CHAPTER 3

DIGITAL IMAGE FILTERING

Introduction

- Direct implementation of two-dimensional FIR digital filters
- Fast Fourier transform implementation of FIR digital filters
- Block methods in the linear convolution calculation
- Inverse filter implementations
- Wiener filters
- Median filter algorithms

Introduction

- Digital filters based on order statistics
- Adaptive order statistic filters
- Histogram and histogram equalization techniques
- Pseudocolouring algorithms
- Digital image halftoning
- Image interpolation algorithms

Direct Implementation of 2D FIR digital filters

The output of a *two-dimensional FIR filter* is given by the linear convolution:

$$y(n_1,n_2)=\sum_{k_1=0}^{M_1-1}\sum_{k_2=0}^{M_2-1}h(k_1,k_2)x(n_1-k_1,n_2-k_2)$$

If the *region of support* of the FIR filter is $[-v_1,v_1]\times[-v_2,v_2]$ where $M_i=2v_i+1$, $i=1,2$, then:

$$y(n_1,n_2)=\sum_{k_1=-v_1}^{v_1}\sum_{k_2=-v_2}^{v_2}h(k_1,k_2)x(n_1-k_1,n_2-k_2)$$

Direct Implementation of 2D FIR digital filters

- Moving Average filter:

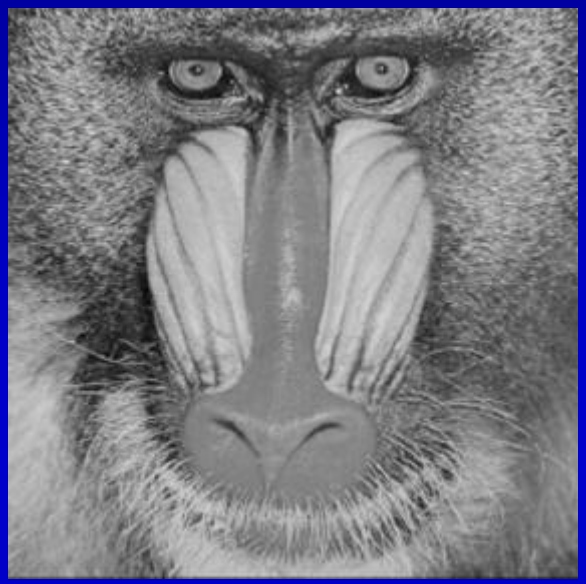
$$y(n_1,n_2)=\left(\frac{1}{M_1M_2}\right)\sum_{k_1=-v_1}^{v_1}\sum_{k_2=-v_2}^{v_2}x(n_1-k_1,n_2-k_2)$$

where $M_i=2v_i+1$, $i=1,2$.

Characteristics of the moving average filter:

- It is very effective in removing white additive Gaussian noise.
- It tends to blur edges and image details (e.g. lines) and degrade image quality.

Moving Average Filter



(a)

(b)

(a) Baboon image,
(b) Filtered Baboon image by a 3×3 moving average filter.

Direct implementation of 2D FIR digital filters

- Zero-phase FIR filters can satisfy $H(\omega_1,\omega_2)=H^*(\omega_1,\omega_2)$ having the spatial symmetry:

$$h(n_1,n_2)=h(-n_1,-n_2)$$

This reduces the number of multiplications almost to half:

$$y(n_1,n_2)=\sum_{k_1=-v_1}^{v1}\sum_{k_2=-v_2}^{v_2}h(k_1,k_2)[x(n_1-k_1,n_2-k_2)+x(n_1+k_1,n_2+k_2)]$$

$$+\sum_{k_1=1}^{v_1}h(k_1,0)[x(n_1-k_1,n_2)+x(n_1+k_1,n_2)]+h(0,0)x(n_1,n_2)$$

FFT implementation of FIR digital filters

Discrete Fourier Transform(DFT):

$$X(k_1,k_2)=\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}x(n_1,n_2)W_{N_1}^{n_1k_1}W_{N_2}^{n_2k_2}$$

Circular Convolution:

$$y(n_1,n_2)=h(n_1,n_2)\otimes\otimes x(n_1,n_2)$$

$$=\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}x(k_1,k_2)h(((n_1-k_1))_{N_1},((n_2-k_2))_{N_2})$$

FFT implementation of FIR digital filters

Important Property of DFT for Circular Convolution:

$$y(n_1, n_2) = h(n_1, n_2) \otimes\otimes x(n_1, n_2) \longleftrightarrow$$

$$Y(k_1, k_2) = H(k_1, k_2) X(k_1, k_2)$$

Circular Convolution Calculation using DFT:

$$y(n_1, n_2) = IDFT[DFT[x(n_1, n_2)]DFT[h(n_1, n_2)]]$$

Linear Convolution can be calculated in the same way if both sequences x ($N_1 \times N_2$) and h ($M_1 \times M_2$) are augmented to dimensions $L_i \geq N_i + M_i - 1$, $i=1,2$.

FIR digital filter implementation

Remarks

- For small filter window dimensions, direct implementation is faster than FFT implementation of FIR filters, provided that:

$$M_1 M_2 < 6\log_2(N_1 N_2) + 4$$

- Memory requirements of the FFT implementation approach are relatively higher than the ones of the direct one.

