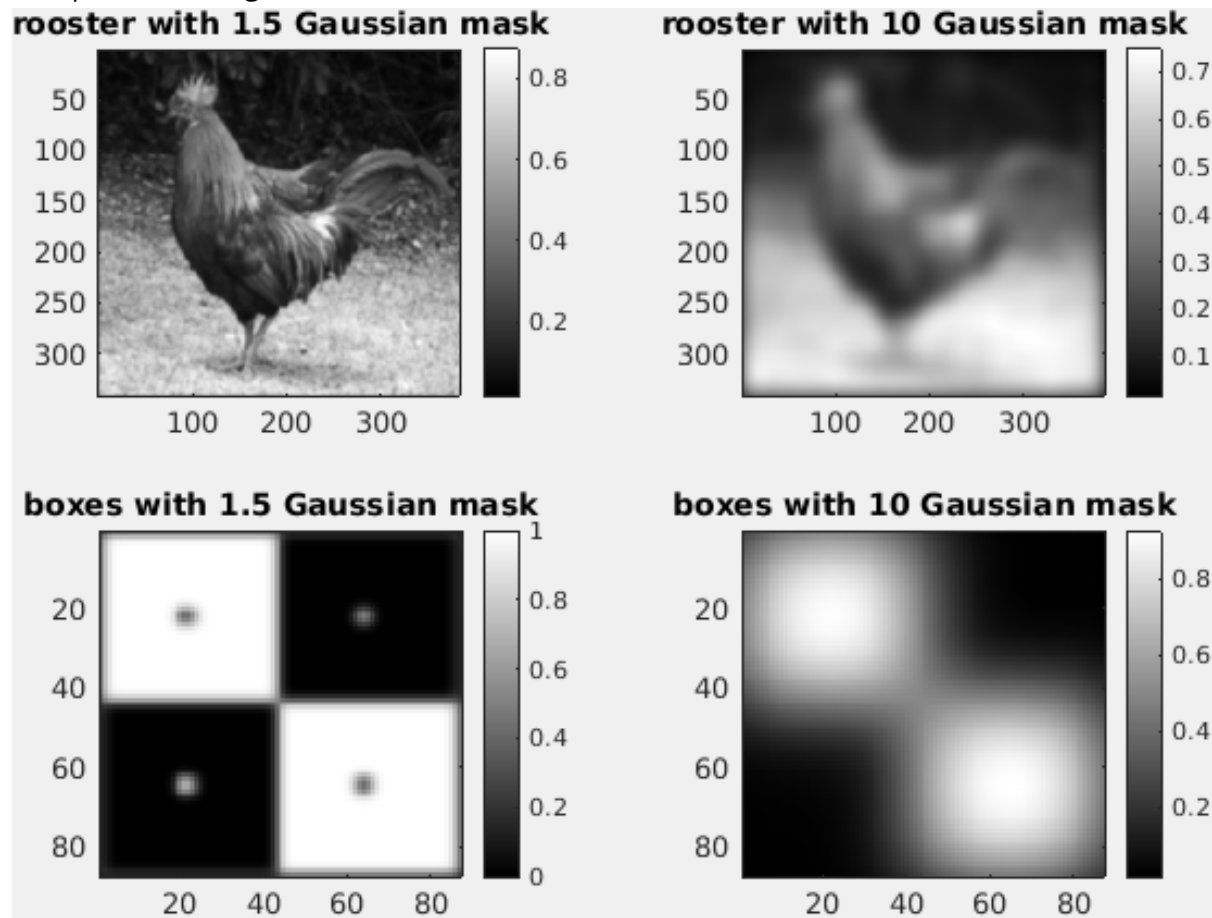


```

Icg1=conv2(Ic,g1,'same');
Icg2=conv2(Ic,g2,'same');
figure(2), clf
subplot(2,2,1), imagesc(Iag1), colorbar, title('rooster with 1.5 Gaussian mask');
subplot(2,2,2), imagesc(Iag2), colorbar, title('rooster with 10 Gaussian mask');
subplot(2,2,3), imagesc(Icg1), colorbar, title('boxes with 1.5 Gaussian mask');
subplot(2,2,4), imagesc(Icg2), colorbar, title('boxes with 10 Gaussian mask');

```

And produced a figure like this:



Observe that all pictures are more blurred than the originals. This is because each pixel is the average of a number of pixels in the input image. As the

mask gets larger the number of pixels included in the average increases, and hence, so does the smoothing. The small (large) Gaussian masks produce a similar level of blurring to the small (large) box masks. The Gaussian masks are better at removing high spatial frequencies than the box masks as the averaging is weighted by distance. This can be seen in the boxes image when it is convolved with the large boxes mask there are steps in the output, whereas when it is convolved with the large Gaussian mask the result is smoother.

You should have obtained the numerical values, as follows (changing the variable names to match those used in your code):

Pixel intensity at row=337, column=208 of I_{ag}1: `Iag1(337,208)`

Pixel intensity at row=337, column=208 of I_{ag}2: `Iag2(337,208)`

Pixel intensity at row=83, column=56 of I_{cg}1: `Icg1(83,56)`

Pixel intensity at row=83, column=56 of I_{cg}2: `Icg2(83,56)`

Question 3

Correct

Mark 1.00 out of 1.00

Differencing with First-Derivative Masks

The first-derivative in the y-direction can be approximated by convolving an image with the following mask: $[-1;1]$.

The first-derivative in the x-direction can be approximated by convolving an image with the following mask: $[-1,1]$.

Convolve the elephant image with both these masks, using the "valid" shape option, and show the results.

