Topic 6: Digital Filtering

6.1 Introduction

Digital filtering is the main tool in image processing, being used for a variety of processing applications such as edge enhancement and detection, noise reduction, data drop out removal and image restoration. The output of such filtering is some combination of the input image pixels, typically taken over a region, or neighbourhood. This operation in thus distinct from the point by point processing considered in the previous chapter, where the output pixel values depended on the value of the input image at a single point. As we will see this extension to a neighbourhood extends the flexibility of the processing and allows many more useful tasks to be performed.

In general we can split the filtering operations into two classes, these being *Linear* and *Non-Linear*. As suggested by the names, a the output from a *Linear* filter consists of a linear combination of the pixel values of the input image, for example sum, average, while the *Non-Linear* filter extends this to non-linear combinations, for example, minimum, maximum, median. As will be shown below, the linear filtering can be expressed as *convolution*, and then analysed using the Fourier techniques of previous chapters, while the non-linear filters have to be treated in real space, with the operation of the filter being highly dependent on the type of non-linear operation involved. We will consider both these schemes in this section.

6.2 Linear Digital Filtering

The linear filter takes a linear combination of the pixels of a region of the input image and forms a single output value for each location. This operation is therefore just the *convolution* of the input image with a *Filter Function* that contains the weights used in forming the linear combination of the pixels. This allows us to write the filtering operation as the familiar convolution operation of,

$$g(i,j) = h(i,j) \odot f(i,j)$$

where $f(i,j)$ is the input image, $h(i,j)$ is the filter function and $g(i,j)$ is the output or *Filtered image*. From this formulation the filter function can be seen to have a similar role to the Point Spread Function in incoherent imaging, and thus controls the operation performed by the filtering operation.

From the *Convolution Theorem* this operation can be performed in either **Real** or **Fourier** space. In either case the mathematical operations are identical, although, as discussed below, the computational cost and digital implementation method differs significantly.

6.2.1 Fourier Space Convolutions

To form a convolution of two functions by Fourier techniques it is required to form the Fourier transforms of both functions, multiply them together and then inverse Fourier transform: ie. the output image $g(i,j)$ is given by,

$$g(i,j) = \mathcal{F}^{-1}\{F(k,l)H(k,l)\}$$

where $F(k,l)$ and $H(k,l)$ are the Fourier transforms of $f(i,j)$ (the input image) and $h(i,j)$ (the filter function) respectively.

To implement this filtering operation it is required to form a minimum of two Discrete Fourier transforms of the image, and a complex multiplication[1]. Due to the dynamic range of the Fourier Transform it is required to perform this operation in floating point format, as discussed previously. Since the almost all the computational time is taken up performing the the Fourier transforms, the computational cost of this operation in not dependent on the filter function $h(i,j)$.

Since the Fourier filtering technique uses the Convolution theorem, it is limited to Linear Operations, and unlike the real space case, cannot be extended to non-linear operations.

6.2.2 Real Space Convolutions

As apposed to the Fourier transform implementation, the real space convolution is formed by direct application of the *shift and multiply* definition of a convolution. For a filter function of size M by M, the convolution is obtained by,

$$g(i,j) = \sum_{m=-M/2}^{M/2-1} \sum_{n=-M/2}^{M/2-1} h(m,n)\, f(i-m, j-n) \tag{1}$$

This operation is shown schematically in figure 1 This formulation removes the need for Fourier transforms, and since the input image has a small dynamic range, typically $0 \rightarrow 255$, then provided that the filter elements as small integers, the real space convolution can be performed in integer arithmetic. In this case, however the computational cost is directly related to the filter size, in fact for a 3×3 filter, there are 9 multiplies and 9 adds for each output pixel, while for a 5×5 filter there are 25 multiplies and 25 adds. So that the computational cost is proportional to the number of elements in the filter.

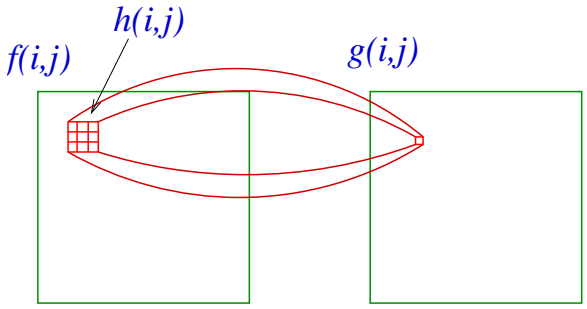


Figure 1: Real space filtering with a 3×3 filter.

Since the computational cost of the real space formulation rises with the size of the filter, while the Fourier space implementation is independent of the filter size, then for filters greater than a certain size the Fourier technique will be more efficient. Where this cross over occurs depends strongly on the computer hardware being, and in particular the relation between integer and floating point performance. For as a typical scalar Workstation this cross-over typically occurs for filters of 9×9 pixels, after which it is computationally preferable to perform the convolutions in Fourier space.

It should be noted that since the convolution is performed in real space, the linear summation operator, above, can easily be modified to a non-linear operation such as Minimum, Maximum or Median, which will be considered later in this chapter.

[1] If it is requires to form $H(k,l)$ from $h(i,j)$ a third DFT is required

6.3 Fourier Space Filters

The operation of Fourier filtering is determined by the filter shape $H(k,l)$, which modifies the Fourier transform of the input image. In most applications the input image is real and it is required that the output image is also real. It is known, that the complex Fourier transform of a real image obeys the familiar symmetry properties of *Real part symmetric* and *Imaginary part anti-symmetric*, shown in figure 2. Therefore if the output image is to be real, then the Fourier filter must not effect these symmetry relations so that the Fourier filter must also obey these symmetry relations.

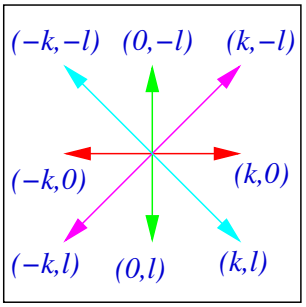


Figure 2: Symmetry of the Fourier transform.

6.3.1 Low-pass Filters

The operation of low pass filtering allow the low spatial frequencies to pass unattenuated while attenuating, or completely blocking, the higher spatial frequencies. This attenuation of high spatial frequencies can be used to reduce the effect of random noise mostly associated with high spatial frequencies as discussed in the previous section. This operation also results in the removal of some of the image information. This type of processing is used extensively in image segmentation and edge detection where the presence of noise corrupts the boundary information.

It should be noted that these filters are applied as a multiplication in Fourier space which results in a convolution in real space. So when designing a filter, the effect in both spaces has to be considered.

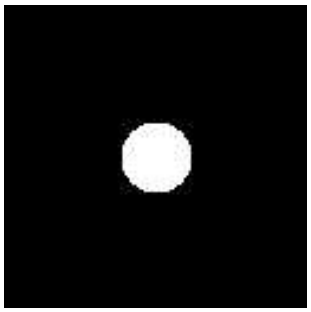
Ideal Low-Pass Filter

The simplest low-pass filter simply blocks all frequencies greater than some cut-off value, being specified as

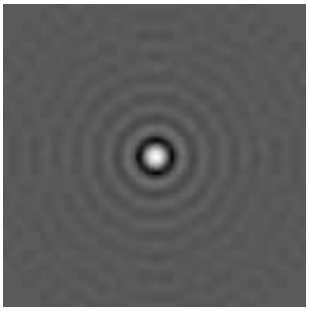
$$\begin{aligned} H(k,l) &= 1 \quad \text{for } k^2+l^2 \leq w_0^2 \\ &= 0 \quad \text{else} \end{aligned}$$

which is a simple *top-hat*. This filter blocks all spatial frequencies greater than w_0. The result of applying a low pass filter with radius 30 pixels to a 256 by 256 pixel image is shown in figure 3. In the filtered image the high frequency information, about the leaves is lost, as expended, but there is also significant ringing at sharp intensity boundaries.

