Image Processing I

EECE 4353/5353 Fall 2018

Homework Assignment 2: Point Processing of Images: Brightness

Updated Friday 14 September 2018

Due Sunday 16 September 2018

The goal of this homework is to learn how manipulating the brightness of an image changes it in ways that may not be expected.

In this homework (and all the others) you will be writing your own image processing functions. Unless explicitly stated otherwise, you may not use the functions from the MATLAB image processing toolkit by themselves or within your functions to solve the problems.

Help can be gotten on any MATLAB command (or function) by typing the command's name in the text box in the upper right corner of the main MATLAB window. It will often give you many choices in the widow that drops down. Usually you will want to select the first choice that has "MATLAB" beside it on the right. Also you can type `help command_name`, in the command window. *E.g.* to get help on `imwrite`, type `help imwrite` at the command prompt. This will print the help information in the command window itself. Another way is to press the <F1> key, select the "Index" tab in the window, and type the command name into the search box.

Always open a new figure window before displaying an image, otherwise the image will appear in the previous window with the attributes of that window such as its size and color map (if there is one). Close the figure windows that are not relevant to the current part of the assignment so not to overwhelm your computer's resources.

Get in the habit of terminating every line of MATLAB code with a semicolon `;` If you do not, the program writes the result of the operation to the command window. If an image happens to be the result, all $R \times C \times B$ numbers will be dumped to the screen. (That doesn't hurt anything but it's annoying.)

In completing this assignment (and the others) if specific images are not supplied, you may use any images you like providing that they are of the type specified by the problem description (*e.g.* 24-bit truecolor). Please do not use images that are obscene or gruesome. In general, an image is OK if you could show it to your grandmother without upsetting her or embarrassing yourself. A wide variety of images are available under creative commons license on the web site `www.flickr.com`.

Provide a photo credit for every image you use. That is, *the first time* you use an image in your report it must have a reference. That reference should include the name of the photographer, if it is available. Credit yourself if you took the picture. Include the source of the image, *e.g.* a URL. Use "personal collection" for your own images. If you know neither the photographer nor the source you should state that. There is no need to credit the same photo more than once.

In your report, include the images you use, the numerical results of any experiments you run, and copies of any transformed images. Always include a description of what you did, and an explanation of results.

I suggest that you use MATLAB 's Code Sections feature when coding your work. Start a new section for each numbered problem – start each one with %%. Include comments (lines that begin with %) in the section that describe what you are doing what the results mean. Then use MATLAB 's publish function to run the entire assignment. In the editor, select the "Publish" tab. Then select the "Publish" dropdown menu, "Edit publishing options..." In the lower-right box in that window that opens find "Output settings." Click on the word "html". Drop down to "pdf" and select that. Then click on the "Publish" button on the bottom.

It will create a pdf file with everything in it, in the order in which you programmed it.

Homework 2: Point Processing of Images: Brightness

The "brightness" of an image is a subjective term. The National Telecommunications and Information Administration (NTIA)[1] publishes standard definitions for the names of attributes of signals and images. The NTIA's US Federal Glossary of Telecommunication Terms (FS-1037C), states that "'brightness' should now be used only for non-quantitative references to physiological sensations and perceptions of light."[2] So, the term conveys no precise technical meaning. For this course, we will define the brightness of an image to be its average intensity on a scale of $\{0, \ldots, 255\}$ or $[0, 1]$ depending on the class of the image.

1. Find a 24-bit truecolor image of your choice to use in these problems. Display it in your report with a photo credit. The image should exhibit a full range of values and colors. If both its linear dimensions are greater than 512, resize it so the the larger of the two dimensions is 512. Read the MATLAB documentation on `imresize()`. Then use it with the "bicubic" option to resize your image. If both the image's dimensions are less than or equal to 512 you may keep it as is. Determine the image's class and dimensions (after resizing) and include them in your report. Also compute the mean intensity value of the image using the MATLAB code, `mu = mean(I(:));` where, of course, `I` is the image array.

2. Write a MATLAB function to alter the brightness of an image of class `uint8`. The function should work on either grayscale or truecolor images. It does not need to work on colormapped images.

   The average intensity of an image can be used as an indicator of its perceived brightness. Many off-the-shelf image processing programs have a brightness control based on a percent change in image brightness. Therefore the function you write should input an image array, say `I`, of class `uint8` and a number, `p`. If `p` is less than 100 the result should be a darker image; if it is greater than 100 the result should be brighter than the original.

   The output should be an image, say `J`, of class `uint8` whose mean intensity is (in theory) `p` percent of the mean value of `I`.

   The first thing your function should do is convert the image to double.

   Then compute the mean of the image, `mu = mean(I(:));`.

   Then compute the brightness shift,

   ```
   g = (p/100 - 1)*mu;
   ```

   Add `g` to every pixel in the image and convert the resulting image to `uint8`. (That can be done without loops in one line of code.)

   Return the brightness-shifted, class `uint8` image.

   This function should not be long, so include the code in your report here, rather than in an appendix. If you are using MATLAB 's code sections and publish function just cut and paste the function's code into the section and comment every line of it.

3. Increase and decrease the brightness of your image by 10% and by 50%.[3] Then you will have 4 new images: one 10% dimmer image, one 10% brighter image, one 50% dimmer image, and one 50% brighter image.

   Display these in your report, each with a caption indicating which one it is. If you were to look at the 10% brightness-shifted images and the original separately in random order, viewing other images between them, do you think you could accurately tell which was which? What about the 50% shifted images?

[1] `http://www.its.bldrdoc.gov`.

[2] `http://www.its.bldrdoc.gov/fs-1037/dir-005/_0719.htm`.