EXERCISE:

Report the value of the first derivatives you have calculated at the following locations (correct to 2 decimal places):

row=241, column=360 first derivatives in the y-direction ✓ and in the x-direction ✓

row=174, column=476 first derivatives in the y-direction ✓ and in the x-direction ✓

You should have written code equivalent to the following:

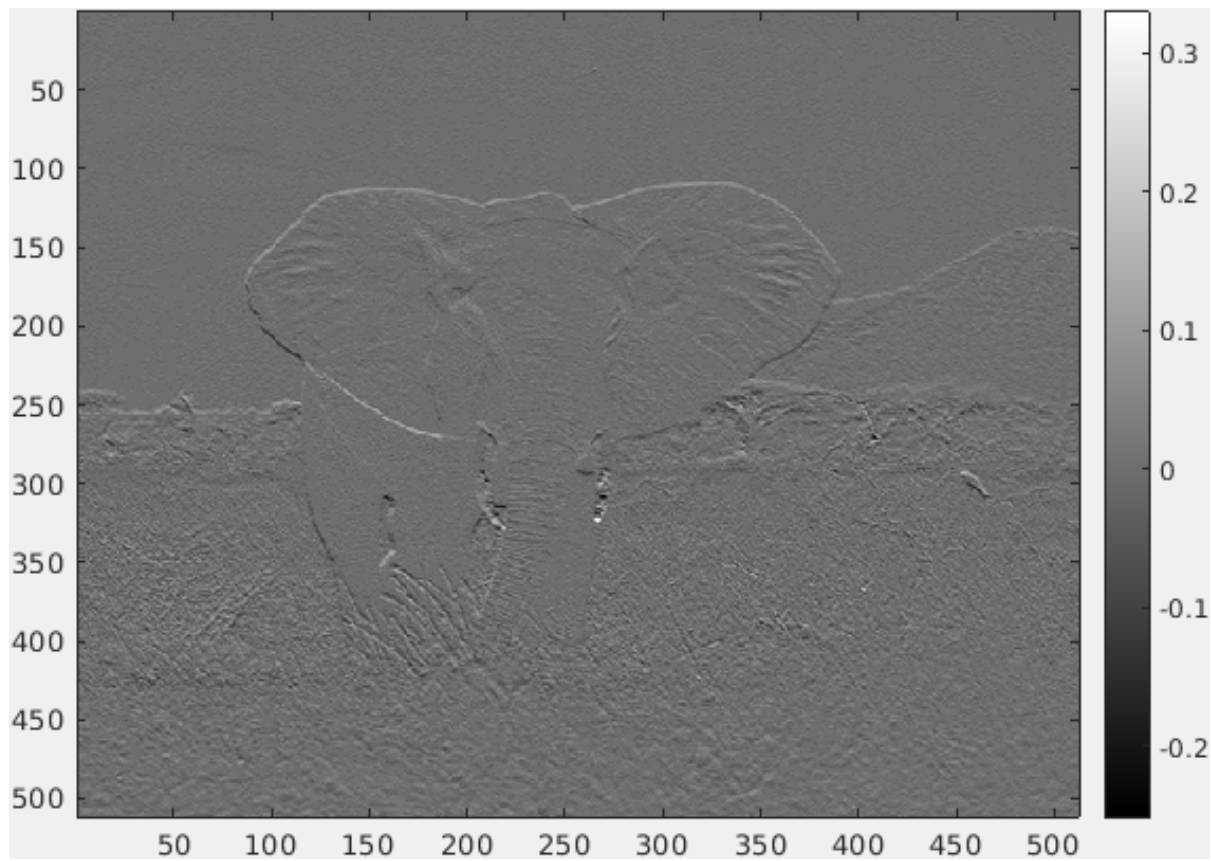
```
Ibdiffy=conv2(Ibd,[-1;1],'valid');
```

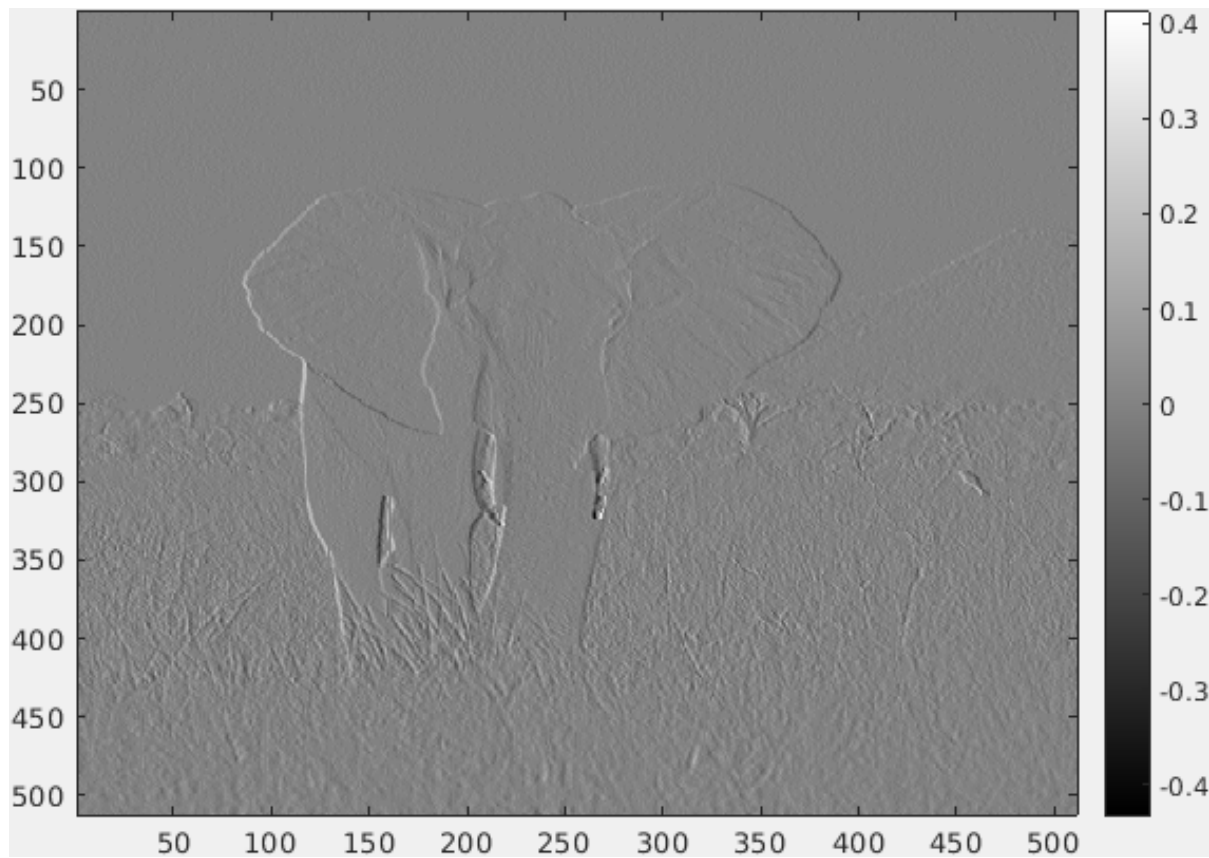
```
figure(3), clf, imagesc(Ibdiffy); colormap('gray'); colorbar;
```

```
Ibdiffx=conv2(Ibd,[-1,1],'valid');
```

```
figure(4), clf, imagesc(Ibdiffx); colormap('gray'); colorbar;
```

And produced images like this:





Note these results are identical to those obtained in last week's coursework in the section "Intensity Change Detection using Image Shifts".

