Name:Kuimu Ren
USC ID Number:1473482531
USC Email:kuimuren@usc.edu
Submission Date: 01/27/2024

Problem1

(a) Bilinear Demosaicing

Abstract and Motivation:

As we know, image sensors in digital cameras often use a Bayer array, a color filter array (CFA), to capture color information. However, since each pixel only captures one of the primary colors (R, G, B), demosaicing is necessary to reconstruct the missing color information based on neighboring pixel values. The motivation behind demosaicing is to overcome the limitation of image sensors that capture only one primary color per pixel. By reconstructing the missing color information, the full color image can be obtained, providing a more visually accurate representation.

Approach and Procedures:

Bilinear demosaicing is chosen for this problem, as it is a straightforward and commonly used technique for interpolating color values in the demosaicing process. The goal is to implement bilinear demosaicing and subsequently manipulate the image histogram to enhance certain characteristics of the demosaiced image.

First, we need to get each pixel and pixels in other fields. For edge pixels, I chose two kinds of boundary extensions here, zero padding and mirror reflection(Odd).

Second, for each pixel to be interpolated, it is calculated by its neighboring pixels around it.

Assume for a 3 x 3 window((2,2) is the center point):

For (2,2) is green and (2,1) is red:

$R(2,2) = (R(2,1) + R(2,3)) / 2$
$G(2,2) = G(2,2)$
$B(2,2) = (B(1,2) + B(3,2)) / 2$

For (2,2) is green and (1,2) is red:

$R(2,2) = (R(1,2) + R(3,2)) / 2$
$G(2,2) = G(2,2)$
$B(2,2) = (B(2,1) + B(2,3)) / 2$

For (2,2) is red:

$R(2,2) = R(2,2)$
$G(2,2) = (G(2,1) + G(2,3) + G(1,2) + G(3,2)) / 2$
$B(2,2) = (B(1,1) + B(1,3) + B(3,1) + B(3,3)) / 2$

For (2,2) is blue:

$R(2,2) = (R(1,1) + R(1,3) + R(3,1) + R(3,3)) / 2$
$G(2,2) = (G(2,1) + G(2,3) + G(1,2) + G(3,2)) / 2$
$B(2,2) = B(2,2)$

Experimental Results

Shown below are the results for Problem 1(a):

(a)

(b)

(c)

Figure1:(a)The original image and(b)demosaiced image and (c)color image House

Discussion and Answer

(1) Figure1

(2) From (a) to (b), we can see that the image can indeed be reconstructed by the method of bilinear demosaicing, and the original grayscale image can be reconstructed into a color image, which makes up for the shortcomings of the image sensor.
But we can observe any artifacts by bilinear demosaicing, and these artifacts often occur in edges or high-contrast regions, which indicates that the true color may not be accurately represented when interpolating color values in edges or high-contrast regions. Moreover, bilinear demosacicing is a simple interpolation method, which easily leads to the loss of image details in the high-frequency information region. In addition, we can see that there are a lot of blurred places in the image, this is because bilinear demosacicing amplifies the noise, resulting in a rough shape of the image.
So we should consider more advanced demosaicing techniques to reduce the appearance of image artifacts, such as MHC Demosaicing, and we can combine other noise reduction methods to improve the quality of image reconstruction.

(b) Histogram Manipulation

Abstract and Motivation:

As we know, histogram is an important tool used to describe the distribution of image pixel values in image processing. The histogram of an image represents the frequency of occurrence of each pixel value in the image, that is, the grayscale distribution of the image.

Histogram equalization is a technique used for image enhancement, aiming to adjust the contrast and brightness of an image to make it visually more appealing. The method involves redistributing the pixel values in the image to achieve a more uniform distribution of intensity levels, thereby improving the overall quality of the image. And we will talk about two methods about histogram equalization.

Approach and Procedures:

Method A: the transfer-function-based method

The Transfer Function method is one implementation of histogram equalization. It involves defining a transfer function that maps the pixel values of the original image to the equalized pixel values, thereby achieving histogram equalization.

First, convert the original image to grayscale to obtain the grayscale histogram of the image.

Second, compute the Cumulative Distribution Function(CDF) based on the grayscale histogram, which represents the cumulative frequency of each grayscale level. This can be achieved by summing up the grayscale histogram cumulatively.

