EECE 4353 Image Processing

Lecture Notes: Image Convolution – Spatial

Richard Alan Peters II

Department of Electrical Engineering and
Computer Science

Fall Semester 2016

Point Operators & Neighborhood Operators

are the two major categories of image transformation. A point process changes the value of a pixel as a function only of its current value. *I.e.*

$$f : \mathbb{R} \to \mathbb{R} \quad \text{such that} \quad \mathbf{I}_{\text{out}}(r,c) = f\left(\mathbf{I}_{\text{in}}(r,c)\right).^{1}$$

Neither the distribution of values nor the metric structure of an image has any effect whatsoever on the result of the operation. Spatially dependent processing requires an operator whose output at a point is a function of the area surrounding it – a neighborhood.

$$\Phi : \mathbb{R}^2 \to \mathbb{R} \quad \text{such that} \quad \mathbf{I}_{\text{out}}(r,c) = \Phi\left(\mathfrak{N}_{M\times N}\left\{\mathbf{I}_{\text{in}};(r,c)\right\}\right).$$

That is, Φ operates on the $M \times N$ square of pixels in $\mathbf{I}_{\text{in}}$ centered on (r,c) to produce a single output value for $\mathbf{I}_{\text{out}}$ at location (r,c).

[1] if $\mathbf{I}_{\text{in}}$ is a color image then the mapping is from $\mathbb{R}^3$ to $\mathbb{R}^3$.

Spatial Filtering

Neighborhood operators and moving window transforms such as convolutions and morphological operators (cf. Lectures 17 & 18) are spatial filters. Here is a more specific statement of the def:

Let $\mathbf{I}$ and $\mathbf{J}$ be images such that $\mathbf{J} = \Phi[\mathbf{I}]$. Then on an $M \times N$ rectangular window (M&N both odd),

$$\begin{aligned}\mathbf{J}(r,c) &= \Phi[\mathbf{I}](r,c) \\ &= \Phi\left[\left\{\mathbf{I}(\rho,\chi) \,\middle|\, \rho \in \{r-m,\ldots,r,\ldots,r+m\},\ \chi \in \{c-n,\ldots,c,\ldots,c+n\}\right\}\right].\end{aligned}$$

That is, the value of the transformed image, $\mathbf{J}$, at pixel location (r,c) is a function of the values of the original image, $\mathbf{I}$, in an $M \times N = (2m+1) \times (2n+1)$ rectangular neighborhood centered on pixel location (r,c).

Image Neighborhood & Shift Invariance

In image, $\mathbf{I}$, a neighborhood (abbreviated nbhd) of pixel location (r,c) is a rectangular set of pixels that includes (r,c) and has a specific shape (as determined by its zero elements). It is the set of pixels

$$\mathfrak{N}_{M\times N}\left\{\mathbf{I};(r,c)\right\}=\left\{(\rho,\chi)\in\mathbf{I}\middle|\rho-r\in\left[-\left\lfloor\tfrac{M}{2}\right\rfloor,\left\lfloor\tfrac{M}{2}\right\rfloor\right],\chi-c\in\left[-\left\lfloor\tfrac{N}{2}\right\rfloor,\left\lfloor\tfrac{N}{2}\right\rfloor\right]\right\},$$

if M and N are odd. If M and/or N is even then

$$\rho-r\in\left[-\left\lfloor\tfrac{M}{2}\right\rfloor,\left\lfloor\tfrac{M}{2}\right\rfloor-1\right]\ \text{and/or}\ \ \chi-c\in\left[-\left\lfloor\tfrac{N}{2}\right\rfloor,\left\lfloor\tfrac{N}{2}\right\rfloor-1\right].$$

Neighborhood operator, Φ, is called "shift invariant" if both the geometry of the nbhd and the action of Φ is the same for every pixel location $(r,c)\in\mathbf{I}$. All the nbhd ops in this course will be shift invariant.

Convolution

A convolution is a linear operator[2] $\Phi[\mathbf{I};\mathbf{h}]$ denoted $\mathbf{I}*\mathbf{h}$ such that

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

for a real image ($\mathbf{I}:\mathbb{R}\times\mathbb{R}\rightarrow\mathbb{R}$). Or for a digital image ($\mathbf{I}:\mathbb{Z}\times\mathbb{Z}\rightarrow\mathbb{Z}$):

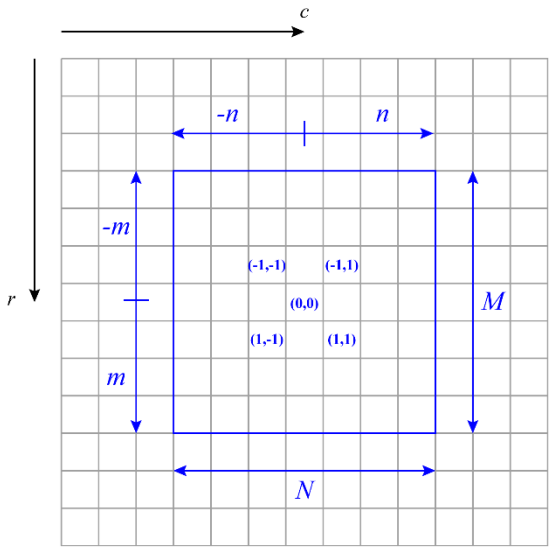
$$\mathbf{J}(r,c) = \mathbf{I}*\mathbf{h}\ (r,c) = \sum_{\rho=-m}^{m}\sum_{\chi=-n}^{n} B(r-\rho, c-\chi)\mathbf{h}(\rho,\chi)$$

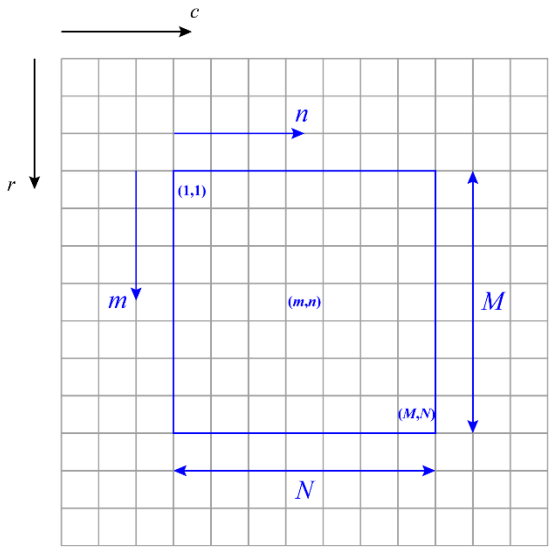
where $\mathbf{I}$ and $\mathbf{J}$ are $R\times C$ images and $\mathbf{h}$ is a $2m+1 \times 2n+1$ matrix that defines the size and shape of the convolution neighborhood.

[2]Image operator Φ is said to be linear if $\Phi[\alpha\mathbf{I}+\beta\mathbf{J}] = \alpha\Phi[\mathbf{I}]+\beta\Phi[\mathbf{J}]$, where $\alpha,\beta\in\mathbb{R}$, $\mathbf{I},\mathbf{J}$ are images.

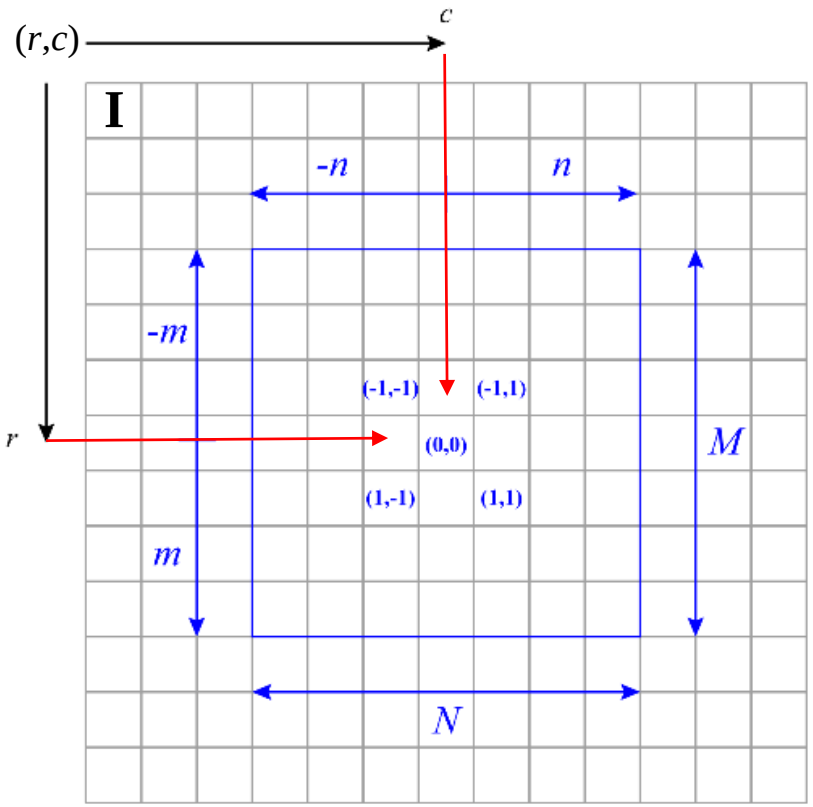
Convolution

Shift invariant neighborhood operators are sometimes called "moving window transforms" (MWT). Convolutions are linear MWTs and are the mathematical dual of multiplication in the frequency domain. Algebraically convolution is defined on a nbhd like the one on the left. Computationally convolution is performed on a nbhd like the one on the right. (Larger diagram on next slide.)

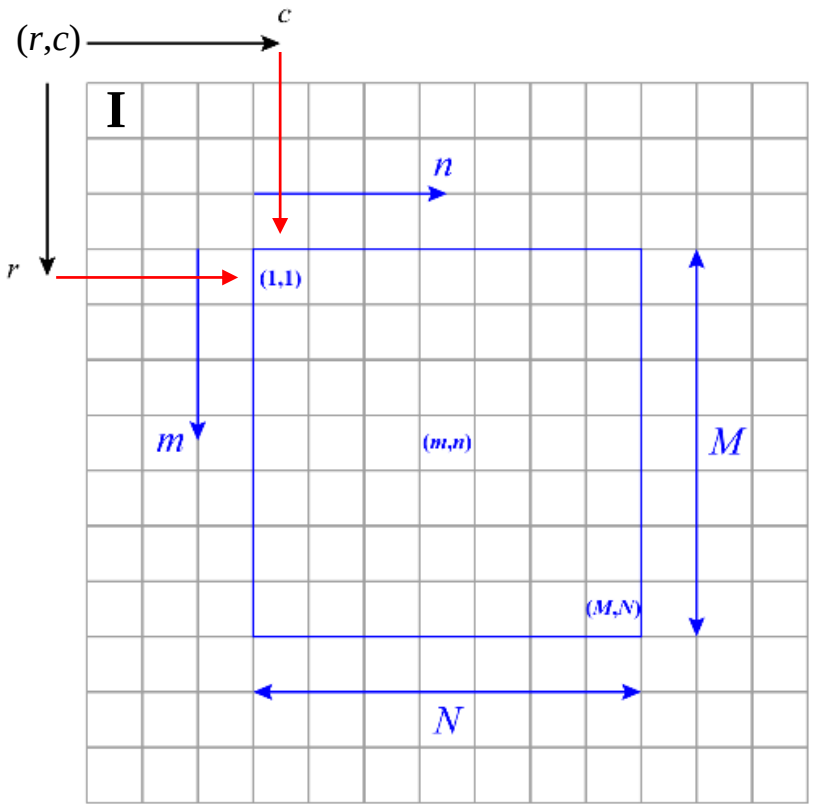




Convolution Neighborhood



used in math definition



used in computation

Boundary Effects in Convolution

If a convolution's matrix, $\mathbf{h}$, is $M \times N$ and has its origin at

$$\left(\left\lfloor \frac{M}{2} \right\rfloor + 1, \left\lfloor \frac{N}{2} \right\rfloor + 1 \right)$$

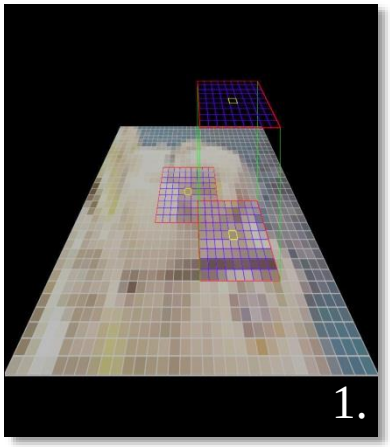
what happens when

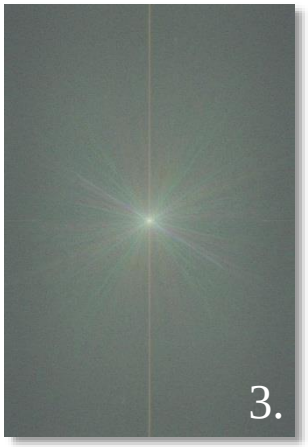
$$r < \left\lfloor \frac{M}{2} \right\rfloor + 1, \quad r > R - \left\lfloor \frac{M}{2} \right\rfloor, \quad c < \left\lfloor \frac{N}{2} \right\rfloor + 1, \text{ or } c > C - \left\lfloor \frac{N}{2} \right\rfloor ?$$

That is when r or c is close to one of the image boundaries? The convolution is indeterminate there; its value depends on the algorithm used to compute it. For example the image can be zero-padded (slides 45-48). Or the image's right side can be joined to its left, its top joined to its bottom and the convolution performed on the resulting torus (slides 51-59).

Three ways to compute a convolution

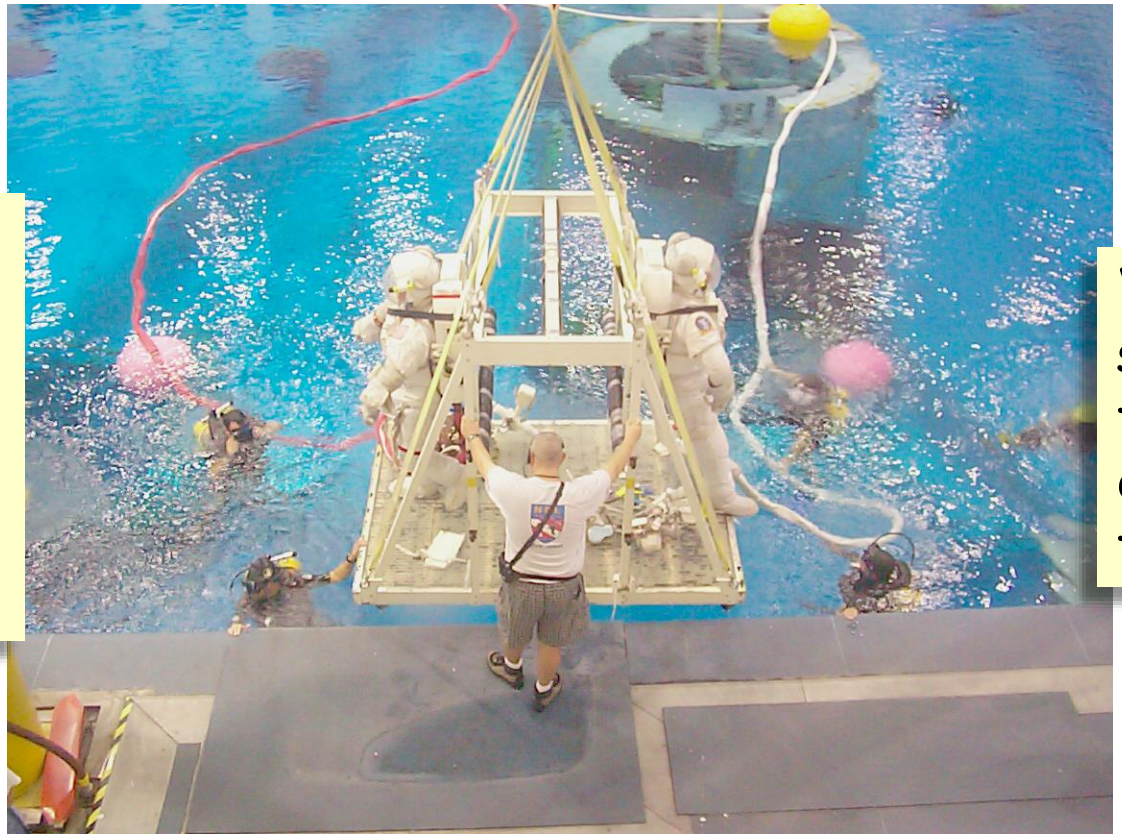
1. Moving window transform.
2. Shift multiply add.
3. Fourier transform.





