ENGG1003 - Programming Assignment 2

Brenton Schulz

1 Instructions

Due: 5pm Monday June 8th (ie: Monday of Week 13 corrected for the extra week of break) - grading during your enrolled lab that week.

Weighting: **15%** of your final grade.

NB: This assignment can be completed in either MATLAB or the free alternative Octave. Demonstration may be done in either environment.

Task 1: (5 Marks) - Function Plotting

Write a MATLAB script which plots the following function:

$$y = (1 - e^{-kt})cos(2\pi ft) \tag{1}$$

for:

$$k = 0.7$$
$$f = 3.5$$
$$t = [0, 5]$$

The $[0, 5]$ syntax means "from 0 to 5, including both 0 and 5".

Your script should plot 1000 data points with the default line style and colours provided by the `plot()` function.

For full marks you also need to:

- Write *vectorised* code. You are not permitted to use any loops or conditional statements. Pay careful attention to MATLAB element-wise operator behaviour. **-1 if `for` loops etc are used.**
- Comment your code with a description of what the script does. **-0.5 marks if missing**
- The values of t *must* include both 0 and 5. **-1 mark if t stops close to, but not exactly on, 5.**
- Allow the script to be easily modified with new values for k, f, and t. **-1 mark if these are not clearly initialised and easy to modify** (eg: if they are hard coded into the equation instead of being variables).
- Annotate the plot with axis labels and a title. **-1 mark if missing**. Read the MATLAB help documentation on `xlabel()`, `ylabel()`, and `title()` to learn how to do this. The labels are to be:
  - x-label: "Time (s)"
  - y-label: "Voltage (V)"
  - Title: "Transient"

Task 2: (5 Marks) - Noisy Image Restoration

You have been provided with seven images named `IMG_5698.JPG` to `IMG_5704.JPG`. These images were taken in low light conditions with a very high ISO setting which results in "noise" on the image (random fluctuations in pixel intensity) and "bright pixels" (pixels which appear bright in every image, despite not being hit by light).

In addition to the seven source images you are also provided with a "dark frame", `dark.png`. This is a photograph taken with the lens cap installed and measures the "bright pixels" intensity.

Write a MATLAB script which performs the following tasks:

1. Calculates the mean of all seven images
2. Subtracts the dark frame from the "mean" image
3. Performs a *threshold* to blacken "nearly black but not quite" pixels. The threshold value, assuming a 0-255 intensity range, is to be 25. ie: any intensity value on any pixel channel less than 25 should be set to zero.
4. Displays the result.

The mean calculation and dark frame subtraction should be calculated on all channels *independently* so that the colour balance of the result is not affected.

Marking notes:

- Comments are not required for this task
- The images are all loaded correctly: **1 mark**
- The mean calculation is performed correctly: **2 marks**
- You will be awarded one of the following for vectorisation:
  - Fully vectorised code: **2 marks**
  - Unvectorised code: **1 mark**

  NB: a loop may be used to loop over the 7 images but the mean calculation and dark frame subtraction must be vectorised for full marks.

To save time, you may use the template below to load the seven images and convert the to the `double` data type for processing. Remember to convert back to `uint8()` or scale [0,1] for display with `image()`.

```
i1 = double(imread("IMG_5704.JPG"));
i2 = double(imread("IMG_5703.JPG"));
i3 = double(imread("IMG_5702.JPG"));
i4 = double(imread("IMG_5701.JPG"));
i5 = double(imread("IMG_5700.JPG"));
i6 = double(imread("IMG_5699.JPG"));
i7 = double(imread("IMG_5698.JPG"));
```

Task 3: (5 Marks) - Unsharp Mask Image Enhancement

Write a MATLAB script which processes the result of the previous task with an *unsharp mask* algorithm.

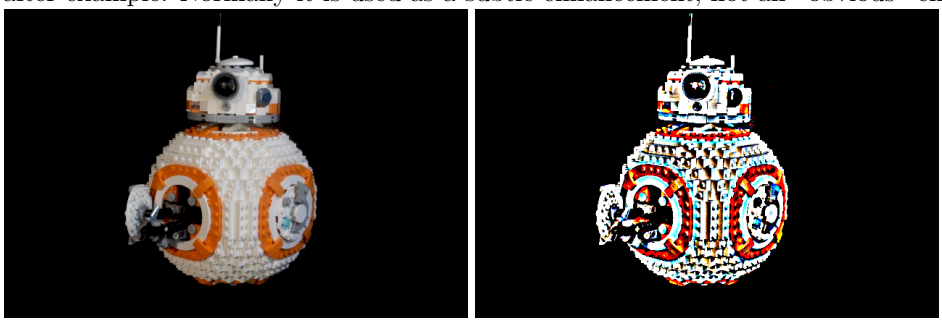
This algorithm requires the following processing steps:

1. Load the original image and convert it to `double()` for processing.
2. Create a copy and blur it. See the Week 10 lab for a suitable blurring algorithm or research a built-in MATLAB function which performs an image blur. Experiment with the "amount" of blur. **1 mark.**
3. Subtract the blurred copy from the original. This is done by applying the formula

$$y = original - blurred \tag{2}$$

4. Scale the intensity of y. ie: multiply the intensity of each pixel by, say, 0.1 or 10 or 100. Experiment with this value.
5. The output image is the blurred and intensity scaled data y plus the original image.
6. Display the result. A conversion to `uint8()` will clip any negative values or values over 255. You do not need to worry about this clipping. **3 marks.**
7. Compare results between processing the noise-reduced image to an original image. How susceptible is the unsharp mask process to image noise? Does it make the noise worse or suppress it?
8. Show your demonstrator the result of varying the parameters (blur amount and scaling of y) and the comparison between a "noisy" image and the "denoised" one. **1 mark.**

This effect is easy to over-do but for demonstration purposes the figures below provides an extreme before & after example. Normally it is used as a subtle enhancement, not an "obvious" effect.



More information about this algorithm, plus examples, can be found on Wikipedia: `https://en.wikipedia.org/wiki/Unsharp_masking`