Third, normalize the Cumulative Distribution Function (CDF) to the range [0, 1] so that it can be used as the input for the transformation function.

Fourth, use the normalized CDF as the transformation function. Then apply the transformation function to each pixel value of the original image, mapping it to the equalized pixel value.

Method B: the Bucket-Filling method

The Bucket-Filling method in Histogram Equalization is a straightforward and effective approach. It involves mapping pixel values to buckets in a cumulative distribution function, directly achieving histogram equalization without explicitly calculating the cumulative distribution function. This method simplifies the implementation process, especially in real-time image processing scenarios.

First, perform histogram analysis on the input image to obtain the distribution of pixel values.

Second, normalize the histogram and calculate the cumulative distribution function of pixel values.

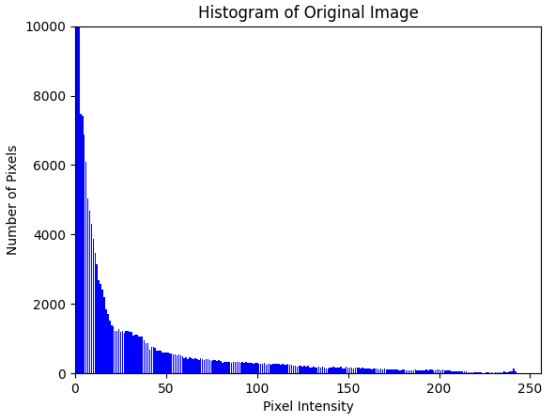
Third, create a bucket for each possible pixel value. The number of buckets is equal to the range of pixel values.

Fourth, divide the range of pixel values evenly among the buckets. Each bucket corresponds to a pixel value, and its cumulative frequency represents the mapped value after equalization.

Fifth, for each pixel in the input image, map its original pixel value to the corresponding bucket. Take the cumulative frequency of the bucket as the equalized pixel value. And assemble the mapped pixel values to generate the equalized image.

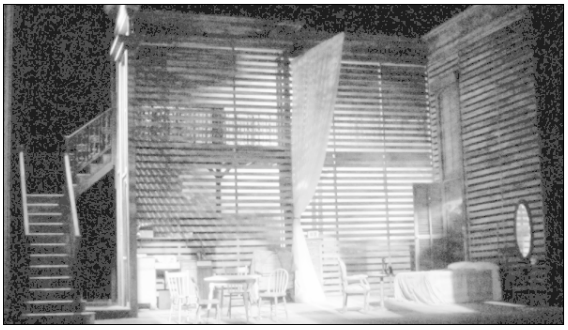
Experimental Results

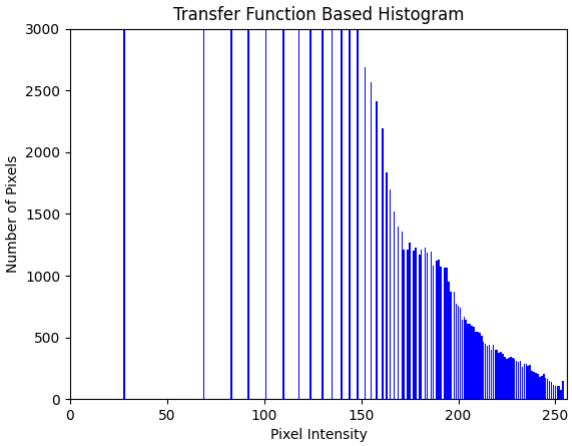
Shown below are the results for Problem 1(b):



(a) (b)

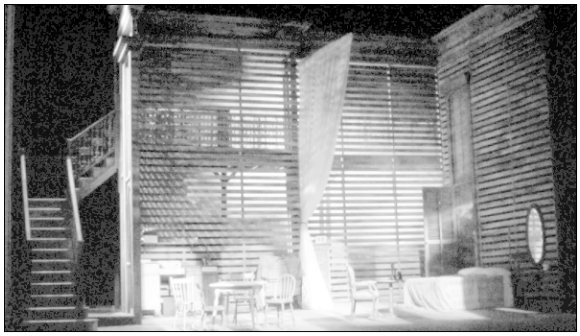
Figure2:(a)The original image and(b)its histogram