Moving-Window Transformations

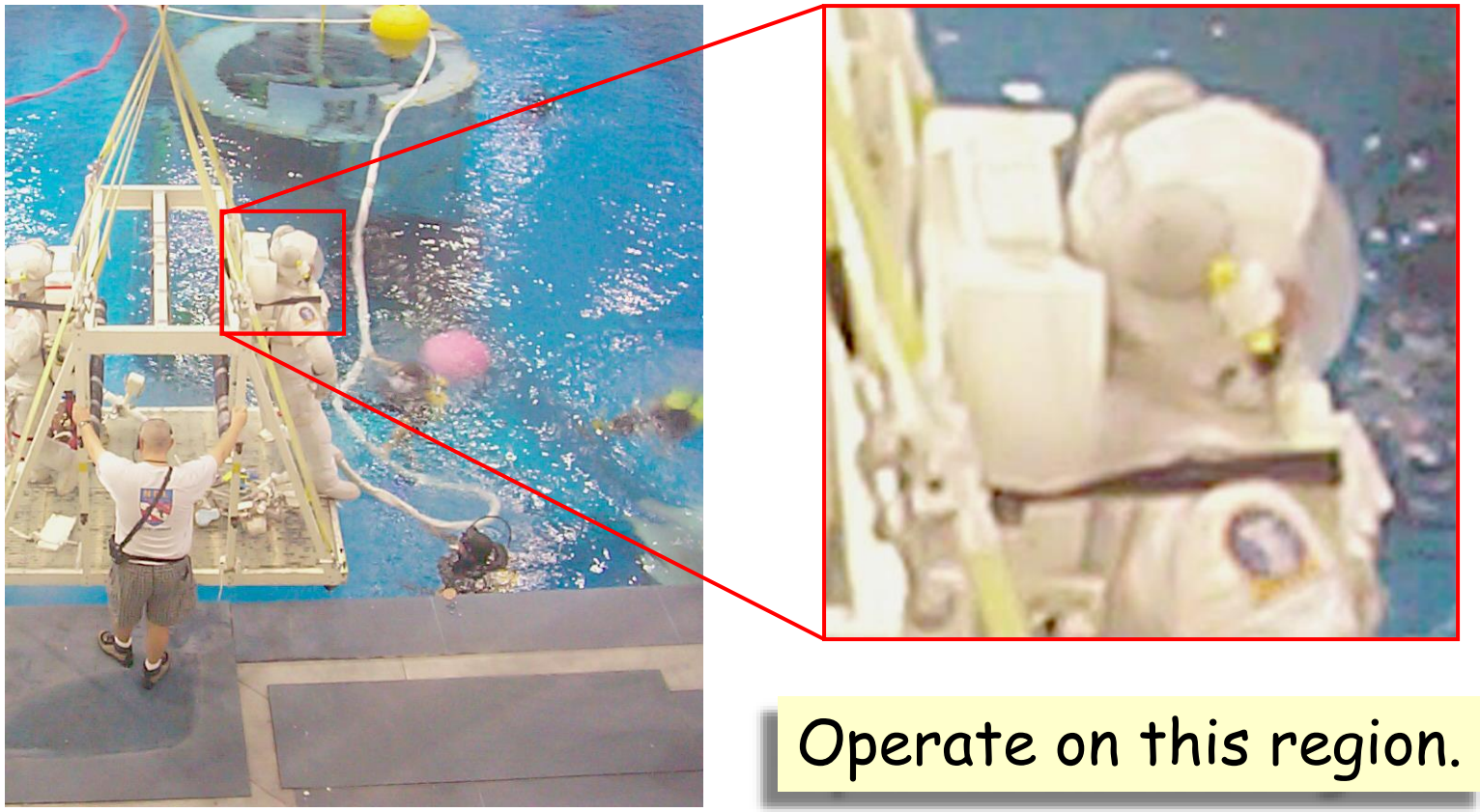
Neutral Buoyancy Facility at NASA Johnson Space Center



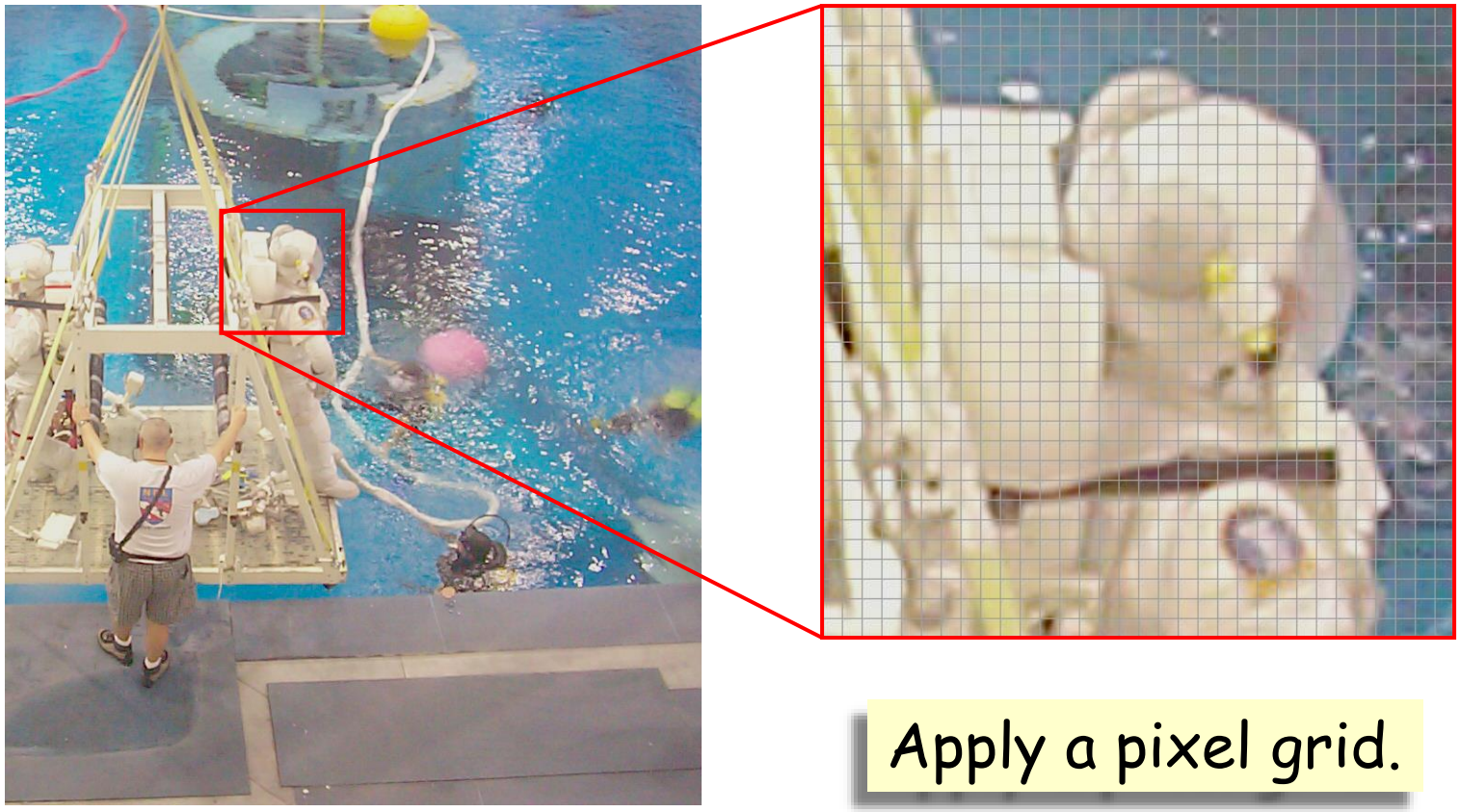
We'll take a section of this image to demonstrate the MWT.

photo: R.A.Peters II, 1999

Moving-Window Transformations



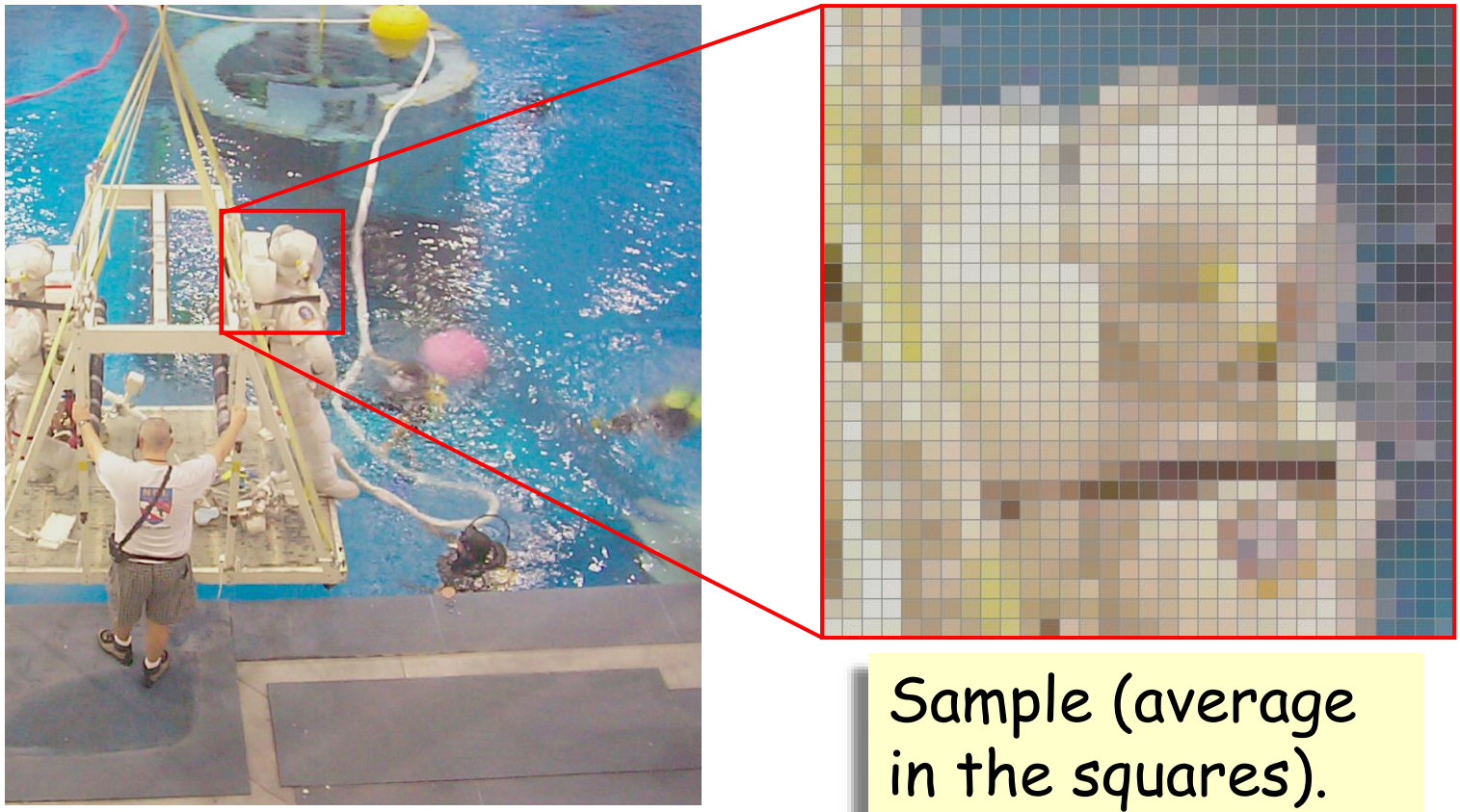
Moving-Window Transformations



Apply a pixel grid.

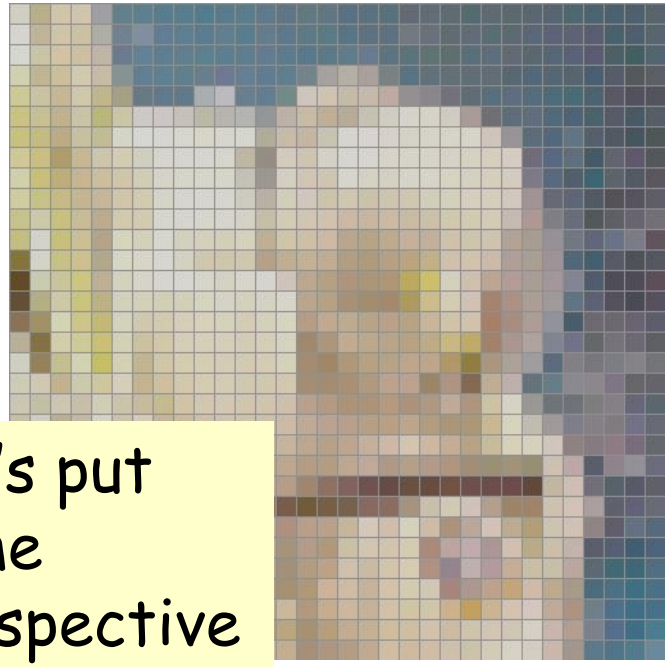
Pixelize the section to better see the effects.

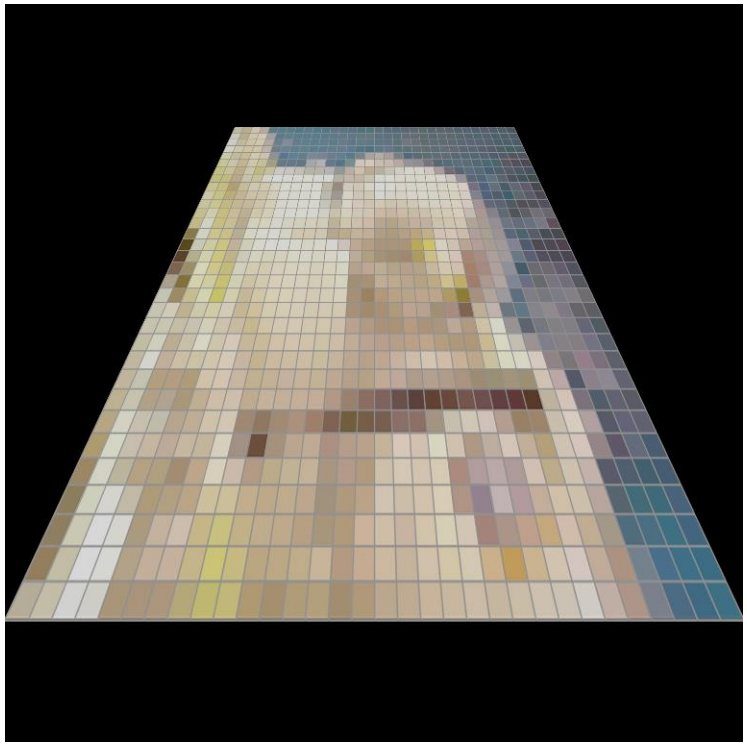
Moving-Window Transformations



Sample (average in the squares).