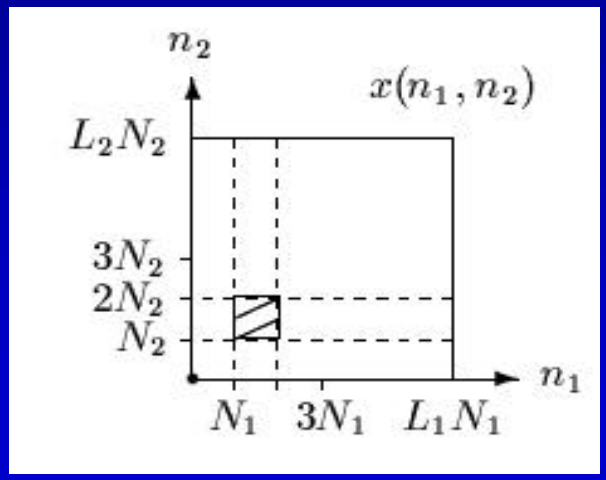
Overlap-add method

The overlap-add method is based on the distributive property of linear convolution and is given as follows:

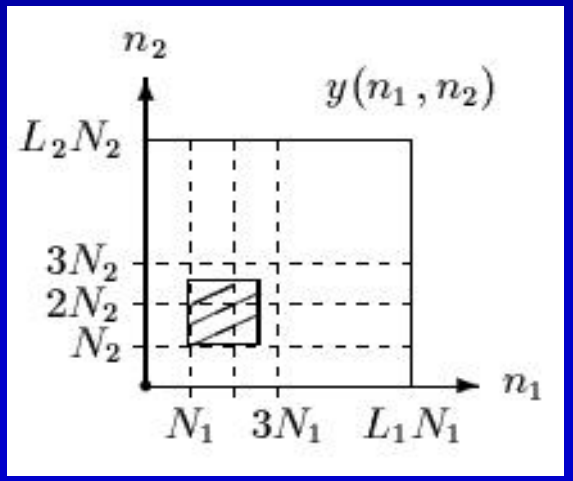
$$y(n_1,n_2) = x(n_1,n_2) **h(n_1,n_2)$$

$$= \sum_{i=1}^{K_1}\sum_{j=1}^{K_2}(x_{ij}(n_1,n_2) **h(n_1,n_2)) = \sum_{i=1}^{K_1}\sum_{j=1}^{K_2} y_{ij}(n_1,n_2)$$

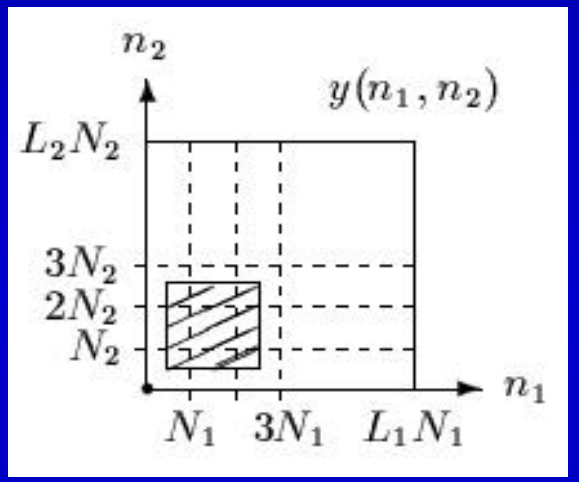
Overlap - Add Block Method



(a)



(b)



(c)

Overlap-add method for linear convolution.
(a) Non-overlapping blocks
(b) convolution output block when h is defined over $[0,M_1) \times [0,M_2)$
(c) convolution output block when h is defined over $[-\acute{\imath}_1, \acute{\imath}_1] \times [-\acute{\imath}_2, \acute{\imath}_2]$.

Block methods in the linear convolution calculation

Overlap-save method

It is based on a simple property of convolution:

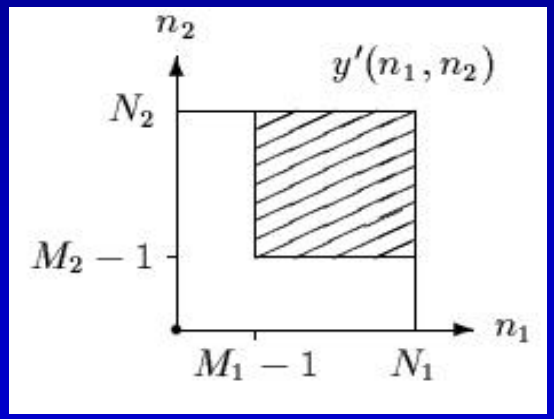
the linear convolution output $y'(n_1,n_2)$ *is equal to the circular convolution output of extent* $N_1 \times N_2$ *only within the rectangle:* $[M_1\text{-}1,N_1] \times [M_2\text{-}1,N_2]$ **:**

$$y'(n_1,n_2) = w(n_1,n_2) **h(n_1,n_2) = w(n_1,n_2) \otimes\otimes h(n_1,n_2)$$

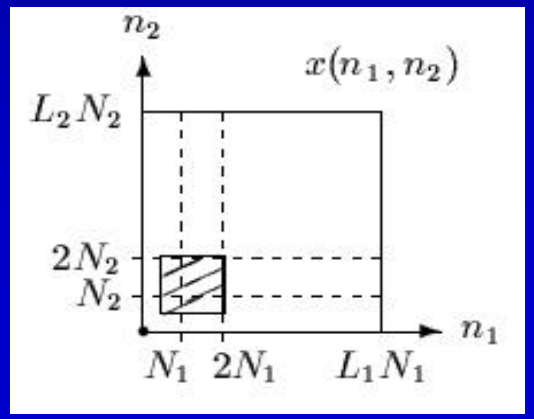
$$\text{for } M_1 - 1 \le n_1 < N_1, M_2 - 1 \le n_2 < N_2$$

$w(n_1,n_2)$ is a sequence defined over $[0,N_1) \times [0,N_2)$.
$h(n_1,n_2)$ is an impulse response defined over $[0,M_1) \times [0,M_2)$.