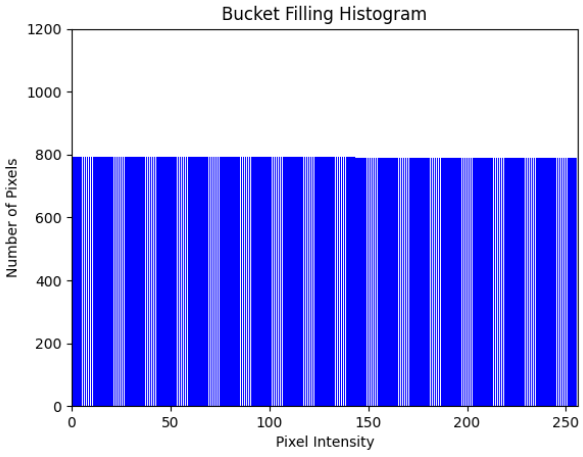




(a) (b)

Figure3:(b)Histogram equalization by transfer-function-based method and (a)its image





(a) (b)

Figure4:(b)Histogram equalization by bucket-filling method and (a)its image

Discussion and Answer

(1) Figure2
(2) Figure3
(3) Figure4
(4) After comparing these two enhanced images, I think method b is better in this case.
Due to its simpler operations, the bucket-filling method is more likely to preserve image details and local features. In contrast, the transfer-function-based method may introduce some nonlinear transformations, leading to detail loss or image distortion.
The bucket-filling method ensures a more uniform distribution of pixel values during equalization.
Moreover, in this example, in the image enhanced by transfer-function-based method, the number of some pixels of its histogram is still too large, which will cause the roughness of the image.

(c) Contrast Limited Adaptive Histogram Equalization

Abstract and Motivation:

Traditional histogram equalization is effective in enhancing the contrast of images by spreading out the intensity values across the entire dynamic range. However, it tends to amplify noise and can result in over-enhancement, especially in regions with already high contrast. To address these limitations, CLAHE method provide contrast enhancement while limiting the amplification **of** noise and preventing over-enhancement. Next I will show the difference between the transfer-function-based method, bucket-filling method and CLAHE method applied to the image.

Approach and Procedures:

First, transform the image from RGB color space to YUV color space.
For each pixel:

$$Y = (0.257 * R) + (0.504 * G) + (0.098 * B) + 16$$
$$U = -(0.148 * R) - (0.291 * G) + (0.439 * B) + 128$$
$$V = (0.439 * R) - (0.368 * G) - (0.071 * B) + 128$$

Second, apply different histogram equalization algorithms on the Y channel to get Y'.
Third, combine Y' with U and V, and transform them back to RGB color space.
For each pixel:

$$R = 1.164 * (Y - 16) + 1.596 * (V - 128)$$
$$G = 1.164 * (Y - 16) - 0.813 * (V - 128) - 0.391 * (U - 128)$$
$$B = 1.164 * (Y - 16) + 2.018 * (U - 128)$$

Experimental Results

Shown below are the results for Problem 1(c):

(a)

(b)

(c)

Figure5:(a)The original image and(b)enhanced image by transfer-function-based method and (c)enhanced image by bucket-filling method

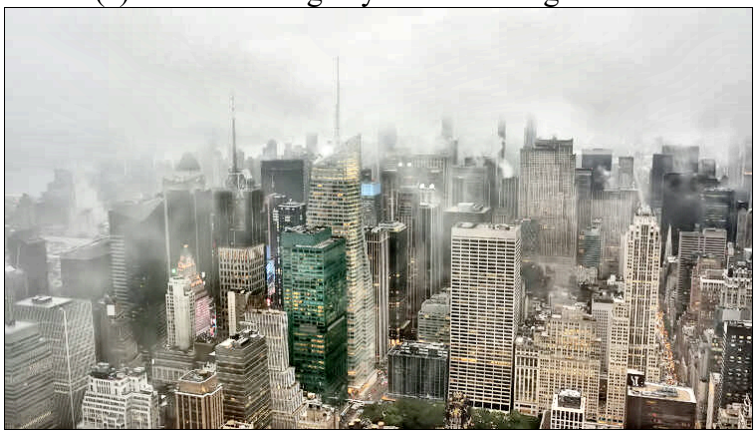


Figure6: enhanced image by CLAHE method

Discussion and Answer

(1) Unlike traditional histogram equalization, which operates globally on the entire image, CLAHE divides the image into smaller regions and performs histogram equalization independently within each region. This adaptive approach allows CLAHE to address variations in contrast across different parts of the image, leading to better results, particularly in regions with varying illumination levels or contrast.