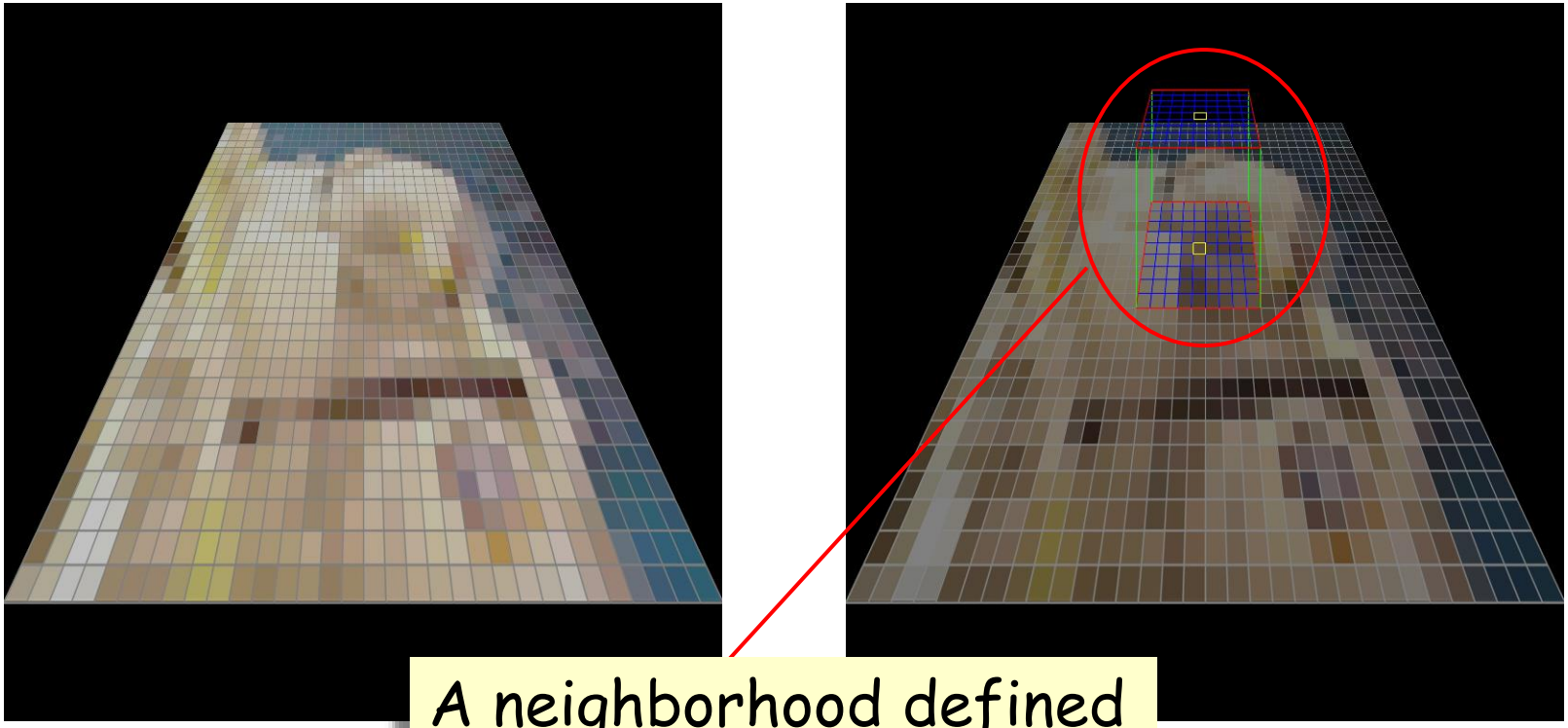
Moving-Window Transformations



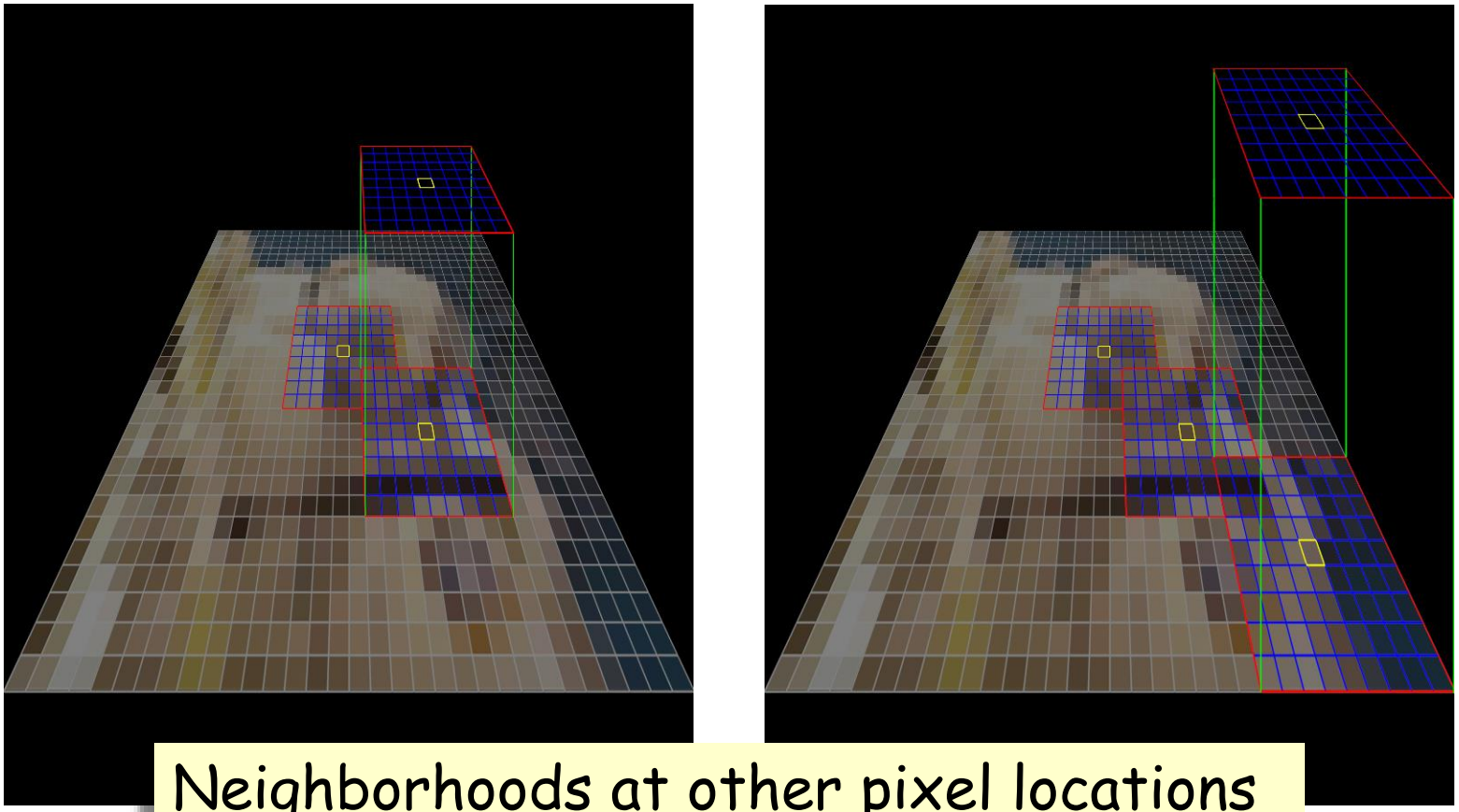


Let's put some perspective on this.

Moving-Window Transformations

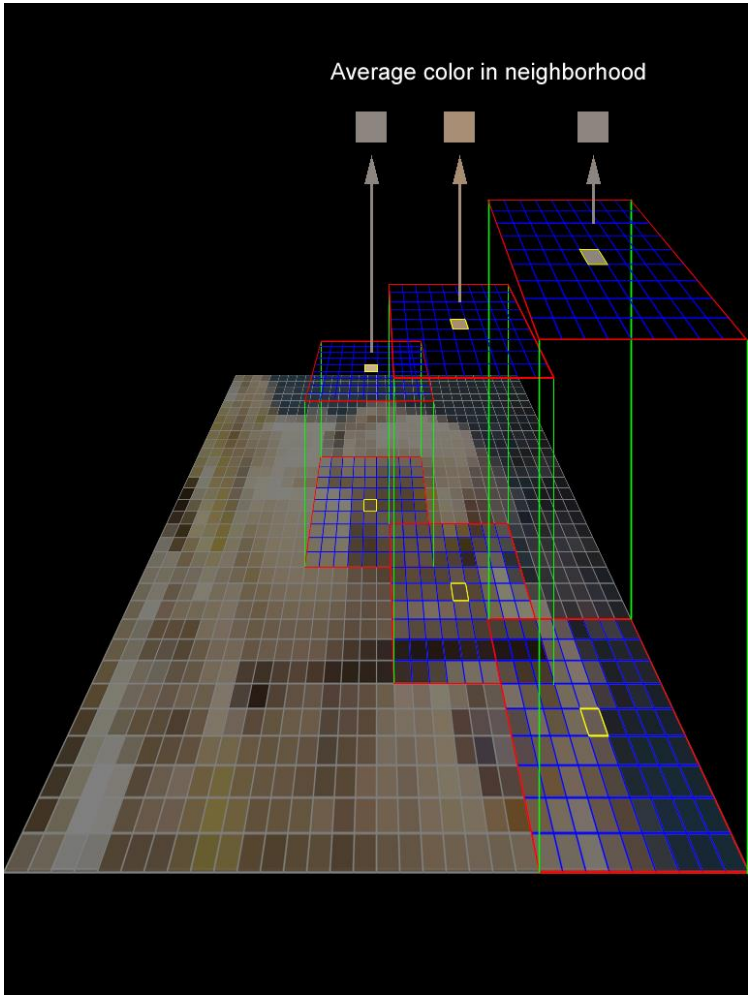


Moving-Window Transformations

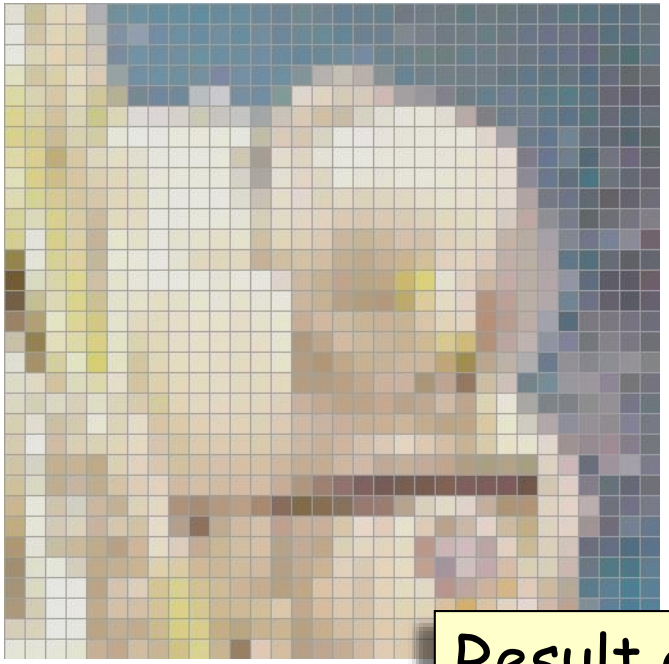


Linear Moving-Window Transformations (*i.e.* a Convolution)

