Implementation of Gaussian mean filters for noise

**Md. Nazim Uddin**

***Abstract*—*Gaussian noise estimation and removal is proposed by using 3x3, 5x5, 7x7 and 9x9 sub windows in which the test pixel appears. The standard deviation (STD) for all sub-windows are used to define reference STD (σref) and minimum(σmin) and maximum (σmax) standard deviations. The algorithm estimates the amount of noise corruption, later the center pixel is replaced by the mean value of the some of the surrounding pixels based on a threshold value. Noise removing with edge preservation and computational complexity are two conflicting parameters. This method removes Gaussian noise and the edges are better preserved with less computational complexity***

Keywords— standard deviation, noise corruption, threshold value, Gaussian noise, edges.

1. Introduction

Noise having Gaussian-like distribution is very often encountered in acquired data. Gaussian noise is characterized by adding to each image pixel a value from a zero-mean Gaussian distribution. The zero-mean property of the distribution allows such noise to be removed by locally averaging pixel values. Conventional linear filters such as arithmetic mean filter and Gaussian filter smooth noises effectively but blur edges. The Gaussian smoothing operator is a 2-D http://homepages.inf.ed.ac.uk/rbf/HIPR2/mote.gifconvolution operator that is used to `blur' images and http://homepages.inf.ed.ac.uk/rbf/HIPR2/mote.gifremove detail and noise. In this sense, it is like the mean filter, but it uses a different kernel that represents the shape of a Gaussian (`bell-shaped') hump. This kernel has some special properties. Since the goal of the filtering action is to cancel noise while preserving the integrity of edge and detail information, nonlinear approaches generally provide more satisfactory results than linear techniques

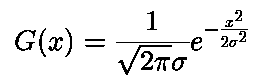
# 2.Gaussian Filter

Let ‘X’ is an original image, ‘A’ is observed image, and a general discrete time model for image degradation can be expressed as

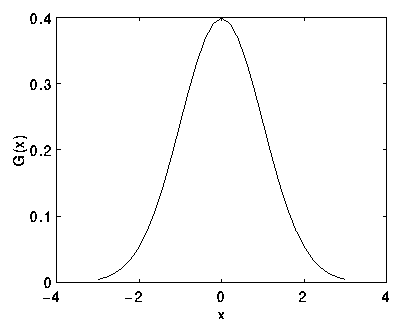
Ai, j = Xi, j + ηi, j

For i, j = 1,2…. N, where Xi, j is original image pixels, ηi, j is additive Gaussian noise and Ai, j is the observed image. The objective of the restoration scheme is to recover the original image from the observed image.

## The Gaussian distribution in 1-D has the form

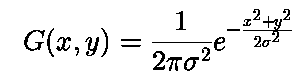


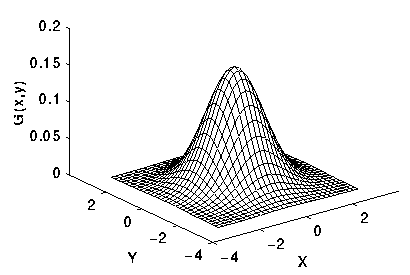
where **σ** is the standard deviation of the distribution. We have also assumed that the distribution has a mean of zero (*i.e.* it is centered on the line *x*=0)



*Fig: 1-D Gaussian distribution with mean 0 and* ***σ****=1*

## 2-D, an isotropic (i.e. circularly symmetric) Gaussian has the form





*Fig: 2-D Gaussian distribution with mean (0,0) and****σ****=1*

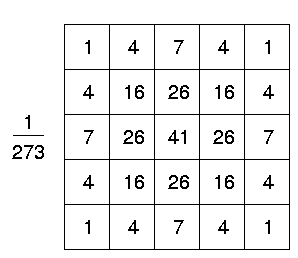
The idea of Gaussian smoothing is to use this 2-D distribution as a `point-spread' function, and this is achieved by convolution. Since the image is stored as a collection of discrete pixels, we need to produce a discrete approximation to the Gaussian function before we can perform the convolution. In theory, the Gaussian distribution is non-zero everywhere, which would require an infinitely large convolution kernel, but in practice it is effectively zero more than about three standard deviations from the mean, and so we can truncate the kernel at this point. We can use the value of the Gaussian at the center of a pixel in the mask, but this is not accurate because the value of the Gaussian varies non-linearly across the pixel. We integrated the value of the Gaussian over the whole pixel (by summing the Gaussian at 0.001 increments). The integrals are not integers: we rescaled the array so that the corners had the value 1. Finally, the 273 is the sum of all the values in the mask.

