Noise Reduction

Earl Wong

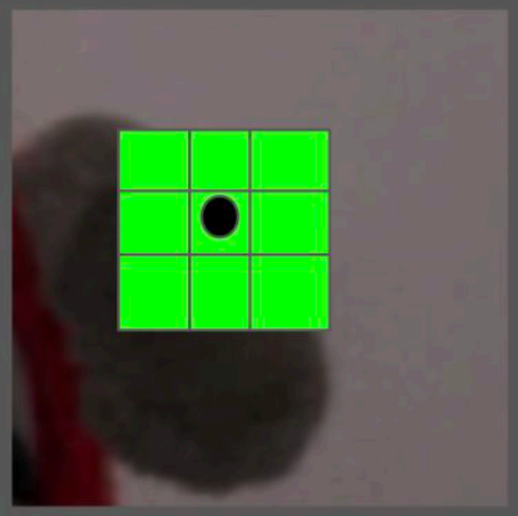
Noise Reduction Algorithms

- Non Local Means
- BM3D
- Wavelet
- Wiener
- Bilateral
- Diffusion
- Median

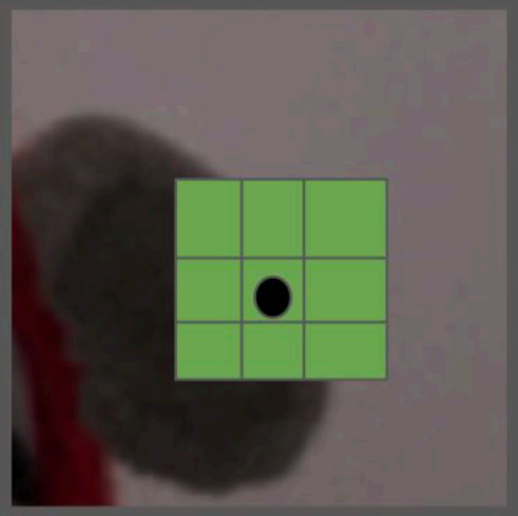
Non Local Means

- The Non Local Means algorithm exploits self similarity.
- For a given reference patch, the algorithm searches for similar patches in a surrounding neighborhood.
- Dis-similar patches are rejected (or given less weight) while similar patches are combined (or given more weight).
- By construction, Non Local Means is a “natural” denoising filter, since weighted average de-noising is performed.