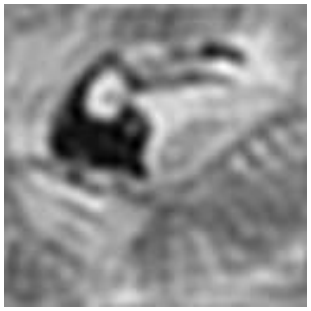
(a) Input image



(b) Low-pass filter



(c) Filter in real space.



(d) Filtered Image

Figure 3: Example of ideal low pass filtering.

This ringing results from the sharp of the filter function in real space, being the Fourier transforms of a top-hat. This gives,

$$h(i,j) = \frac{\mathrm{J}_1(r/w_0)}{r/w_0}$$

where $r^2 = i^2 + j^2$. which is shown plotted in figure 3 (c).

Smooth Cut-off Low-Pass Filter

This ringing seen in the *Ideal Filter* severely limits its usefulness, and to reduce this effect a range of smooth cut-off filters can be used. The most popular is a *Gaussian*, being given by,

$$H(k,l) = \exp\left(-\frac{w^2}{w_0^2}\right)$$

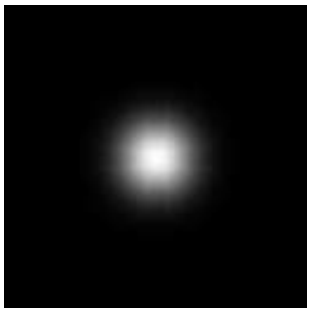
where $w^2 = k^2 + l^2$ and w_0 characterises the width of the filter since $H(k,l) = e^{-1}$ when $k^2 + l^2 = w_0^2$. The result of filtering with a w_0 30 pixels is shown in figure 4. As with the ideal filter, the image is smoothed, but in this case there is no edge ringing. In this case the real space filter response function, $h(i,j)$, is given by[2]

$$h(i,j) = \frac{\pi}{w_0^2}\exp\left(-\pi^2 w_0^2 r^2\right)$$

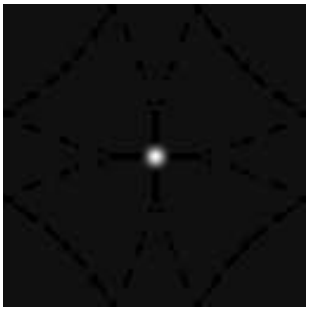
where $r^2 = i^2 + j^2$. is also a Gaussian which does not introduce any ringing, being asymptotic towards zero. This filter is infinite in both Fourier and real space, so attenuates rather than totally removing the high spatial frequencies.

[2]See Fourier transform booklet.

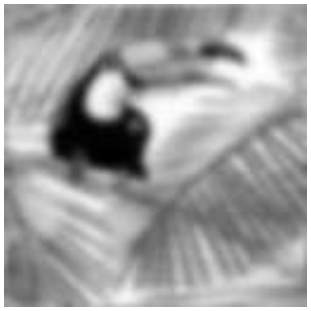
(a) Input image



(b) Low-pass filter



(c) Filter in real space.



(d) Filtered Image

Figure 4: Example of Gaussian low pass filtering.

A common approximation to the Gaussian filter is given by the *Butterworth* filter. This is given by,

$$H(k,l) = \frac{1}{1+(\frac{w}{w_0})^n}$$

where w_0 is the *half point* cut-off and n is the order of the filter, and is plotted in figure 5 for $w_0 = 15$ and orders $n = 2,4,6$. This filter has very similar properties to the ideal Gaussian, but being an approximation exhibits some real space ringing. This filter is computationally less expensive than the Gaussian. Visually it produces almost identical results to the Gaussian filter in figure 4[3]. This filter has been inherited from signal processing where it is possible to build a Butterworth filter from simple analogue electronic components.

An alternative, rather ah-hoc modification of the ideal low pass filter is given by the "*Trapezoidal*" filter, where

$$\begin{aligned} H(k,l) \quad &= 1 \quad && \text{for } w < w_0 \\ &= \tfrac{(w-w_1)}{(w_0-w_1)} \quad && \text{for } w_0 \le w \le w_1 \\ &= 0 \quad && \text{for } w > w_1 \end{aligned}$$

where w_0 is the start of the cut-off slope, and w_1 is the final cut-off of the filter. Spatial frequencies less than w_0 are passed unaltered, frequencies greater than w_1 are completely blocked and frequencies between w_0 and w_1 are attenuated. The result of this filter is similar to the other low-pass filters, where, subject to a sensible choice of the relative values of w_1 and w_0 the filtered image will typically exhibit less ringing than the ideal filter but more than the Gaussian of Butterworth filters.

[3]The difference are smaller that can be resolved on the printed page!.

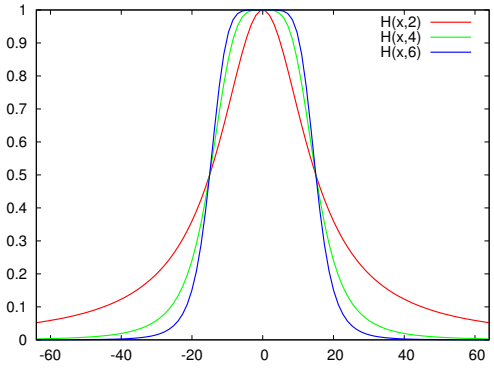


Figure 5: The Butterworth lowpass filter with $w_0 = 15$ for order $n = 2, 4, 6$.

6.3.2 High Pass Filters

As opposed to the Low-Pass filter, the high pass filter attenuates (or removes) low spatial frequencies while allowing high spatial frequencies to pass. These filters have the effect of enhancing the edges in images, which are associated with the high spatial frequencies, (since the rapid intensity variations in real space result in high spatial frequency components in Fourier space). It should be noted that much of the image noise is also associated with high spatial frequencies and therefore high pass filtering will enhance the effect of noise.

For the low-pass filters, considered above, the equivalent high-pass filter can be formed by "*inverting*" the filter profile, such that the high frequencies are allowed to pass and the low frequencies attenuated.

6.3.3 Ideal High-Pass Filter

The *ideal* high filter is simply given by,

$$\begin{aligned} H(k,l) &= 0 \qquad \text{for } k^2 + l^2 \leq w_0^2 \\ &= 1 \qquad \text{else} \end{aligned}$$

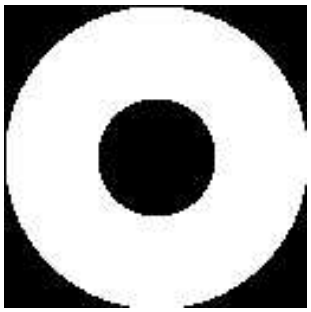
The shape of this filter with $w_0 = 25$ pixels, and it effect on the standard `toucan` image is shown in figure 6. This filter suffers from sever ringing at edges, typically to an extent that it is not possible to distinguish the true edge from the associated ringing artifacts. This filter is of little practical use and much better results can be obtained from smooth cut-off high-pass filters.

