Lab 3: Arithmetic and Logic Operations

Shido Nakajima

02/17/2023

02/17/2023

**OBJECTIVE:** The objective of this lab was to learn basic MATLAB arithmetic operations on digital images.

**METHODS:** In part 1, I created 4 custom MATLAB functions, each for addition of two images, subtraction of two images, element-wise multiplication of two images, and outputting the complement of a single image. In addition and subtraction functions, I used + and – arithmetic operators to add and subtract images, respectively. In multiplication function, I used .\* operator, which is a multiplication operator with period prior to it indicating that it is an element-wise operator rather than a matrix multiplication. In complement function, I decided to use bitcmp() function instead of not() or imcomplement(). I decided to move the whole part 1 section to the bottom of the script, as function definitions must always be at the bottom of the script.

In part 2, I resized the image using imresize with size multiplied by 3 and methods being “nearest”, “bilinear”, and “bicubic”, respectively. For constructing the binary mask for displaying only the central 1/9 of the image, I started from zeros() of the resized image, creating an all-zero binary image with the size 3 times the original image. Then, I took the length and width of the original image, which is 1/3 in width and length of the initiated binary mask. Then, using a for-loop integrated in a for-loop using the dimension of the original image, I changed the central 1/9 of the mask to 1. Finally, to apply the mask on the image, I used element-wise multiplication with bilinear resized image, converting the mask into 8-bit using uint8() function.

In part 3, I loaded the MRI image, which was then compressed into a 3-dimensional image using squeeze() function, which eliminates all non-relevant dimensions. Then, I decided to use slice 15 of the MRI, which I defined using the coordinates (:, :, 15). This image was resized in a similar manner as from part 2, with scaling factor being 4 and method being bicubic. Then, 5 unique gaussian noises were introduced to this resized image using imnoise() function. The average of these 5 noisy images were taken by simple mathematical averaging: y1/5 + y2/5 + y3/5 + y4/5 + y5/5.

**RESULTS:** For part 1, the addition function led to a useful image, where the data from both images were well-combined. The subtraction function led to a similar image, but with the brightness being extremely low and thus not very useful. The multiplication function led to very whitened image, due to the maximum value of 8-bit image being 255, and multiplication of the image element values easily surpassed 255. The complement function simply outputs the complement of the MRI image, where the dark part became bright and bright part became dark. Initially, not() and imcomplement() functions were tried before bitcmp() function, but both led to an error message regarding the data type of the variable that it accepts. Although I could not figure out how to use these functions without getting an error message, I believe that this error is happening because the custom function that I made is not specifying any data type for the input, which means that the variable that the two functions can be inputted with may not be an image.

For part 2, I decided to output a single figure with four images: nearest neighbor resized image, bilinear resized image, bicubic resized image, and the bicubic resized image with the created mask applied to it.

For part 3, 4 figures were output: the original image of slice 15, the 4x resize of the image, a figure with the 5 different noise-introduced images, and the averaged image. Since the gaussian noise was unique for each of the 5 different images, the averaged image of the 5 noisy images seemed to be smoother than the 4x resized image without any noise introduced. I believe that this is because the gaussian noise introduced unique differences in the edges of each of the images and taking the average of the 5 identical images with different noise caused some of the pixelated edges to be smudged, causing it to seem smoother.

**CONCLUSION:** This lab exercise introduced me to some basic arithmetic operations for images, which are the basis of image manipulation using MATLAB. Additionally, the use of enlargement of the images allowed me to further understand the importance of image details (number of pixels).

**APPENDIX:**