1 Exercises

Please answer the following questions in the report.

1.1 Color Spaces (10 Points)

1. Give one example where the HSI color space is advantageous to the RGB color space. Also give one example where RGB is advantageous to HSI. (5 Points)

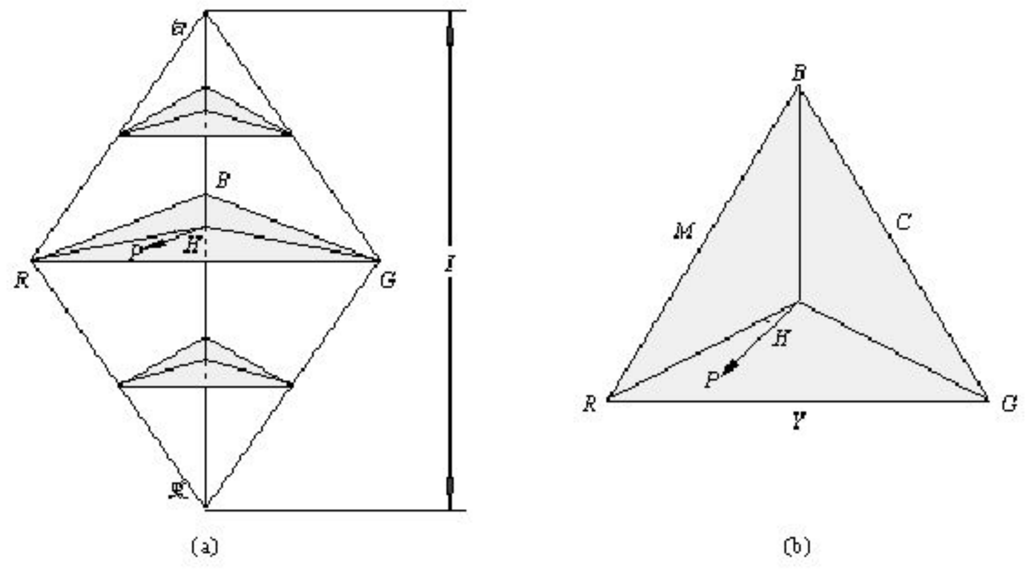
2. What is the effect of adding 60 to the Hue components on the R, G, and B components of a given image? (5 Points)

Sol:

1、Different model has different advantage, for example, if we want to change the intensity of an image, HSI color space is advantageous to the RGB color space because it only need to change the Intensity matrix while the RGB model needs to change three matrix R、G、B.

If we want to find the complement color for a image, RGB is advantageous to HSI because simple linear operation can solve this problem, we only need to apply 255-(originValue) for each item to get the result.

2、The Hue describe the property of pure color, which indicates which color is the basis color for this image, if we change the Hue value, we changes the color of the image,



The increase of H will not change S and I, in this way, if the origin basic color is red, after adding 60 to the Hue components, the basic color becomes yellow, in the RGB component, the R component decreases while the G component increase. Similarly, if the origin basic color is green, after adding 60 to the Hue components, the basic color becomes cyan, in the RGB component, the G component decreases while the B component increase. In general, it will decrease one of the R、G、B, and increase one of the R、G、B to change to another basic pure color.

1.2 Color Composition (10 Points)

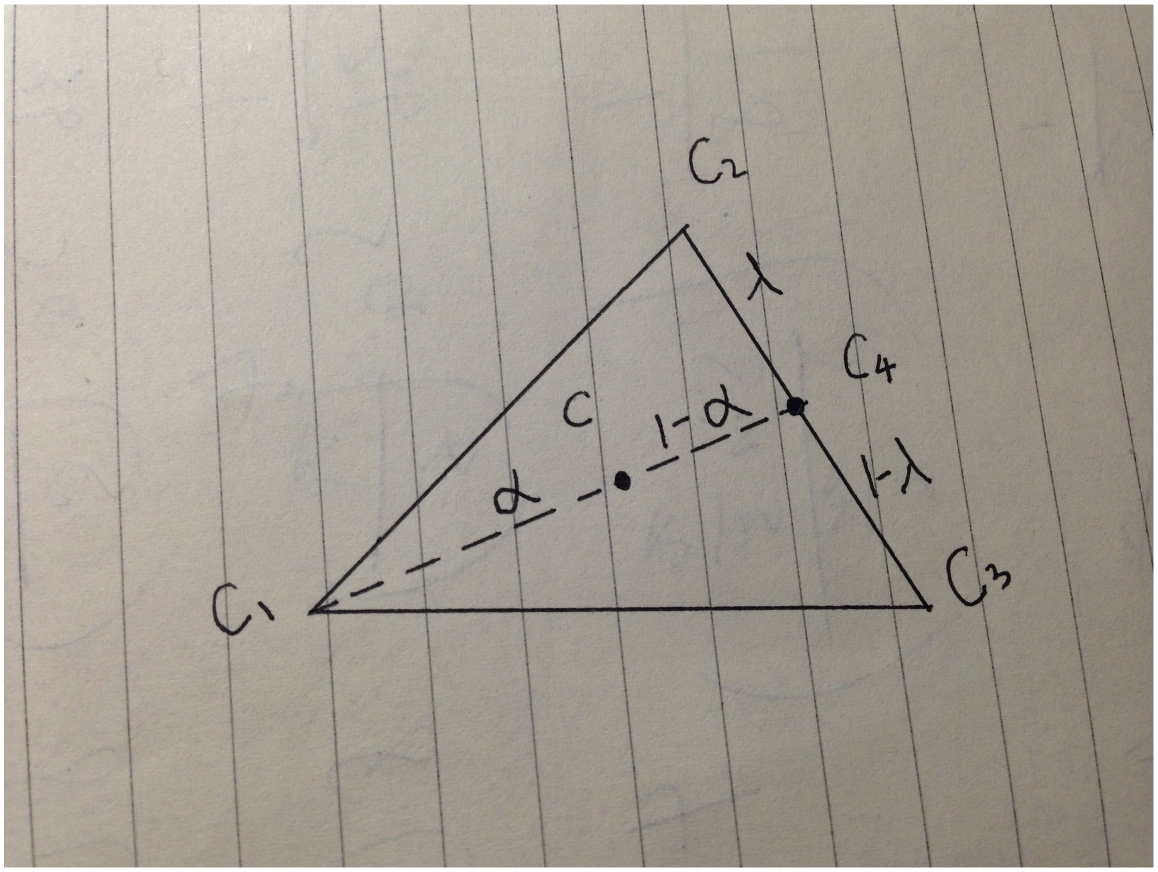
Consider any three valid colors c_1; c_2 and c_3 with coordinates $(x_1; y_1)$; $(x_2; y_2)$ and $(x_3; y_3)$, in the chromaticity diagram in Fig. 1 (or Fig. 6.5 of the textbook).

Derive the general expression(s) for computing the relative percentages of c1; c2 and c3 composing a given color that is known to lie within the triangle whose vertices are at the coordinates of c1; c2 and c3.

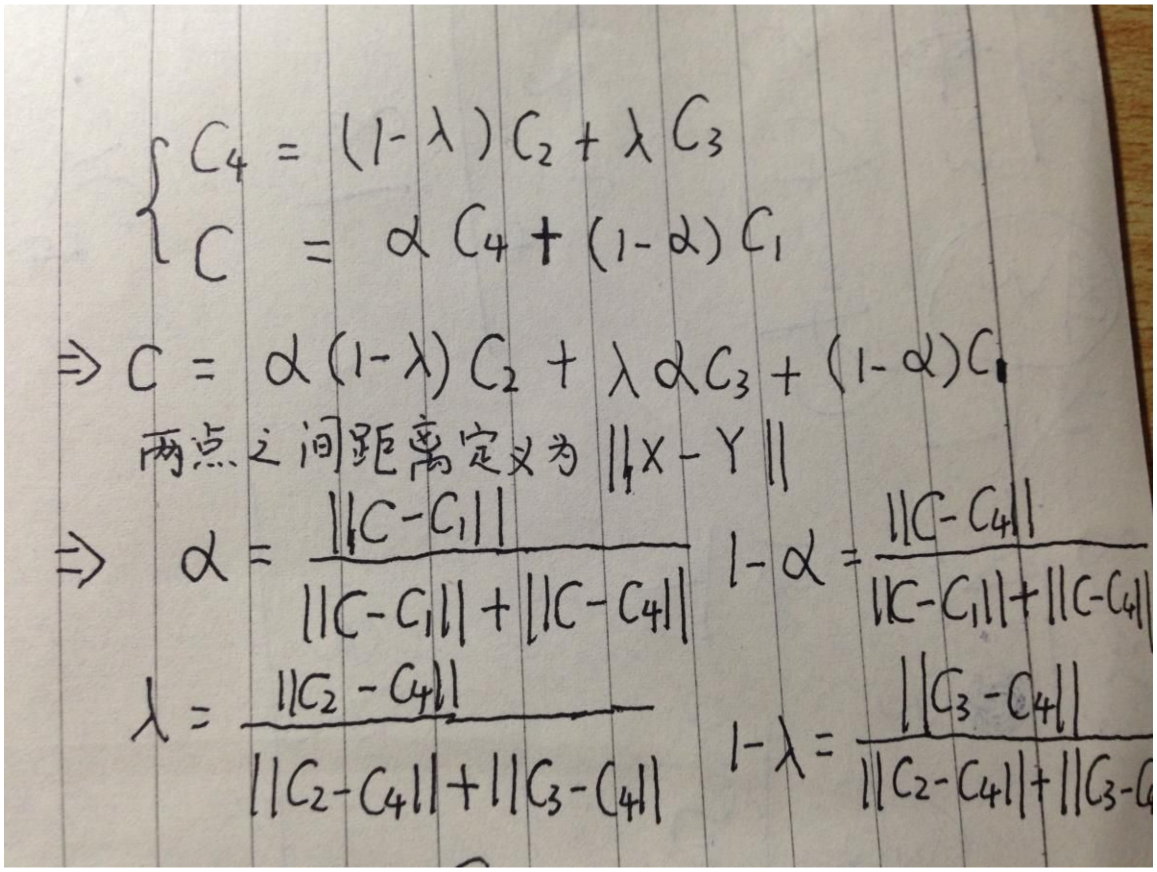
Sol:

Suppose the color we want to get has coordinate (x,y), denoted as C, is a point lying within the triangle whose vertices are at the coordinates of c1; c2 and c3.

