中国科学技术大学
University of Science and Technology of China

Image Convolution

Juyong Zhang
School of Mathematics, USTC

Spatial Filtering

Let **I** and **J** be images such that **J** = T[**I**].

T[·] represents a transformation, such that,

$$\mathbf{J}(r,c) = \mathrm{T}[\mathbf{I}](r,c) =$$
$$f\left(\left\{\mathbf{I}(\rho,\chi) \,\middle|\, \rho \in \{r-s,\ldots,r,\ldots r+s\}, \chi \in \{c-d,\ldots,c,\ldots c+d\}\right\}\right).$$

That is, the value of the transformed image, **J**, at pixel location (*r,c*) is a function of the values of the original image, **I**, in a $2s+1 \times 2d+1$ rectangular neighborhood centered on pixel location (*r,c*).

Moving Windows

- The value, J(r,c) = T[I](r,c), is a function of a rectangular neighborhood centered on pixel location (r,c) in I.

- There is a different neighborhood for each pixel location, but if the dimensions of the neighbor-hood are the same for each location, then transform T is sometimes called a moving window transform.

Moving-Window Transformations

Neutral Buoyancy Facility at NASA Johnson Space Center

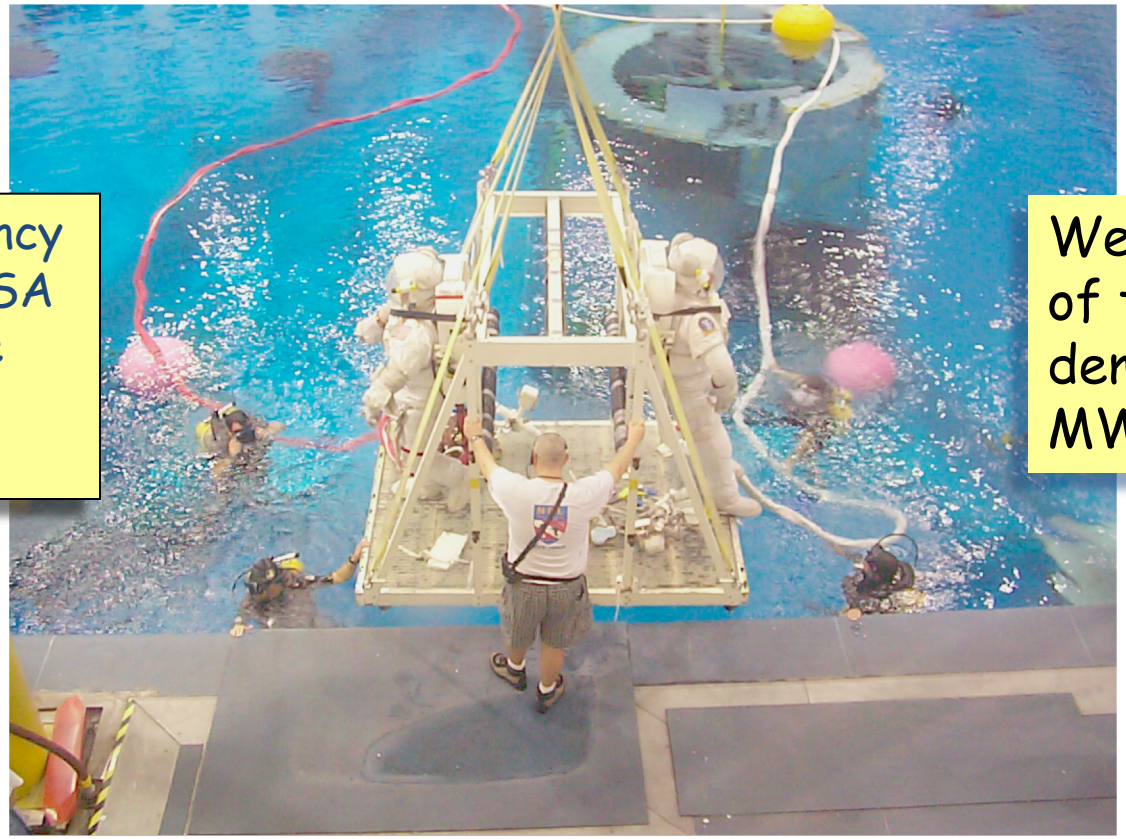
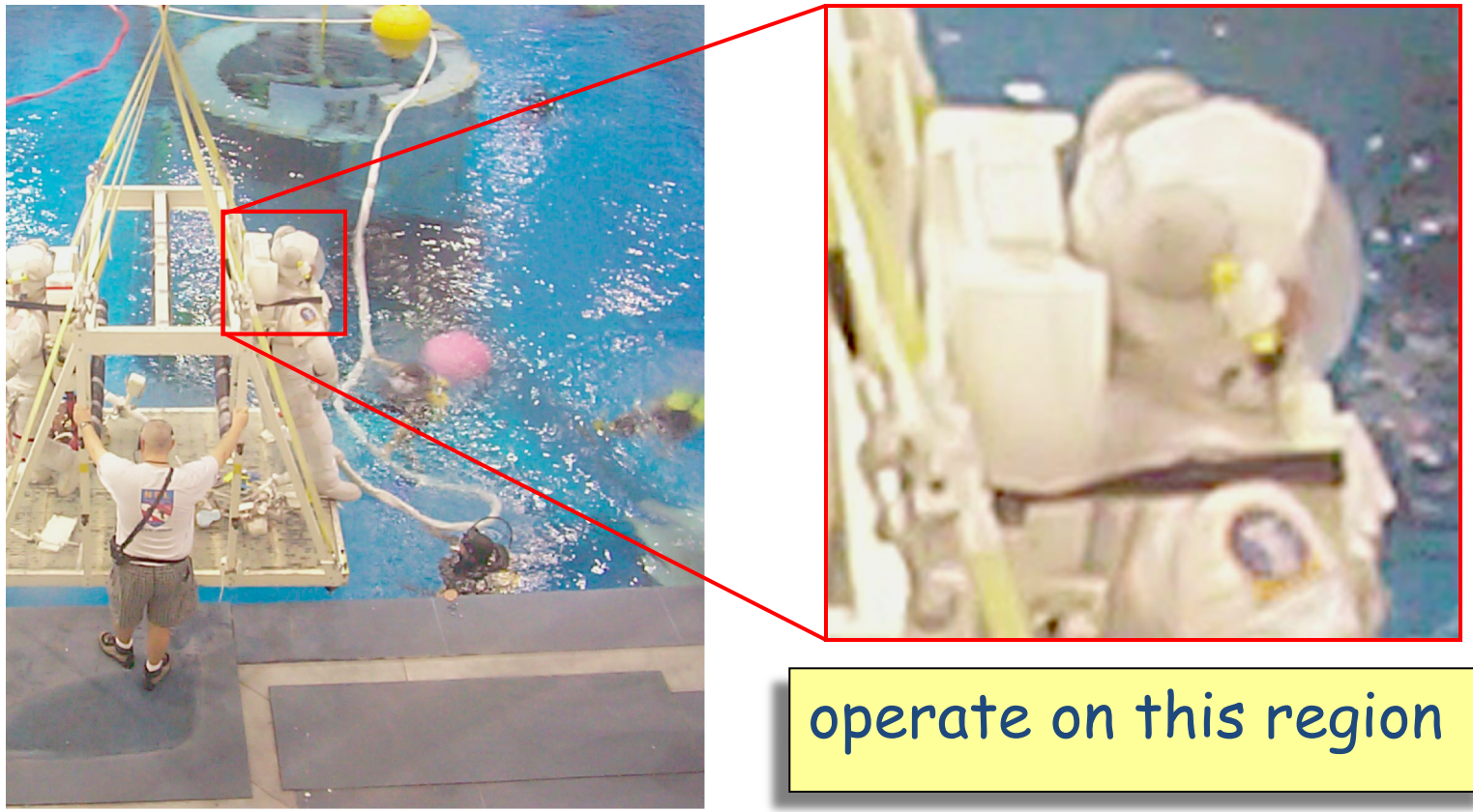


photo: R.A.Peters II, 1999

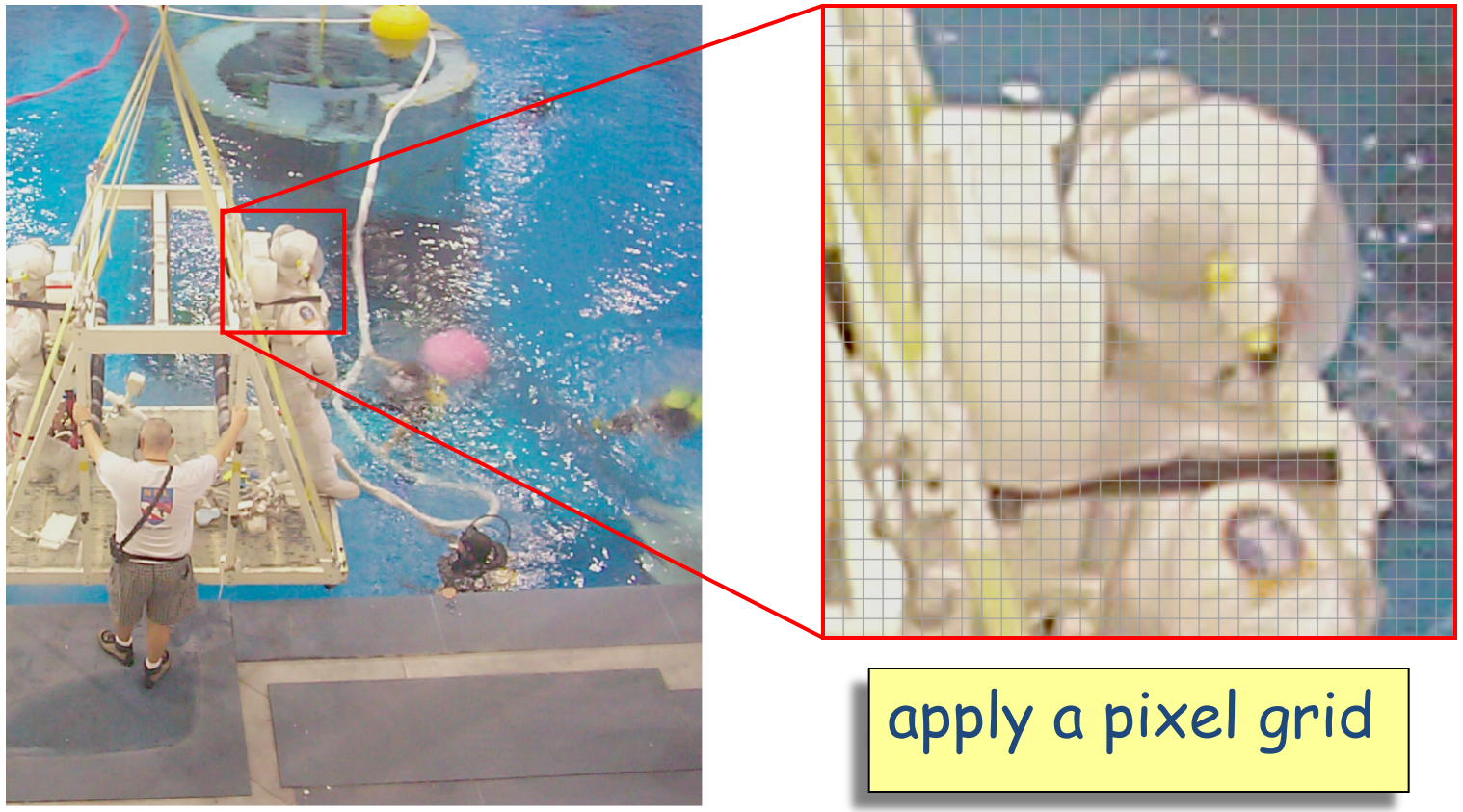
We' ll take a section of this image to demonstrate the MWT

Moving-Window Transformations



Moving-Window Transformations

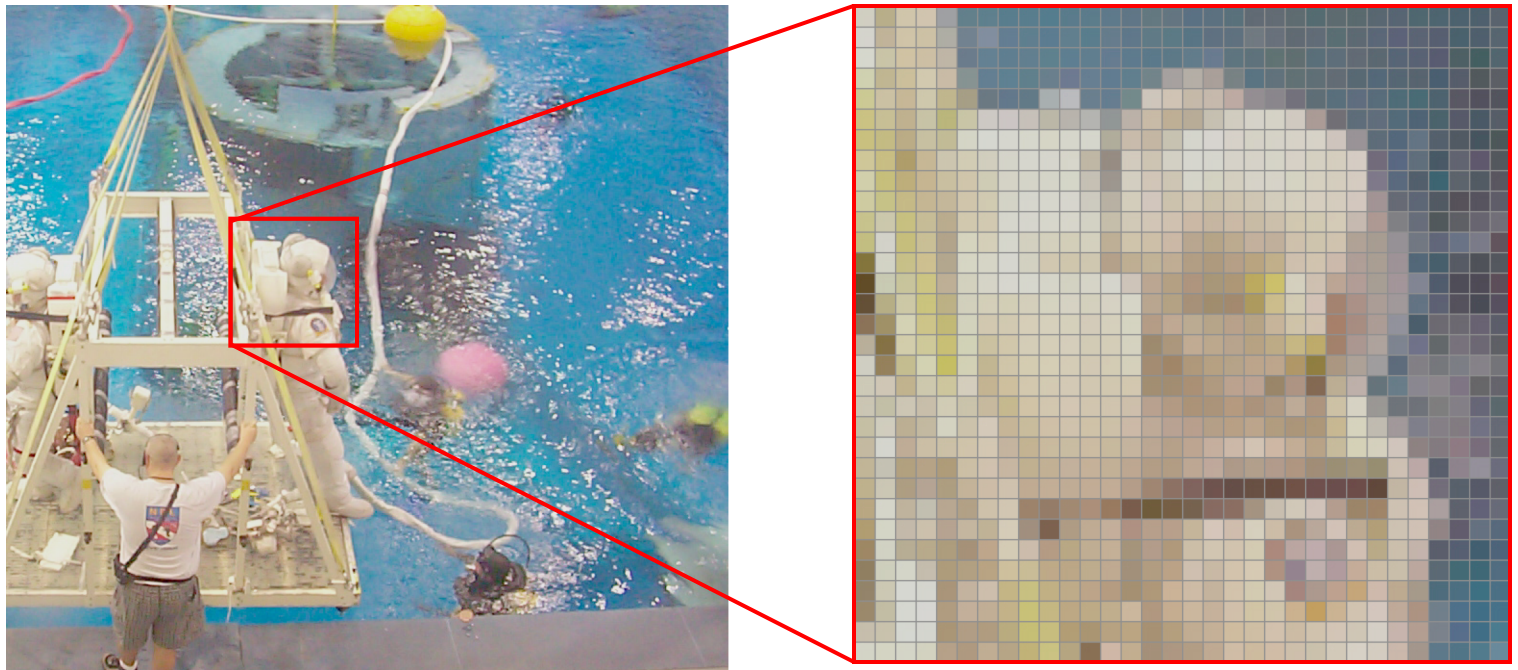
Pixelize the section to better see the effects.



apply a pixel grid

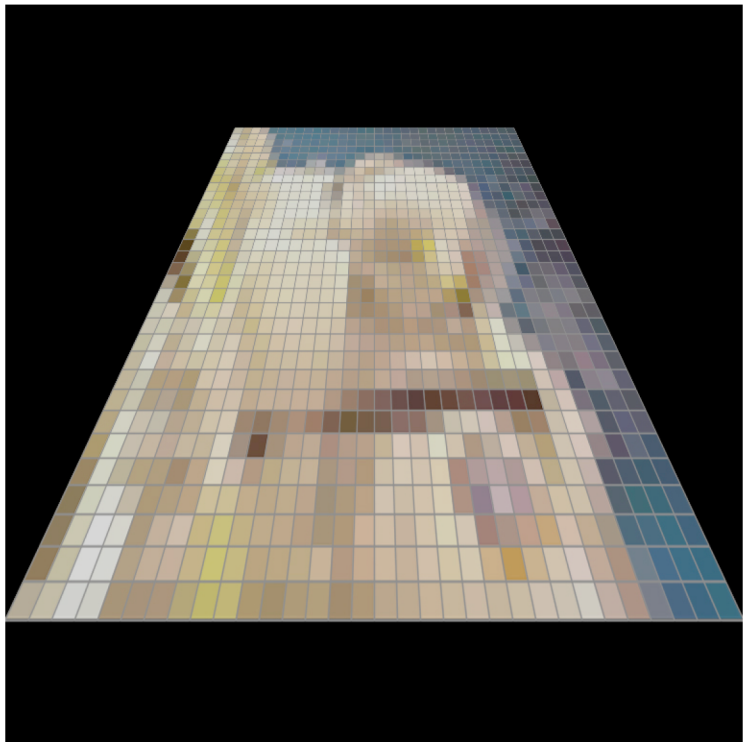
Moving-Window Transformations

Pixelize the section to better see the effects.

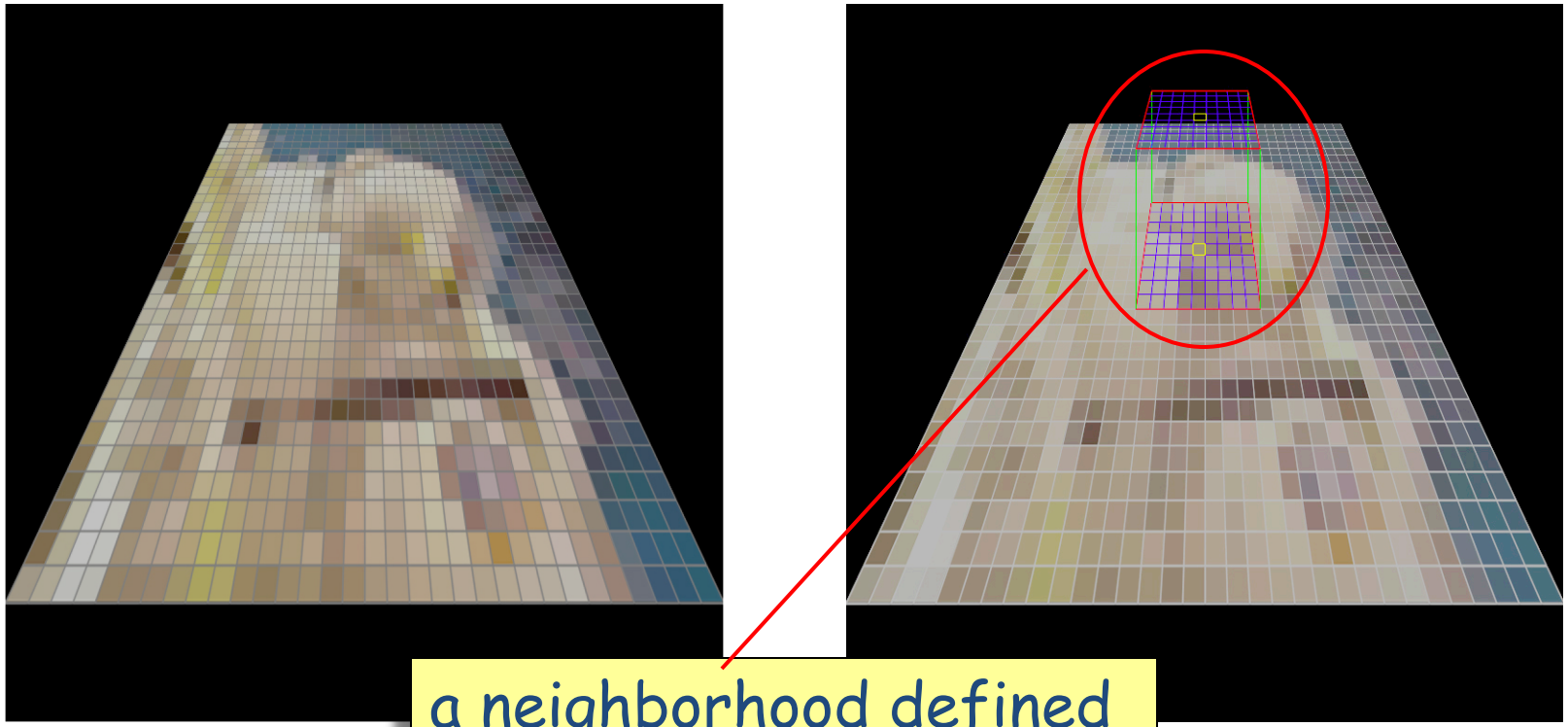


sample (average in the squares).