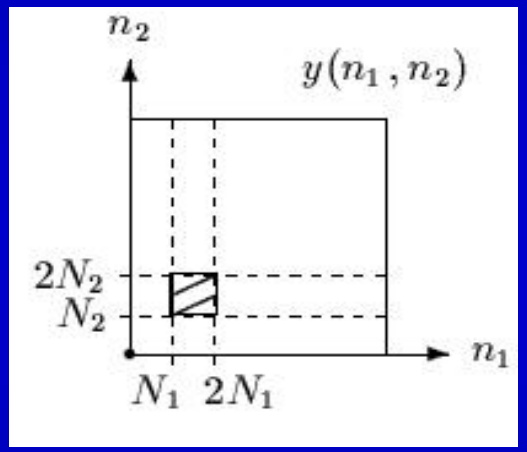
Overlap - Save Block Method



(a)



(b)



(c)

Overlap-save method for linear convolution.
(a) Result of the circular convolution of two sequensces without zero padding
(b) partition of the input sequence in overlapping blocks
(c) output blocks of the overlap-save method

Block methods in the linear convolution calculation

Characteristics of the overlap-add and overlap-save methods

- Both methods have approximately the same computational load, if the block sizes used are approximately equal in both methods.
- If the block size is carefully chosen, they give very good computational savings in comparison to the direct method.
- Although both methods are similar, sometimes the overlap-add method is preferred due to its conceptual simplicity.

Inverse filter implementations

Inverse filtering (for digital image restoration)

$$\hat{F}(w_1, w_2) = \frac{G(w_1, w_2)}{H(w_1, w_2)}$$

where $H(\grave{u}_1, \grave{u}_2)$ is the a priori known degradation function and $G(\grave{u}_1, \grave{u}_2)$ is the observed (degraded) image.

Characteristics :

- It cannot be defined in regions $(\grave{u}_1, \grave{u}_2)$ of the transform domain, where $H(\grave{u}_1, \grave{u}_2)$ is zero.
- It is very sensitive to the presence of the formation noise.

Inverse filter implementations

(a)

(b)

(a) Original image,
(b) Image corrupted by horizontal blur of length $\acute{I}_1 = 5$ and white additive Gaussian noise having variance equal to 20.

Inverse filter implementations

The last problem can be solved by using a pseudoinverse filter:

$$H^{-}(w_1,w_2)=\begin{cases}\dfrac{1}{Ç(w_1,w_2)} & \text{if } |H(w_1,w_2)|>e \\ 0 & \text{otherwise}\end{cases}$$

Implementation of the inverse filter using the DFT

$$\hat{f}(n_1,n_2)=IDFT\left[\frac{G(k_1,k_2)}{H(k_1,k_2)}\right]$$

Inverse filter implementations

Iterative implementation of the inverse filter

$$\hat{F}_{k+1}(w_1,w_2) = \hat{F}_k(w_1,w_2) + m[G(w_1,w_2) - \hat{F}_k(w_1,w_2)H(w_1,w_2)]$$

Characteristics of the iterative implementation

- The advantage of the iterative method is that it can be stopped after a certain number of iterations if the filtering output is acceptable.
- The convergence parameter ì can be changed in order to alter the convergence speed.

Wiener filters

Non-casual Wiener filter

$$H_w(w_1,w_2)=\frac{P_{sg}(w_1,w_2)}{P_{gg}(w_1,w_2)}$$

P_{sg} and P_{gg} are the cross-power spectrum of s (original), g (observed image) and the power spectrum of g respectively.

- If the signal $s(n_1,n_2)$ is uncorrelated with noise $n(n_1,n_2)$:

$$H_w(w_1,w_2)=\frac{P_{ss}(w_1,w_2)}{P_{ss}(w_1,w_2)+P_{nn}(w_1,w_2)}$$

- This implementation can be efficiently used for additive noise removal.

Modification of the Wiener filter used in digital image restoration

$$H_w(w_1,w_2)=\frac{H^*(w_1,w_2)P_{ff}(w_1,w_2)}{|H(w_1,w_2)|^2\ P_{ff}(w_1,w_2)+P_{nn}(w_1,w_2)}$$

An *important problem* in the design and implementation of Wiener filters is the estimation of the blur transfer function $H(\grave{u}_1,\grave{u}_2)$ and of the power spectra $P_{ff}(\grave{u}_1,\grave{u}_2)$, $P_{nn}(\grave{u}_1,\grave{u}_2)$.

Median filter algorithms

The median value is the middle observation $x_{(v+1)}$ of the statistically ordered observations x_i, $i=1,..,n$:

$$x_{(1)} < x_{(2)} < \ldots < x_{(n)}$$

$x_{(1)}$: minimum, $x_{(2)}$: maximum

A two-dimensional median filter is defined as follows:

$$y(i,j)=\text{med}\{\, x(i+r,j+s),\ (r,s) \in A \quad (i,j) \in Z^2 \,\}$$

Median filter properties

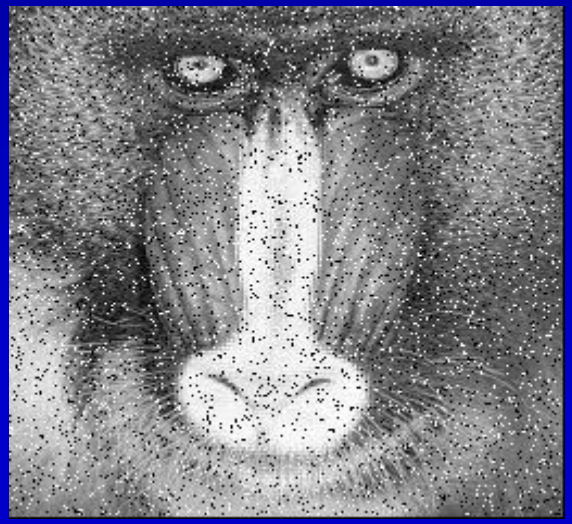
- They have low-pass characteristics and they remove additive white noise.
- They are very efficient in the removal of noise that has a long-tailed distribution (e.g. Laplacian distribution).