First we choose c1, draw a line from C1 to C which join the line(C2-C3) with C4(x4,y4), first we need to represent C4 with C2 and C3, then we use C4 and C1 to represent C.



In this way, we need to compute the relation as following figure:



$$\begin{cases} C_4 = (1-\lambda)C_2 + \lambda C_3 \\ C = \alpha C_4 + (1-\alpha)C_1 \end{cases}$$

$$\Rightarrow C = \alpha(1-\lambda)C_2 + \lambda\alpha C_3 + (1-\alpha)C_1$$

两点之间距离定义为 $\|X-Y\|$

$$\Rightarrow \alpha = \frac{\|C-C_1\|}{\|C-C_1\|+\|C-C_4\|} \qquad 1-\alpha = \frac{\|C-C_4\|}{\|C-C_1\|+\|C-C_4\|}$$

$$\lambda = \frac{\|C_2-C_4\|}{\|C_2-C_4\|+\|C_3-C_4\|} \qquad 1-\lambda = \frac{\|C_3-C_4\|}{\|C_2-C_4\|+\|C_3-}$$

After doing some abbreviation, we can get the following general equation:

C = $\dfrac{\frac{C1}{CC1}+\frac{C2}{CC2}+\frac{C3}{CC3}}{\frac{1}{CC1}+\frac{1}{CC2}+\frac{1}{CC3}}$, CC1 represent the distance between C and C1, so are CC2 and CC3. In this way, we replace the CC1 with $\sqrt{(x-x_1)^2+(y-y_1)^2}$, CC2 with $\sqrt{(x-x_2)^2+(y-y_2)^2}$, CC3 with $\sqrt{(x-x_3)^2+(y-y_3)^2}$ to get the result C =

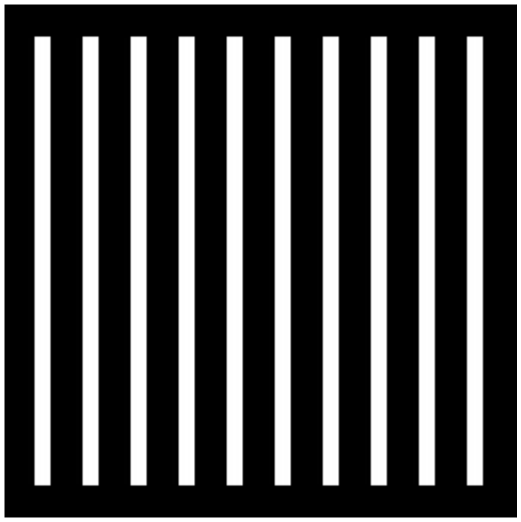
$$\frac{\frac{C1}{\sqrt{(x-x_1)^2+(y-y_1)^2}}+\frac{C2}{\sqrt{(x-x_2)^2+(y-y_2)^2}}+\frac{C3}{\sqrt{(x-x_3)^2+(y-y_3)^2}}}{\frac{1}{\sqrt{(x-x_1)^2+(y-y_1)^2}}+\frac{1}{\sqrt{(x-x_2)^2+(y-y_2)^2}}+\frac{1}{\sqrt{(x-x_3)^2+(y-y_3)^2}}}$$

2 Programming Tasks

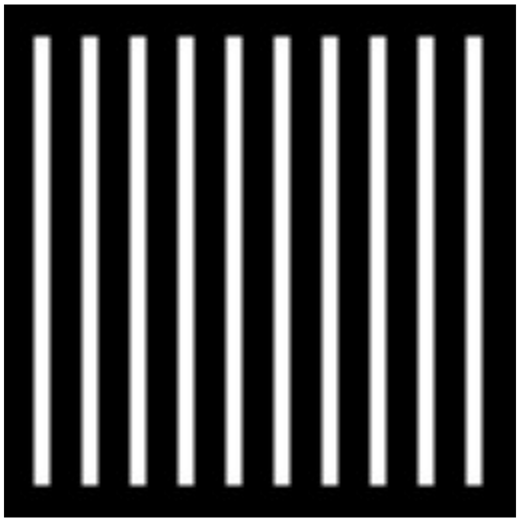
2.2 Image Filtering (10 Points)

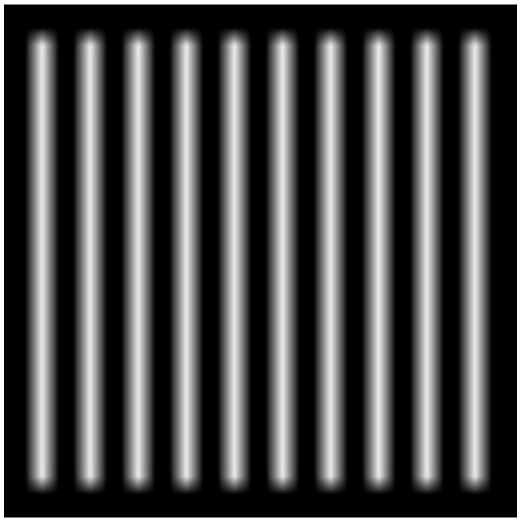
Target The white bars in Fig. 2 are 8 pixels wide and 224 pixels high. The separation between bars is 16 pixels. For other details please refer to your input image.

1. Filter your input image with 3   3 and 9   9 arithmetic mean filters respectively. Paste your two results on the report. Also briefly describe what each result looks like, e.g. The width/height/color of bars in the report. (2 Points)



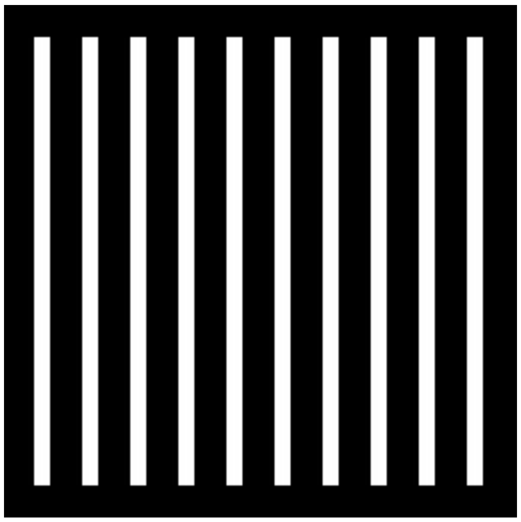
Original   image


3*3 arithmetic mean filter

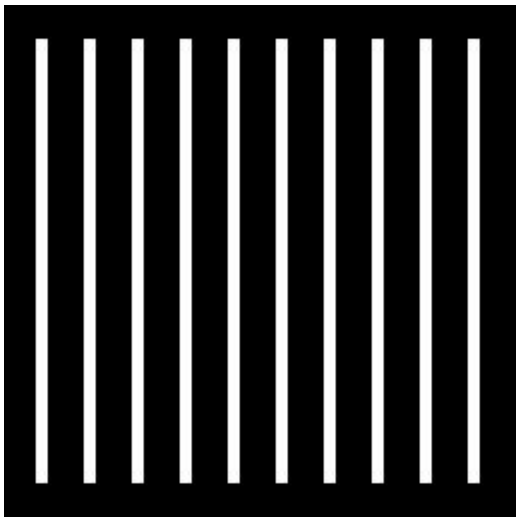

9*9 arithmetic mean filter

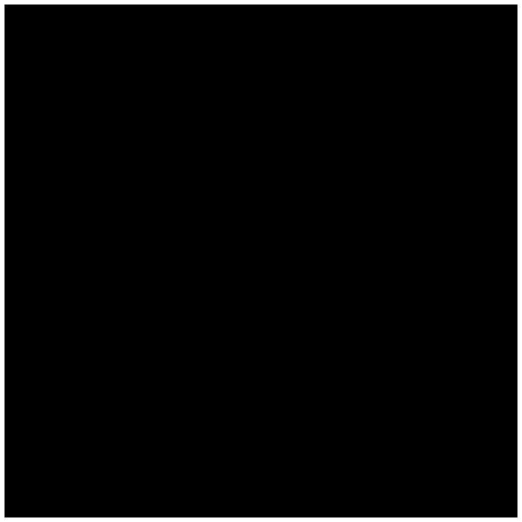
As you can see, the width and height of white bar becomes a little larger, and with the filter becomes larger, the image becomes blur, the value of gray level decrease as the filter becomes larger.