You should have obtained the numerical values, as follows (changing the variable names to match those used in your code):

```
Ibdiffy(241,360)
```

```
Ibdiffx(241,360)
```

```
Ibdiffy(174,476)
```

```
Ibdiffx(174,476)
```

Question 4

Correct

Mark 1.00 out of 1.00

Differencing with Laplacian Masks

In two dimensions a finite difference approximation to the second derivative is given by a Laplacian mask:

$$\begin{array}{ccc} -1/8 & -1/8 & -1/8 \\ -1/8 & 1 & -1/8 \\ -1/8 & -1/8 & -1/8 \end{array} \quad \text{or} \quad \begin{array}{ccc} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{array}$$

(Strictly speaking these masks are the additive inverse of the Laplacian, and hence, approximate the minus of the second derivative).

Convolve both the boxes image and the rooster image with the Laplacian with the smaller amplitude, as defined on the left above. Ensure that the output of the convolution is the same size as the input image. Show the results.

EXERCISE:

For the output produced by convolving the boxes image with the Laplacian, report the values at the following locations (correct to 2 decimal places):

row=22, coloumn=41 ✓

row=22, coloumn=42 ✓

row=22, coloumn=43 ✓

row=22, coloumn=44 ✓

row=22, coloumn=45 ✓

You should have written code equivalent to the following:

```
laplacian=[-0.125,-0.125,-0.125;-0.125,1,-0.125;-0.125,-0.125,-0.125];
```

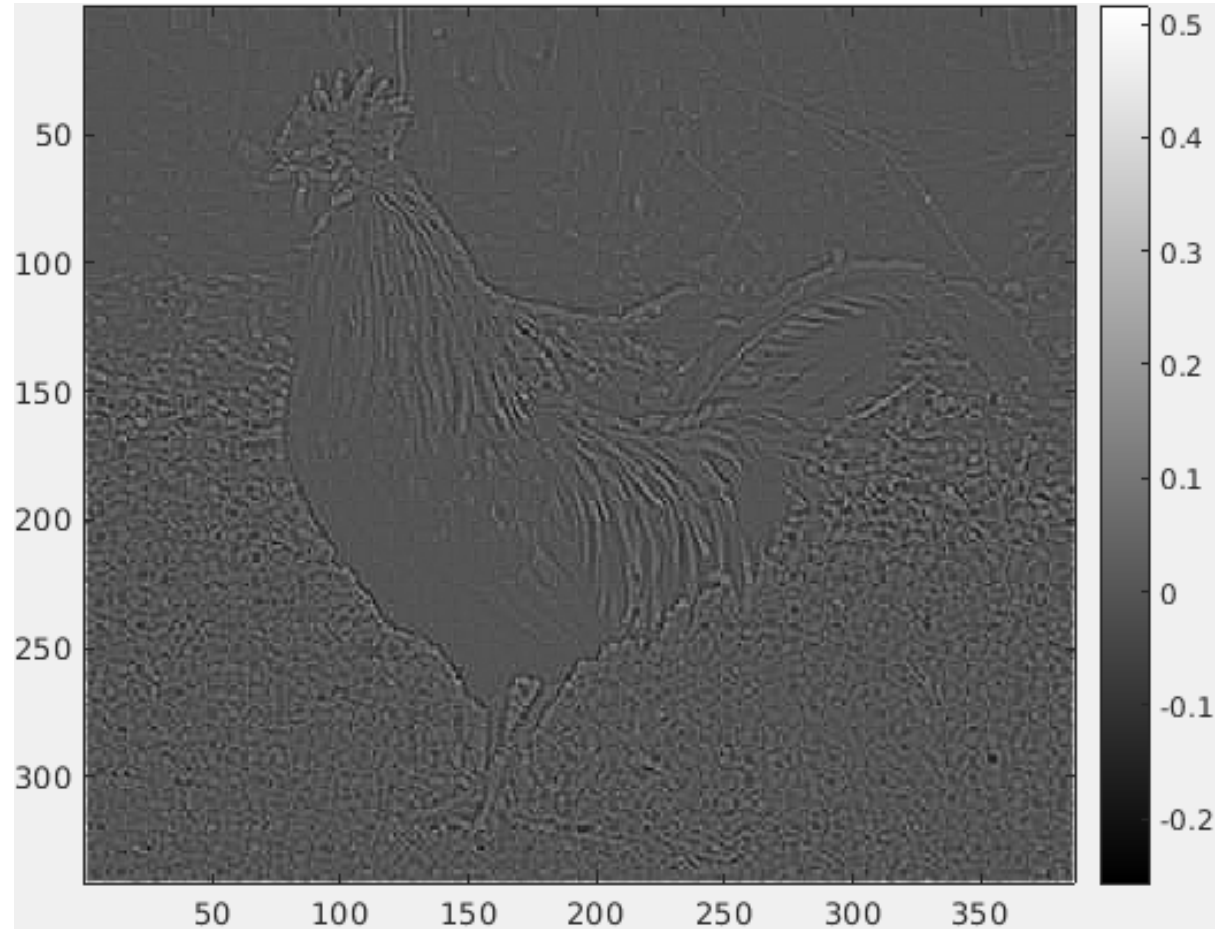
```
Ialap=conv2(Ia,laplacian,'same');
```