Non Local Means



Reference Patch



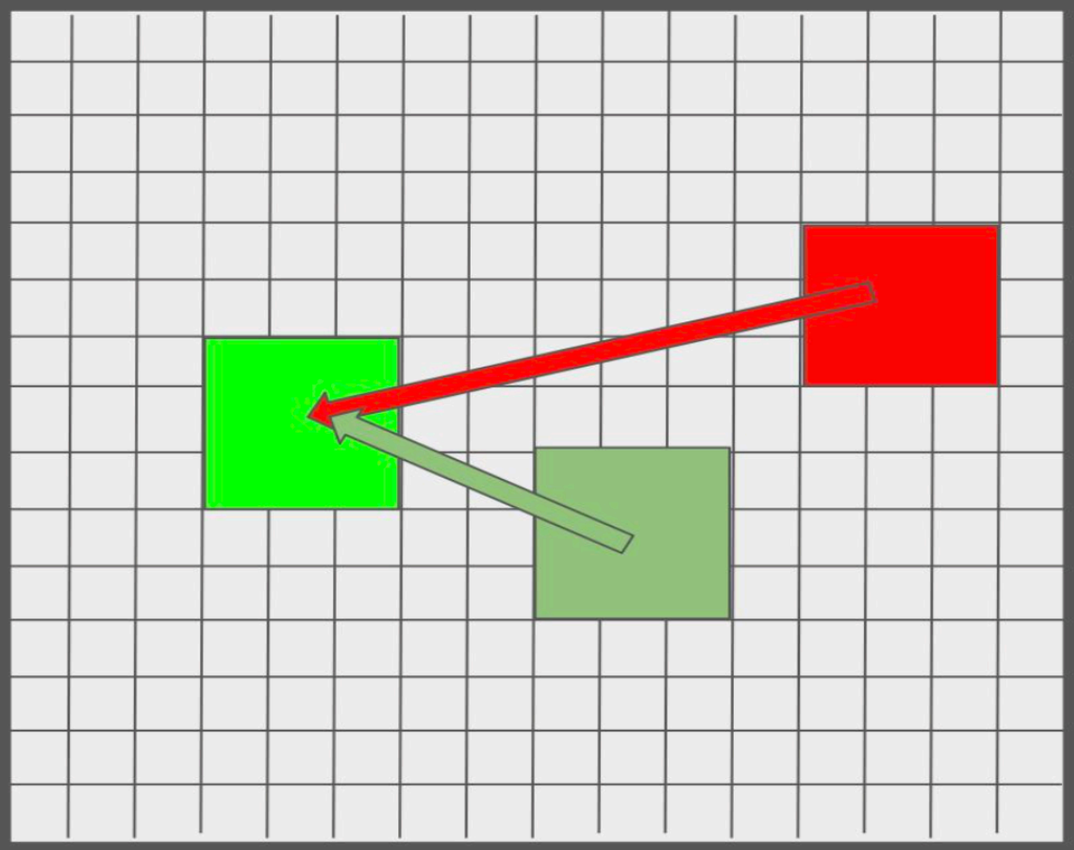
Search Patch A

Search Patch Z

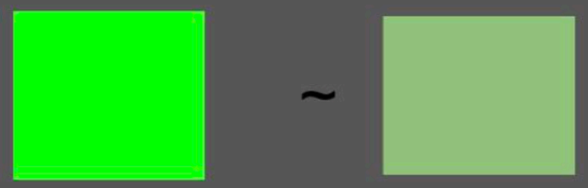
Non Local Means

Engineering One Pager

(MPEG-2, 4, H264, 265 codecs use the same concept to find the best matching patch)
(openGL ES 3.1 provides a new mechanism to perform this same operation.)



