Exercise 1, HT2020
Introduction

Assignment in partial fulfilment of the requirements for the course

Computer-Assisted Image Analysis I

UPPSALA
UNIVERSITET

Master's Programme in Computer Science

Department of Information Technology

Authors:
Abraham Mathewos Meja
Marcello Pietro Vendruscolo
Shubhomoy Biswas

November 9, 2020

Contents

List of Figures

1 Question 1

Where in the image is the pixel $(1,1)$ ***located and what is the graylevel value? In command window type*** $I(1,1)$***. Did you get the same value?***

The pixel $(1,1)$ is located at the top-left corner of the image. Its graylevel value is 89, as shown by the status bar of the Pixel Region window in Figure 1. Accessing the greylevel value through the command window by indicating the coordinates of the desired pixel, in this case $I(1,1)$, resulted in the same graylevel value of 89, as shown in Figure 2.

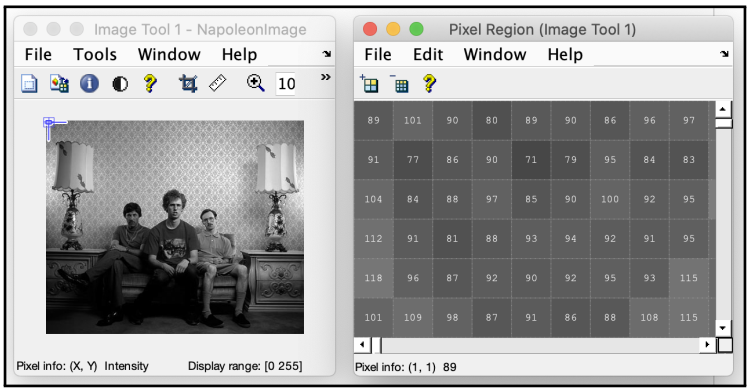

Figure 1: The Pixel Region window in this figure shows the pixel $(1,1)$ location at the top-left corner and its graylevel value, which can also be observed through the status bar.

```
>> I(1,1)

ans =

  uint8

   89
```

Figure 2: This figure shows the obtained graylevel value of pixel $(1,1)$ when using the command window in MATLAB.

2 Question 2

Explain what contrast and brightness are and include figures of the histograms of the images* napoleon.png, napoleon_light.png, *and* napoleon_dark.png. *Can you tell from the histograms which figure has the highest contrast and which figure is the brightest?

Brightness is a relative attribute of the human visual perception that regards the quantity of radiation or reflection of energy (i.e., light) emitted by a source in comparison to the original object (in our case, image). In terms of image intensity transformations, brightness can be understood as the value of a constant which can be added or subtracted from all the pixels' intensity values to either make the image brighter or darker, respectively.

Contrast makes a picture discernible and can be understood as the difference in intensity levels among the pixels composing an image. In terms of image intensity transformations, contrast can be understood as the slope value of the transform function of an image. This slope value defines the increase or decrease rate in intensity value with respect to the input pixels' intensity value.

From the shapes of the histograms in Figure 3, one can affirm that the NapoleonImage Light, corresponding to napoleon_light.png, is the brightest one. It contains no pixels with intensity values lower than approximately 100 while the other two images contain. While no bin with intensity lower than 100 is populated, there is a significant number of populated bins in the lighter region of the greylevel scale, according to the histogram. In addition, it is also possible to conclude that the NapoleonImage Original, corresponding to napoleon.png, has the highest contrast as its histogram shape (i.e., x-axis) stretches more extended than the histogram shape of the other two images.

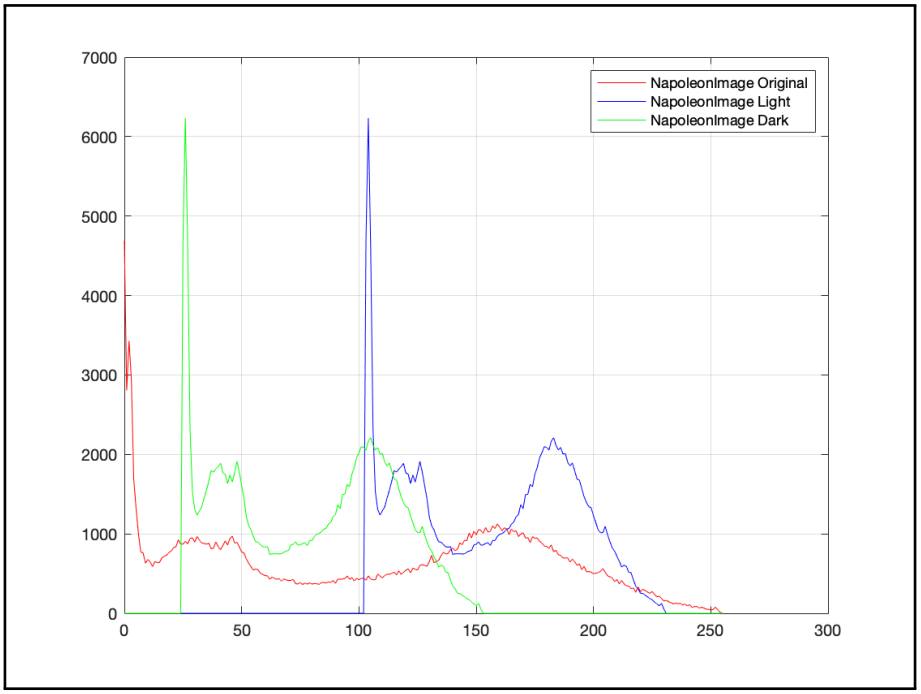

Figure 3: This figure shows the histogram of each and all the three images.

