10. Noise modeling and digital image filtering

10.1. Introduction

Noise represents unwanted information which deteriorates image quality. Noise is defined as a process (n) which affects the acquired image (f) and is not part of the scene (initial signal – s). Using the additive noise model, this process can be written as:

$$f(i,j) = s(i,j) + n(i,j) \qquad (10.1)$$

Digital image noise may come from various sources. The acquisition process for digital images converts optical signals into electrical signals and then into digital signals and is one processes by which the noise is introduced in digital images. Each step in the conversion process experiences fluctuations, caused by natural phenomena, and each of these steps adds a random value to the resulting intensity of a given pixel.

10.2. Noise modeling

Noise (n) may be modeled either by a histogram or a probability density function which is superimposed on the probability density function of the original image (s). In the following, the models for the most common types of noise will be presented: salt and pepper noise and Gaussian noise. Other types of noise, such as negative exponential model, gamma/Erlang model, Rayleigh model are also presented in the literature (see the course notes).

10.2.1. The salt & pepper noise

In the salt & pepper noise model only two possible values are possible, a and b, and the probability of obtaining each of them is less then 0.1 (otherwise, the noise would vastly dominate the image). For an 8 bit/pixel image, the typical intensity value for pepper noise is close to 0 and for salt noise is close to 255.
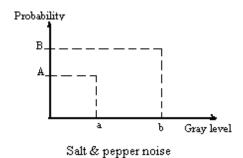

Fig. 10.1 Probability density function for the salt & pepper noise model.

$$PDF_{Sare\&piper} = \begin{cases} A & pentru\ g = a\ ("piper") \\ B & pentru\ g = b\ ("sare") \end{cases} \qquad (10.2)$$

The salt & pepper noise is generally caused by malfunctioning camera's sensor cells, by memory cell failure or by synchronization errors in the image digitizing or transmission.
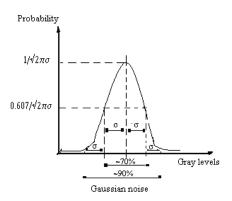
10.2.2. Gaussian noise

The Gaussian noise has a normal (Gaussian) probability density function:

Fig. 10.2 Probability density function for the Gaussian noise model

$$PDF_{Gaussian} = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(g-\mu)^2}{2\sigma^2}}$$ (10.3)

where:

g = gray level;

μ = mean;

σ = standard deviation;

Approximately 70% of the values are contained between $\mu \pm \sigma$ and 90% of the values are contained between $\mu \pm 2\sigma$. Although, theoretically speaking, the PDF is non-zero everywhere between -∞ and +∞, it is customary to consider the function 0 beyond $\mu \pm 3\sigma$.

Gaussian noise is useful for modeling natural processes which introduce noise (e.g. noise caused by the discrete nature of radiation and the conversion of the optical signal into an electrical one (detector/shot noise), the electrical noise during acquisition (sensor electrical signal amplification) etc).

10.3. Noise removal using spatial filters

10.3.1. Ordered filters (non-linear)

Ordered filters are based on a specific image statistic, called ordered statistic. They are called non-linear, because they cannot be applied as a linear operator (such as a convolution kernel). These filters operate on small windows, and replace the value of the central pixel (similarly to convolution). The ordered statistic is a technique which arranges all the pixels in sequential order, based on their gray-level value. The position of an element in this ordered set can be characterized by its rank. Given a NxN window W, the pixel values can be sorted in ascending order:

$$I_1 \leq I_2 \leq I_3 \leq \ldots \leq I_{N^2}$$ (10.4)

Where:

$\{ I_1, I_2, I_3, \ldots, I_{N^2} \}$ represent the intensity values of the pixels located within the NxN window W.