Ref Search Patch A' Weight



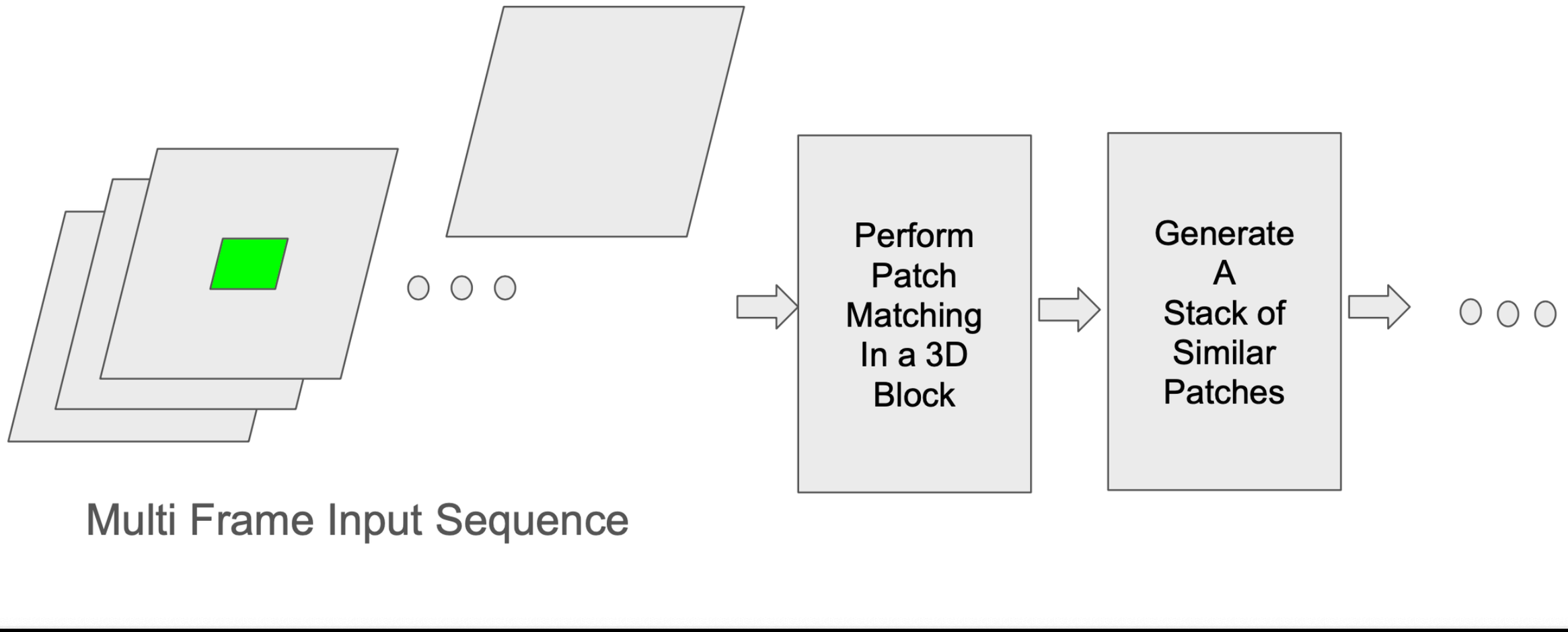
Ref Search Patch Z' Weight

New Pixel Value = ((0.9) * SP A' + …. + …. (0.0) * SP Z')) / NormalizationTerm

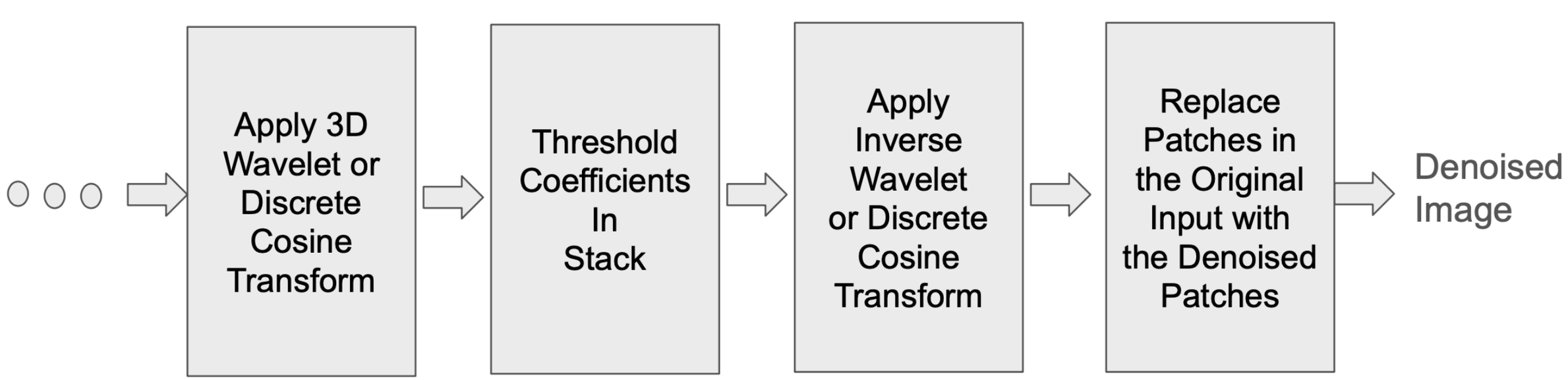
BM3D

- BM3D is a more powerful denoising algorithm, relative to its predecessor, Non Local Means.
- Like Non Local Means, a search for similar patches is performed.
- Unlike Non Local Means, once the similar patches are found, they are then aggregated into a 3D stack.
- A 3D transform - DCT or DWT is applied to the 3D stack, generating a 3D stack of “coefficients”.
- A thresholding algorithm is applied to the coefficients.
- This is followed by an inverse 3D transform, generating the de-noised patches.