Median filter algorithms

Median filter properties *continued*

- The median becomes unreliable only when more than 50% of the data are outliers.
- The robustness properties of the median make it very suitable for impulse noise filtering.
- The median filter tends to preserve edge sharpness.
- The median filter not only smooths noise in homogeneous image regions but tends to produce regions of constant or nearly constant intensity.

Median filter algorithms



Baboon image corrupted by mixed impulsive noise

The output image of a 7 x 7 median filter

The output image of a 7 x 7 moving average filter

Separable 2D median filter

It results from two successive applications of 1D median filters of length n along rows and then columns (or vice-versa):

$$y_{ij}=\text{med}(z_{i,j-v}\ ,\ \dots\ ,\ z_{ij}\ ,\ \dots\ ,\ z_{i,j+v})$$

$$z_{ij}=\text{med}(x_{i-v,j}\ ,\ \dots\ ,\ x_{ij}\ ,\ \dots\ ,\ x_{i+v,j})$$

$$n=2\acute{\imath}+1.$$

Advantage: Low computational complexity in comparison to that of the non-separable median filter because it sorts n numbers two times instead of n^2.

Median filter algorithms

Recursive median filter

$$y_{ij}=\text{med}(y_{i-v}, \ldots, y_{i-1}, x_i, \ldots, x_{i+v})$$

- Its output tends to be much more correlated than that of the standard median filter.
- Recursive median filters have higher immunity to impulsive noise than the non-recursive median filters.

Separable recursive median filter

$$y_{ij}=\text{med}(y_{i,j-v}, \ldots, y_{i,j-1}, z_{ij}, \ldots, z_{i,j+v})$$

$$z_{ij}=\text{med}(z_{i-v,j}, \ldots, z_{i-1,j}, x_{ij}, \ldots, x_{i+v,j})$$

Weighted median filter

The *weighted median* is the estimator T that minimizes the weighted L_1 norm of the form:

$$\sum_{i=1}^{n} w_i \mid x_i - T \mid \rightarrow \min$$

The weighted median filter is described by:

$$y_i = \text{med}\{w_{-v} \diamond x_{i-v} , \ldots , w_v \diamond x_{i+v}\}$$

where $w \diamond x$ denotes duplication of x ($x,\ldots,x$, w times)

Multistage median filter

$$y_{ij}=\text{med}(\text{med}(z_1,z_2,x_{ij}) , \text{med}(z_3,z_4,x_{ij}) , x_{ij})$$

$$z_1=\text{med}(x_{i,j-v} , \cdots , x_{ij} , \cdots , x_{i,j+v})$$

$$z_2=\text{med}(x_{i-v,j} , \cdots , x_{ij} , \cdots , x_{i+v,j})$$

$$z_3=\text{med}(x_{i+v,j-v} , \cdots , x_{ij} , \cdots , x_{i-v,j+v})$$

$$z_4=\text{med}(x_{i-v,j-v} , \cdots , x_{ij} , \cdots , x_{i+v,j+v})$$

It can preserve edges in horizontal, vertical and diagonal directions.

Ranked order filters

An r-th ranked filter of the signal x_i is the r-th order statistic:

$$y_i = r\text{-th order statistic of } \{x_{i-v}, \ldots, x_i, \ldots, x_{i+v}\}$$

- It introduces a strong bias in the estimation of the mean,when the rank is small or large.
- The bias is even stronger when the input data have a long-tailed distribution.

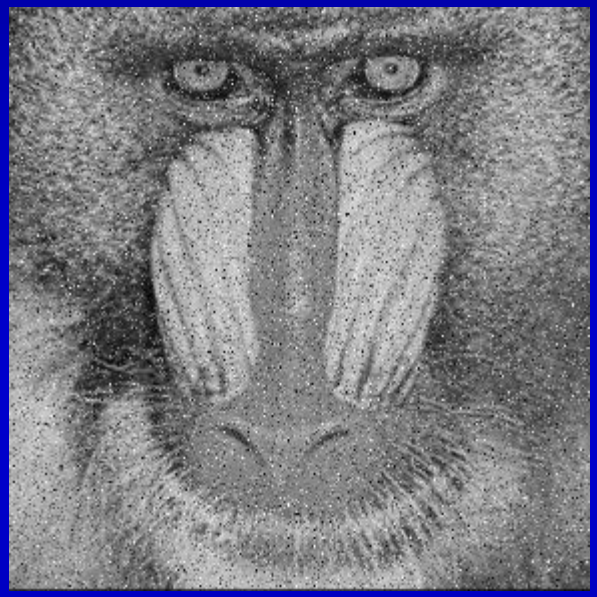
Max/min filters

The maximum $x_{(n)}$ and the minimum $x_{(1)}$ are the two extremes of the ranked-order filters.

