Efficient Nonlinear Image Processing Algorithms

SANJIT K. MITRA

Department of Electrical & Computer Engineering

University of California

Santa Barbara, California

Outline

- Introduction
- Quadratic Volterra Operators
- Image Processing Applications
  - Contrast Enhancement
  - Impulse Noise Removal
  - Image Zooming
  - Image Halftoning

Introduction

- Linear image processing algorithms have received considerable attention during the last several decades
- They are easy to implement and are computationally less intensive
- The basic hypotheses for the development of linear models and linear signal processing algorithms are stationarity and Gaussianity

Introduction

- To achieve improved performance, algorithms must take into account
  - nonlinear effects in the human visual system
  - nonlinear behavior of the image acquisition systems

Introduction

- The hypotheses of stationarity and Gaussianity do not hold in the case of image signals
- Linear filtering methods applied to an impulse-noise-corrupted image blur sharp edges and remove fine details
- Linear algorithms are not able to remove signal-dependent or multiplicative noise in images

Introduction

- This has led to a growing interest in the development of nonlinear image processing methods in recent years
- Due to rapidly decreasing cost of computing, image storage, image acquisition, and display, complex nonlinear image processing algorithms have also become more practical for implementation

Introduction

- Types of Nonlinear Algorithms:
  - Homomorphic filters
  - Nonlinear mean filters
  - Morphological filters
  - Order statistic filters
  - Polynomial filters
  - Fuzzy filters
  - Nonlinear partial differential equation based filters

Discrete-Time Volterra Filters

- The Volterra filter is a special case of the polynomial filters
- It is based upon an input-output relation expressed in the form of a truncated Volterra series
- Simplest types are the quadratic filters corresponding to the first nonlinear term in the Volterra expansion

Discrete-Time Volterra Filters

- Two attractive and important properties of the Volterra filters, and in particular, of the quadratic filters

Discrete-Time Volterra Operators

First Property

- Output depends linearly on the coefficients of the filter itself
- Is used to analyze the behavior of the filters, find new realizations, derive adaptive algorithms, etc.

Discrete-Time Volterra Operators

Second Property

- Results from the representation of the nonlinearity by means of multi-dimensional operators working on the products of input samples
- Allows for the frequency domain description of the filters by means of multi-dimensional convolution

Discrete-Time Volterra Operators

- The general form of the Volterra filter is described by the input-output relation:

$$y[n] = h_0 + \sum_{k=1}^{\infty} \hat{h}_k(x[n])$$

- $y[n]$ and $x[n]$ are, respectively, the output and input sequences, and

$$\hat{h}_k(x[n]) = \sum_{i_1=0}^{\infty} \cdots \sum_{i_k=0}^{\infty} \hat{h}_k[i_1, \ldots, i_k] \cdot x[n - i_1] \cdots x[n - i_k]$$

Discrete-Time Volterra Operators

- In the expression for $\hat{h}_k(x[n])$, the discrete variables $i_1,\ldots,i_k$ are usually defined on a causal support
- h_0 is an offset term
- $\hat{h}_k[i_1]$ is the impulse response of a linear FIR filter
- $\hat{h}_k[i_1,\ldots,i_k]$ can be considered as a generalized k-th order impulse response characterizing the nonlinear behavior

1-D Quadratic Volterra Filters

Infinite Memory Quadratic Filters

- Input-output relation

$$
\begin{aligned}
y[n] &= h_0 + \hat{h}_1(x[n]) + \hat{h}_2(x[n]) \\
&= h_0 + \sum_{i_1=0}^{\infty} h_1[i_1] \cdot x[n-i_1] \\
&\quad + \sum_{i_1=0}^{\infty} \sum_{i_2=0}^{\infty} h_2[i_1, i_2] \cdot x[n-i_1] \cdot x[n-i_2]
\end{aligned}
$$

1-D Quadratic Volterra Filters

Finite Memory Quadratic Filters

- Input-output relation

$$y[n] = h_0 + \sum_{i_1=0}^{N_1-1} h_1[i_1] \cdot x[n-i_1]$$

$$+ \sum_{i_1=0}^{N_2-1} \sum_{i_2=0}^{N_3-1} h_2[i_1, i_2] \cdot x[n-i_1] \cdot x[n-i_2]$$

1-D Quadratic Volterra Filters

Transform-Domain Representation

- Convolution form of quadratic term

$$y[n] = \hat{h}_2(x[n]) = \sum_{i_1=0}^{\infty} \sum_{i_2=0}^{\infty} h_2[i_1, i_2] \cdot x[n - i_1] \cdot x[n - i_2]$$

can be expressed as

$$w[n_1, n_2] = \sum_{i_1=0}^{\infty} \sum_{i_2=0}^{\infty} h_2[i_1, i_2] \cdot v[n_1 - i_1, n_2 - i_1]$$

1-D Quadratic Volterra Filters

- For

$$v[n_1 - i_1, n_2 - i_2] = x[n - i_1] \cdot x[n - i_2]$$

  and

$$n_1 = n_2 = n$$

  so that

$$y[n] = w[n,n]$$

1-D Quadratic Volterra Filters

- The two-dimensional (2-D) Fourier transform of $h_2[n_1,n_2]$ given by

$$H_2(e^{j\omega_1},e^{j\omega_2}) = \sum_{k_1=-\infty}^{\infty}\sum_{k_2=-\infty}^{\infty} h_2[k_1,k_2]e^{-j\omega_1 k_1}e^{-j\omega_2 k_2}$$

is defined as the **frequency response** of the quadratic Volterra filter

1-D Quadratic Volterra Filters

- The properties of the 2-D Fourier transform can be used to characterize the quadratic kernel $h_2[i_1,i_2]$
- For example, the expression for $y[n]$ can be derived using the inverse 2-D Fourier transform

$$y[n] = \int_{-1/2}^{1/2}\int_{-1/2}^{1/2} H_2(e^{j2\pi f_1}, e^{j2\pi f_2}) X(f_1) X(f_2) \cdot e^{j2\pi(f_1+f_2)} df_1 df_2$$

1-D Quadratic Volterra Filters

- where $X(f)$ is the Fourier transform of $x[n]$
- Note: If the input to a quadratic filter is a sinusoid, i.e. if $x[n] = Ae^{j2\pi f_a n}$

  then the output is

  $$y[n] = A^2 H_2\left(e^{j2\pi f_a}, e^{j2\pi f_a}\right)e^{j2\pi 2 f_a n}$$

  which is still a sinusoid but with a frequency $2 f_a$

1-D Quadratic Volterra Filters

- If the input is a sum of two sinusoids with frequencies f_a and f_b , then the output contains three sinusoids of frequencies $2f_a$, $2f_a$, and $f_a + f_b$

1-D Quadratic Volterra Filters

- As every kernel can be transformed into a symmetrical form, we restrict our attention here to Volterra filters with a symmetric impulse response, i.e.,

$$h_2[n_1,n_2] \quad h_2[n_2,n_1]$$

Teager's 1-D Operator

- An example of the quadratic Volterra filter is the **Teager's operator**

$$y[n] = x^2[n] - x[n-1]x[n+1]$$

- Introduced by Kaiser to calculate the energy $y[n]$ of a one-dimensional (1-D) signal $x[n]$

Teager's 1-D Operator

- If the input is $x[n] = A\cos(\omega_o n + \phi)$, then the Teager's operator generates an output

$$
\begin{aligned}
y[n] &= A^2 \cos^2(\omega_o n + \phi) \\
&\quad - A^2 \cos\big(\omega_o(n+1) + \phi\big)\cos\big(\omega_o(n-1) + \phi\big) \\
&= A^2 \sin^2(\omega_o) \cong A^2\omega_o^2
\end{aligned}
$$

for small values of ω_o

Teager’s 1-D Operator

- Thus, for sinusoidal inputs, the Teager operator develops a constant output which is an estimate of the physical energy of a pendulum oscillating with a frequency ω_o and an amplitude A

Teager's 1-D Operator

- Under some mild conditions, the 1-D Teager operator can be approximately represented as

$$y[n] \cong \mu_x \cdot \left(2x[n] - x[n-1] - x[n+1]\right)$$

- In the above

$$\mu_x = \frac{1}{3}\left(x[n-1] + x[n] + x[n+1)\right)$$

is the **local mean**

Teager's 1-D Operator

and the quantity

$$2x[n]-x[n-1]-x[n+1]$$

is the Laplacian operator which is an FIR highpass filter

- Thus, the 1-D Teager operator behaves as a mean-weighted highpass filter

Teager’s 1-D Operator

- The 1-D Volterra filters that can be represented approximately as a local-mean-weighted highpass filter satisfy the following three conditions:

(1) $H_2(e^{j0}, e^{j0}) = \sum_{k_1} \sum_{k_2} h_2[k_1, k_2] = 0$

(2) $H_2(e^{j\omega_1}, e^{j0}) = H_2(e^{j0}, e^{j\omega_2})$

and

Teager's 1-D Operator

(3) $$\sum_{k_1}\sum_{\substack{k_2 \\ k_1 \neq k_2}} h_2[k_1,k_2]\left(h_2[k_1,k_3]+h_2[k_1,k_3]\right)$$

$$+\sum_{k_1}\sum_{\substack{k_2 \\ k_1 \neq k_2}}\sum_{\substack{k_3 \\ k_2 \neq k_3}} h_2[k_1,k_2]\left(h_2[k_1,k_3]+h_2[k_1,k_3]\right) \geq 0$$

- A large class of such filters satisfies the above three conditions

Teager’s 1-D Operator

- The frequency-domain input-output relation of filters belonging to this class can be expressed as:

$$Y(e^{j\omega}) \approx 2\mu_x \cdot H_2(e^{j\omega}, e^{j0}) X(e^{j\omega})$$

- $Y(e^{j\omega})$ and $X(e^{j\omega})$ denote, respectively, the 1-D Fourier transforms of the output $y[n]$ and the input $x[n]$

