Write a program that removes noises from a picture.

Requirements:

- You can use any picture as long as it's two dimensional image.
- If a pixel has brightness that is "too different" from surrounding four pixels (up, bottom, right, left), let's assume the pixel has a "noise".
- If the brightness of the pixel is different by more than 10 points than the average brightness of the surrounding four pixels, let's assume that it is "too different".
- If a pixel has a noise, replace its original brightness with the average brightness of the surrounding four pixel.

Write a program that removes noises from a picture.

Requirements:

- You can use any picture as long as it's two dimensional image.
- If a pixel has brightness that is "too different" from surrounding four pixels (up, bottom, right, left), let's assume the pixel has a "noise".
- If the brightness of the pixel is different by more than 10 points than the average brightness of the surrounding four pixels, let's assume that it is "too different".
- If a pixel has a noise, replace its original brightness with the average brightness of the surrounding four pixel.

Image processing techniques would be beneficial for variety of applications including data extraction or identification/segmentation of an object in an image. Nonetheless, in practice, there is always some kind of noises involved in obtained images, and these noises make the use of image processing methods troublesome. Hence, Noise cancellation is one of the crucial area in image processing, and therefore there is plenty of algorithms available to use for this purpose. However, Noise cancelling methods should preserve the data in the image that is required for the application intended.

I wrote two different noise cancellation code with respects to the requirements that you have provided. The results are as depicted in the Figure 1-3. For this purpose, I have added salt and pepper noise to an image as can be seen in figure 1. Then, by applying the first algorithm which search in a 4-neighborhood (up, down, left, and right) and replace the center pixel with the average of the 4-neighborhood, figure 2 is obtained. Finally, the second algorithm, which replace 4 pixel in an 8-neighborhood with the median average of 5 pixels, is acquired and shown in figure 3.

Figure 1 Noisy Image

Figure 2 Noise Cancelation with respect to the first requirements

Figure 3 Noise Cancelation with respect to the second requirements

Pros and Cons of Code number 1:

This noise cancelation code is faster than the second one. In this code, the kernel matrix is implemented in a way that uses 4 pixels in the neighbors of image pixels and take the average of these four pixels. These few number of pixels will increase the speed of our processing as the computational cost is low. This technique could be suitable in real time detections as it provide faster processing. Besides, it is way better than a typical blurring filter (Low Pass Filter (LPF)) as the new algorithm preserves those area in which the noise is not presented.

Besides this sublime advantage, in my opinion, this code will cause propagation of noise in neighbor pixels. As can be seen in figure 3, although the brightness of (salt) noises will be compensated during noise cancellation, it will propagate the noise on the neighbor pixels in occasional circumstances.