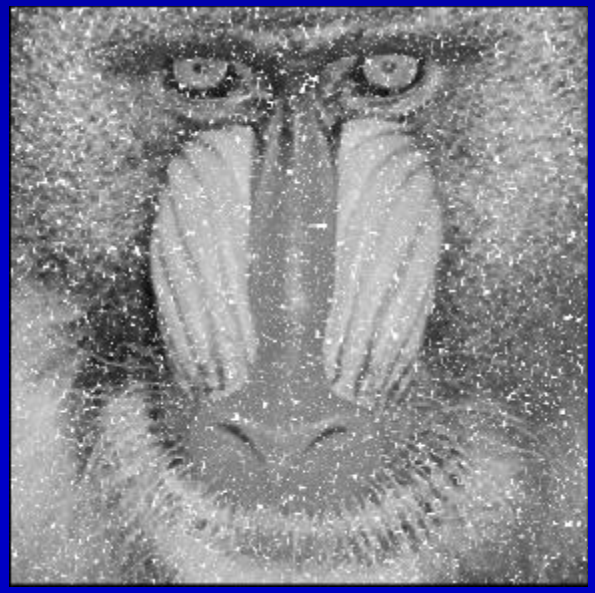
- The maximum filter effectively removes negative impulses in an image. The minimum filter removes positive impulses.
- Both filters fail in the removal of mixed impulse noise.
- Both filters have good edge preservation properties.
- Both filters tend to enhance the bright and the dark regions of the image respectively (max/min).

Digital filters based on statistics

Max/min filters



Baboon image
corrupted by
mixed impulsive noise



The output of a
cascade of a
min and a max
filter

Max filter (running implementation)

$$y_i = \begin{cases} x_i & \text{if } x_i \geq y_{i-1} \\ y_{i-1} & \text{if } x_i < y_{i-1} \text{ and } x_{i-n} < y_{i-1} \\ \max(x_i, \ldots, x_{i-n+1}) & \text{if } x_i < y_{i-1} \text{ and } x_{i-n} = y_{i-1} \end{cases}$$

In average, only 3 comparisons are needed. A similar algorithm exists for min filtering.

á-trimmed mean filters

$$y_i = \frac{1}{n(1-2a)} \sum_{j=an+1}^{n-an} x_{(j)}$$

- The á-trimmed mean filter rejects á% of the smaller and á% of the larger observation data.
- It can be used as a compromise between the median filter and the moving average filter for varying á.
- Its performance is poor for short-tailed distributions.

The *midpoint* is defined as follows: $MP = \frac{1}{2}(x_{(1)} + x_{(n)})$

Modified trimmed mean filter (MTM)

$$y_{ij} = \frac{\sum\sum_A a_{rs} x_{i+r,j+s}}{\sum\sum_A a_{rs}}$$

Its coefficients are chosen by: ⇒

$$a_{rs} = \begin{cases} 1 & |x_{i+r,j+s} - med\{x_{ij}\}| \le q \\ 0 & \text{otherwise} \end{cases}$$

Modified nearest neighbour filter (MNN)

Its coefficients are chosen by: ⇒

$$a_{rs} = \begin{cases} 1 & |x_{i+r,j+s} - x_{ij}| \le q \\ 0 & \text{otherwise} \end{cases}$$

This filter trims out pixels deviating strongly from the central pixel. Therefore it has good edge preservation properties.

L-filters

The L-filter (or order statistic) filter is defined as follows:

$$y_i = \sum_{j=1}^{n} a_j x_{(j)}$$

Location Invariance constraint:

$$\sum_{j=1}^{n} a_j = \mathbf{a}^T \mathbf{e} = 1$$

Choice of coefficient vector after MSE minimization:

$$\mathbf{a} = \frac{\mathbf{R}^{-1}\mathbf{e}}{\mathbf{e}^T \mathbf{R}^{-1}\mathbf{e}}$$

L-filters

- The optimal L-filter for the Gaussian noise is the moving average.
- The optimal L-filter for the Laplacian distribution is close to the median.
- The optimal L-filter for the uniform distribution is the midpoint.
- The L-filter has no streaking effects, provided that its coefficients are not similar to those of the median.
- It has greater computational complexity than the median or the moving average.

Minimal Mean Square Error filter (MMSE)

An adaptive filter for additive white noise:

$$x_{ij}=s_{ij}+n_{ij}$$

Linear Minimal Mean Square Error filter output:

$$\hat{s}_{ij} = (1-\frac{s_n^2}{s_x^2})x_{ij} + \frac{s_n^2}{s_x^2}\hat{m}_x$$

- The MMSE filter preserves edges, although it does not filter the noise in edge regions.
- The performance of the adaptive MMSE filter depends on the choice of the local measures of signal mean and standard deviation and of the noise standard deviation.

