EE569: HOMEWORK #1

EE569 - Intro to Image Processing
Session: Mon, Wed, 8AM
Date: January 22, 2024

Kaisen Ye
kaisenye@usc.edu
(346) 368-1097

Problem 1: Image Demosaicing and Histogram Manipulation

1.1 Abstract and Motivation

In the world of capturing images, making sure they are clear and detailed is about more than just technicality; it's about making them as true to what we see with our own eyes as possible. When looking through a security camera in a city or from a satellite above, being able to pick out the details in dark corners or bright spots could mean catching something crucial that might otherwise be missed. The usual ways of making an image's contrast better often don't cut it, leaving us with pictures that either look too flat or are noisy.

This is where Contrast Limited Adaptive Histogram Equalization (CLAHE) comes into play, offering a smarter approach to detail and contrast. It breaks them down into smaller pieces, improves each section separately, and makes sure that noise doesn't ruin the overall quality. We're diving deep into how CLAHE works, putting it up against common methods like the transfer function and bucket filling. We want to find out if CLAHE is the answer to not just improving how images look but making them feel more like what we see in the real world.

1.2 Approach and Procedures

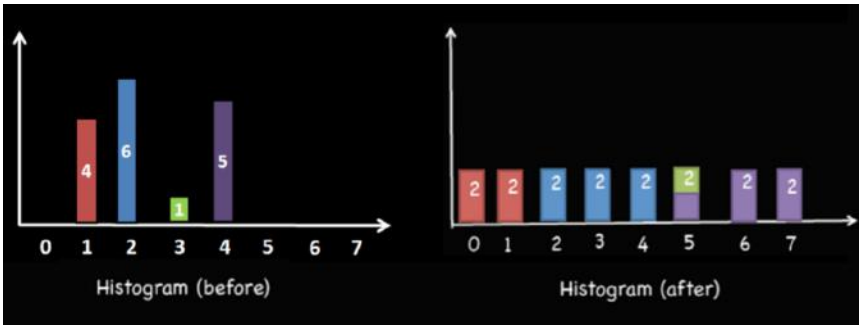
For the first part of the problem, the key step is to interpolate the missing components of each color channel by averaging the color values of adjacent pixels. For each missing color in a pixel, calculate its value by averaging the values of the adjacent pixels that do contain that color. This process is repeated for the red, green, and blue channels, respectively.

$$\hat{B}_{3,4} = \frac{1}{4}(B_{2,3} + B_{2,5} + B_{4,3} + B_{4,5})$$

$$\hat{G}_{3,4} = \frac{1}{4}(G_{3,3} + G_{2,4} + G_{3,5} + G_{4,4})$$

The second part involves adjusting the image's histogram to improve contrast. Two techniques will be applied: transfer function-based and bucket filling histogram equalization. The transfer function-based technique starts by computing the cumulative distribution function (CDF) from the histogram, then proceeds to create a mapping function that relates original pixel values to

new values based on the CDF. Whereas the bucket filling method distributes the pixel intensities across the histogram, aiming for an equal number of pixels in each intensity level (bucket).



Lastly, we employ the CLAHE technique to intensify the contrast in digital images. The procedural workflow initiates with the conversion of the standard RGB color space to YUV, effectively segregating the luminance channel (Y) for targeted contrast enhancement. Subsequently, the image is partitioned into a grid of non-overlapping tiles, upon which a discrete histogram equalization is performed, adhering to a pre-established contrast ceiling. This pivotal step involves the meticulous clipping of the histogram, ensuring that the contrast enhancement remains within the bounds of noise control.

Following equalization, the enhanced luminance channel is then recombined with the chrominance channels (U and V), and the composite is transformed back to the RGB color space. The culmination of this process is subjected to a comparative analysis with traditional histogram manipulation methods to validate the efficacy of CLAHE in augmenting image clarity while preserving the integrity of the original details.

1.3 Experimental Results

(a) (b)

Figure 1: (a) the house_ori image and (b) the demosaiced image by bilinear interpolation

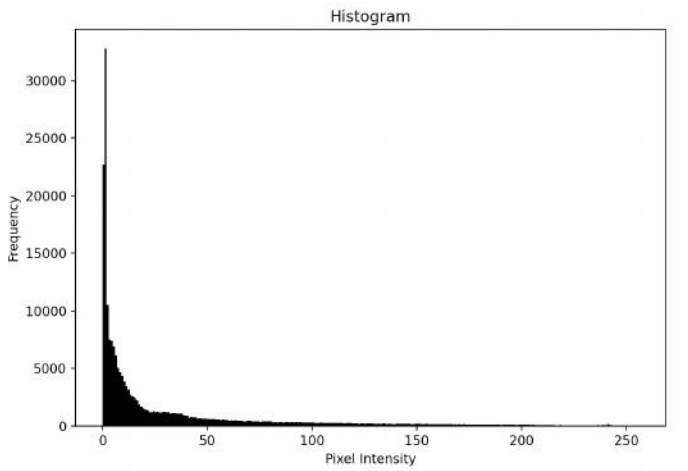


Figure 2: Histogram of original DimLight image

(a) (b)