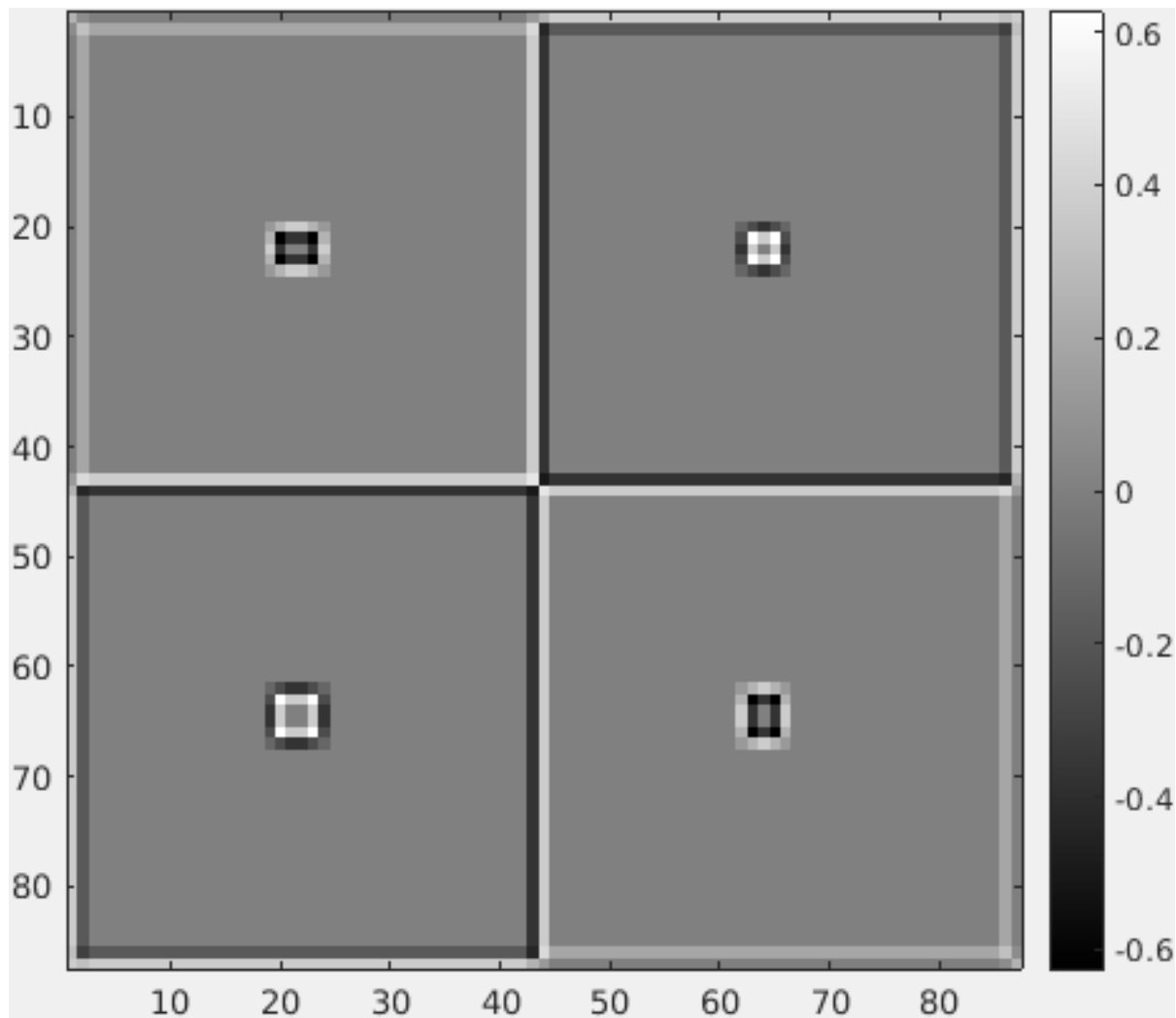
```
Iclap=conv2(Ic,laplacian,'same');
```

```
figure(5), clf, imagesc(Ialap), axis('equal','tight'); colormap('gray'); colorbar
```

```
figure(6), clf, imagesc(Iclap), axis('equal','tight'); colormap('gray'); colorbar
```

And produced images like this:





You should have obtained the numerical values, as follows (changing the variable names to match those used in your code):

```
Iclap(22,41)
```

```
Iclap(22,42)
```

```
Iclap(22,43)
```

```
Iclap(22,44)
```

```
Iclap(22,45)
```


Observe that either side of an intensity discontinuity the amplitude is high (either positive or negative). In theory, the output passes through zero at the location of the discontinuity.

Information

Other Difference Masks

A number of other difference masks for approximating intensity-level gradients of an image have been proposed, and are in common usage for intensity discontinuity or edge detection. Two of these are the Sobel and the Prewitt masks.

Use the command `fspecial` to generate a Sobel mask.

Convolve the boxes image once with the Sobel mask, and once with the transpose of the Sobel mask. Display the output of these two convolutions.

Use the command `fspecial` to generate a Prewitt mask.

Convolve the boxes image once with the Prewitt mask, and once with the transpose of the Prewitt mask. Display the output of these two convolutions.

Question 5

Correct

Mark 1.00 out of 1.00

Edge Detection with Gaussian Derivative Masks

To perform edge detection (i.e. to locate intensity discontinuities in an image that are more likely to be meaningful), first and second order derivative masks are usually combined with a Gaussian mask to help suppress noise. Combining first derivative masks with a Gaussian results in Gaussian derivative masks.

Create two Gaussian derivative masks (one for the derivative in the x direction, and one for the derivative in the y direction). To do this convolve a 2-D Gaussian (with standard deviation 2.5) once with the mask $[-1, 1]$ and once with the transpose of this mask. Ensure that size of the resulting mask is sufficiently large to accurately represent the Gaussian derivative function.

Generate `mesh` plots of the two Gaussian derivative masks you have created, put these plots as `subplot(2,2,1)` and `subplot(2,2,2)` in a figure. Convolve the boxes image with each of the two Gaussian derivative masks you have created. Display images showing the output of these two convolutions as `subplot(2,2,3)` and `subplot(2,2,4)` in the same figure window.

The result is two images, one with large values at the locations where there are vertical discontinuities in the boxes image, and one where there are horizontal discontinuities. It is possible to combine these into a single image showing intensity-level discontinuities in both directions. This is achieved using the L2-norm, as follows:

```
Icdg=sqrt(Icdgx.^2+Icdgy.^2);
```

The above assumes that you have given your image convolved with the horizontal Gaussian derivative mask the variable name `Icdgx`, and that you have given your image convolved with the vertical Gaussian derivative mask the variable name `Icdgy`. Create an image showing the L2-norm image generated.