(2) Figure5

(3) Figure6

(4) The transfer-function-based method cannot deal with the case of local non-uniform contrast, and still leads to an excessive number of partial pixel values, and bucket-filling method over-averages the pixel frequencies of the whole image. These two methods are conventional histogram equalization methods that adjust the contrast of the image

globally but cannot effectively address uneven local contrast. Both of these methods lead to the problem of uneven contrast, which is reflected in the image above as some areas are too low bright.

CLAHE divides the image into multiple local regions and performs histogram equalization within each region. By applying restriction and interpolation to each region, it avoids over-enhancement and over-equalization issues and employ local equalization to better handle local contrast variations, resulting in more natural effects.

Problem2

(a) Basic denoising methods

Abstract and Motivation:

In image processing, noise reduction is performed to remove random disturbances present in the image, aiming to enhance the quality and usability of the image. In this problem, I will be using two basic noise reduction methods called uniform weight function and the Gaussian weight function.

Approach and Procedures:

Assume (k, l) is the neighboring pixel location within the window centered around (i, j), I is the image with noise, Y is the filtered image. σ is the standard deviation of Gaussian distribution.

For uniform weight function:

First, for each pixel, calculate the average of its surrounding pixels.

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

Second, Replace the original pixel value with the calculated average.

For Gaussian weight function:

First, for each pixel, calculate the distance to its surrounding pixels.

Second, calculate the weight for each pixel based on a Gaussian function with respect to the distance from the center pixel.

$$w(i,j,k,l) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(k-i)^2 + (l-j)^2}{2\sigma^2}\right)$$

Third, multiply the values of the surrounding pixels by their respective weights and sum them up.

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

Last, replace the original pixel value with the weighted sum.

Experimental Results

Shown below are the results for Problem 2(a):

(a)


(b) (c)

Figure7:(a)The original image and(b)denoised image by uniform weight function and (c) denoised image by Gaussian weight function

Discussion and Answer

(1) Figure7, PSNR for uniform weight function is 27.1837, PSNR for Gaussian weight function is 27.3835

As we know, uniform weight function and Gaussian weight function are both linear filter to the noisy image. So in the above two figures, the difference between the two methods is not very large.

(b) Bilateral Filtering

Abstract and Motivation:

Bilateral filtering is a non-linear filtering technique used in image processing for edge-preserving smoothing. I will introduce and show how it works.

Approach and Procedures:

Assume (k, l) is the neighboring pixel location within the window centered around (i, j), I is the image with noise, Y is the filtered image. σ_c and σ_s are parameters of your choice

First, generate a spatial kernel that represents the spatial proximity between pixels. This kernel defines the extent of smoothing based on spatial distance.

$$\exp\left(-\frac{(i-k)^2+(j-l)^2}{2{\sigma_c}^2}\right)$$

Second, generate an intensity kernel that represents the similarity between pixel intensities. This kernel defines the extent of smoothing based on intensity similarity.

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

Third, combine the spatial and intensity kernels to form the bilateral filter kernel. This combined kernel determines the extent of smoothing based on both spatial proximity and intensity similarity.

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

Fourth, convolve the bilateral filter kernel with the input image. At each pixel, compute a weighted average of its neighboring pixels' intensities, where the weights are determined by the bilateral filter kernel.

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

Last, generate the filtered image by applying the bilateral filter to each pixel in the input image.

Experimental Results

Shown below are the results for Problem 2(b):

Figure8: Denoised image by bilateral filter

Discussion and Answer

(1) Figure8, PSNR for bilateral filter is 24.1504

(2) σ_c determines how much the filter should consider the spatial distance between pixels when computing the weighted average and σ_s determines how much the filter should consider the difference in pixel intensities when computing the weighted average.
A smaller σ_c value results in a more localized filter response, where only pixels in close proximity to the target pixel are considered for smoothing. And a larger σ_c value allows for a broader consideration of spatial distances, resulting in a smoother output but potentially with more blurring across different regions of the image.
A smaller σ_s value results in a more localized filter response, where only pixels with very similar intensities are considered for smoothing. And a larger σ_s value allows for a broader consideration of intensity differences, resulting in a smoother output but potentially with less preservation of edges and details.