Smooth Cut-off High-Pass Filters

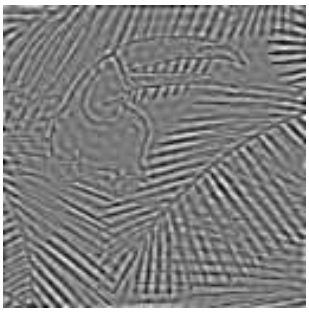
There is a range to smooth high pass filter which can be derived from the low pass versions discussed above, the most common being the Gaussian high pass given by

$$H(k,l) = 1 - \exp\left(-\frac{w}{w_0}\right)^2$$

This filter has the effect of severely reducing the low spatial frequencies while allowing the higher spatial frequencies to be passed, while the smooth transition minimised *ringing* This results in positive and negative sections containing the edges and other areas of rapidly changing intensity in an image.

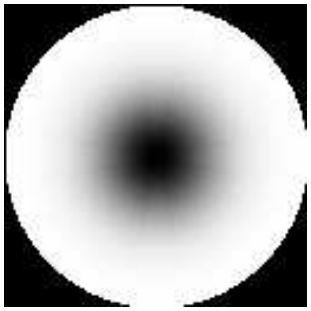


(a) High pass Filter



(b) Output Image

Figure 6: Shape of ideal high pass filter and it effect on the toucan image showing severe ringing.



(a) High pass Filter

(b) Output Image

Figure 7: Shape of Gaussian high pass filter and it effect on the toucan image.

As above, there is a highpass version of the Butterworth filter

$$H(k,l) = 1 - \frac{1}{1+\left(\frac{w}{w_0}\right)^n} = \frac{1}{1+\left(\frac{w_0}{w}\right)^n}$$

which is plotted in figure 8 for $w_0 = 15$ for orders $n = 2,4,6$, which has an almost identical effect to the Gaussian filter but, especially for the higher orders, gives a sharper transition about w_0.

6.3.4 Band Pass filters

A *low pass* can be combined with *high pass* filter to give a filter that passes a range, or band of spatial frequencies. Known as a *band pass* filter. Since the filtering occurs in Fourier space and the filter is implemented by a simple multiply, so if we want to apply two filters, $H_L(k,l)$ followed by $H_H(k,l)$, then the filtered Fourier transform is just

$$G(k,l,) = H_L(k,l)\,H_H(k,l)\,F(k,l)$$

As the order of multiplication has no effect, we can simply form a composite filter Fourier filter

$$H(k,l) = H_L(k,l)\,H_H(k,l)$$

We will consider this is greater detail when considering the *Difference of Gaussians* filter in a later section.

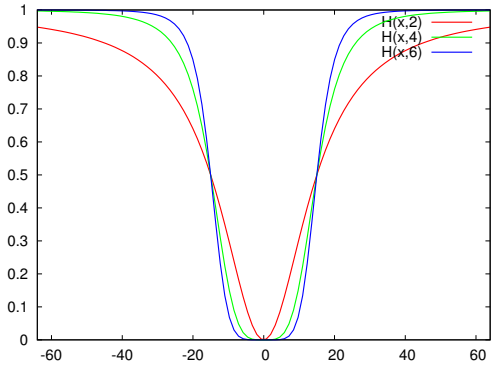


Figure 8: The Butterworth highpass filter with $w_0 = 15$ for order $n = 2, 4, 6$.

6.4 Real Space Filters

For real space filters the filtering mask $h(i, j)$ is specified over a finite range, typically 3×3, 5×5[4]. For windows greater than 7×7 pixels the processing time typically becomes extensive. The convolution is formed directly in real space using equation 1, where the filtering mask is of size $M \times M$, also shown schematically in figure 1. The various filtering operations are then performed by varying the mask elements.

Real Space Averaging

The averaging operation replaces each pixel by a local average taken over a neighbourhood. This is used to suppress the effect of image noise, being the real space equivalent to low-pass filtering. For a 3-by-3 window, we can consider a 5 pixel average formed by the mask

$$\begin{matrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{matrix}$$

which form a window with an effective (average) radius of one pixel, or alternatively we can form a 9 pixel average from the the mask

$$\begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

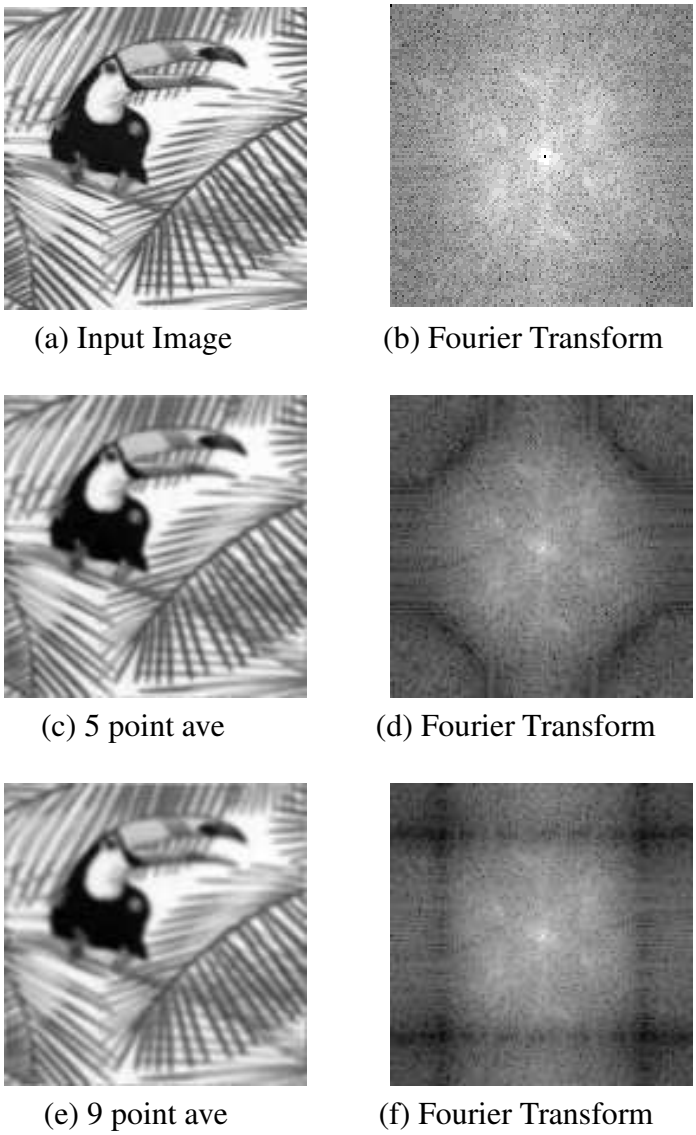
which has an effective radius of $\sqrt{2}$ pixels. In the case of the 9 pixel average, it should be noted that it is approximately equivalent[5] to multiplication in Fourier space with a functions

$$H(k,l) = \text{sinc}(Nk/3)\,\text{sinc}(Nl/3)$$

where the image size is $N \times N$. The effect of real space averaging is shown in figure 9. The processed image look *smoother* with some edge blurring as would be expected. The effect is Fourier space is very obvious, especially for the 9-pixel average where, due to the convolution theorem, the Fourier transform has been multiplied by a two-dimensional sinc() function. The displayed Fourier transform is actually the modulus squared, so the vertical and horizontal line zeros of the sinc()^2 are clearly visible.

[4]The filter mask is almost always odd in size since even masks result is a shift in image information.