EXERCISE:

For the output produced by convolving the boxes image with the Gaussian derivative masks (for the derivative in the x direction), report the values at the following locations (correct to 2 decimal places):

row=22, coloumn=41 0.12 ✓

row=22, coloumn=42 0.15 ✓

row=22, coloumn=43 0.16 ✓

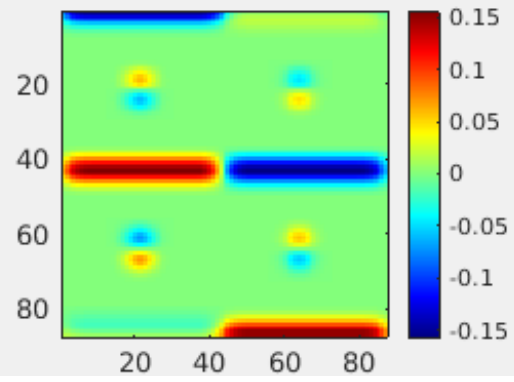
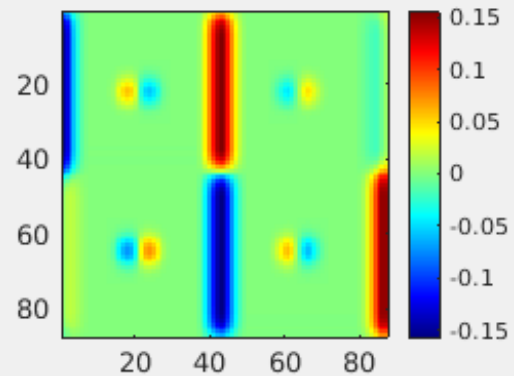
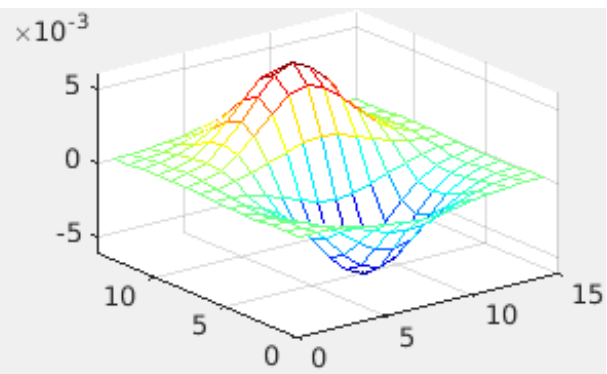
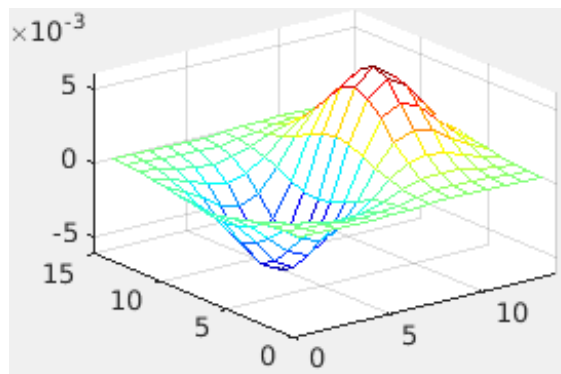
row=22, coloumn=44 0.15 ✓