BM3D

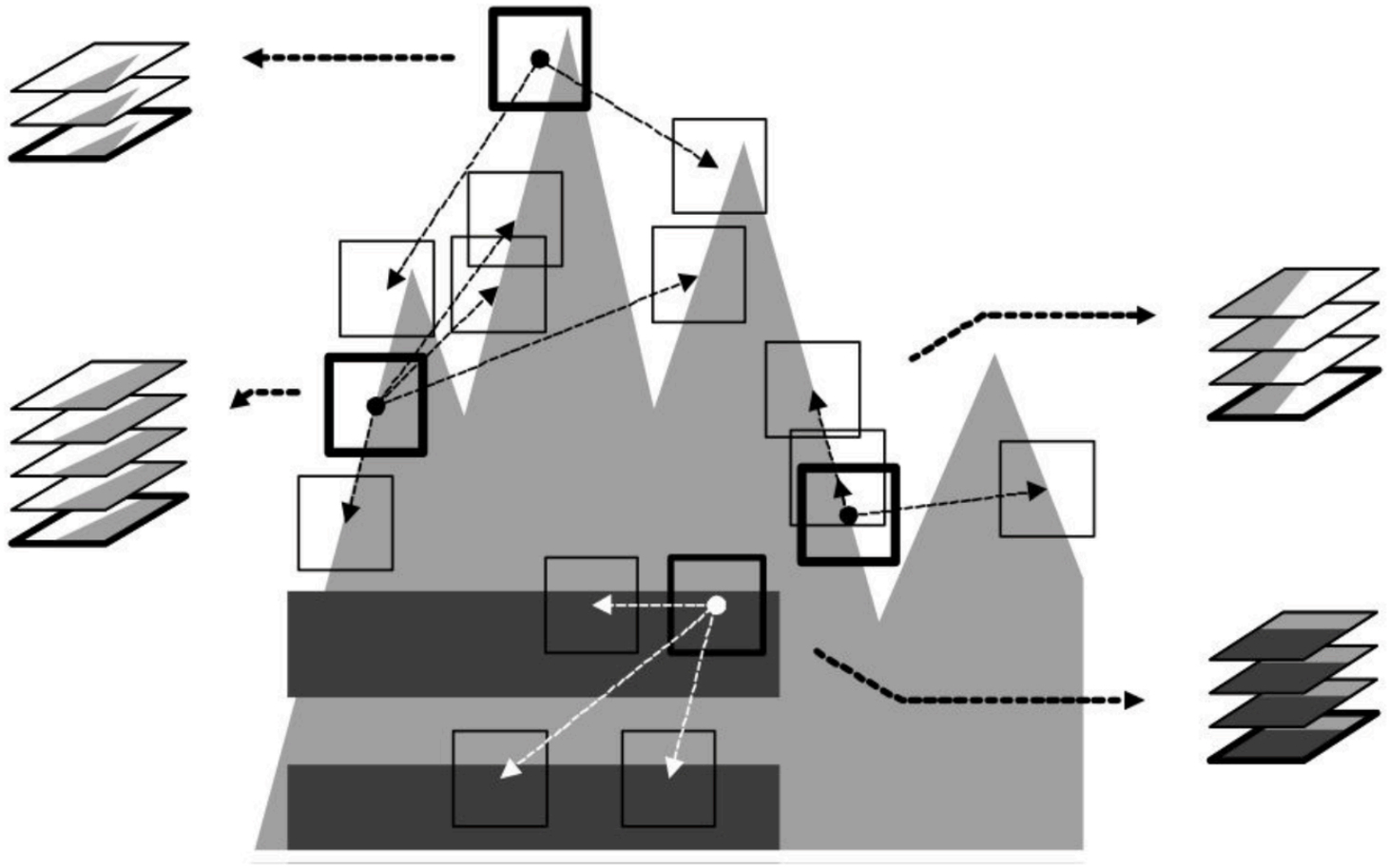

Perform Patch Matching In a 3D Block
Generate A Stack of Similar Patches
Multi Frame Input Sequence

BM3D


Apply 3D Wavelet or Discrete Cosine Transform
Threshold Coefficients In Stack
Apply Inverse Wavelet or Discrete Cosine Transform
Replace Patches in the Original Input with the Denoised Patches
Denoised Image

BM3D

Generate
A
Stack of
Similar
Patches



Wavelet

- A wavelet transform is applied to the image.
- The wavelet transform produces a sparse representation of the image.
- Thresholding is then applied, accepting or rejecting different coefficients.
- The inverse wavelet transform is then performed, producing the de-noised image.