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



Moving-Window Transformations



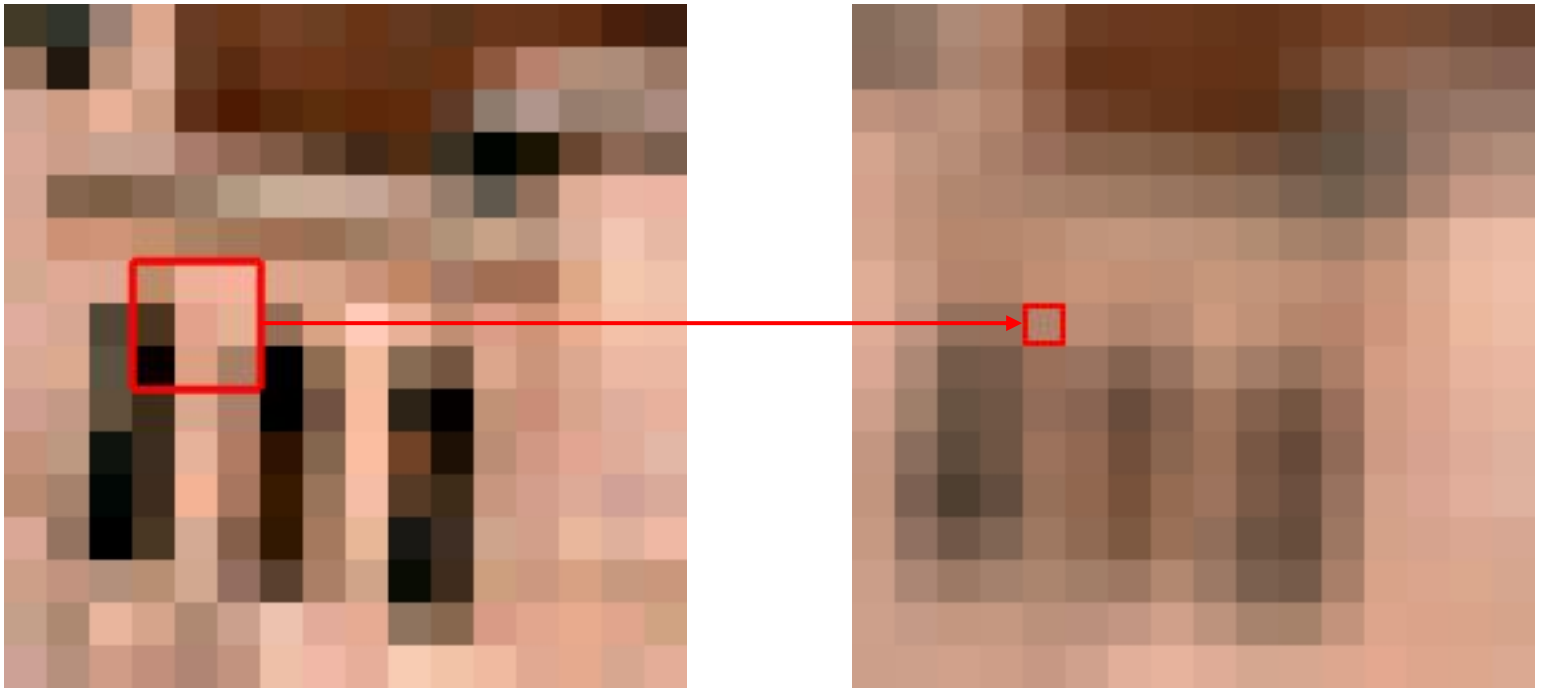
Result of a 9 x 9 uniform averaging

Moving Window Transform: Example

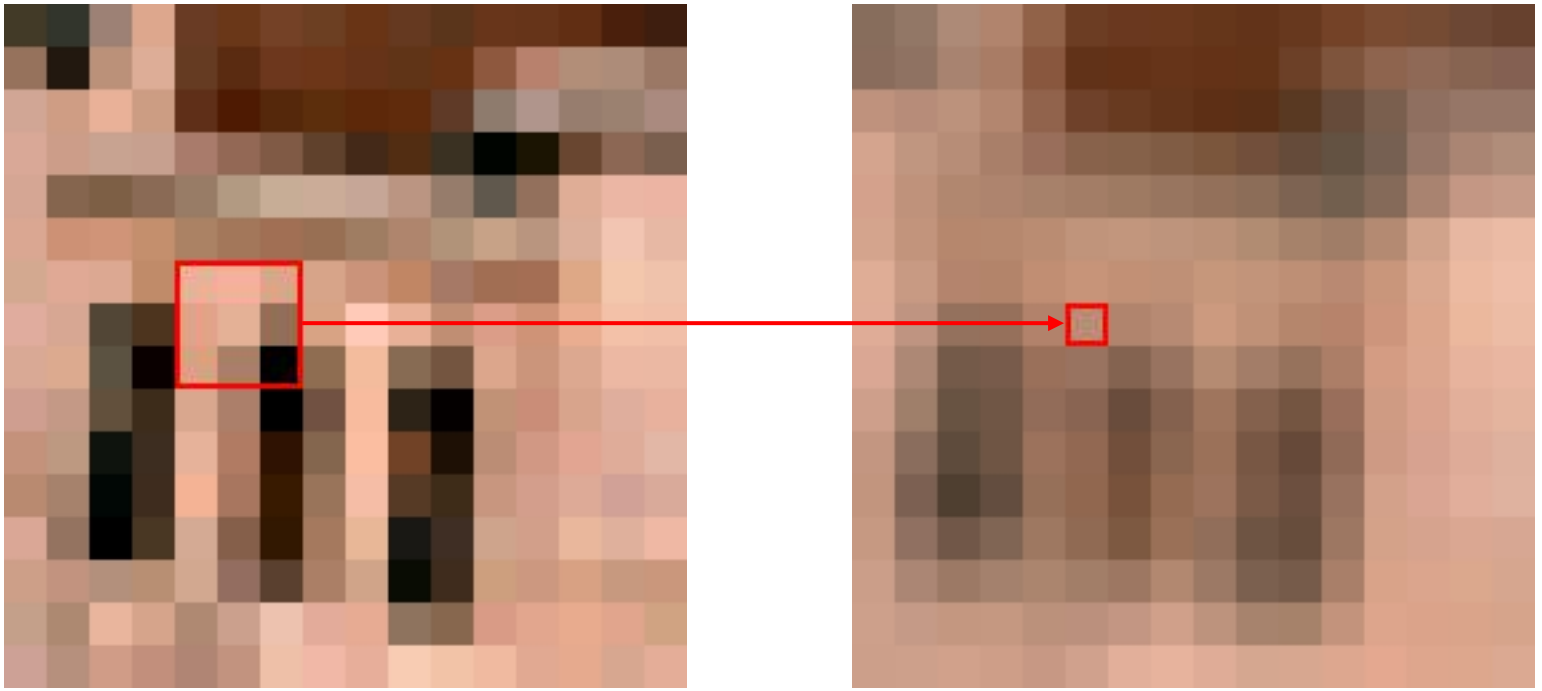
original

3x3 average

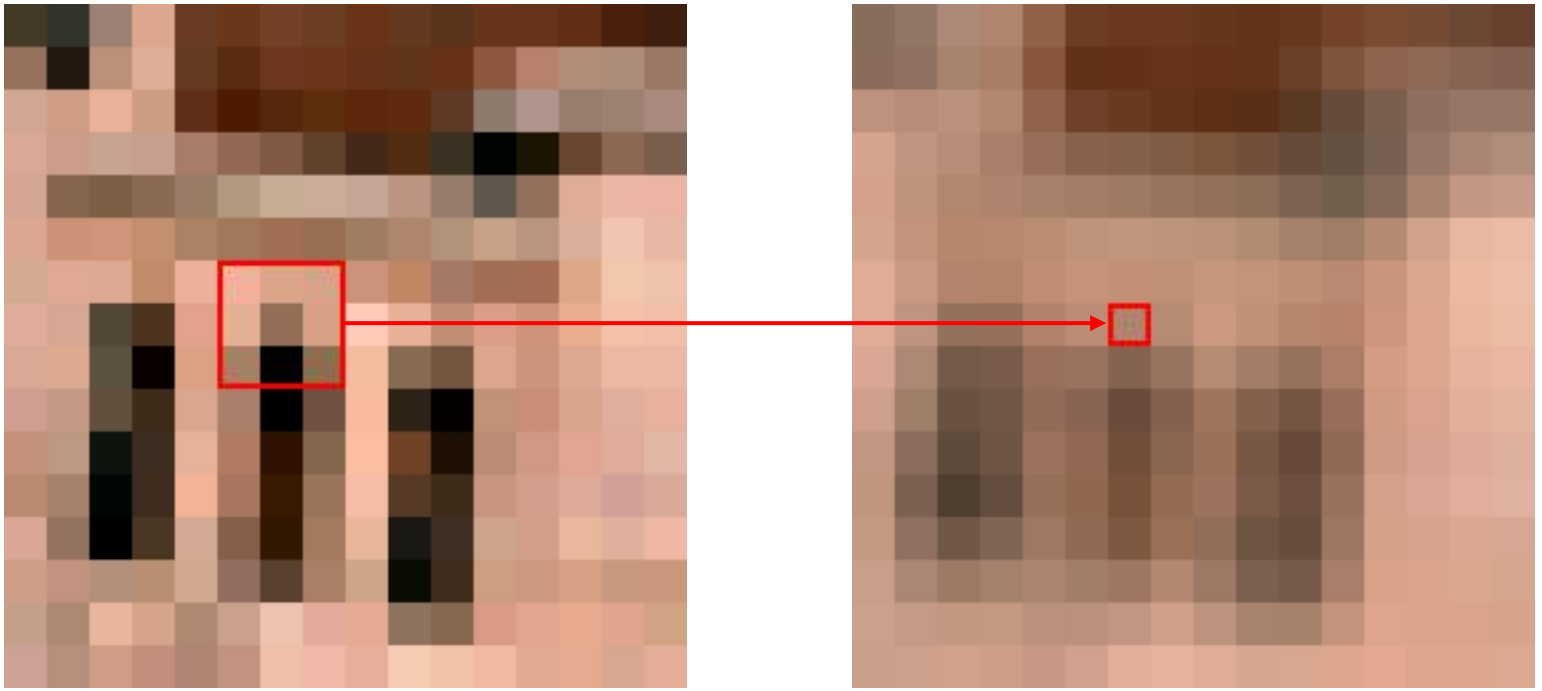
Moving Window Transform: Example



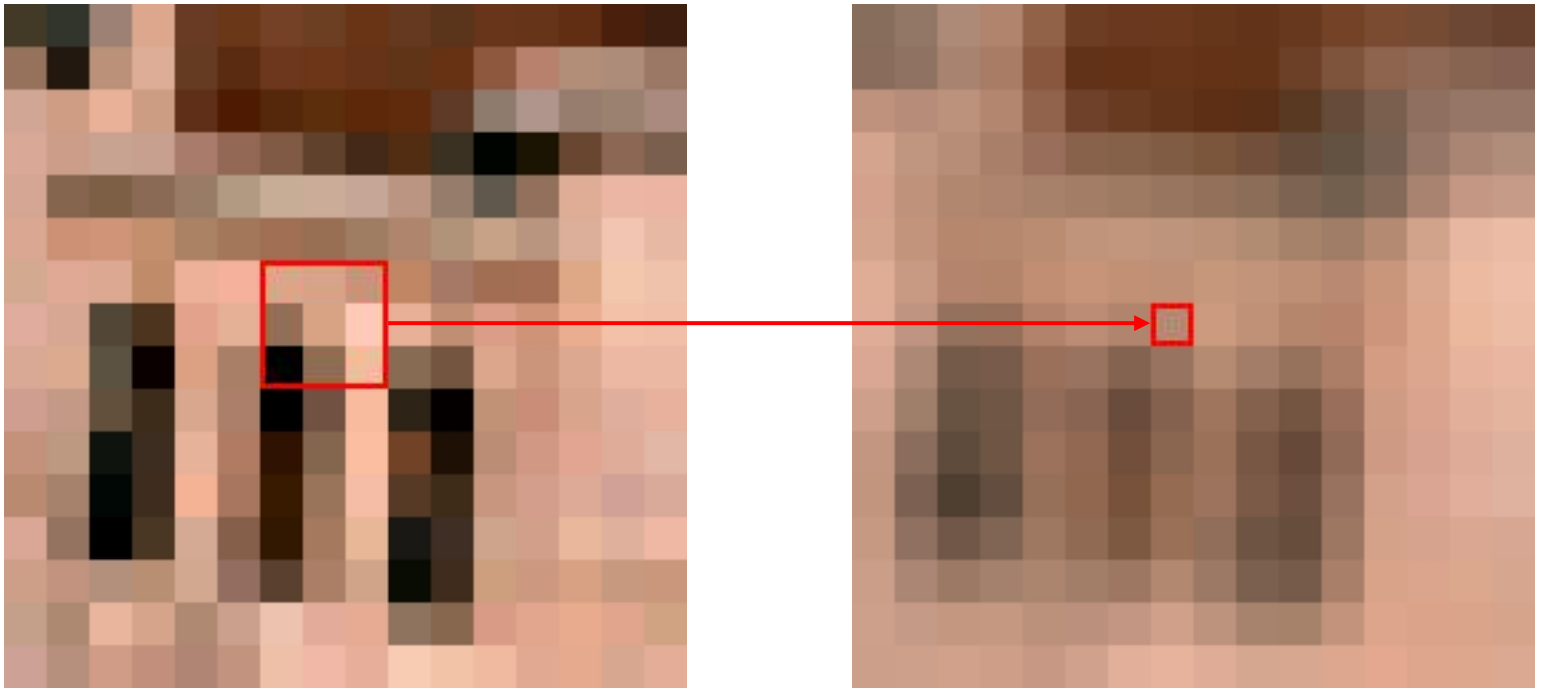
Moving Window Transform: Example



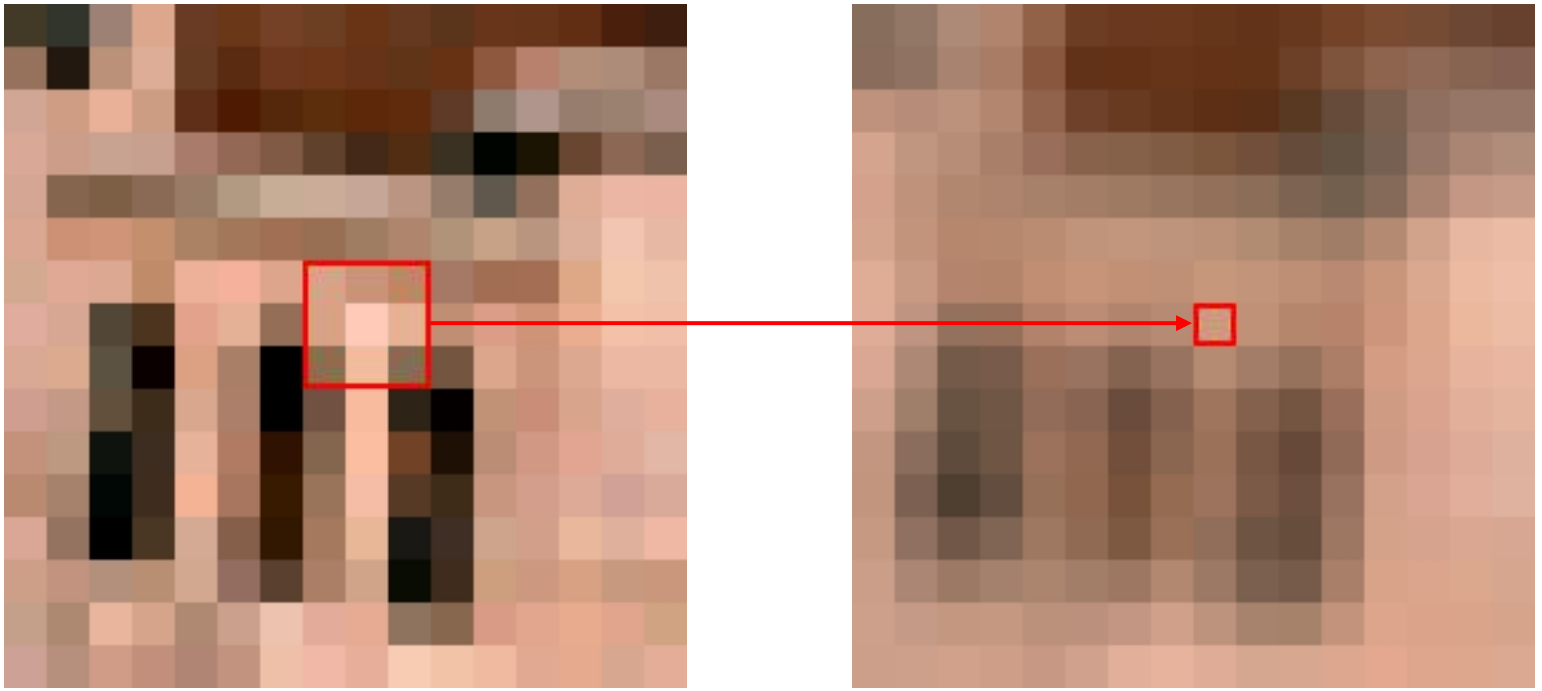
Moving Window Transform: Example



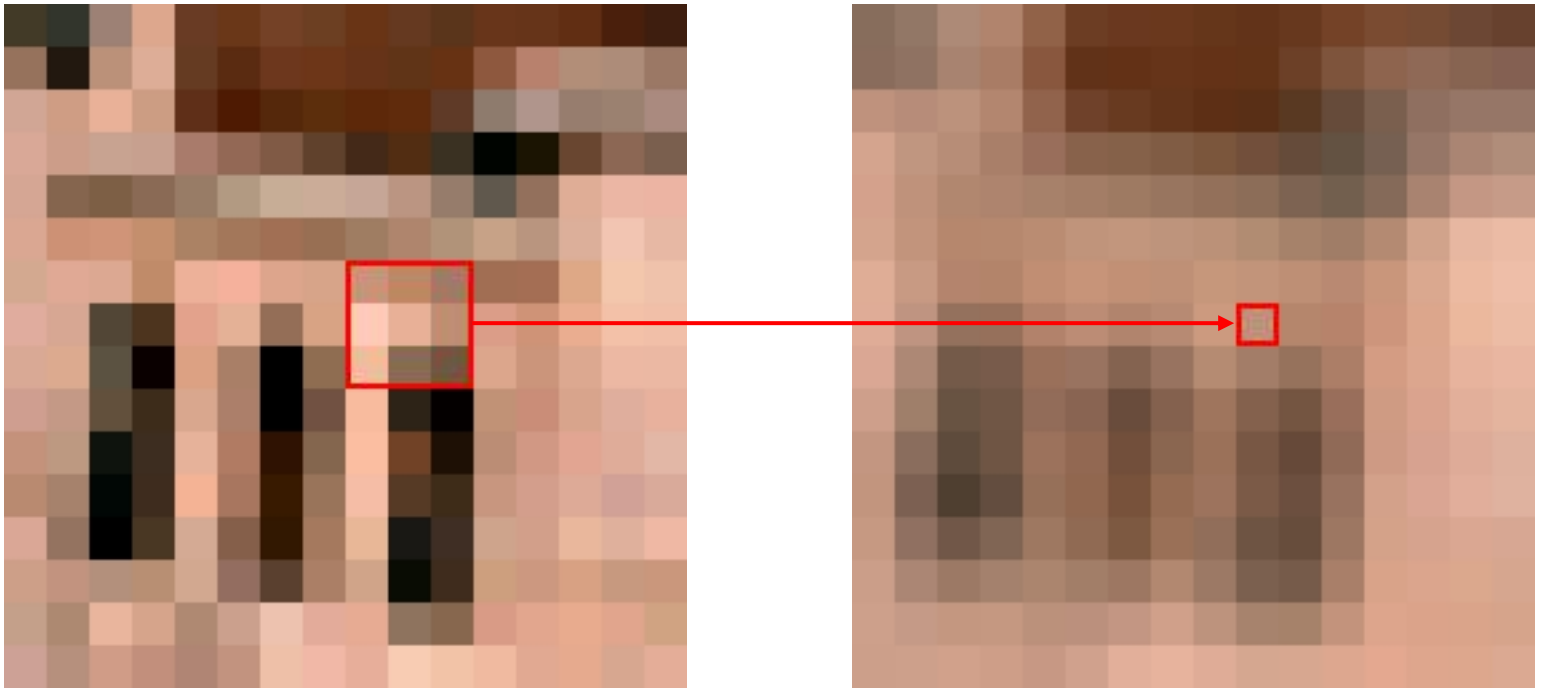
Moving Window Transform: Example



Moving Window Transform: Example



Moving Window Transform: Example

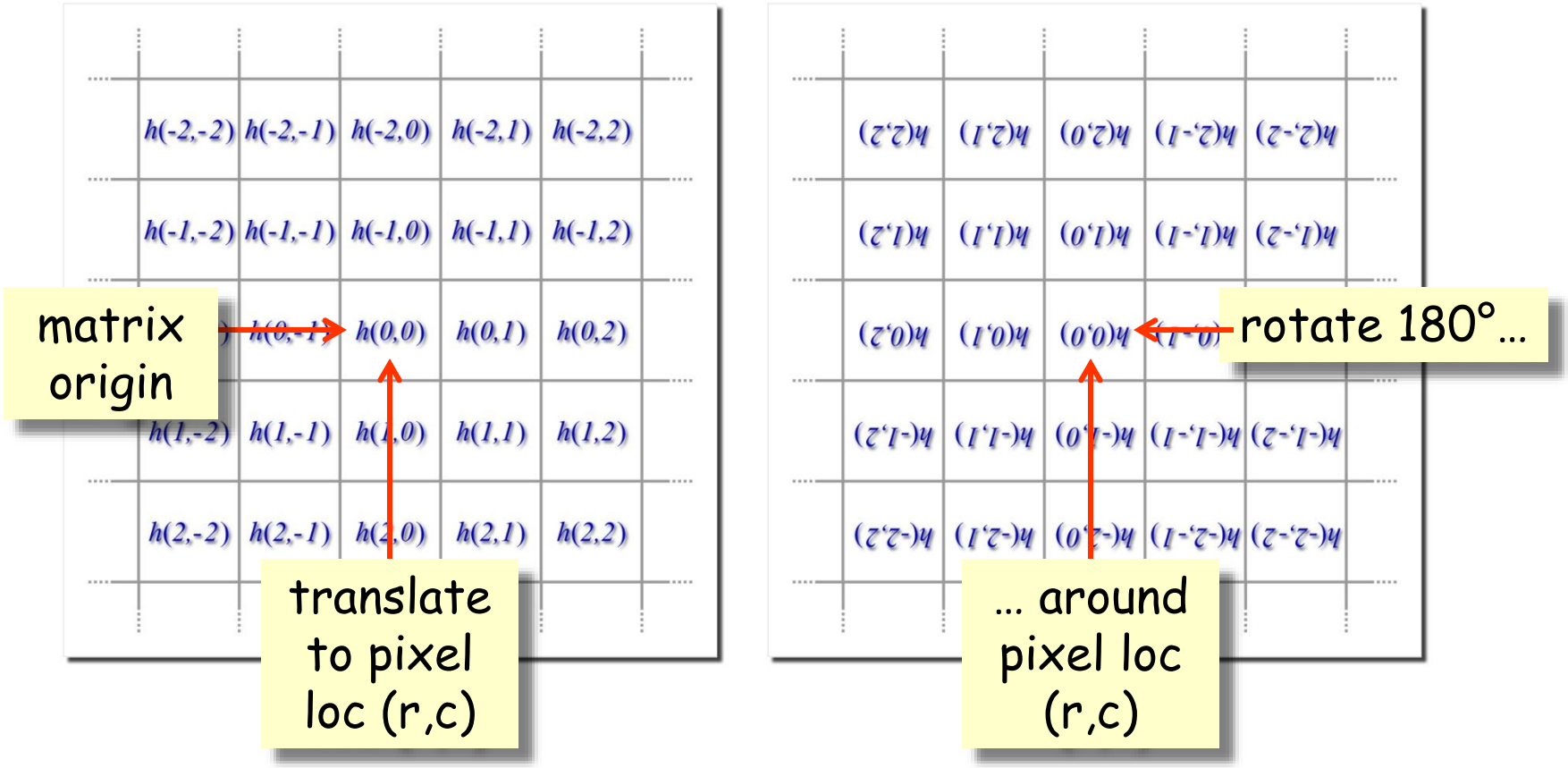


Convolution Matrix (Weight Matrix)

- 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. It is called variously the weight matrix; convolution matrix, kernel, mask, or neighborhood; or a filter, filter kernel, or filter matrix. All the names mean the same thing.
- 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 180^{o}. (See slide 30).
- The sum of those products is the value of pixel (r,c) in the output image.

