Low-Level Computer Vision with MATLAB

1. Introduction

This laboratory session provides the opportunity to experiment with a number of image processing techniques used for low-level computer vision. The hours timetabled for this laboratory are insufficient to complete it. You should therefore work on this in your own time, and use the timetabled hours to get help and clarification.

This laboratory is assessed. A type-written report needs to be submitted online through KEATS by the deadline specified on the module's KEATS webpage.

> Boxes like this one indicate instructions that you need to carry out, or commands you need to execute, in MATLAB.
>
> 1.0.1: Boxes like this one describe what is required for your report.
>
> Note that it is not necessary to write a formal report (one with an abstract, introduction, methods, results and discussion), you simply need to provide the information requested in boxes like this one. You also do not need to provide your code, except where you are specifically asked to do so.

Note that some sections do not require you to report any results, and carry no marks. These optional exercises are there only to enhance your education.

I suggest you write a MATLAB script file, containing all your work, so that you can easily save your work and return to it at a later time.

This coursework requires you to use the rooster.jpg, boxes.pgm, woods.png and elephant.png images, all of which can be downloaded from the the module's KEATS page. These files should be copied to the directory in which you are running MATLAB. Once you load these images into MATLAB convert them from integer format to double precision floating-point format.

2. Convolution

In MATLAB, 2-D (and 1-D) convolution is performed using the function `conv2(I,H,shape)`, where `I` is a matrix representing an image, `H` is a matrix representing the mask, and `shape` is a parameter defining the size of the output. To perform convolution we need to convert an image grayscale (or apply the convolution to a single channel of a colour image).

> Execute the command:
> `help conv2`
> Determine what value of parameter "shape" is required to generate an output image the same size as the input image.

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 box (or average) masks and Gaussian masks.

3. Smoothing Masks

3.1 Box masks

A box, or average, mask is just an array of equal numbers that sum to one. Such a filter 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 both these masks so that the resulting four images 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.
>
> 3.1.1: In your report include a print out of the figure with four subplots that you have just created. Briefly describe and explain the results you have obtained. [4 marks]

3.2 Gaussian mask

Repeat the preceding experiment 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.

3.2.1: In your report include a print out of the figure with four subplots that you have created using Gaussian masks. Briefly describe and explain the results you have obtained, and compare these results with those obtained using the box masks. [4 marks]

3.3 Separable masks

Using the command `fspecial` create a 1-D Gaussian mask of size [1, 60] and with standard deviation 10.

Convolve this mask with the **transpose** of itself (using 'full' as the shape parameter in the `conv2` command) in order to generate a 2-D Gaussian with standard deviation 10.

(a) Use the `tic` and `toc` commands to measure the total time taken to convolve the rooster image with the 1-D Gaussian, and to convolve the result of that convolution with the **transpose** of the 1-D Gaussian.
(b) Use the `tic` and `toc` commands to measure the time taken to convolve the rooster image with the 2-D Gaussian.

You should find that (a) and (b) produce the same result (to within the floating point quantization error of the computer) but that (a) takes much less time than (b).

4. Difference Masks

4.1 Difference Masks (1-D)

To gain insight into the effects of difference masks we will first look at their effects on a simple 1-D signal.

Execute the following commands:

```
y=sin([0:0.01:2*pi]); subplot(3,1,1), plot(y);
yd1=conv2(y,[-1,1],'valid'); subplot(3,1,2), plot(yd1)
yd2=conv2(y,[-1,2,-1],'valid'); subplot(3,1,3), plot(yd2)
```

4.1.1: Plot the results in your report. Briefly describe and explain the results you have obtained [2 marks]

4.2 Difference Masks (2-D)

In two dimensions a finite difference approximation to the second derivative is given by a Laplacian mask (strictly speaking the masks below are the additive inverse of the Laplacian, and hence approximate the minus of the second derivative).

Create two Laplacian masks with different amplitudes, i.e.:

```
-1/8 -1/8 -1/8            -1  -1  -1
-1/8   1  -1/8     and    -1   8  -1
-1/8 -1/8 -1/8            -1  -1  -1
```

Convolve the boxes image with both these masks, and show the results.

4.2.1: In your report include the resulting images and note any difference in the two images that result from these two convolutions. [2 marks]

4.3 Other Difference masks

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

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 as `subplot(2,2,1)` and `subplot(2,2,2)` in the same window. Use a "jet" colormap and provide a colorbar.

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 as `subplot(2,2,3)` and `subplot(2,2,4)` in the same window. Use a "jet" colormap and provide a colorbar.