3 Question 3

Explain the difference of imagesc((I/64)*64 and imagesc((Is/64)*64), where I is uint8 and Is is single.

The datatype **uint8** in MATLAB corresponds to 8-bits unsigned integer variables ranging within the $[0, 255]$ interval while the datatype **single** corresponds to single-precision variables that are stored as 32-bit floating-point values.

In the first processing case (i.e., **imagesc((I/64)*64)**), when dividing the intensity values of the pixels in image I by 64, information of the original pixel values is lost, as such division returns an integer value. For example, dividing the pixel $f(3, 1)90$ by 64 results in 1. After the multiplication by 64, the intensity value is not the same as the original. However, in the second processing case (i.e., **imagesc((Is/64)*64)**), information is retained by the floating point precision. For example, dividing the same pixel $f(3, 1)90$ by 64 results in 1.41. Hence, multiplying it again by 64 outcomes the original pixel intensity value. Figure 4 shows the visual outcome of both commands.

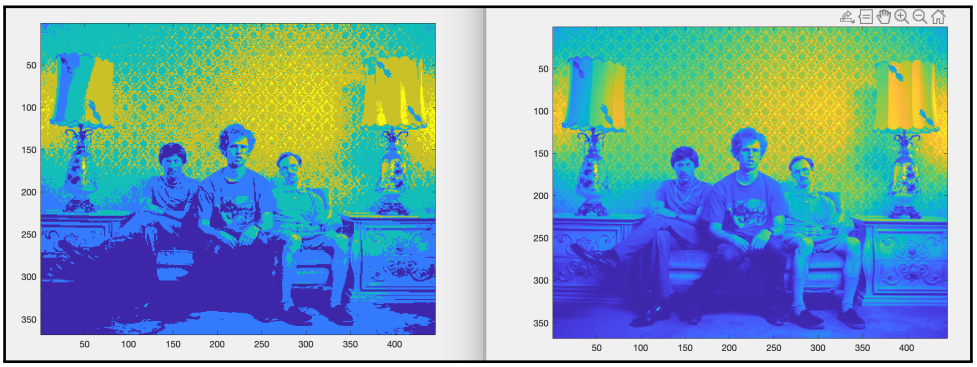

Figure 4: The picture on the left is imagesc((I/64)*64) and the picture on the right is imagesc((Is/64)*64).

4 Question 4

Demonstrate a mathematical expression involving I to make it brighter.

$$g(x, y) = i(x, y) + C$$

$$G = I + 30$$

The addition of 30 to I, where I denotes the image's pixels, increases or shifts each pixel's intensity by 30 units. It is important to ensure that the value of pixels are handled adequately, as such operation may result in intensities beyond the upper-limit interval boundary value of 255 when working with uint8 variables.

Figure 5 shows an original image with it's intensity histogram and Figure 6 shows the corresponding brighter image. It is possible to observe that the histogram of the brighter image is shifted by 30 units.

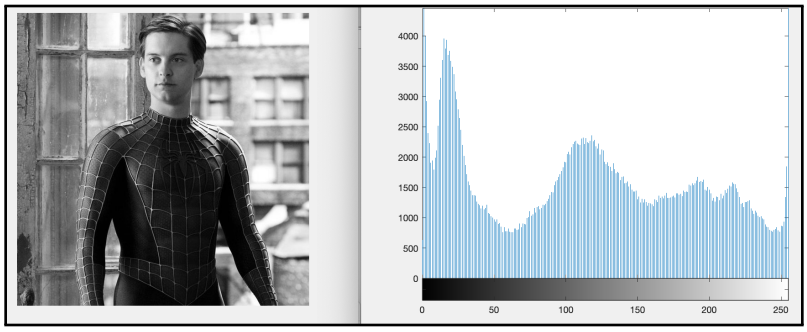

Figure 5: This figure shows an original image and its histogram.

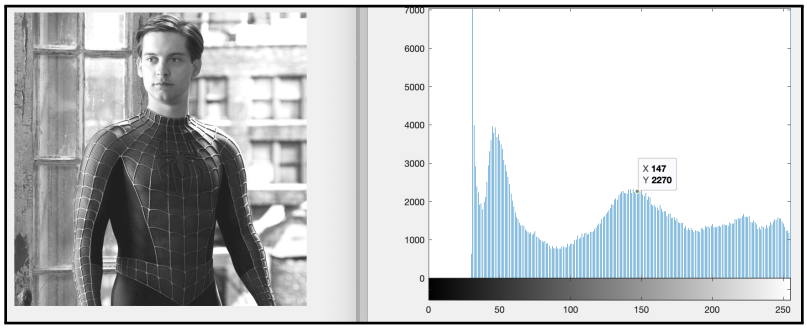

Figure 6: This figure shows the output image after adding brightness value of 30 to the image in Figure 5 and the resulting histogram.