2. Repeat the first application with 3 3 and 9 9 harmonic mean filters. Paste your results in the report and briefly describe what each result looks like. (4 Points)



Original image

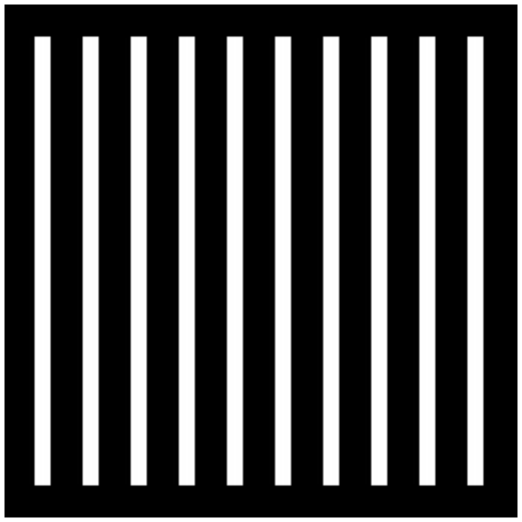




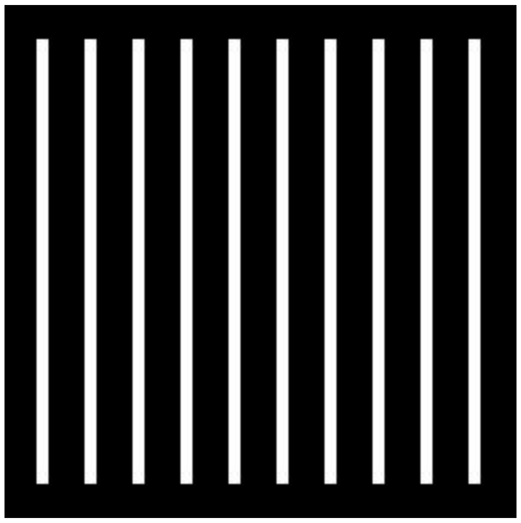
3*3 harmonic mean filter 9*9 harmonic mean filter

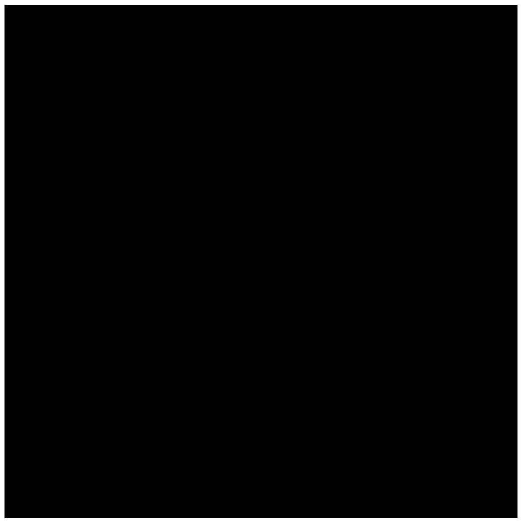
After processing the image with 3*3 harmonic mean filter, the height and width of the white bar decrease since once one of the neighbors is 0, the result of the pixel is 0. But the gray level of the white bar remain the same. After processing the image with 9*9 harmonic mean filter, the image becomes dark and there is no white bar. This is because the white bar is not wide enough, after reducing the width and height by ((9-1)/2)*2, the white bar disappears.

3. Repeat the first application with 3 3 and 9 9 contraharmonic mean filters with Q = -1.5. Paste your results in the report and briefly describe what each result looks like. (4 Points)



Original image





3*3 contraharmonic mean filter 9*9 contraharmonic mean filter

As you can see, when Q < 0, the contraharmonic mean filter functions like harmonic filter, the height and width of the white bar decrease since once one of the neighbors is 0, the result of the pixel is 0. But the gray level of the white bar remain the same, after processing with 9*9 filter, the white bar disappears since the white bar is not wide enough.

2.3 Image Denoising

1. Write a noise generator to add Gaussian noise or salt-and-pepper (impulse) noise to an image. Your generator should be able to specify the noise mean and standard variance for Gaussian noise, and the probabilities of each of the two noise components for salt-and-pepper noise. (5 Points)

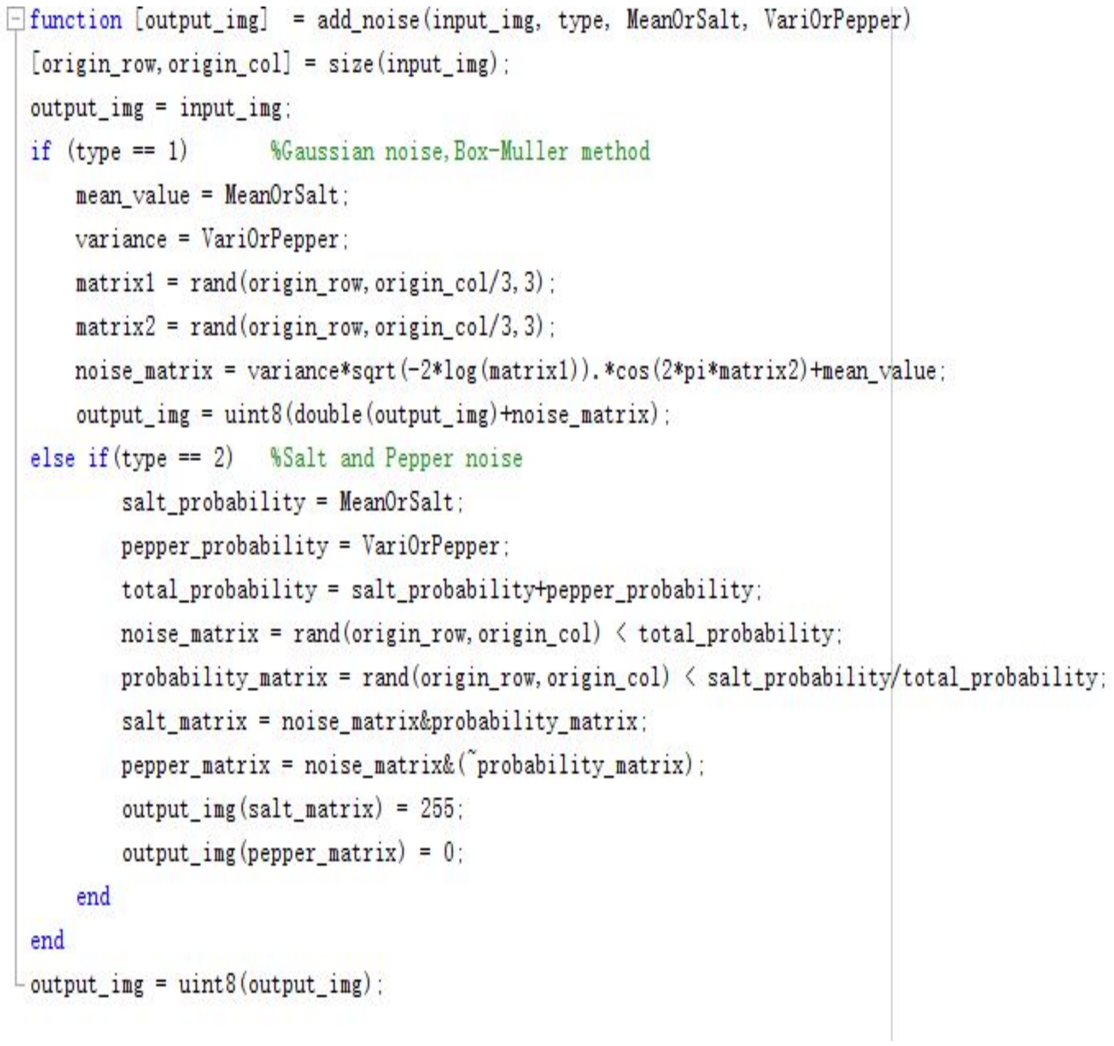
Sol:

For this part, please see the corresponding matlab document add_noise.m,

```
    function [output_img]  = add_noise(input_img, type, MeanOrSalt, 
VariOrPepper)
```

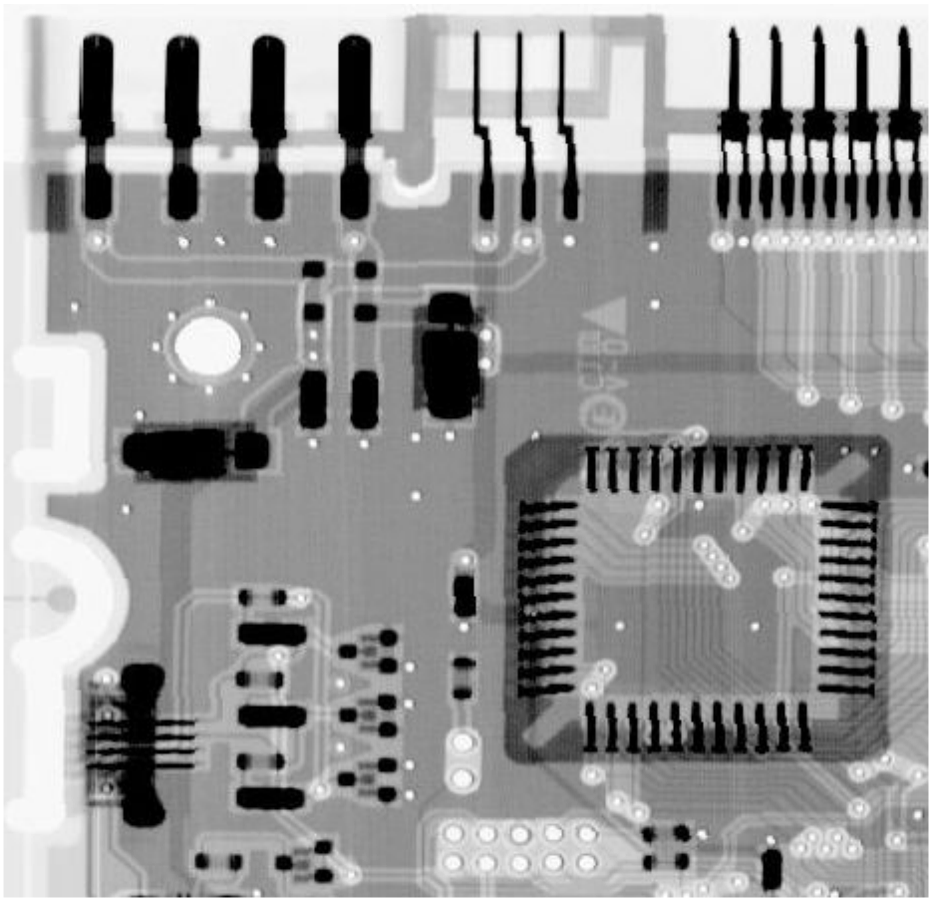
In this function, if the type is 1, we will generate the Gaussian noise with mean value(`MeanOrSalt`) and variance(`VariOrPepper`), if the type is 2, we will generate the salt-and-pepper noise with probability for salt noise(`MeanOrSalt`) and probability for pepper noise(`VariOrPepper`).