Wavelet

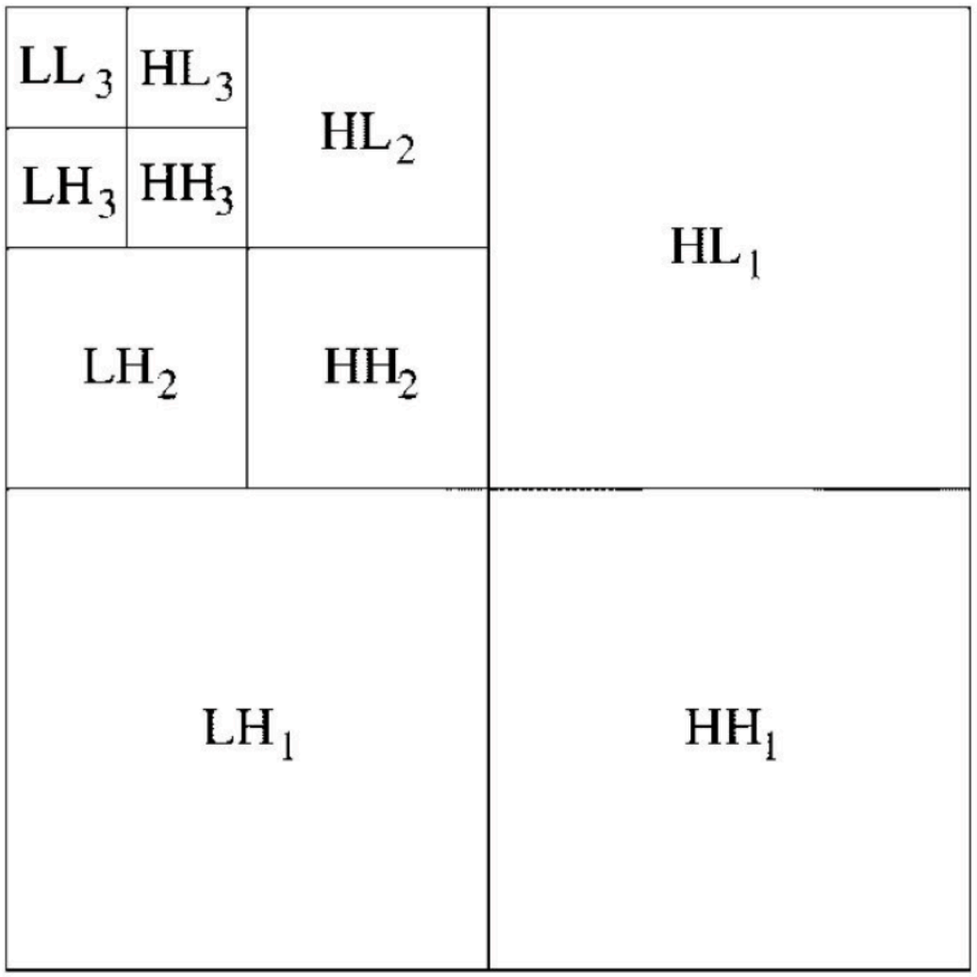
Apply Wavelet Transform

Threshold Resulting Coefficients

Apply Inverse Wavelet Transform

Denoised Image

Wavelet



In practice, the wavelet transform produces a sub band decomposition.

The picture to the left represents a 3 level sub band decomposition.

Thresholding is applied to the coefficients, before applying the inverse wavelet transform.

Wavelet

Two 1D filters are defined.

A **low-pass filter** $g[n]$ (scaling function)

A **high-pass filter** $h[n]$ (wavelet function)

There are many candidate filters.

Some of the more popular ones include the Haar filter and variations of the Daubechies filter.

Wavelet

Haar Filter

Low-pass: $g = \frac{1}{\sqrt{2}}[1, 1]$

High-pass: $h = \frac{1}{\sqrt{2}}[1, -1]$

DB2 Filter

$$g = [0.48296, 0.83652, 0.22414, -0.12941]$$

$$h = [-g_3, g_2, -g_1, g_0]$$

DB4 Filter

$$g = [0.48296, 0.83652, 0.22414, -0.12941, -0.07576, 0.03223, 0.00047, -0.00567]$$

$$h = [-g_7, g_6, -g_5, g_4, -g_3, g_2, -g_1, g_0]$$

Wavelet

Each row in an image is convolved with these filters, producing:

$$L_r(x,y) = \sum_n f(x,n) \cdot g(n-2y)$$

$$H_r(x,y) = \sum_n f(x,n) \cdot h(n-2y)$$

Critical downsampling occurs during the process:

$$L_r(x,y) = L_r(x,2y)$$

$$H_r(x,y) = H_r(x,2y)$$

Wavelet

Next, the same filters are now applied in the column direction:

$$LL(x,y) = \sum_m L_r(m,y) \cdot g(m-2x)$$

$$LH(x,y) = \sum_m L_r(m,y) \cdot h(m-2x)$$

$$HL(x,y) = \sum_m H_r(m,y) \cdot g(m-2x)$$

$$HH(x,y) = \sum_m H_r(m,y) \cdot h(m-2x)$$

Wavelet

This produces the following

decomposition:

LL LH

HL HH

LL, LH, HL, and HH subbands of Lena image obtained by using DWT.

Wavelet

Hard of soft threshold filters are then applied to the filter coefficients. [Many other filters exist.]

$$T_h(x) = \begin{cases} x, & \text{if } |x| \geq T \\ 0, & \text{if } |x| < T \end{cases}$$

$$T_s(x) = \begin{cases} \text{sign}(x) \cdot (|x| - T), & \text{if } |x| \geq T \\ 0, & \text{if } |x| < T \end{cases}$$

Wiener

- The Wiener filter originates from 1D signal processing theory (Norbert Wiener).

- Given a 1D signal corrupted by additive noise, the Wiener filter produces the optimal Minimum Mean Square Estimate (MMSE) between the corrupted and the de-noised signal.

The optimality result is valid under the following assumptions:

- 1) The signal and the noise are stationary. (i.e. The statistics are not time varying.)

- 2) The signal and the noise are gaussian.

- 3) The signal and the noise are uncorrelated. (i.e. The covariance between the signal and the noise is zero.)

Wiener

Apply FFT

Filter With G(u,v)

Apply Inverse FFT

Denoised Image

FFT is an NlogN operation. It computationally intensive, even with the use of an optimized library.