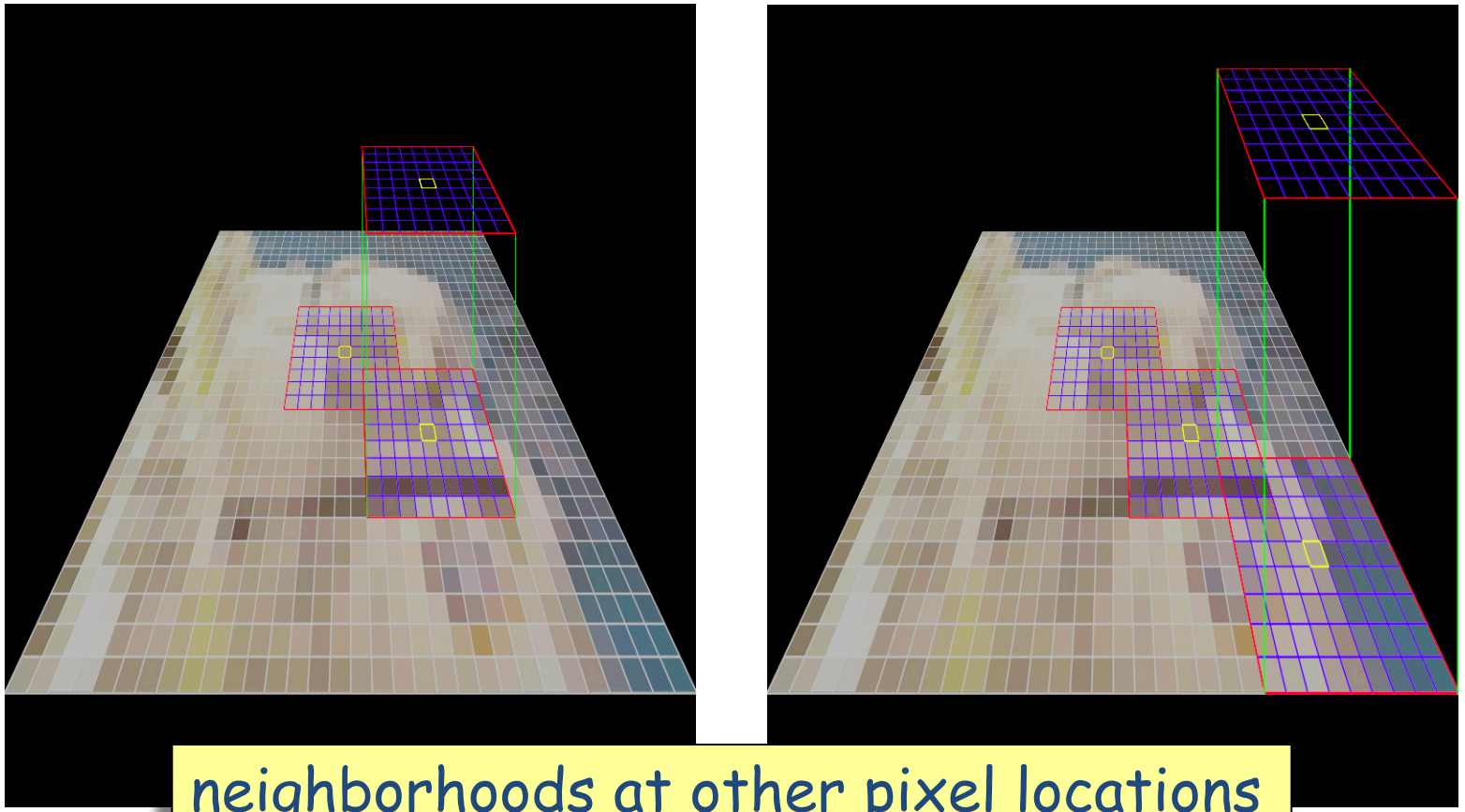
Moving-Window Transformations



Moving-Window Transformations

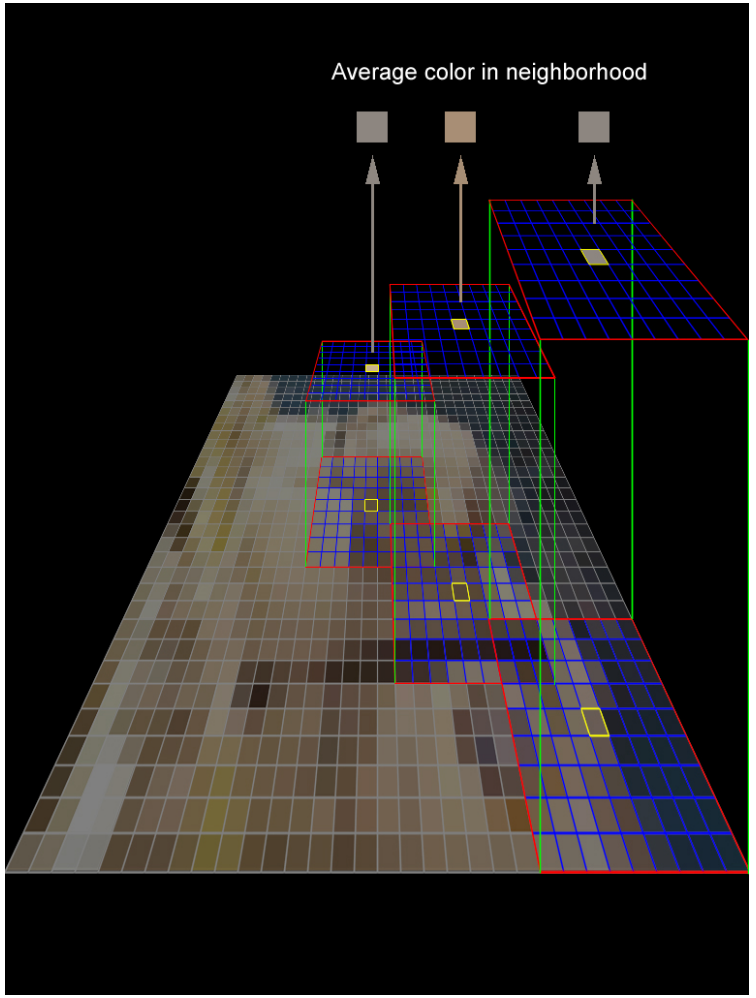


Moving-Window Transformations

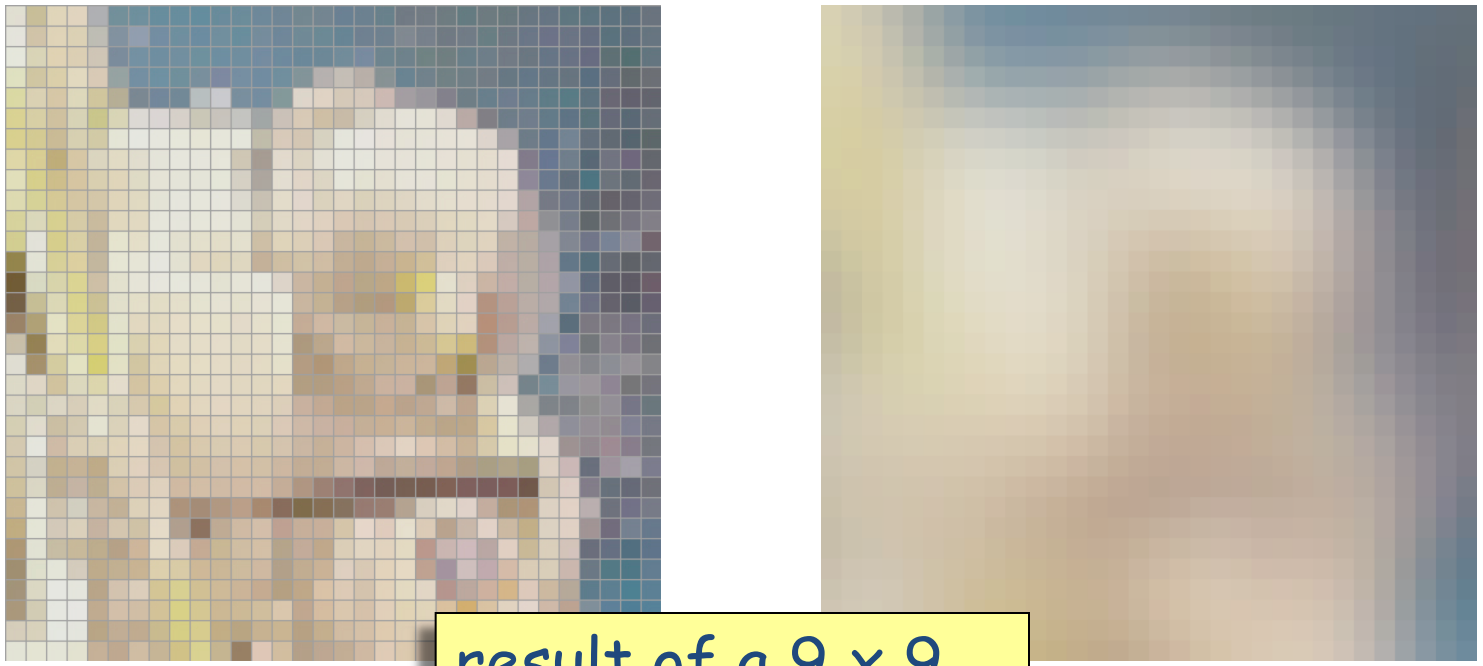


Linear Moving-Window Transformations
(*i.e.* convolution)

The output of the transform at each pixel is the (weighted) average of the pixels in the neighborhood.



Moving-Window Transformations



result of a 9 x 9
uniform averaging

Convolution: Mathematical Representation

If a MW transformation is *linear* then it is a *convolution*:

$$\mathbf{J}(r,c) = \left[\mathbf{I} * \mathbf{h}\right](r,c) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathbf{I}(r-\rho, c-\chi)\mathbf{h}(\rho,\chi)\, d\rho\, d\chi\,,$$

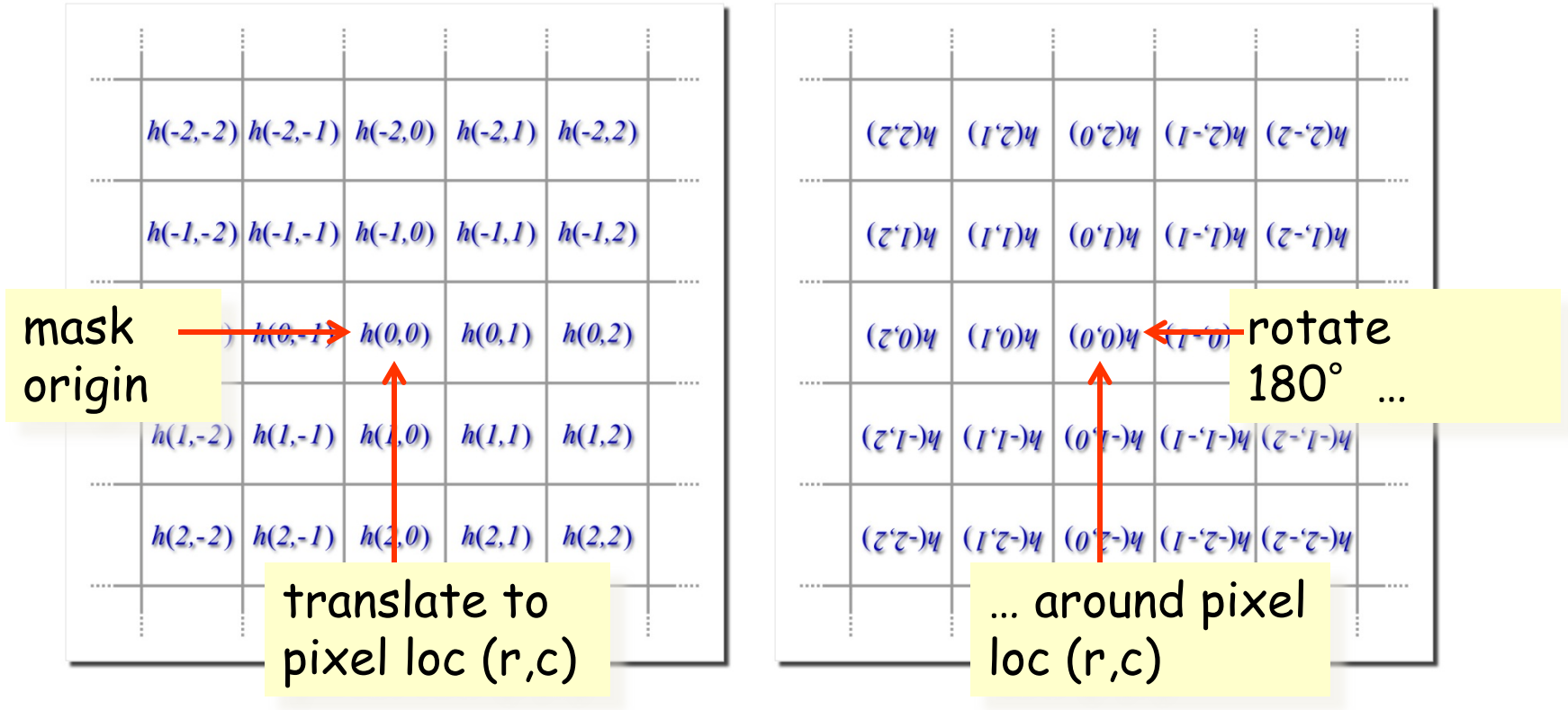
for a real image ($\mathbf{I}: R \times R \rightarrow R$), or for a digital image ($\mathbf{I}: Z \times Z \rightarrow Z$):

$$\mathbf{J}(r,c) = \left[\mathbf{I} * \mathbf{h}\right](r,c) = \sum_{\rho=-s}^{s}\sum_{\chi=-d}^{d} \mathbf{I}(r-\rho, c-\chi)\mathbf{h}(\rho,\chi)$$

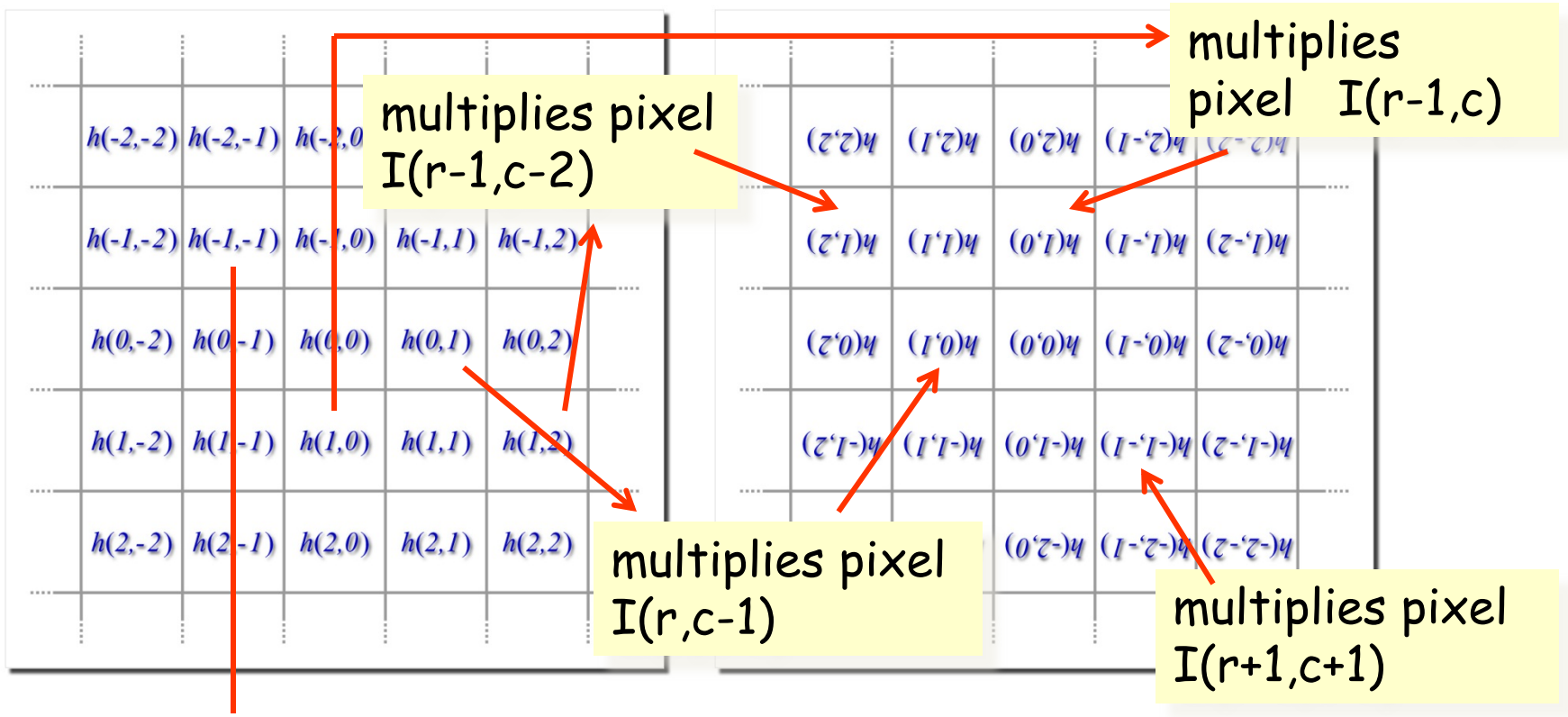
Convolution Mask (Weight Matrix)

- The object, $\mathbf{h}(\rho, \chi)$, in the equation is a weighting function, or in the discrete case, a rectangular matrix of numbers.
- The matrix is the moving window.
- Pixel (*r*,*c*) in the output image is the weighted sum of pixels from the original image in the neighborhood of (*r*,*c*) traced by the matrix.
- Each pixel in the neighborhood of (*r*,*c*) is multiplied by the corresponding matrix value — after the matrix is rotated by 180º.
- The sum of those products is the value of pixel (*r*,*c*) in the output image