[5]See tutorial question 6.3 for the full expression.



(a) Input Image (b) Fourier Transform

(c) 5 point ave (d) Fourier Transform

(e) 9 point ave (f) Fourier Transform

Figure 9: Effect of real space averaging.

The extent of the averaging, and thus smoothing, is controlled by the window size. It should be noted that although the real space image appears effectively smoothed, it should be remembered that the Fourier transform of the image has been multiplied by a sinc term. This is not normally a problem for computer vision and segmentation applications where all processing is in real space, but must be considered with great care if the filtered image is subsequently processed by Fourier techniques (eg. in tomographic imaging).

6.4.1 Real Space Differentiation

Much of real space filtering is associated with edge detection which is performed by differential operations of various types. Differentiation is most easily seen in one dimension: the extension to two dimensional images is then obvious. If we consider a continuous one dimensional function $f(x)$, then the first differential is given by,

$$\frac{\mathrm{d}\, f(x)}{\mathrm{d}\, x} = \lim_{\delta \to 0} \frac{f(x+\delta) - f(x)}{\delta}$$

Now in the discrete case, where we have a sampled function $f(i)$, then $\delta = 1$ and the differential becomes

$$\frac{\mathrm{d}\, f(i)}{\mathrm{d}\, i} = f(i+1) - f(i)$$

This is equivalent to convolving the one dimensional discrete signal with a window of size 2 pixels given by

$$[-1 \quad 1]$$

If, however, we take $\delta = 2$, we obtain the similar result that

$$\frac{\mathrm{d}\, f(i)}{\mathrm{d}\, i} = f(i+1) - f(i-1)$$

which can be implemented by the convolution with a filter mask of 3 pixels in length given by

$$[-1 \quad 0 \quad 1]$$

as shown in figure 10.

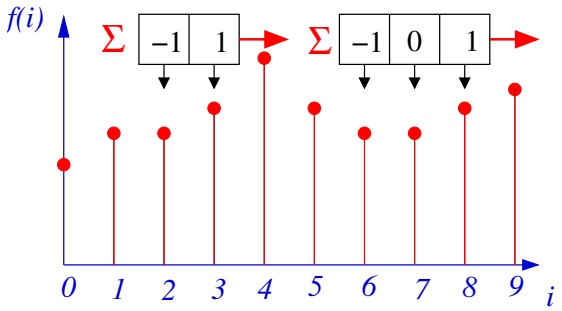


Figure 10: Differential of a sampled signal.

Similarly the second differential of a function, $f(x)$, is given by,

$$\frac{\mathrm{d}^2 f(x)}{\mathrm{d}\, x^2} = \lim_{\delta \to 0} \frac{f(x+\delta) - 2f(x) + f(x-\delta)}{\delta}$$

which in the discrete case becomes,

$$\frac{\mathrm{d}^2 f(i)}{\mathrm{d}\, i^2} = f(i+1) - 2f(i) + f(i-1)$$

This it now equivalent to convolving with a filter mask of 3 pixels in length given by,

$$[1 \quad -2 \quad 1]$$

as shown in figure 11.

Two Dimensional Differential Filters

For the two dimensional case we can form differentiation with respect to the i or j direction by convolution with the above filters suitably rotated for example

$$\frac{\partial f(i,j)}{\partial i} = [-1 \quad 0 \quad 1] \odot f(i,j) \quad \text{and} \quad \frac{\partial f(i,j)}{\partial j} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \odot f(i,j)$$

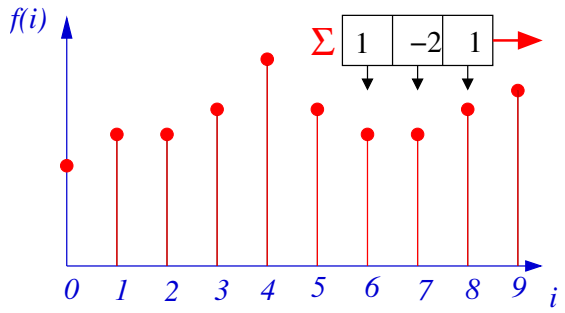


Figure 11: Second order differential of a sampled signal.

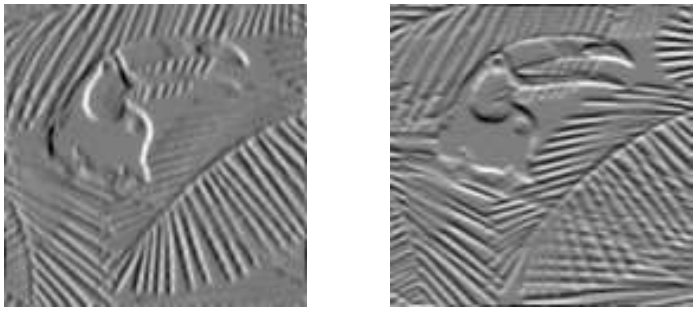
In both cases, the filter is typically described as a 3×3 filter[6] where the one dimensional filter is repeated over three adjacent rows or columns respectively. This results in averaging over the three rows and thus reduces the effect of noise; therefore the first partial derivatives can be written as

$$\frac{\partial f(i,j)}{\partial i} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \odot f(i,j)$$

and

$$\frac{\partial f(i,j)}{\partial j} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \odot f(i,j)$$

The effect of applying these filters is shown in figure 12. The X-differential has the effect on enhancing *vertical* edges, while the Y-differential enhanced *horizontal* edges, with these edges appearing as large *positive* or *negative* values[7].



(a) X differential (b) Y differential

Figure 12: Effect first order real space differentials.

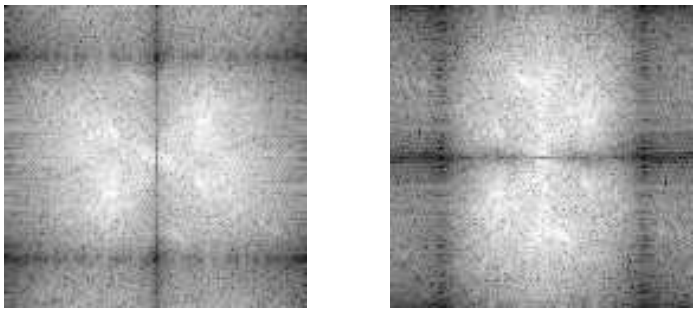
In Fourier space we have that the the Fourier transform of the first-order differentials are given by

$$\mathcal{F}\left\{\frac{\partial f(x,y)}{\partial x}\right\} = \imath 2\pi u F(u,v) \quad \text{and} \quad \mathcal{F}\left\{\frac{\partial f(x,y)}{\partial y}\right\} = \imath 2\pi v F(u,v)$$

[6]Technically know as the Prewitt filter.

[7]The images are displayed with the most *negative* pixel displayed at black and the most *positive* as white, so 0 corresponds to *mid grey*.

so that the first order differential is equivalent to Fourier space multiplication by $\imath 2\pi u/v$. This can be seen from the Fourier transforms of the first order differentials shown in figure 13. The X-differential shows a sharp vertical zero through the origin while the Y-differential shows horizontal zero. The broader lines of zeros displaced from the centre are due to the averaging over the three adjacent lines which is equivalent to convolving with a sinc() function as in the 3×3 averaging shown in figure 9 (f). The example again shown the relation between real and Fourier space, and that applying what appears to be a very simple 3×3 filter in real space can have a rather complex effect in Fourier space.