Figure 3: (a) original DimLight (b) after transfer function

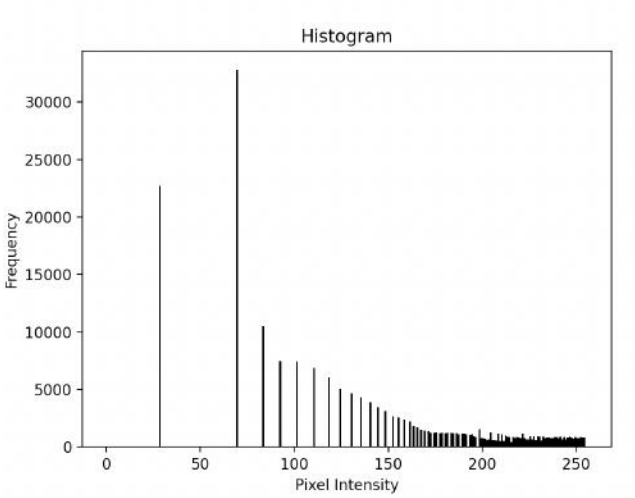
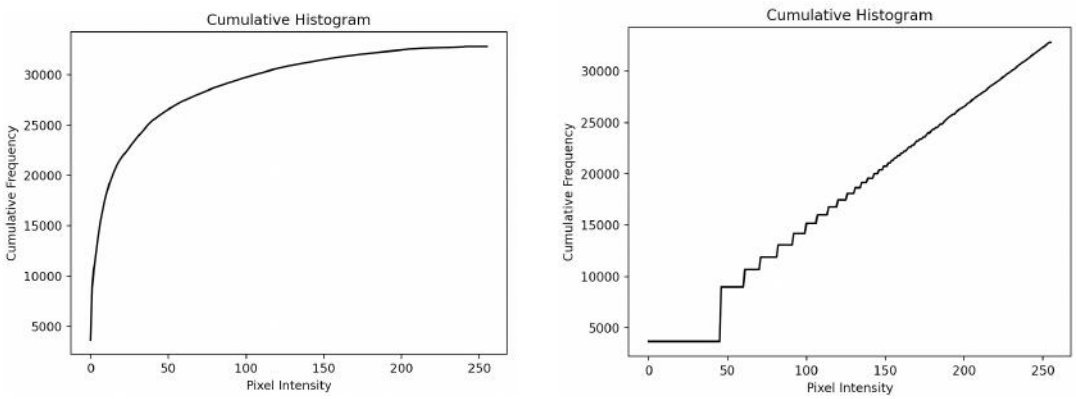
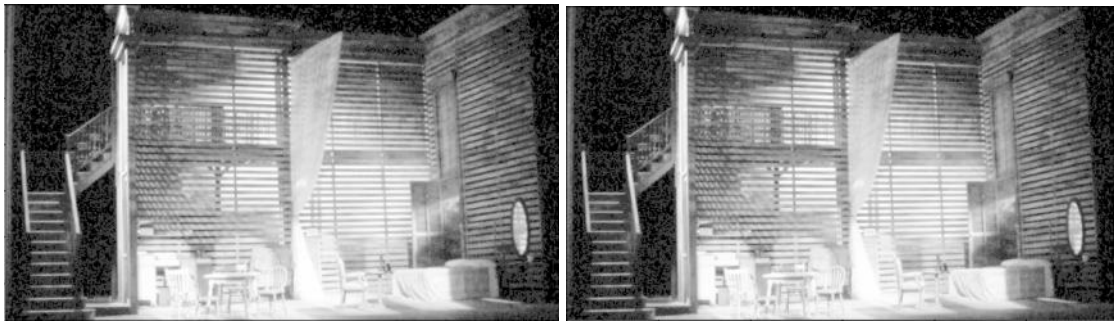


Figure 4: Histogram of transfer function



(a) (b)

Figure 5: (a) original DimLight cumulative graph (b) after bucket filling



(a) (b)

Figure 6: (a) after transfer function (b) after bucket filling

(a)

(b) (c)

Figure 7: (a) original City image; (b) after transfer function on Y channel; (c) after bucket filling on Y channel

Figure 8: after CLAHE with 4x4 number of clips and 20 clip limit

1.4 Discussion

Figure 1 shows that using bilinear demosaicing to reconstruct the full image might miss some details. We can see colors mixing into areas where they should not, especially where there are sharp changes from light to dark. This mix-up occurs because the method averages nearby colors without checking if they are part of the same object or area. To fix this and make the image clearer, we could use more advanced methods that take into account the changes in color and texture, such as bicubic interpolation or methods that adjust for how colors change across the image.