The code is like following:

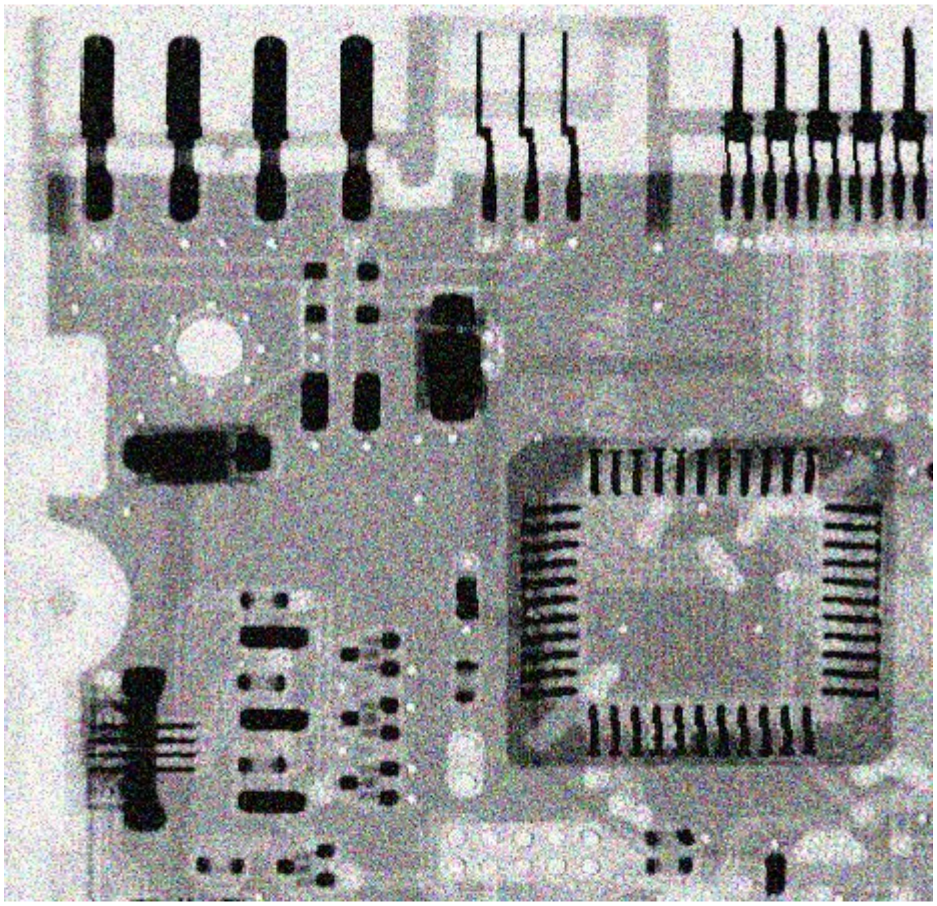


```
function [output_img]  = add_noise(input_img, type, MeanOrSalt, VariOrPepper)
[origin_row,origin_col] = size(input_img);
output_img = input_img;
if (type == 1)       %Gaussian noise,Box-Muller method
    mean_value = MeanOrSalt;
    variance = VariOrPepper;
    matrix1 = rand(origin_row,origin_col/3,3);
    matrix2 = rand(origin_row,origin_col/3,3);
    noise_matrix = variance*sqrt(-2*log(matrix1)).*cos(2*pi*matrix2)+mean_value;
    output_img = uint8(double(output_img)+noise_matrix);
else if(type == 2)   %Salt and Pepper noise
        salt_probability = MeanOrSalt;
        pepper_probability = VariOrPepper;
        total_probability = salt_probability+pepper_probability;
        noise_matrix = rand(origin_row,origin_col) < total_probability;
        probability_matrix = rand(origin_row,origin_col) < salt_probability/total_probability;
        salt_matrix = noise_matrix&probability_matrix;
        pepper_matrix = noise_matrix&(~probability_matrix);
        output_img(salt_matrix) = 255;
        output_img(pepper_matrix) = 0;
    end
end
output_img = uint8(output_img);
```

2. Add Gaussian noise to your input image with mean 0 and standard variance 40, and paste the noisy image in your report. Then try to denoise it via arithmetic mean filtering, geometric mean filtering, and median filtering. Paste your filtering results in the report. Compare these results, and discuss which one looks better / worse, and why, within 1 page.

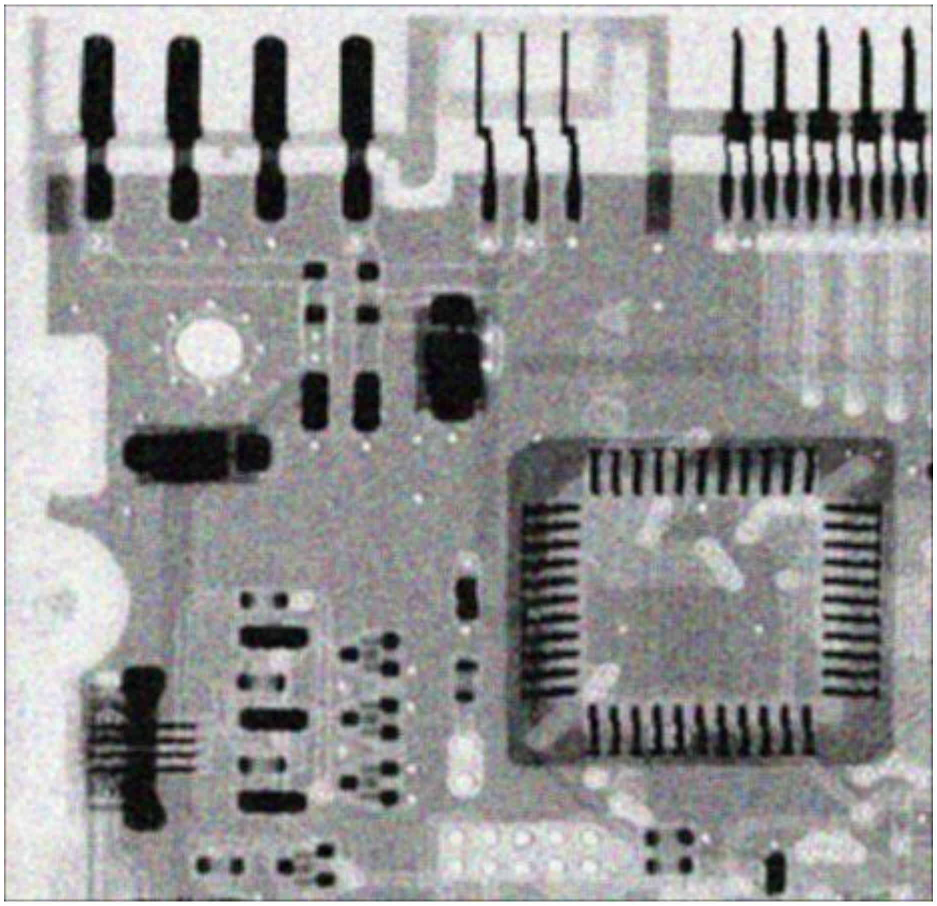


Original image



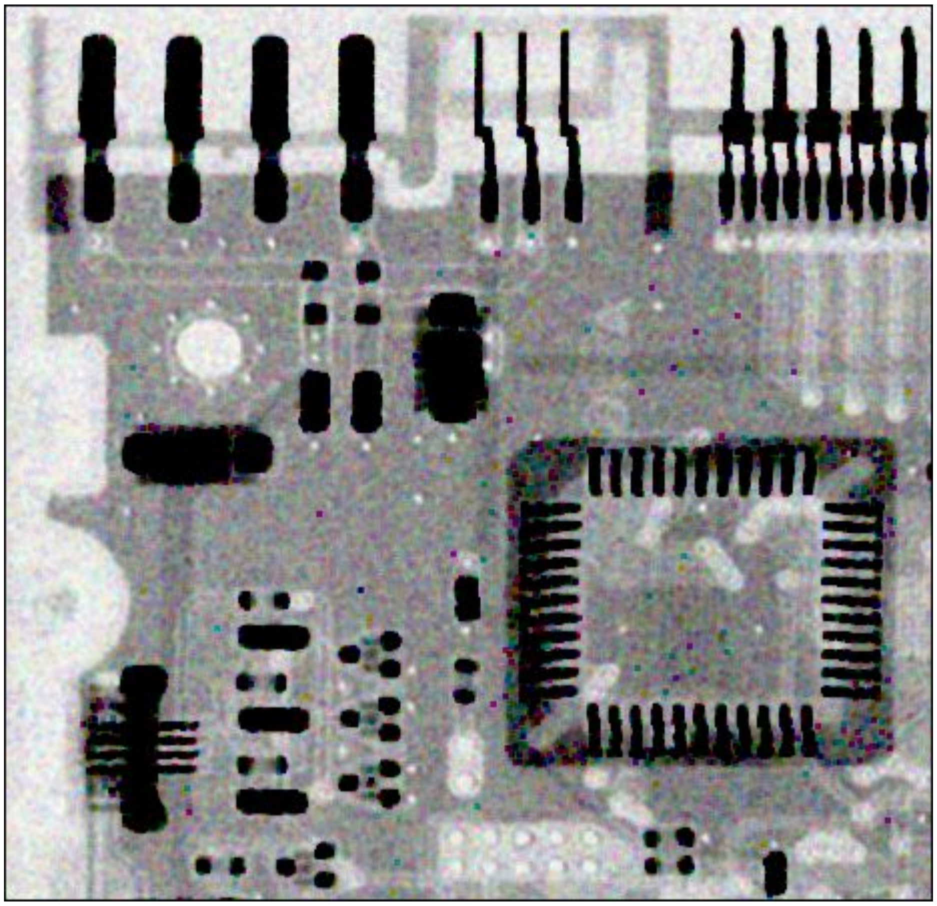
Gaussian polluted image

After adding Gaussian noise, the image becomes rough and blur.