(a) X differential (b) Y differential

Figure 13: Fourier transform of the first order differentials.

The second partial derivatives are similarly given as,

$$\frac{\partial^2 f(i,j)}{\partial i^2} = [\,1 \quad -2 \quad 1\,] \odot f(i,j) \quad \text{and} \quad \frac{\partial^2 f(i,j)}{\partial j^2} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \odot f(i,j)$$

Now noting that convolution is a linear operation, then the addition of the second partial derivatives can be formed by the addition of the two filters prior to convolution, so that,

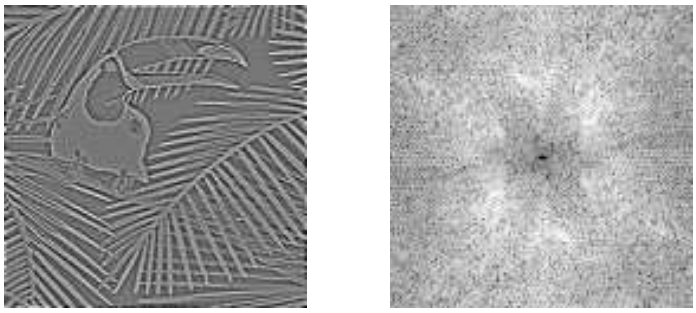
$$\nabla^2 f(i,j) = \frac{\partial^2 f(i,j)}{\partial i^2} + \frac{\partial^2 f(i,j)}{\partial i^2} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \odot f(i,j)$$

which is the Laplacian of the image. Additionally in Fourier space, we have from *Fourier Transform* booklet equation (17) that:

$$\mathcal{F}\left\{\nabla^2 f(x,y)\right\} = -(2\pi w)^2 F(u,v)$$

where $w^2 = u^2 + v^2$, so that in Fourier space taking the laplacian is equivalent to multiplication by a quadratic. The real space and Fourier space images are shown in figure 14. Note that in Fourier space, the laplacian is a *high pass* filter which enhances the high spatial frequencies which contains the edges, but, as seen in the last section, also where noise has the greatest effect.

The shape of first and second order derivatives of an edge are shown in figure 15. For a *positive* edge the first order differential is a positive peak while the second order differential is a positive peak followed by a negative peak with the edge located at the *zero crossing* of the second order differential. These two properties will form the basis of edge detection in subsequent sections.



(a)Laplacian (b)Fourier Transform

Figure 14: Laplacian in real and Fourier space.

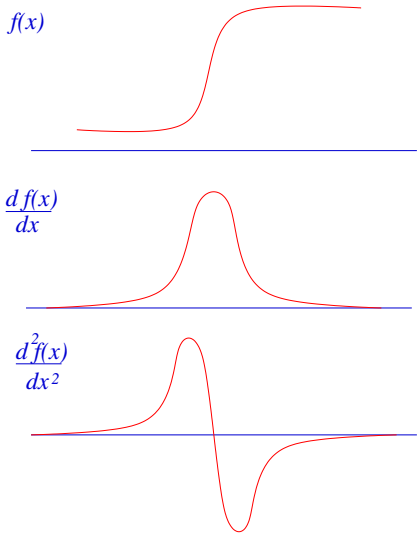


Figure 15: Shape of first and second order derivatives of positive edge.

Importantly the second order differential is independent of direction, so the same operation in applied to edges in all directions.

As can be seen graphically, in figure 16 the edges may be enhanced by the subtraction of the second differential from the original signal giving a *dip* before the edge and a *peak* after it. It also make the edge gradient steeper, and so more visible. In two dimensions this involves the subtraction of the Laplacian. This can be formed by a single filtering operation given by,

$$f(i,j) - \nabla^2 f(i,j) = \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \right) \odot f(i,j)$$

$$= \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \odot f(i,j)$$

which is frequently described as an *edge sharpening* filter found in many image processing packages. The effect of this is shown in figure 17 and shown clear sharpening of the edges of the image.

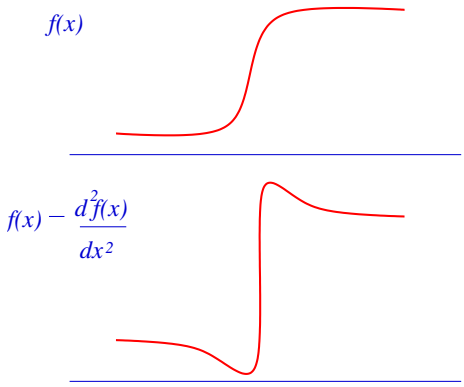


Figure 16: Enhancement of an edge by subtraction of the second order differential.

(a)Original (b)Edge Enhanced

Figure 17: Effect of laplacian edge enhancement.

6.5 Uses of Linear Filters

Low-pass filters, formulated in both real and Fourier space are used to reduce the effect of noise that is uncorrelated with the image. All these filters either attenuate or completely block the high spatial frequency components, therefore some image information is also lost, resulting in a blurred image. All filters, except the Gaussian filter will produce some types of artifacts, with, typically the artifacts occurring in the opposite space to which the filter is applied in. These filters are used frequently in many image processing applications, especially involving noisy data, where only general shape of an object is required.

High-pass filters, which include the differential filters, are used to enhance the high spatial frequencies, and thus regions of rapid variation in image brightness. These filters therefore enhance edges. This enhancement also has the effect of enhancement of noise, which is also associated with the high frequency components and cannot be separated from the image information.

These two types of filters are frequently used in combination, ie. an image may be smoothed with a low-pass filter to reduce the effect of noise, and the result processed with a high-pass filter to enhance the edges. Since this filtering operation is linear, then the filters may be combined to form a composite filter that will both smooth the image **and** enhance the edges. Such a filtering technique is known as *band-pass* filtering.

In *real* space filtering the resultant filter is formed by convolving a differential filter with an averaging filter. This results in a larger filter, ie. the convolution of two 3-by-3 filters results in

a 5-by-5 filter. The effect of a 3-by-3 low-pass and a Laplacian can be written as,

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \odot \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & -2 & -1 & -2 & 1 \\ 1 & -1 & 0 & -1 & 1 \\ 1 & -2 & -1 & -2 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

which will produce a smoothed Laplacian version of the image.

6.6 Real Space Non-Linear Filtering

In real space convolution filtering is defined by the *shift and multiply* operation, giving an output image $g(i,j)$ as a convolution of an input image $f(i,j)$ and a finite filtering function, $h(i,j)$. The modification to non-linear filtering involves the replacement of the summations by the specified non-linear operator to give,

$$g(i,j) = \mathrm{O}_{m,n \in w}[h(m,n)\, f(i-m, j-n)]$$

where the range of $h(i,j)$ is defined by w. The characteristics of the filtering operations are now determined by both the *filter mask* $h(i,j)$ and the operator $O[]$ as shown in figure 18. In many applications the elements of the filter mask are set to unity and the filtering operations is controlled by the characteristics of the operator. A range of such modifications will be considered in the following sections.

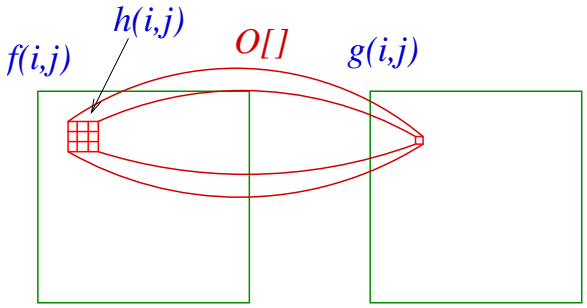


Figure 18: Non-linear filtering in real space.