Teager’s 1-D Operator

- If $H_2(e^{j\omega}, e^{j0})$ is a highpass filter, then the quadratic Volterra filter given by

$$y[n] = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h_2[k_1, k_2] x[n-k_1] x[n-k_2]$$

  satisfying the three conditions stated earlier can be approximated as a local-mean-weighted highpass filter

Teager's 1-D Operator

- The 1-D Teager operator

$$y[n] = x^2[n] - x[n-1]x[n+1]$$

  is an example of such a filter
- It maps sinusoidal inputs to constant outputs
- Every filter belonging to the class of local-mean-weighted highpass filters has the above property

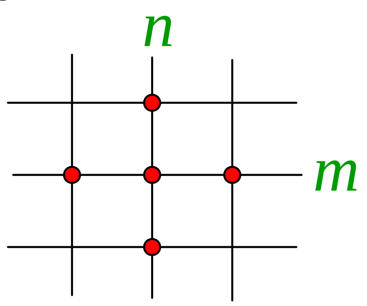
2-D Teager Operator

- A 2-D extension of the Teager operator is obtained by applying the filtering operation

$$y[n] = x^2[n] - x[n-1]x[n+1]$$

along both the vertical and horizontal directions:

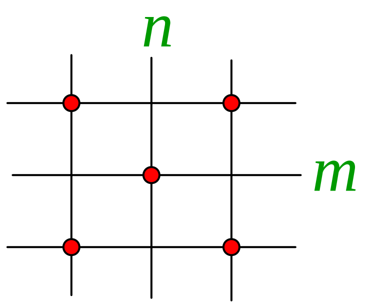
$$y[m,n] = 2x^2[m,n] - x[m-1,n]x[m+1,n] - x[m,n-1]x[m,n+1]$$



2-D Teager Operator

- Another 2-D extension is obtained by applying the 1-D operator along the two diagonal directions

$$y[m,n] = 2x^2[m,n] - x[m-1,n+1]x[m+1,n-1] - x[m-1,n-1]x[m+1,n+1]$$



2-D Teager Operator

- Both of the above two 2-D quadratic filters can be approximated as a local-mean-weighted highpass 2-D filter
- The general class of 2-D quadratic Volterra filters that can be approximately represented as a mean-weighted highpass filter is characterized by three conditions similar to that satisfied by the 1-D Teager operator

2-D Teager Operator

- Based on this analysis, a number of other local-mean-weighted highpass 2-D filters have been developed
- Another member of this class, for example, is the filter defined by

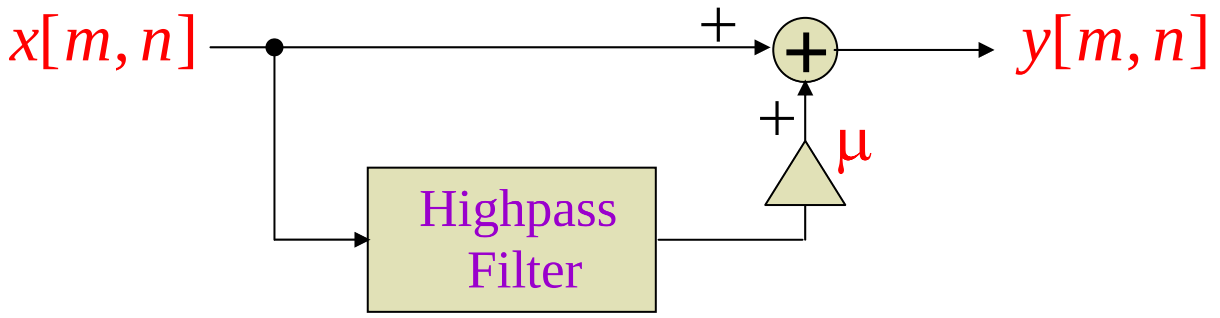
$$
\begin{aligned}
y[m,n] = 3x^2[m,n] &- \tfrac{1}{2}x[m+1,n+1]x[m-1,n-1] \\
&- \tfrac{1}{2}x[m+1,n-1]x[m-1,n+1] \\
&- x[m+1,n]x[m-1,n] - x[m,n+1]x[m,n-1]
\end{aligned}
$$

Image Processing Applications

- The mean-weighted highpass filtering property of the 2-D Teager filters has been exploited in developing a number of image processing applications
- We present next four specific applications

Contrast Enhancement

- The conceptually simple unsharp masking approach is a widely used image contrast enhancement method
- Based on the addition of an amplitude-scaled linear highpass filtered version of the image to the original image



Contrast Enhancement

- A commonly used linear highpass filter is the Laplacian operator:

$$y[n_1,n_2]=4x[n_1,n_2]-x[n_1-1,n_2]-x[n_1+1,n_2]$$
$$-x[n_1,n_2-1]-x[n_1,n_2+1]$$

- Its main advantage is computational simplicity

Contrast Enhancement

- The highpass filter enhances those portions of the image that contains mostly high frequency information, such as edges and textured regions
- Often yields visually pleasing results by utilizing an effect called **simultaneous contrast**
- The perceptual impression is improved because the image appears sharper and better defined

Contrast Enhancement

- Apparent problem of this technique is that it does not discriminate between actual image information and noise
- Thus, noise is enhanced as well
- Unfortunately, visible noise tends to be mostly in the medium to high frequency range

Contrast Enhancement

- The contrast sensitivity function (CSF) of the human visual system (HVS) shows that the eye (and the higher level processing system in the visual cortex) is less sensitive to low frequencies
- To eliminate the noise enhancement problem we need to make use of Weber’s law

Contrast Enhancement

- A visual phenomenon according to which the difference in the **perceived** brightness of neighboring regions depends on the sharpness of the transition occurring at edges

Contrast Enhancement

- We modify the unsharp masking method such that the image enhancement is dependent on the local average pixel intensity
- In bright regions we can enhance the image more because noise and other gray level fluctuations are much less visible

Contrast Enhancement

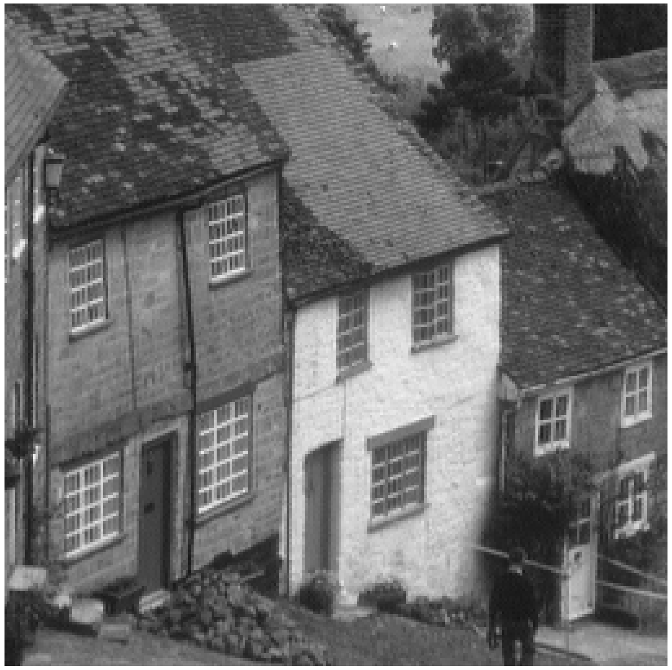
- On the other hand, in darker regions we want to suppress the enhancement process since it might deteriorate image quality
- This simple idea indicates the need for a highpass filter that depends on local mean:

$$H(e^{j\omega}) \propto H_{high}(e^{j\omega}) \cdot (local\,mean)$$

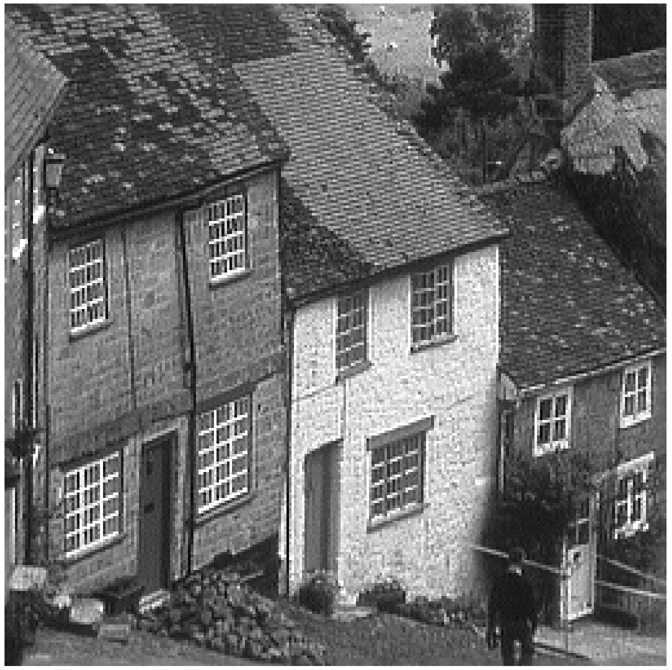
Contrast Enhancement

- Improvement in the visual quality of the image obtained using a **nonlinear unsharp masking approach** in which the linear highpass filter is replaced with a 2-D Teager operator
- The filter output depends on the local background brightness, and as a result, it follows **Weber's Law**

