Homework 1 Questions

Instructions

- Compile and read through the included MATLAB tutorial.
- 2 questions.
- Include code.
- Feel free to include images or equations.
- Please make this document anonymous.
- **Please use only the space provided and keep the page breaks.** Please do not make new pages, nor remove pages. The document is a template to help grading.
- If you really need extra space, please use new pages at the end of the document and refer us to it in your answers.

Submission

- Please zip your folder with **hw1_student id_name.zip** (ex: hw1_20191234_Peter.zip)
- Submit your homework to KLMS.
- An assignment after its original due date will be degraded from the marked credit per day: e.g., A will be downgraded to B for one-day delayed submission.

Questions

Q1: We wish to set all pixels that have a brightness of 10 or less to 0, to remove sensor noise. However, our code is slow when run on a database with 1000 grayscale images.

Image: grizzlypeakg.png

```
A = imread('grizzlypeakg.png');
[m1,n1] = size( A );
for i=1:m1
    for j=1:n1
        if A(i,j) <= 10
            A(i,j) = 0;
        end
    end
end
```

Q1.1: How could we speed it up?

A1.1: We could speed up the code by replacing the for loops with logical indexing. The code could be edited as the following.

```
image = imread('grizzlypeakg.png');
noise_index=image <= 10;
image(noise_index) = 0;
```

Q1.2: What factor speedup would we receive over 1000 images? Please measure it.

Ignore file loading; assume all images are equal resolution; don't assume that the time taken for one image $\times 1000$ will equal 1000 image computations, as single short tasks on multitasking computers often take variable time.

A1.2: Ignoring image loading time, the speedup method takes around 3.88 times less time. Original method takes around 16.7s, while the speedup method takes around 4.3 s. The following code measures the improvement of the speedup method.

```
image = imread('grizzlypeakg.png');
tic;
for x=1:1000
    noise_index=image<=10;
    image(noise_index)=0;
end
t1 = toc;
image = imread('grizzlypeakg.png');
tic;
for x=1:1000
    [m1,n1] = size( image );
    for i=1:m1
        for j=1:n1
            if image(i,j) <= 10
                image(i,j) = 0;
            end
        end
    end
end
t2 = toc;
fprintf('Speedup method: %g s\n', t1)
fprintf('Original method: %g s\n', t2)
fprintf('Improvement: %g times\n', t2/t1)
```

Q1.3: How might a speeded-up version change for color images? Please measure it.

Image: grizzlypeak.jpg

A1.3: For color images, the speedup versinon should include the conversion from color to grayscale image. The edited version is shown below. The speed improvement is around 3.82 times. The speedup method takes around 4.7 s, while the original method takes around 17.9 s.

```
colorImage = imread('grizzlypeak.jpg');
tic;
for x=1:1000
    image=rgb2gray(colorImage);
    noise_index=image<=10;
    image(noise_index)=0;
end
t1 = toc;
tic;
for x=1:1000
    image=rgb2gray(colorImage);
    [m1,n1] = size(image);
    for i=1:m1
        for j=1:n1
            if image(i,j) <= 10
                image(i,j) = 0;
            end
        end
    end
end
t2 = toc;
fprintf('Speedup method: %g s\n', t1)
fprintf('Original method: %g s\n', t2)
fprintf('Improvement: %g times\n', t2/t1)
```

Q2: We wish to reduce the brightness of an image but, when trying to visualize the result, all we sees is white with some weird “corruption” of color patches.

Image: gigi.jpg

```
I = double( imread('gigi.jpg') );
I = I - 20;
imshow( I );
```

Q2.1: What is incorrect with this approach? How can it be fixed while maintaining the same amount of brightness reduction?

A2.1: The image is read as an integer matrix but converted to a double matrix. Therefore the argument I of imshow function is of double matrix type. However, imshow expects that an argument double matrix has elements in the range 0 to 1. Because in the current code matrix I does not contain elements in the range 0 to 1, imshow shows a strange image. There are two solutions for this issue. First, the double conversion can be removed as the following.

```
I = imread('gigi.jpg');
I = I - 20;
imshow(I)
```

Otherwise, the amount of brightness reduction can be adjusted. The original amount of 20 is out of 255, so it should be converted to a value out of 1. Also, the double matrix should be divided by 255 to make the elements exist in between 0 and 1.

```
I = double(imread('gigi.jpg'))/255;
I = I - 20/255;
imshow(I)
```

Q2.2: Where did the original corruption come from? Which specific values in the original image did it represent?

A2.2: In the original code, values in the range 0-255 are put into a function that expects values in the range 0-1. Therefore, only the values in the range 0-1 are shown. Consequently, only the regions in the original image that have values 0-1 (almost black) are represented.