6.7 Shrink and Expand Filters

If we consider the case where the filter mask elements are set to unity and the operator is *min* or *max*, so that the output image contains the minimum or maximum of the pixel values of the original image from a window. This filter will then act as *Shrink* and *Expand* operator, respectively.

Shrink Filter

By taking the *min* operation across a filter mask all objects will effectively be *reduced* in size by the size of the filter mask, with isolated objects of size less than the filter size being completely

removed. This is shown for the 1-D case in figure 19, where a *min* operation over a window of size 3-pixels is taken. In this case features of sizes less than 3 pixels are completely removed and large objects are reduced in size by 2-pixels.

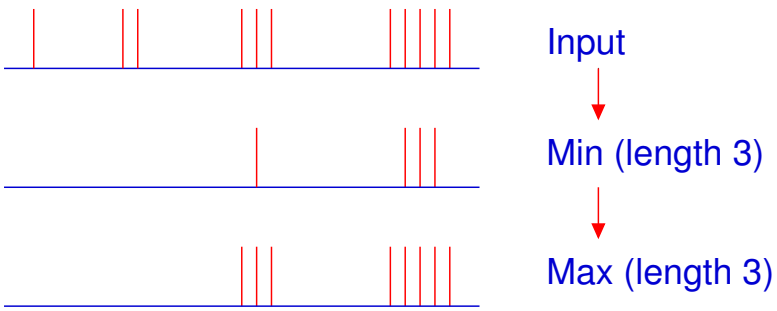


Figure 19: Effect of Shrink and Expand Filtering operations in one dimension

Expand Filter

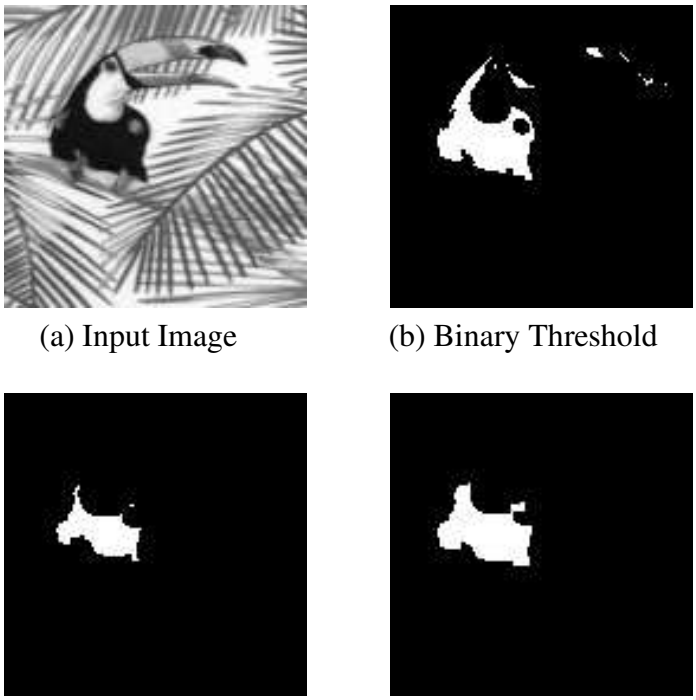
If alternatively the *max* operator is taken then all objects will be effectively *expanded* by the size of the filter mask independent of their size. In practice this filter is of very little use in isolation, but is typically applied to an image that has been previously filtered by a *Shrink* filter. This *Expand* filter reverses the effect of the shrinking effect of large objects as shown in dimension in figure 19. The combined effect of these two filters is to remove all *bright regions* smaller than the filter size while leaving large regions *almost* unaltered.

These filters are most commonly used on binary images, where their operation is obvious, (ie. small, isolated features are removed while large objects are retained without smoothing of edges) as shown figure 20. On grey level images as similar smoothing effect is obtained, however there is no simple analysis. These filters are typically less computationally expensive to perform than linear operations since in most computer hardware the *min* or *max* operations are formed by conditional testing which is faster than arithmetic operations.

6.7.1 Threshold Average Filter

Consider an image corrupted with random noise, particularly in the form of random *data drop-outs*, which appear as random corrupted pixels where these values are completely uncorrelated with the image. Such corruption is common in video images transmitted over a noisy data link giving the *snow* effect on images. This type of image corruption can be dealt with by forming local average about, but not including, the central pixel. This average is then compared with the central pixel value, and if it differs by more than a preset threshold, then the central pixel is considered to be *corrupted* and is replaced by the local average. This is implemented in two stages, by first forming,

$$A = \sum_{m=-M/2}^{M/2-1} \sum_{n=-M/2}^{M/2-1} h(m,n)\, f(i-m, j-n)$$



(a) Input Image (b) Binary Threshold

(c) Binary shrink (d) Binary expand

Figure 20: Use of the binary shrink and expand operators on a thresholded image.

where for the case of $M = 3$ we have that,

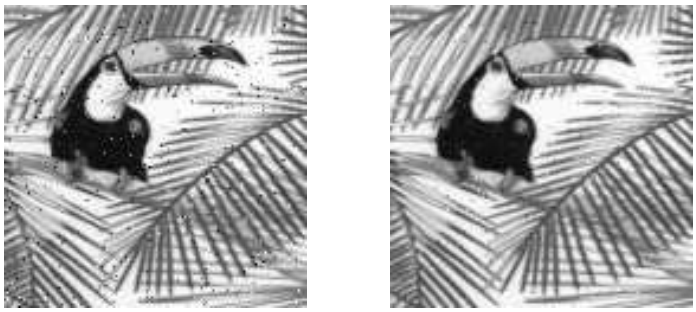
$$h(i,j) = \begin{bmatrix} k & k & k \\ k & 0 & k \\ k & k & k \end{bmatrix}$$

where $k = 1/M^2$, and then forming the output of

$$\begin{aligned} g(i,j) &= A \qquad \text{for } |A - f(i,j)| > T \\ &= f(i,j) \qquad \text{else} \end{aligned}$$

This filter selectively removes pixels that deviate significantly from their surrounding neighbours, while leaving the rest of the image unaltered. This filter is particularly successful in the removal of the *snow* effect where the corruption consists of isolated pixels set to either the minimum or maximum pixels values, (0 or 255 in a 8-bit system). The choice of the threshold T is dependent on the type of image noise, if the threshold is too large then some noise points will be missed and if too small the image will be needlessly smoothed.

Consider an example of the usual 128×128 `toucan` image with $1:50$ bits artificially corrupted as shown in figure 21 (a). The most serious corruption will occur when the most significant bit is corrupted, which for a 128×128 is expected to occur at approximately 325 pixels. The number of pixels classified as *noise* and corrected against threshold value is plotted in figure 22 which suggested that the optimal threshold is approximately 66, being about $1/4\, f_{\max}$. The resultant corrected image is shown in figure 21 (b) which shows that *most* of the corrupted pixels have been removed but without significant smoothing of the image. This threshold is optimal for this particular image, but is typical of the optimal value for high contrast images.



(a) Bit corrupted (b) Filtered

Figure 21: (a) Bit corrupted image with $1:50$ bits corrupted, (b) Average threshold processed image with threshold of $T = 66$.