Wiener

Given a system:

$$y(t) = (h * x)(t) + n(t)$$

where $*$ denotes convolution and:

- $x(t)$ is some original signal (unknown) at time t.
- $h(t)$ is the known impulse response of a linear time-invariant system
- $n(t)$ is some unknown additive noise, independent of $x(t)$
- $y(t)$ is our observed signal

Our goal is to find some $g(t)$ so that we can estimate $x(t)$ as follows:

$$\hat{x}(t) = (g * y)(t)$$

where $\hat{x}(t)$ is an estimate of $x(t)$ that minimizes the mean square error

$$\epsilon(t) = \mathbb{E}|x(t) - \hat{x}(t)|^2 ,$$

with $\mathbb{E}$ denoting the expectation. The Wiener deconvolution filter provides such a $g(t)$. The filter is most easily described in the frequency domain:

$$G(f) = \frac{H^*(f)S(f)}{|H(f)|^2 S(f) + N(f)}$$

where:

- $G(f)$ and $H(f)$ are the Fourier transforms of $g(t)$ and $h(t)$,
- $S(f) = \mathbb{E}|X(f)|^2$ is the mean power spectral density of the original signal $x(t)$,
- $N(f) = \mathbb{E}|V(f)|^2$ is the mean power spectral density of the noise $n(t)$,
- $X(f)$, $Y(f)$, and $V(f)$ are the Fourier transforms of $x(t)$, and $y(t)$, and $n(t)$, respectively,
- the superscript * denotes complex conjugation.

The filtering operation may either be carried out in the time-domain, as above, or in the frequency domain:

$$\hat{X}(f) = G(f)Y(f)$$

and then performing an inverse Fourier transform on $\hat{X}(f)$ to obtain $\hat{x}(t)$.

Wiener

- N(f) can be estimated using a flat patch in the image.

- S(f) is estimated / approximated by smoothing the image with a low pass filter.

Bilateral

- The Bilateral filter is an edge preserving filter.
- The Bilateral filter performs spatial averaging in the flat regions of an image.
- Unlike a Gaussian blur filter, the Bilateral filter respects / preserves edges.
- Whereas the Diffusion filter (discussed next) is iterative, the Bilateral filter is not.

Bilateral

The bilateral filter is defined by the following equations:

$$I_{\text{filtered}}(x, y) = \frac{1}{W(x, y)} \sum_{x'=-k}^{k} \sum_{y'=-k}^{k} I(x', y') \cdot G_s(||(x', y') - (x, y)||) \cdot G_r(|I(x', y') - I(x, y)|)$$

W is a normalization term, and is given by:

$$W(x, y) = \sum_{x'=-k}^{k} \sum_{y'=-k}^{k} G_s(||(x', y') - (x, y)||) \cdot G_r(|I(x', y') - I(x, y)|)$$

Bilateral

Gs is defined to be:

$$G_s(d) = \exp\left(-\frac{d^2}{2\sigma_s^2}\right)$$

While Gr is defined to be:

$$G_r(\Delta I) = \exp\left(-\frac{(\Delta I)^2}{2\sigma_r^2}\right)$$

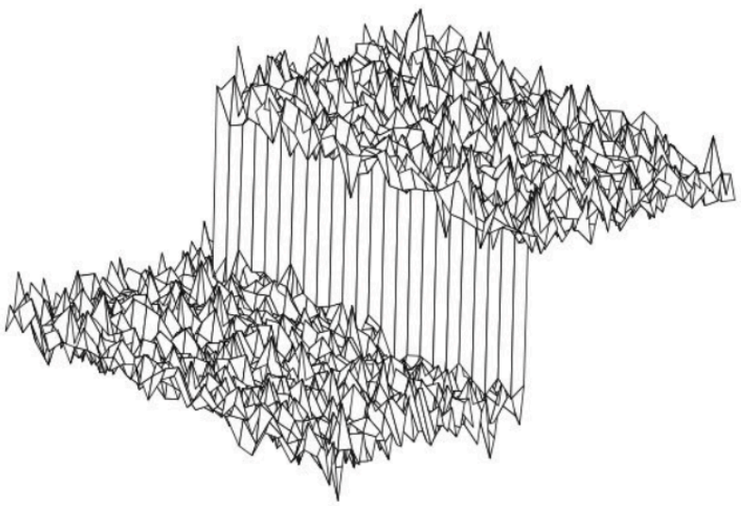
Bilateral

- These two terms control the effective “reach” of the filter, based on spatial position and pixel intensities.