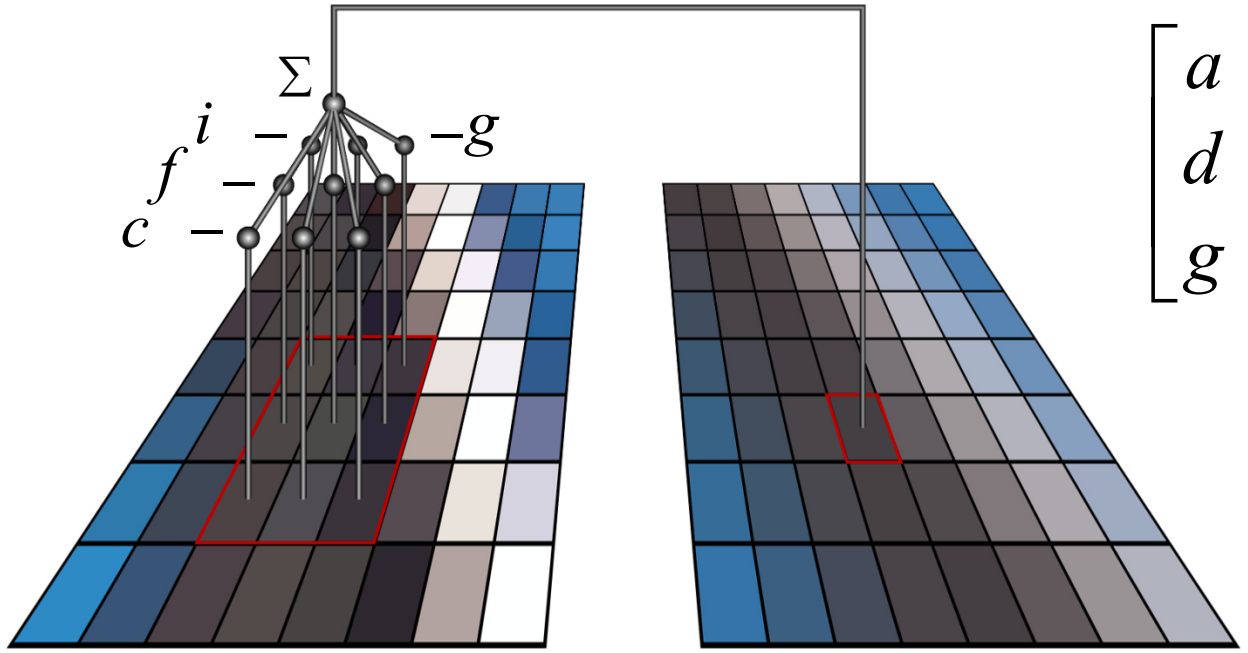
Convolution Masks: Moving Window



Convolution Masks: Moving Window



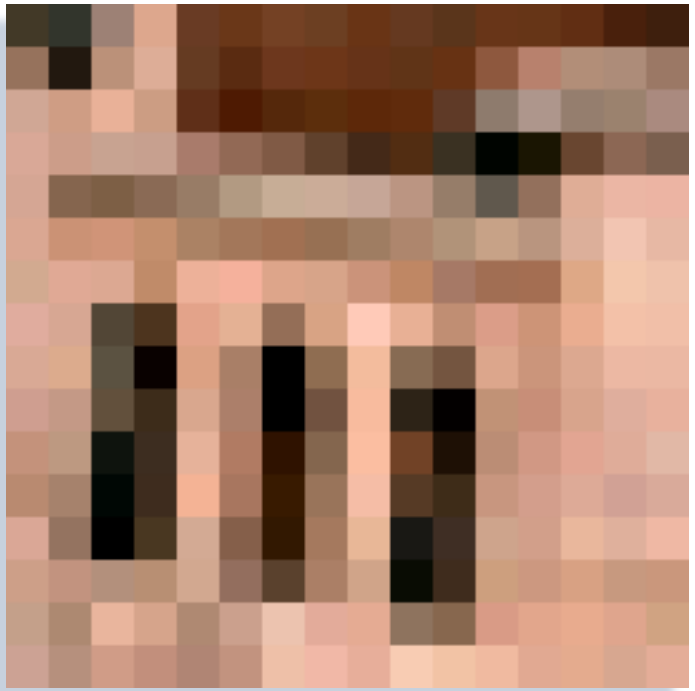
Convolution by Moving Window



$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

Moving Window Transform: Example

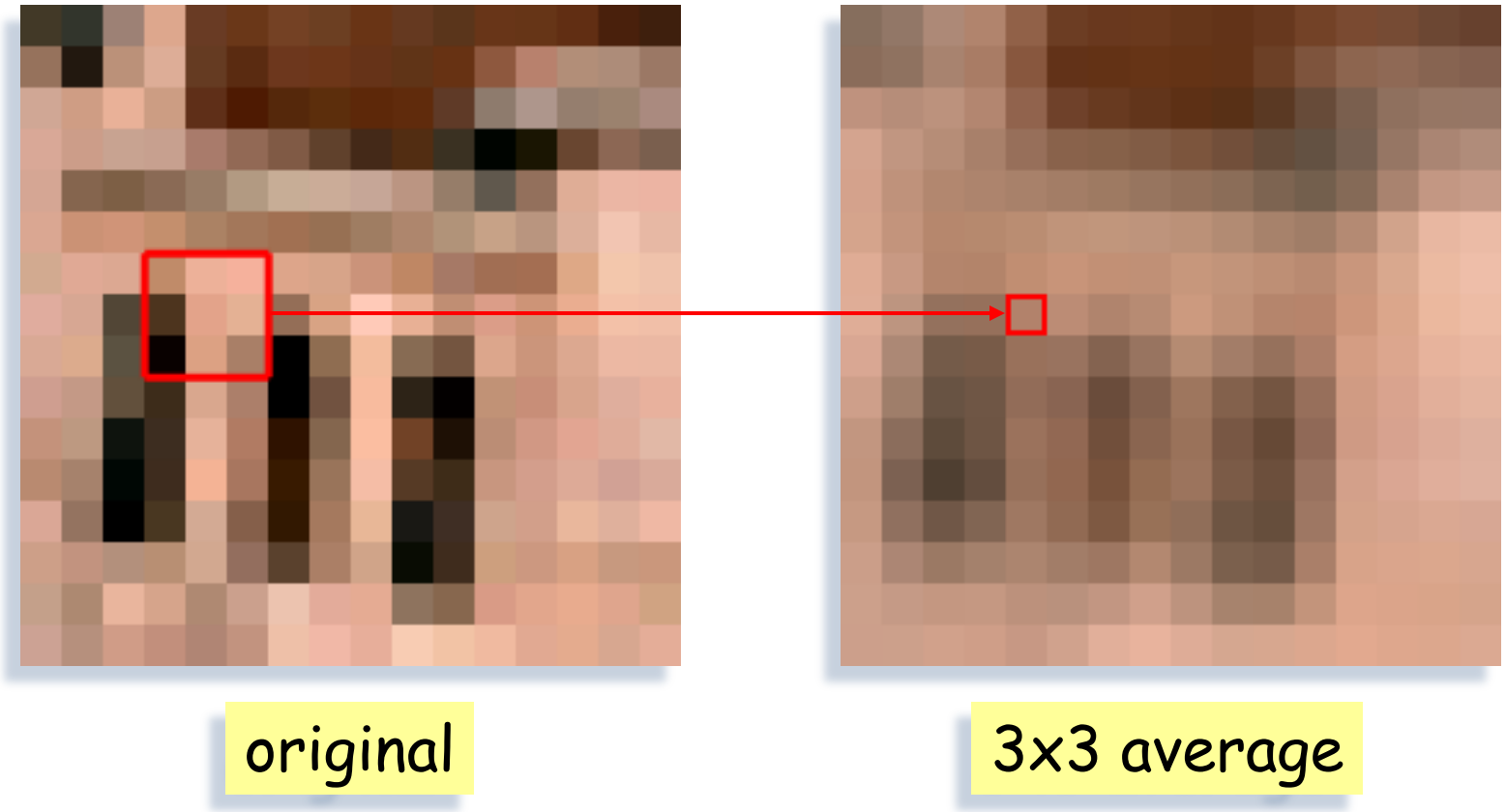
Another example



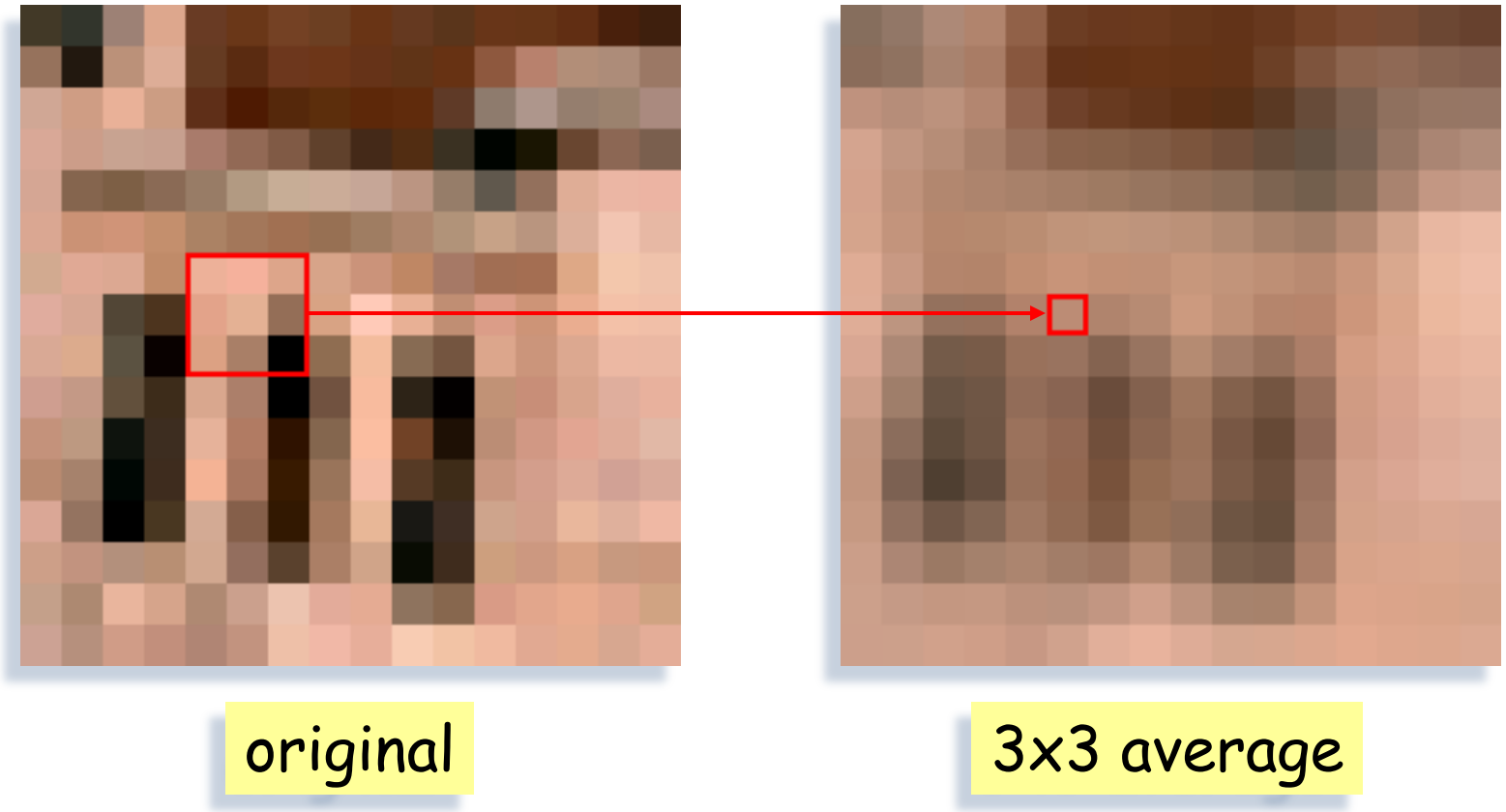
original

3x3 average

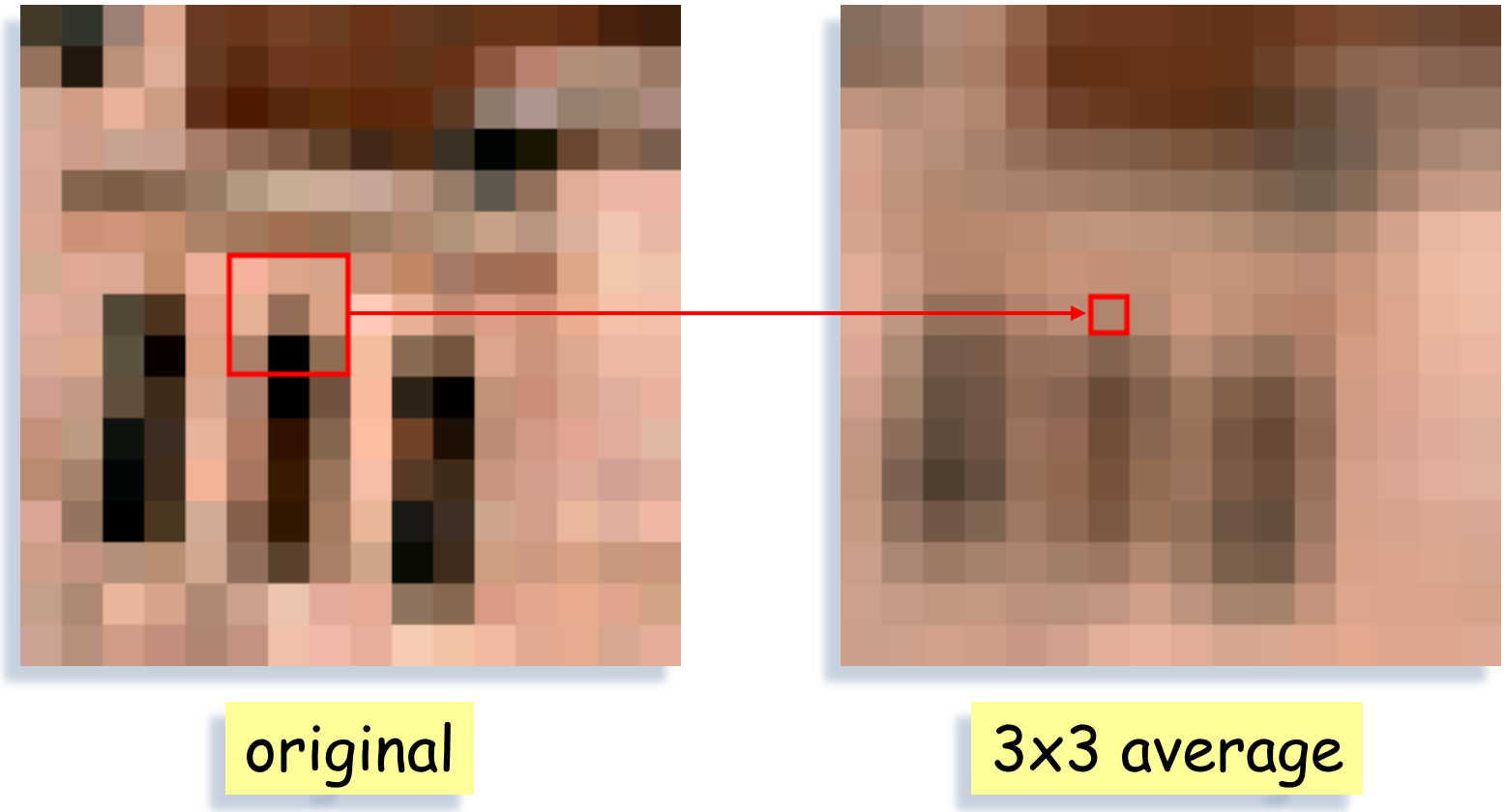
Moving Window Transform: Example



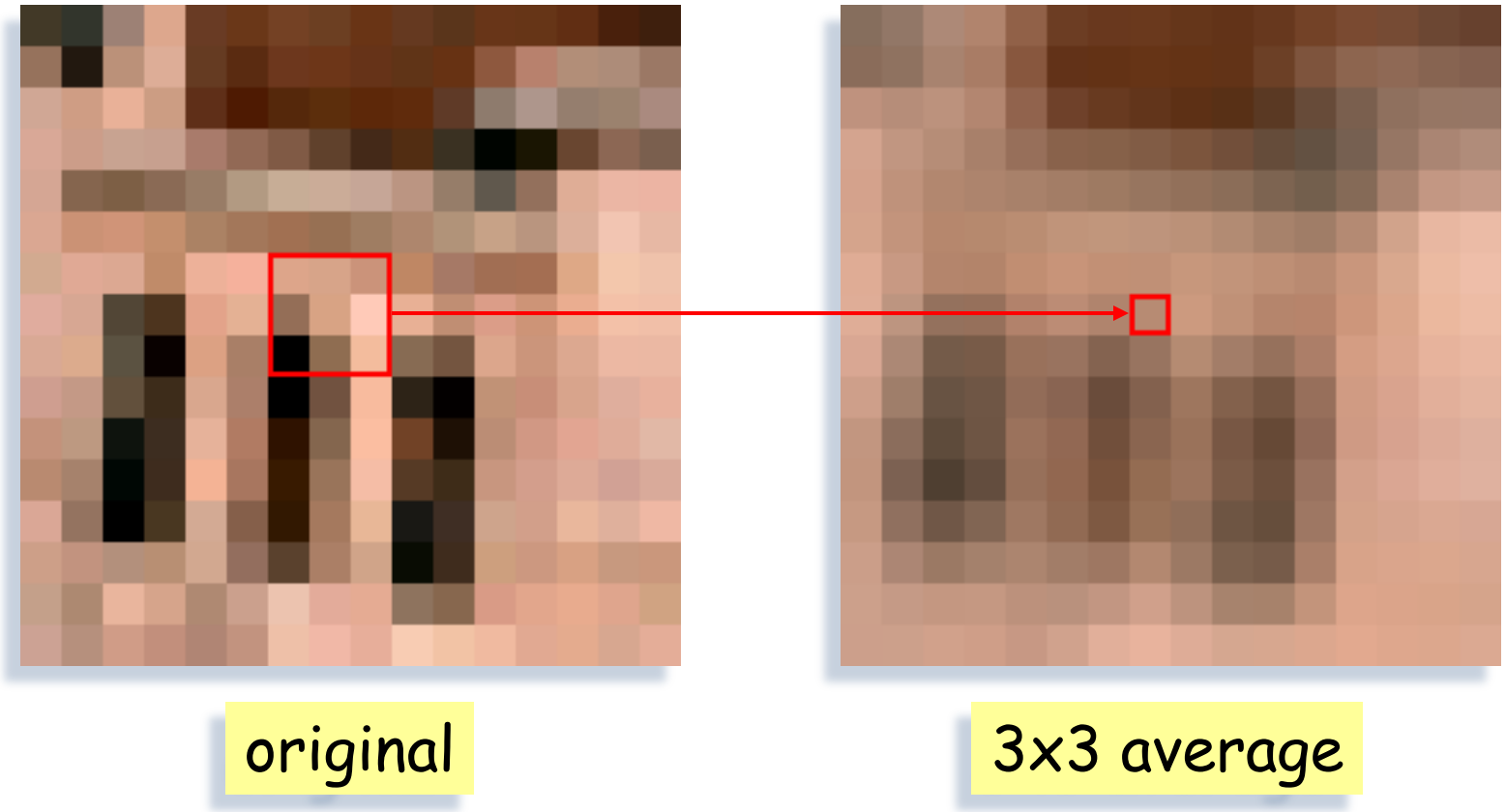
Moving Window Transform: Example



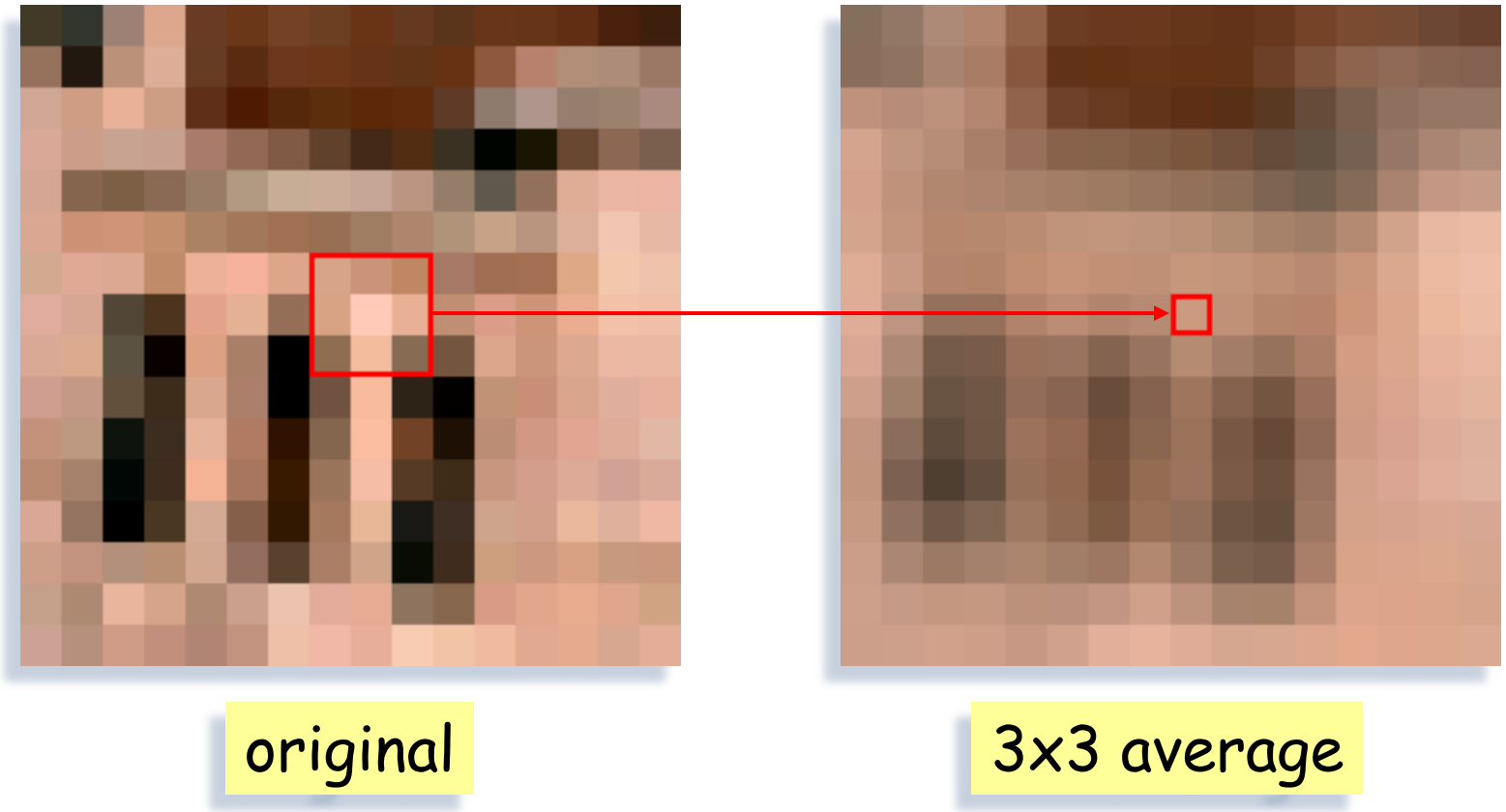
Moving Window Transform: Example



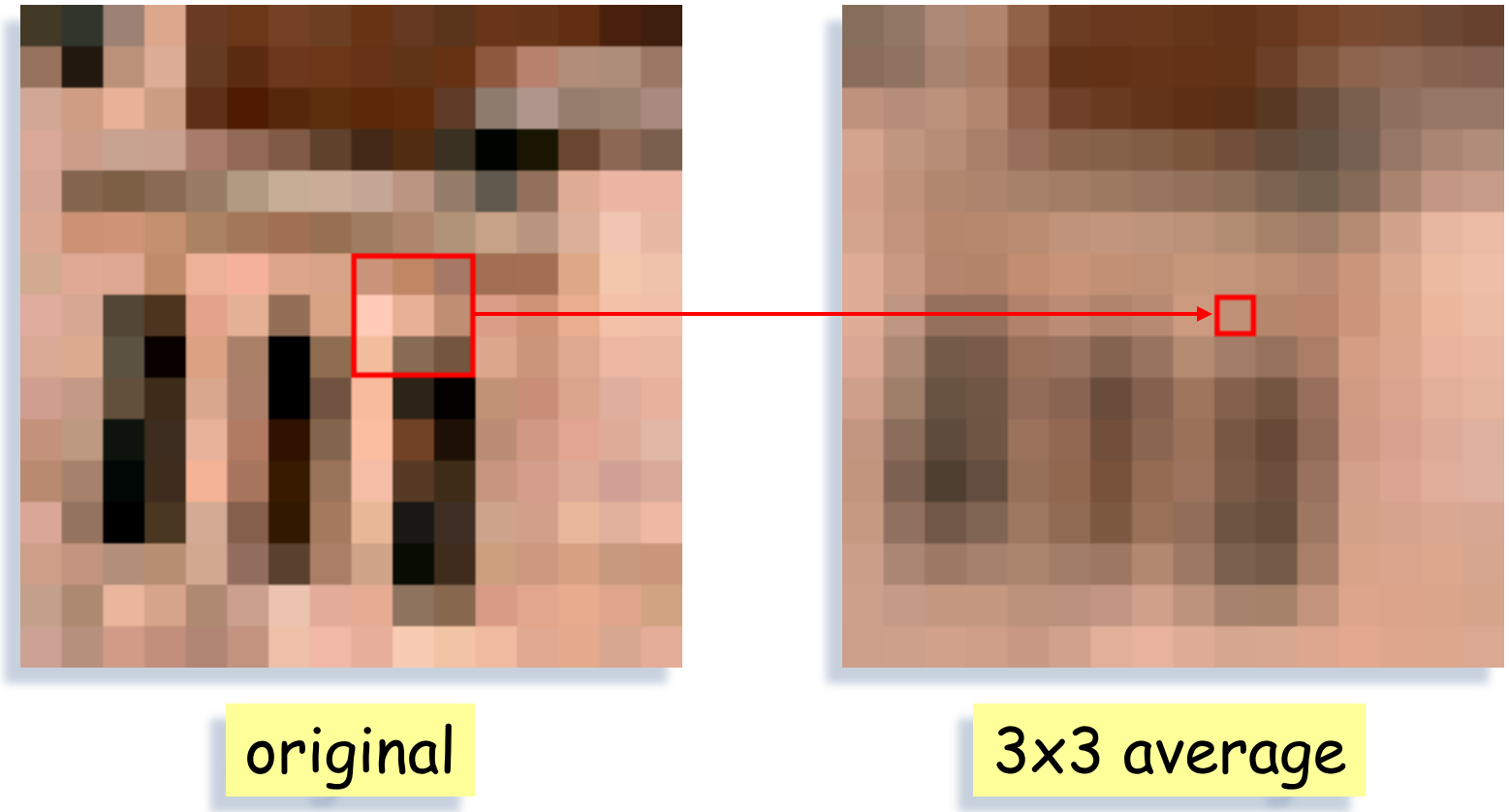
Moving Window Transform: Example



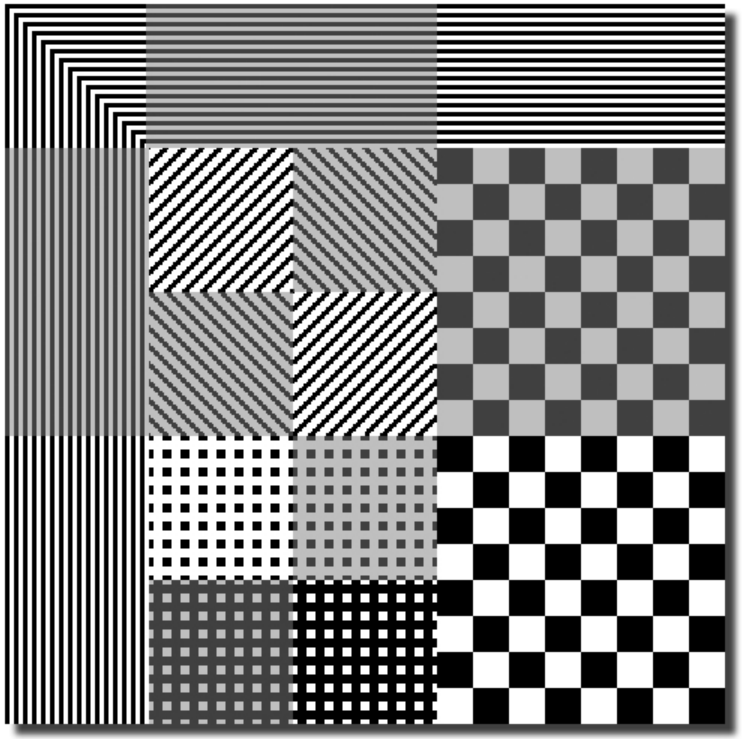
Moving Window Transform: Example



Moving Window Transform: Example

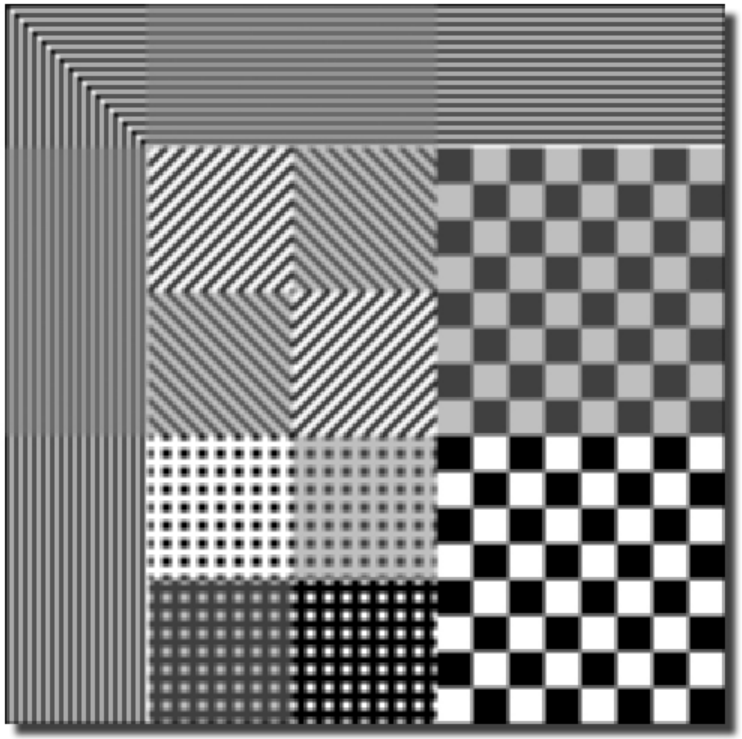


Convolution Examples: Original Images



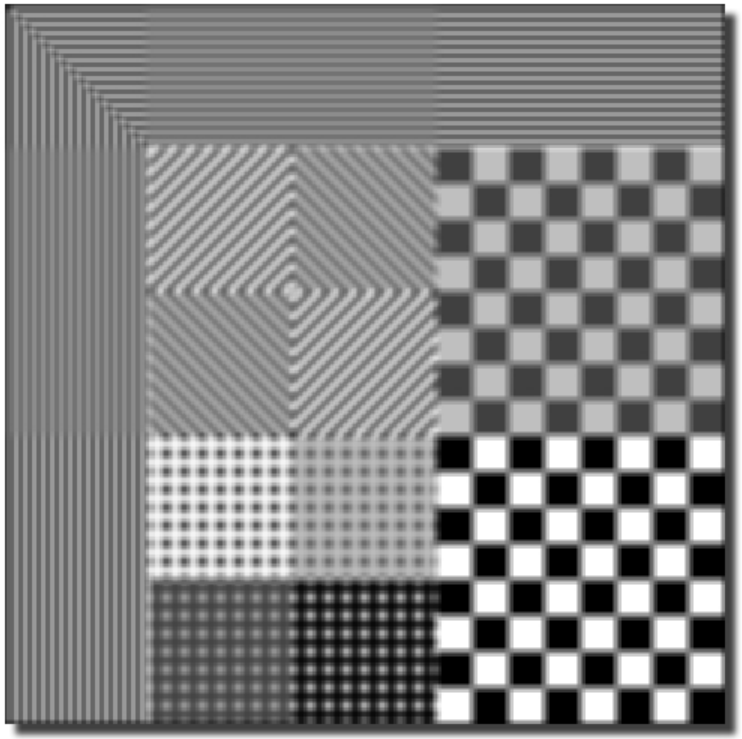
Convolution Examples: 3×3 Blur

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



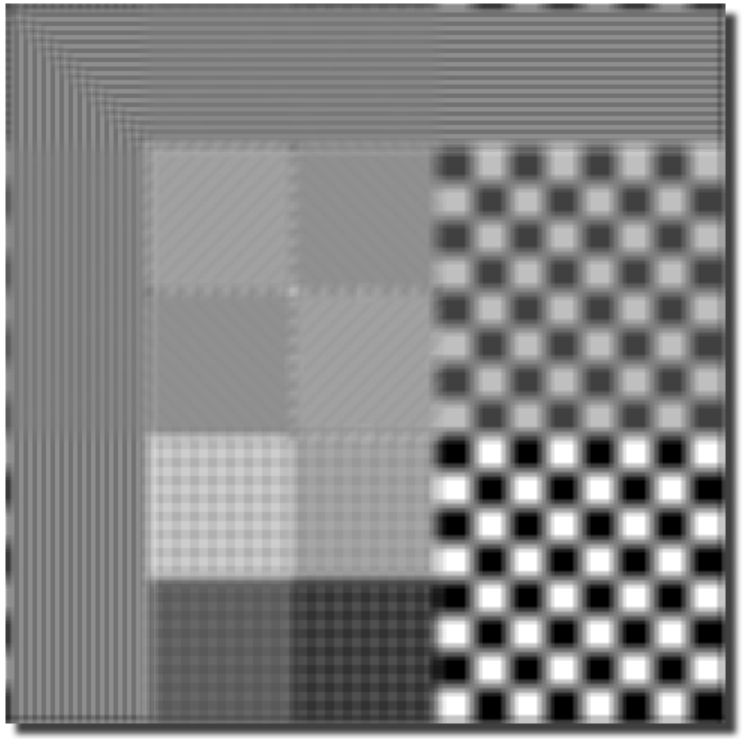
Convolution Examples: 5×5 Blur