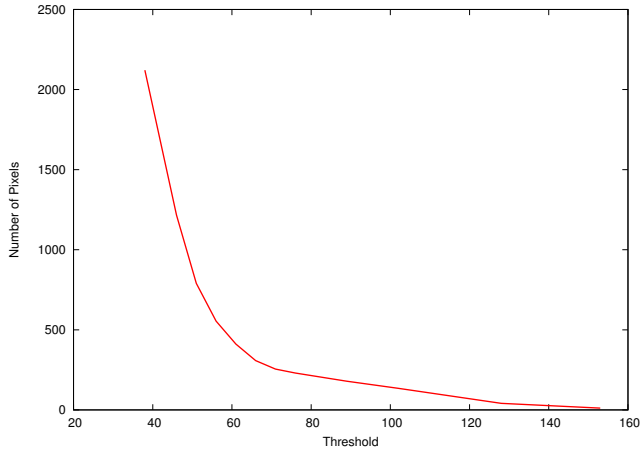


Figure 22: Graph of number of corrected pixels against threshold for the images in figure 21

6.8 Median Filter

The most important real space non-linear filter is the *Median* filter where the summation operation of the linear convolution is replaced by the *Median* operation, to give

$$g(i,j) = \text{Median}_{m,n\in w}[h(m,n)\, f(i-m, j-n)$$

This filter is typically considered with $h(i,j) = 1$, where the effects of the filter are easily analysed. If we have a set of N sample values, $(f(i) : i = 0,\ldots,N-1)$ then the *Median* of this set is defined by,

$$\begin{aligned} \text{Median}[f(i)] \quad = \quad & f(j) \\ & N/2 \text{ members of } f(i) < f(j) \\ & N/2 \text{ members of } f(i) > f(j) \end{aligned}$$

where N *must* be *odd*. This definition is equivalent to saying that the *Median* is the *middle* value. It should be noted that the *Median* is always one of the input values. For example if we have $N = 5$ and

$$f(i) = 61, 10, 9, 11, 9$$

then the *Median* is 10 since there are two values greater than 3 and two values less than 3, note this has the effect or ignoring the large, possibly corrupted, value. The effect of this filter

is most shown in one-dimensions in figure 23 for a *median* over a one-dimensional mask of 5 pixels compared to the *average* over isolated features of varying sizes. The result is that the *median* filter of length M will remove small isolated features of less that $M/2+1$, while retaining larger features. Alternatively the *average* smooths out the small features and also *blurs* the edges of the larger features but does not remove them. The filter can thus be *tuned* to filter out objects of different sizes, while retaining larger objects.

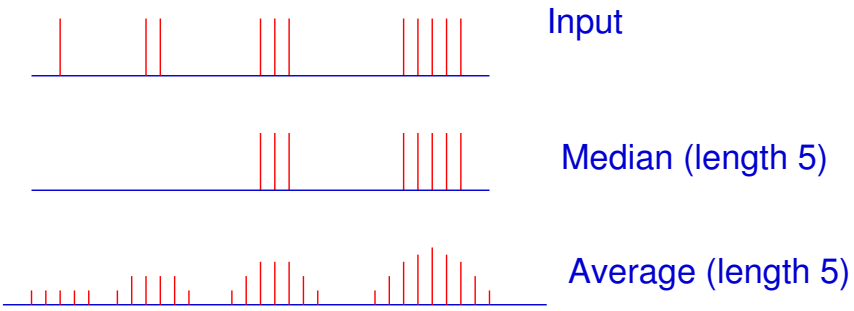


Figure 23: Comparison of Average and Median Filtering in one dimension

The main power of the *median* is when applied to edges as shown in figure 24 where the edge is perfectly preserved by the *median* while an *average* of 5-pixels will effectively smooth the edge out over 5-pixels. The *median* is therefore a smoothing, or *low pass* filter that also preserves edged which solves the major problem encountered in other linear low pass filters.

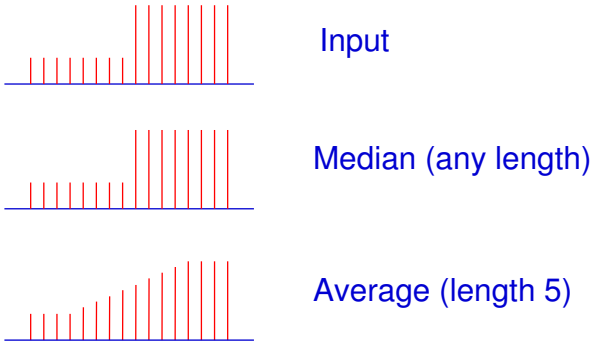
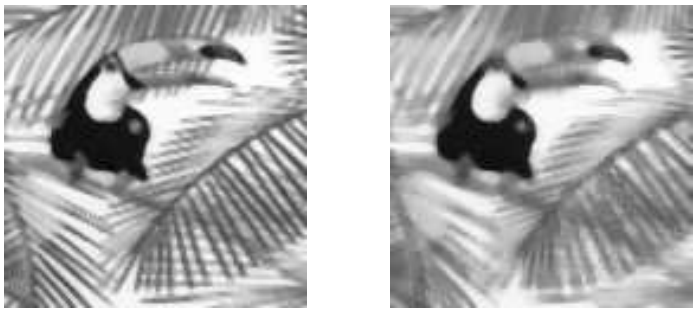


Figure 24: Comparison of Average and Median Filtering at an edge.

The filter operates similarly in two dimensions by removing all objects of size less than $\frac{M^2}{2}-1$ pixels while retaining larger objects and edges. This filter is one of the most widely used image processing filters for noise reduction since it smooths images without, seriously effecting the image edges. The effect of 3×3 and 5×5 median filters on the `toucan` image are shown in figure 25. This shown the smoothing effect with, especially for the 5×5 median the image is blocks of almost constant intensity, but still with sharp edges.

To implement this filter the local medians must be formed for each pixel, which for a 3×3 filter requires N^2 medians of 9 points to be formed. To calculate a median the data has to be sorted into order (actually partially sorted), which is a computationally very expensive algorithm. For example using *Quick-sort* (the fastest sorting algorithm), a 5×5 median filter is about the same computational cost as a two-dimensional Fourier transform of the same size of image. The



(a) 3×3 median (b) 5×5 median

Figure 25: (a) Result of 3×3 median filter, (b) Result of 5×5 median filter.

computational cost then rises as the number of image pixels **and** as $M^2 \log(M)$ where M is the size of the window.

The *Median* filter has a low-pass filtering effect on an image, thus reducing its information content. For a typical image, this smoothing effect is equivalent to a real space averaging filter of about half the size, while it is more effective at suppressing random noise, see workshop question 6.5 for its effect in Fourier space.

6.9 Homomorphic Filtering

While Fourier space filtering is an inherently linear process, the image space may undergo a non-linear operation prior to processing. If we consider an image model of an incident intensity distribution, denoted by $i(x,y)$, falling on an object with reflectance $r(x,y)$, then the received image will be a multiplication of the incident intensity distribution and the reflectance, giving

$$f(x,y) = i(x,y)\, r(x,y)$$

This model allows for variation in lighting of a scene, and in particular if a three-dimensional scene is lit by a single point source as shown in figure 26, then the illumination intensity will obey the inverse square law, with object further from the source being lit less brightly. In many applications we would wish to remove the effect of this lighting variation which does not convey useful information about the image.