Contrast Enhancement



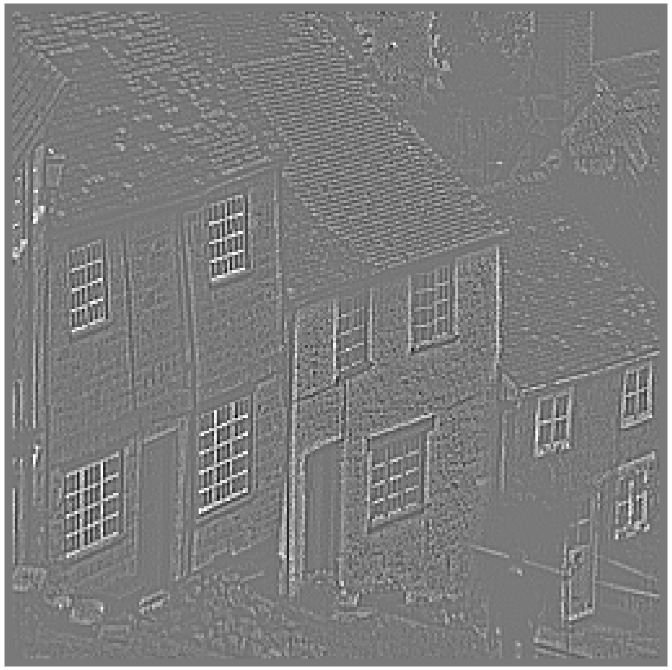
Original image

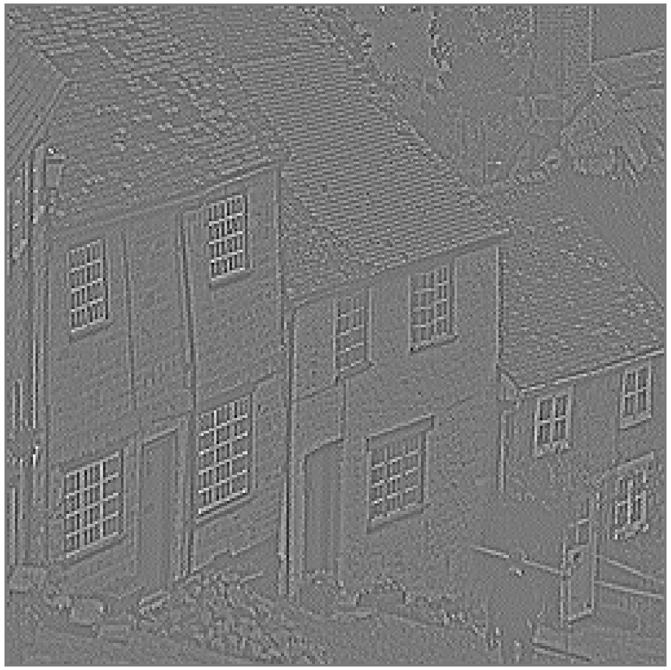


Enhanced image

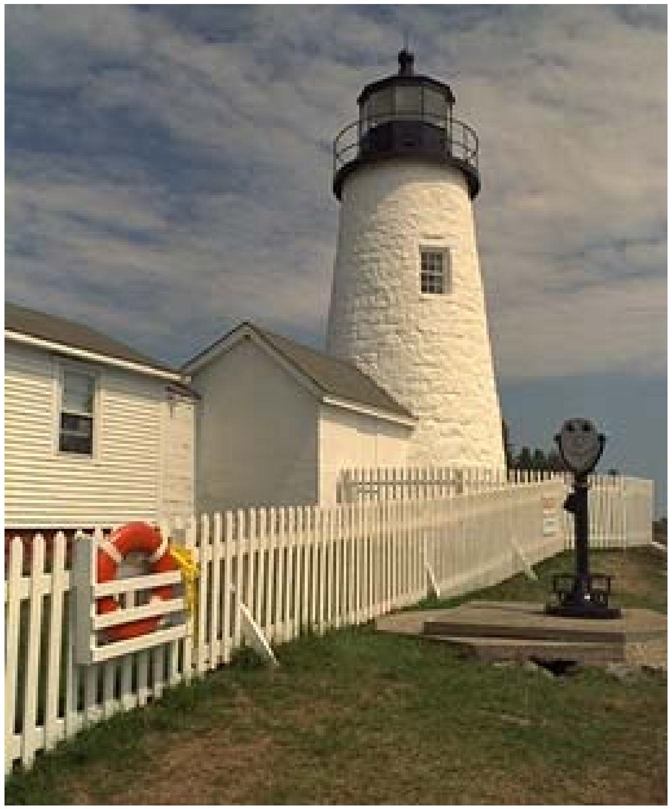
Contrast Enhancement

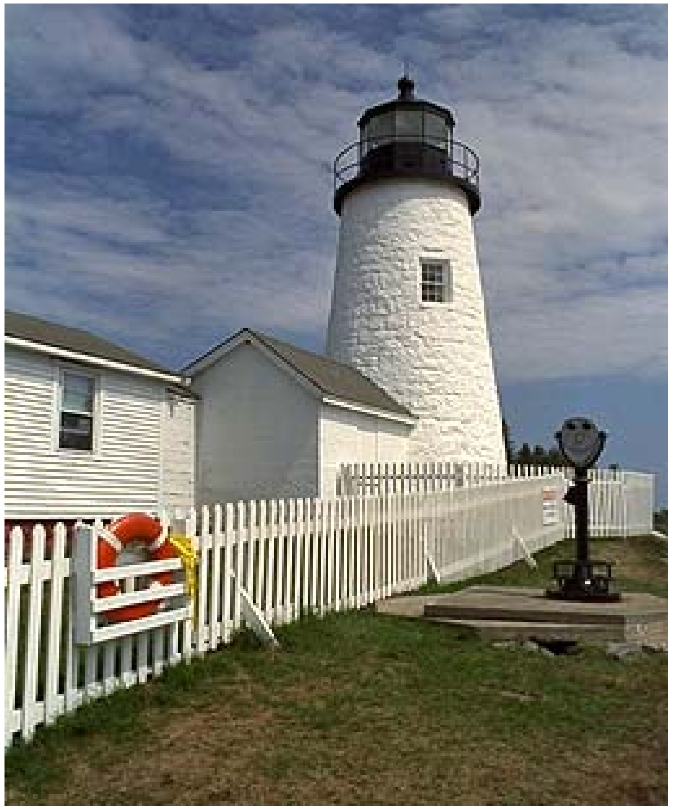
- Outputs of the Teager and the Laplacian filters are shown below


Teager filter output

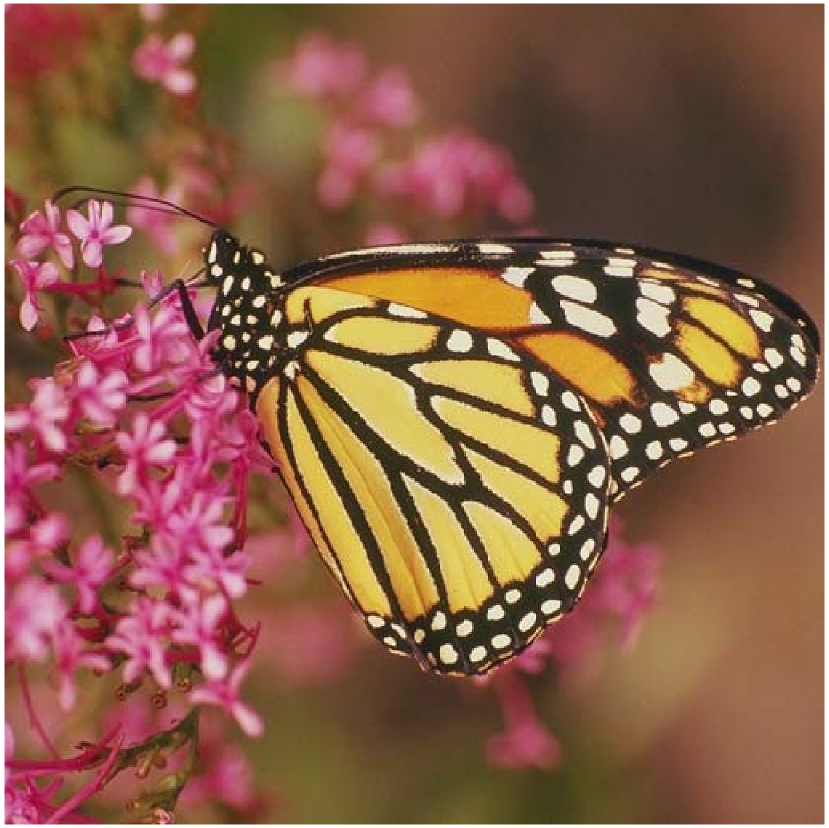

Laplacian filter output

Contrast Enhancement

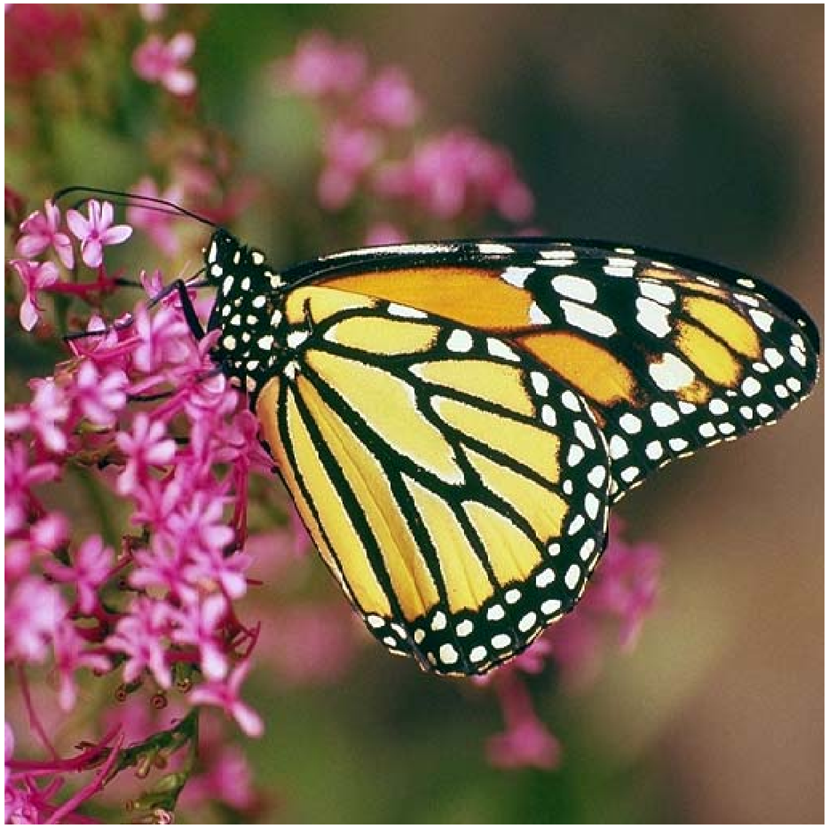

Original


Contrast Enhanced

Contrast Enhancement



Original



Contrast Enhanced

Contrast Enhancement

- The Laplcian filter output shows a uniform response to edges independent of background intensity
- The Teager filter output is weaker in darker regions (e.g., the darker areas of the roof) and stronger in brighter areas (e.g., the bright wall)