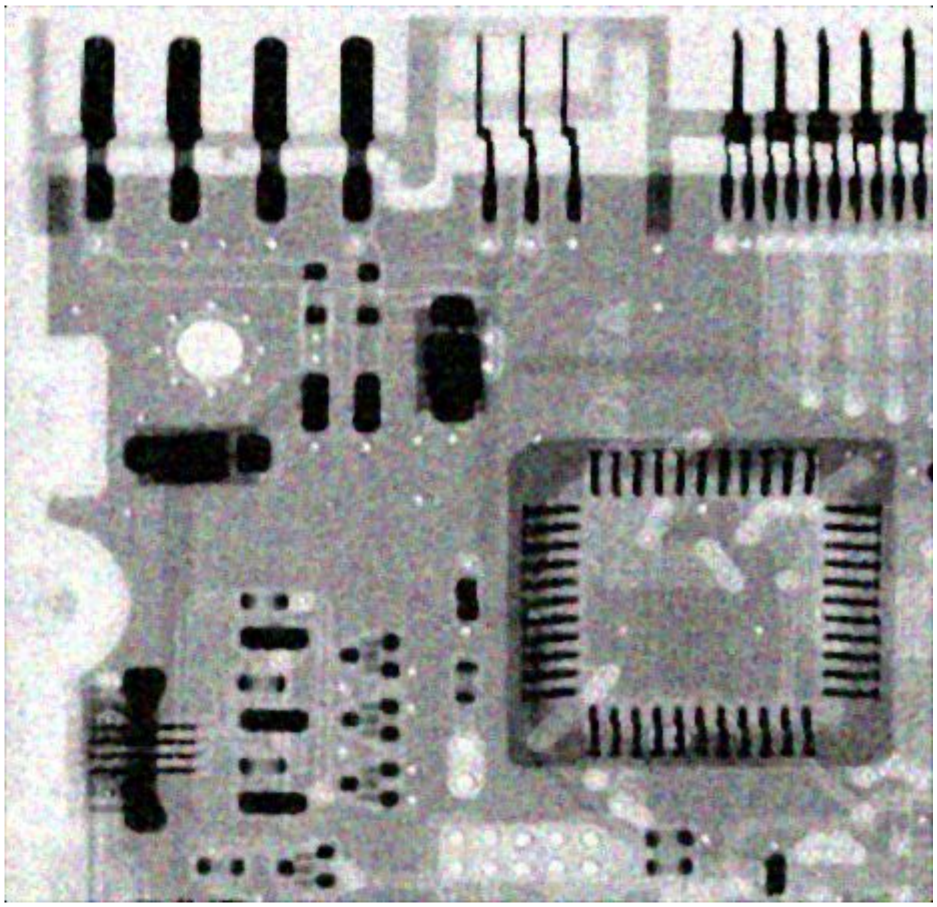
arithmetic mean filter

After filtering with 3*3 arithmetic filter, the image removes most of the noise though the image becomes blur.



geometric mean filter

After filter with geometric mean filter, the image removes noise and is not blur, but the black area becomes larger and some details disappeared because once there is a 0, the neighbor of the pixel will become 0 after filtering.



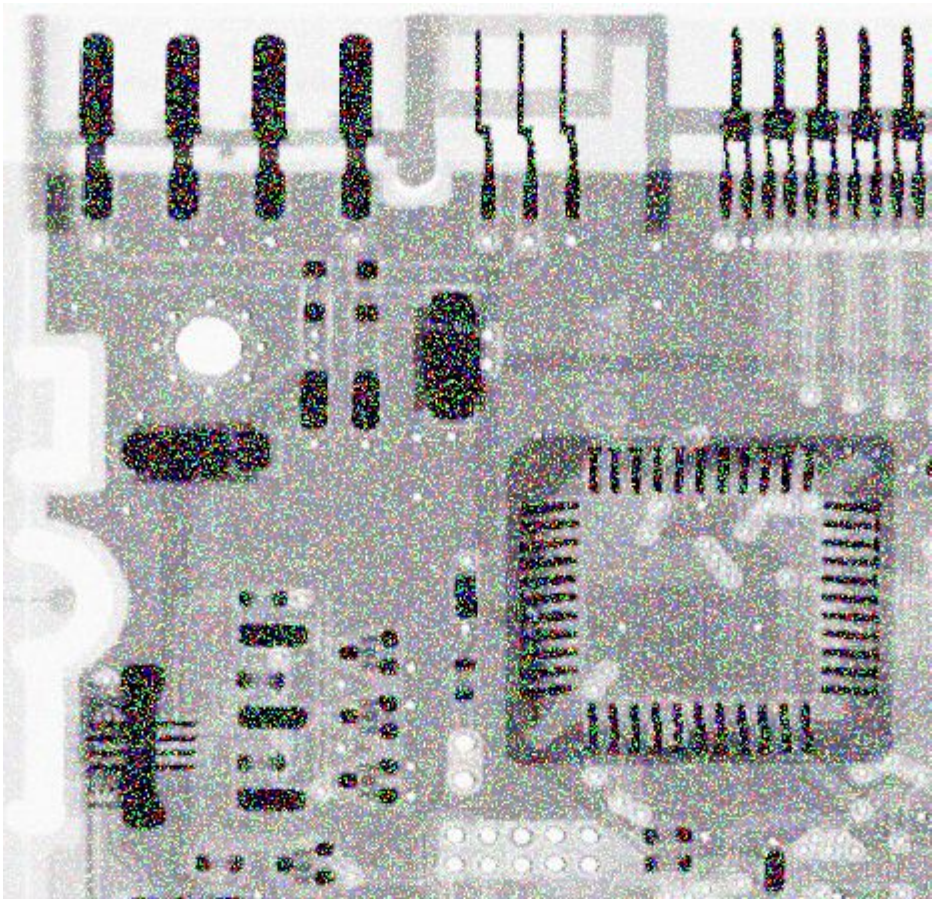
Mean value filter

After processing the image with Mean value filter, most of the noise disappeared and the image is not blur, and the details remains.

As you can see, after adding Gaussian noise on the image, we can use different filters to process it, the arithmetic filter and mean value filter functions the same and are suitable for processing Gaussian noise.

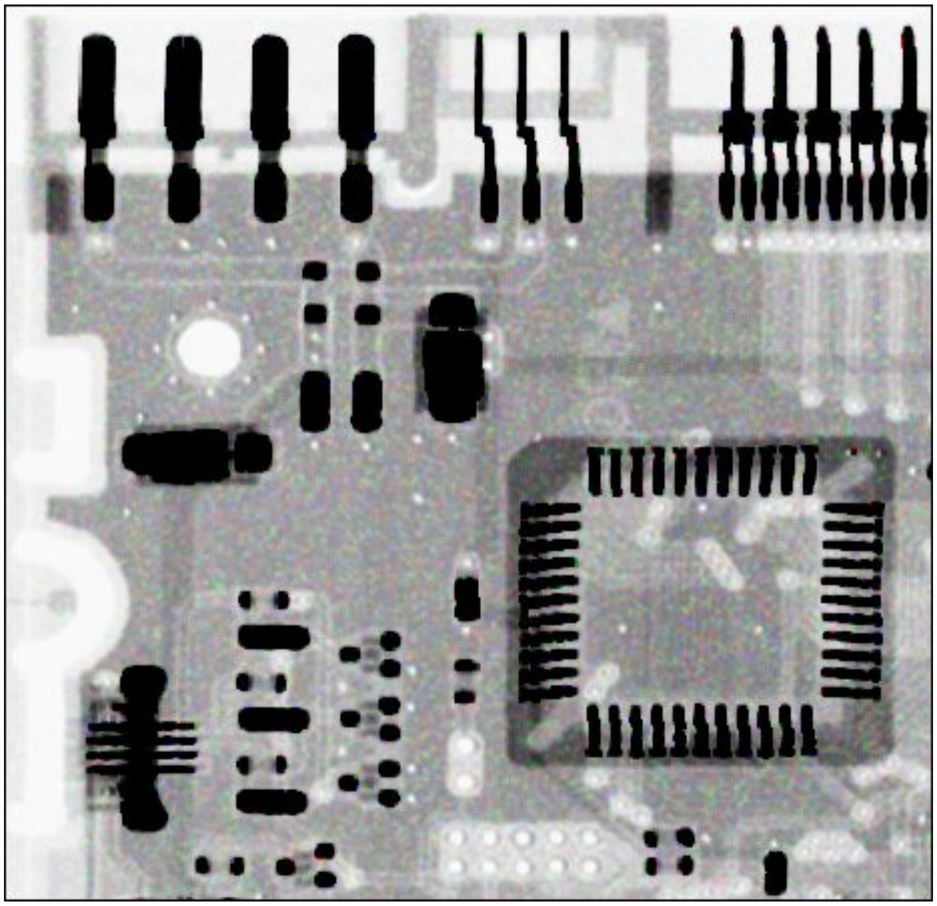
3. Add salt noise to your input image by setting its probabilities to 0:2, and paste the noisy image in your report. Then try to denoise it via harmonic mean filtering and contraharmonic mean filtering. Paste your filtering results in the report. In addition, for contraharmonic mean filtering, you are required to paste two results: one for $Q > 0$ and the other for $Q < 0$.

Discuss why setting a wrong value for Q would lead to terrible results within 1 page.