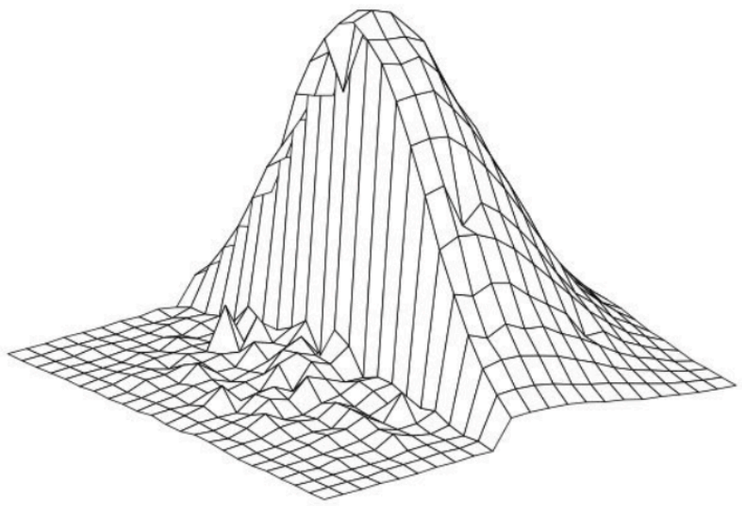
- By construction, the impact of a neighboring pixel on the new, estimated pixel value decreases, the further the neighboring pixel is from the reference pixel.

Bilateral

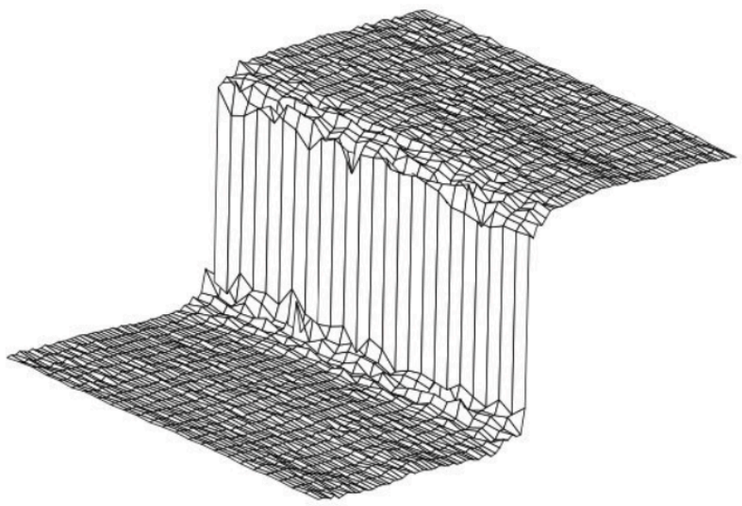
The following picture illustrates how a bilateral filter denoises a noisy step edge, using the Gs and Gr terms, combined.



Noisy Input Step Edge



Filter Kernel from BL Filter

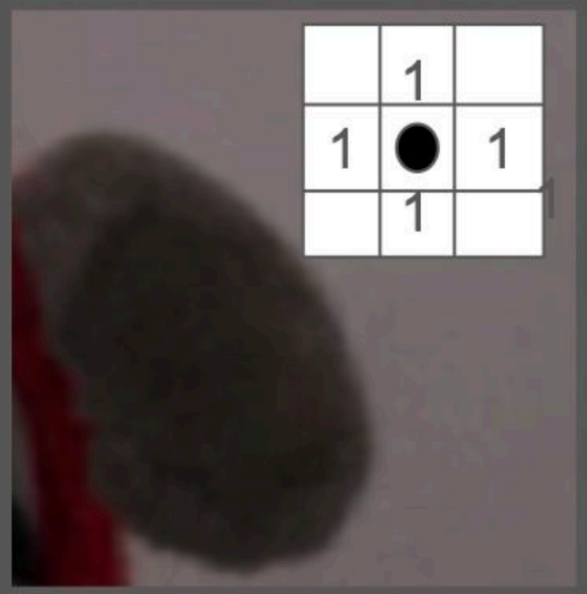


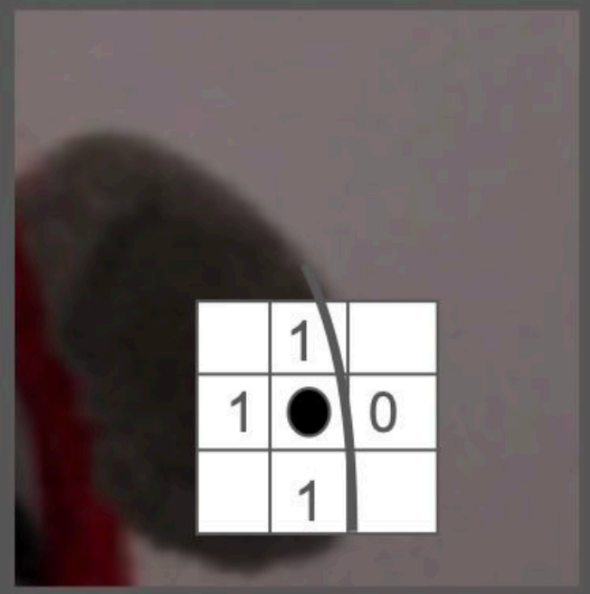
Denoised Step Edge

Diffusion

- The Diffusion filter is an adaptive, edge preserving filter.
- The Diffusion filter performs spatial averaging in the flat regions of an image.
- Unlike a Gaussian blur filter, the Diffusion filter respects / preserves edges.
- The Diffusion filter is a numerical discretization of the non-linear heat diffusion equation.