Impulse Noise Removal

- Goal: To suppress the impulse noise while preserving the edges and the details
- A number of nonlinear methods have been advanced for impulse noise removal
- Among these, the most common is the median filtering

Impulse Noise Removal

- Median filtering is computationally efficient and does suppress impulse-corrupted pixels effectively
- In median filtering, whether a pixel is corrupted by impulse noise or not, it is replaced by its local median within a window
- Thus, median filtering not only removes the impulse noise but also introduces distortion

Impulse Noise Removal

- A tradeoff needs to be made between the suppression of noise and the preservation of details and edges
- For effective noise suppression in highly corrupted images, median filtering with a large window is required
- Large window increases computational complexity while introducing unacceptable visible degradation in the filtered image

Impulse Noise Removal

- A detection-estimation-based approach has been developed to remove impulse noise from highly corrupted image while preserving edges and fine details
- First, a 2-D Teager operator is used to detect the locations of the impulse noise corrupted pixels
- Then a selectively chosen local mean operator is used to estimate the original value of the corrupted pixel

Impulse Noise Removal

- Let $x[m,n]$ denote the current pixel of an impulse-corrupted image with $y[m,n]$ denoting the output of the 2-D Teager operator
- If

$$y[m,n] > T$$

where T is a suitably chosen threshold value, then $x[m,n]$ is considered to be a pixel corrupted by a **positive impulse**

Impulse Noise Removal

- The corrupted is replaced by the average value of the uncorrupted pixels within the window (typically, 3×3), called the **selective local mean**
- To detect pixels corrupted by a negative impulse, a complement of the input image is first generated according to

$$x'[m,n]=B-x[m,n]$$

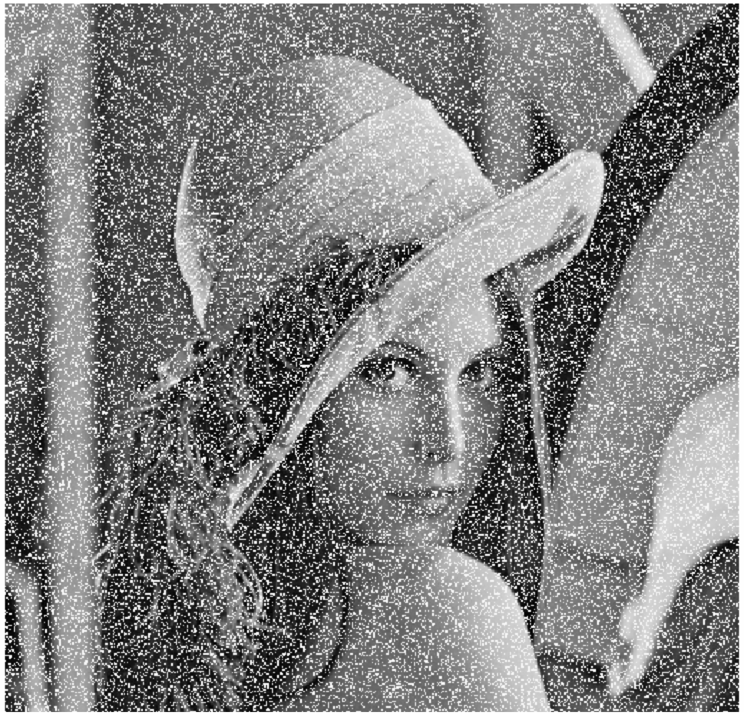
where B is the maximum gray value in the dynamic range

Impulse Noise Removal

- The Teager operator is next applied to detect the positive impulse corrupted in $x'[m,n]$
- The above method does work effectively in most cases
- Figure on next slide shows an original uncorrupted image and the noisy image corrupted with 20% positive impulse noise

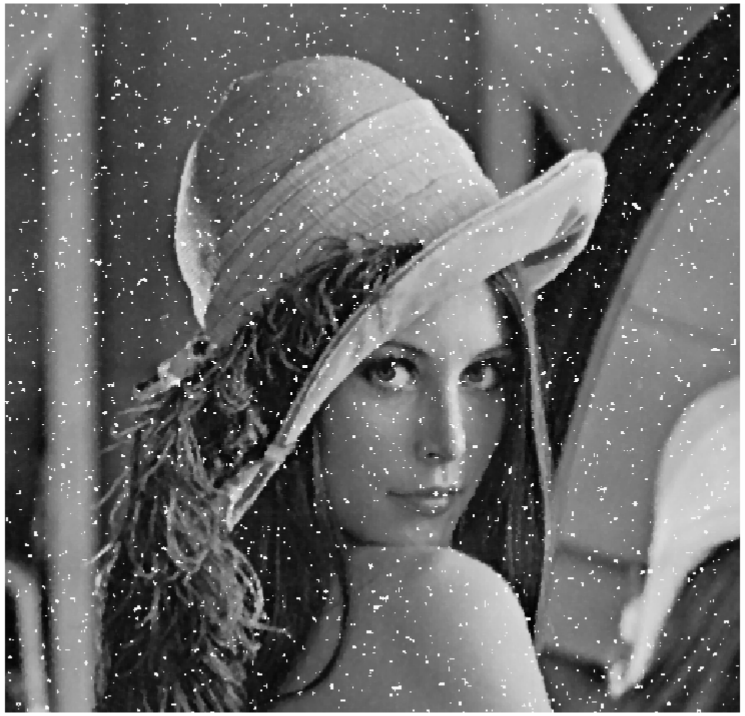
Impulse Noise Removal

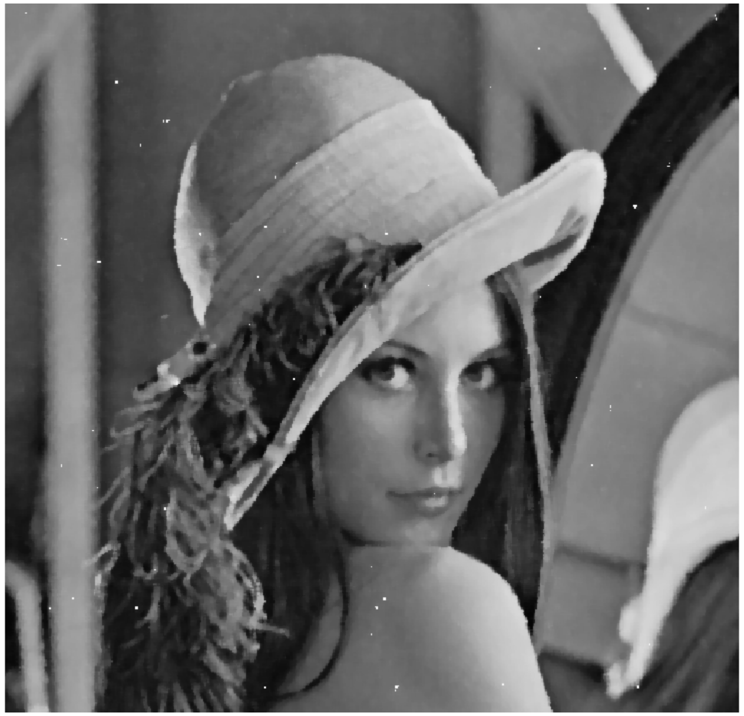
Original


Noise corrupted

Impulse Noise Removal

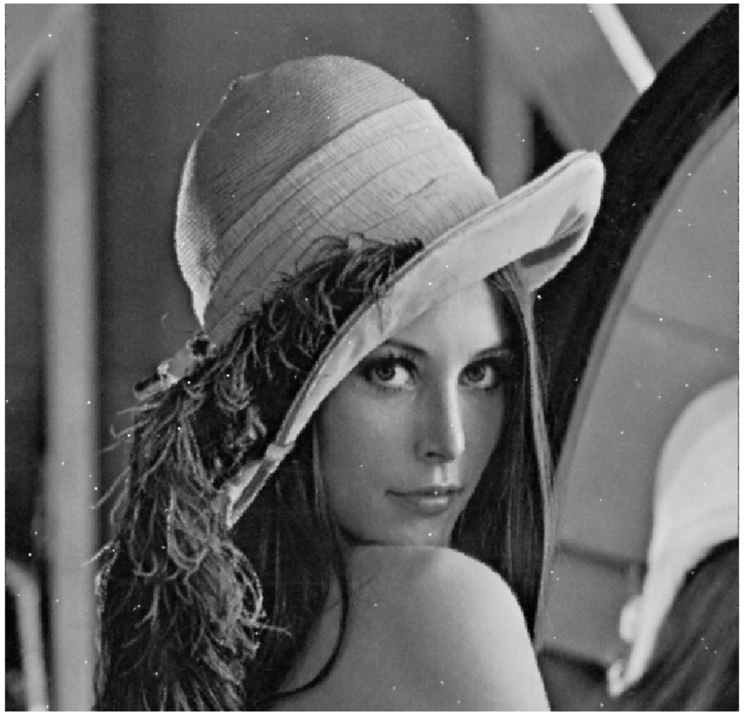
- Figures below shows the images obtained using median filters with a 3×3 window and a 5×5 window