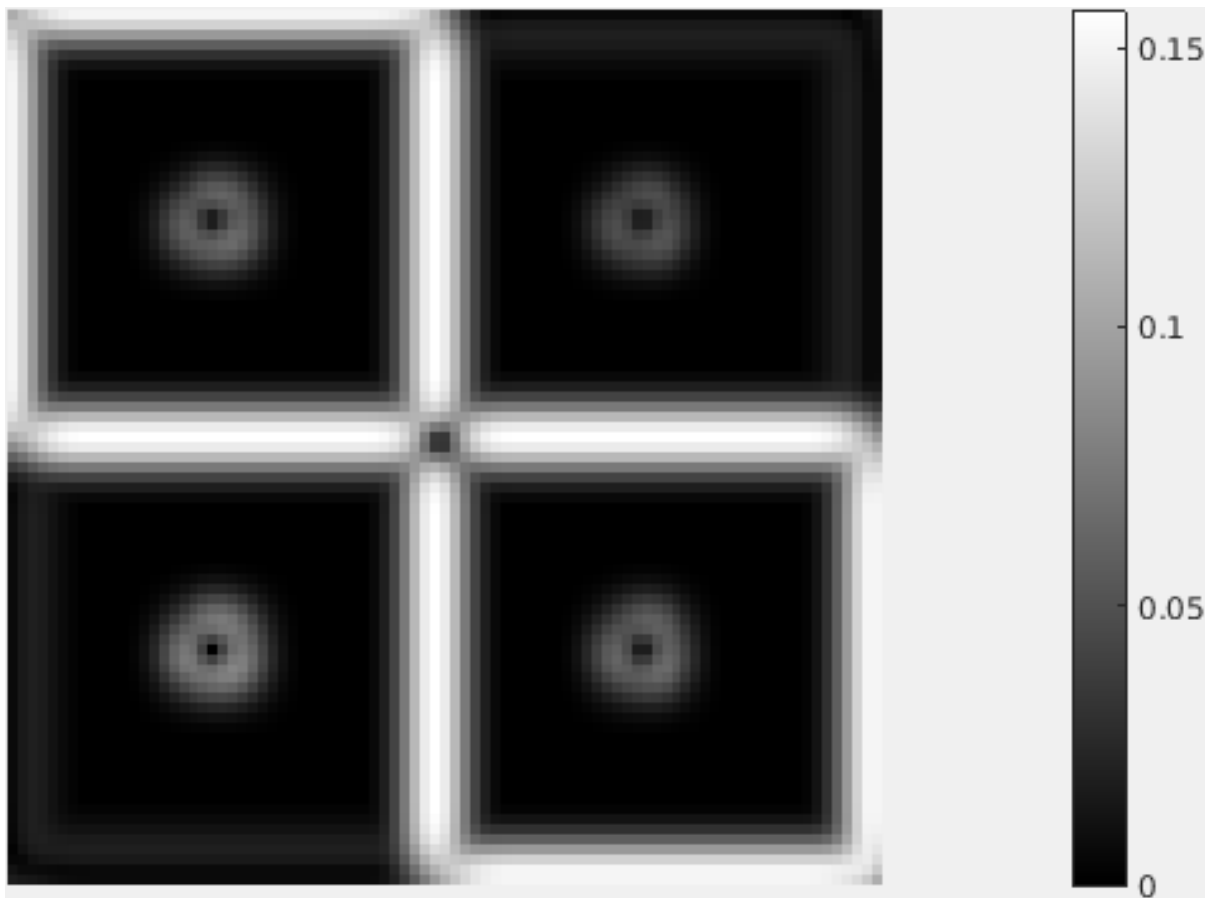
row=22, coloumn=45 0.12 ✓

You should have written code equivalent to the following:

```
figure(7), clf
g=fspecial('gaussian',15,2.5);
dgx=conv2(g,[-1,1],'valid');
dgy=conv2(g,[-1;1],'valid');
subplot(2,2,1), mesh(dgx);colormap('jet');
subplot(2,2,2), mesh(dgy);colormap('jet');
Icdgx=conv2(Ic,dgx,'same');
Icdgy=conv2(Ic,dgy,'same');
subplot(2,2,3), imagesc(Icdgx); colormap('jet'), axis('equal','tight'); colorbar
subplot(2,2,4), imagesc(Icdgy); colormap('jet'), axis('equal','tight'); colorbar
figure(8), clf
Icdg=sqrt(Icdgx.^2+Icdgy.^2);
imagesc(Icdg),colormap('gray'); axis('equal'); colorbar
```

And produced images like this:





You should have obtained the numerical values, as follows (changing the variable names to match those used in your code):

Icdgx(22,41)

Icdgx(22,42)

Icdgx(22,43)

Icdgx(22,44)

Icdgx(22,45)

Observe that the output generated by convolving an image with a Gaussian derivative mask has high amplitude (either positive or negative) at locations where there are intensity discontinuities. These large magnitude outputs are blurred due to the blurring caused by the Gaussian. Furthermore, this blurring causes the outputs at the edges of the smaller features (i.e. the small black and white squares within each quadrant of the

image) to have smaller amplitude than the outputs at the edges of the larger features (i.e. the large black and white squares forming each quadrant of the image).

Question 6

Correct

Mark 1.00 out of 1.00

Edge Detection with Laplacian of Gaussian (LoG) Masks

Another method of combining a derivative mask with a smoothing mask for edge detection is to combining the omni-directional second-derivative mask (the Laplacian) with a Gaussian which results in a Laplacian of Gaussian mask.

Create two Laplacian of Gaussian masks by convolving a 2-D Gaussian (one with a standard deviation 1.5 and another with a standard deviation of 5) with the smaller amplitude Laplacian from the section "Differencing with Laplacian Masks". Ensure that size of the resulting masks is sufficiently large to accurately represent the LoG functions.

Generate `mesh` plots of the two Laplacian of Gaussian masks you have created, put these as `subplot(2,2,1)` and `subplot(2,2,2)` in a figure. Convolve the boxes image with each of the Laplacian of Gaussian masks you have created. Display images showing the output of these two convolutions as `subplot(2,2,3)` and `subplot(2,2,4)` in the same figure window.

Repeat the above using the rooster image instead of the boxes image.

EXERCISE:

For the output produced by convolving the boxes image with the LoG (produced using the Gaussian with the smaller standard deviation), report the values at the following locations (correct to 2 decimal places):