$$\frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$



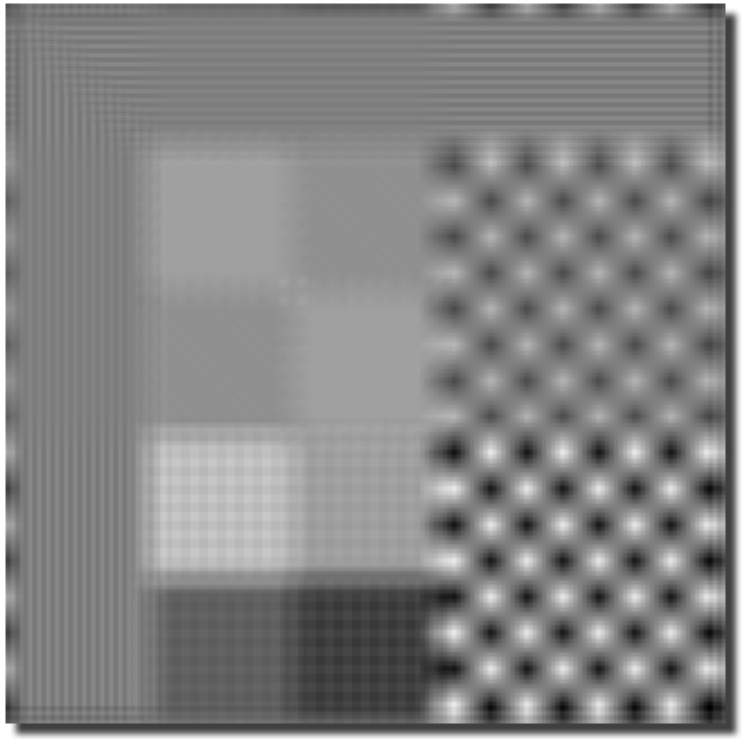
Convolution Examples: 9×9 Blur

$$\frac{1}{81}\begin{bmatrix} 1&1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1&1 \end{bmatrix}$$

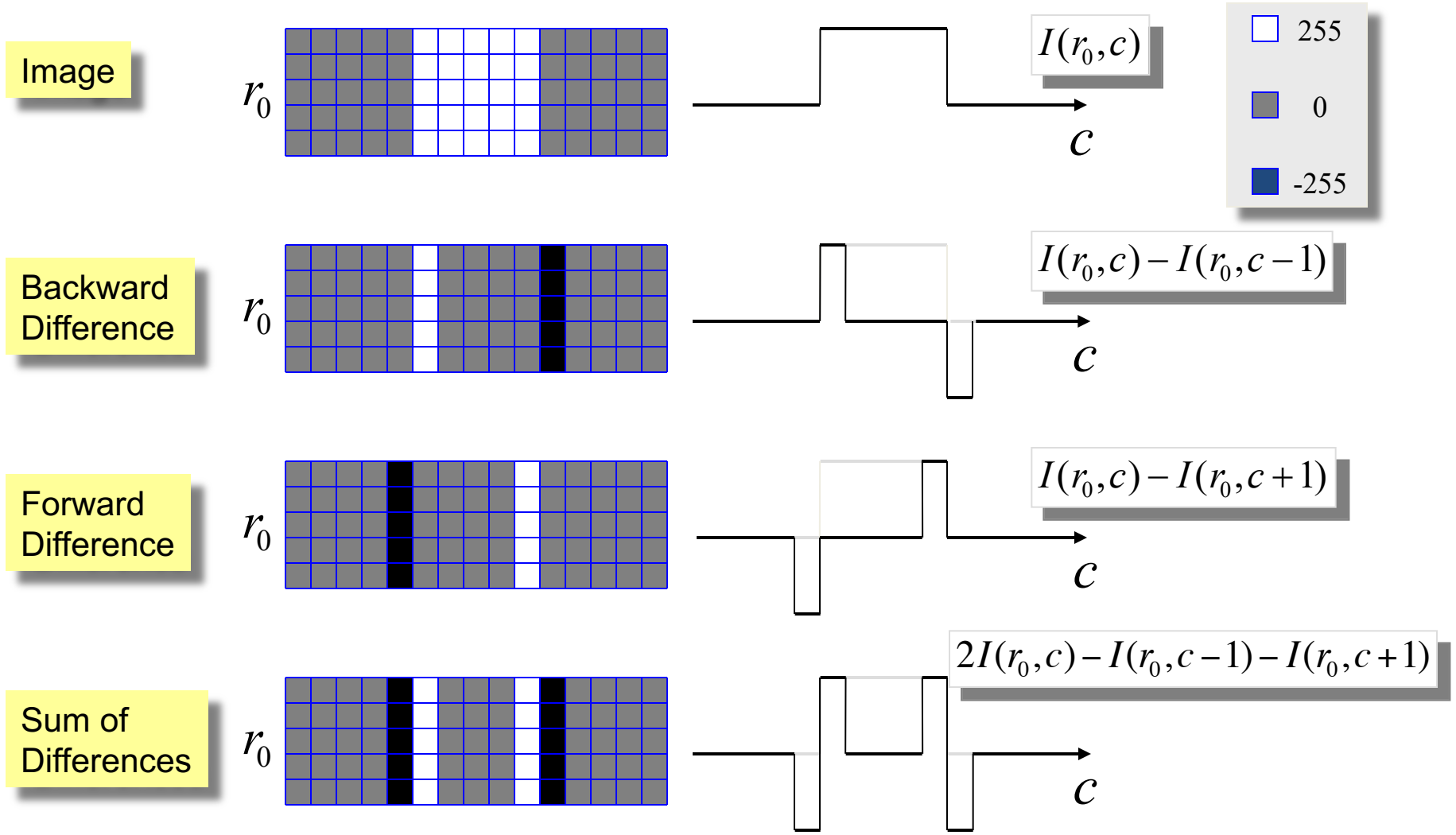


Convolution Examples: 17×17 Blur

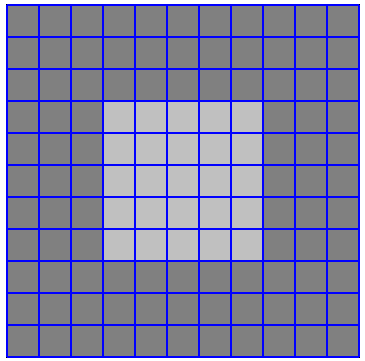
$$\frac{1}{289}\begin{bmatrix}1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\end{bmatrix}$$



Vertical Edge Detection

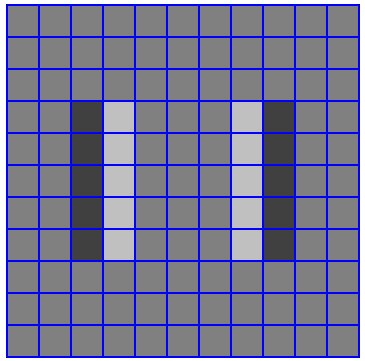


Symmetric Edge Detection



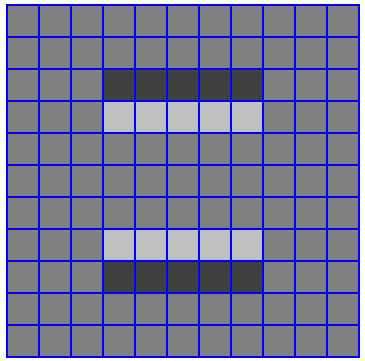
$$I(r,c)$$

510
255
0
-255



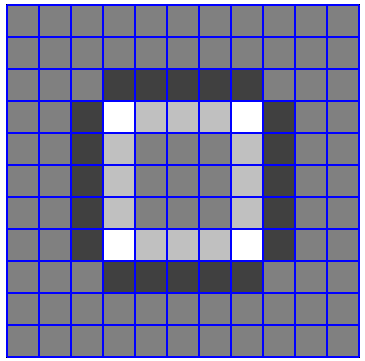
$$2I(r,c) - I(r,c-1) - I(r,c+1)$$

-1	2	-1



$$2I(r,c) - I(r-1,c) - I(r+1,c)$$

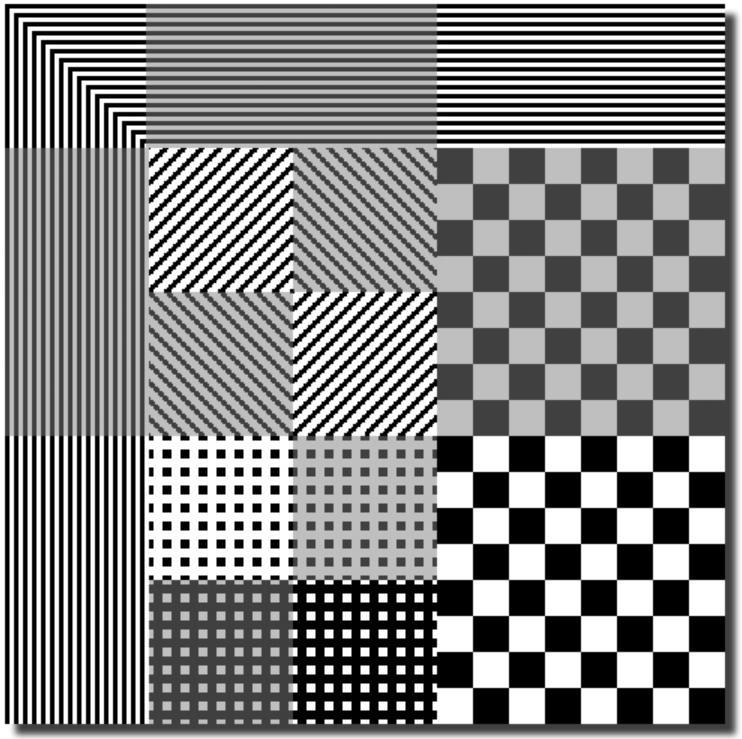
	-1	
	2	
	-1	



$$4I(r,c) - I(r-1,c) - I(r+1,c) - I(r,c-1) - I(r,c+1)$$

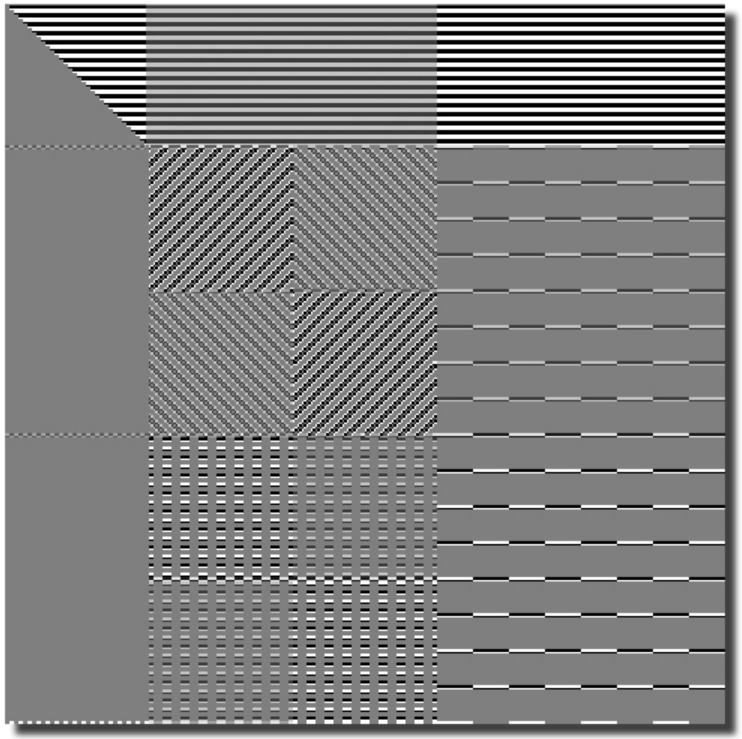
	-1	
-1	4	-1
	-1	

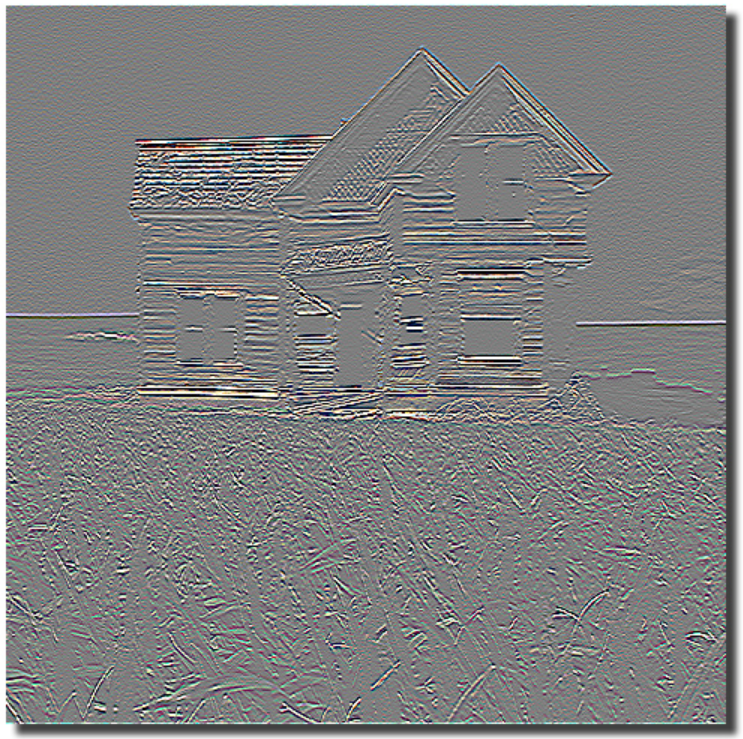
Convolution Examples: Original Images



Convolution Examples: Vertical Difference

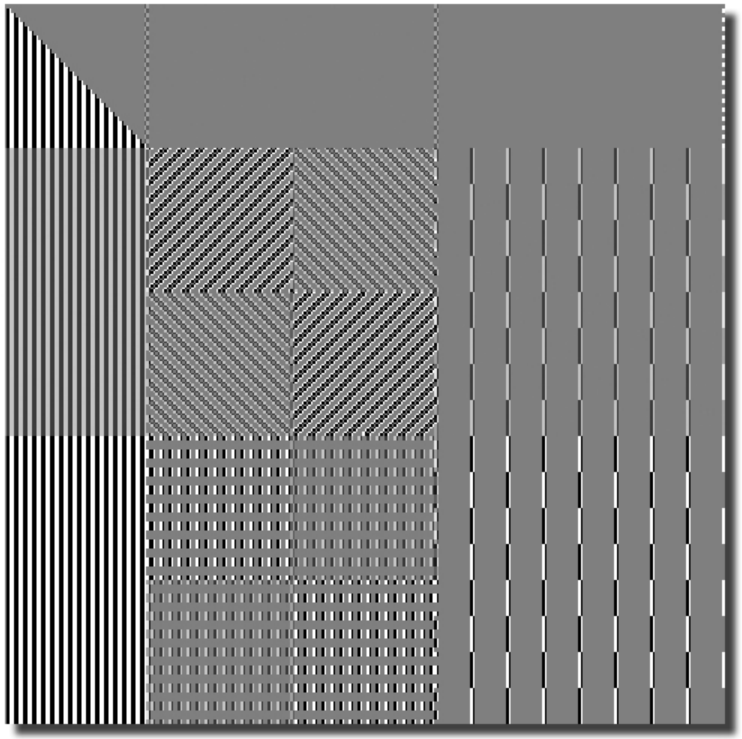
$$\begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}$$