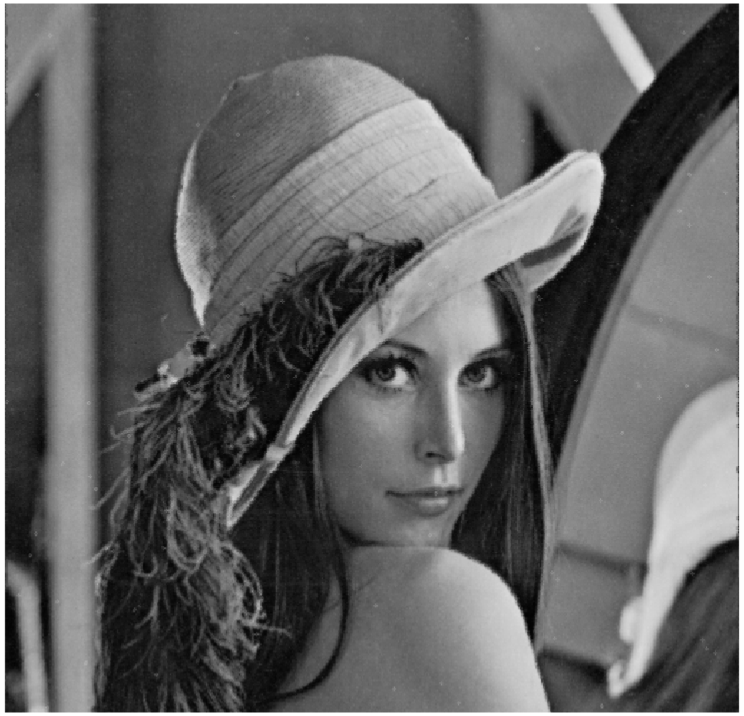

3×3 Median filter


5×5 Median filter

Impulse Noise Removal

- Figures below show the images obtained applying the Teager filter based methods


Teager filter


Two-pass Teager filter

Impulse Noise Removal

- There are two cases, where the 2-D Teager operator fails to detect the noisy pixels
- Case 1: When there is a group of impulse corrupted pixels matching the structure of the 2-D nonlinear operator, i.e., a crossing of the horizontal and vertical directions

Impulse Noise Removal

- Case 2: When the positive noisy pixels are located in the white areas, or negative noisy pixels are located in the dark areas
- To solve the problem with the first case, a joint-structure 2-D nonlinear operator has been employed

Impulse Noise Removal

- Here, to detect a positive impulse noise, the following nonlinear operator is used:

$$y[m,n] = \max\{y_1[m,n], y_2[m,n], y_3[m,n], y_4[m,n]\}$$

- In the above, $y_i[m,n]$, $i = 1, 2, 3, 4$, are the outputs of four different 2-D quadratic operators defined by

Impulse Noise Removal

$$y_1[m,n] = 2x^2[m,n] - x[m-1,n]\cdot x[m+1,n] \\ - x[m,n-1]\cdot x[m,n+1]$$

$$y_2[m,n] = 2x^2[m,n] - x[m-1,n-1]\cdot x[m+1,n+1] \\ - x[m+1,n-1]\cdot x[m-1,n+1]$$

$$y_3[m,n] = 2x^2[m,n] - x[m-2,n]\cdot x[m+2,n] \\ - x[m,n-2]\cdot x[m,n+2]$$

$$y_4[m,n] = 2x^2[m,n] \\ - x[m-2,n-2]\cdot x[m+2,n+2] \\ - x[m+2,n-2]\cdot x[m-2,n+2]$$

Impulse Noise Removal

- To solve the problem in the second case, the following nonlinear operator is used:

$$y_m[m,n] = (a\mu_x^2 + b\mu_x + c)\cdot y[m,n]$$

where μ_x is the normalized local mean defined within a 3×3 window and

$$y[m,n] = \max\{y_1[m,n], y_2[m,n], y_3[m,n], y_4[m,n]\}$$

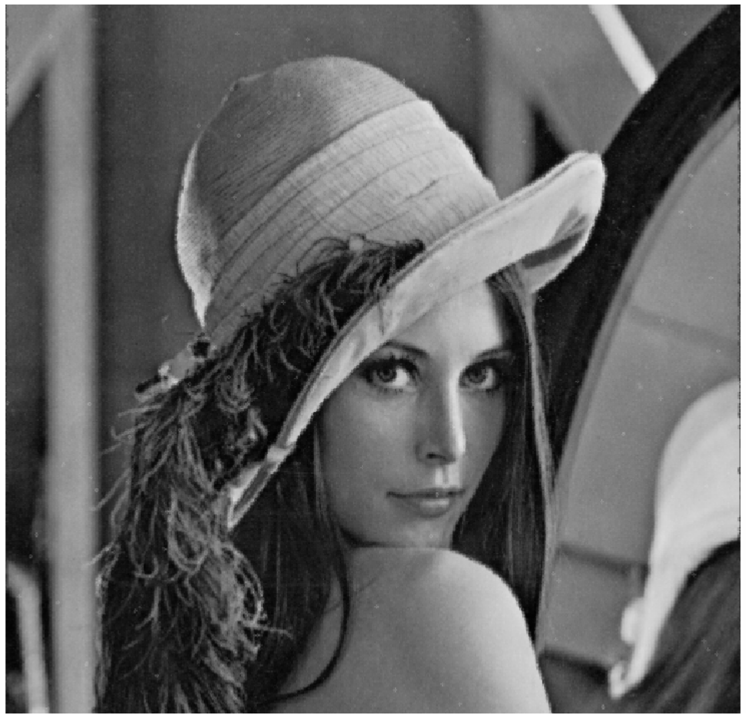
Impulse Noise Removal

- Noisy pixel corrupted by a positive impulse is detected by comparing the value of $y_m[m,n]$ with respect to a given threshold
- The modified operator can also be used to detect noisy pixels corrupted by a negative impulse by applying them to the complementary image $x'[m,n]$

Impulse Noise Removal

- Application of this modified approach has been found to provide improved performance in comparison to traditional median filtering-based methods
- Figure on next slide shows the image obtained using the improved detection-estimation method

Impulse Noise Removal



Improved Teager filter

Image Interpolation

- Image zooming is usually implemented in two steps
- First the image is up-sampled by an integer factor M, in both the horizontal and the vertical directions
- Up-sampling inserts $(M–1)$ zero-valued pixels among each consecutive pairs of pixels

Image Interpolation

- More appropriate values of these new pixels are obtained using some type of interpolation method in the second step
- Commonly used interpolation methods are the bilinear transformation or splines which tend to introduce artifacts in the zoomed version that degrade the visual quality of the image

Image Interpolation

- A more effective approach, based on the use of the 2-D Teager operators, adapts to local characteristics of the image while enhancing the quality
- The adaptive technique can better incorporate properties of human visual system (HVS) and yields more pleasing results

Image Interpolation

- Figure below shows the block diagram of the edge-enhanced zooming method

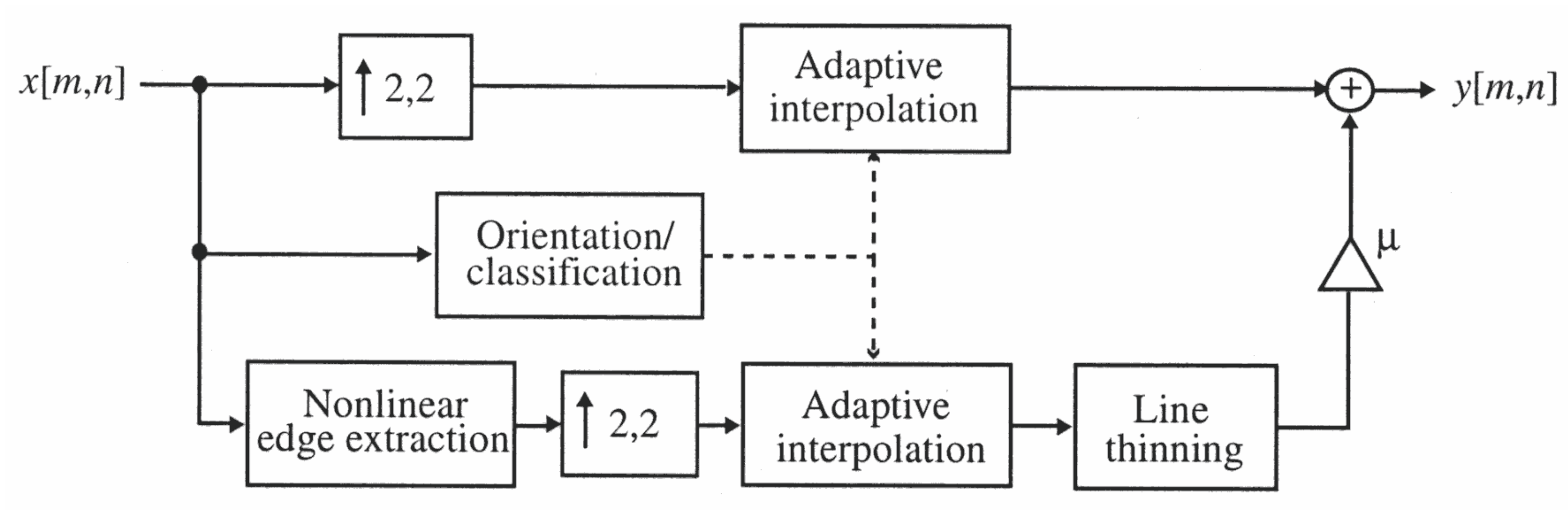


Image Interpolation

- In the top branch, the input image is first up-sampled by a factor of 2 in both horizontal and vertical directions
- Then, the missing samples are found using an adaptive interpolation
- Bottom branch extracts perceptually important edge and texture information using the quadratic Volterra filter