Decision-directed filters

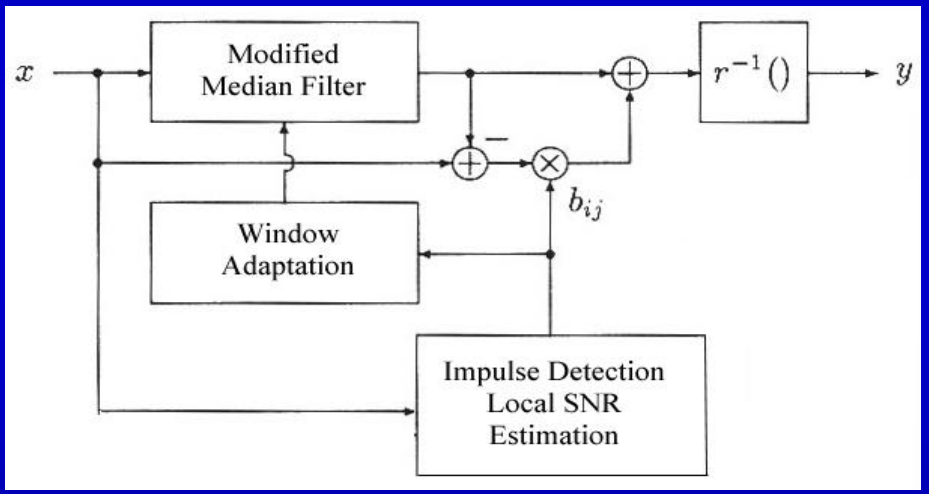
- They can take into account both edge and impulsive noise information.
- Impulses, when detected, can be removed from the estimation of the local mean, median and standard deviation.
- When an edge is detected, the windows of the filter can become smaller so that edge blurring is minimized.
- Such an impulsive-sensitive filter is the adaptive window edge detection (AWED) filter.

Two-component model filters

An adaptive filter based on the two-component model is the signal-adaptive median (SAM) filter:

$$y_{1ij} = \hat{m}_x + b_{ij}(x_{ij} - \hat{m}_x)$$

$$y_{ij} = r^{-1}(y_{1ij})$$



The SAM filter has excellent performance in noise filtering, edge and image detail preservation.

Adaptive order statistic filters

Two component model filters

Original
Lenna image

Lenna image
corrupted by Gaussian
noise (variance=100) and
mixed impulsive noise

The output of a
SAM filter

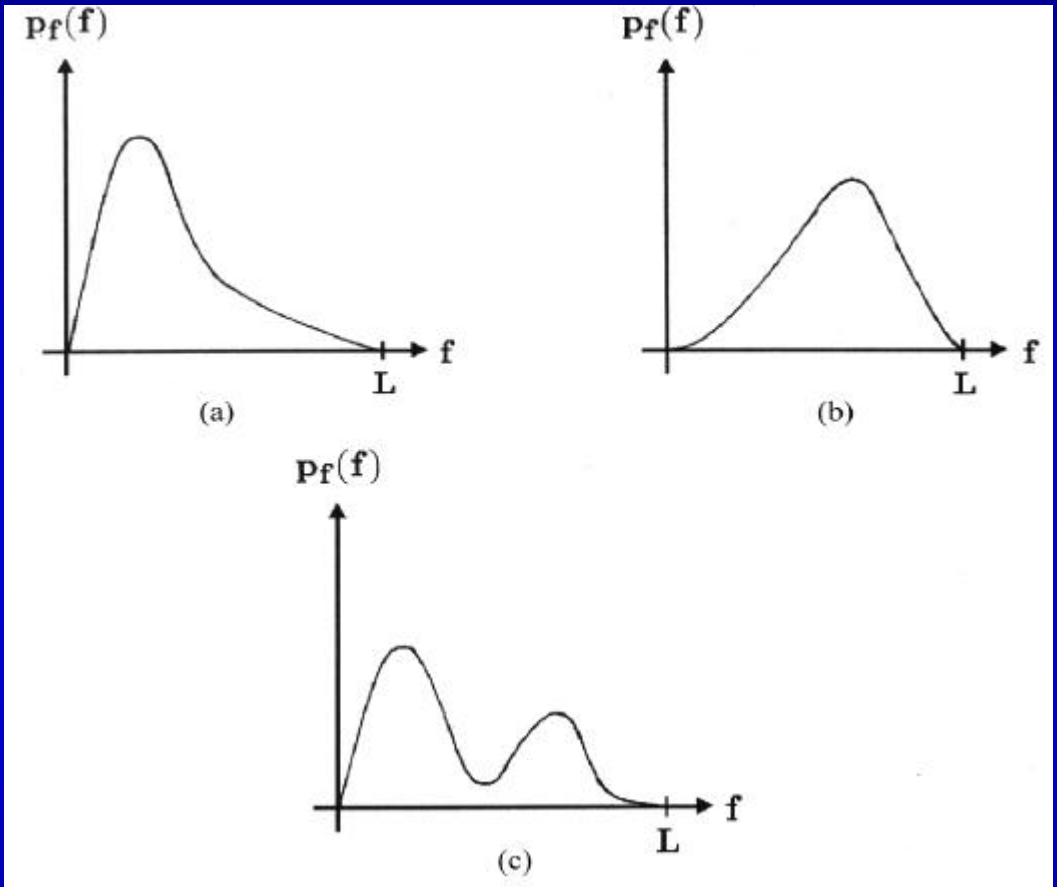
Histogram and histogram equalization techniques

The histogram $\hat{p}_f$ (empirical pdf) is given by the relation:

$$\hat{p}_f(f_k) = \frac{n_k}{n} \quad k = 0,1,\ldots, L-1$$

- The image quality can be enhanced by modifying its histogram.
- This can be performed by a technique called histogram equalization.