row=22, column=41 ✓

row=22, column=42 ✓

row=22, column=43 ✓

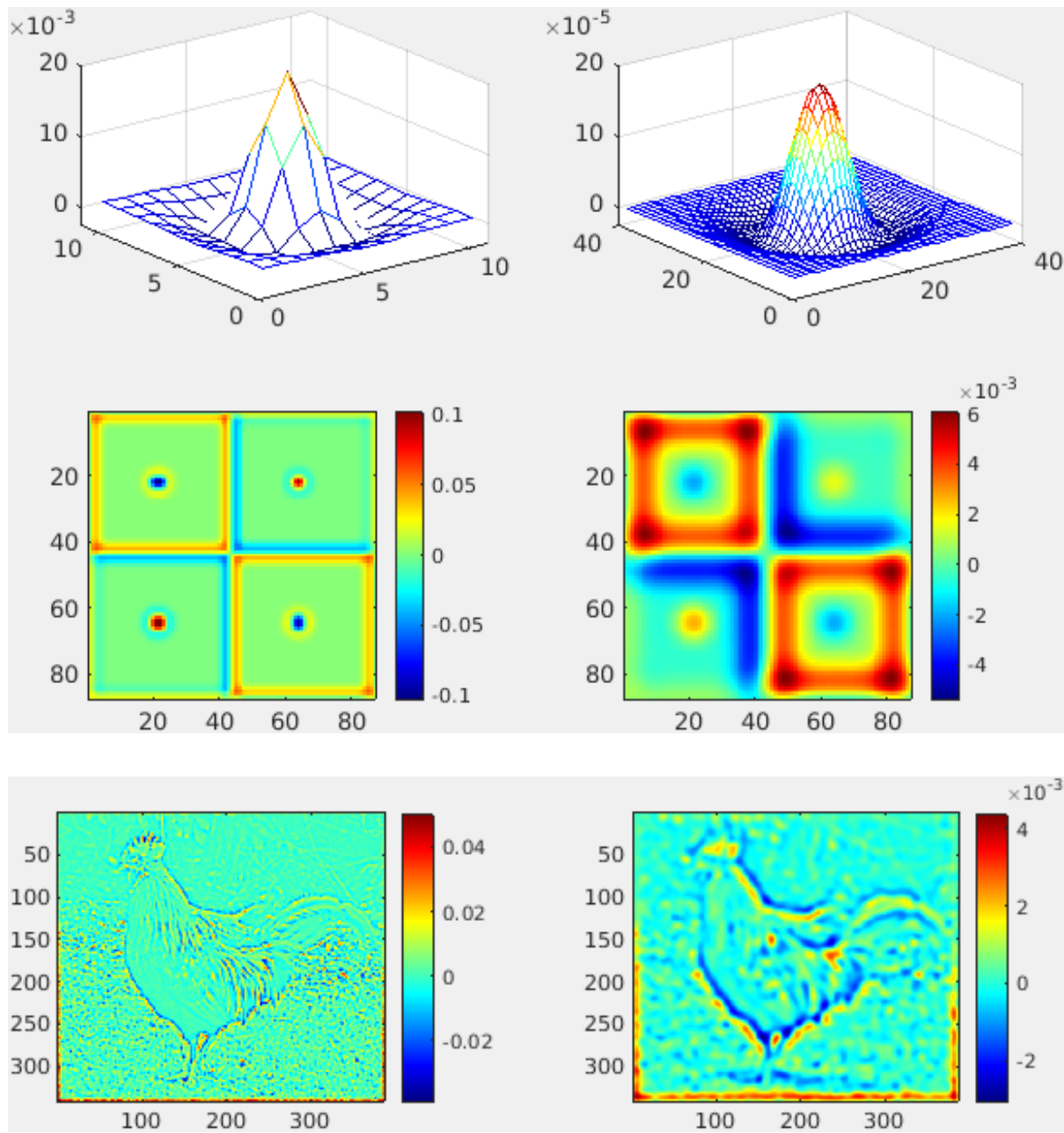
row=22, column=44 ✓

row=22, column=45 ✓

You should have written code equivalent to the following:

```
figure(9), clf
g1=fspecial('gaussian',13,1.5);
g2=fspecial('gaussian',41,5);
g1l=conv2(g1,laplacian,'valid');
g2l=conv2(g2,laplacian,'valid');
subplot(2,2,1),mesh(g1l)
subplot(2,2,2),mesh(g2l)
Icg1l=conv2(Ic,g1l,'same');
Icg2l=conv2(Ic,g2l,'same');
subplot(2,2,3), imagesc(Icg1l), axis('equal','tight'), colorbar
subplot(2,2,4), imagesc(Icg2l), axis('equal','tight'), colorbar
colormap('jet');
figure(10), clf
Iag1l=conv2(Ia,g1l,'same');
Iag2l=conv2(Ia,g2l,'same');
subplot(1,2,1),imagesc(Iag1l),axis('equal','tight'),colorbar
subplot(1,2,2),imagesc(Iag2l),axis('equal','tight'),colorbar
colormap('jet')
```

And produced images like this:



Note that if the masks are too small (the values around the edges, as seen on the mesh plots, are not close to zero) they cut off the edges of the function, and hence, do not produce the expected results.

You should have obtained the numerical values, as follows (changing the variable names to match those used in your code):

```
Icg2l(22,41)
```

```
Icg2l(22,42)
```

```
Icg2l(22,43)
```

```
Icg2l(22,44)
```

```
Icg2l(22,45)
```

Observe that the output generated by convolving an image with a the LoG mask has high amplitude (either positive or negative) either side of an intensity discontinuity, and that the amplitude is close to zero at the location of the discontinuity. This is similar to the result generated by the Laplacian mask, but the outputs are more spread out due to the blurring caused by the Gaussian. Furthermore, this blurring causes the outputs at the edges of the smaller features (i.e. the small black and white squares within each quadrant of the image) to have smaller amplitude than the outputs at the edges of the larger features (i.e. the large black and white squares forming each quadrant of the image). Increasing the standard deviation increases the blurring and reduces the amplitude of the output at small features. Hence, for the rooster image, when a small standard deviation is used, fine edges (such as those between some of the feathers) produce high amplitude outputs. Whereas, when a large standard deviation is used, only the more prominent edges (such as the boundary of the rooster) produce high amplitude output.

Question 7

Correct

Mark 1.00 out of 1.00

Multi-Scale Representations: The Gaussian Image Pyramid

A Gaussian image pyramid is a multiscale representation of a single image at different resolutions. The first image in a Gaussian image pyramid is the original image. Subsequent images in the pyramid are created by recursively convolving the current image with a Gaussian mask and down-sampling.

Create a four level Gaussian image pyramid of the rooster image, using a Gaussian with a standard deviation of 1.5, using "same" as the shape parameter for the `conv2` function, and resizing by a scale factor of 0.5 using nearest-neighbour interpolation. Display the images in the pyramid as four subplots in the same window.

EXERCISE:

For the fourth level of the pyramid report the values at the following locations (correct to 2 decimal places):

