Assigment 3 – Image Enhancement

Department of Electrical and Computer Engineering
Ryerson University

ELE 882 – Fall 2017

1 Introduction

Image enhancement is a set of techniques that improves the *subjective* quality of an image. The reason for this is that it's more or less impossible (aside from a few very specific cases[1]) to define an objective measure for what is an "ideal" image. Finding the set of operations that can best improve an image's subjective quality is very difficult and depends very much what is "wrong" with that particular image.

For this assignment you will be looking at three types of enhancement operations: contrast, sharpness and noise-reduction. You have already seen some examples of contrast enhancement in Assignment 1. However, those were somewhat crude in that they were global operators that did not consider the content of the image. This assignment will focus more on automated methods based on processing the image's global and local histograms.

Sharpness enhancement, sometimes called "deblurring", is any operation that improves the *apparent* sharpness of an image. The result of a sharpness enhancement will make it easier to discriminate between objects in an image by making their edges appear stronger. Unlike contrast enhancement, sharpness enhancement is implemented by spatial operators and actually changes the "structure" of the image. An important point about sharpness enhancement is that it has nothing to do with an image's contrast. A low-contrast image can still have its sharpness improved; it will just be very difficult to see that improvement.

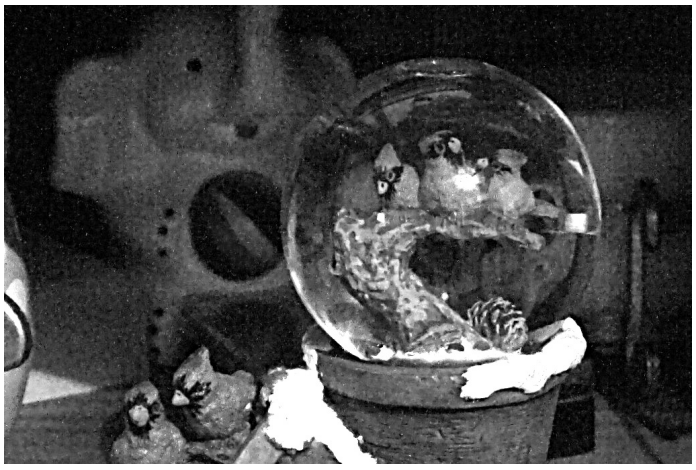
Related to sharpness enhancement is noise reduction. In fact, noise reduction is the exact *opposite* operation to sharpness enhancement. That's because noise reduction is often implemented with low-pass filters because noise is most visible as "higher frequency" content. Generally, most of an image's frequency content is in the lower, near-DC frequencies. Visually these regions are large, flat areas. Edges, i.e. high-frequency content, only makes up a small portion of the entire image's area. This makes noise, which is usually uniformly spread throughout an image, very noticeable in large flat areas. A linear or non-linear (such as a median filter) LPF is the best way to address this problem.

In practice, a combination of these approaches is often used to enhance an image and the order in which they are applied can make a difference. For instance, performing a contrast enhancement followed by a sharpness enhancement may actually degrade an image's quality! If noisy enough, the initial contrast enhancement will amplify the noise and the sharpness enhancement will then "sharpen" the noise, making it even more apparent. Figure 1 shows an example of this.

[1] In some rare cases, there is a **ground truth** that is the desired image without any noise or other factors that can reduce its quality. Usually this is only available when a noise model is available for the particular imaging sensor or a noise-free image is available for comparison.

(a) Original "Snowglobe" Image

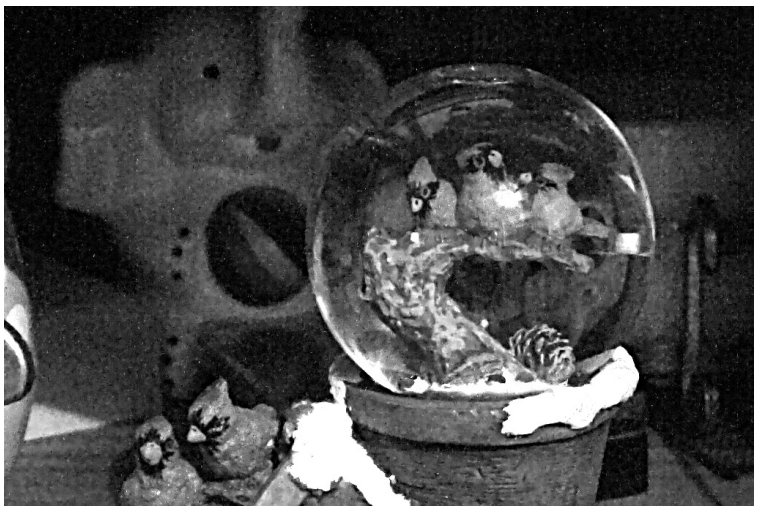
(b) After Contrast Stretching



(c) After Unsharp Masking

Figure 1: An example of a "bad" enhancement. First the image is contrast stretched in (b). This improves the overall contrast but also makes the noise hiding in the dark region more visible. Finally, an unsharp masking is applied to make the edges more distinct in (c). But because the noise makes spurious edges, the sharpening operation just makes the noise even more visible.