5 Question 5

Demonstrate a mathematical expression involving I to give it lower contrast.

$$g(x, y) = i(x, y) \times C$$

$$G = I \times 0.7$$

The multiplication of I by 0.7, where I denotes the image's pixels, decreases the difference between the pixel intensities, thus making the foreground and the background of the image less distinguishable.

Figure 7 shows the original image with it's pixel intensity histogram and Figure 8 shows the corresponding image with a lower constrast.

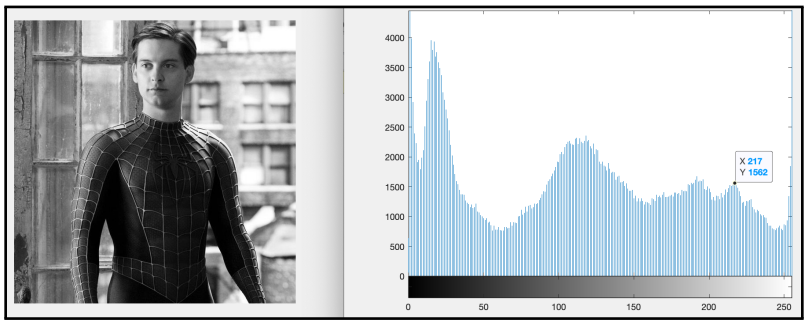

Figure 7: This figure shows an original image and its histogram.

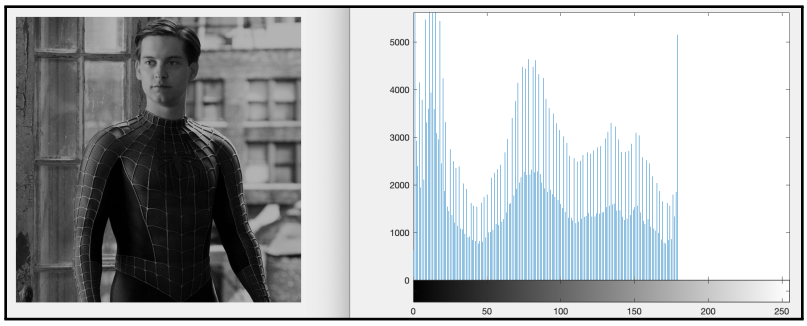

Figure 8: This figure shows the output image after multiplying the image in Figure 7 by 0.7 and the resulting histogram. The output image has lower contrast.

6 Question 6

Use $g = 2$ ***and*** $g = 12$ ***and explain the resulting images.***

The Gamma Transformation function $g(x, y) = C \times f(x, y)^{\gamma}$ for different values of γ is shown in Figure 9.

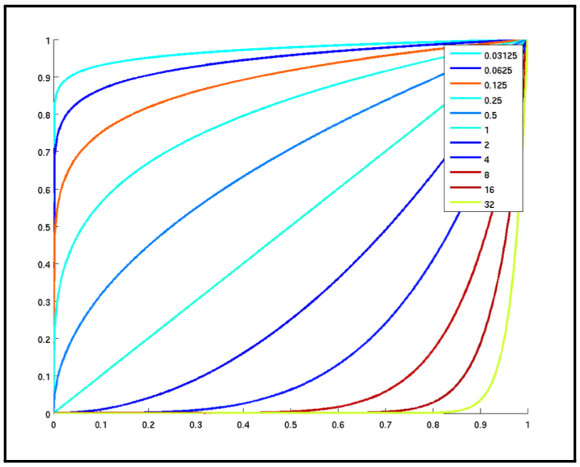

Figure 9: This figure shows the Gamma transformation function for several values of γ. Image retrieved from the lecture material.

From the function shapes, it is possible to understand that Gamma transformation functions with $\gamma > 1$ stretches the lighter regions of the original input picture while it compresses the darker regions. However, on the other hand, for $\gamma < 1$ the opposite happens; the transformation function stretches the darker regions of the original input image while it compresses the lighter regions.

The histogram of the original Napoleon image is observed in Figure 10. The result of using $\gamma = 2$ can be observed in Figure 11. It is possible to observe that the outcome of the transformation is a darker image, as this was the compressed region. On the other hand, the result of using $\gamma = 0.5$ can be observed in Figure 12. It is possible to observe that the outcome of the transformation is a brighther image, as this was the compressed region.

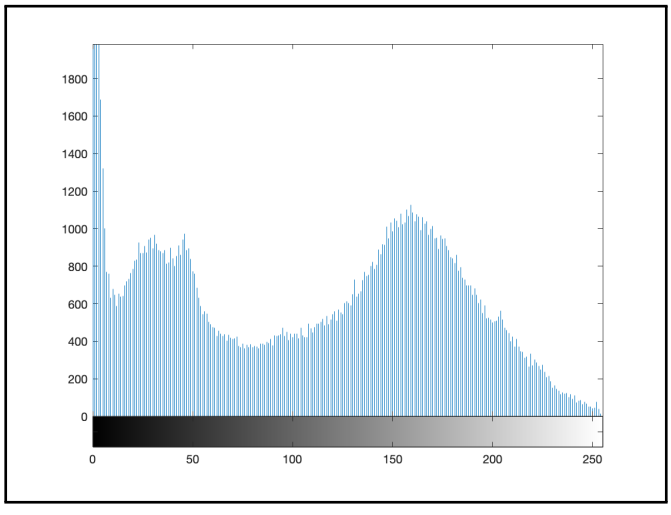

Figure 10: Histogram of the original napoleon.png image.