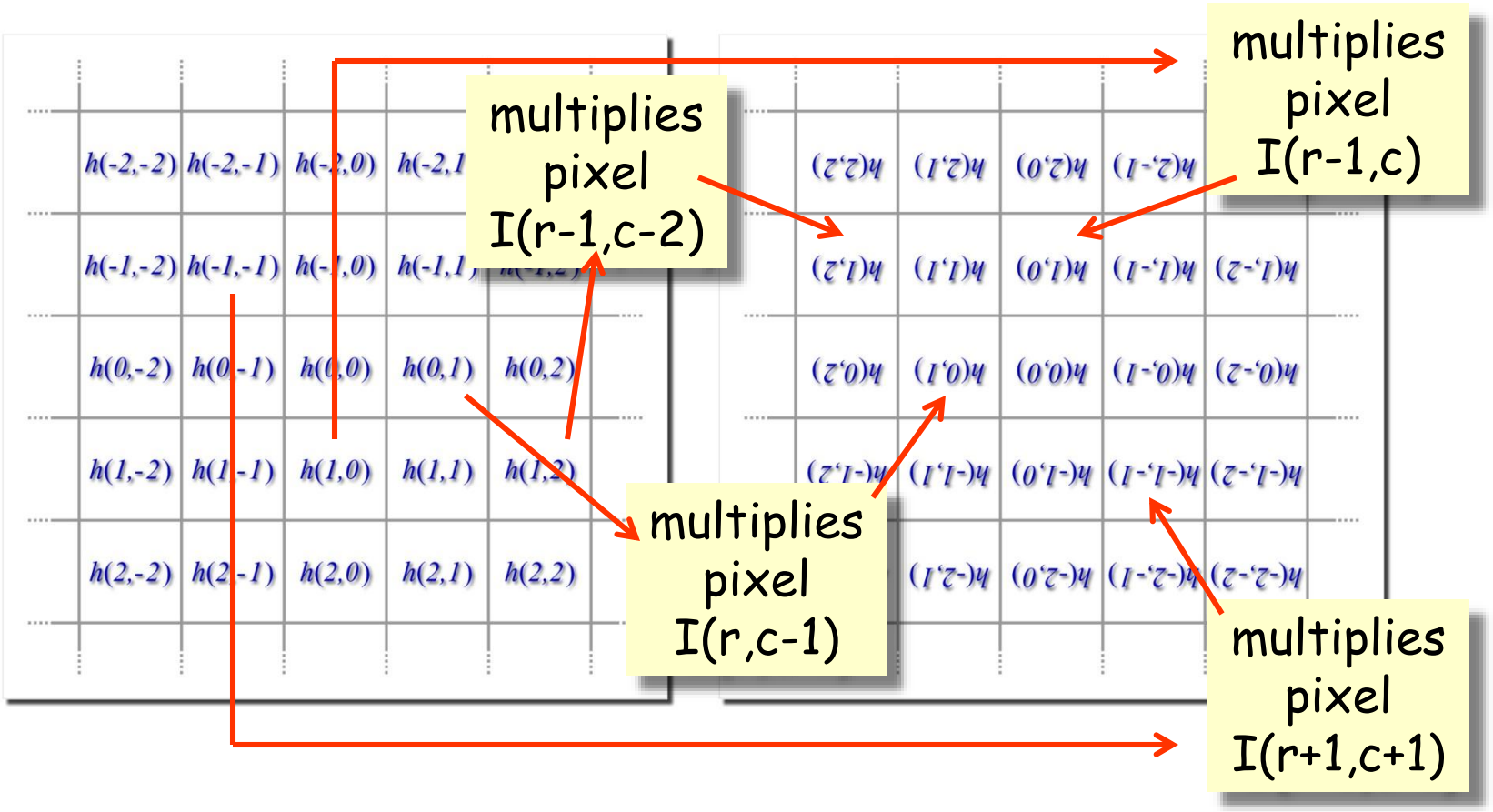
When performed per the definition on p. 5, the convolution's matrix must be rotated by 180° before the raster scan.

Convolution Matrices: Moving Window

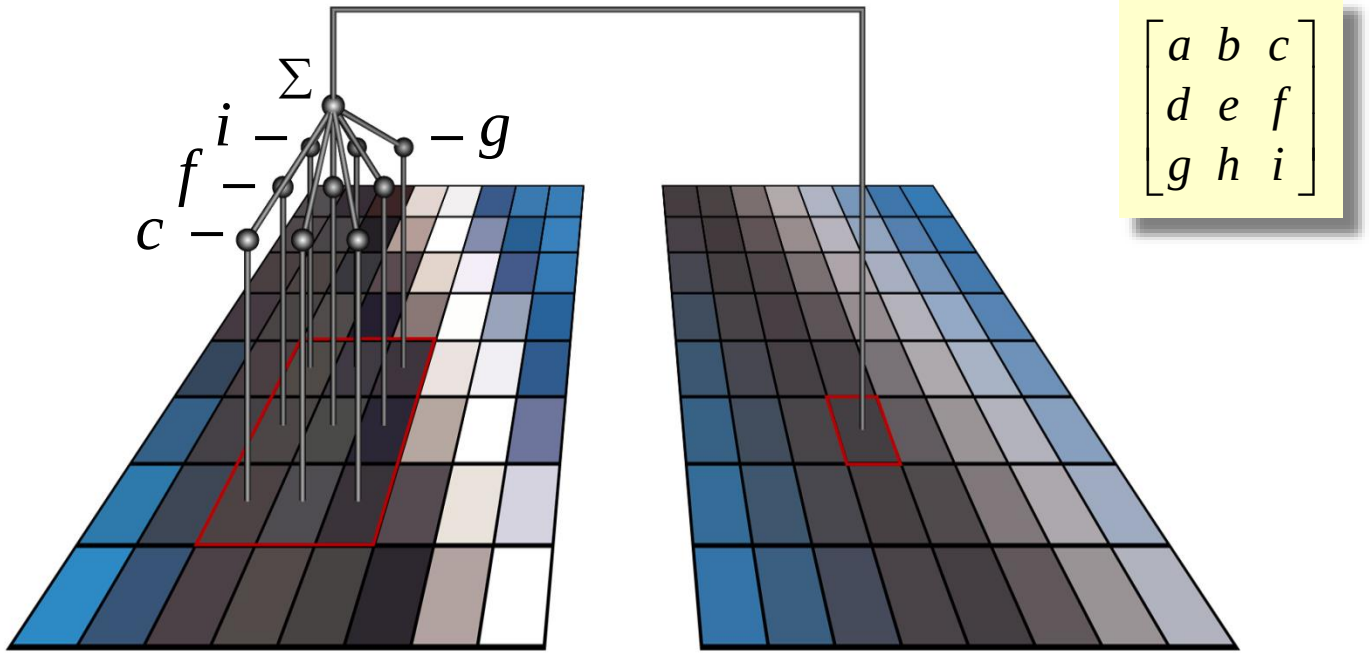


When performed per the definition on p. 16, the convolution's matrix must be rotated by 180° before the raster scan.

Convolution Matrices: Moving Window

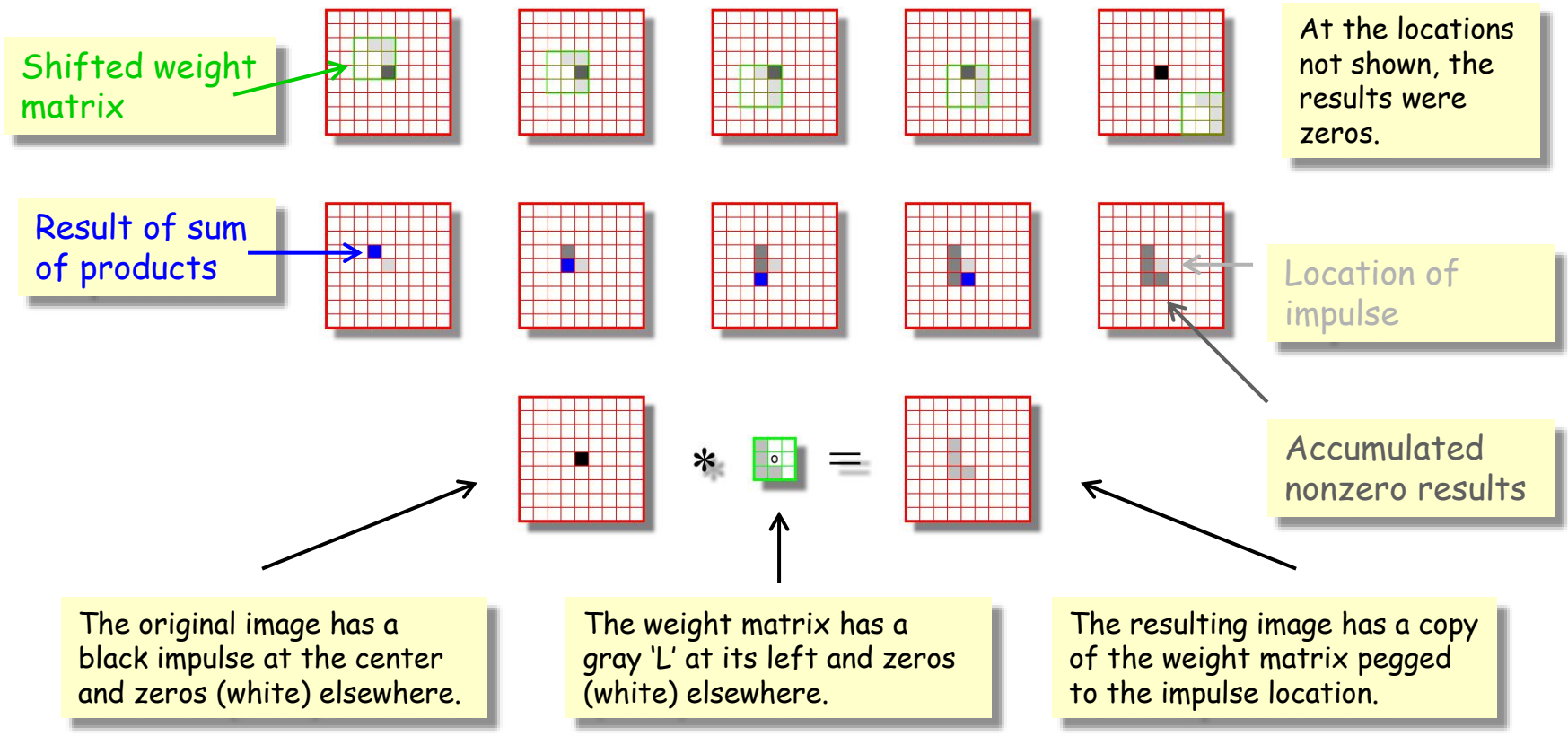


Convolution by Moving Window



The convolution of an impulse with a matrix places a copy of the matrix at the location of the impulse. That's why we rotate the matrix by 180°.

Convolution by Rotating and Shifting the Weight Matrix



Symmetric Weight Matrix

f	*e*	*d*	*e*	*f*
e	*c*	*b*	*c*	*e*
d	*b*	*a*	*b*	*d*
e	*c*	*b*	*c*	*e*
f	*e*	*d*	*e*	*f*

A symmetric weight matrix is unchanged by rotation through 180°.
Many commonly used filters have symmetric matrices. Many do not.

Convolution by an Impulse

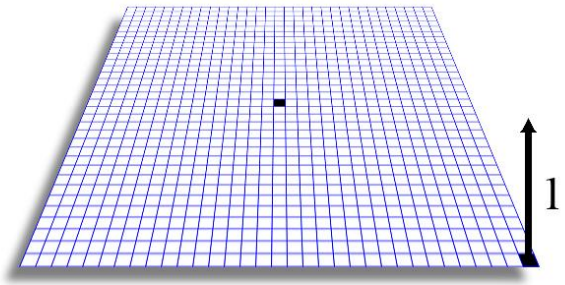
An *impulse* is a digital image, that has a single pixel with value 1; all others have value zero. An impulse at location (ρ, χ) is represented by:

$$\delta(r-\rho, c-\chi) = \begin{cases} 1, & \text{if } r=\rho \text{ and } c=\chi \\ 0, & \text{otherwise} \end{cases}$$

If an image is convolved with an impulse of weight w at location (ρ, χ), then the image is multiplied by w and shifted in location down by ρ pixels and to the right by χ pixels.

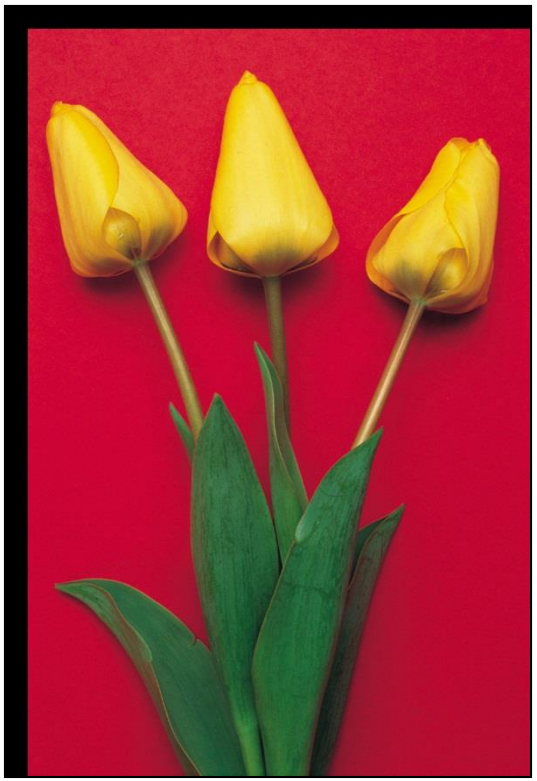
$$\left[\mathbf{I} * w\delta(r-\rho, c-\chi)\right](r,c) = w\mathbf{I}(r-\rho, c-\chi).$$

Convolution by an Impulse

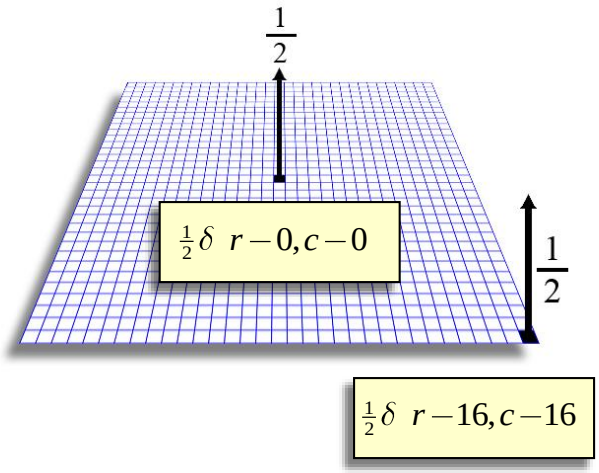


$\delta(r-16, c-16)$

Shifted down and to the right by 16 pixels.

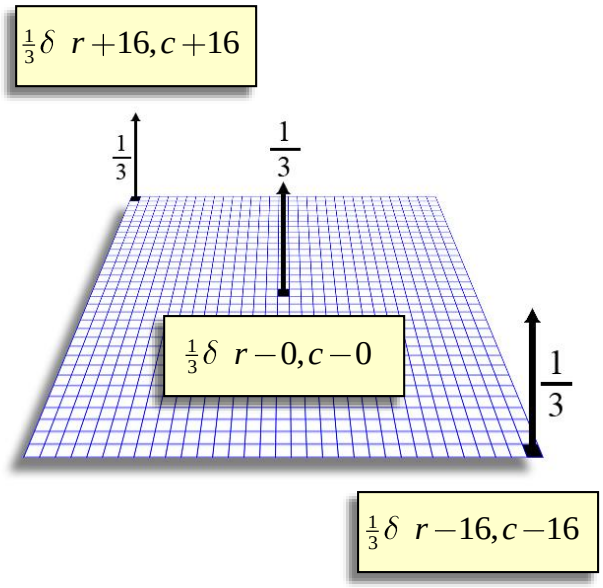


Convolution by Two Impulses



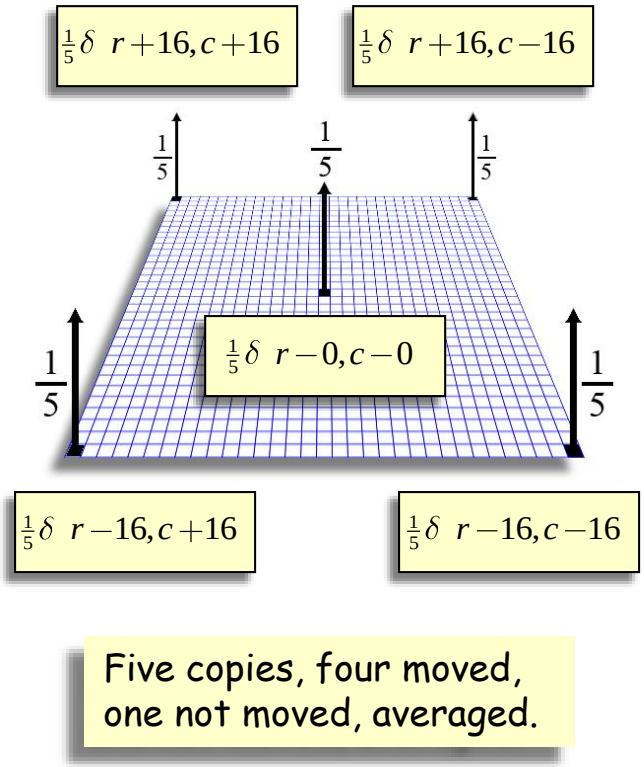
Two copies, one moved, one not moved, averaged.