A better, option, as shown in Figure 3 is to first pre-filter the "snow globe" image, in this case with a Gaussian filter with a $\sigma = 1.5$, and *then* do the contrast and sharpness enhancement. This way the noise is mitigated before any operations that could make the noise more visible. But again, there is a caveat: noise reduction, by necessity, removes small-scale detail from the image. A comparison between the final results is shown in Figure 2. While the filtered result looks "cleaner", the edges are slightly less distinct even after the sharpening enhancement due to the initial filtering.



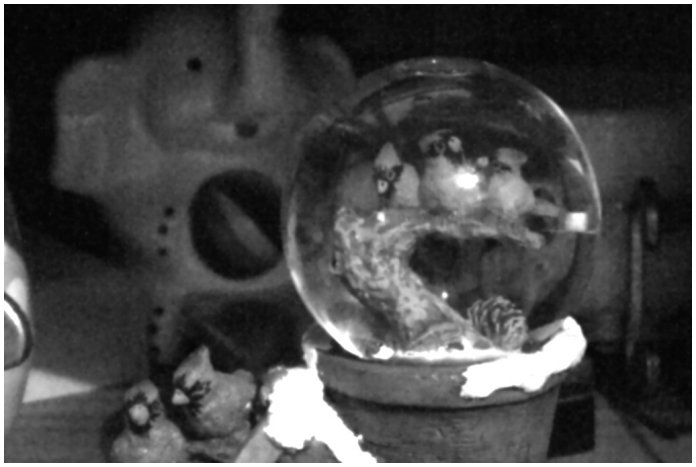
(a) No Pre-filtering

(b) With Pre-filtering

Figure 2: A comparison between enhancement with, and without, pre-filtering.

In some cases you can actually be quite aggressive (a large blurring filter radius, for example) such as when you are producing a lower resolution image. Because the image is being shrunk, a fair amount of detail is already being lost so more filtering doesn't necessarily make the image "worse". This same logic cannot be applied in other cases because the small-scale details may be important, as is common in medical imaging. Some very complex techniques have been developed to maintain as much detail as possible but trade-offs will always have to be made.

(a) Filtered "Snow Globe" Image



(b) After Contrast Stretching

(c) After Unsharp Masking

Figure 3: The enhancement pipeline from Figure 1 but with the image first being filtered to remove some of the noise.

2 Assignment

In this assignment you will be implementing contrast and sharpness enhancement operators and applying them to different images in order to enhance them. You will be expected to use the spatial filtering function that you developed in Assignment 2 as that will be used for the sharpness enhancement. You are allowed to use Matlab's `medfilt2` function to perform median filtering.

Once you have finished implementing the enhancement operators you are expected to apply them to the provided test images. The test images are provided for you on Blackboard in two sets. The first set is for testing the individual operators. The second set is for investigating how a combination of operators can be used to enhance an image. Once this is complete you are expected to answer all of the analysis questions (Section 3) in your final report. You must also include *all* of the source code that you used to generate the results in your report.

2.1 Enhancement Operators

All of the operators below must be implemented as a MEX or M-script function according to the specified prototype, i.e. that is how the functions are to be called from the command window. You have some flexibility in *how* to implement the operators. Because of the work you have done in the past two assignments, you can actually reuse some of your existing functions (such as the spatial filters from Assignment 2).

2.1.1 Problems [14 Marks]

1. **[3 marks]** Implement the global histogram equalization operator as a MEX function with the prototype

   ```
   out = histogram_equalize(img)
   ```

   where `img` is the original image and `out` is the image after the histogram equalization. Ensure that `histogram_equalize()` will only accept images of type `unsigned char`.

2. **[5 marks]** Implement the *local* histogram equalization operator as a MEX function with the prototype

   ```
   out = adaptive_histogram(img, H, W)
   ```

   where `H` and `W` is the height and width of the local processing window, respectively. Again, ensure that `adaptive_histogram()` will only accept images of type `unsigned char`.

3. **[3 marks]** Implement an unsharp mask operator, either as a MEX function or a normal Matlab function with the prototype

   ```
   out = unsharp_mask(img, r, k)
   ```

   where `r` is the radius of the blurring kernel used to create the "unsharpening" mask and `k` is the strength of the mask. Recall that the unsharp masking operation is defined as

   $$I'(x,y) = I(x,y) + k(I(x,y) - h_{blur} * I(x,y)), \tag{1}$$

   where h_{blur} is a blurring kernel (usually Gaussian) of radius r and '$*$' is the convolution operation.