Diffusion





Input Image

Diffusion Filter
Smooths

Diffusion Filter
Preserves Edges

Diffusion

Diffusion is based the non linear heat equation.

This equation is given as:

$$\frac{\partial I}{\partial t} = \nabla \cdot (g(|\nabla I|)\nabla I)$$

Diffusion

The terms in the equation are defined as follows:

$I(x, y, t)$ is the image intensity.

∇I is the image gradient.

$g(|\nabla I|)$ is the conductance function (edge-stopping function).

$\nabla \cdot$ is the divergence operator.

Diffusion

Where we can define:

$$g(|\nabla I|) = e^{-\left(\frac{|\nabla I|}{K}\right)^2}$$

Where K is a tuning parameter.

Diffusion

Numerically discretizing the respective terms in the continuous equation, we obtain:

$$I_x = \frac{I_{i+1,j} - I_{i,j}}{\Delta x}, \quad I_y = \frac{I_{i,j+1} - I_{i,j}}{\Delta y}$$

$$g_{i,j} = e^{-\left(\frac{|\nabla I_{i,j}|}{K}\right)^2}$$

Diffusion

Now, approximating the terms in the continuous equation, we obtain:

First term $\frac{\partial}{\partial x}\left(g\frac{\partial I}{\partial x}\right)$**:**

$$\frac{\partial}{\partial x}\left(g\frac{\partial I}{\partial x}\right) \approx \frac{g_{i,j}(I_{i+1,j} - I_{i,j}) - g_{i-1,j}(I_{i,j} - I_{i-1,j})}{\Delta x^2}$$

Second term $\frac{\partial}{\partial y}\left(g\frac{\partial I}{\partial y}\right)$**:**

$$\frac{\partial}{\partial y}\left(g\frac{\partial I}{\partial y}\right) \approx \frac{g_{i,j}(I_{i,j+1} - I_{i,j}) - g_{i,j-1}(I_{i,j} - I_{i,j-1})}{\Delta y^2}$$

Diffusion

Combining these terms, we then obtain the following update equation:

$$I_{i,j}^{n+1} = I_{i,j}^{n} + \Delta t \left[\frac{g_{i,j}\left(I_{i+1,j} - I_{i,j}\right) - g_{i-1,j}\left(I_{i,j} - I_{i-1,j}\right)}{\Delta x^2} + \frac{g_{i,j}\left(I_{i,j+1} - I_{i,j}\right) - g_{i,j-1}\left(I_{i,j} - I_{i,j-1}\right)}{\Delta y^2} \right]$$

Median

- The Median filter chooses the median pixel value in an given neighborhood.
- The Median filter is a special case of an order statistics filter.

Median

10	32	45	41	27
36	33	15	11	23
87	92	55	57	120
93	65	81	15	22
240	15	55	87	12

The grid to the left represents a 5x5 window, containing different pixel intensity values.

After processing with a 3x3 median filter, a value of 55 is returned.

After processing with a 5x5 median filter, a value of 45 is returned.

References

- Antoni Buades, et. al., Non Local Means Denoising
- Kostadin Dabov, et. al., Image Denoising by Sparse 3D Transform Domain Collaborative Filtering
- Marc LeBrun, An Analysis and Implementation of BM3D
- Grace Chang, et. al., Adaptive Wavelet Thresholding for Image Denoising and Compression
- Stephen Mallat, A Theory for Multi-resolution Spatial Decomposition: The Wavelet Transform
- Norbert Wiener, Extrapolation, Interpolation and Smoothing of Stationary Time Series
- S. Haykin, Adaptive Filter Theory
- C Tomasi and R Manduchi, Bilateral Filter for Gray and Color Images
- Sylvian Paris, et. Al., A Gentle Introduction to Bilateral Filtering and It's Applications
- Patrick Guidotti, Anisotropic Diffusions of Image Processing From Perona and Malik On
- Pietro Perona and Jitendra Malik, Scale Space and Edge Detection Using Anisotropic Diffusion
- Thomas Huang, et. al., A Fast Two Dimensional Median Filter Algorithm
- S. Perreault and P. Herbert, Median Filtering in Constant Time