row=5, coloumn=3 ✓

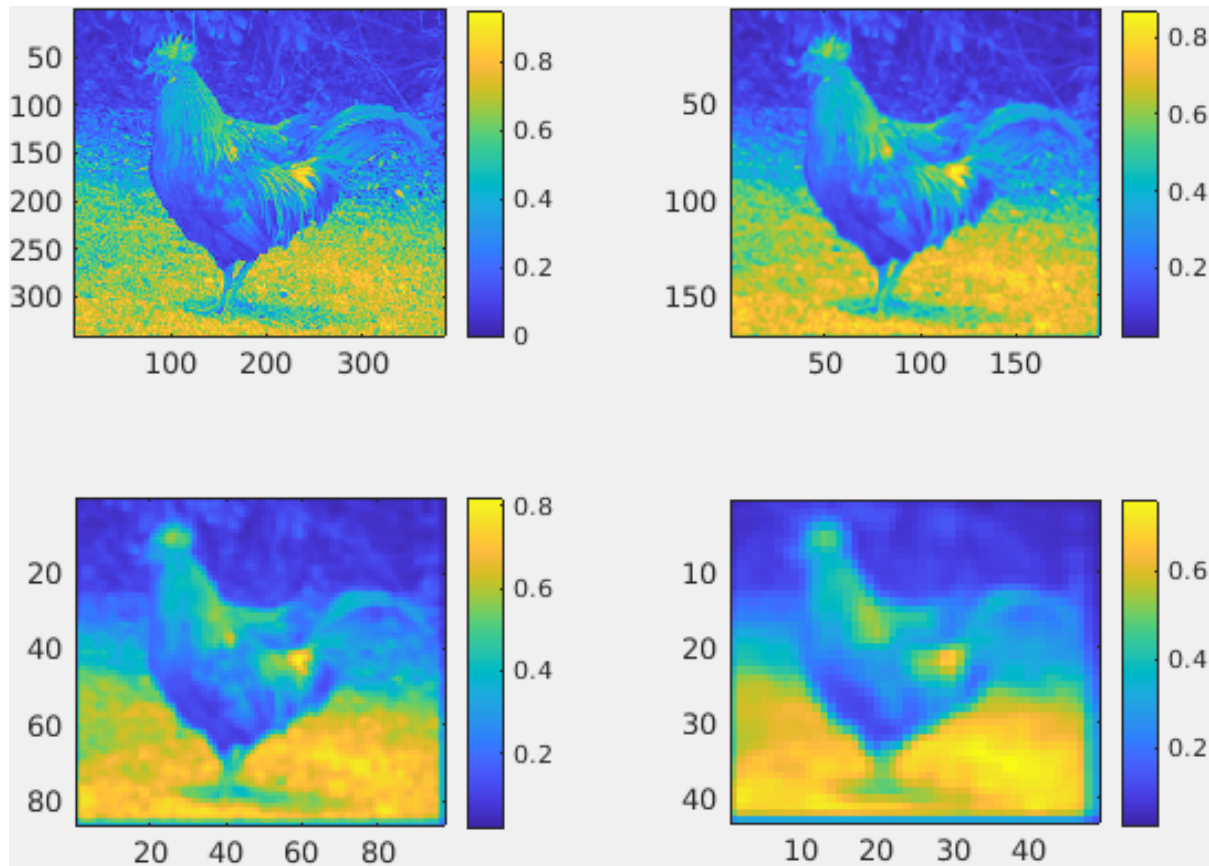
row=14, coloumn=15 ✓

You should have written code equivalent to the following:

```
figure(11), clf
g=fspecial('gaussian',9,1.5);
IpyrG=Ia;
subplot(2,2,1),imagesc(IpyrG); axis('equal','tight'),colorbar
for i=2:4
    IpyrG=imresize(conv2(IpyrG,g,'same'), 0.5, 'nearest');
    subplot(2,2,i),imagesc(IpyrG); axis('equal','tight'), colorbar
```

```
end
```

And produced images like this:



You should have obtained the numerical values, as follows (changing the variable names to match your code):

```
IpyrG(5,3)
```

```
IpyrG(14,15)
```

Question 8

Correct

Mark 1.00 out of 1.00

Multi-Scale Representations: The Laplacian Image Pyramid

A Laplacian image pyramid is a multiscale representation of a single image that highlights intensity discontinuities at multiple scales. It is obtained by convolving an image with a Gaussian mask and subtracting the smoothed image from the original one. The next level of the pyramid is obtained by repeating this process on the smoothed image after it is down-sampled.

Create a four level Laplacian image pyramid of the rooster image, using a Gaussian with a standard deviation of 1.5, using "same" as the shape parameter for the `conv2` function, and resizing by a scale factor of 0.5 using nearest-neighbour interpolation. Display the images in the pyramid as four subplots in the same window.

EXERCISE:

For the fourth level of the pyramid report the values at the following locations (correct to 2 decimal places):

row=5, column=12 ✓

row=10, column=10 ✓

You should have written code equivalent to the following:

```
figure(12), clf
```

```
g=fspecial('gaussian',9,1.5);
```

```
IpyrG=Ia;
```

```
for i=1:4
```

```
    IpyrGsmooth=conv2(IpyrG,g,'same');
```

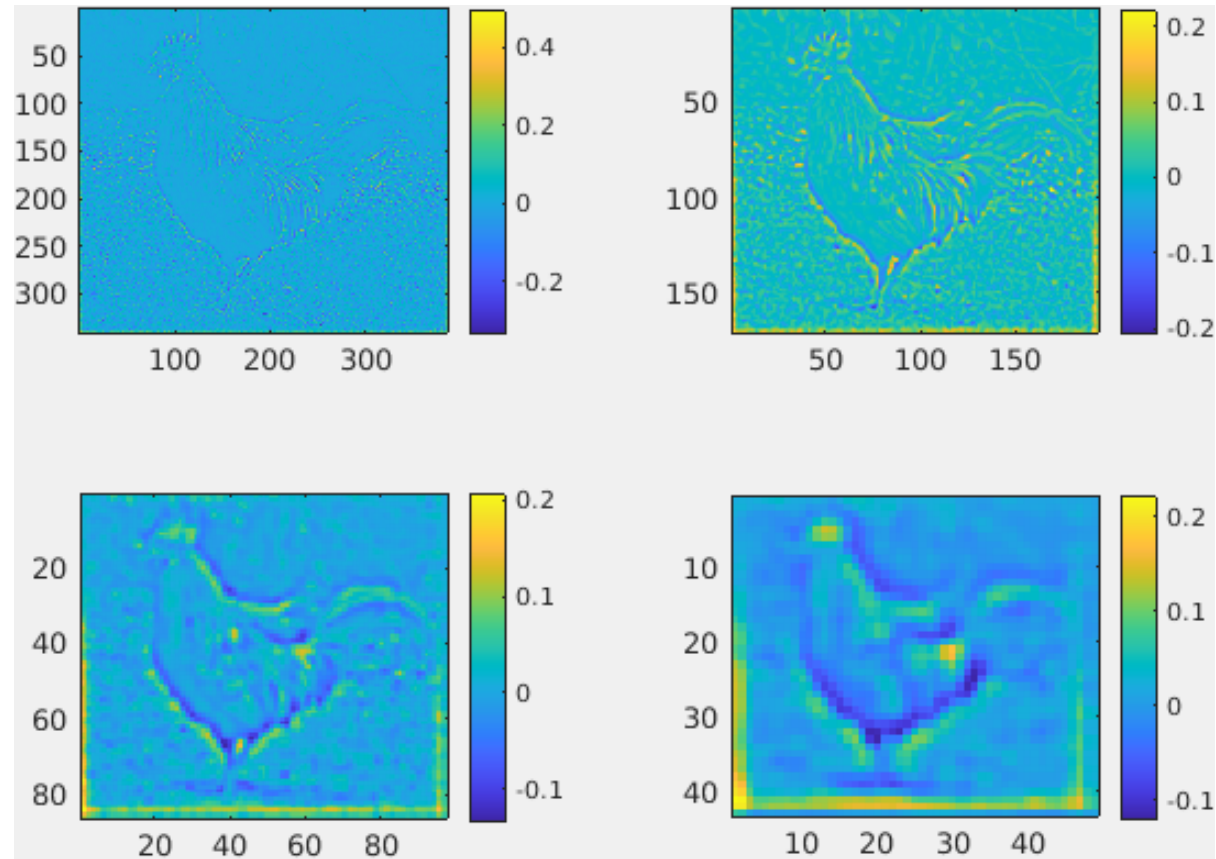
```
    IpyrL=IpyrG-IpyrGsmooth;
```

```
subplot(2,2,i),imagesc(IpyrL); axis('equal','tight'),colorbar
```

```
IpyrG=imresize(IpyrGsmooth, 0.5, 'nearest');
```

```
end
```

And produced images like this:



You should have obtained the numerical values, as follows (changing the variable names to match your code):

```
IpyrL(5,12)
```

```
IpyrL(10,10)
```

