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_20181234_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: Your answer here.

```
A = imread('grizzlypeakg.png');
[m1,n1] = size( A );
for k = 1:1000
        B = A<=10;
        A(B) = 0;
end
```

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: Your answer here.

The result change every time, so I conduct the same experiment for 3 times and claculate the average value of them. It took 19.603sec for provided code and 6.369sec for new method. It got 3.08 times faster. The time for generating random image is excluded.

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

Image: grizzlypeak.jpg

A1.3: Your answer here. I used 100 random noise and the time for generating random image is excluded. It took 155.5590 sec for extended version of provided way.

```
for iter = 1:100
        for i=1:l1
                for j=1:m1
                        for k=n1
                                if A(i,j,k) <= 10
                                        A(i,j,k) = 0;
                                end
                        end
                end
        end
end
```

On the other hand, following method took 7.2290 sec.

```
for iter = 1:100
        B = A<=10;
        A(B) = 0;
end
```

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: Your answer here.

First, imshow took the normalized value for the image data, so it need to use im2double. Which means that in order to handle the image with floating point, it requries normalization. Thus, you need to normalize the subtracting value when you decrease the brightness. Following is the right version of code.

```
I = im2double( imread('gigi.jpg') );
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: Your answer here.

The imshow() function in Matlab convert the negative value to 0, and conver the value grater than 1 to 1. In other word, if the original (normalized) value of one cell was (2, 1.5, -1) in RGB format, imshow represent this cell as (1,1,0) which is bright yellow.

When I decrease the brightness by subtracting 20 from the I = double(imread('gigi.jpg')), there were 39282 element in I which has samller value than I. Every cell where contains smaller value than 1 represented in different color from white. Therefore, these cells are the corruption.