Image Interpolation

- The quadratic Voletrra filter used is

$$y[m,n]=3x^2[m,n]-\frac{1}{2}x[m+1,n+1]x[m-1,n-1]$$
$$-\frac{1}{2}x[m+1,n-1]x[m-1,n+1]$$
$$-x[m+1,n]x[m-1,n]-x[m,n+1]x[m,n-1]$$

- As mentioned earlier, this filter extracts and enhances fewer edges in the darker portion of the image

Image Interpolation

- Also, noise is amplified to a lesser degree in these darker areas
- An important issue, because due to Weber’s law, noise is more visible in the darker areas than in the bright portions
- As a result, edges in the zoomed image are enhanced without generating perceptually significant noise

Image Interpolation

- Note that the overall structure follows the simple idea of unsharp masking
- It yields a sharper output image, because the high-frequency components are emphasized by adding a fraction of the lower branch output back to the zoomed image

Image Interpolation

- The proposed method adapts to local edge and texture orientation and uses several interpolation filters instead of only one
- We consider zooming by factors of 2 in each direction
- Method can be extended to other factors

Image Interpolation

- Consider the figure below where the filled circles represent original pixels and the empty circles represent the zero-valued pixels obtained after up-sampling

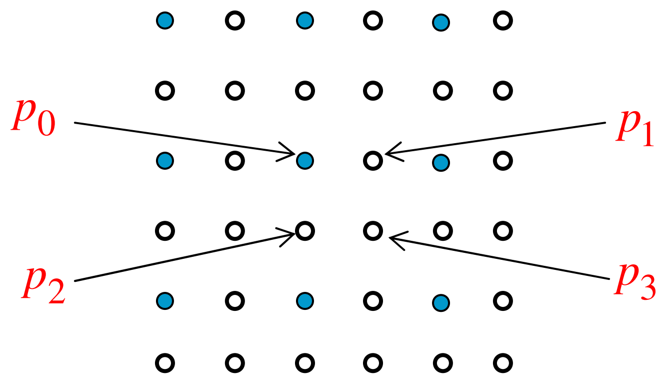


Image Interpolation

- Objective of interpolation is to replace these zero-valued samples with appropriate values
- For example, we must estimate the values of pixels p_1, p_2, and p_3 from the information given in the local neighborhood

Image Interpolation

- The local neighborhood of p_0 is classified into one of 3 categories: constant, oriented or irregular
- “Constant” means without clear features
- “Oriented” block shows a prominent orientation like edges or directional patterns or texture

Image Interpolation

- "Irregular" are those that exhibit structure without clear edge direction or features that are too small to make up an oriented area
- With this classification we control both the interpolation of the extracted edges and the original image, but use only the pixels in the original image to find the proper classification for each pixel and its neighborhood

Image Interpolation

- **Classification Criteria**
- A neighborhood is classified as constant, if there are no sharp features like edges or corners, i.e., it is essentially a flat portion of the image
- Therefore, the local gray-level difference must be below a certain threshold

Image Interpolation

- Mathematically, the pixels in the region satisfy the condition

$$\ell_{\max} - \ell_{\min} < T_1(g_{\max} - g_{\min})$$

- In the above $\ell_{\max}$ and $\ell_{\min}$ are the local maximum and maximum, $g_{\max}$ and $g_{\max}$ are the global extremes, and T_1 is a threshold controlling the classification with a value between 0 and 1

Image Interpolation

- A reasonable value of T_1 has been found to be 0.1
- For the other two cases, we first determine whether a certain orientation is dominant in the neighborhood or not
- We compute the gradient at all pixel locations within a 4×4 block by applying the Sobel operator resulting in 16 gradients

Image Interpolation

- Every gradient's angle is then quantized to one of 12 possible directions as shown below

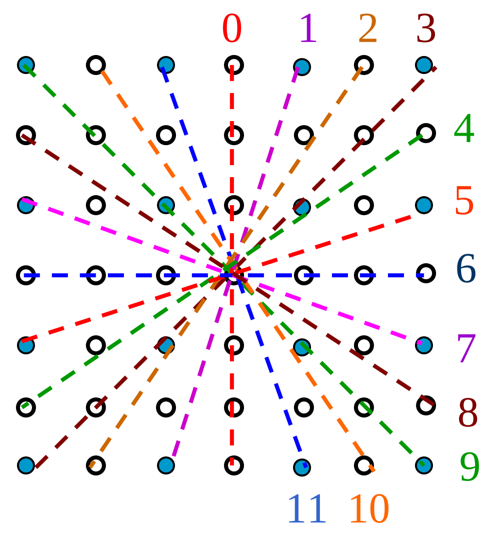


Image Interpolation

- The choice of 4×4 blocks yields a directional resolution of about 15 degrees and has been found to be sufficient
- If more than 50% of the 16 gradients point in the same direction or in two adjacent directions, we assume that this orientation is dominant and the neighborhood is classified as oriented

Image Interpolation

- The values of the missing samples are then determined by interpolating along the dominant direction
- For example, p_2 is determined using a linear combination of the adjacent known pixels shown by the green arrows

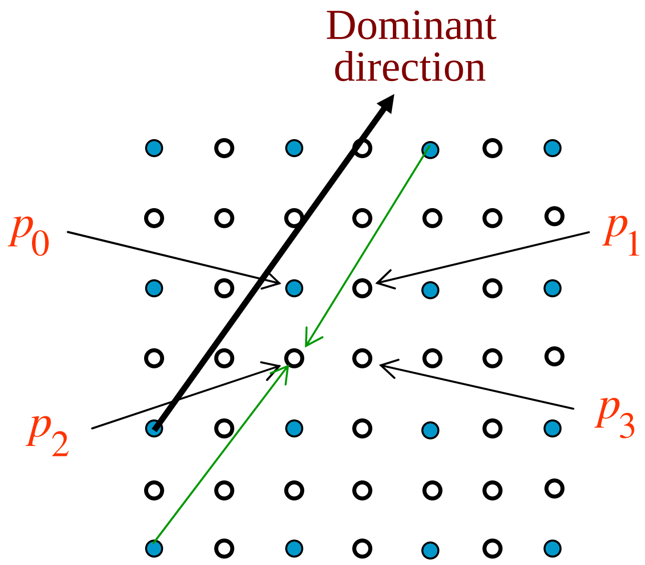


Image Interpolation

- Likewise, p_1 is determined using a linear combination of the adjacent known pixels shown by the purple solid and dashed arrows

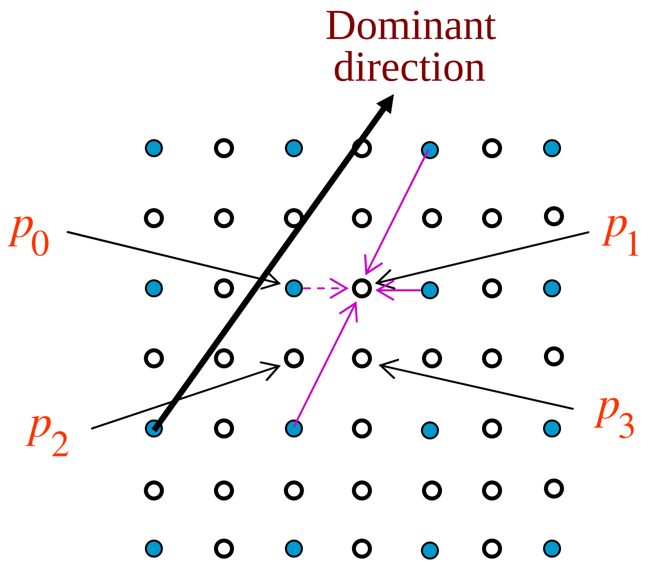


Image Interpolation

- Similarly, p_3 is determined using a linear combination of the adjacent known pixels shown by the blue solid and dashed arrows

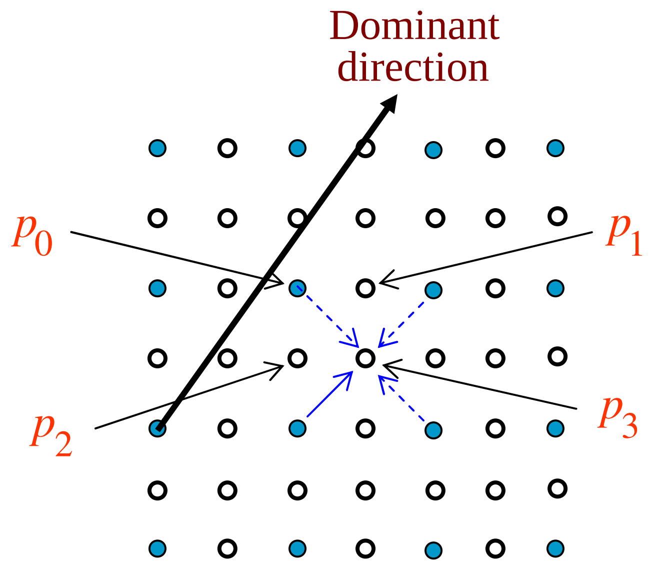


Image Interpolation

- The process is equivalent to a low-pass interpolation with a kernel shown below

$$\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0.6 \\ 0 & 0 & 0 & 0.6 & 0 \\ 0 & -0.1 & 0 & 0.6 & 0 \\ 0 & -0.1 & \boxed{1} & -0.1 & 0 \\ 0 & 0.6 & 0 & -0.1 & 0 \\ 0 & 0.6 & 0 & 0 & 0 \\ 0.6 & 0 & 0 & 0 & 0 \end{array} \quad \xleftarrow{} \text{Origin of kernel}$$