(3) Bilateral filter performs better than linear filters. Because unlike average and Gaussian filters which perform linear smoothing, bilateral filtering is a non-linear filtering technique. This allows it to adaptively smooth different regions of the image based on local features, leading to better preservation of image details. Bilateral filtering considers both spatial proximity and intensity similarity when smoothing an image. This means it

tends to preserve edges by only smoothing areas with similar intensities, while preserving strong intensity changes.

(c) Non-Local Means (NLM) Filtering

Abstract and Motivation:

In image processing, noise removal is a significant challenge. Traditional filtering methods such as mean filtering, mean filtering, and Gaussian filtering typically rely on local pixel information for smoothing. However, these methods often lead to loss of image details or blurring in the presence of strong noise. The motivation behind NLM filtering is to address the limitations of traditional filtering methods.

Approach and Procedures:

Assume I, Y are the noisy and filtered images, respectively, $N_{x,y}$ is the window centered around location (x, y), and h is the filtering parameter, $N' \leq N$ and $M' \leq M$ denote the window size of your choice. where ℵ denotes the local neighborhood centered at the origin, $n1, n2 \in$ ℵ denotes the relative position in the neighborhood window. $a > 0$ is the standard deviation of the Gaussian kernel.

First, define the size of the filter window (usually a square or rectangular window) and the size of the search window used to compute similarity.

Second, for each pixel in the image, apply the filter to its neighborhood region.

Third, for the current pixel position, calculate similarity weights based on the similarity of its neighborhood region with other similar regions in the entire image.

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

Gaussian weighted Euclidian distance can be used to calculate the weights.

$$||I(N_{i,j}) - I(N_{k,l})||_{2,a}{}^2 = \sum_{n1,n2 \in \aleph} G_a(n_1,n_2)((I(i - n_1, j - n_2) - I(k - n_1, l - n_2)))^2$$

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

Fourth, use the computed similarity weights to perform weighted averaging of the pixel values around the current pixel. Pixels with higher weights contribute more to the average.

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

Fifth, repeat steps 3 and 4 for every pixel position in the image until the entire image has been processed.

Last, once processing is complete, output the image after being filtered by the Non-Local Means filter.

Experimental Results

Shown below are the results for Problem 2(c):

Figure9: Denoised image by NLM filter

Discussion and Answer

(1) Figure9, PSNR for NLM filter is 29.5895
the big search window size is 7
the small neighbor window size is 21
strength of the filter is 10

(2) From the figure, we can clearly see that the NLM filter performs much better than the previous method. Because NLM filter utilizes information from similar regions in the image, preserves more image details, effective on various noise types including salt-and-pepper noise and Gaussian noise.

(d) Denoising for color images

Abstract and Motivation:

For color images, we need to choose appropriate filters to balance denoising and preserve image details. And with a certain sequence, you can smooth color images while preserving image details and deal with various types of noise to get sharper and more accurate image results.

Approach and Procedures:

First, convert the color image from the RGB color space to the YUV or HSV color space. This conversion helps to separate the luminance information from the chrominance information.

Second, apply denoising techniques to the luminance channel in the YUV or HSV color space. We can use bilateral filtering, NLM filtering, Gaussian weight function, Median filtering to denoise image.

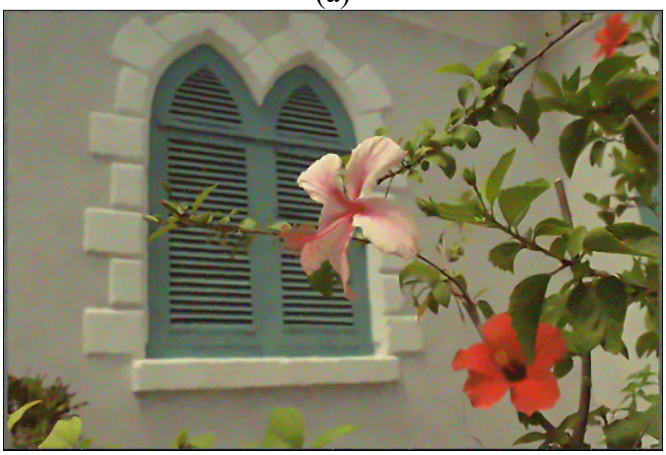
Third, convert the processed image back to the RGB color space.