[3] The approach we are taking implies that if we want to decrease the average intensity by 10%, then `p=90`. To increase it by 10%, `p=110`.

4. From the uint8 image, `I`, compute and display the numbers,

   ```
   >> muI = mean(I(:)),
   >> muI * 0.5,
   >> muI * 0.9,
   >> muI * 1.1, and
   >> muI * 1.5.
   ```

   Compute and display the mean intensities of the 4 brightness-shifted `uint8` images. (Be sure to label the numbers in your report.)

   Are the means of the brightness shifted images equal to the corresponding percentages of the mean of the original image?

   Depending on the image it is likely they are not the same. The following problems will explore the reasons for that.

5. Write a new version of your brightness shift program that does not convert the results to uint8. After shifting the brightness of the class double image, return it with out converting it to uint8.

6. Compute and display the mean intensities of the 4 brightness-shifted class `double` images. (Be sure to label the numbers in your report.)

   Are the means of the brightness shifted images equal to the corresponding percentages of the mean of the original image?

7. The difference between the means that should, theoretically, be the same but are not is due to two phenomena: "round-off" errors and "clipping." When a class `double` image is converted to class `uint8`, any intensity values that are not integers are converted to integers. Try this:

   ```
   >> uint8([128 128.4 128.5 128.6])
   ```

   Based on what you see how are `double` values being converted to `uint8`?

   Now try this:

   ```
   >> uint8([-128 0 255 383])
   ```

   Why, do you suppose, is this phenomenon called clipping?

   If there were no clipping, the mean intensity would shift along with all the intensities.

   Let `I` be the original image and let `R` = number of rows, `C` = number of columns, and `B` = number of bands in I. Then `N = R * C * B`, is the total number of scalar intensity values in `I`. The mean of `I` is

   $$a_0 = \frac{1}{N}\sum_{b=1}^{B}\sum_{r=1}^{R}\sum_{c=1}^{C} I(r,c,b),$$

   If all the intensities were shifted by `d` then

   $$\begin{aligned} a_1 &= \frac{1}{N}\sum_{b=1}^{B}\sum_{r=1}^{R}\sum_{c=1}^{C} (I(r,c,b)+d) \\ &= \frac{1}{N}\sum_{b=1}^{B}\sum_{r=1}^{R}\sum_{c=1}^{C} I(r,c,b) + \frac{1}{N}Nd \\ &= a_0 + d. \end{aligned}$$

   However, if clipping occurs this no longer holds. Try this:

```
>> mean(0:256)
>> mean(-128:128)
>> mean(uint8(-128:128))
>> mean(128:384)
>> mean(uint8(128:384))
```

Given what you have observed, how does the mean change when the image is clipped at 0? How does it change when it is clipped at 255?

8. Operate on the 50% shifted images to "restore" them to their original brightness levels. Assuming no clipping, what percentage `p` should you use to restore

   (a) the 50% dimmer image, and

   (b) the 50% brighter image.

   Use these numbers to generate two new `uint8` images. Compare the mean values of these restored `uint8` images to that of the original. Display the images, include titles so you know which is which, and comment on any differences you can see between them and the originals. You do not need to include the restored images in your report. However, if you use the MATLAB publish feature to generate your report they will be included by default. And, of course, that's OK.

9. If clipping occurs due to processing of any kind, that means that information has been lost. Visually significant features of the image may be obliterated. That's why I placed the word, restore, in quotes above. One may be able to re-shift the mean to the original number but the features that are lost will not return. Here, we will determine the extent of the damage by comparing the restored images to the originals.

   Compute two images that are the squared differences between the original image and the restored 50% brightness increased and decreased images (the as follows:

   ```
   >>  D = (double(I) - double(J)).^2;
   ```

   where `I` is the original `uint8` image and `J` is one of the `uint8` restored images. Find and display the minimum and maximum values of each difference image.

   The best way to display the difference images is by using

   ```
   >> figure
   >> imagesc(D)
   >> truesize
   ```

   Comment on the results. What does each one show? How do the 10% difference images compare to the 50% difference images. Include all four of the difference images in your report.

10. **Additional problem for graduate credit.**

    Color Modification through Brightness Shifting

    If the brightness of only one color band is changed, the overall color of the image is changed.

    Separate the truecolor image, `I`, into its three bands. That is, create three monochrome images, `IR` that is the red intensity, `IG` that is the green intensity, and `IB` that is the blue intensity from your truecolor image.

    Display each of them as grayscale images. You may need to use the MATLAB function `gray(256)` to generate the correct colormap. That is, to display one of the images do

```
>>  figure;image(IR);truesize;colormap(gray(256);
```

Use your brightness shifting function to:

(a) Increase the brightness of the red band by 20%. Then recombine it with the unchanged, original green and blue bands to create a new truecolor image. Display it and comment on its change from the original.

(b) Decrease the brightness of the red band by 20%. Then recombine it with the unchanged, original green and blue bands to create a new truecolor image. Display it and comment on its change from the original.

(c) Increase the brightness of the green band by 20%. Then recombine it with the unchanged, original red and blue bands to create a new truecolor image. Display it and comment on its change from the original.

(d) Decrease the brightness of the green band by 20%. Then recombine it with the unchanged, original red and blue bands to create a new truecolor image. Display it and comment on its change from the original.

(e) Increase the brightness of the blue band by 20%. Then recombine it with the unchanged, original red and green bands to create a new truecolor image. Display it and comment on its change from the original.

(f) Decrease the brightness of the blue band by 20%. Then recombine it with the unchanged, original red and green bands to create a new truecolor image. Display it and comment on its change from the original.

In general, how might you use this approach to tint the image with any given color?