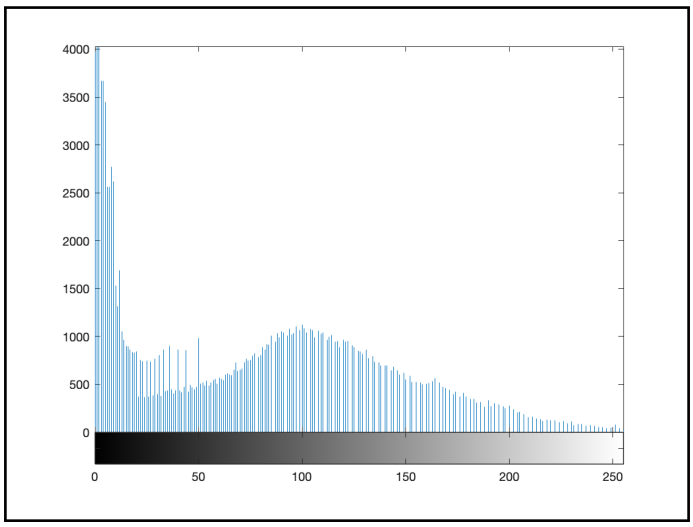

Figure 11: Histogram after Gamma transforming the input image with $\gamma = 2$.

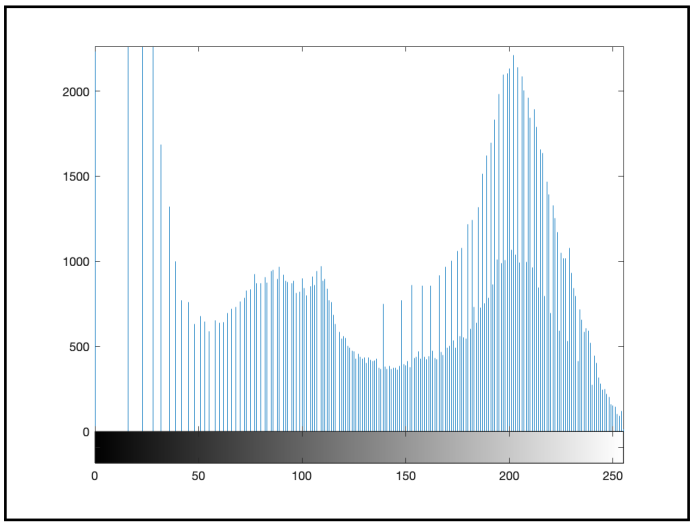

Figure 12: Histogram after Gamma transforming the input image with $\gamma = 0.5$.

7 Question 7

Explain how histogram equalization works in theory. Include histograms of one of the images before and after applying equalization in your report and explain what you see. Do the changes to the histograms and the images agree with the theory of histogram equalization?

The theoretical concept of histogram equalization is to evenly distribute an image's intensity values throughout its entire histogram. This transformation flattens the histogram and is often utilised for automatic enhacement of contrast. It transforms regions of lower contrast into regions of higher contrast levels, and often enhances the image if the background and the foreground have both dark and lighter values. However, it is valuable to mention that the outcome image of a histogram equalization transformation does not always correspond to the expectations (i.e., is better than the input image).

The transformation is executed by first calculating the ***cumulative distribution function (cdf)***, which is the cummalative summation of all the probabilities of the number of pixels in each grayscale level. Then, the intensity transfer function is calculated by multiplying the *cdf* with the total number of grayscale levels - 1. This enables the mapping of the new grayscale values into the number of pixels.

The following two figures, Figure 13 and Figure 14, show the Napoleon image and its corresponding histogram before and after applying histogram equalization, respectively.

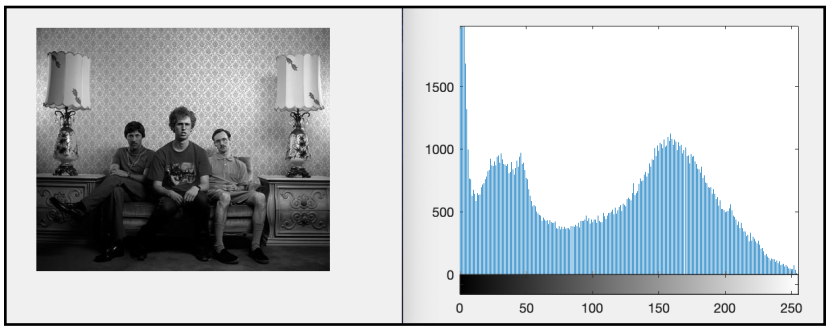

Figure 13: Image and histogram before equalization.

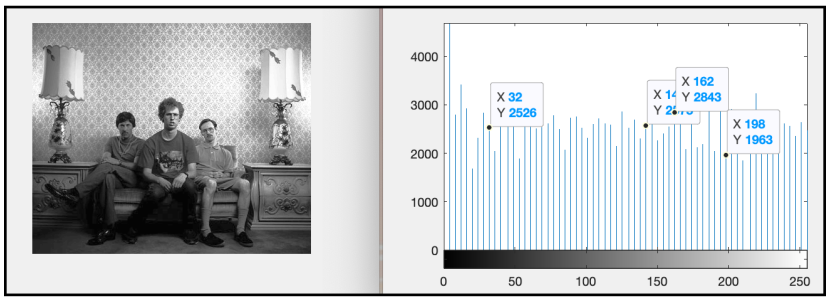

Figure 14: Image and histogram after equalization.