## 3.Remove Gaussian noise

If the pixel is found corrupted, then a filter is invoked. The corrupted pixel is replaced with a new value obtained from the following formula.

xnew (i, j) = [µ-0.5x σavg ]

Where xnew is the new value for the pixel position represented by (i, j), µ is the mean of the 3x3 central sub window, σavg is the average standard deviation defined in the previous section.



*Fig -3: Discrete approximation to Gaussian function with****σ****=1.0*

Once a suitable kernel has been calculated, then the Gaussian smoothing can be performed using standard convolution methods. The convolution can in fact be performed quickly since the equation for the 2-D isotropic Gaussian shown above is separable into *x* and *y* components. Thus the 2-D convolution can be performed by first convolving with a 1-D Gaussian in the *x* direction, and then convolving with another 1-D Gaussian in the *y* direction. (The Gaussian is in fact the *only* completely circularly symmetric operator which can be decomposed in such a way.) Figure 4 shows the 1-D *x* component kernel that would be used to produce the full kernel shown in Figure 3 (after scaling by 273, rounding and truncating one row of pixels around the boundary because they mostly have the value 0. This reduces the 7x7 matrix to the 5x5 shown above.). The *y* component is the same but is oriented vertically.

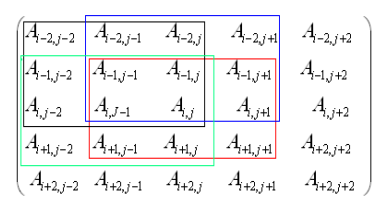
A further way to compute a Gaussian smoothing with a large standard deviation is to convolve an image several times with a smaller Gaussian. While this is computationally complex, it can have applicability if the processing is carried out using a hardware pipeline.

The Gaussian filter not only has utility in engineering applications. It is also attracting attention from computational biologists because it has been attributed with some amount of biological plausibility, *e.g.* Some cells in the visual pathways of the brain often have an approximately Gaussian response.

## 4.Algorithm

## Consider a 5 x 5 test window AT from the noisy image as:

Divide this window into 3 x 3 sub-windows such that the test pixel should appear in each of the sub-windows. Nine such sub-windows are possible and four of them as shown below.



1. For each 3x3 sub-window calculate the standard deviation, σi, i=1, 2 ………N where N is maximum number of the sub-windows, for this paper it is equal to 9.

2. Set reference standard deviation, (σref), as median of σi, i=1, 2 ………N.

3. Set σmin = k1x σref.

4. Set σmax = k2 x σref.

5. Calculate average (σavg) of the standard deviations σi, i=1, 2 ………N whose standard deviation lies in the range [σmin, σmax].

6. This σavg is used as a parameter to decide whether the test pixel is corrupted or not.

The above process is repeated by sliding 5x5 window one step forward row wise and then column wise to cover the entire image Both filters attenuate high frequencies more than low frequencies, but the mean filter exhibits oscillations in its frequency response. The Gaussian on the other hand shows no oscillations. In fact, the shape of the frequency response curve is itself (half a) Gaussian. So, by choosing an appropriately sized Gaussian filter we can be confident about what range of spatial frequencies are still present in the image after filtering, which is not the case of the mean filter. This has consequences for some edge detection techniques, as mentioned in the section on http://homepages.inf.ed.ac.uk/rbf/HIPR2/mote.gifzero crossings. (The Gaussian filter also turns out to be very like the optimal smoothing filter for edge detection under the criteria used to derive the Canny edge detector.

##### C:\Users\mdnaz\AppData\Local\Microsoft\Windows\INetCache\Content.Word\sta2noi2.gif

*Fig 4: Image is used to demonstrate Gaussian*

*filtering process*

##### which has been corrupted by 1% salt and pepper noise (*i.e.* individual bits have been flipped with probability 1%). The image became

##### http://homepages.inf.ed.ac.uk/rbf/HIPR2/images/sta2gsm1.gif

shows the result of Gaussian smoothing (using the same convolution as above). Compare this with the original

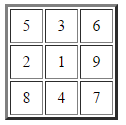


Notice that much of the noise still exists and that, although it has decreased in magnitude somewhat, it has been smeared out over a larger spatial region. Increasing the standard deviation continues to reduce/blur the intensity of the noise, but also attenuates high frequency detail (*e.g.* edges) significantly, as shown in pictures.