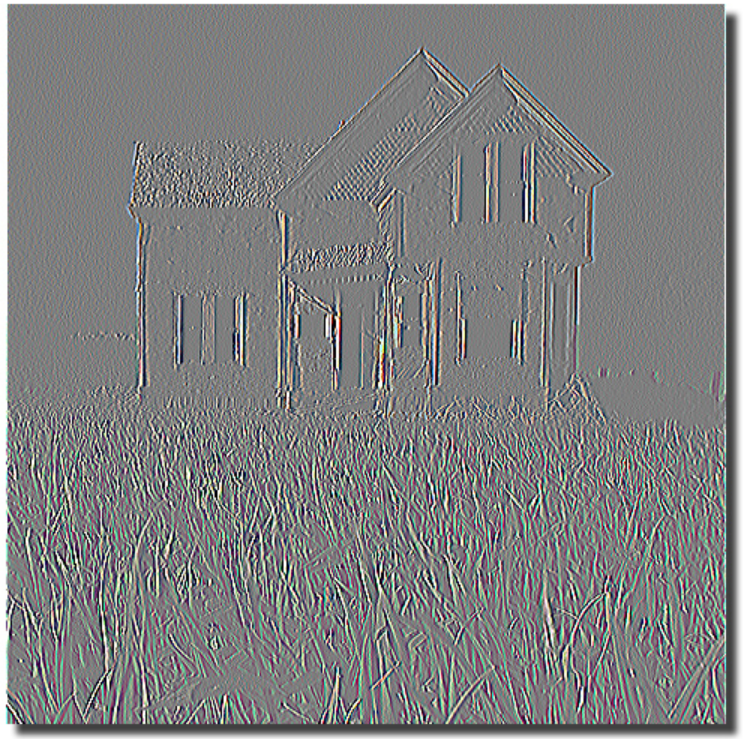




Convolution Examples: Horizontal Difference

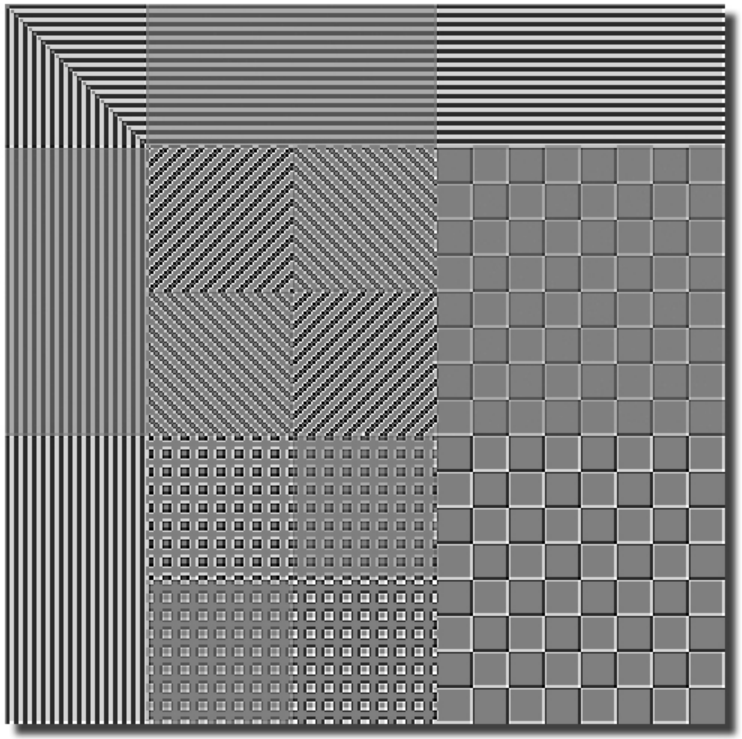
$$\begin{bmatrix} -1 & 2 & -1 \end{bmatrix}$$

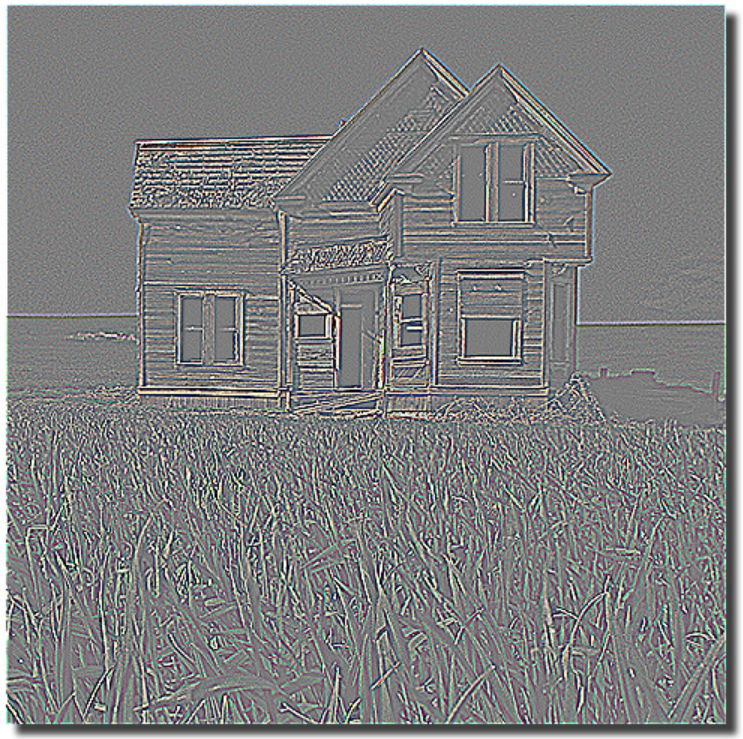




Convolution Examples: H + V Diff.

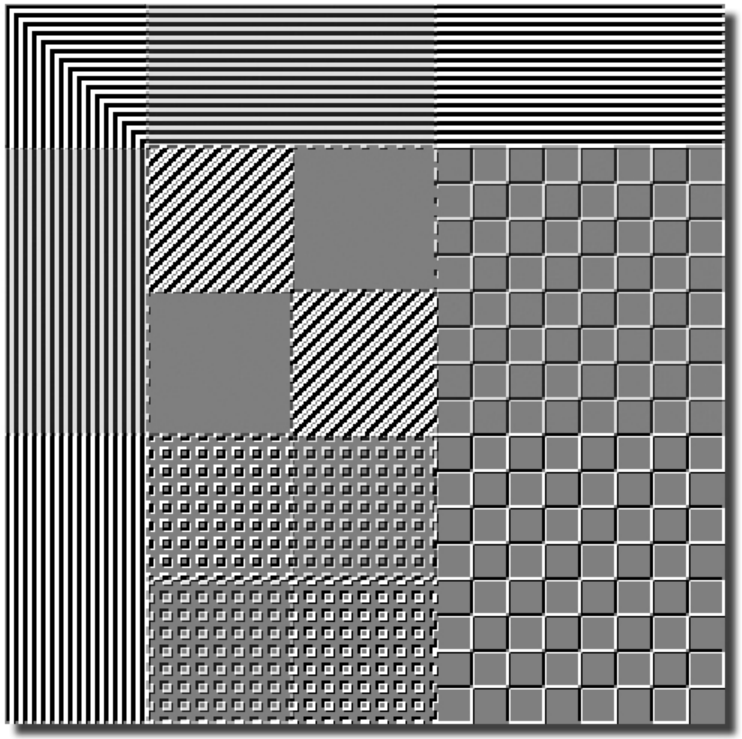
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

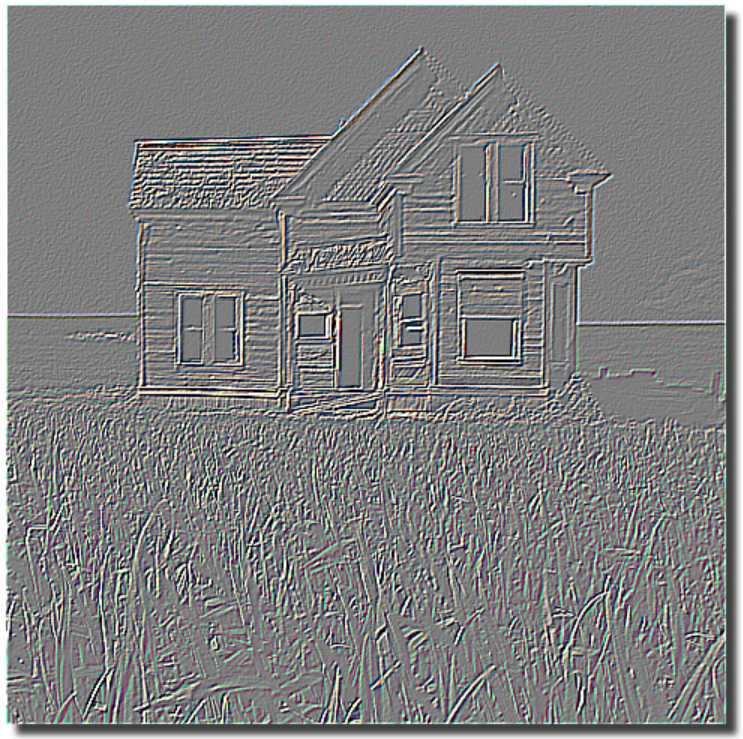




Convolution Examples: Diagonal Difference

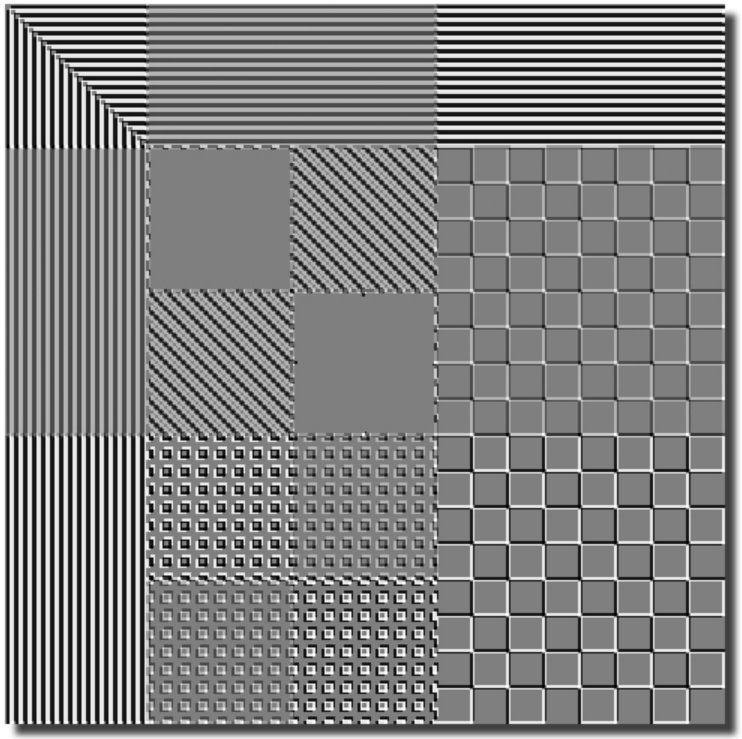
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$





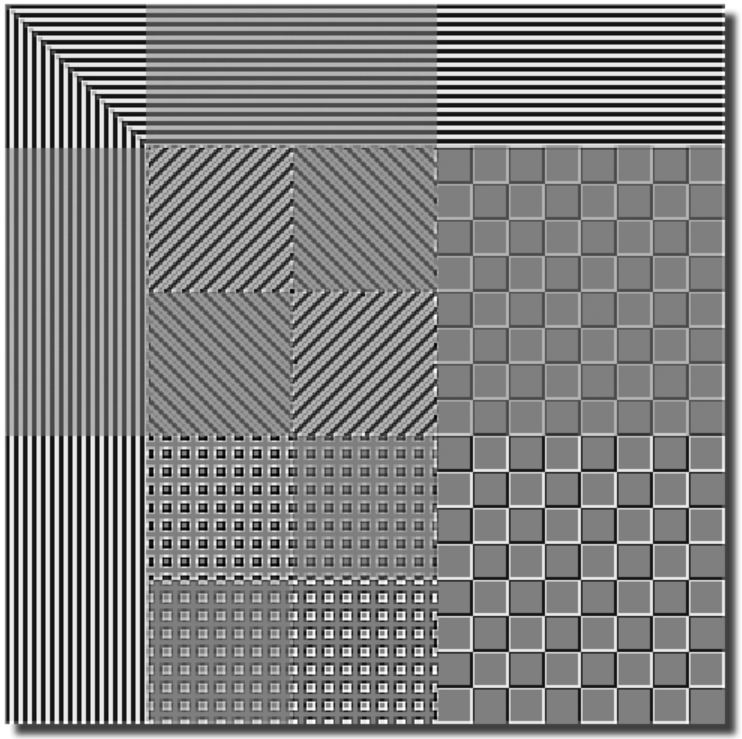
Convolution Examples: Diagonal Difference

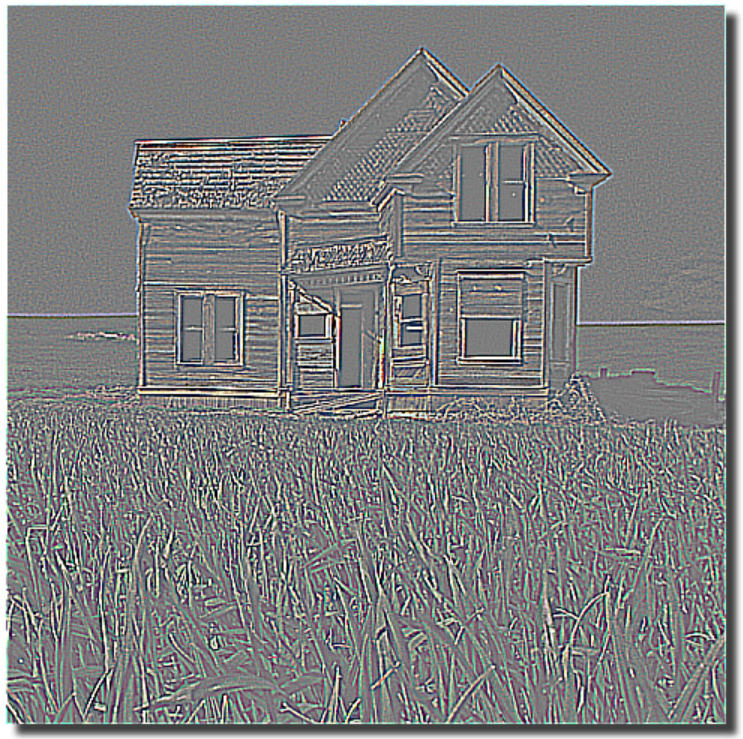
$$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 2 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$



Convolution Examples: D + D Difference

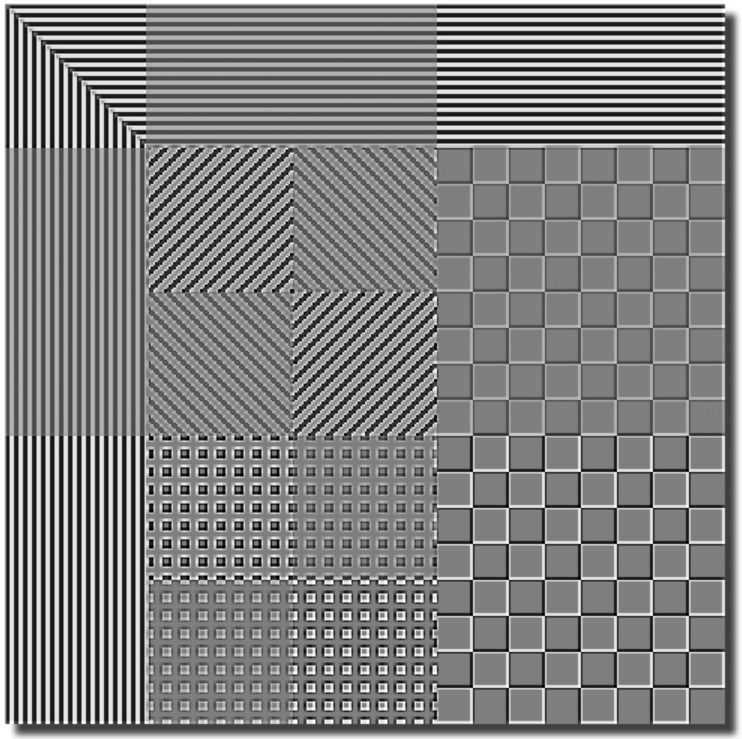
$$\begin{bmatrix} -1 & 0 & -1 \\ 0 & 4 & 0 \\ -1 & 0 & -1 \end{bmatrix}$$

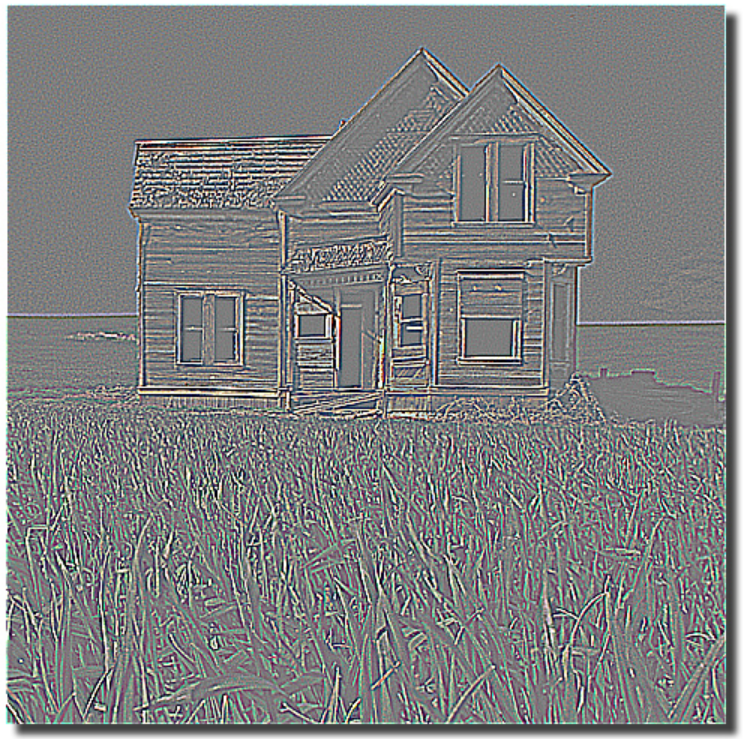




Convolution Examples: H + V + D Diff.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$





The Median Filter

- Returns the median value of the pixels in a neighborhood
- Is non-linear
- Is similar to a uniform blurring filter which returns the mean value of the pixels in a neighborhood of a pixel
- Unlike a mean value filter the median tends to preserve step edges