From the figures, it is possible to observe that applying histogram equalization made the darker regions of the image slightly brighter to get an even distribution of dark and light pixels. This observation can be confirmed by analysing the images' respective histogram distribution, as expected according to the theory explanation.

8 Question 8

Explain the role of the interpolation method as well as the role of the lowpass-filter. Which combination of options do you prefer? And why?

Image **interpolation** is a relevant technique when resizing or remapping (i.e., resampling) an image where it is transformed from one grid system to another. Interpolation works by estimating the intensity values in the new grid system according to the data in the existing grid system. There are different methods for interpolation, such as Nearest-Neighbour and Bilinear, and the suitability of each method depends on the characteristics of the input image as well as the desired outcome.

In image processing, **Low-pass filtering** is a type of anti-aliasing filter that allows frequencies lower than a defined threshold frequency value to pass unchanged while it filters (i.e., decreases) frequency values that are higher than that threshold value. These higher frequency values are usually the edges of the objects in a picture, corresponding to sharp changes in intensity values. Thus a low-pass filtering blurs the image on the edges. One way to remove high-frequencies in the spatial domain is to average out the pixel intensity by its neighbouring pixels (i.e., box or Gaussian kernels).

From testing the different combinations of interpolation methods and antialiasing options in the cameraman.png image, the preferred image outcome was produced by the combination of bilinear interpolation with the antialiasing option set to true. This image outcome was smoother and more accurately resembled the input image in comparison to the images produced by the other combination possibilities. Resampling the input image with the nearest neighbour method or with the antialiasing option set to false produced images that greatly differ from expectations of reality.

9 Question 9

Can you give a real-life example of aliasing outside the area of image analysis and signal processing in general? Have you seen this phenomenon before?

There are real-life situations where aliasing can be observed outside the field of signal processing and image analysis. The first example to be mentioned is when humans observe the blades of a helicopter as it speeds up their circumrotation. The human brain samples the movement of the blades slower than their real motion and, therefore, the rotation speed may be perceived to be slower than it really is. A second example, slightly related to the field of digital images, is when cars traveling along a road are filmed. In such videos, the wheels seem to be rotating backwards because cameras recording the video often undersample the rotation of the wheels.

10 Question 10

How can a "standard" healthy brain, or a mean image, of the two images brain1.png and brain2.png be constructed? In your report include a figure showing the standard brain.

The "standard" or the mean image from the two healthy brains, brain1.png and brain2.png, can be constructed by averaging (i.e., weighted summing) the intensity values of corresponding pixels from the two images. Figure 15 shows the brain1.png and brain2.png images as well as the resulting mean image of the healthy brain.

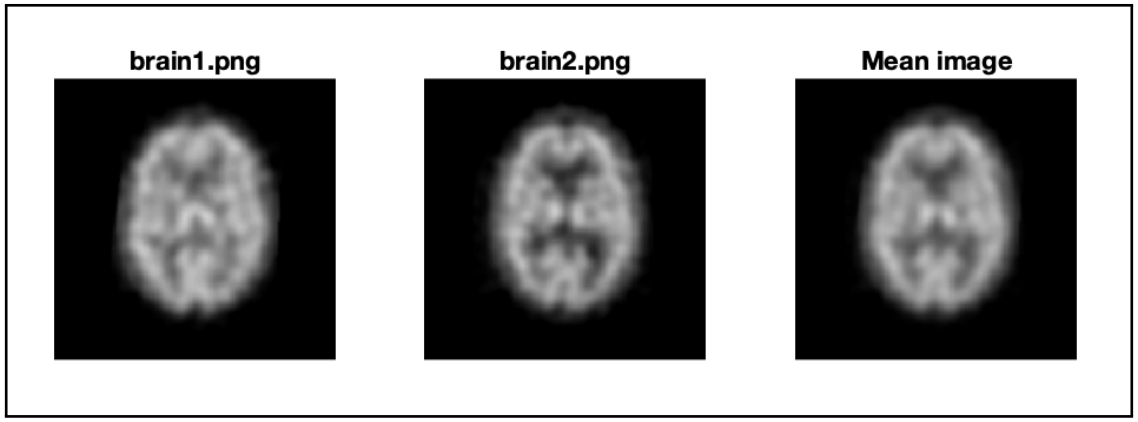

Figure 15: This figure shows the resulting Mean image from the weighted sum of brain1.png and brain2.png images.

11 Question 11

Find the difference between the "standard" brain and the image from the stroke patient (brain3.png). Where in the brain is the change located?

To find the difference between the image of the healthy brain and the image of the brain from the stroke patient, or vice-versa, the elementwise subtraction operation is performed. The result of the calculation $healthy\_brain - stroke\_brain$ is shown in Figure 16 and its analysis reveals that the stroke patient's brain has changed majorly in the top-right region. On the other hand, the outcome of the computation $stroke\_brain - healthy\_brain$ is shown in Figure 17 and, by analysising the image, it is also possible to observe that changes took place in the center region of the brain.