Histogram and histogram equalization techniques



(a) Histogram of a dark image,
(b) Histogram of a bright image,
(c) Histogram of an image with two intensity concentrating regions

Histogram equalization

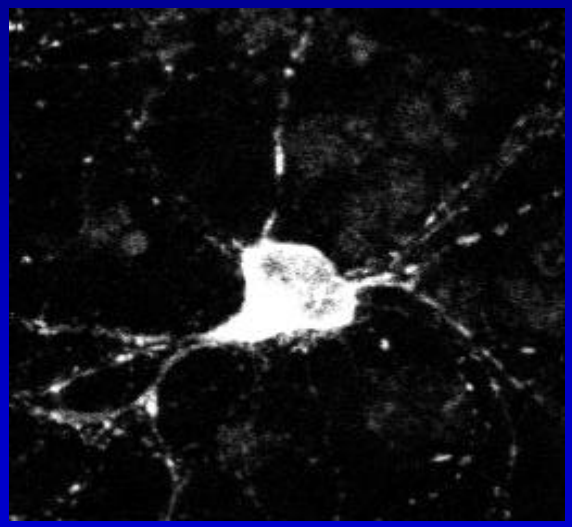
- Transformation function:

$$T(f) = \int_0^f p_f(w)dw$$

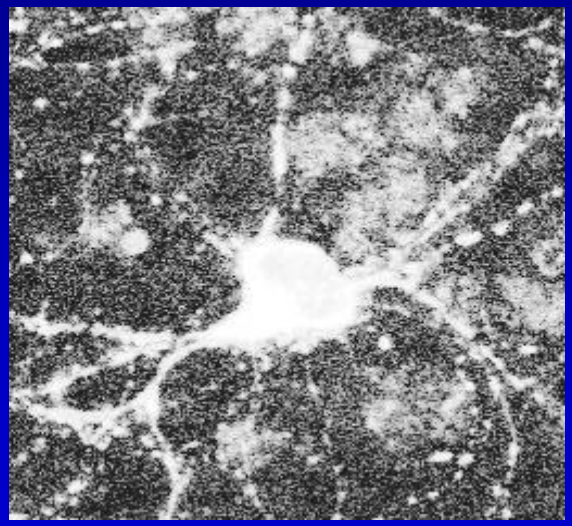
- $T(f)$ can be calculated from the following relation:

$$g_k = T(f_k) = \sum_{j=0}^{k} \hat{p}_f(f_j) = \sum_{j=0}^{k} \frac{n_j}{n}$$

Histogram and histogram equalization techniques



(a)



(b)

(a) Original image, (b) image after histogram equalization

Histogram modification

- Transformation function:

$$g = G^{-1}[T(f)]$$

Pseudocolouring algorithms

Pseudocolouring encoding of the intensity of black and white (BW) images by using colour information:

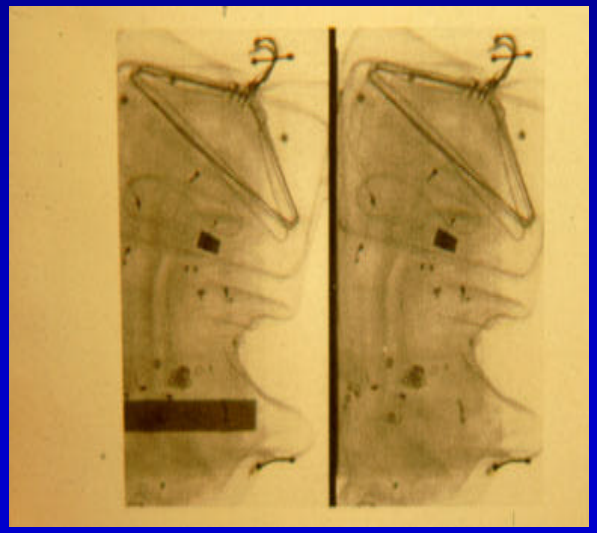
Pseudocolouring is a digital image transformation of the form:

$$c(x, y) = T(f(x, y))$$

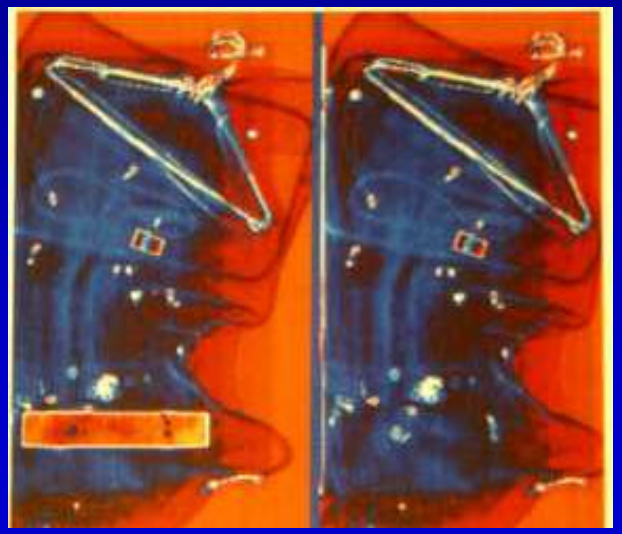
where $f(x,y)$ is a BW image and $\boldsymbol{c}(x,y)$ is a colour image.