For example: given a 3x3 window:

$$\begin{bmatrix} 110 & 110 & 114 \\ 100 & 104 & 104 \\ 95 & 88 & 85 \end{bmatrix}$$

The result of applying the ordered statistic will be:

{85, 88, 95, 100, 104, 104, 110, 110, 114}

The median filter: selects the middle value from the ordered statistic and replaces the destination pixel with it. In the example above, the selected value would be 104. The median filter allows the elimination of *salt & pepper* noise.
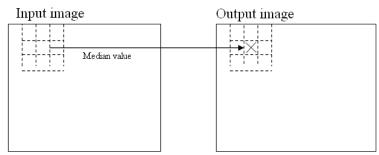

Fig. 10.3 Applying the median filter

The maximum filter: selects the largest value amongst the ordered values of pixels from the window. In the above example, the value selected is 114. This filter can be used to eliminate the *pepper noise*, but it amplifies the *salt* noise if applied to a salt & pepper noise image.

The minimum filter: selects the smallest value amongst the ordered values of pixels from the window. In the above example, the value selected is 85. This filter can be used to eliminate the salt *noise*, but it amplifies the *pepper* noise if applied to a salt & pepper noise image.

10.3.2. Linear filters

These filters are applied by convolution (a linear operation) with a low-pass filter convolution kernel. In the following, the computation of the elements of a convolution kernel for Gaussian noise elimination will be presented.

10.3.3. Designing a variable size Gaussian convolution kernel

Gaussian noise removal must be performed using a filter with adequate shape and size, correlated to the amount of the Gaussian noise that corrupts the image. The filter size w of such a filter is usually 6σ (for example, for a Gaussian noise with $\sigma=0.8 \Rightarrow w = 4.8 \approx 5$).
Constructing the elements of such a kernel/Gaussian filter will be performed using the following equations:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{\frac{-((x - x_0)^2 + (y - y_0)^2)}{2\sigma^2}}$$

(10.5)

Where:

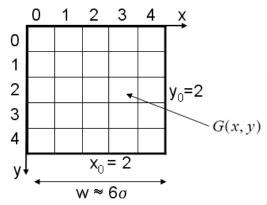 (x_0, y_0) – are the coordinates of the central row and column of the kernel (see Fig. 10.4).

Fig. 10.4 Design example of a Gaussian kernel/filter having a 5x5 size.

10.3.4. Image filtering/restoration

It is accomplished by the convolution of the source image with a Gaussian kernel/filter computed previously:

$$I_D(x, y) = G(x, y) * I_S(x, y) \qquad (10.6)$$

When the filter size w is large, the convolution may be time consuming (w x w multiplications for each pixel). In this case, the Gaussian decomposition may be used:

$$G(x,y)=G(x)G(y) \qquad (10.7)$$

and replacing the convolution of a 2D nucleus with 2 convolution of a 1D nucleus:

$$I_D(x, y) = (G(x)G(y)) * I_S(x, y) = G(x) * (G(y) * I_S(x, y)) \qquad (10.8)$$

Where:

   G(x) and G(y) are the central line and column vectors of the 2D nucleus (Fig. 10.5):

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-x_0)^2}{2\sigma^2}} \qquad (10.9)$$

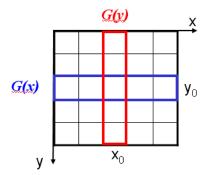$$G(y) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(y-y_0)^2}{2\sigma^2}} \qquad (10.10)$$

Fig. 10.5 The two vectors G(x) and G(y) into which a 2D Gaussian kernel may be separated

In this case, the number of multiplications needed for each pixel is w for each of the two convolutions.

10.4. Practical work

1. Implement a median filter with a variable dimension (w=3, 5 or 7) specified by the user.
2. Implement the filtering operation with a 2D Gaussian filter, with standard deviation σ and variable with w (w = 3, 5 or 7), specified by the user. The values of the kernel's components will be automatically computed as a function of σ, as in equation (10.5).
3. Implement Gaussian filtering by using a Gaussian kernel separated into 2 vector components $G(x)$ and $G(y)$ having a standard deviation of σ and a variable size w (w = 3, 5 or 7), specified by the user. The vector components values $G(x)$ and $G(y)$ will be computed automatically as a function of σ, as in equations (10.9) and (10.10).
4. **Save your work. Use the same application in the next laboratories. At the end of the image processing laboratory you should present your own application with the implemented algorithms.**

References

[1]. R.C.Gonzales, R.E.Woods, *Digital Image Processing. 2-nd Edition*, Prentice Hall, 2002.