5. Edge Detection

To perform edge detection (i.e. to locate intensity discontinuities in an image), first and second order directional 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, whereas combining the omni-directional second-derivative mask (the Laplacian) with a Gaussian results in a Laplacian of Gaussian mask.

5.1 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 5) once with the mask [-1,1] and once with the transpose of this mask. Ensure the size of the Gaussian mask is sufficient so that the resulting Gaussian derivative masks are accurately represented.

Generate a `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 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 window. Use a "jet" colormap and provide a colorbar.

5.1.1: In your report include a print out of the figure with four subplots that you have created. What is the value of the convolved image at the locations of large intensity discontinuities (i.e. at edges)? [2 marks]

Repeat the above steps using a Gaussian with standard deviation 1.5 (rather than 5).

5.1.2: In your report include a print out of the second figure with four subplots that you have created . What is the value of the convolved image at the locations of large intensity discontinuities (i.e. at edges)? [2 marks]

To combine the horizontal and vertical edge images into a single image showing intensity-level discontinuities in any direction, we can calculate the L2-norm as follows:

```
Ibdg=sqrt(Idgx.^2+Idgy.^2);
```

The above assumes that you have given your image convolved with the horizontal Gaussian derivative mask the variable name "Idgx", and that you have given your image convolved with the vertical Gaussian derivative mask the variable name "Idgy".

Create a image showing the L2-norm generated from the Gaussian derivative mask produced using a Gaussian with standard deviation 1.5 for (a) the boxes image, and (b) the rooster image. Use the "gray" colormap.

5.1.3: In your report include a figure with both these images as two subplots. [2 marks]

5.2 Laplacian of Gaussian (LoG) mask

Create a Laplacian of Gaussian mask by convolving a 2-D Gaussian (with standard deviation 1.5) with the smaller amplitude Laplacian from section 4.2 (use 'valid' as the shape parameter in `conv2`).

Generate a `mesh` plot of the Laplacian of Gaussian mask you have created, put this as `subplot(2,2,1)` in a figure.

Convolve the boxes image with the Laplacian of Gaussian mask you have created.

Display images showing the output of this convolution as `subplot(2,2,3)` in the same window. Use a "jet" colormap and provide a colorbar.

Repeat the above steps using a Gaussian with standard deviation 5 (rather than 1.5), putting the mesh plot of the mask as `subplot(2,2,2)` and the convolved image as `subplot(2,2,4)`

5.2.1: In your report include a print out of the figure with four subplots that you have created. What is the value of the convolved image at the locations of large intensity discontinuities (i.e. at edges)? What is the value of the convolved image near to image locations of large intensity discontinuities (i.e. near edges)? [2 marks]

Convolve the rooster image (converted to grayscale, with double data type) with each of the two Laplacian of Gaussian masks you created above, and display the resulting output images. Ensure each subplot has a colorbar.

5.2.2: What effect does changing the standard deviation of the Gaussian used to create the Laplacian of Gaussian mask have on the result? [2 marks]

5.3 Gabor masks (only attempt after lecture 4)

Gabor masks can also be used for edge detection.

The m-file gabor2.m (available from KEATS) provides a function that will generate a 2D gabor mask. Copy this m-file into the directory where you are executing MATLAB.

Use the commands:
```
help gabor2
imagesc(gabor2(10,15,90,1,0))
```
(and variations on the numerical values in the latter command) to explore the effects of the different parameter values.

Using a Gabor with a specific phase and orientation will detect edges with those properties. Equivalently, this can be considered as a simple model of the responses of all V1 simple cells with the same stimulus preferences but different RF locations.

Convolve the elephant image with a Gabor mask generated with the following parameters:
`gabor2(4,8,90,0.5,0);`
Use 'valid' as the shape parameter in `conv2`).

Produce an image of the output.

Combining the outputs of Gabor masks at different phases can be used to detect edges regardless of phase (the contrast polarity of the intensity discontinuity). Equivalently, this can be considered as a simple model of the responses of all V1 complex cells with the same orientation preference but different RF locations.

Convolve the elephant image with a second Gabor mask which has a phase of 90, but is otherwise identical to the previous mask.

These two masks form a quadrature pair. If we combine the output produced by convolving the image with these two masks, then the combined output will be invariant to the phase of the input. To combine them take the L2-norm of corresponding pixel values (i.e. calculate the square root of the sum of the squared responses of the two simple cells at each image location).

Convolve (using 'valid' as the shape parameter) the elephant image with both these masks and combine the two outputs using the L2-norm.

Produce an image of the result.

Combining the outputs of Gabor masks at different phases and orientations can be used to detect edges at all phases and orientations.