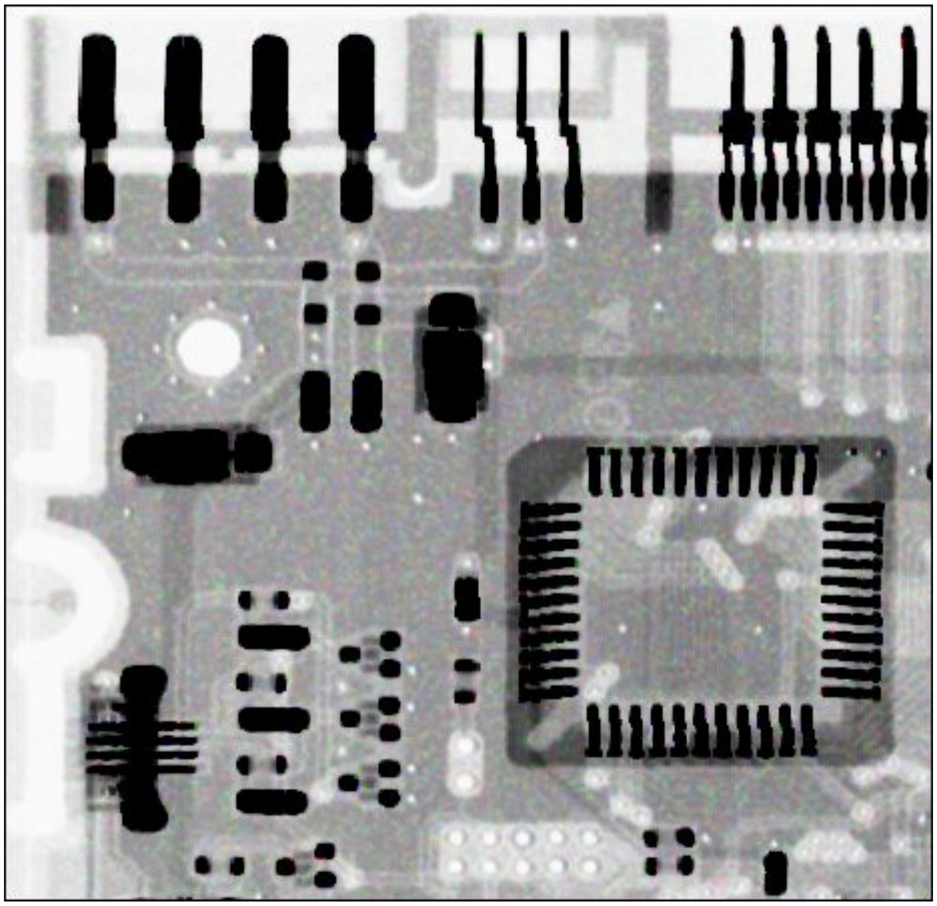
salt polluted image

After adding salt polluted noise, many white pots appear on the image.



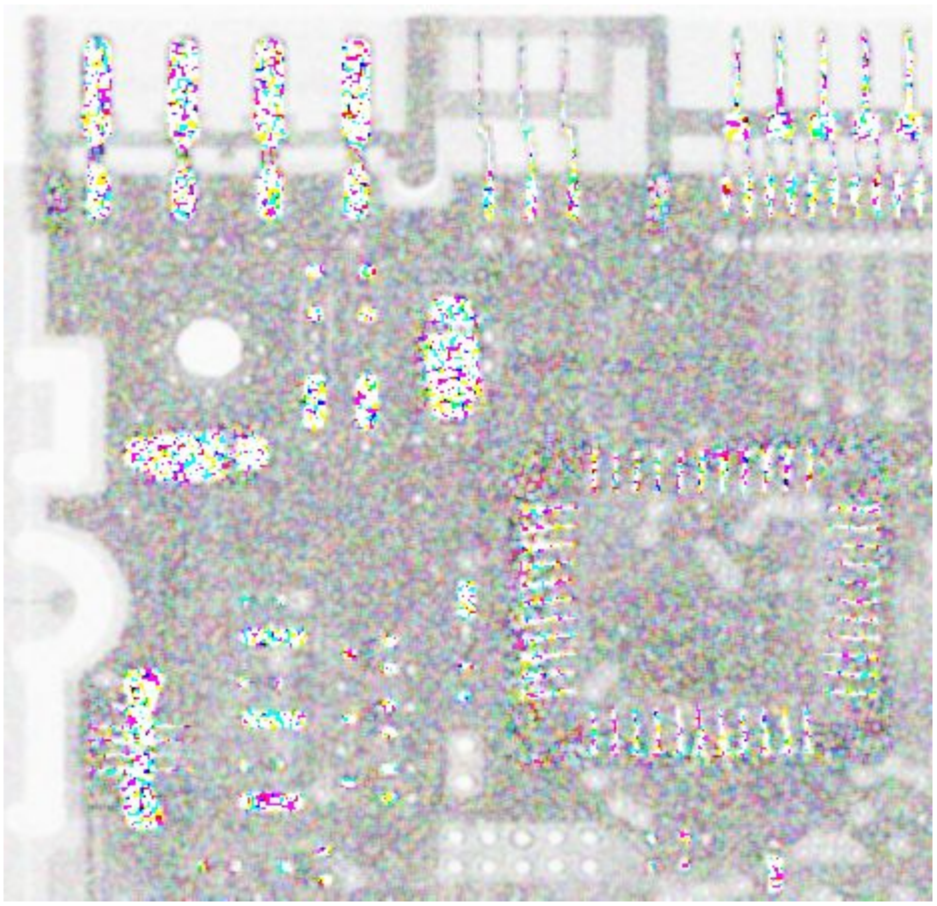
after using harmonic filter

After filtering with harmonic filter, the image removes noise and is not blur, it functions well since the salt noise(255) will contribute little on the filtering and it will not influence the image.



after using contraharmonic filter with $Q = -1.5$

When $Q < 0$, the contraharmonic filter functions like harmonic filter and works well for salt noise.

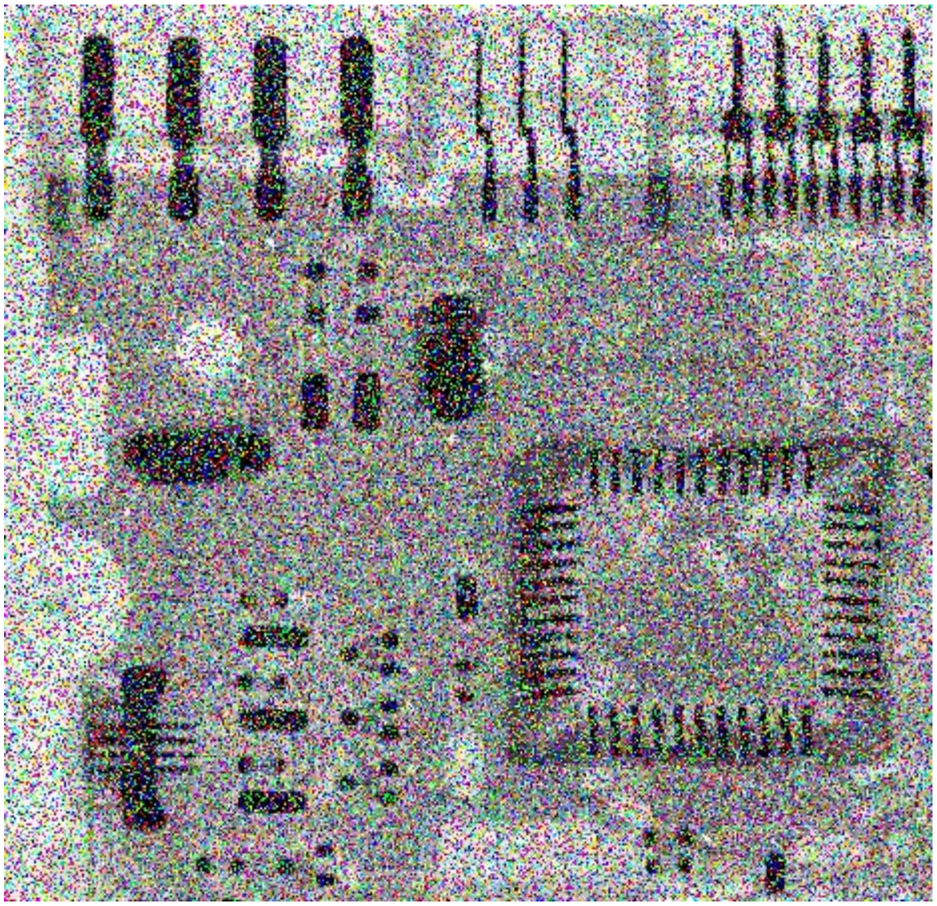


after using contraharmonic filter with $Q = 1.5$

When $Q > 0$, the contraharmonic filter can handle pepper noise but it can't solve salt noise and will worsen the image since the salt noise(255) will contribute a lot on the origin image, turning most of the image to white color, thus it is not good to use contraharmonic filter.

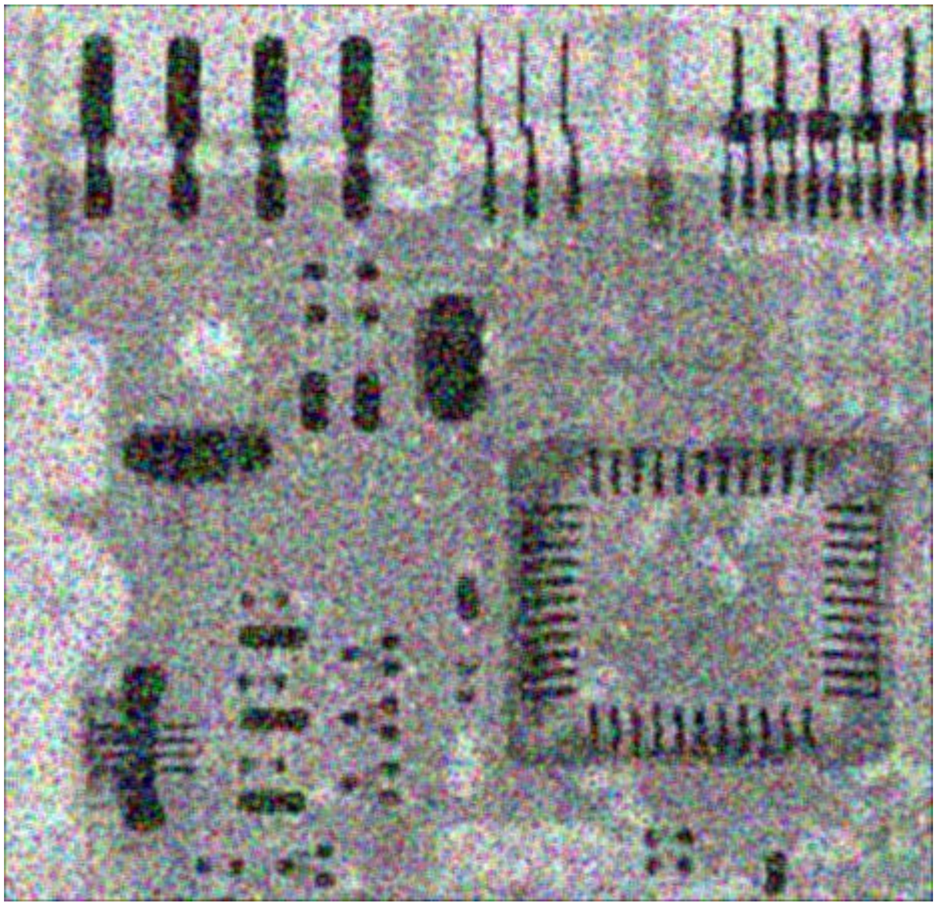
As we can see above, the harmonic filter and contraharmonic filter with $Q < 0$ works well when processing the image with salt noise.