Figure 6 showcases that the resultant images enhanced by both the transfer function and bucket filling methods exhibit negligible differences when subjectively assessed. However, from a theoretical standpoint, the transfer function approach is generally more adept at enhancing global contrast and in achieving a more uniform histogram distribution.

In figures 7 and 8, CLAHE slightly improves image clarity. However, it also introduces visible lines between the tiles it processes. No matter how much we adjust the settings, these lines still show up, which seems to be a basic downside of how CLAHE works on some images. This is especially important to consider when we need a perfectly smooth image without any visible breaks or sections.

Problem 2: Image Denoising

2.1 Abstract and Motivation

The challenge of refining images to remove random specks of color and intensity, known as 'noise,' is central to enhancing image quality. This problem delves into three noise reduction techniques: linear filtering, smoothing out the image; bilateral filtering, preserving edges while removing noise; and non-local means filtering, using a wider range of pixels for a more comprehensive cleanup. Our goal is to implement these techniques and measure their effectiveness using the peak signal-to-noise ratio (PSNR).

$$\text{PSNR (dB)} = 10\log_{10}\left(\frac{\text{Max}^2}{\text{MSE}}\right)$$

$$\text{where MSE} = \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\left(Y(i,j)-X(i,j)\right)^2$$

X : Original Noise-free Image of size $N \times M$

Y : Filterd Image of size $N \times M$

Max: Maximum possible pixel intensity = 255

The motivation stems from the need to retain the image's original details while reducing the noise as much as possible. The Law of Large Numbers, a fundamental principle of probability, guides our approach. It suggests that by averaging a set of observations, we can filter out the randomness and get closer to the expected result - in this case, a clearer image. The following sections will introduce the algorithms we implemented, the results of our experiments, and our recommendations for their use.

2.2 Approach and Procedures

For the first part of the problem, we implement a linear filter utilizing two distinct sets of parameters: the uniform weight function and the Gaussian weight function, to then compare their respective performances.

Next, we aim to use bilateral filtering, a nonlinear method, to denoise an image while preserving its edges. This technique differs from traditional linear filters by considering both the spatial proximity and the intensity similarity when averaging pixel values, thus preventing edge blurring.

$$Y(i,j) = \frac{\sum_{k,l} I(k,l) w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$

$$w(i,j,k,l) = exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_c^2} - \frac{\|I(i,j) - I(k,l)\|^2}{2\sigma_s^2}\right)$$

Following, the non-local mean (NLM) filtering technique is selected for its advanced capability to denoise images by utilizing information from a broader area around each pixel, as opposed to simply considering a local mean. This method assesses the similarity of pixels over larger patches of the image, which helps in preserving detailed structures and reducing noise more effectively.

$$Y(i,j) = \frac{\sum_{k,l} I(k,l) w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$

$$w(i,j,k,l) = \exp\left(-\frac{\left\|I(N_{i,j}) - I(N_{k,l})\right\|_{2,a}^2}{h^2}\right)$$

$$\left\|I(N_{i,j}) - I(N_{k,l})\right\|_{2,a}^2 = \sum_{n_1,n_2 \in \aleph} G_a(n_1,n_2)\left(I(i-n_1, j-n_2) - I(k-n_1, l-n_2)\right)^2$$

$$G_a(n_1,n_2) = \frac{1}{\sqrt{2\pi}a} \exp\left(-\frac{n_1^2+n_2^2}{2a^2}\right)$$

For the last part, we will address the challenge of removing mixed types of noise from a RGB colored image. Mixed noise may include a combination of various noise types such as Gaussian, salt-and-pepper, speckle, or others that can degrade image quality. The selection of algorithms will be based on the nature of the noise identified.

2.3 Experimental Results

(a) (b)

Figure 9: (a) Gaussian weight function (b) Uniform weight function

Figure 10: Bilateral filtering

Figure 11: Non-Local Means (NLM) Filtering

(a) (b)

Figure 12: (a) Noisy flower image; (b) after Gaussian and Median filter

2.4 Discussion

Figure 9 presents the denoised floral representations subjected to Gaussian and Uniform weighting functions, respectively. Upon comparative analysis with the original image beset by noise, both denoising methodologies yield substantial improvements. Subjective evaluation through visual inspection does not reveal marked distinctions between the two; however, objective assessment based on the Peak Signal-to-Noise Ratio (PSNR) metric suggests a marginal superiority of the Gaussian method. This is evidenced by its slightly elevated PSNR value of 28.9151 dB, as opposed to the Uniform method's PSNR of 28.5032 dB.