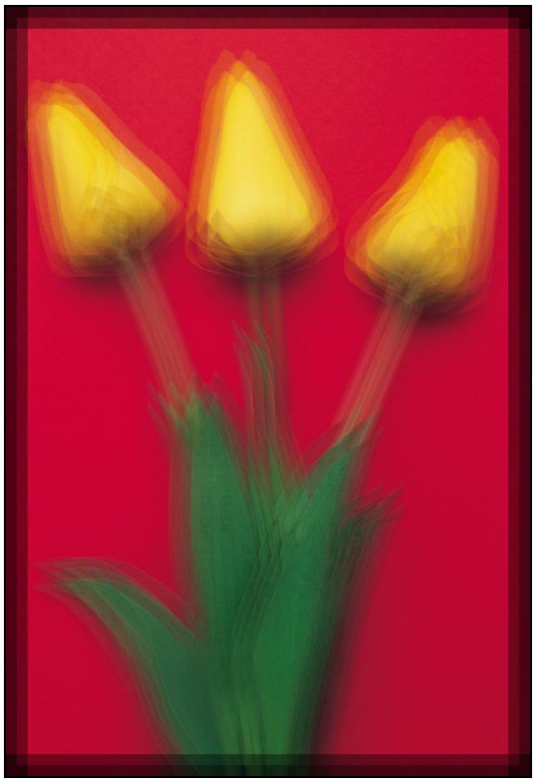
Convolution by Three Impulses



Three copies, two moved, one not moved, averaged.

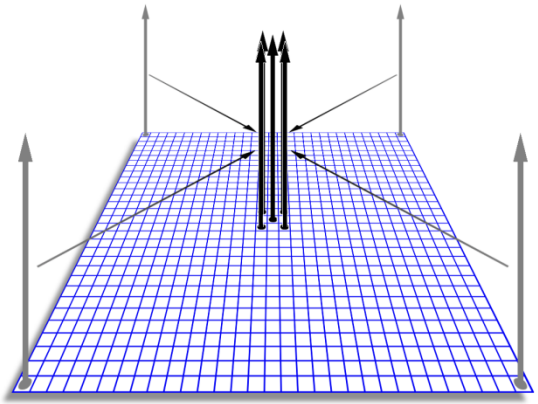
Convolution by Five Impulses





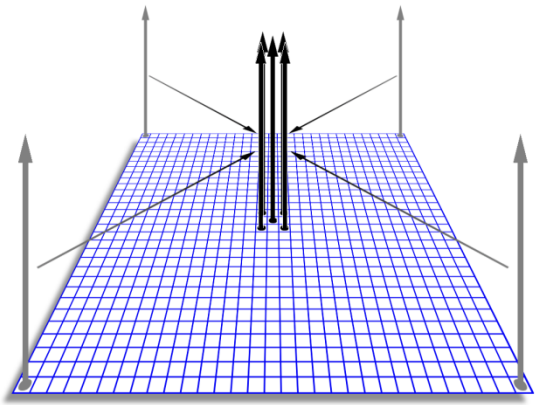
Five copies, four moved, one not moved, averaged.

Convolution by Five Impulses

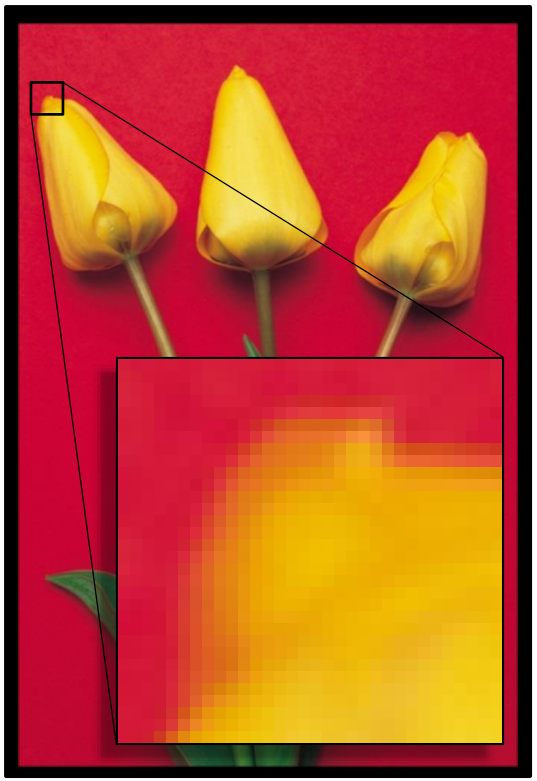


Moved adjacent to each other, the convolution becomes a blurring filter.

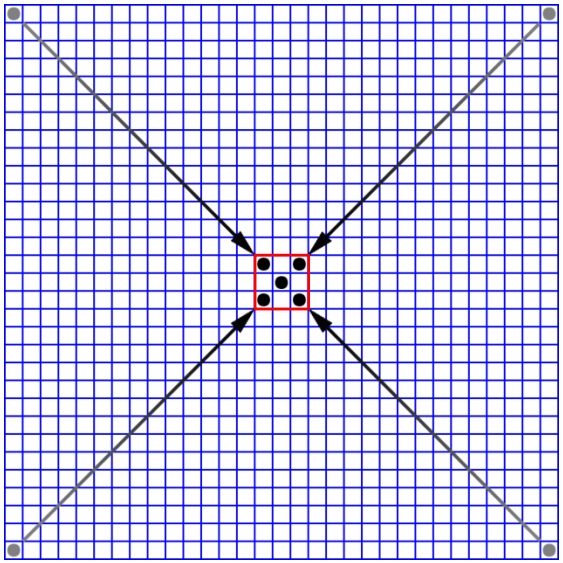
Convolution by Five Impulses



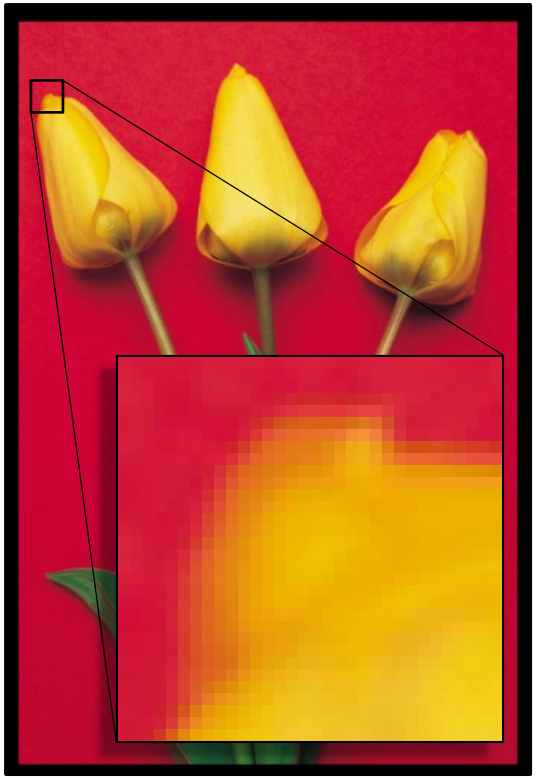
The impulses become values in a 3x3 neighborhood.



Convolution by Five Impulses



This convolution matrix has five elements at 1/5 and four at 0.



Shift-Multiply-Add Convolution

- The image is copied 1 time for each element in the convolution matrix.
- Each copy is shifted relative to the original by the displacement of its associated matrix element.
- Each copy is multiplied by the value of its associated matrix element.
- The set of shifted and multiplied images is summed pixel wise.

Shift-Multiply-Add Convolution

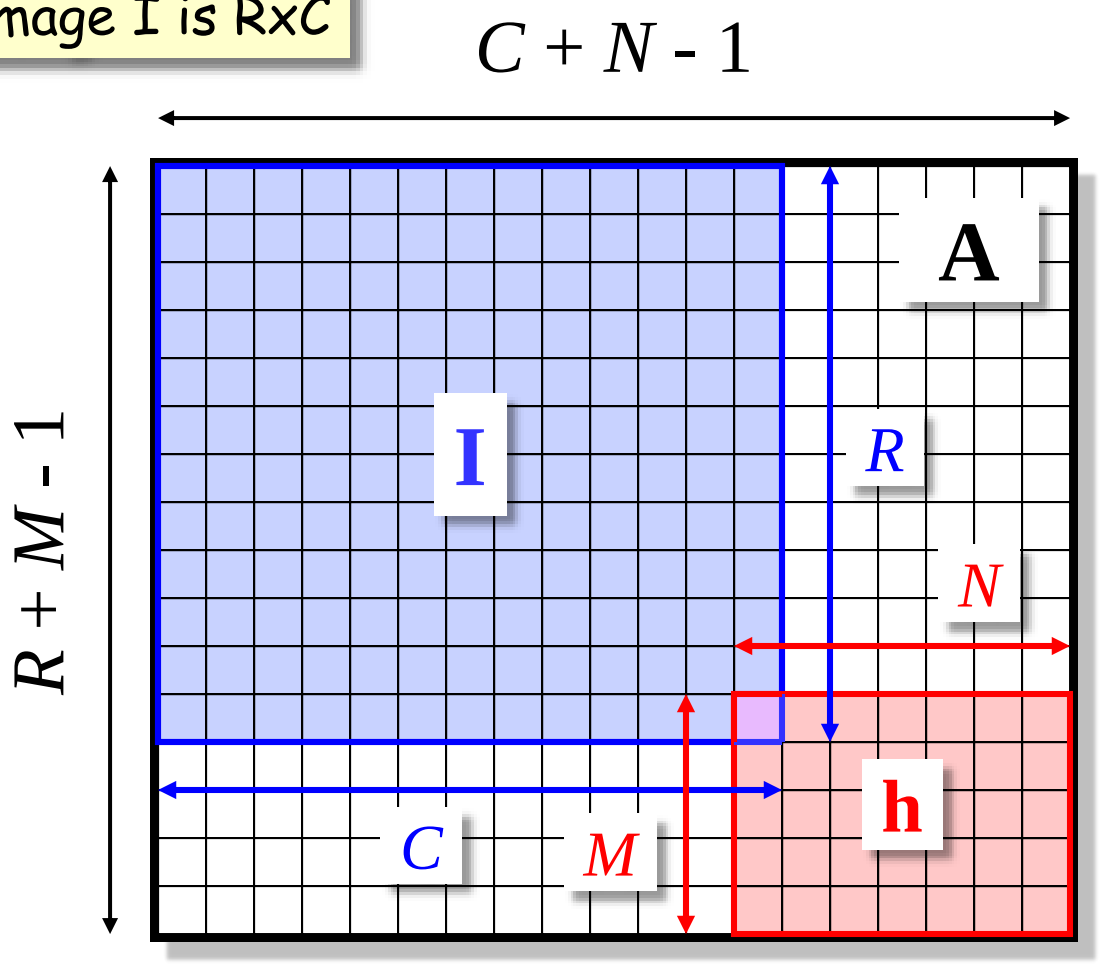
Assume images **I** and **J** are $R \times C \times B$ and that **h** is $M \times N \times 1$

1. Create a zero image, **A**, of size $R+M-1 \times C+N-1 \times B$
2. For $m \in \{1, \ldots, M\}$ and $n \in \{1, \ldots, N\}$ do:
3. $\mathbf{i} = [m \;\; m+1 \cdots m+R-1]$, $\mathbf{j} = [n \;\; n+1 \cdots n+C-1]$
4. $\mathbf{A}(\mathbf{i},\mathbf{j},:) = \mathbf{A}(\mathbf{i},\mathbf{j},:) + \mathbf{h}(m,n) \cdot \mathbf{I}$
5. End for
6. $\mathbf{r} = [\lfloor M/2 \rfloor+1 \;\; \lfloor M/2 \rfloor+2 \;\; \cdots \;\; \lfloor M/2 \rfloor+R]$
7. $\mathbf{c} = [\lfloor N/2 \rfloor+1 \;\; \lfloor N/2 \rfloor+2 \;\; \cdots \;\; \lfloor N/2 \rfloor+C]$
8. $\mathbf{J} = \mathbf{A}(\mathbf{r},\mathbf{c},:)$

Convolution by Copying and Shifting the Image

Image I is RxC

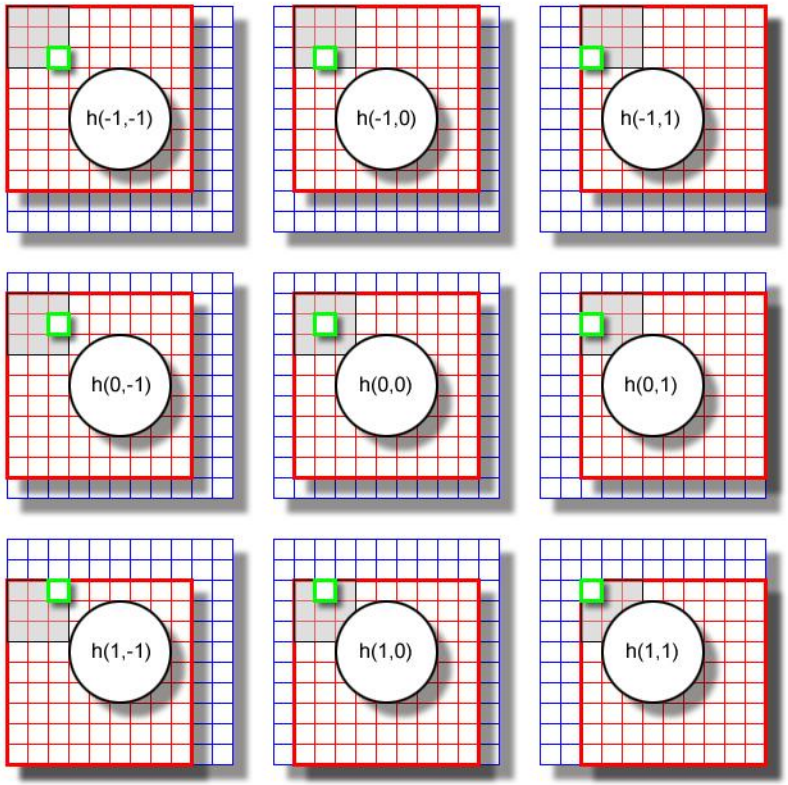
To use the image shift-multiply-accumulate algorithm, create an accumulator image, A, that is R+M-1 rows by C+N-1 columns



Convolution via Copy, Multiply, and Shift of the Image

For each element **h**(*m*,*n*) in an *M*×*N* weight matrix, **h,** *R*×*C* image **I** is multiplied by **h**(*m*,*n*) then added to a zero-padded accumulator image, **A**, in the *R*×*C* rectangle with upper right corner (*m*,*n*) and lower right corner (*m*+*R*-1,*n*+*C*-1).

After all *M*×*N* matrix elements have been looped over, the *R*×*C* result is cropped out of **A** starting at the origin of **h**, typically (⌊*M*/2⌋+1, ⌊N/2⌋+1)



original image, **I**

accumulator image, **A**

effective neighborhood

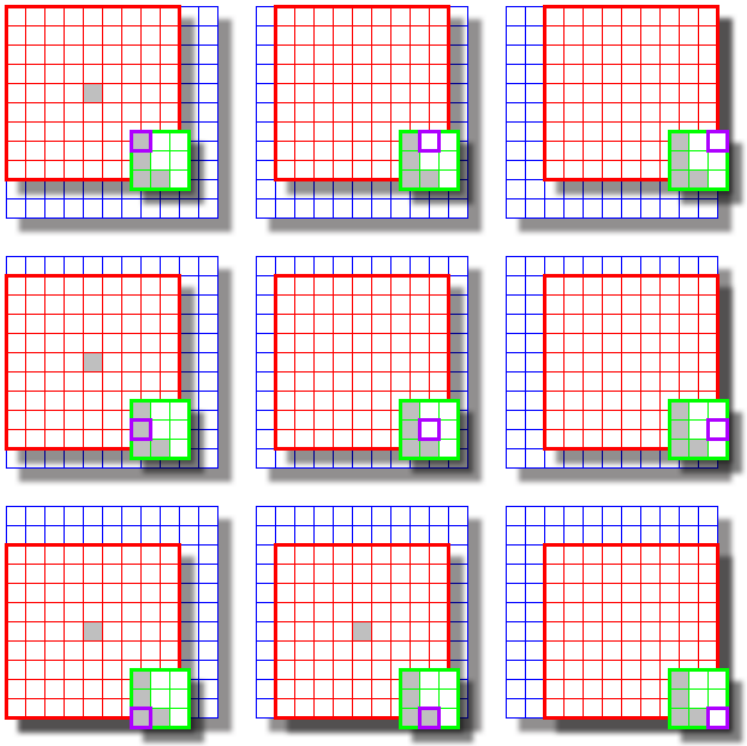
h(-1,-1)	h(-1,0)	h(-1,1)
h(0,-1)	h(0,0)	h(0,1)
h(1,-1)	h(1,0)	h(1,1)

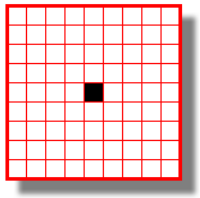
weight matrix

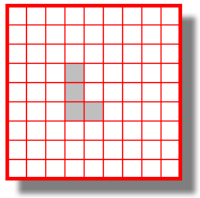
aligned pixels (*r*,*c*) are summed

weight for image ○

Convolution via Copy, Multiply, and Shift of the Image







The original image has a black impulse at the center and zeros (white) elsewhere.