In this case we can assume that the lighting variation is smooth, ie. it contains only low spatial frequencies, while the reflection term contain mostly high spatial frequencies, such as intensity edges which we wish to enhance. In real space we have a multiplication, so a simple Fourier transform will give a convolution, and therefore no improvement in separation of the two terms. However we can take ln[] to form

$$z(x,y) = \ln(f(x,y)) = \ln(i(x,y)) + \ln(r(x,y))$$

which we can then Fourier transform to give,

$$Z(u,v) = \mathcal{F}\{\ln(i(x,y))\} + \mathcal{F}\{\ln(r(x,y))\}$$

where $Z(u,v)$ is known as the *Cepstrum* of the original image $f(x,y)$. Now if we have a smooth varying function, $i(x,y)$, then the $\ln(i(x,y))$ will also be a smooth varying function, and similarly $\ln(r(x,y))$ will be a rapidly varying function if $r(x,y)$ is a rapidly varying function. So the

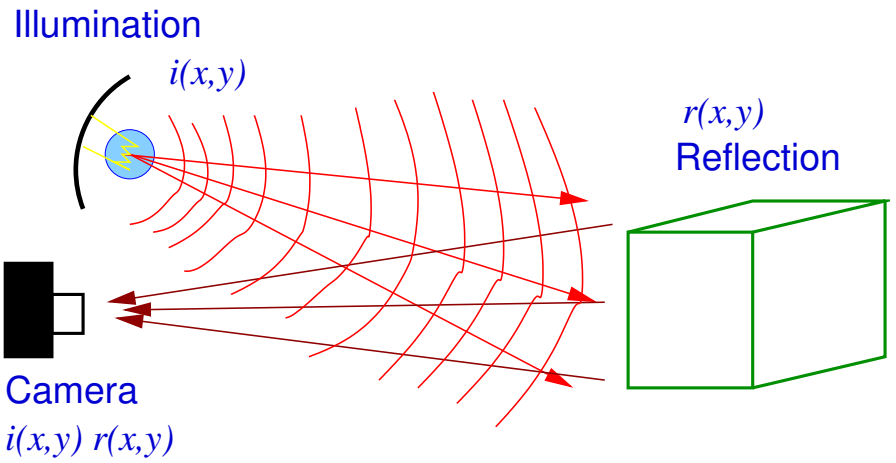


Figure 26: Illumination of a three-dimensional object with a single light source.

low frequency components of the *Cepstrum* will be associated with $\ln(i(x,y))$, the illumination, and the high frequency components with $\ln(r(x,y))$, the object reflectance. Therefore applying a low-pass filter to $Z(u,v)$ will enhance the effect of the illumination and conversely a high-pass filter will enhance the effect of the reflectance. The filtered *Cepstrum* is given by

$$Y(u,v) = Z(u,v)\, H(u,v)$$

Then to form the filtered image, this modified *Cepstrum* must firstly be inverse Fourier transformed, and then the log operation inverted by exp[], giving

$$g(x,y) = \exp\left[\mathcal{F}^{-1}\{Y(u,v)\}\right]$$

It should be noted that the ln[] operation is unstable when $f(x,y) = 0$, so it is typically required to threshold the image to form a wholly positive image $\tilde{f}(x,y)$ by

$$\begin{aligned} \tilde{f}(x,y) &= f(x,y) && \text{for } f(x,y) \geq T \\ &= T && \text{for } f(x,y) < T \end{aligned}$$

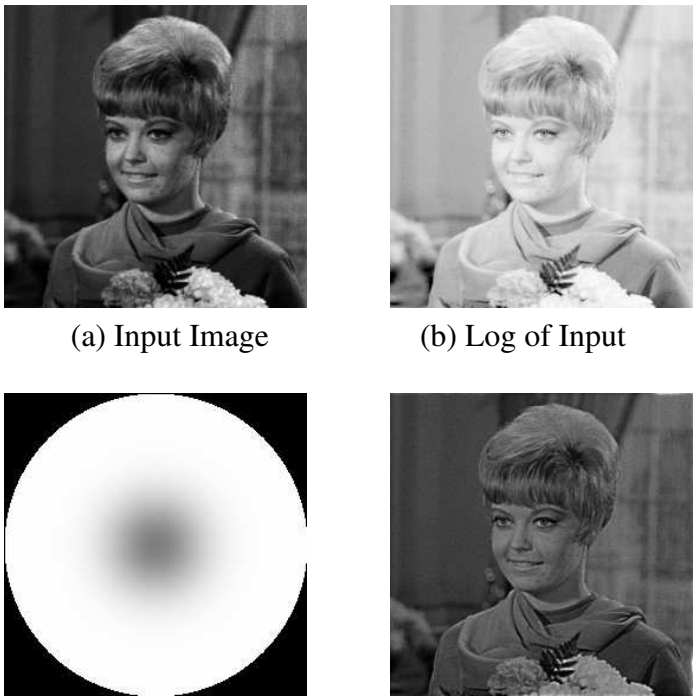
This, in practice, in not a problem since most digital images are in the range $0 \rightarrow 255$ and the above operation can be formed by setting $T = 1$ without significant effect on the image. The filter function, $H(u,v)$ applied to the *Cepstrum* is typically a high frequency enhancement filter that boosts the high frequencies and reduces the low frequencies which enhances the effect of the reflectivity. An example is shown in figure 27 where the Fourier space filter $H(u,v)$ is as shown in figure 27 (c). This has the effect of flattening the overall illumination and enhancing the image highlights.

6.10 Summary

This long section covers digital filtering, which is the most common operation in digital image processing. It has covered,

1. Linear filtering in both real and Fourier space.

2. Examples low and high pass Fourier filters and their basic properties.



(a) Input Image (b) Log of Input

Filter Output

Figure 27: Example of Homomorphic filtering, with (a) the original image, (b) the $\ln()$, (c) the Fourier space filter, (d) final reconstruction.

3. Example in real space linear filters for image smoothing and formation of differentials.
4. Method of combining linear filters in both real and Fourier space.
5. Real space non-linear filters.
6. Shrink and expand filters for image segmentation.
7. Average threshold filters for data drop-out noise removal.
8. Median filters and its edge preserving properties.
9. Homomorphic filtering for correction of illumination variation.

This gives a route of all the main filtering techniques, but to be be actually useful you must play with them and see what they actually do.

Workshop Questions

6.1 At the Edge of an Image

When an image is convolved in real space with a $M \times M$ filter there is a problem of how deal with the edge of the image. Show, with the aid of diagrams how this problem arrises.

There are three conventional schemes for dealing with this problem, there being

1. Cyclic wrap-around of the image.
2. Extend the image with a constant, (zero or image mean).
3. Replicate pixels at the edge.

discuss the merits and de-merits of these three techniques.

6.2 Edge of image: the Fourier Model

There is the same edge problem at he edge of an image when the filter is applied in Fourier space. Which of the above solution to the problem applies in this case.

6.3 In Real and Fourier Space

The convolution theorm states that real space and Fourier space convolution is equivalent. If in real space you apply a 3×3 averaging filter calculate the effect in Fourier space.

it Use the `convolve` and `fourier` programs to show confirm this result.

6.4 Applying Two Filters

You wish to smooth am image by applying a 3×3 9 point average filter followed by a 3×3 laplacian filter. Show that this can be implemented in a single convolution using a 5×5 filter and calculate the elements of this filter.

6.5 Taking Median Filters

Select a standard image and use the program `median` to form Median filters of various sizes. Confirm that you get a smoothed image but retain the edges as theory predicts.

View the Fourier transform of each of these Median Filtered images and comment on what you find. (Save the image from the `median` program from within `xv`, and input this into the `fourier` program.)