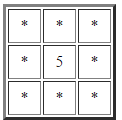
# 5.Fundamental theory of Mean Filter

The mean filter is a simple sliding-window spatial filter that replaces the center value in the window with the average (mean) of all the pixel values in the window. The window, or kernel, is usually square but can be any shape. An example of mean filtering of a single 3x3 window of values is shown below.

Unfiltered values



5 + 3 + 6 + 2 + 1 + 9 + 8 + 4 + 7 = 45  
45 / 9 = 5

Mean filtered

The idea of mean filtering is simply to replace each pixel value in an image with the mean ('average') value of its neighbors, including itself. This has the effect of eliminating pixel values which are unrepresentative of their surroundings. Mean filtering is usually thought of as a http://homepages.inf.ed.ac.uk/rbf/HIPR2/mote.gifconvolution filter. Like other convolutions it is based around a kernel, which represents the shape and size of the neighborhood to be sampled when calculating the mean. Often a 3×3 square kernel is used, as shown in Figure 1, although larger kernels (*e.g.* 5×5 squares) can be used for more severe smoothing. (Note that a small kernel can be applied more than once to produce a similar but not identical effect as a single pass with a large kernel.)

Image smoothing refers to any image-to-image transformation designed to smoothen or flatten an image by reducing the rapid pixel-to-pixel variation in grey levels. Smoothing may be accomplished by applying an averaging mask that computes a weighted sum of the pixel grey levels in a neighborhood and replaces the center pixel with that grey level. The image is blurred, and its brightness retained as the mask coefficients are all-positive and sum to one. The mean filter is one of the most basic smoothing filters. Mean filtering is usually thought of as a convolution operation as the mask is successively moved across the image until every pixel has been covered. Like other convolutions it is based around a kernel, which represents the shape and size of the neighborhood to be sampled when calculating the mean. Larger kernels are used when more severe smoothing is required. Fig. (a) shows a mean mask for a 3 × 3 window, while a more general n × n mask is shown in Fig. (b). Variations on the mean filter include threshold averaging, wherein smoothing is applied subject to the condition that the center pixel grey level is changed only if the difference between its original value and the average value is greater than a preset threshold. This causes the noise to be smoothed with a less blurring in image detail. It must be noted here that the smoothing operation is equivalent to low-pass filtering as it eliminates edges and regions of sudden grey level change by replacing the center pixel grey level by the neighborhood average. It effectively eliminates pixel grey levels that are unrepresentative of their surroundings. Noise, due to its spatial decorrelations, generally has a higher spatial frequency spectrum than the normal image components. Hence, a simple low-pass filter can be very effective in noise cleaning. Smoothing filters thus find extensive use in blurring and noise removal. Blurring is usually a preprocessing step bridging gaps in lines or curves, helping remove small unwanted detail before the extraction of relevant larger objects

6.Conclusion

The fast Gaussian and Mean filtering technique presented here has successfully reduced the time requirements for smoothing operations with mean filters, especially for large images. This implementation reduces the number of additions to approximately 1/nth of the original number, where n×n is the neighborhood size and completely eliminates the division operation by store and-fetch methods very efficiently. The gain in the performance and usefulness of this method has been demonstrated for different images.

The various applications of Gaussian and Mean filtering, as described in detail at the beginning of this paper can all benefit tremendously from this improved implementation. Noise removal by threshold averaging in remote-sensing applications is one such important example. The effect of this filtering technique will be tremendous in such an application. This method can easily be extended to higher bit-level grayscale images or color images.

##### 7.References

[1] https://www.cs.auckland.ac.nz/courses/compsci373s1c/PatricesLectures/Gaussian%20Filtering\_1up.pdf

[2] https://www.markschulze.net/java/meanmed.html

[3] http://www1.inf.tu-dresden.de/~ds24/lehre/bvme\_ss\_2013/ip\_03\_filter.pdf

[4] http://homepages.inf.ed.ac.uk/rbf/HIPR2/mean.htm

[5] http://www.cns.nyu.edu/pub/eero/simoncelli96c.pdf

[6] http://www.ripublication.com/irph/ijece/ijecev

5n1\_\_3.pdf

[7] https://pdfs.semanticscholar.org/df57/2b11b245267686aba59a564fd56f472cf845.pdf

[8] http://www.intelligence.tuc.gr/~petrakis/courses/computervision/filtering.pdf