Model the output of complex cells at the following orientations
[0,15,30,45,60,75,90,105,120,135,150,165].

Combine the outputs of these complex cells by taking the maximum response at each pixel location. This can be done using the following command:
`max(Ic,[],3)`
Assuming Ic is a three-dimensional matrix, the third dimension of which represents different orientations.

Produce an image that shows the maximum response across orientations.

5.3.1: In your report include a print out of the three images that you have created in this section.

Include colorbars for each image. [6 marks]

5.4 Edge Detection Challenge

MATLAB provides a function edge that will apply one of several edge detection methods to an image, and return a binary image in which only "edges" (i.e. significant intensity discontinuities) have a value of 1. The edge detection methods available include several we have looked at in this laboratory session (e.g. Prewitt, Sobel, LoG) and a popular method (canny) that is derived from the Gaussian derivative method explored earlier. Below are three grey-scale images. Next to each image are four binary images showing the edges identified by four different human observers. Notice that human observers do not agree on what constitutes an edge.

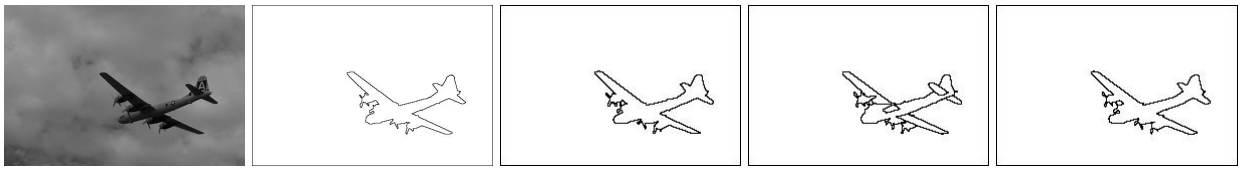
Load the grey-scale images into MATLAB.

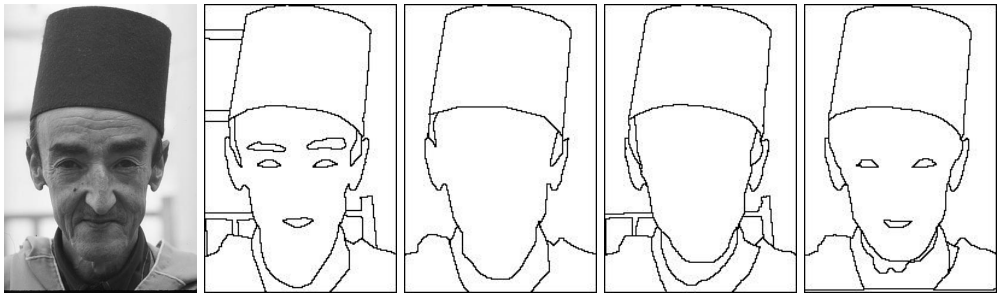
For one image, use the command edge to generate a binary edge image.

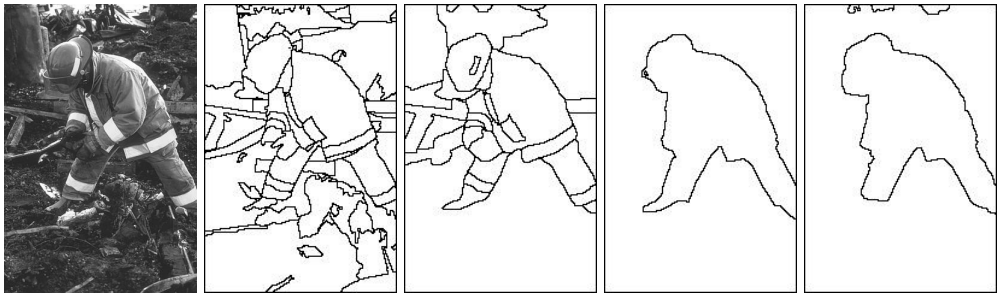
Spend 10 minutes experimenting with changing the parameters of the edge command to try to improve the edge image: try to find parameters that generate an edge image that includes true edges (ones that at least one human observer has marked as an edge), and excluded false edges (ones that no human observer has marked as an edge).

Without changing any parameters, perform the same edge detection on the other two images.

You should find that it is difficult to find parameters that work well for one image, and that these parameters generally produce poor results on other images. As a consequence it is very difficult to reliably and automatically find edges in arbitrary images.