Last, calculate quality metrics such as PSNR (Peak Signal-to-Noise Ratio) between the processed image and the original image to evaluate the effectiveness of denoising.

Experimental Results

Shown below are the results for Problem 2(d):

(a)



(b)

Figure10:(a)The original image and (b) Denoised image by NLM filter and Median filter

Discussion and Answer

(1) We can see from the figure that there are many colored and random pixels. Such burst discrete noise points do not conform to Gaussian distribution and belong to impulse noise.

(2) I chose to use the Median filter first to reduce the number of colored pixels and then chose to use the NLM filter to perform local noise reduction and preserve the details and textures of the image.

(3) Figure10, PSNR for Red channel is 21.7673, PSNR for Green channel is 21.6137, PSNR for Blue channel is 22.03

Because the Median filter is a nonlinear method, it cannot completely eliminate all colored patches, but there are still some lighter colored patches. Although the NLM method can remove noise while preserving details, some details and textures are still blurred from the image.

Block-matching and 3D filtering (BM3D) is an image denoising algorithm that leverages block matching and three-dimensional filtering techniques to reduce image noise. BM3D effectively reduces various types of noise, including Gaussian noise, salt-and-pepper noise, while preserving image details and textures.

Problem3

Abstract and Motivation:

From the previous implementation, we can find that different filters have different effects, so we can remove noise from the image from different perspectives by combining multiple filters, while preserving image clarity and details as much as possible to obtain a higher-quality image output. For example, linear combination combines the results of bilateral and Gaussian filtering, leveraging the strengths of both filters to further improve image quality.

Approach and Procedures:

First, by Median Filtering:

For each pixel in the original color image, select the median pixel value in its N neighborhood. This median value becomes the representative color for the pixel in an output image denoted as Im.

Second, by Bilateral Filtering:

Apply the previously implemented bilateral filter to the output image Im. Set the window size, σ_c, and σ_s. Repeat the bilateral filter K times. The resulting output image is denoted as Ib.

Third, by Gaussian Filtering:

Apply Gaussian blur to the original color. The output image is denoted as Ig.

Last, Linear Combination:

Linearly combine the output Ib and the output Ig to obtain the final output Iout. The combination is calculated using the formula Iout = 1.4 * Ib - 0.4 * Ig.

Experimental Results

Shown below are the results for Problem 3:

(a) (b)

Figure11:(a) Im and (b) Iout

(a) (b)

Figure12:(a) Iout with N = 7 and (b) Iout with N = 11

(a) (b)

Figure13:(a) Iout with K = 1 and (b) Iout with K = 10

Figure14: Iout with source image is Flower_noisy

Discussion and Answer

(1) Figure11

(2) Figure12.

From the figure above, we can see that the Iout for N=11 is a little bit more blurred and darker than the Iout for N=7.

For N=7, a smaller window size results in a stronger emphasis on local details and textures. This is because the median filtering with N = 7 only consider a smaller neighborhood, resulting in little variations in details and textures across the image.

For N=11, a larger window size leads to a more extensive smoothing effect. With a larger neighborhood considered during median and bilateral filtering, the resulting image tends to appear smoother and more continuous. This can reduce noise in the image but also result in some loss of fine details.

(3) Figure13

It can be seen from the figure that when k=1, the brightness of the picture is very high, and when k=10, the brightness of the picture is very dark.

The bilateral filter is effective at reducing noise while preserving edges and details in the image. However, with each iteration of the bilateral filter, the image undergoes further smoothing. This gradual smoothing process can lead to a reduction in overall contrast and brightness in the image. Additionally, the parameters in linear combination further amplifies the effect of bilateral filter. This may result in an overall decrease in the brightness of the final output image.

(4) Figure14

When we use Flower_noisy as input, we can see that the result not only meets the Watercolor Painting effect, but also effectively reduces the noise types such as salt and pepper noise.

We can find that linearly combining the results of bilateral filtering and Gaussian filtering achieve better denoising results. Therefore, it is necessary to choose the appropriate filtering method and parameters according to the actual situation in image processing, and then the linear combination method can be used to achieve the best denoising effect.