Figure 10 displays the result of using a special kind of noise-cleaning filter on an image called a bilateral filter, with settings of 12 for intensity and 16 for how wide the filter reaches. This method made the image much clearer, with its PSNR reaching 31.7978 dB, which has improved from the results of Gaussian and Uniform methods. Bilateral filtering works by considering both how close pixels are to each other and how similar their colors are. The intensity setting controls how much the filter pays attention to differences in color, smoothing out areas where the colors are similar. The setting for how wide the filter reaches determines how far around a pixel the filter looks to decide what's similar. This helps keep the edges in the image sharp while getting rid of noise, which is why we see a higher PSNR—it means a cleaner image with less blur.

Figure 11 shows the result of the denoised image using the NLM filter, which uses a patch size of 10 and a search window size of 21. The filter strength, set by a value called h, was 16.0, and the filter's blur level, controlled by a value called sigma, was 10.0. After trying out different settings, these numbers gave us the best cleaned-up image. Even though the settings were fine-tuned, you can still see a bit of noise left when you look closely at the image. When we measure the image quality with PSNR, it's a bit lower at 28.1105 dB compared to other methods we tried before.

The noisy image (b) of figure 7 in homework instruction, displaying a grainy look that usually means there's Gaussian noise. It also has random specks or dots which suggest there's impulse noise, often called salt-and-pepper noise. These assumptions are made by looking at what these kinds of noise generally look like.

For the last part, the noisy image has a grainy look that usually means there's Gaussian noise. It also has random specks or dots which suggest there's impulse noise, often called salt-and-pepper noise. These guesses are made by looking at what these kinds of noise generally look like. To clean up different kinds of noise in a picture, we can use two steps. First, we use a median filter, which is good at getting rid of the random dots of noise that look like specks of salt and pepper. Following, either a Gaussian filter or a bilateral filter is good for smoothing out the fuzzy noise that's spread all over the picture, like a thin layer of dust.

In Figure 12, we first introduced a median filter with a kernel size of 5 to reduce salt-and-pepper noise, effectively removing random speckles by replacing each pixel with the median of its neighbors. Following this, a Gaussian filter with the same kernel size and a sigma of 3 is applied, smoothing the image by weighting closer pixel values more heavily. While these techniques significantly diminish noise, they also slightly blend the noise into the image and reduce sharpness, highlighting the delicate balance between noise reduction and image clarity in digital image processing.

Problem 3: Watercolor Painting Effect

3.1 Abstract and Motivation

Turning digital pictures into watercolor art with a computer program is a fun way to mix art with technology. We take clear, sharp photos and use special effects to make them look like they were painted with watercolors. This lets us see ordinary photos in a more artistic and emotional way. Our goal is to make it easy for anyone who likes art to try out this cool mix of real-life photos and artsy watercolor effects.

3.2 Approach and Procedures

Our method uses a few steps with digital tools that act like watercolor paints. First, we use a median filter to soften the colors, making them blend like in a real painting. Then we use bilateral filtering technique to smooth the colors but keep the important lines sharp. We do this a couple of times to get it just right. After that, we add a little blur using Gaussian filtering to make it look even more like a painting. The last step is to put it all together into one picture that has both the details of the original photo and the soft, dreamy look of a watercolor painting. The final picture looks like a mix of a real-life photo and a painting, just like watercolor art.

3.3 Experimental Results

(a) (b)

Figure 13: (a) after Step 1 median filter; (b) final output

(a) (b)

Figure 14: (a) final output with median filter N=7; (b) final output with median filter N=11

(a) (b)

Figure 15: (a) final output with K=1; (b) final output with K=10

(a) (b)

Figure 16: (a) Flower_noisy after Step 1 median filter; (b) Flower_noisy final output

(a) (b)

Figure 17: (a) Flower_noisy final output with median filter N=7; (b)Flower_noisy final output with median filter N=11

(a) (b)

Figure 15: (a) Flower_noisy final output with K=1; (b) Flower_noisy final output with K=10

3.4 Discussion

When noise gets mixed into the watercolor filter process, the final output image can end up looking messy and cloudy, subjectively. It's kind of like when a watercolor painting gets smudged – the colors that should be light and flowing get mixed up in a way that wasn't intended. Noise adds random bright and dark spots that shouldn't be there, and when we try to smooth out the colors, these spots turn into smears that make the picture look dirty; like a clean painting that has had mud splashed on it. This takes away from the soft, gentle look that watercolors are known for and makes the whole image less pleasing to look at.