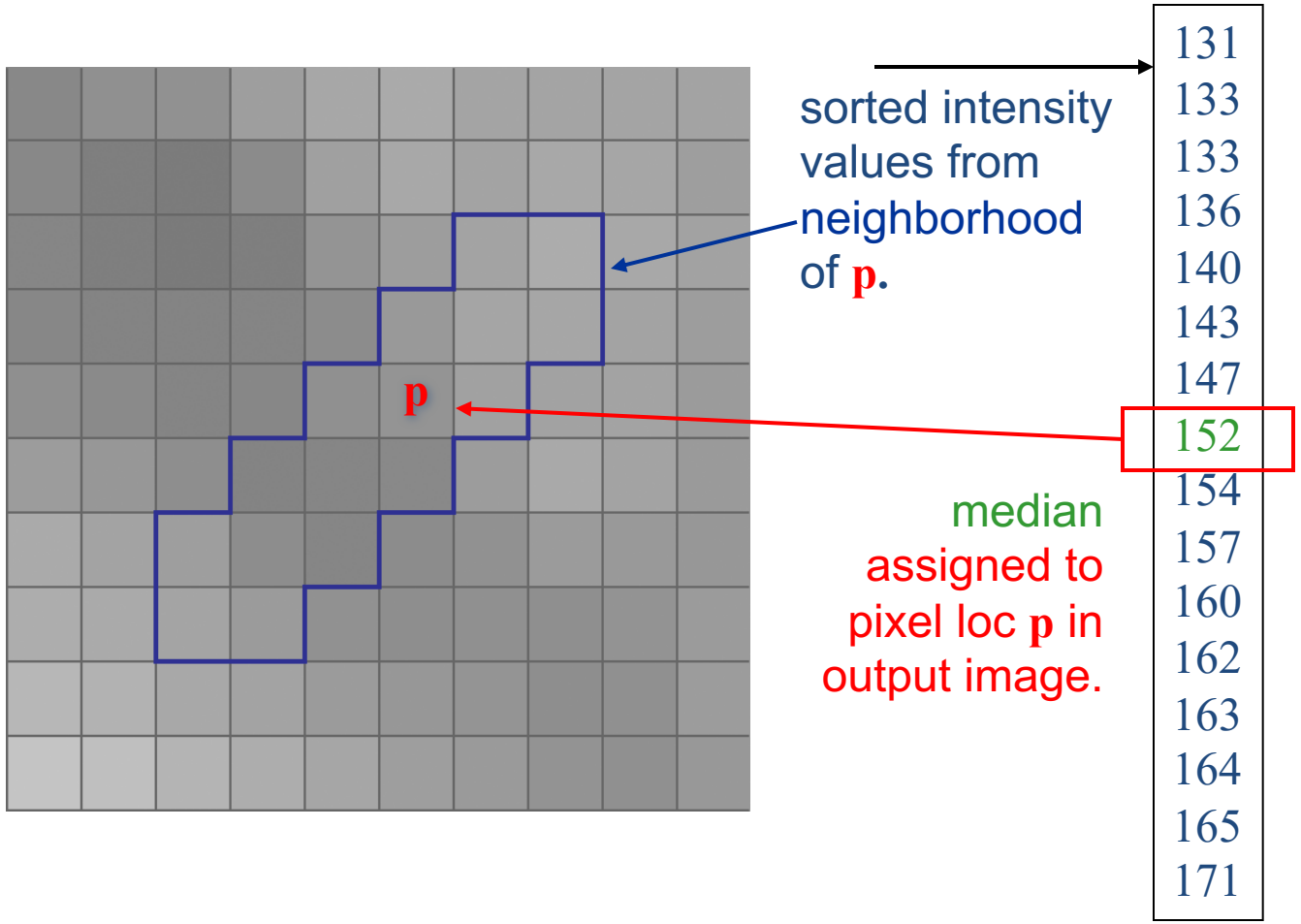
Median Filter: General Definition

$$\text{med}\{\mathbf{I},\mathbf{Z}\}(\mathbf{p}) = \underset{\mathbf{q}\in\text{supp}(\mathbf{Z}+\mathbf{p})}{\text{median}}\{\mathbf{I}(\mathbf{q})\}$$

This can be computed as follows:

1. Let **I** be a monochrome (1-band) image.
2. Let **Z** define a neighborhood of arbitrary shape.
3. At each pixel location, **p** = (*r*,*c*), in **I** …
4. … select the *n* pixels in the **Z**-neighborhood of **p**,
5. … sort the *n* pixels in the neighborhood of **p,** by value, into a list $L(j)$ for $j = 1,\ldots,n$.
6. The output value at **p** is $L(m)$, where $m = \lfloor \frac{n}{2} \rfloor + 1$

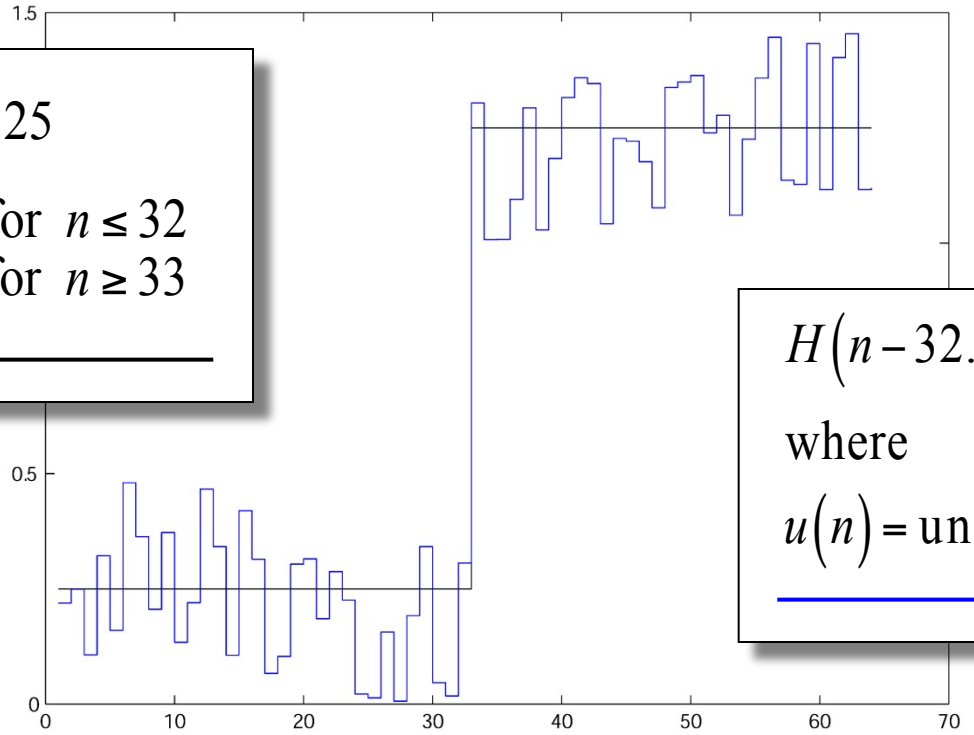
Median Filter: General Definition



A Noisy Step Edge

$$H(n-32.5)+0.25$$

$$= \begin{cases} 0.25 & \text{for } n \le 32 \\ 1.25 & \text{for } n \ge 33 \end{cases}$$



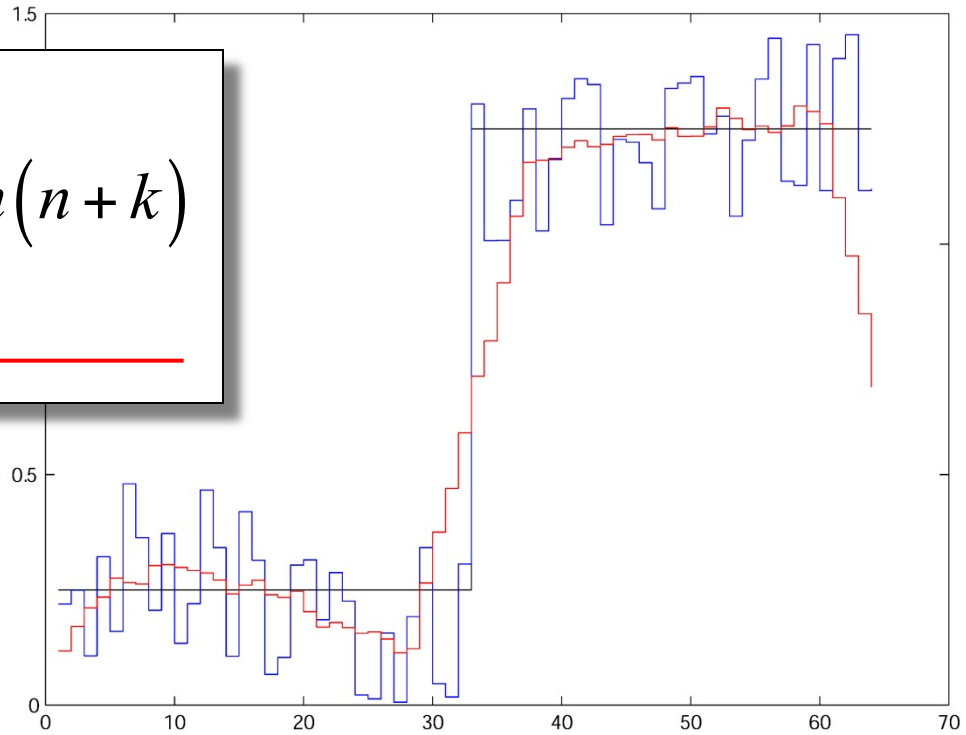
$$H(n-32.5)+u(n)$$

where

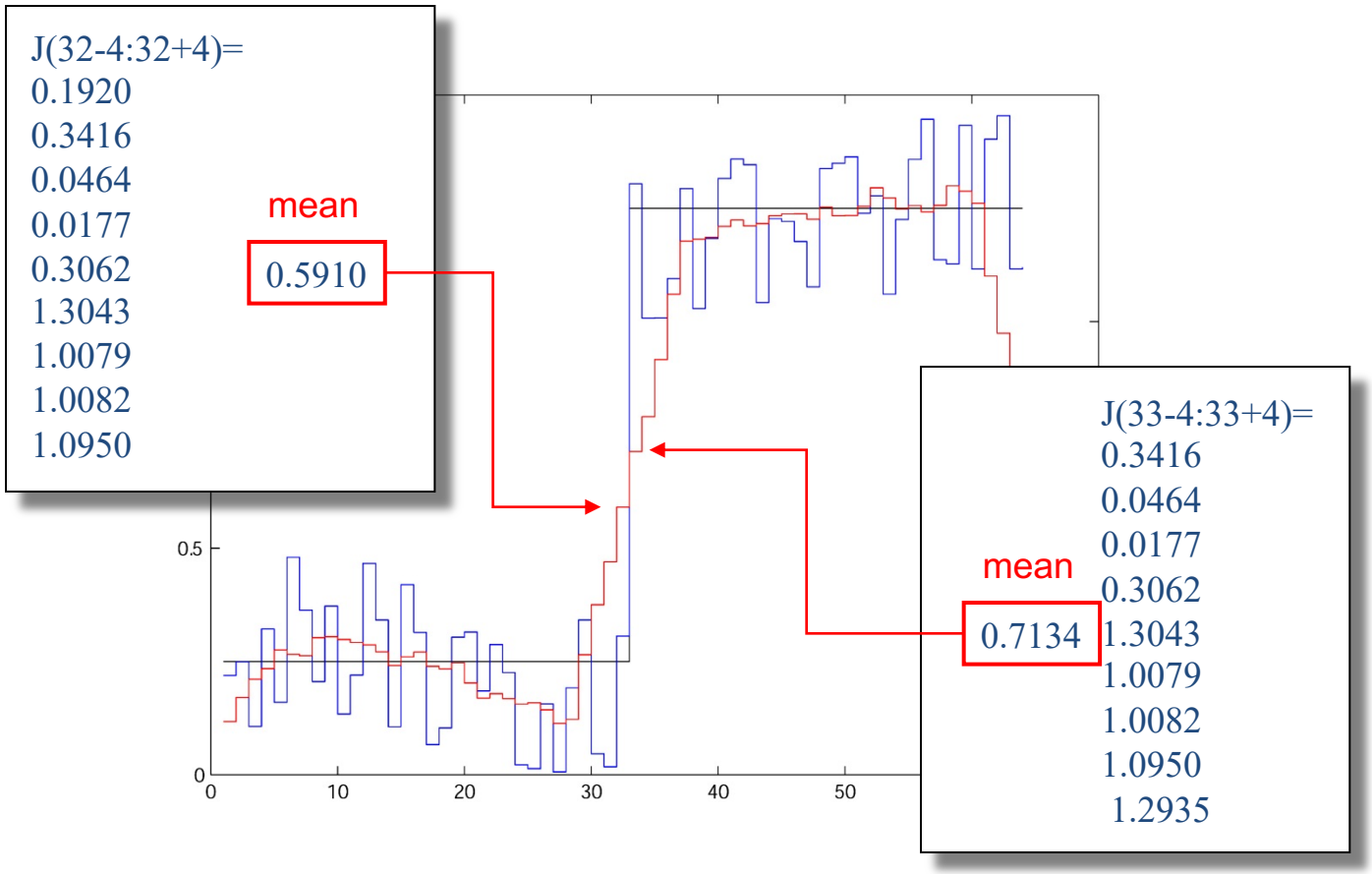
$$u(n) = \text{unif}(-0.25, 0.25)$$

Blurred Noisy 1D Step Edge

$$h(n) = \frac{1}{9}\sum_{k=-4}^{4} h(n+k)$$

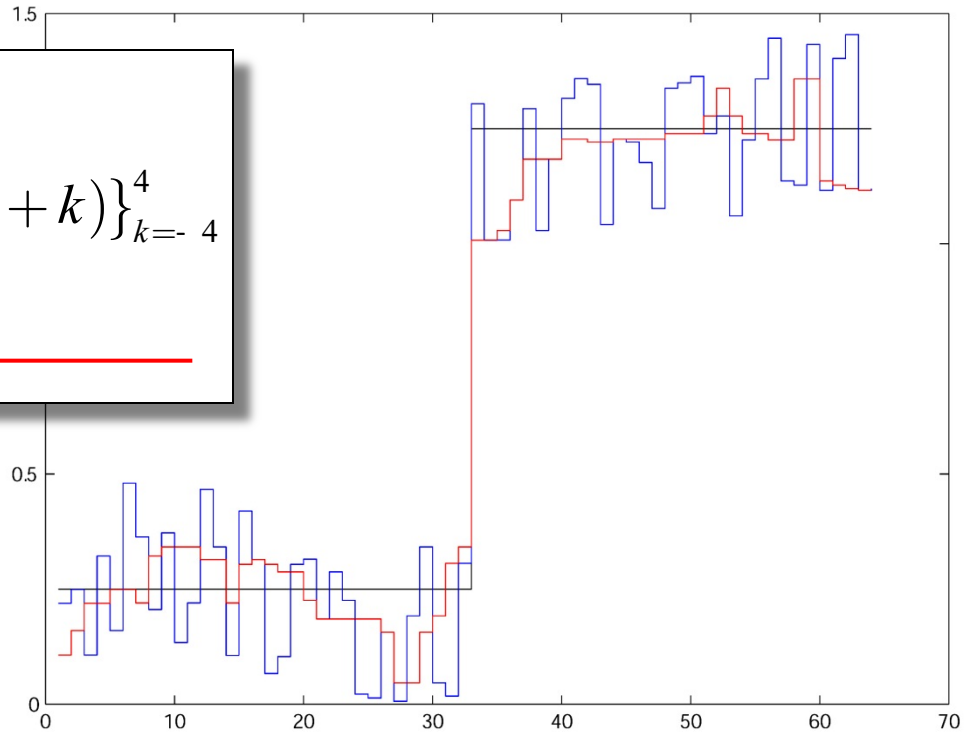


Blurred Noisy 1D Step Edge

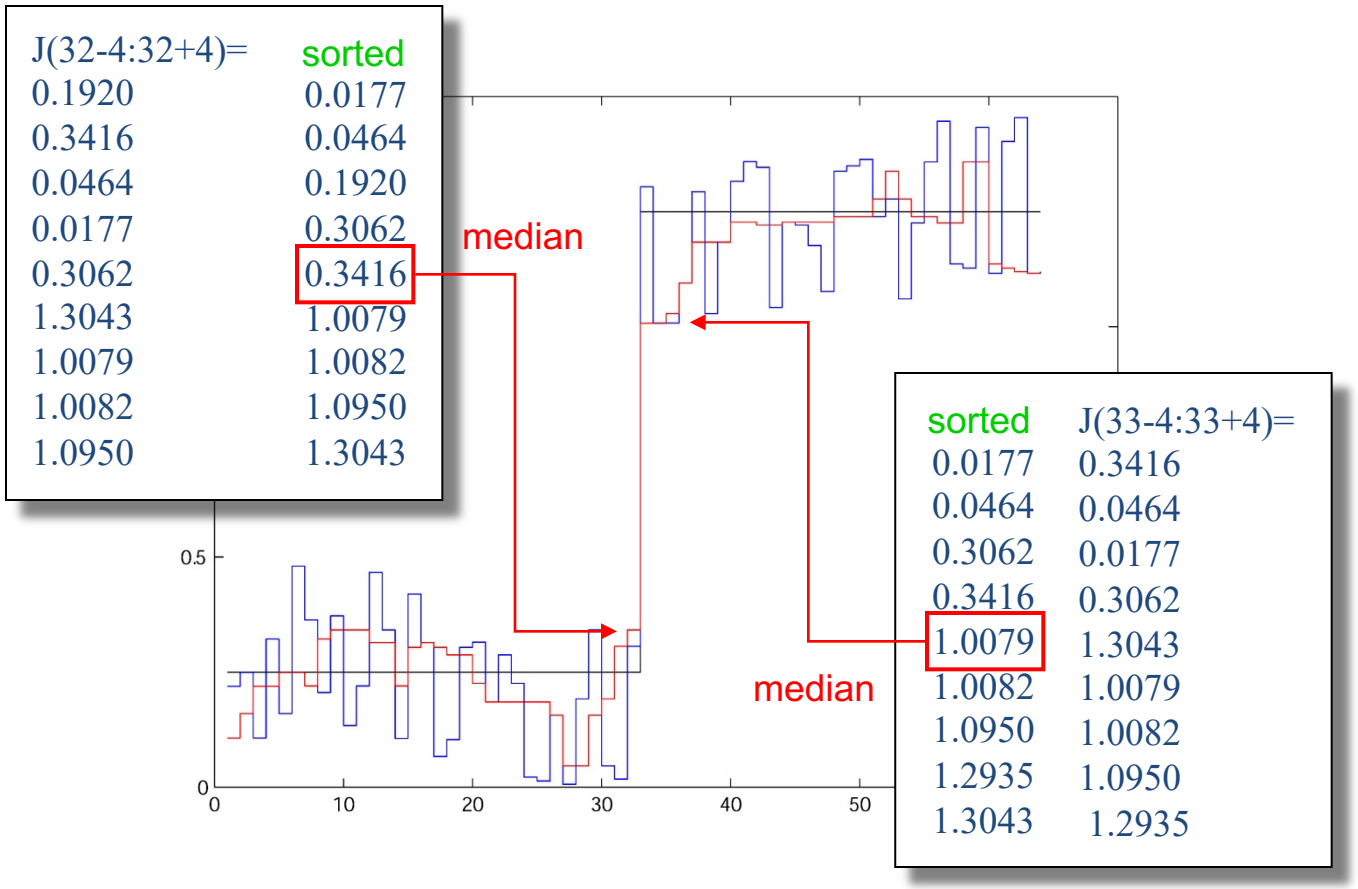


Median Filtered Noisy 1D Step Edge

$$h(n) = \text{med}\{h(n+k)\}_{k=-4}^{4}$$

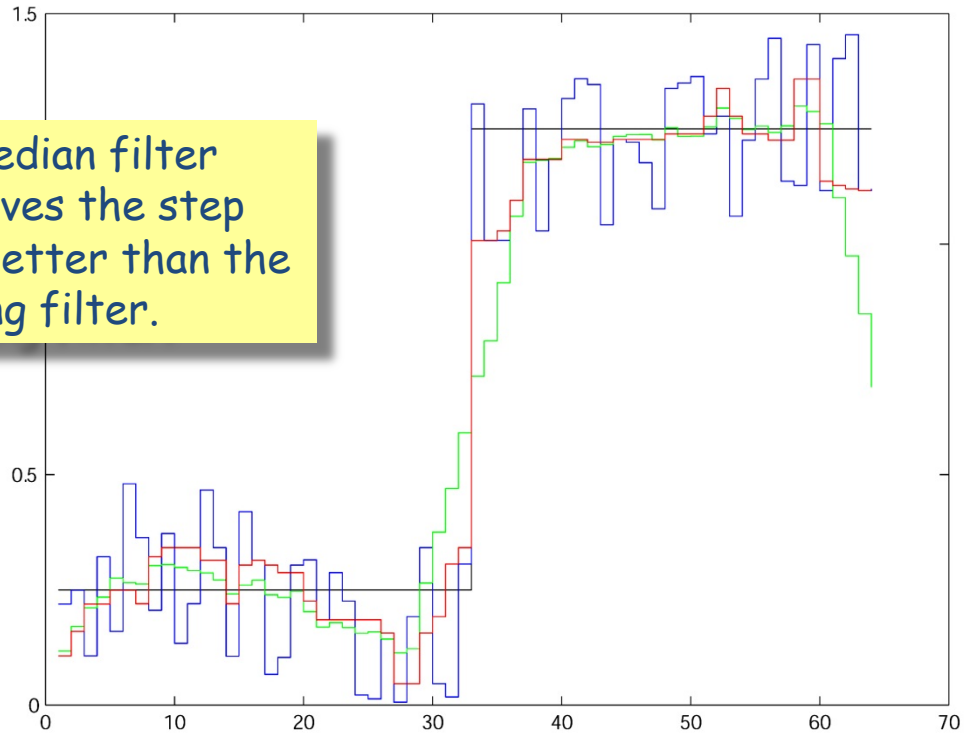


Median Filtered Noisy 1D Step Edge



Median vs. Blurred

The median filter preserves the step edge better than the blurring filter.