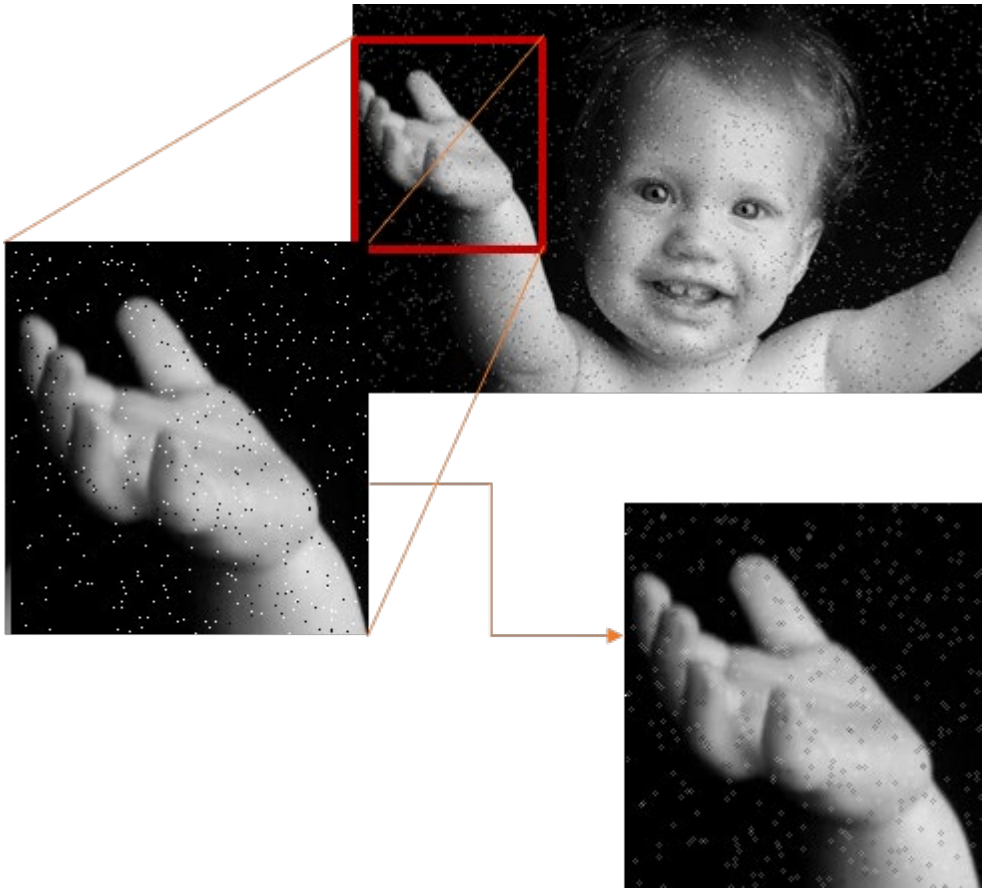

Figure 4 demonstration of propagation of the noise in the algorithm

To elaborate on this problem, Imagine a noise with brightness of 255 in a black background (each pixel has the value of 0) if the kernel matrix reach this noisy pixel it will change the center pixel's value to the average (which is 255/4). As an example, as can be seen in figure 4, when the kernel reach to a pixel that is located at top of the noisy one, because the brightness of the average ((0+0+255)/3) will be more than the predefined value (10), the average will be replace to that of the pixel .So, in this cases, not only does it solve the problem of noise presence, but it also makes a perfect pixel into a corrupted one. This process will be continuing for several times spreading these corrupted pixels. It would be a good idea to add some conditional limitation in which the algorithm only change the value if it is indeed a noisy pixel.

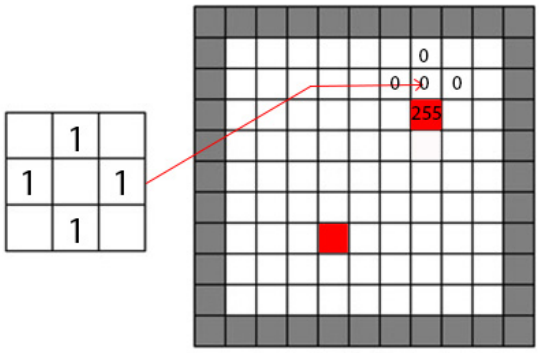

Figure 4, the schematic of the problem

Speckles noises are always present in the output profile of light interactions setups, are highly similar to salt and pepper noises as I added to my image (Figure 1). So, this code can be suitable for reducing the brightness of speckles in output profile of light.

Pros and Cons of Code number 2:

This code, as you can see in the Figure 3, is more powerful in reducing noises. The most discernible disadvantage of this code, as it can be seen in the output image, is the amount of blurriness which had been added to the original figure, which is more than previous code. Besides, this algorithm does not preserve edges very well which are crucial to image processing applications. In this code, Kernel matrix, which had been used for scanning all the pixels of the image, is a 3*3 matrix. By increasing the number of pixels and introducing several conditional statements, the computational cost will increase so the averaging process will take more time. Besides, the algorithm is similar to the low-pass filter and because of this, the image will be blurred.

By the way, I prefer the second method for noise cancelation. If you want to extract Moiré pattern from with high quality and without noise, this could be very beneficial way.