4. Add salt-and-pepper noise to your input image by setting both of the probabilities to 0:2, and paste the noisy image in your report. Then try to denoise it via arithmetic mean filtering, geometric mean filtering, max filtering, min filtering and median filtering. Paste your filtering results in the report. Compare these results, and discuss which one looks better / worse, and why, within 1 page. (10 Points)



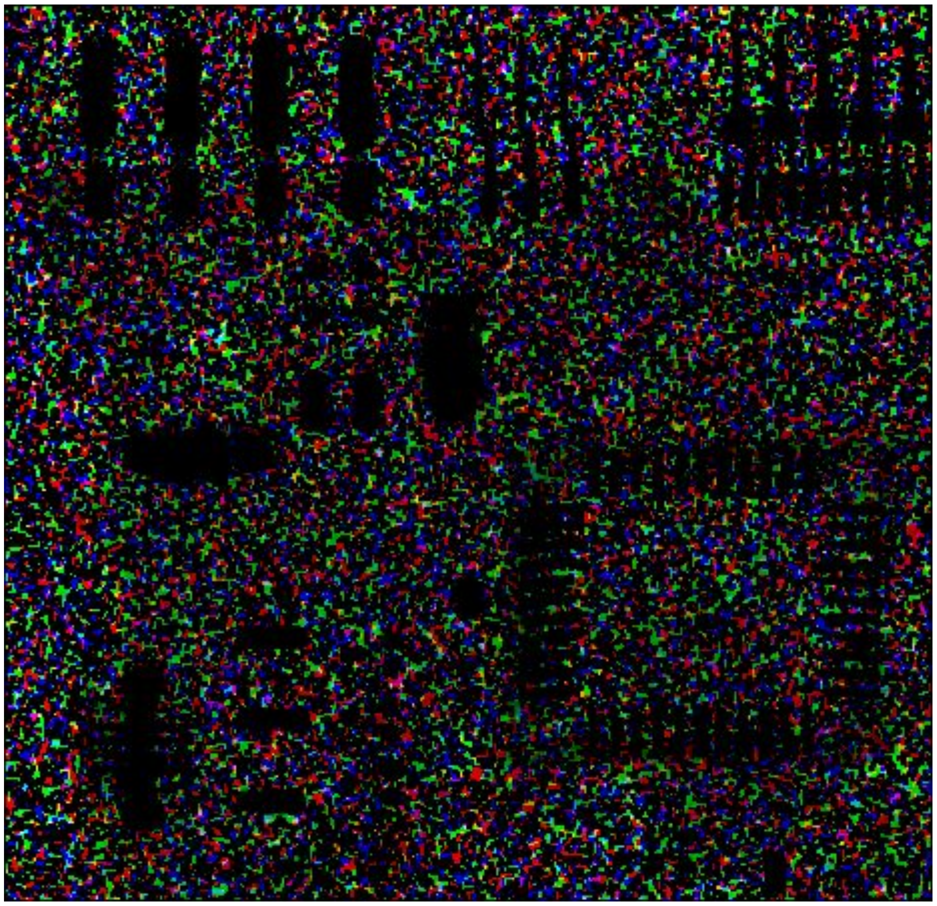
salt-and-pepper polluted image

After adding salt-and-pepper noise, there are many white and black pots on the image.



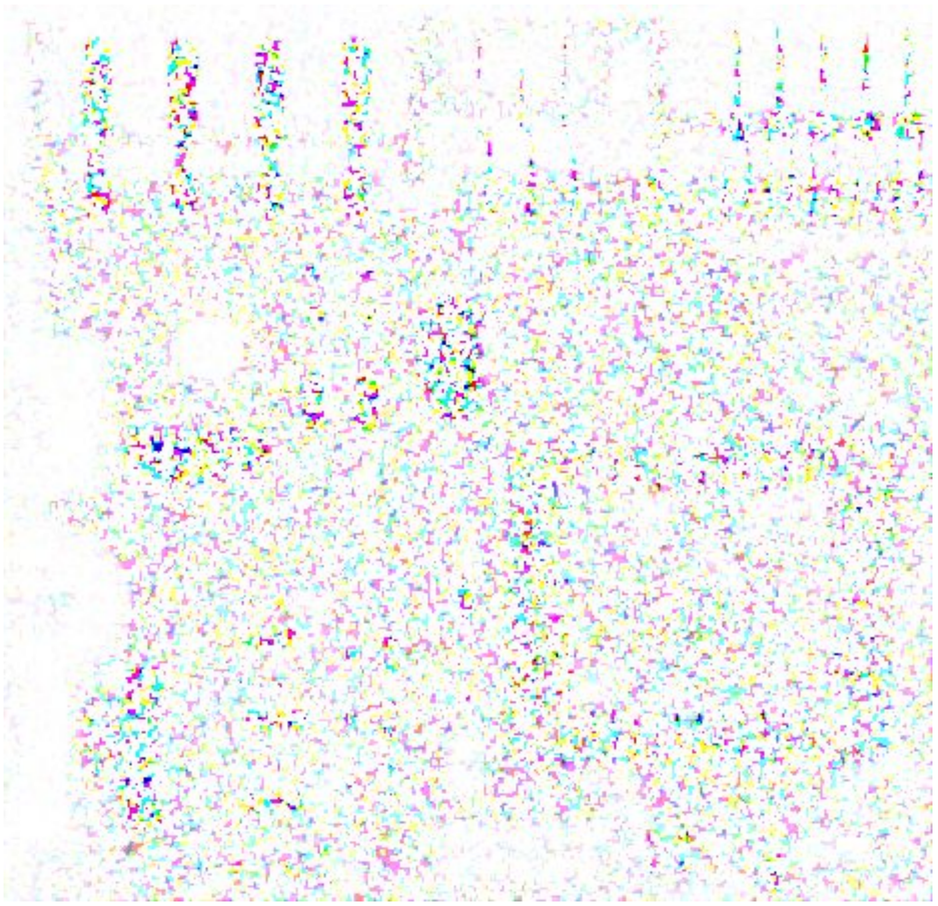
Arithmetic filter

Since the extreme value will have big influence on the filtering, and the salt-and-pepper noise have uniform distribution and the number of noise is very large, thus the arithmetic filter doesn't work well.



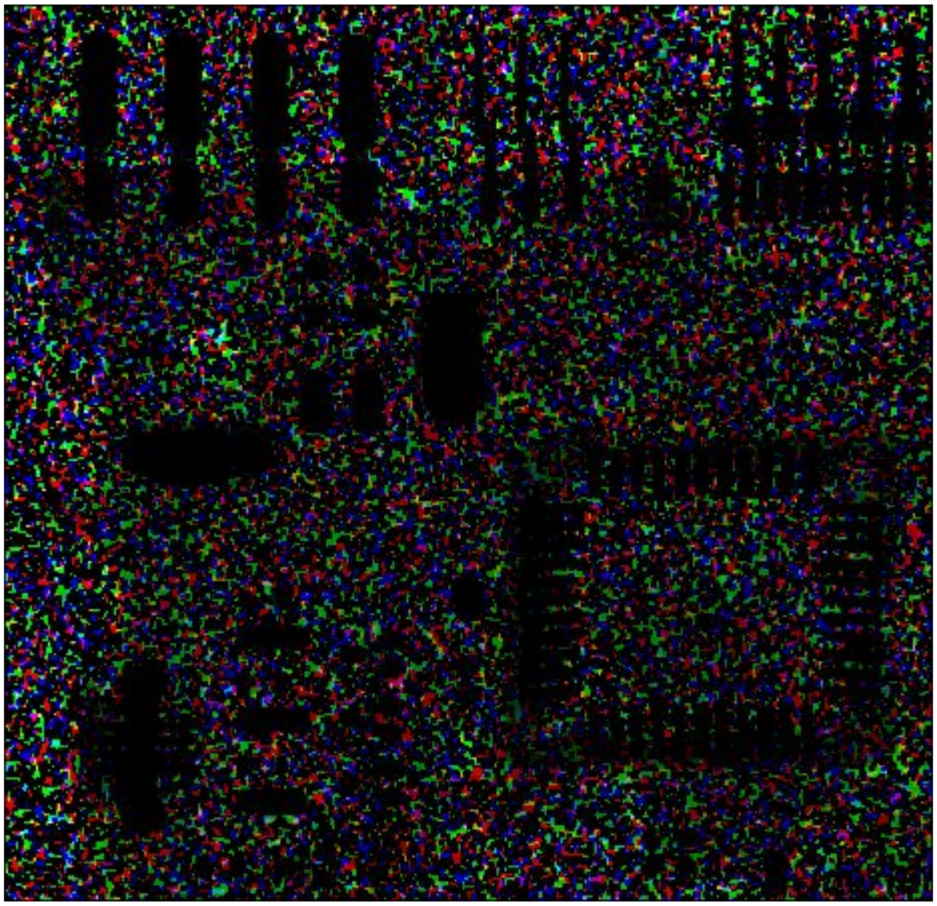
Geometric filter

Since the geometric filter can process white noise but it can't handle black noise, and the black noise contribute a lot to the image, thus the pepper noise will turn the image into black and geometric filter can't work well.