The weight matrix has a gray 'L' at its left and zeros (white) elsewhere.

The resulting image has a copy of the weight matrix pegged to the impulse location.

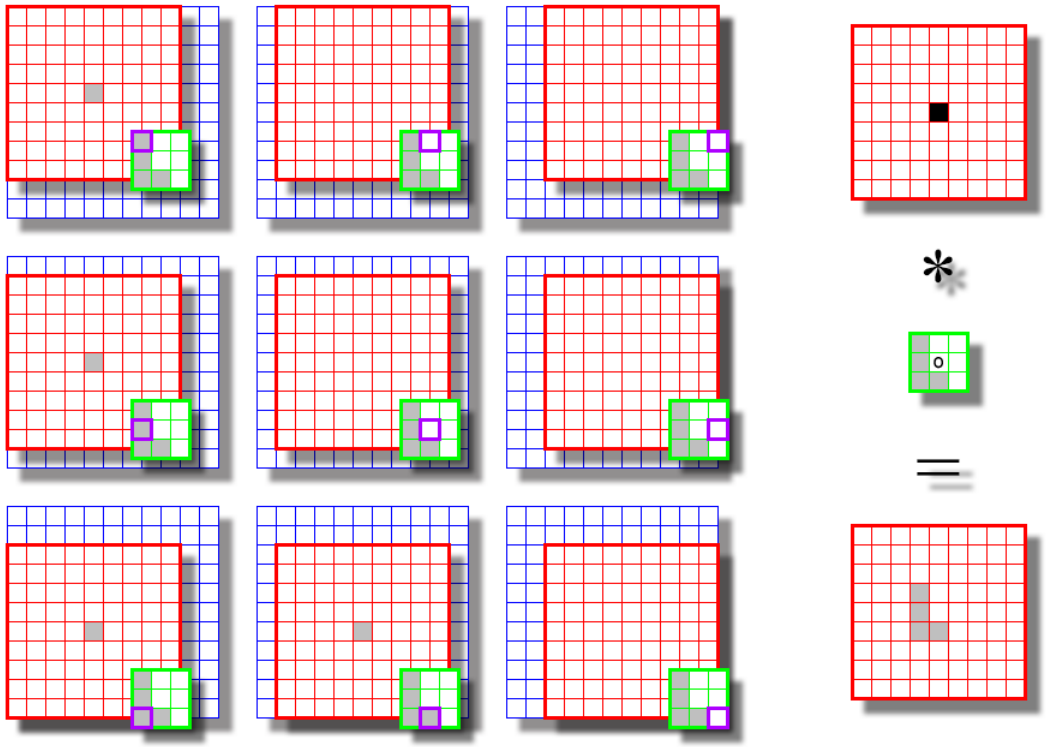
original image, **I**
effective neighborhood
accumulator image, **A**

In the result, the origin of the weight matrix coincides with the original location of the impulse.

Convolution by Copying, Multiplying, and Shifting the Image

Each copy of the (entire) image is multiplied by the value of the weight matrix in purple square (here, white = 0) before being accumulated (pixelwise) in the padded image

The position of the purple square relative the center of the weight matrix indicates the shift of the original image relative to the middle of the padded image.

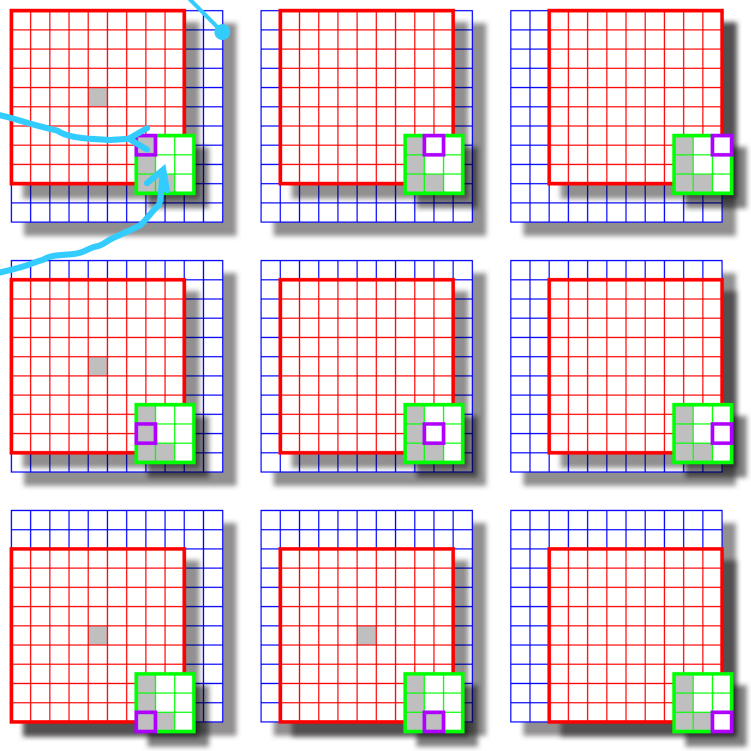


In this image, only the pixel in the center is nonzero so only it shows a result when the image is multiplied by a nonzero value

Convolution by Copying, Multiplying, and Shifting the Image

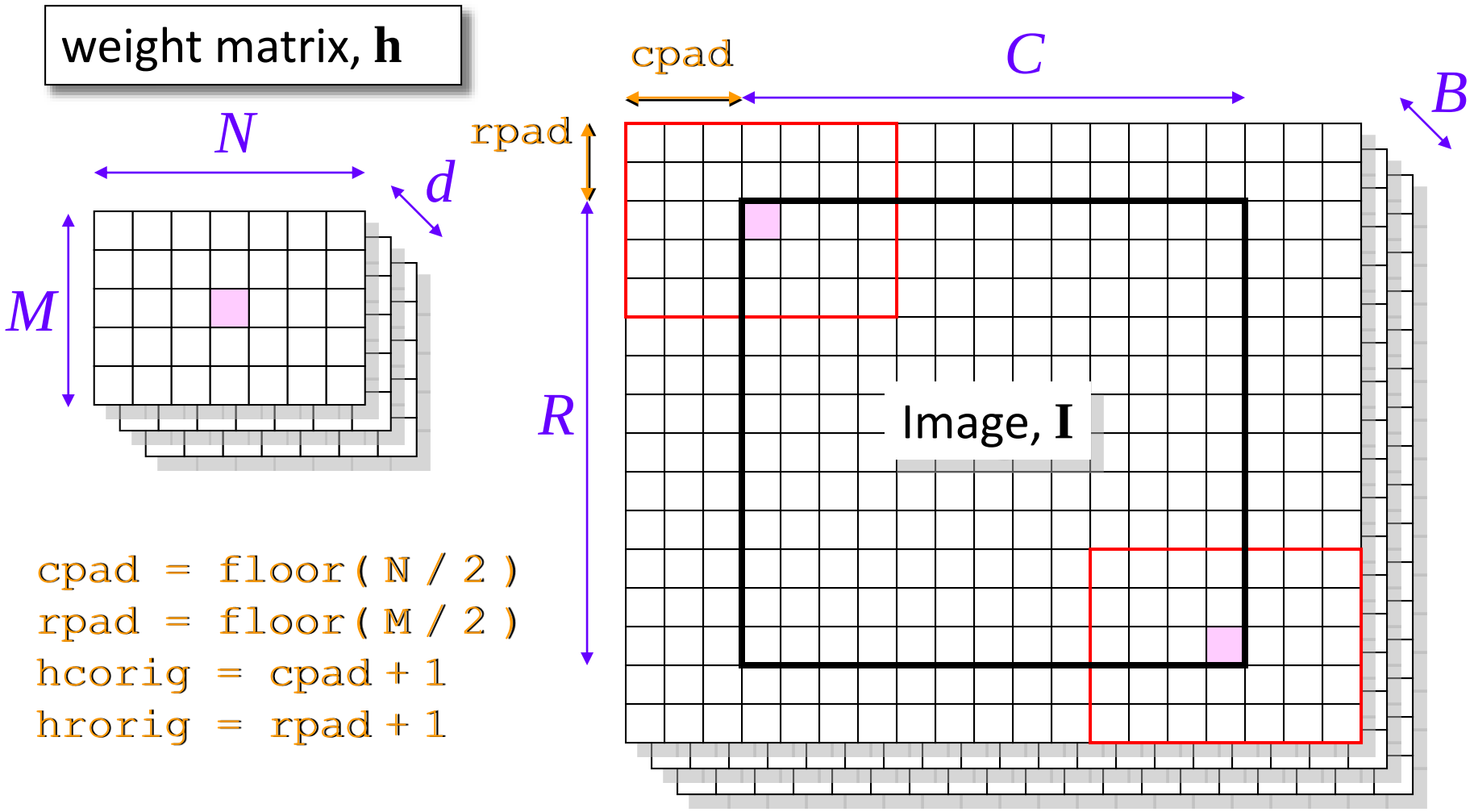
Each copy of the (entire) image is multiplied by the value of the weight matrix in purple square (here, white = 0) before being accumulated (pixelwise) in the padded image

The position of the purple square relative the center of the weight matrix indicates the shift of the original image relative to the middle of the padded image.



In this image, only the pixel in the center is nonzero so only it shows a result when the image is multiplied by a nonzero value

Zero Padding an Image for Convolution: Variable names.

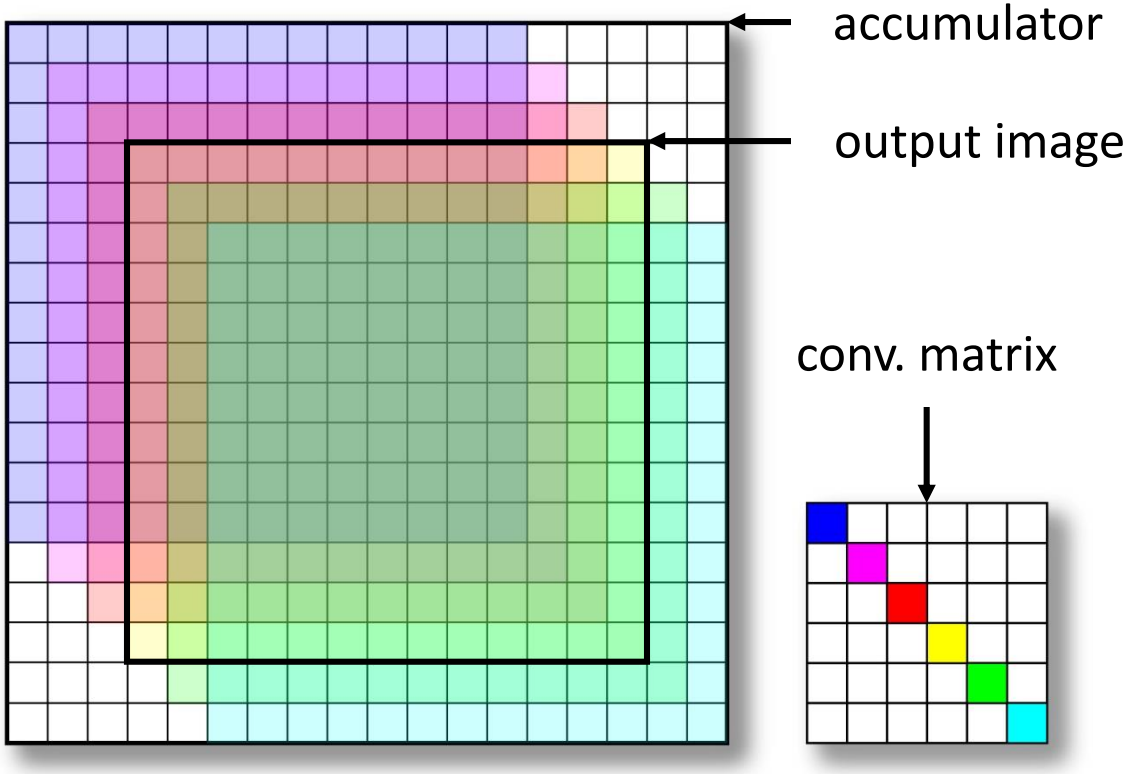


Convolution by Copying, Multiplying, and Shifting the Image

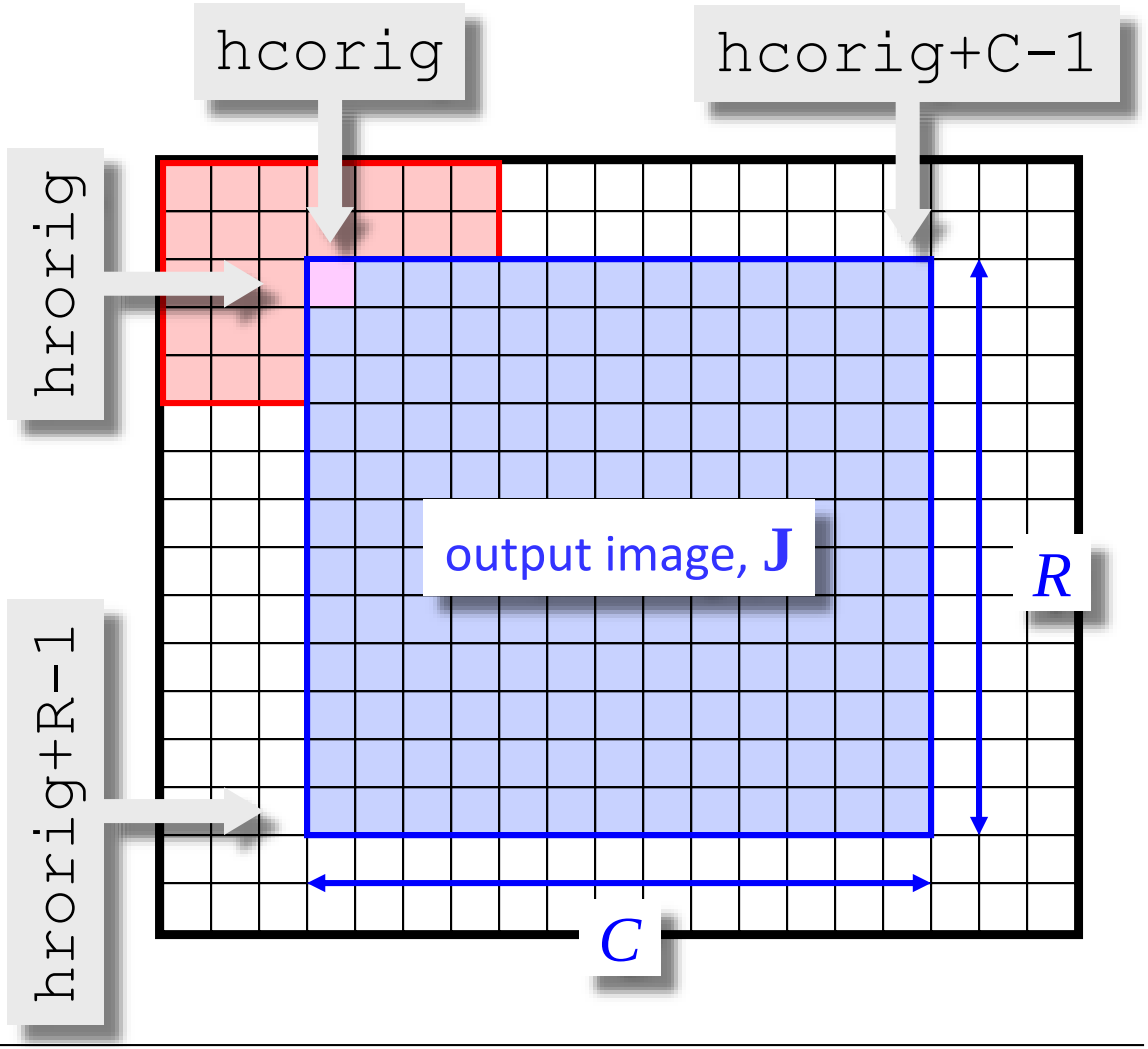
13x13 image convolved by 6x6 matrix.

Image is constant; matrix has only 6 nonzero values all on the diagonal.

Image is shifted to matrix location, multiplied by value, and accumulated.



