Computer Vision
Workshop 4
Edges and their application

Aims of the workshop

The aim of this workshop is to introduce Gradient Images and Sharpening techniques, as described in lectures, but in a practical setting. These exercises will lean on the techniques shown from Filters & Convolutions as well as gradients and edges, including use of these to sharpen.

A template .mlx script is available under Canvas. This provides code to load the tricky ‘peppers_color.tif’ image.

No code will be provided in this workshop, please see “Useful Information” below on how to look up MATLAB functionality. The concept behind this workshop is about discovery, experimentation, and promoting critical thinking around the techniques covered so far in the module.

Feel free to discuss the work with peers, or with any member of the teaching staff.

Useful Information

MATLAB Documentation

MATLAB functionality mentioned during lecture or workshops is fully-documented and available at: https://uk.mathworks.com/help/matlab/

You may find the “Getting Started” section of this documentation to be useful: https://uk.mathworks.com/help/matlab/getting-started-with-matlab.html

In addition, MATLAB Answers may be useful in finding common solutions to any issues you may encounter:
https://uk.mathworks.com/matlabcentral/answers/help?s_tid=gn_mlc_ans_hlp

Useful functions: conv2, edge, rgb2hsv, ones, imfuse,

Reminder

We encourage you to discuss the contents of the workshop with the delivery team, and any findings you gather from the session.

Workshops are not isolated, if you have questions from previous weeks, or lecture content, please come and talk to us.

The contents of this workshop are not intended to be 100% complete within the 2 hours. This session involves substantial comparisons and critical thinking, and as such it’s expected that some of this work be completed outside of the session. Exercises herein represent an example of what to do; feel free to expand upon this and make your own comparisons, in addition, if you wish.

Note: In these workshops, and the ACW, you can work in either .m or a .mlx "live script". Feel free to experiment with using either, and see which you find most useful.

Exercise 1: Using your knowledge of convolution filters (conv2), write and run a first-order derivative convolution filter on RGB (3-channel) images from the 'subset' zip on canvas.

Consider the number of channels per image, and the colour space. Primarily how you might deal with this?

Convert **RGB** to **Greyscale**, and perform the convolution. Display the result.

Exercise 2: Take each channel of R, G, and B, and perform the convolution separately on each channel. Display all three convolution outputs, with an appropriate figure label.

How do these compare with each other? How does this output differ from part a?

Exercise 3: Convert the **RGB** images to **HSV**; perform the convolution on the three separate HSV channels individually; displaying them as before (with appropriate figure labels).

What results do you get from RGB vs HSV? Remember what these colour spaces do and how they treat chroma information and lightness.

Exercise 4: Perform the same convolution operation but with a 2D Laplacian Filter. Compare the difference between first and second-order derivative filters on the output; specifically, the impact on the original latent noise within the image. How do these gradient filters perform on your noisy images from last week?

Exercise 5: Using the built-in Canny Edge detector, extract edges from the standard set of images (cameraman, etc). How do these edges compare with the previous method/steps? Do the edges 'localise' well? I.e Do they align with the true edge in the original input image, or are they slightly off? Hint: You can overlap/blend the edge image over your input image using imfuse.

The Extended Exercises are optional, and are offered as an advanced supplement for those who have completed the existing work and wish to expand on their practical knowledge; challenging themselves further.

Extended Exercise 1: Using the concept of the Laplacian filter for horizontal, and vertical, edges:

1. Produce/Create a Laplacian filter/kernel which captures the diagonal running edges. Consider what the second derivative would be along those axes, and how we combined the horizontal and vertical components to make the single filter. What have the filters we've used so far summed up to? Is this important?
2. Run this edge detector on some images, comparing the differences with the standard horizontal+vertical Laplacian.

Extended Exercise 2: Write your own Unsharp sharpening function. For the blur stage of the algorithm, approximate a 2D Gaussian Distribution a kernel, with parameters for the mean and the sigma. Additionally, your function should allow, as input, the size of kernel to be used.