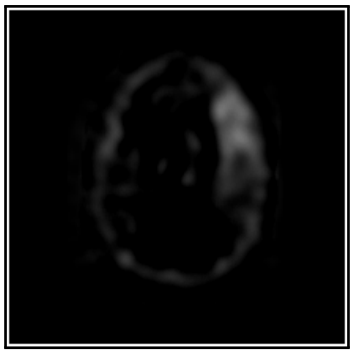

Figure 16: This figure shows the brain differences when subtracting the stroke patient's brain image from the healthy brain image.

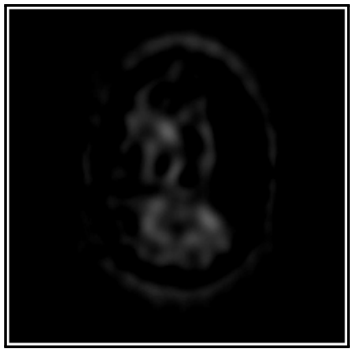

Figure 17: This figure shows the brain differences when subtracting the healthy brain image from the stroke patient's brain image.

12 Question 12

What happens when a pixel gets a value less than 0 or a value greater than 255? Are there other ways this can be handled? Did you think of this when you computed the "standard" brain in the previous exercise?

The decision on how pixel values that get values outside the 8-bit interval $[0, 255]$ are handled can vary depending on the function utilised. A common approach to handle this situation is to truncate values outside the specified range. However, other approaches can be taken, such as wrapping around the interval numbers that get outside it.

When computing the brain mean image, this matter was not noticed and addressed at first. However, after analysing the outcome image, both the overall big-picture as well as individually inspecting some of its pixel values, the matter was perceived and adequately taken into account. To address it, a suitable approach to compute the brain mean image was to first convert the pixel values from *uint8* to *single*. This procedure enabled the original information to be kept after the required operations. After dividing the result by 2 (i.e., performing the weighted sum), the pixel values were converted back to *uint8*. This manipulation helped against losing information throughout the processing.

13 Question 13

Compare rotations performed with and without interpolation. It is easiest to see differences along lines and edges of the images. What does interpolation mean in this case?

According to the assignment, using the command `J = imrotate(I,20);` rotates the image `I` in 20 degrees without interpolation. However, by reading the documentation provided by the command `help imrotate`, one learns that the default argument for the function, in case no interpolation method is specified, is the Nearest Neighbour (NN) interpolation method instead, as shown in Figure 18. Therefore, the comparison in this question regards the different interpolation methods, namely NN and Bilinear. The lines and edges in the image resulting from the Bilinear interpolation are smoother than when using the MATLAB-default NN method, as shown in Figure 19. The reason for such observation is that when applying the Bilinear interpolation method to a pixel, it takes into consideration the pixel's four diagonal neighbours to compute, by averaging, the corresponding resulting intensity value that composes the outcome image. On the other hand, the NN method only considers the pixel's closest neighbour when computing its resulting value.

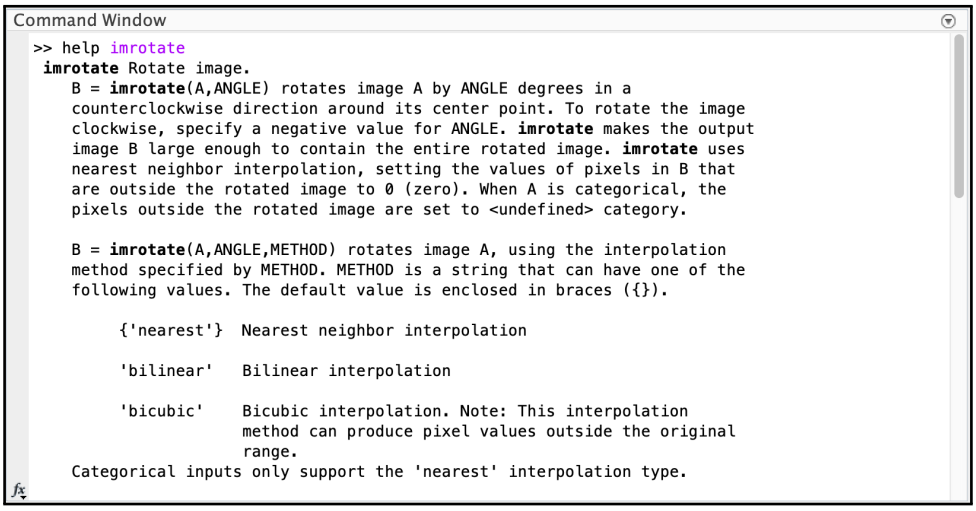

Figure 18: This figure shows the help documentation for the function *imrotate* provided by MATLAB.

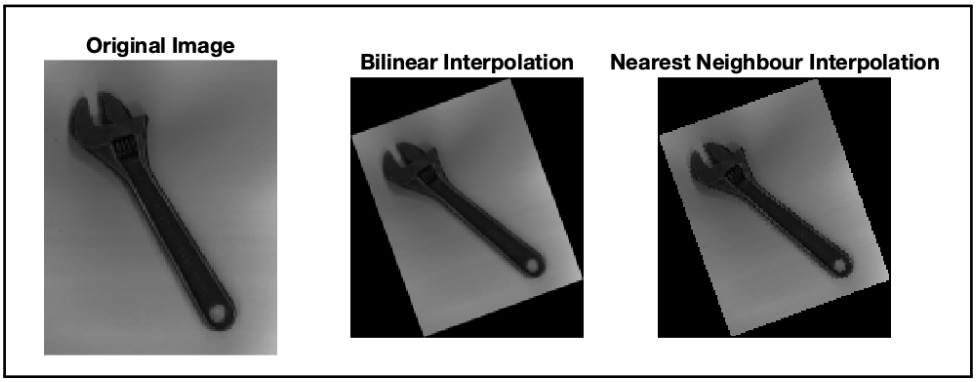

Figure 19: This figure shows the original, Bilinear interpolated and Nearest Neighbour interpolatd wrench.png images from left to right, as indicated by their titles.