4. **[3 marks]** Implement a Laplacian sharpening operator, either as a MEX function or a normal Matlab function with the prototype

   ```
   out = laplacian_sharpen(img, k)
   ```

   where `k` is the strength of the sharpening. Recall that a Laplacian sharpening is defined as

   $$I'(x, y) = I(x, y) - k\nabla^2 I(x, y). \tag{2}$$

   IMPORTANT: Depending on how you define the Laplacian operator, you may actually need to *add* the scaled Laplacian in (2), not subtract it. See Chapter 3.6.2 in the text book.

5. **[5 marks (Bonus)]** Modify the `adaptive_histogram()` function from Question 2.1.1-2 so that it accepts a third argument, `cutoff`. If this value is not provided then the function should act *identically* to the specification given in 2.1.1-2.

   The `cutoff` argument is the value where if a histogram bin, $h[n]$, is greater than `cutoff`, the excess will be "redistributed" throughout the histogram. The goal is to avoid creating false features in regions of nearly uniform intensity. This strategy is known as "Contrast Limited Adaptive Histogram Equalization" or CLAHE.

   There are a number of ways to implement this. One way would be to add the excess to the value of $h[n]$ with the smallest value, check if any other value of $h[n]$ exceeds cutoff and repeat the process until either *all* values of $h[n]$ are less than `cutoff` or to terminate after a set number of iterations. Another way, which avoids getting potentially stuck in a never-ending loop, is to just take the excess and add it equally to all of the histogram bins. This means some bins will be higher than the cutoff but it avoids any loops.

   Regardless of the way that you approach the problem, you should ensure that `cutoff` is always larger than $(N \times M)/256$. This corresponds to the case where there is the same number of pixels of each colour value in the processing window.

2.2 Testing [8 Marks]

Before continuing onto general image enhancement, first test out the implementation of your enhancement operators on the images provided in the `testing` folder.

2.2.1 Problems

1. **[2 marks]** For each test image in the `testing/contrast` folder, apply the `histogram_equalize()` function and put the results in your report.
2. **[2 marks]** Repeat 2.2.1-1 using the `adaptive_histogram()` function instead.
3. **[2 marks]** For each test image in the `testing/sharpen` folder, apply the `unsharp_mask()` function and put the results into your report. You must also report the values used for the radius and strength parameters.
4. **[2 marks]** Repeat 2.2.1-3 using the `laplacian_sharpen()` function instead.
5. **[2 marks (Bonus)]** Repeat 2.2.1-2 using the modified (i.e. CLAHE) version of the `adaptive_histogram()` function.

2.3 Enhancement

Four images have been provided in the `enhance` folder:

noise_additive.png
: An image (with poor contrast) that has also been corrupted by additive, zero-mean Gaussian noise.

noise_multiplicative.png
: An image (with poor contrast) that has also been corrupted by multiplicative, zero-mean noise with a uniform distribution.

noise_impulsive.png
: An image (with poor contrast) that has also been corrupted by impulsive, better known as "salt and pepper" noise.

snowglobe.png
: One of the snow globe images from Assignment 1. The image was shot under low light, short exposure and high ISO and is somewhat noisy.

The "noise" images have all been generated using the `imnoise()` function in Matlab. Very briefly, if $x(t)$ is the signal and $n(t)$ is some zero-mean noise process with variance σ then additive noise is

$$x_n(t) = x(t) + n(t). \tag{3}$$

Similarly, multiplicative noise is

$$x_n(t) = x(t) + n(t)x(t). \tag{4}$$

Finally, impulsive noise is simply the case where a pixel is randomly set to its largest or smallest value. The Matlab documentation has more information on how this function works and how the noise is generated.

2.3.1 Problems [8 Marks]

1. **[8 marks]** *For each image*, write a Matlab script that, when run, will enhance the image with the operations that produce the best results visually. As stated in the introduction the order and type of operations vary from image to image and there is no "right" answer. Instead you must justify, in your report *why* you chose that particular set of operations and parameters. It is quite common to tweak the operations and parameters multiple times before a visually appealing result is obtained and doing the enhancement inside a script makes this process easier.

3 Analysis

You must **completely** answer each question below in order to receive full marks. One word or sentence answers are insufficient and you must justify all your answers.

3.1 Questions [10 Marks]

1. **[4 marks]** What type of low-pass filter was best suited for handling additive noise? What type of filter was best suited for multiplicative noise? Salt and pepper? Explain why based on your results.

2. **[2 marks]** Consider the how the different types of noise are defined. Theoretically, what type of filters would be best suited for removing the different noise types. Compare this to the results that you obtained.

3. **[2 marks]** Compare and contrast the global histogram equalization to the local histogram equalization. Are there any circumstances where the global method is better? The local method?

4. **[2 marks]** You implemented two types of sharpening operators: unsharp masking and Laplacian sharpening. Compare and contrast the two method. Is one better? Why or why not? Ignore the fact that Laplacian sharpening is computationally simpler.

5. **[2 marks (Bonus)]** Based on your results, what is the benefit to the clamping done by the CLAHE method? Earlier in this document it was stated that this helps to avoid boosting noise in regions of uniform value. Why, or why not, would this be the case?