Image Interpolation

- For each of the 12 orientation cases, a different kernel is used
- With this adaptation, interpolation is carried out parallel to edges and not across them
- This results in the desired smoothness along the edge and sharpness across

Image Interpolation

- The neighborhood is classified as irregular if no dominant orientation can be found
- Here, the mean of the neighboring pixels is used to estimate p_1 and p_2
- For p_3 , however, a directional linear interpolation is used as shown in the next slide

Image Interpolation

- In this case, the average of the surrounding four shaded pixels is used with weighting factors determined by the distance of these pixels from the arrow through

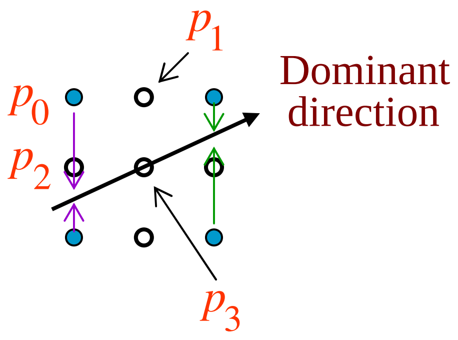


Image Interpolation

Thinning of Zoomed Edges

- An edge in the original edge image is represented by a thin line
- In the zooming process, this line gets thickened while remaining sharp
- In the output image, these thick lines should be thinned so that they enhance the exact position of the edge when added back

Image Interpolation

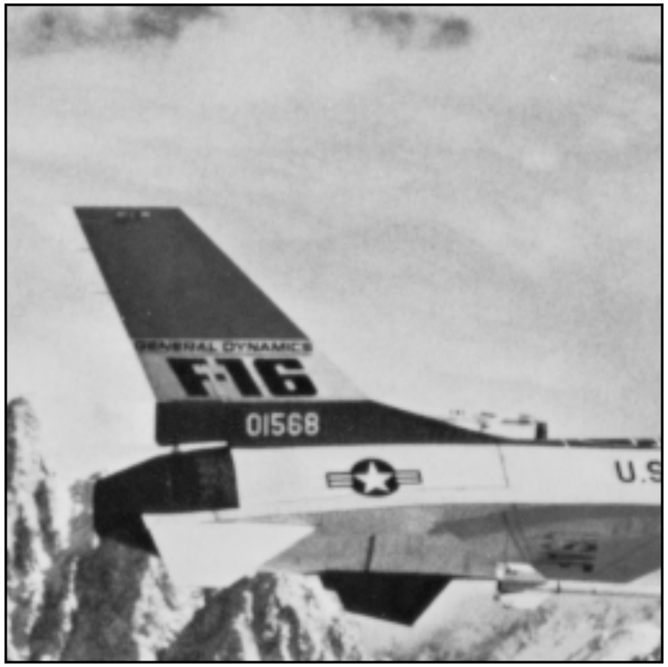
- Thinning operation computes the magnitude of the gradient at each pixel
- If the gradient magnitude is above a certain threshold, the gray level is divided by the square root of the magnitude
- This reduces the amplitude only in the steep areas and leaves flat areas unchanged

Image Interpolation

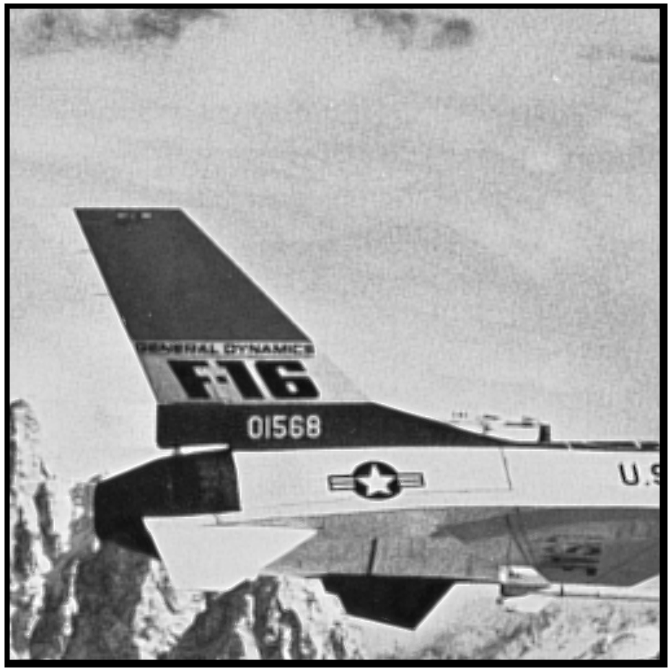
- In experimental comparisons, the adaptive interpolation method has been found to compare favorably with traditional zooming methods in terms of the preservation of edges and overall perceptual quality

Image Interpolation

- Figures below show images zoomed using the bilinear interpolation method, and the edge-enhanced zooming method



Bilinear method



Edge-enhanced method

Image Halftoning

- General problem in printing of images: Gray scale resolution must be adapted to output device (often 1 bpp for gray level images)
- Compensation possible by increasing spatial resolution: HVS perceives local averages (lowpass filter) ➡ Halftoned images can still invoke impression of gray levels

Image Halftoning

Standard technique for b/w halftoning:

- Random dithering
- Ordered dithering (clustered and dispersed)
- Error diffusion

Image Halftoning

Dithering

- Essentially thresholding operations with either random or fixed thresholds
- Advantages – Computational simplicity
- Disadvantages - Details are lost, images are blurred

Image Halftoning

Error Diffusion

- Preferred choice for printing
- Advantages - Preserves details much better and leads to higher quality prints
- Disadvantages - Blurring of details due to the diffusion process
- Disadvantages – Higher complexity

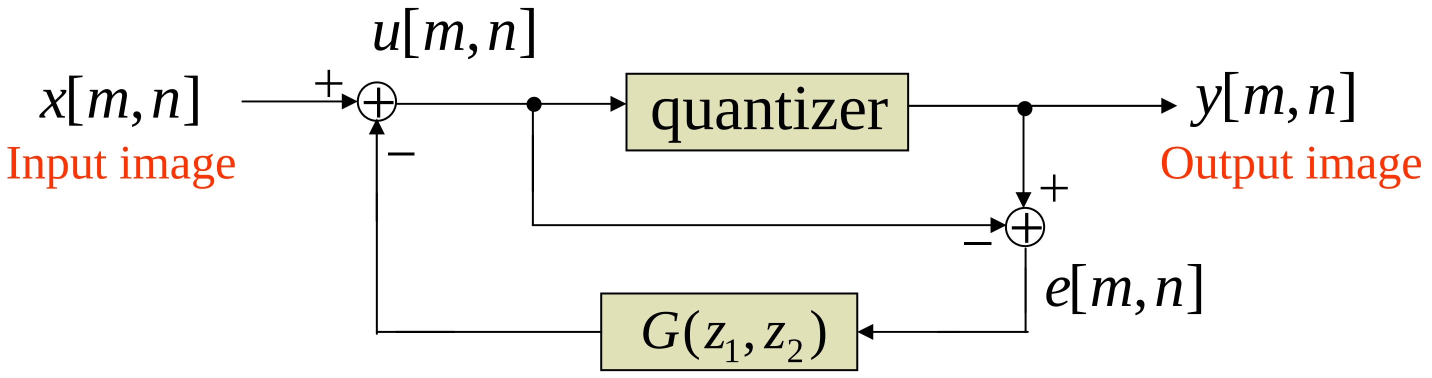
Image Halftoning

Standard Error Diffusion Method

- Basic idea: Use fixed quantizer and spread quantization error out over local neighborhood
- Enables neighboring pixels to compensate each other partially

Image Halftoning

- Block diagram shown below



- Input and output image size typically $0 \leq m \leq M$ and $0 \leq n \leq N$

Image Halftoning

- Quantizer is usually a single thresholding operation
- Multilevel quantizer can be used for a multilevel error diffusion
- Quantizer error $e[m,n]$ is filtered through a lowpass filter $G(z_1,z_2)$ and then subtracted from the original input image to yield $u[m,n]$

Image Halftoning

- If $e[m,n] > 0$, i.e., when the quantizer output is larger than its input, $u[m,n]$ is smaller than the original pixel and quantization of $u[m,n]$ is more likely to yield zero
- $\Longrightarrow$ Some pixels are mapped to larger values and some to smaller values, and they partially compensate each other's quantization error

Image Halftoning

- The kernel of the lowpass filter used is given by

$$g[m,n] = \frac{1}{48}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \boxed{0} & 7 & 5 \\ 3 & 5 & 7 & 5 & 3 \\ 1 & 3 & 5 & 3 & 1 \end{bmatrix}$$

- Box indicates origin

Image Halftoning

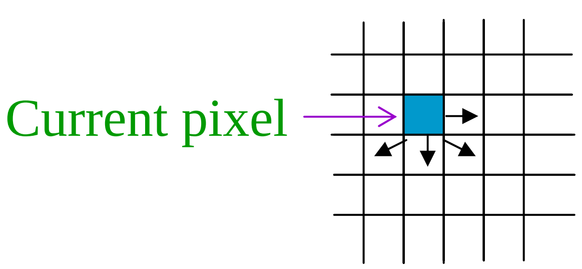
- To preserve the local gray-value impression, the local averages of the half-toned image and the original image must be equal

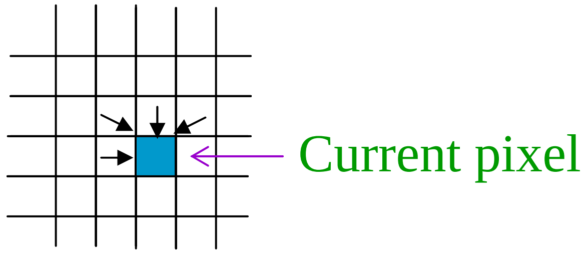
$$\Rightarrow \quad G(\omega_1, \omega_2)\big|_{\omega_1=\omega_2=0} = \sum_m \sum_n g[m,n] = 1$$