- The choice of the transformation function is heuristic based on subjective image quality evaluation

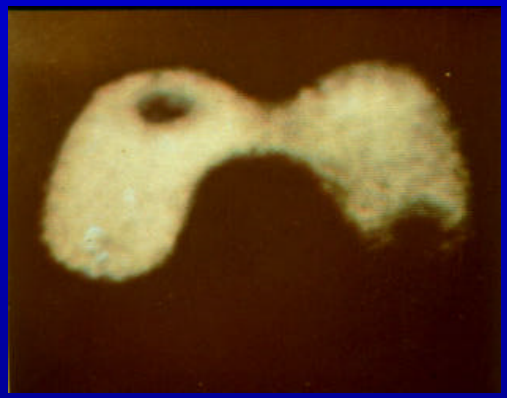
Pseudocolouring algorithms

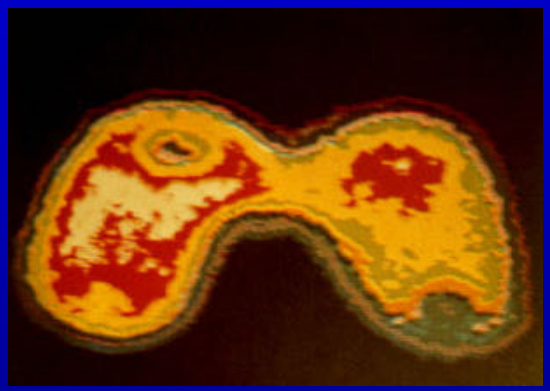


Original Image



Pseudocoloured Image





Intensity Quantization (slicing) method

It is equivalent to the following non-uniform transformation function, that occurs after histogram equalization $g=G(f)$:

$$\mathbf{T}(f(k,l)) = \begin{cases} \mathbf{c}_i & \text{if } i\left[\frac{L}{N}\right] \le G(f(k,l)) < (i+1)\left[\frac{L}{N}\right] \\ & i = 0,1,\ldots,N-2 \\ \mathbf{c}_{N-1} & \text{if } (N-1)\left[\frac{L}{N}\right] \le G(f(k,l)) < L \end{cases}$$

Filtering Approach

$h_L(k,l)$, $h_B(k,l)$, $h_H(k,l)$: impulse responses of a low-pass, a bandpass and a high-pass linear FIR filter.
They are used to produce the colour image components of the pseudocoloured image:

$$c_R(k,l) = f(k,l) **h_H(k,l)$$
$$c_G(k,l) = f(k,l) **h_B(k,l)$$
$$c_B(k,l) = f(k,l) **h_L(k,l)$$

Digital image halftoning

Binary Thresholding

$$g(k,l) = \begin{cases} 1 & \text{if } f(k,l) \geq T \\ 0 & \text{otherwise} \end{cases}$$

$f(k,l)$: greyscale image (input)
$g(k,l)$: thresholded image (output)

- Threshold selection T can be based on the image histogram.
- A locally adaptive threshold is better than a global one.
- Binary thresholding does not produce halftone images.

Digital image halftoning

Binary Thresholding

Original Image

Thresholded Image with $\hat{O}$=100

Thresholded Image with $\hat{O}$=200

Digital image halftoning

Greyscale Binary Fonts

- The greyscale image $f(k,l)$ has L grey levels.
- The halftone image $g(k,l)$ must have N perceived grey levels ($N<<L$, N=nxn+1).
- N matrices $\mathbf{F}_i$ of size nxn containing 1s can be used for halftoning:

$$\mathbf{F} = \begin{cases} \mathbf{F}_i & \text{if } i\left[\frac{L}{N}\right] \le f(k,l) < (i+1)\left[\frac{L}{N}\right],\ i = 0,1,\ldots,N-2 \\ \mathbf{F}_{N-1} & \text{if } (N-1)\left[\frac{L}{N}\right] \le f(k,l) < L \end{cases}$$

- The method is conceptually simple and easy to implement
- It creates false lines and contours in homogeneous regions

Digital image halftoning

Greyscale Binary Fonts

Original Image

Original Image subsampled by a factor of 2

Halftoned Image by using greyscale binary fonts

Digital image halftoning

Pseudorandom Thresholding

- It adds random noise to the image and then thresholds.
- This is equivalent to thresholding with a random threshold.
- *Dither matrices* containing pseudorandom thresholds are used, denoted by D^n if their size is $n \times n$. A 2×2 dither matrix is defined by:

$$D^2 = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}$$

- Halftoning is performed by:

$$g(k,l) = \begin{cases} 1 & \text{if } f(k,l) > T(k,l) \\ 0 & \text{otherwise} \end{cases}$$

$$T(k,l) = D^n(k \bmod n, l \bmod n)$$

Image interpolation algorithms

- *Zero-order hold interpolation*: a (x,y) point is assigned the value of the geometrically closest pixel in the image array. It produces regions with constant intensity and leads to zooming by a factor of $2^n \times 2^n$:

$$f_i(n_1, n_2) = f\left([n_1 / 2], [n_2 / 2]\right)$$

- *First-order (linear) interpolation*: it produces smoother interpolated images

$$f(x, y) = (1-\Delta_1)(1-\Delta_2) f(n_1, n_2) + (1-\Delta_1)\Delta_2 f(n_1, n_2+1) +$$
$$\Delta_1(1-\Delta_2) f(n_1+1, n_2) + \Delta_1\Delta_2 f(n_1+1, n_2+1)$$

$$\Delta_1 = \frac{x - n_1 T_1}{T_1} \quad , \quad \Delta_2 = \frac{y - n_2 T_2}{T_2}$$

Image interpolation algorithms

- *p-order interpolation*: it is performed by convolving the appropriately formed image with the convolution matrix **H** *p* times (e.g. cubic spline interpolation, *p*=3).

  First we interlace the image to be interpolated with zeros.

$$f'(n_1,n_2)=\begin{cases} f(\frac{n_1}{p},\frac{n_2}{p}) & \text{if } n_1=pk, n_2=pl \\ 0 & \text{otherwise} \end{cases}$$

An example of a convolution matrix **H** is:

$$\mathbf{H}=\begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix}$$

Image interpolation algorithms

BABOON image

Output image after zero-order interpolation

Output image after linear interpolation

Output image after cubic spline interpolation