6. Redundancy in Natural Images

Natural images exhibit a high degree of redundancy, i.e. the similarity between the intensity of a pixel at location (x, y) and that at a neighboring location (x + o, y) is inversly proportional to the distance (o) between these points. Matlab provides a command `corr2(A,B)` that can be used to calculate the correlation coefficient between the elements of two equally sized matrices (or image patches).

6.1 Measuring Redundancy

Execute the command:
`corr2(Ia,Ia)`
Where Ia is a matrix representing a grayscale image.

Now perform the same command with two parts of the same image that are shifted one pixel with respect to each other.

Write a simple MATLAB function or script that will automate the process of calculating the correlation

coefficient between overlapping parts of the same image shifted by different values. To produce accurate results ensure that the two image parts are as large as possible.

Using this program, calculate the correlation coefficient for values for shifts between 0 and 30 pixels for both the rooster and the woods image.

6.1.1: Plot a graph of shift vs correlation coefficient for both the rooster and the woods image. Include in your report a print out of this figure. Briefly describe and explain the results you have obtained. [4 marks]

6.2 Redundancy Reduction using DoG Masks

Retinal ganglion cells reduce redundancy. A simple model of the responses of retinal ganglion cells is provided by convolving an image with a radially symmetric Difference of Gaussians (DoG) mask. Off-centre, on surround RF types can be simulated, in addition to on-centre, off-surround types, by using the additive inverse of the DoG mask.

Create a Difference of Gaussians mask by subtracting one Gaussian (with standard deviation 6) from another Gaussian (with standard deviation 2).

Convolve each image with this mask.

Now repeat the process (described in section 6.1) of calculating the correlation coefficients for an image shifted by different amounts using the convolved images.

Repeat the above with a Difference of Gaussians mask generated by subtracting a Gaussian (with standard deviation 4) from another Gaussian (with standard deviation 0.5).

6.2.1: Plot graphs of shift vs correlation coefficient for both the rooster and the woods image after convolution with both the DoG masks. Include in your report a print out of this figure. Briefly explain why the results for the two DoG masks differ, also explain why the current results differ from those in the previous section. [4 marks]

Note that rather than convolving an image with a DoG mask, we can produce an identical result by convolving the image twice with two different Gaussian masks, and taking the difference between these two outputs.

The type of retinal ganglion cell simulated in this section processes intensity. In the next section, a different type of retinal ganglion cell, colour opponent cells, will be simulated.

7. Colour Detection

7.1 Colour Oppoent Cells

Generate an image with 4 subplots, with the subplots showing the response of the following centre-surround colour opponent cell combinations on the rooster image:
1. red-on, green-off
2. green-on, red-off
3. blue-on, yellow-off
4. yellow-on, blue-off

Ensure each subplot has a colorbar.

Use a Gaussian with standard deviation 2 for the centre, and a Gaussian with standard deviation 3 for the surround.

You will need to convolve the separate colour channels of the rooster image with Gaussians. Use a command like this:
```
Rg1=conv2(Ia(:,:,1),g1,'same');
```
Assuming Ia is an RGB image (stored using the double precision data type), this will generate a new image that is the convolution of the red channel (channel 1) with the mask g1.

You will also need to find the mean of two coulour channels. To do this use the following command:
```
mean(Ia(:,:,1:2),3)
```
Assuming Ia is an RGB image (stored using the double precision data type), this will generate a new image that is the mean of the red and green channels (i.e a yellow channel).

7.1.1: In your report include a print out of the figure with four subplots that you have just created. Briefly describe and explain the results you have obtained. [4 marks]

8. Multi-Scale Representations

8.1 Gaussian Image Pyramid

In order to generate an Gaussian image pyramid, we can perform the following command:
`Ia2 = imresize(conv2(Ia1,g,'same'), 0.5, 'nearest');`
Where Ia1 is an image at one scale and Ia2 is the image at the next larger scale in the pyramid, and g is a 2D gaussian with standard deviation 1. This command convolves Ia1 with a Gaussian with standard deviation 1, and then down-samples the result by a factor of 2.

Use this command recursively to create a 4 level Gaussian pyramid of the rooster image (converted to grayscale, with double data type), and display the outputs as four subplots in the same window. Ensure each subplot has a colorbar.

8.1.1: In your report include a print out of the figure with four subplots that you have created. [2 marks]

8.2 Laplacian Image Pyramid

With the aid of the lecture slides, create a 4 level Laplacian image pyramid of the rooster image (converted to grayscale, with double data type), and display the outputs as four subplots in the same window. Ensure each subplot has a colorbar.

8.2.1: In your report include a print out of the figure with four subplots that you have just created. Comment briefly on this result in comparison to the results from section 5.2. [4 marks]