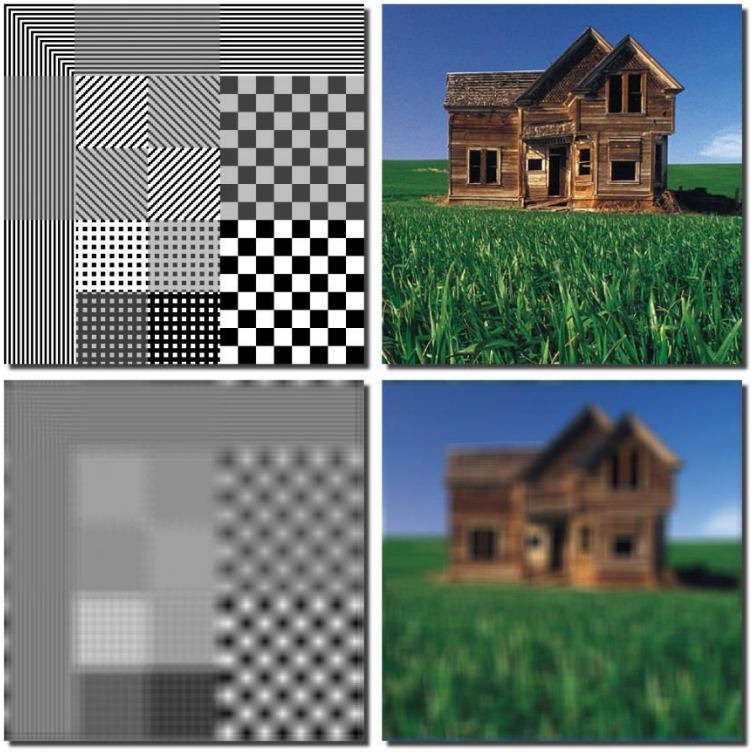
Convolution by Copying and Shifting the Image



When done, copy the output image from the accumulator starting at (hrorig, hcorig) and ending at (hrorig+R-1, hcorig+C-1)

The Box Filter

A square convolution matrix with uniform weights that sum to 1 is called a box filter. Convolution matrices do not have to be uniformly weighted. Nevertheless, if the weights are all positive and sum to 1, the result of convolution is blurring. If the weights are both positive and negative and sum to 0, then the filter is an “edge detector”.

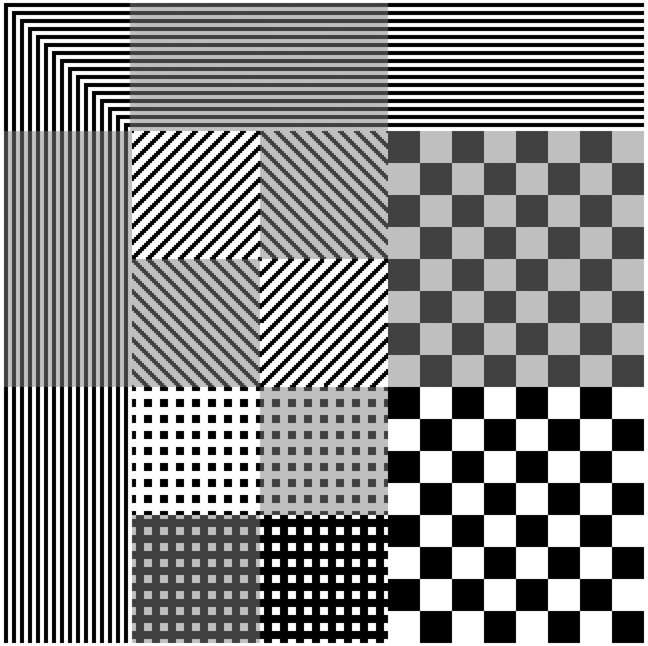


Convolution on a Torus

The 2D Discrete Fourier transform is defined on a torus. And it can be used to perform a convolution.

$$\mathbf{h} = \frac{1}{81}\begin{bmatrix} 1&1&1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1&1&1 \end{bmatrix}$$

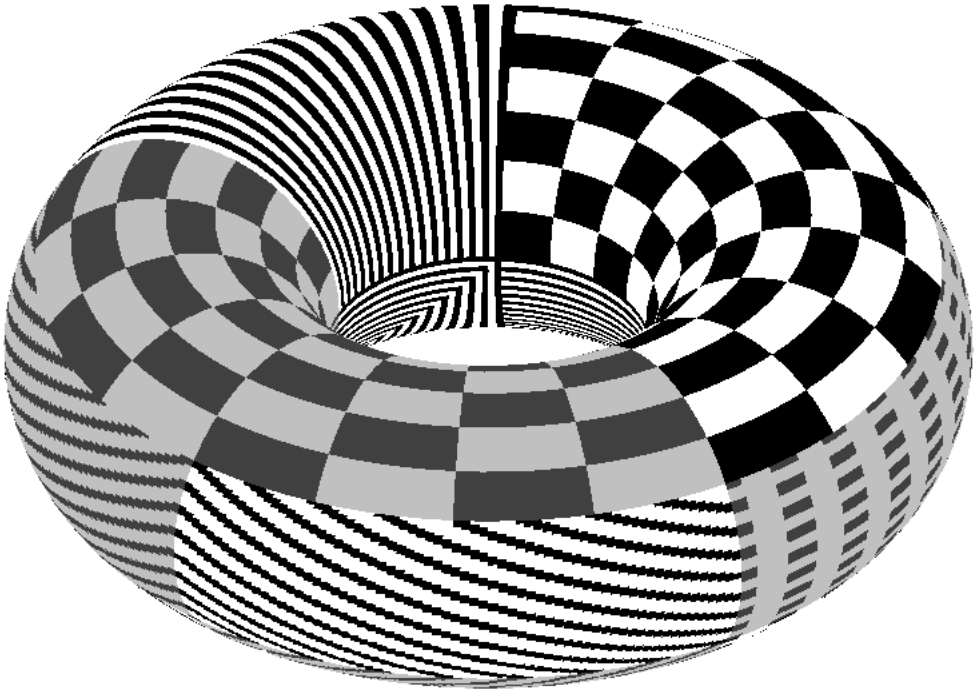
9x9 box filter



test pattern image

Convolution on a Torus

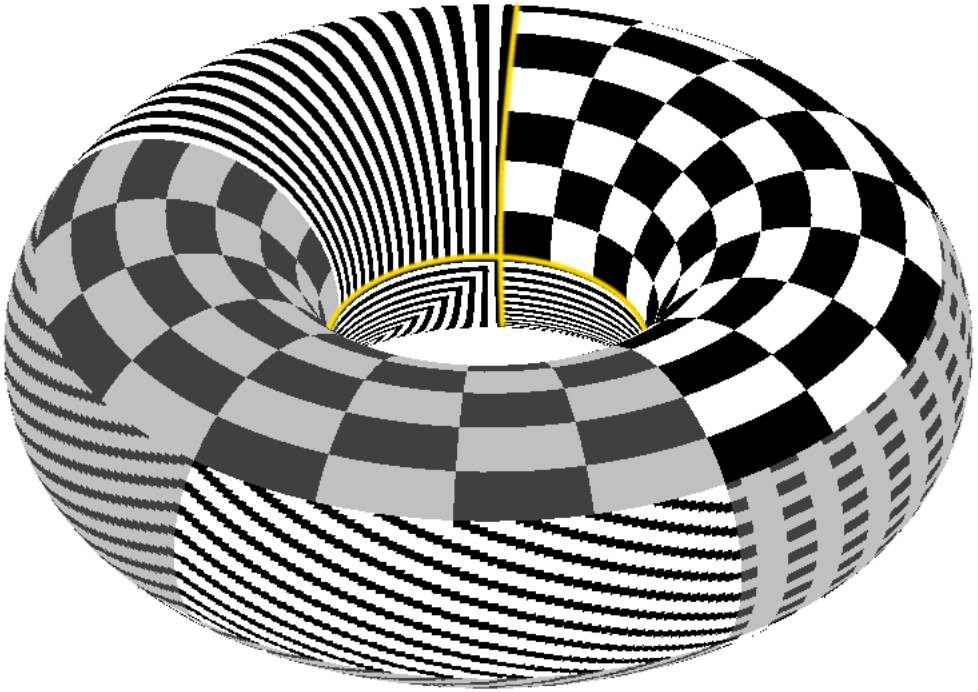
The FT computes the convolution on a torus so opposite sides of the image affect each other.



test pattern image on a torus

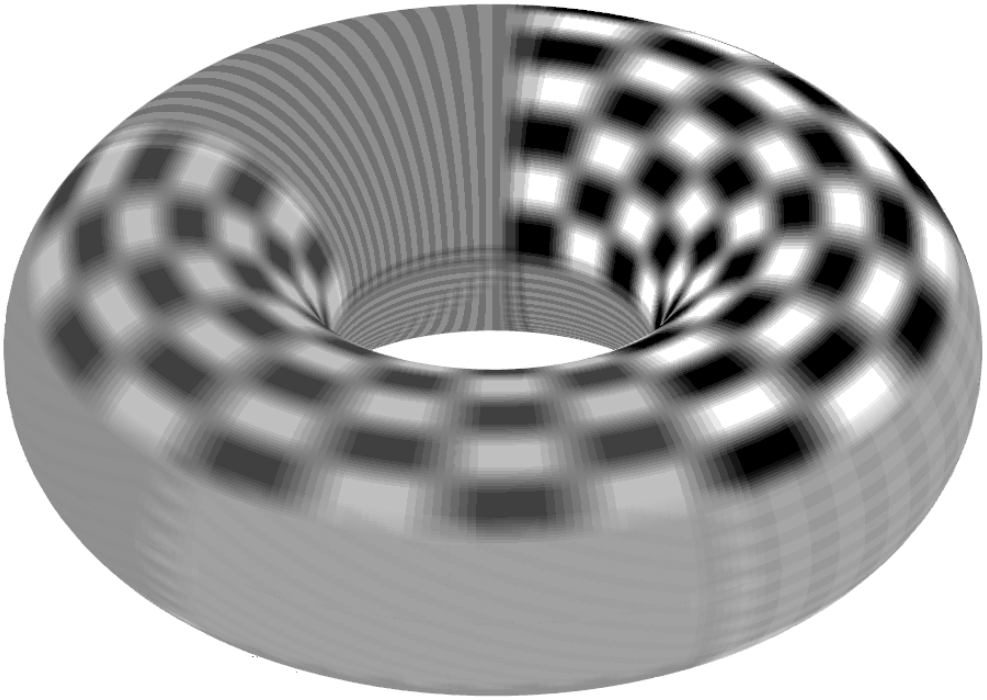
Convolution on a Torus

The effects of convolution on a torus can be seen in the examples that follow.



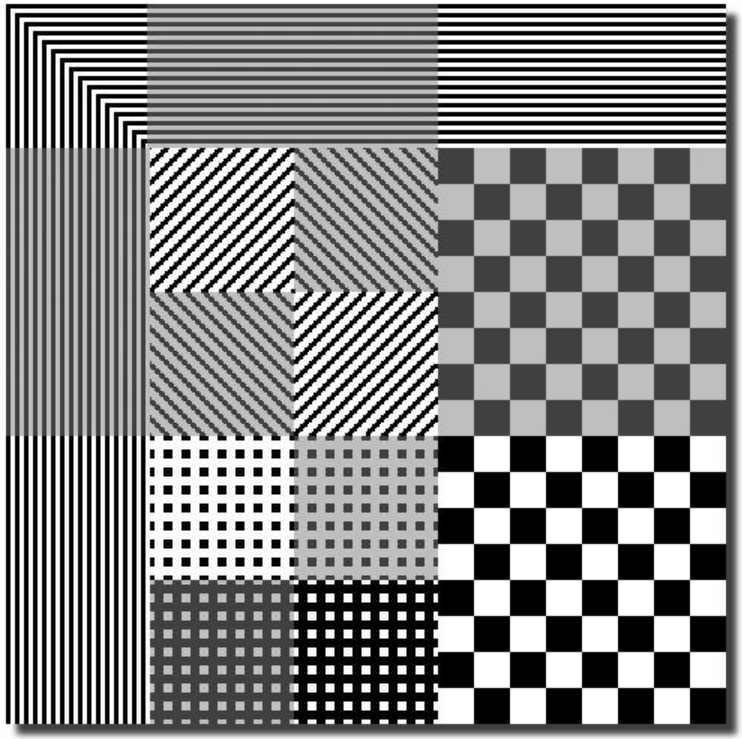
Convolution on a Torus

Here, a 9x9 box filter convolution was applied to the torus.

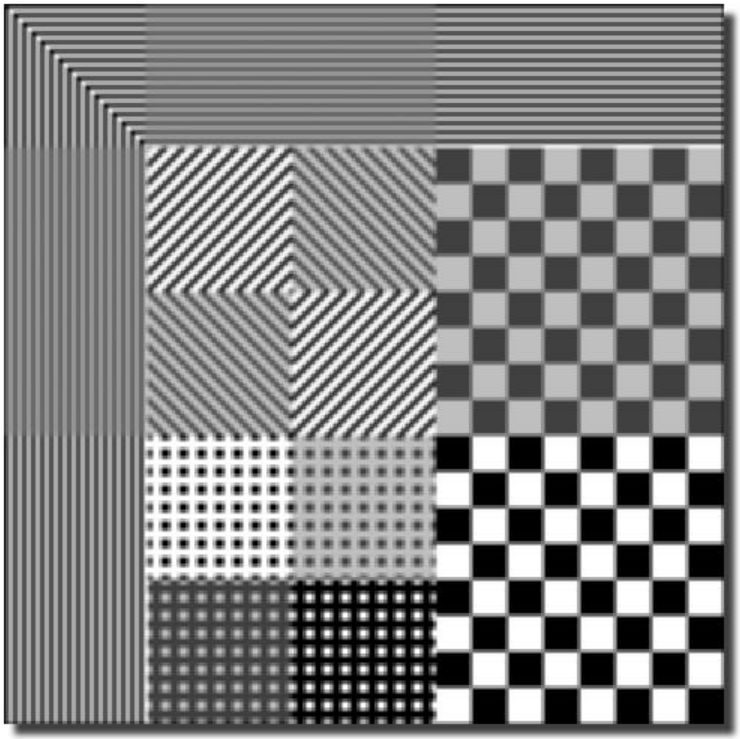


patterns smeared across the boundaries

Convolution on a Torus: Original Images

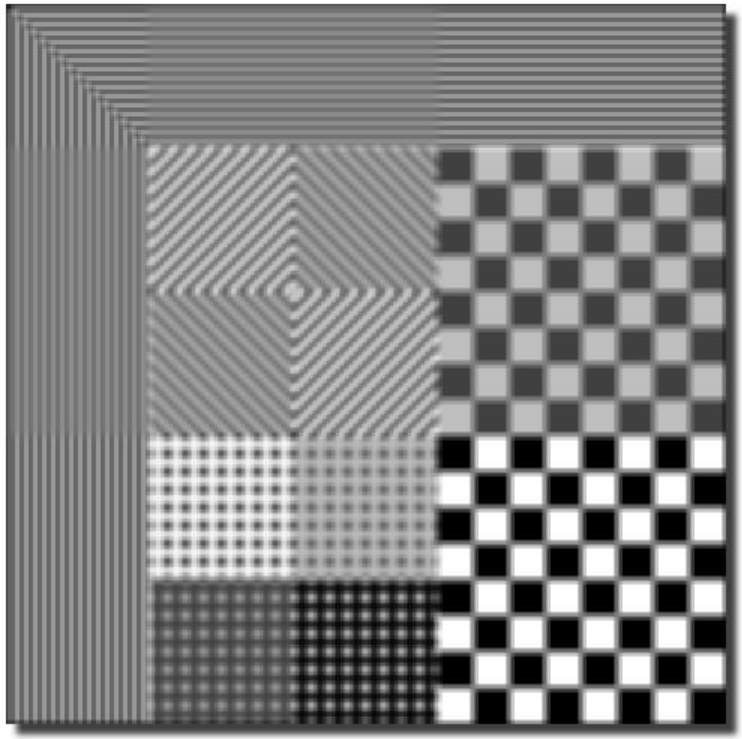


Convolution on a Torus: 3×3 Box Filter



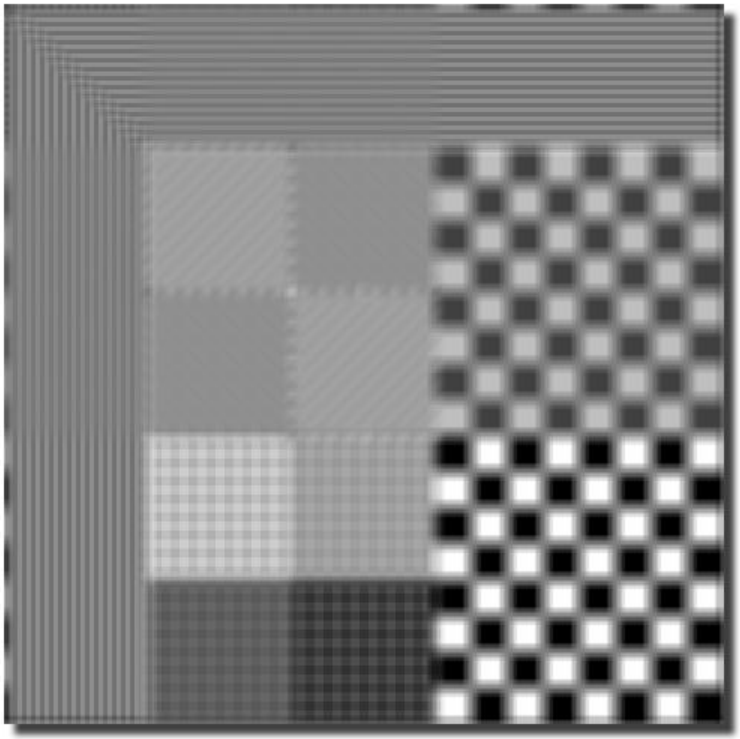
$$\frac{1}{25}\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}$$

Convolution on a Torus: 5×5 Box Filter



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

Convolution on a Torus: 9×9 Box Filter



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

nvolution on a Torus: 17×17 Box Filter