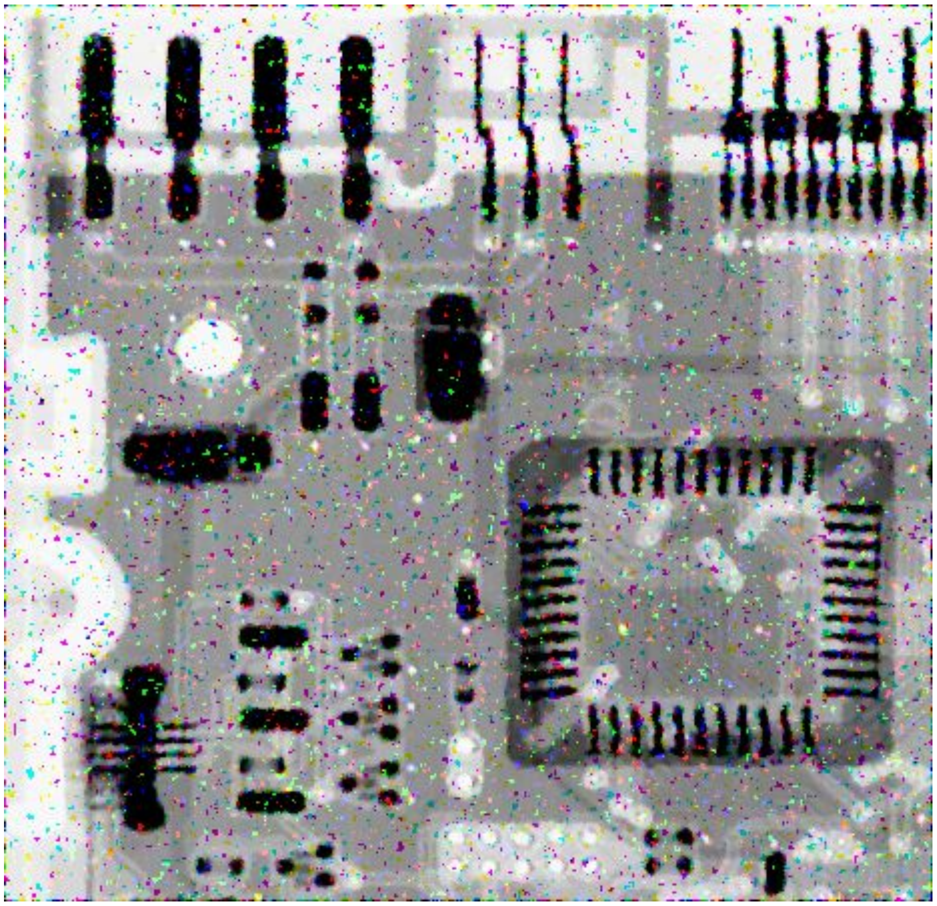
Max filter

The max can process black noise but it can't handle white noise, and the white noise contribute a lot to the image, thus the salt noise will turn the image into white and max filter can't work well.



min filter

The min filter can process white noise but it can't handle black noise, and once there is a pepper noise, the neighbor of it all become pepper noise, thus the pepper noise will turn the image into black and min filter can't work well.



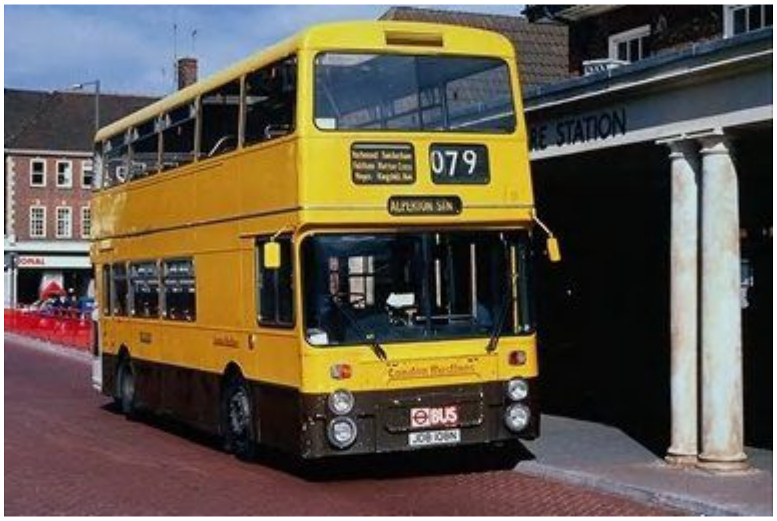
Median filter

The median filter work well compared with black filter and white noise filter, since it will remove salt and pepper noise. The median filter work well and will remove most of the noise.

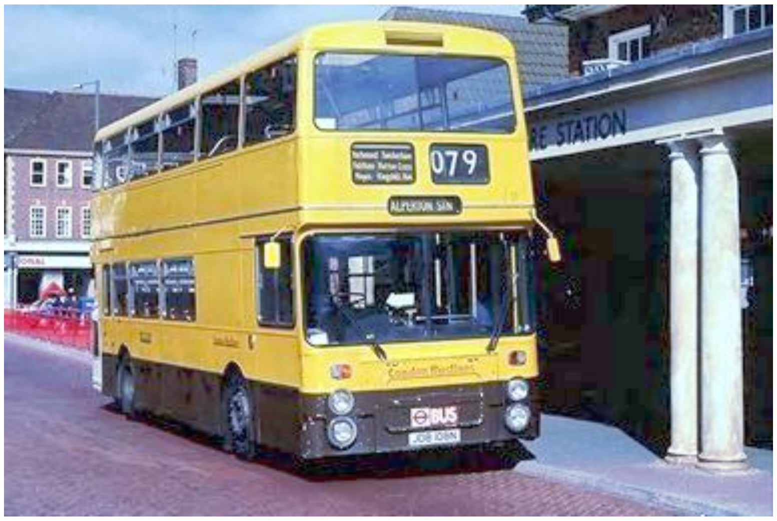
Above all, the median filter works best for salt-and-pepper noise.

2.4 Histogram Equalization on Color Images

1. Use the function \equalize hist" that you have written in HW2 to process the R, G, B channels separately. Rebuild an RGB image from these three processed channels and paste it in the report. (10 Points)



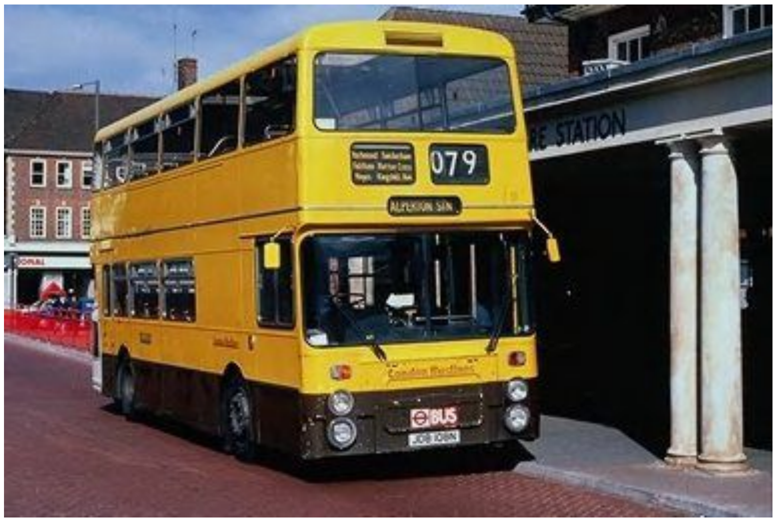
Origin image



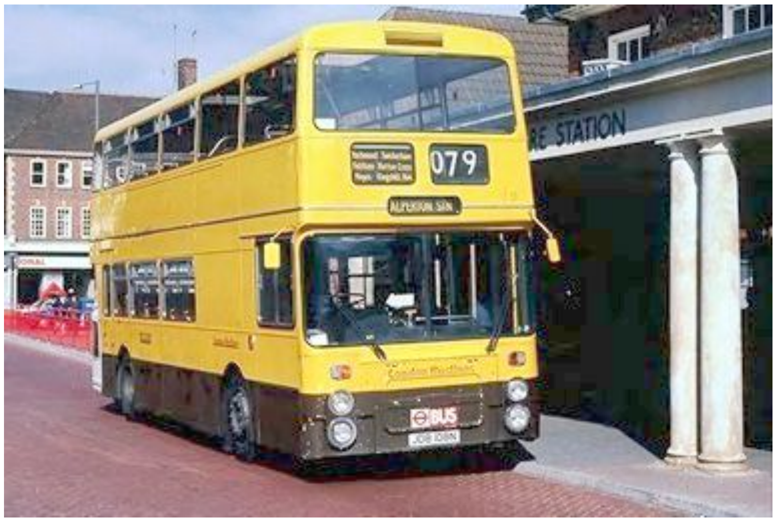
Histogram Equalization separately

2. Calculate the histogram on each channel separately, and then compute an average histogram from these three histograms. Use the average histogram as

the basis to obtain a single histogram equalization intensity transformation function. Apply this function to the R, G and B channels individually, and again rebuild an RGB image from the three processed channels. Paste the RGB image in the report. (10 Points)



Origin image



Histogram Equalization totally

3. Compare and explain the differences in the results produced by the above two applications within 1 page. (10 Points)

The two application are similar because although the histogram is not linear but it is nearly a linear transformation. Processing the R, G, B channels separately will make some of the pixels look strange, because the transformation for each color channel is not equal and the color may change to unknown color and the total of the image will become lighter since the origin image looks dark. Processing the R, G, B by a total histogram will equalize the image automatically and is suitable for processing the color image. It can remain the detail compared with other histogram equalization because all the R、G、B channels use the same transformation and the mapping is rational.