14 Question 14

In general it is faster to rotate the image by a multiple of 90 degrees than by some arbitrary degree. Explain why. For this task use Matlab functions tic and toc.

When rotating an image by a multiple of 90 degrees, the grid axes after the rotation perfectly aligns with the grid axes before the rotation. This makes interpolation faster as less computation is needed (i.e., in worst case, it is only needed to swap pixel values with respect to the axes). On the other hand, when rotating an image by an arbitrary degree that is not a multiple of 90 degrees, the grid axes after the rotation does not align with the grid axes before the rotation. In this case, computing the interpolation (bilinear or NN) is required, which results in slower computational performance.

Figure 20 shows the time taken for rotating the *wrench.png* image by 90 degrees and 137 degrees when using the Bilinear interpolation method. The block of code illustrated was executed three times, sequentially, and the results obtained support the theory.

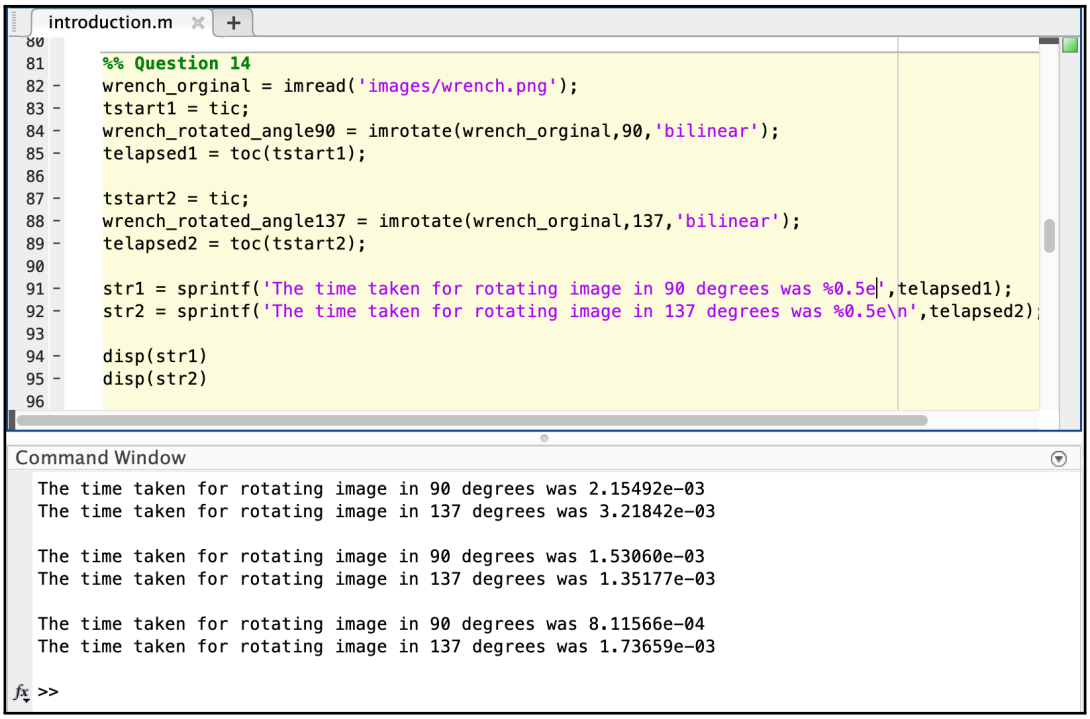

Figure 20: This figure shows the time taken for rotating an image by 90 degrees as well as by 137 degrees using the Bilinear interpolation method. The code was executed three times and the obtained results support the theory.

15 Question 15

- ***Load some image that you have downloaded from the internet.***
- ***Convert the image to grayscale using e.g.rgb2gray.***
- ***Resize and/or crop your image to*** 128×128 ***pixel size, without changing the aspect ratio of the image.***
- ***Loop (!) over the image using a*** 5×5 ***pixel window. For each such window, compute the average pixel value and store the result in a new image. Treat borders in some controlled way and make sure the result has the same size in pixels as the original image. You will need at least two nested for-loops, to scan rows and columns in your image.***
- ***Subtract the original image from your new filtered image.***
- ***Present the original, filtered and subtracted images in a figure, using e.g. subplot.***

The original, filtered and subtracted images are shown in Figure 21 from left to right, respectively. The outcome of using a 5×5 sliding filter kernel that averaged the grayscale values within it was a blur effect. Typically, the blurring effect depends on the dimensions of the kernel as well as its coefficient values. The filtered image shows a faded gray border which is equivalent to the outcome of padding the original image with zeros before the filtering process. The subtracted image highlights, as expected, the edges of regions (i.e., objects) that are surrounded by pixels with significantly different grayscale values (i.e., sharp transitions in greylevel values). The figure outcome was produced by using the *subplot* function in MATLAB, as shown in Figure 22.

