Computer Vision
Workshop 3
Noise Models & Manipulation

Aims of the workshop

The aim of this session is to introduce salt & pepper / Gaussian noise in a practical setting, including methods of adding and removing noise to images within MATLAB. These exercises will lean on the techniques shown in lectures for Filters and Convolution.

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: imread, imnoise, medfilt2, conv2 (or filter2)

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.

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: Click on the **Apps** tab from within MATLAB. Scroll down to **“Image Processing and Computer Vision Toolbox”**, select “**Colour Thresholder**”.

1. Load the Simple Red Car image from canvas - from our lectures. (Either as a file or from your workspace)
2. You should be presented with some colour models to choose from.
3. Click on RGB. You will be presented with histogram breakdowns for each channel, Red, Green, and Blue.
4. Drag the sliders at either end of the histograms to filter a range of values. Can you isolate just the red car body of the image?
5. Experiment with different colour models towards the task of segmenting Red.
6. Replace the red car image with “fancy red car” from Canvas. Try the same task again, is it more difficult? Discuss the difficulties with this task and how the two images may differ and why this impacts segmentation.

Exercise 2: Look at the documentation for the *imnoise* function. Apply Salt & Pepper noise to the provided image on canvas. Note specifically how the image changes as a function of noise amount.

Exercise 3: Produce a MATLAB script to compare the Gaussian Noise to Salt & Pepper Noise. Note the differences between each noise time (you may want to review our lecture slides / panopto for this). How do other forms of noise differ from these? What impact does Gaussian noise appear to have on the image?

Exercise 4: Download various images online with a wide range of textures, details, contrast, and see how the exact same noise (with same parameters) alters each of these images differently.

Exercise 5: Using built-in MATLAB functions, apply both a mean filter, and a median filter to the chosen noisy images. For this exercise you may utilise built-in functions, or methods of your choosing.

Hint: Consider which colour model you might use? Is the noise in each channel equal? Could a different colour model be beneficial when de-noising?

Hint2: You may wish to consult the documentation for medfilt2, and conv2. E.g We can apply a mean filter as a convolution process between our input image and a kernel of a certain size (and values). You may need to refer back to lecture slides for this.

1. Vary the neighbourhood size and see the impact this has on noise reduction (and information loss) on your image.
2. Which filter seems to work the best for each type of noise you have currently tried.

Extended Exercise 1: Using the methodology outlined within Lectures about sliding windows. Implement a 2D convolution function which accepts (as input) an image, and a NxM matrix; producing a resultant convolved image.

1. Consider what border / padding options are available, and how you can programmatically account for these. These can be hard-coded for now.
2. How does your implementation compare against the MATLAB conv2 function output? Produce an image difference between your function's output and that of the conv2 output.