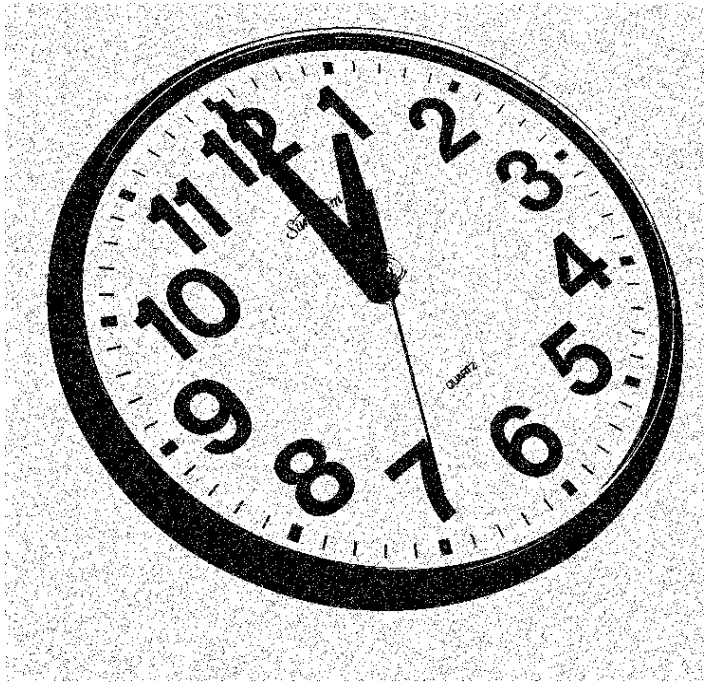


median

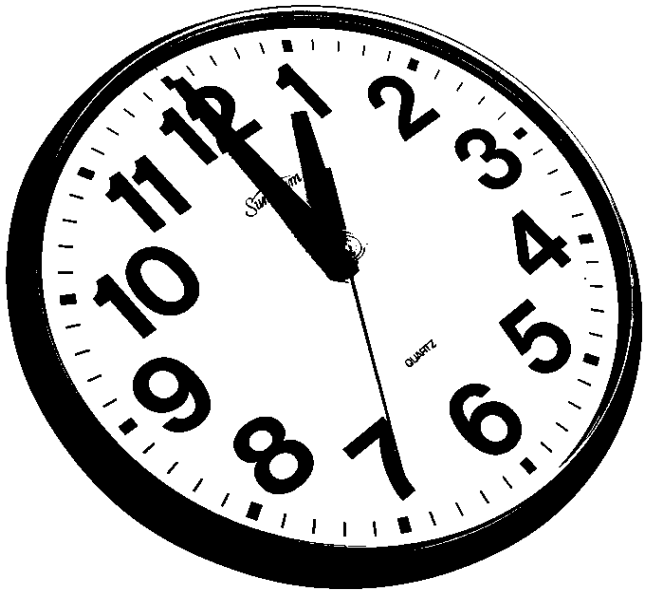
blurred

noisy

Median Filtering of Binary Images

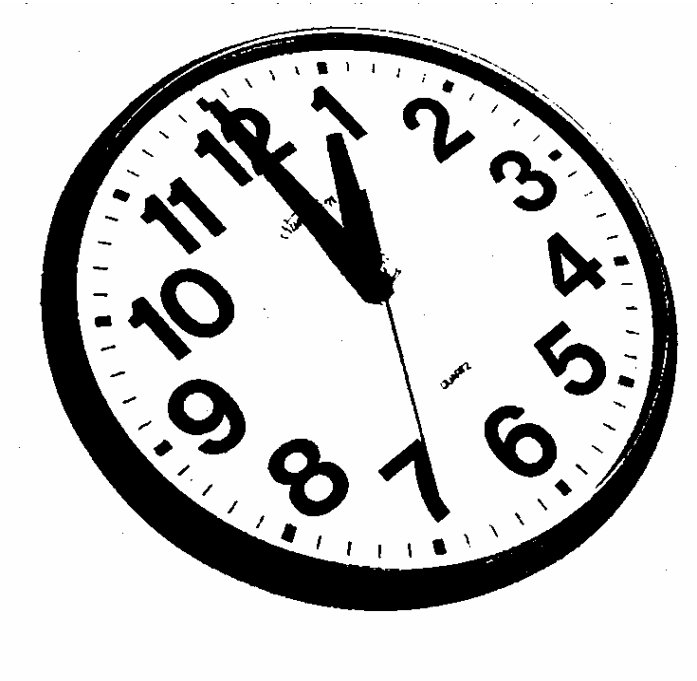


Noisy

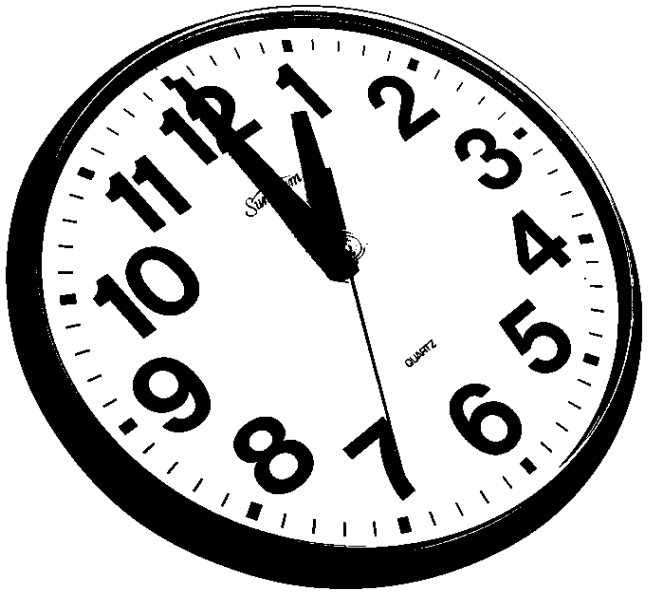


Original

Median Filtering of Binary Images

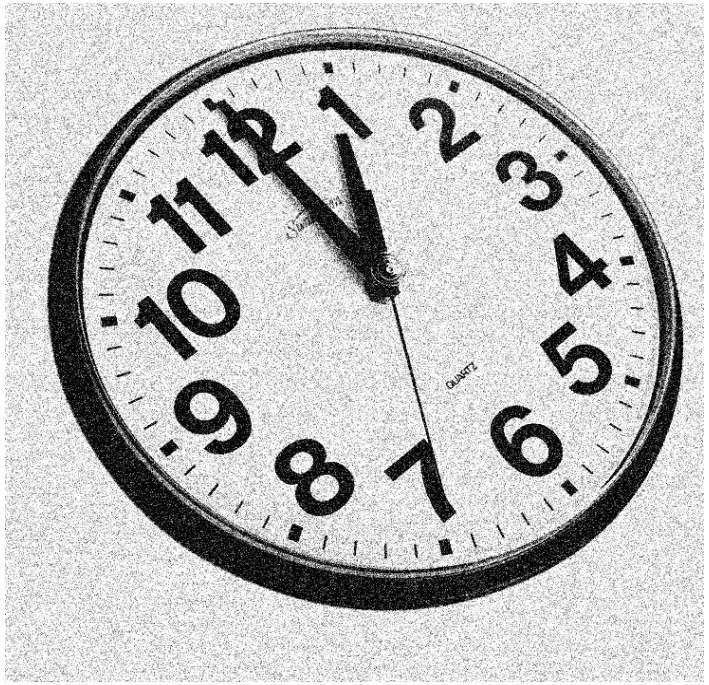


Median Filtered Noisy



Original

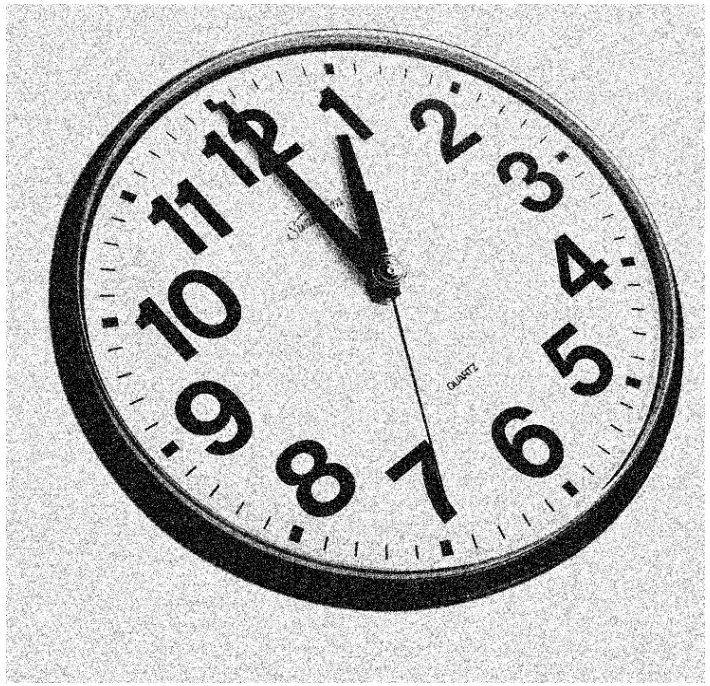
Filtering of Grayscale Images



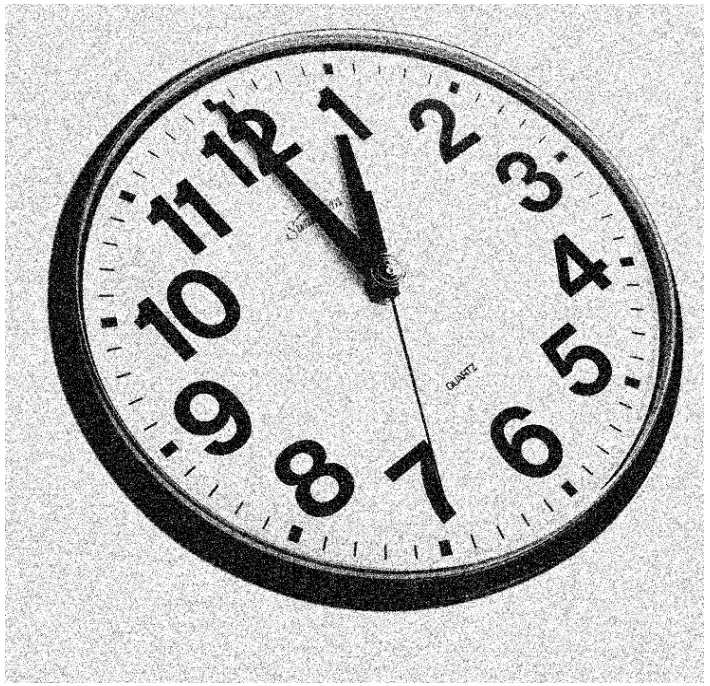
Noisy

Original

Filtering of Grayscale Images

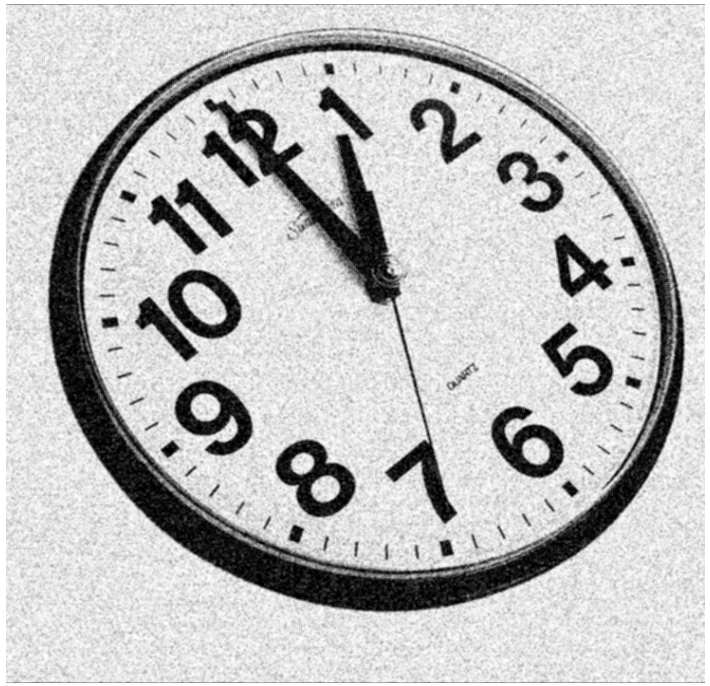


Noisy

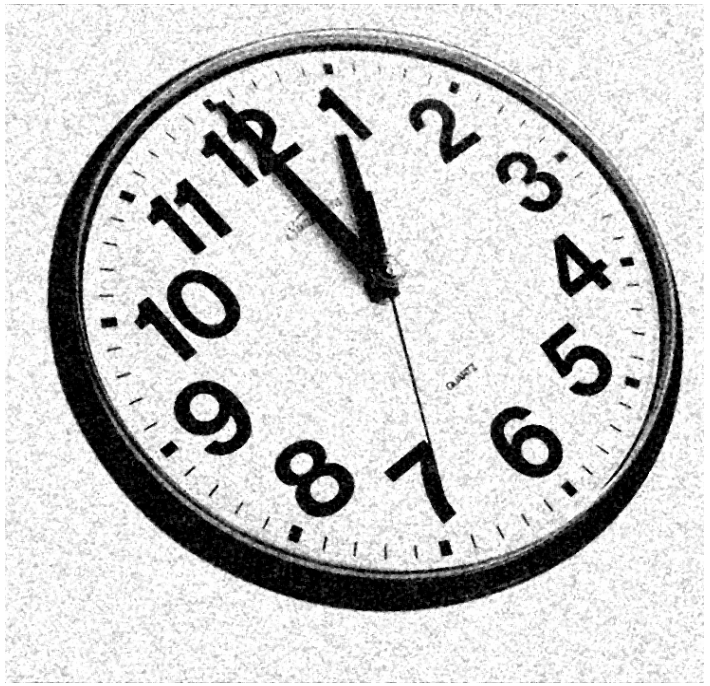


Noisy

Filtering of Grayscale Images

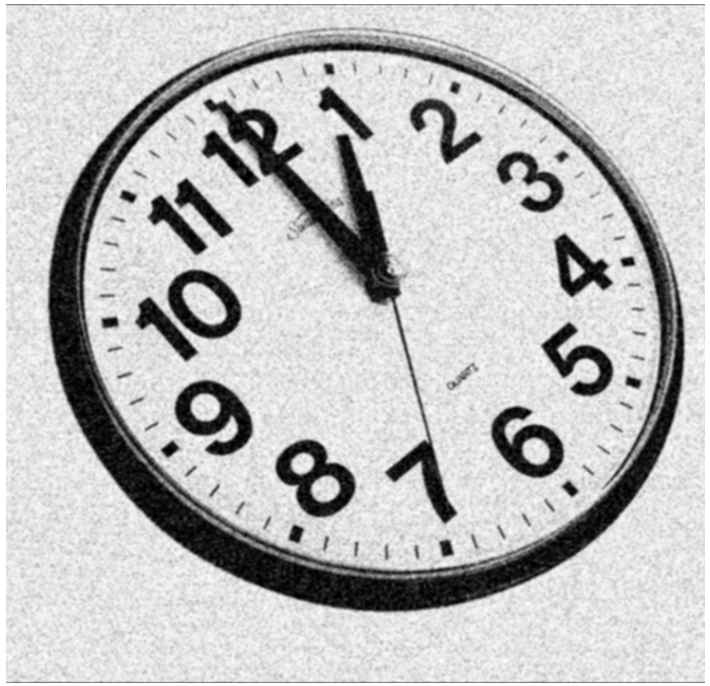


3x3-blur x 1

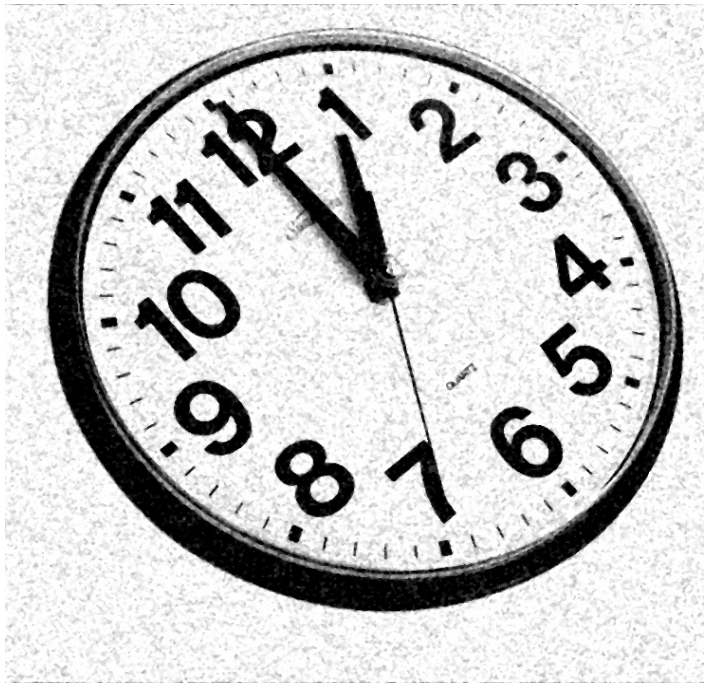


3x3-median x 1

Filtering of Grayscale Images

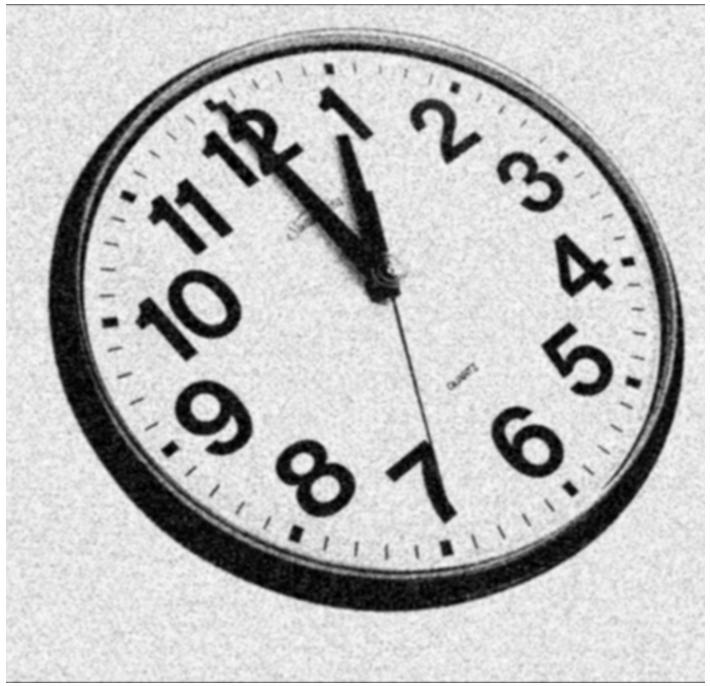


3x3-blur x 2

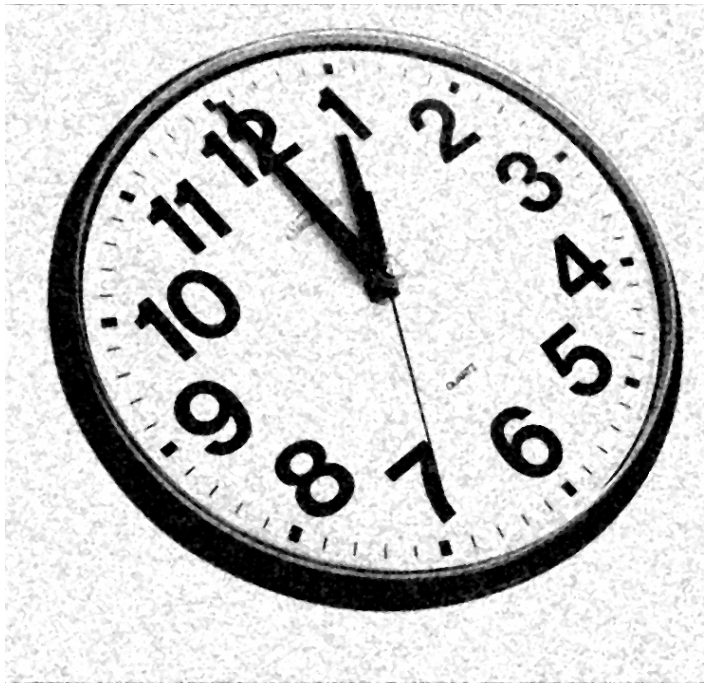


3x3-median x 2

Filtering of Grayscale Images

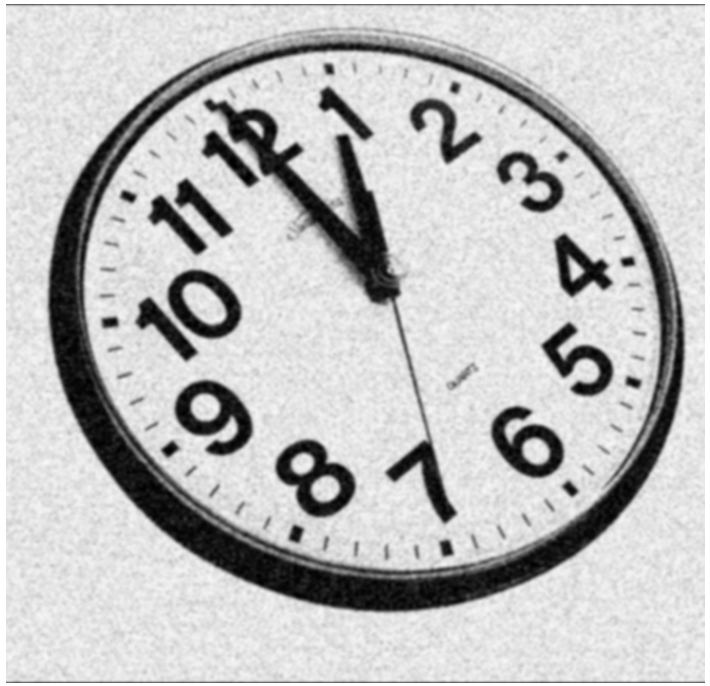


3x3-blur x 3

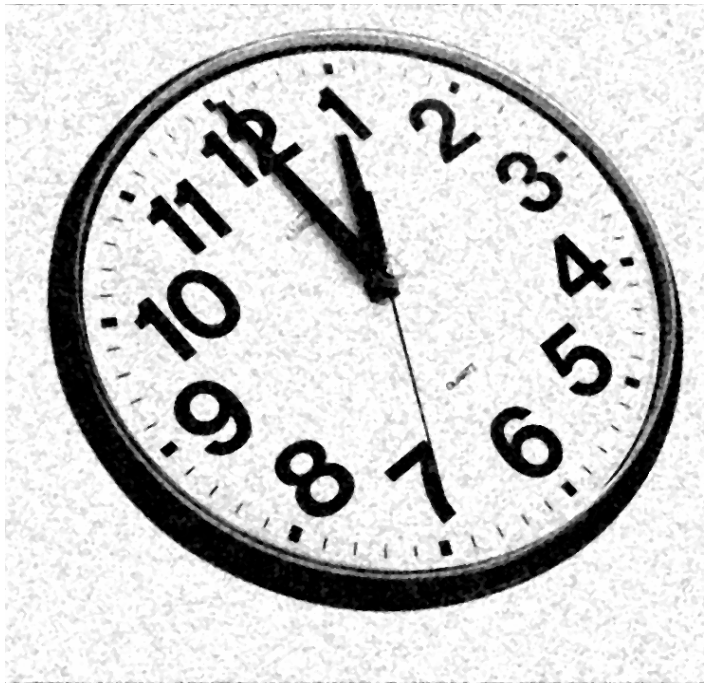


3x3-median x 3

Filtering of Grayscale Images

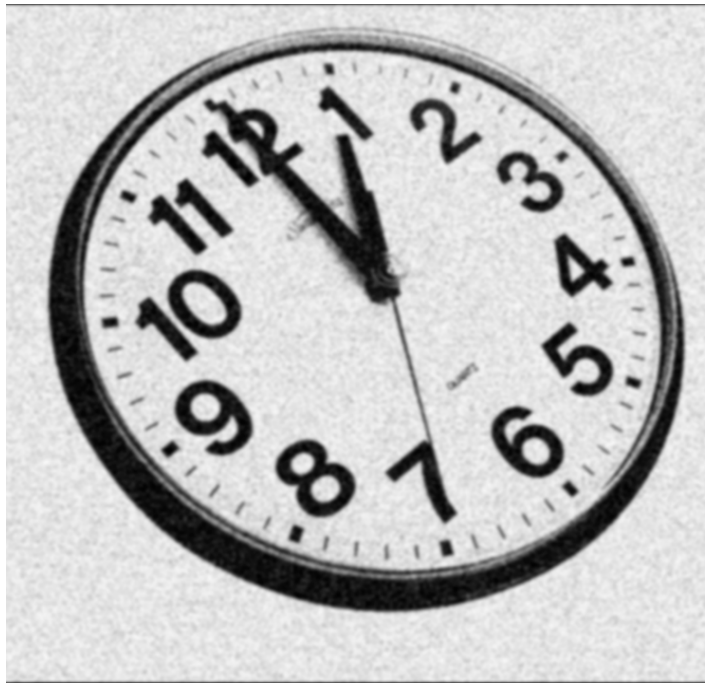


3x3-blur x 4

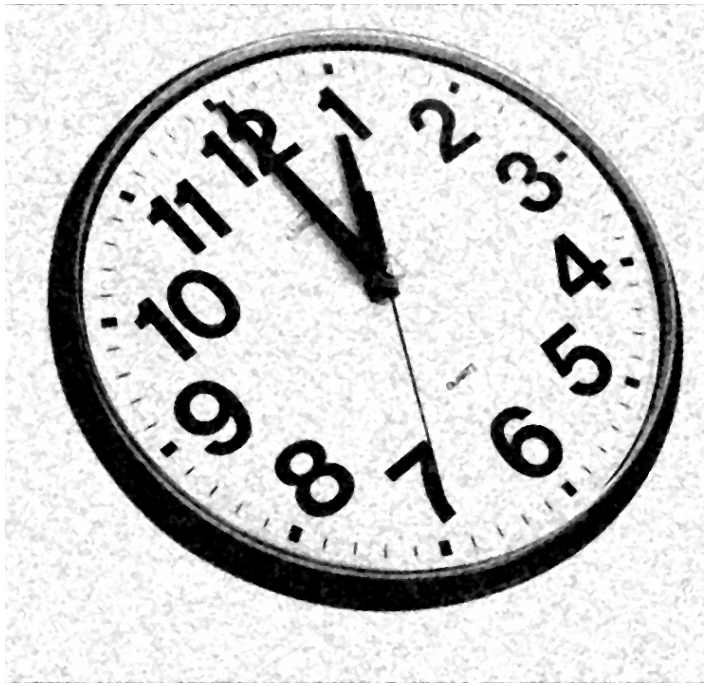


3x3-median x 4

Filtering of Grayscale Images



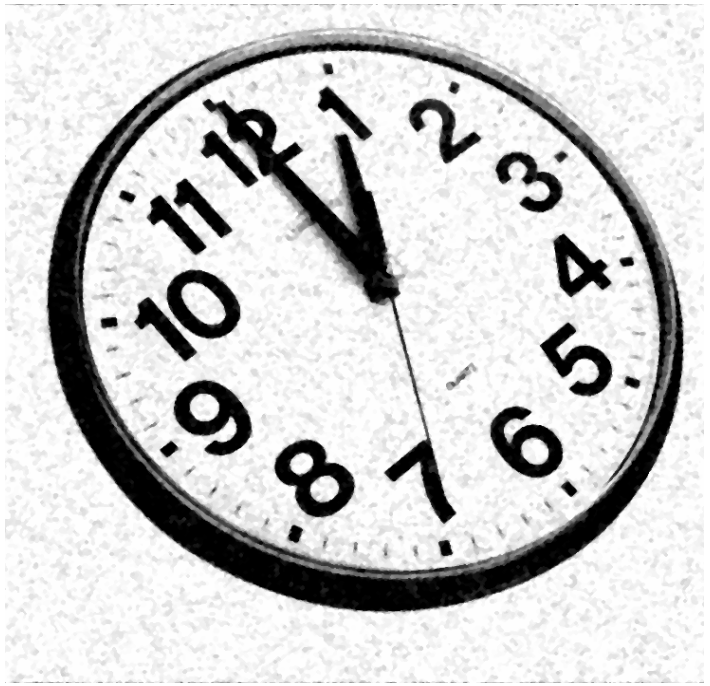
3x3-blur x 5



3x3-median x 5

Filtering of Grayscale Images

3x3-blur x 10



3x3-median x 10

Limit and Root Images

Fact: if you repeatedly filter an image with the same blurring filter or median filter, eventually the output does not change. That is, let

$$\mathbf{I}\left[*\mathbf{h}\right]^k \equiv \left(\left(\left(\mathbf{I} * \mathbf{h}\right) * \mathbf{h}\right) \cdots * \mathbf{h}\right), \quad k \text{ times, and}$$

$$\mathbf{I}\left[\text{med } \mathbf{Z}\right]^k \equiv \left(\left(\left(\mathbf{I} \text{ med } \mathbf{Z}\right) \text{ med } \mathbf{Z}\right) \cdots \text{med } \mathbf{Z}\right), \quad k \text{ times.}$$

Then

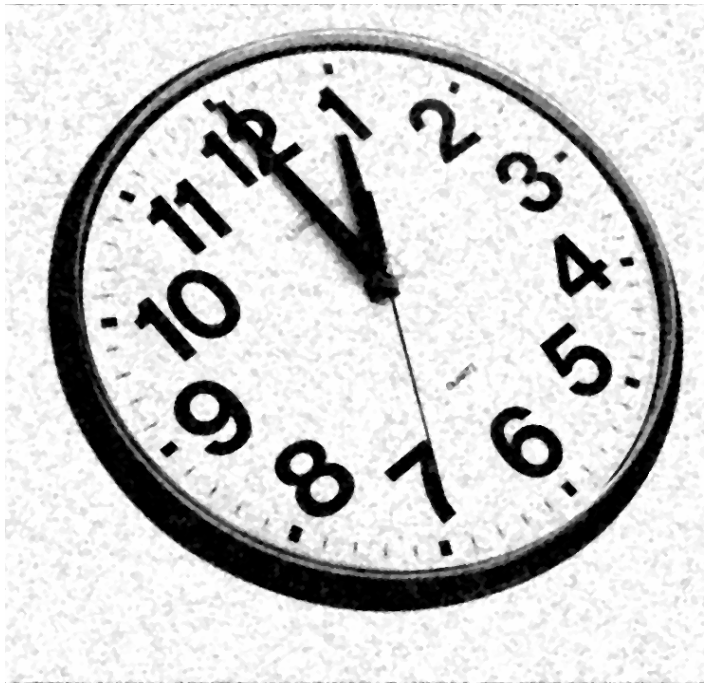
$$\lim_{k\to\infty} \mathbf{I}\left[*\mathbf{h}\right]^k = \mathbf{I}\left[*\mathbf{h}\right]^n = \mathbf{I}_0, \quad \text{and}$$

$$\lim_{k\to\infty} \mathbf{I}\left[\text{med } \mathbf{Z}\right]^k = \mathbf{I}\left[\text{med } \mathbf{Z}\right]^m = \mathbf{I}_r,$$

where n and m are integers ($< \infty$) , $\mathbf{I}_0$ is a single-valued image and $\mathbf{I}_r$ is called the *median root* of $\mathbf{I}$.

Limit and Root Images

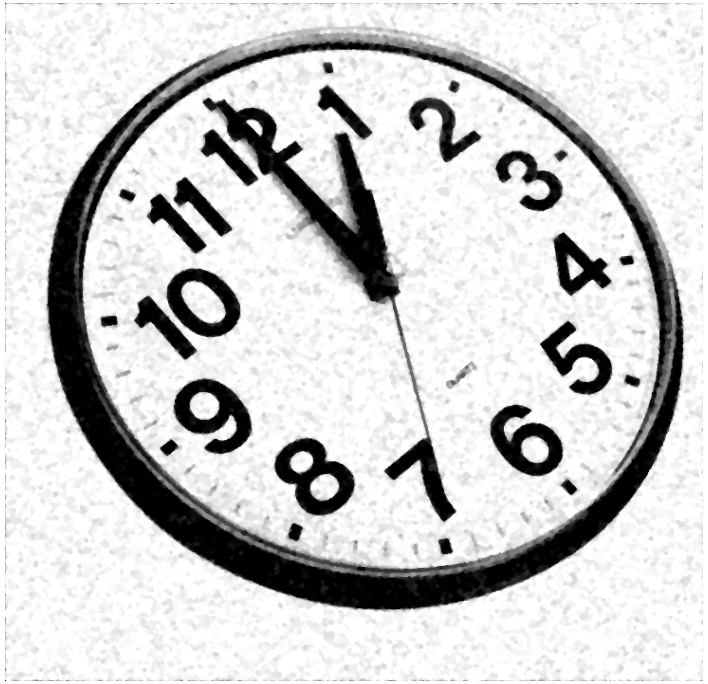
3x3-blur x 10



3x3-median x 10

Limit and Root Images

3×3-blur $\times n \to \infty$



3x3-median root

Median Filter Algorithm in Matlab

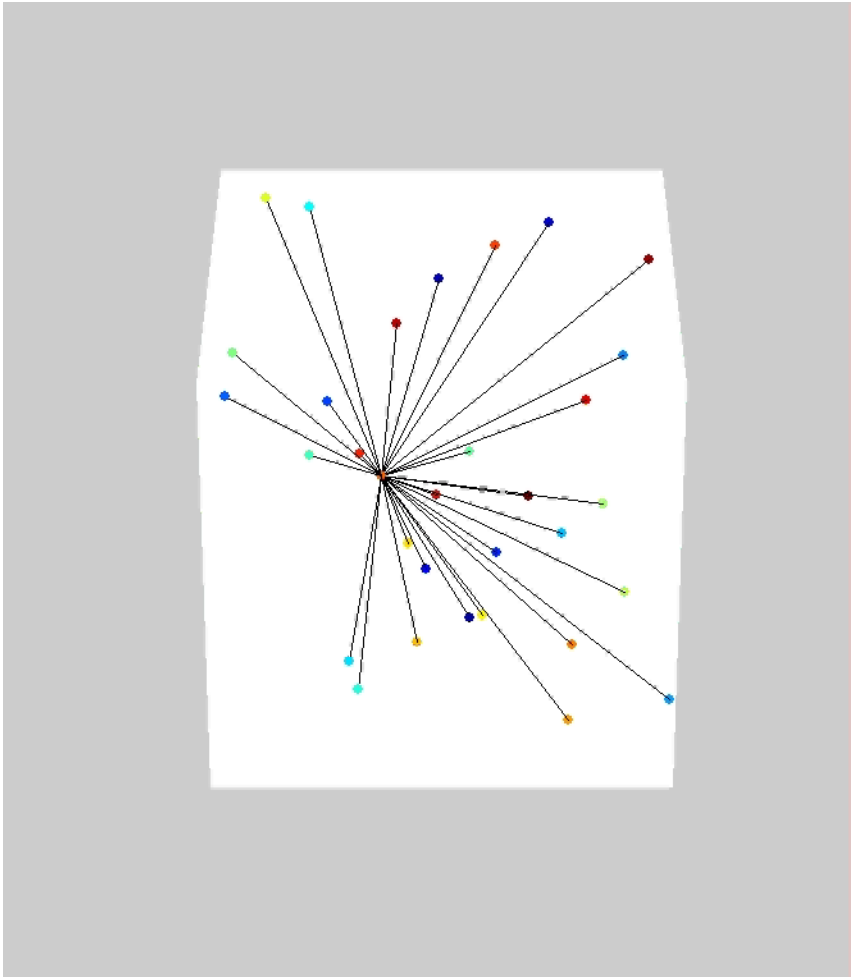
```
function D = median_filt(I,SE,origy,origx)