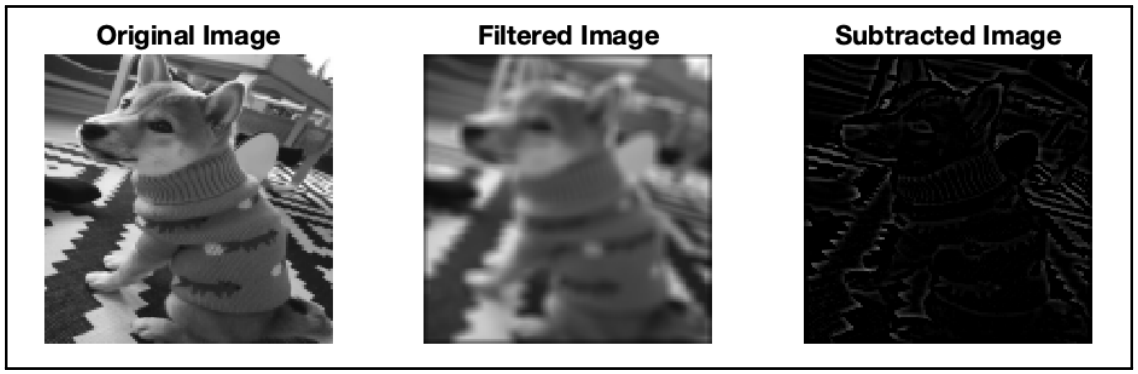

Figure 21: This figure shows the original, filtered and subtracted images from left to right, as indicated by their titles.

```
subplot(1,3,1),
imshow(resized_tomo);
title('Original Image');
subplot(1,3,2),
imshow(filtered_tomo);
title('Filtered Image');
subplot(1,3,3),
imshow(difference_tomo);
title('Subtracted Image');
```

Figure 22: This figure shows the code block responsible for outputing Figure 21. The function *subplot* was utilised to present all images in a single figure.

16 Question 16

- ***Load some image that you have downloaded from the internet.***
- ***Convert the image to grayscale using e.g.rgb2gray.***
- ***Perform histogram equalization on this image using your own algorithm, without using histeq. Efficiency is not so important, you may use for-loops or other ways to compute the transformation and apply it to the image.***
- ***Present the original and equalized images in a figure. Compare with Matlabs native histeq function.***
- ***Make your own histogram equalization into a callable function, e.g. Inew = myhist(I) and include the code in the report.***

The original, equalised and MATLAB-equalised (i.e., using the MATLAV native *histeq* function) images are presented in Figure 23. As it can be observed, the image produced by the designed *histogram_equalise_8bits* function, shown in Figure 24, is brighter than the image produced by the MATLAB-developed function. Nevertheless, the result is still visually acceptable and pleasant for the authors' perception.

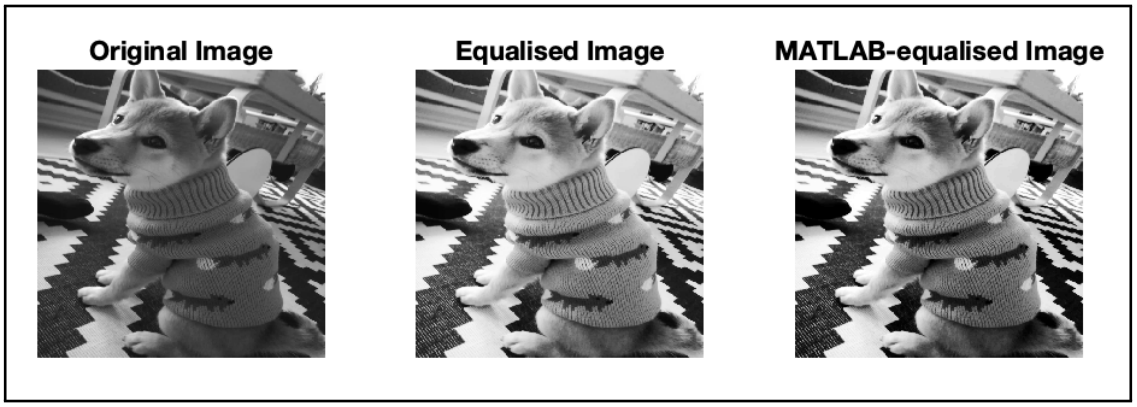

Figure 23: This figure shows the original, equalised and MATLAB-equalised images from left to right, as indicated by their titles.

histogram_equalise_8bits.m

```
function histogram_equalised_image = histogram_equalise_8bits(image_input)
[nrows,ncols] = size(image_input);
histogram_equalised_image = uint8(zeros(nrows,ncols));

num_bins = 256 - 1;
histogram = histcounts(image_input,num_bins);
normalised_df = histogram / (nrows*ncols);
cumulative_df = cumsum(normalised_df);

for x = 1:nrows
    for y = 1:ncols
        histogram_equalised_image(x,y) = num_bins*cumulative_df(1,image_input(x,y));
    end
end

end
```

Figure 24: This figure shows the function responsible for computing the Equalised Image in Figure 23. The function *histogram_equalise_8bits*, as indicated by its name, was designed for images with 8-bits intensity values.