- Above requirement can always be implemented by using an appropriate scale factor

Image Halftoning

- Pictorial way of looking at diffusion process:





Quantization error *e*(*m*,*n*) distributed to its neighbors

Pixel *x*(*m*,*n*) is corrected by a linear combination of weighted quantization Errors from neighbors

Image Halftoning

- Pixel is corrected before quantization
- Error diffusion blurs fine details and edges by spreading the error out over neighborhood (e.g. 5×5)
- Feedback loop and the subtraction of the diffused error leads to a highpass operation
- Process enhances the edges of the image

Image Halftoning

- Even though individual pixels are usually too small to be noticeable in halftoned image, the overall contrast and sharpness are reduced by this effect and the overall halftoned image quality suffers

Image Halftoning

- Improved error diffusion should only diffuse error to pixels on the same side of an edge
- A simple, yet effective, modification of the standard error diffusion technique, has improved significantly the quality

Image Halftoning

- The modified method employs a 2-D Teager filter to reduce the diffusion of the error across edges and details
- Basic idea: Adapt diffusion filter $G(z_1, z_2)$ to local image characteristics
- This has improved the separation of areas on both sides of an edge

Image Halftoning

- Figure below shows the block diagram of the modified error diffusion method

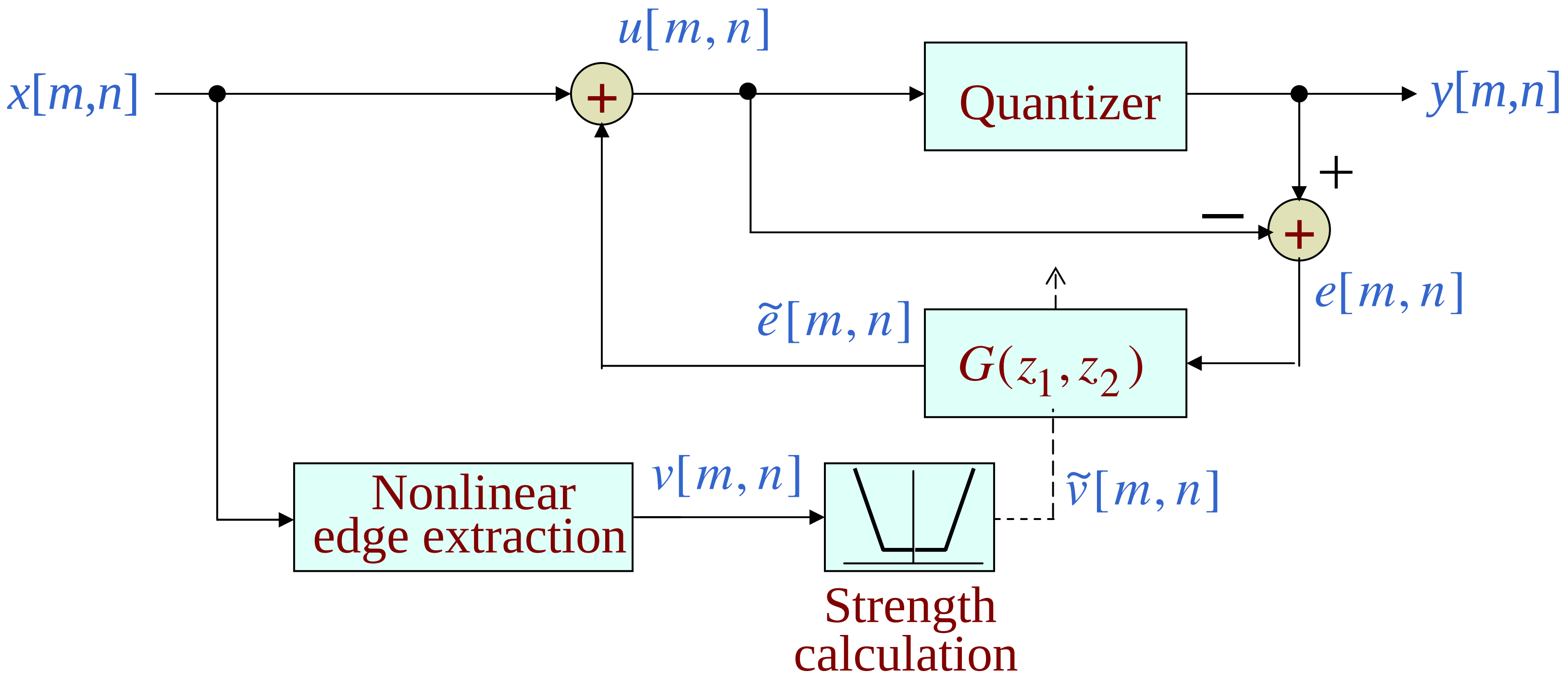


Image Halftoning

- Note $v[m,n]$ can have both negative and positive values
- Only the positive ones are employed for adaptation of diffusion filter $G(z_1,z_2)$
- This is achieved by passing $v[m,n]$ through a block with an input-output relation

$$\tilde{v}[m,n] = \begin{cases} |\rho \cdot v[m,n]|, & \text{if } |\rho \cdot v[m,n]| > 1 \\ 1, & \text{otherwise} \end{cases}$$

Image Halftoning

- Values less than 1 will be lost, i.e. mapped to 1
- Does not lead to any actual loss of information as edges are typically represented by values several orders of magnitude larger
- A value $\rho = 0.00025$ yielded best results

Image Halftoning

- Diffusion filter adapted by dividing the filter coefficients by the corresponding value of $\tilde{v}[m,n]$:

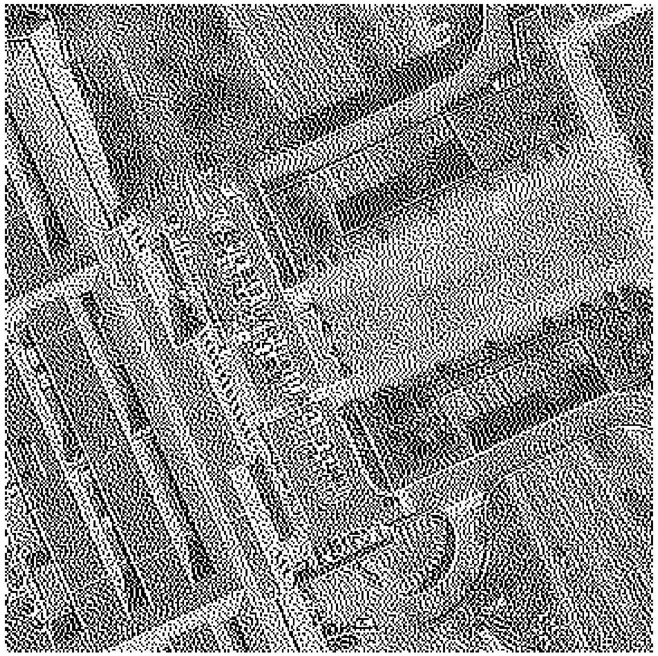
$$\tilde{e}[m,n] = \sum_k \sum_\ell e[m-k,n-\ell] \frac{g[k,\ell]}{\tilde{v}[m-k,n-\ell]}$$

Image Halftoning

- Influence of error pixel $e[m,n]$ on its adjacent pixels is reduced whenever it lies on or close to an edge
- Leads to an above-average number of white pixels on the brighter side of the edge, and an above-average black pixels on the darker side
- Thus, the edge is better defined and appears less fuzzy

Image Halftoning

- Figures below shows the halftoned image obtained using the original and the modified approaches


Original method

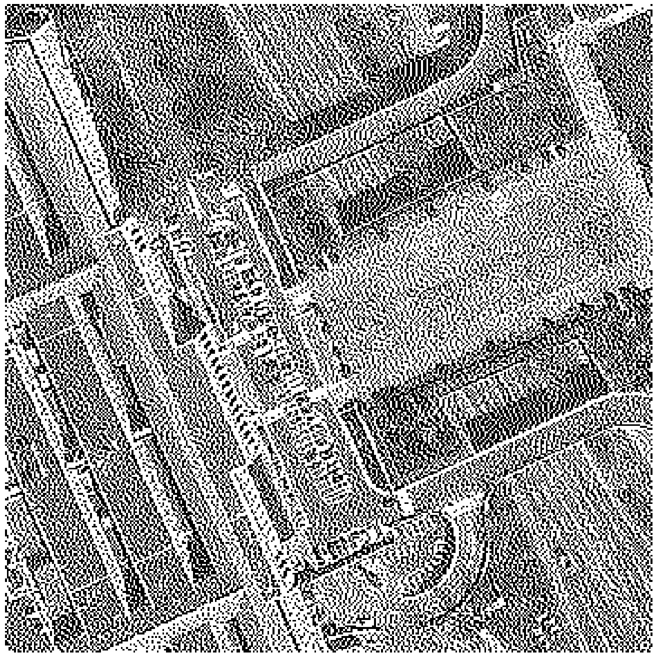

Modified method

Image Halftoning

- It is seen to be sharper with improved contrast compared to that obtained using the standard error diffusion method

Concluding Remarks

- The 2-D Teager filter, a special type of the quadratic Volterra filter, has a number of interesting properties
- These properties have been exploited to develop improved image processing algorithms, such as contrast enhancement, image zooming, impulse noise removal, and image halftoning

Concluding Remarks

- In these applications, the processed images appear perceptually much better in overall quality than those obtained using some well-known methods
- The operators are also computationally quite attractive requiring very few multiplications and additions

The End