[R,C] = size(I);     % assumes 1-band image
[SER,SEC] = size(SE); % SE < 0 not in nbhd

N = sum(sum(SE>=0));  % no. of pixels in nbhd
A = -ones(R+SER-1,C+SEC-1,N); % accumulator
n=1; % copy I into band n of A for nbhd pix n
for j = 1 : SER % neighborhood is def'd in SE
   for i = 1 : SEC
      if SE(j,i) >= 0 % then is a nbhd pixel
         A(j:(R+j-1),i:(C+i-1),n) = I;
         n=n+1; % next accumulator band
      end
   end
end
% pixel-wise median across the bands of A
A = shiftdim(median(shiftdim(A,2)),1);
D = A( origy:(R+origy-1) , origx:(C+origx-1) );
return;
```

Vector Median Filter

A vector median filter selects from among a set of vectors, the one vector that is closest to all the others.

That is, if S is a set of vectors, in $\mathbb{F}^n$ the median, $\overline{\mathbf{v}}$, is

$$\overline{\mathbf{v}} = \arg\min_{k \neq j} \left\{ \left\| \mathbf{v}_k - \mathbf{v}_j \right\| \,\middle|\, \mathbf{v}_k, \mathbf{v}_j \in S \right\}.$$



$\trianglelefteq \mathbb{F}^n$ is an n-dimensional linear vector space over the field, $\mathbb{F}$.)

Color Median Filter

Jim Woodring – A Warm Shoulder

Sparse noise, 32% coverage in each band

Color Median Filter

3 × 3 color median filter applied once

3 × 3 color median filter applied twice

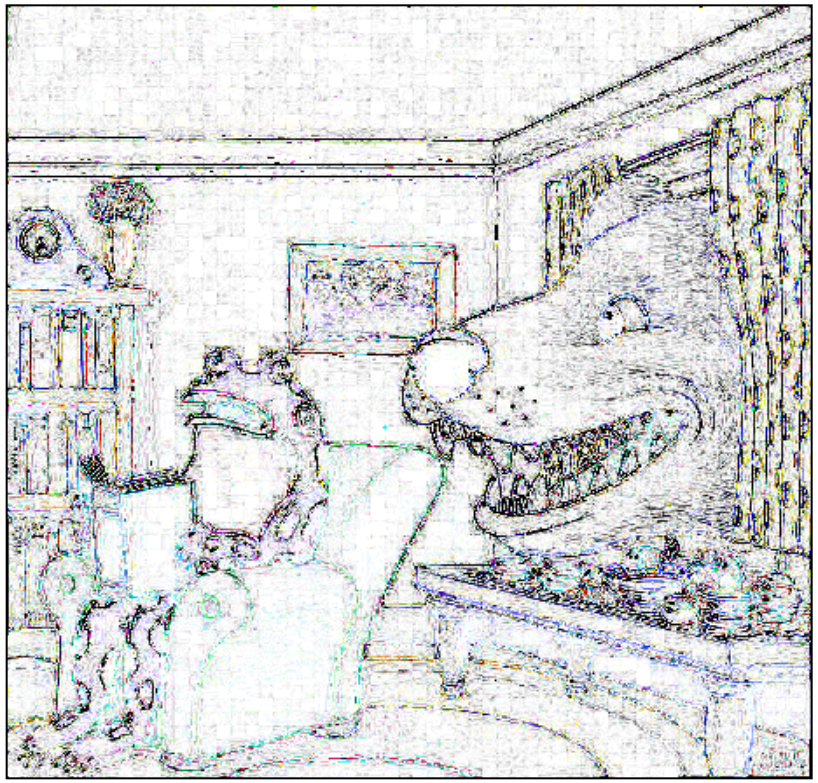
Color Median Filter

Sparse noise, 32% coverage in each band

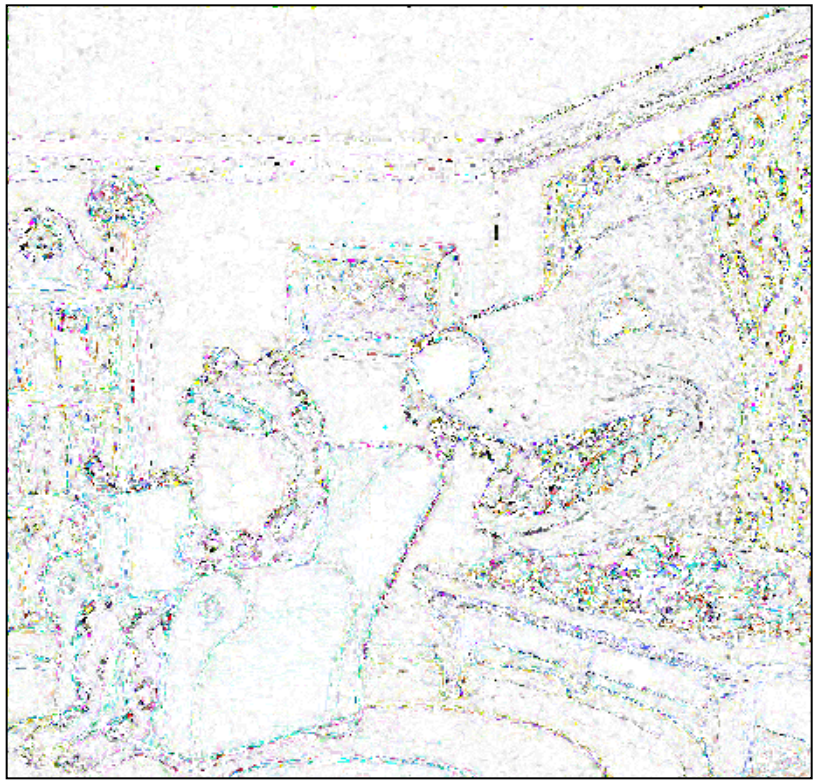
Jim Woodring – A Warm Shoulder

Color Median Filter

Absolute differences displayed as negatives to enhance visibility



(3 × 3 CMF^2 of noisy) – original



(3 × 3 CMF^2 of noisy) – (3 × 3 CMF^2 of original)

CMF vs. Standard Median on Individual Bands

A color median filter has to compute the distances between all the color vectors in the neighborhood of each pixel. That's expensive computationally.

Q: Why not simply take the 1-band median of each color band individually?

A: The result at a pixel could be a color that did not exist in the pixel's neighborhood in the input image. The result is not the median of the colors – it is the median of the intensities of each color band treated independently.

Q: Is that a problem?

A: Maybe. Maybe not. It depends on the application. It may make little difference visually. If the colors need to be preserved, it could be problematic.

CMF vs. Standard Median on Individual Bands

Jim Woodring – A Warm Shoulder

Sparse noise, 32% coverage in each band

CMF vs. Standard Median on Individual Bands

3 × 3 color median filter applied once

3 × 3 color median filter applied twice

CMF vs. Standard Median on Individual Bands

3 × 3 median filter applied to each band once

3 × 3 median filter applied to each band twice