Image Processing

Lecture Notes: Spatial Convolution

Kai-Lung Hua

Spatial Filtering

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

$T[\cdot]$ represents a transformation, such that,

$$J(r,c) = T[I](r,c) = \\ f\left(\left\{I(u,v) \mid u \in \{r-s,\ldots,r,\ldots r+s\}, v \in \{c-d,\ldots,c,\ldots c+d\}\right\}\right).$$

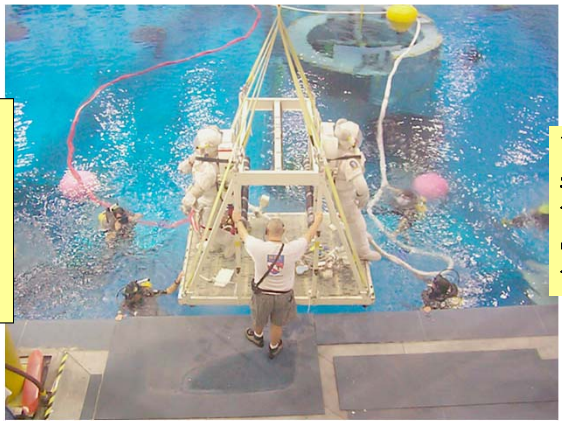
That is, the value of the transformed image, J, at pixel location (r,c) is a function of the values of the original image, I, in a $2s+1 \times 2d+1$ rectangular neighborhood centered on pixel location (r,c).

Moving Windows

- The value, $J(r,c) = \mathrm{T}[I](r,c)$, 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 neighborhood are the same for each location, then transform T is sometimes called a *moving window transform*.

Moving-Window Transformations

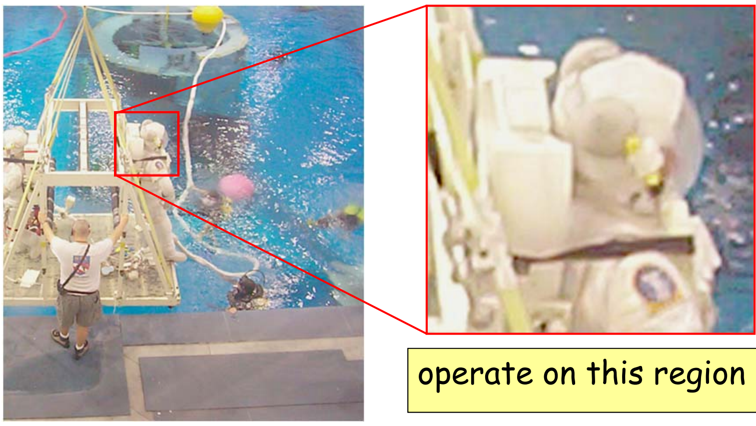
Neutral Buoyancy Facility at NASA Johnson Space Center



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

photo: R.A.Peters II, 1999

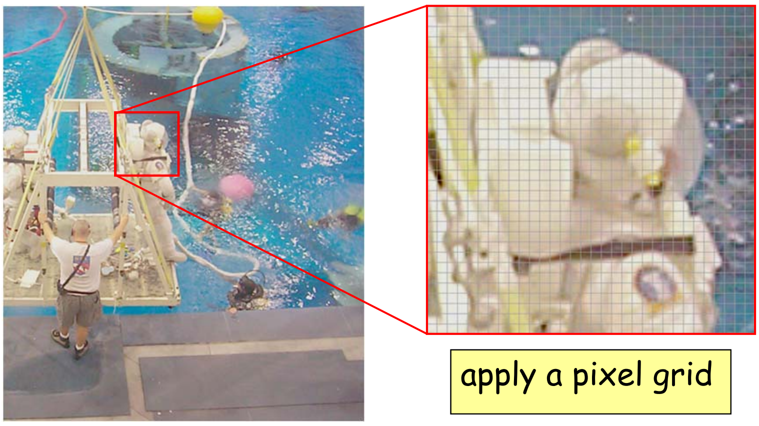
Moving-Window Transformations



operate on this region

Pixelize the section to
better see the effects.

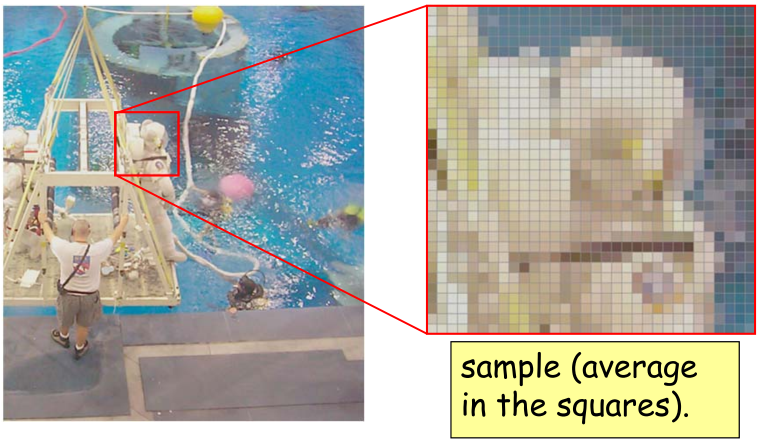
Moving-Window Transformations



apply a pixel grid

Pixelize the section to better see the effects.

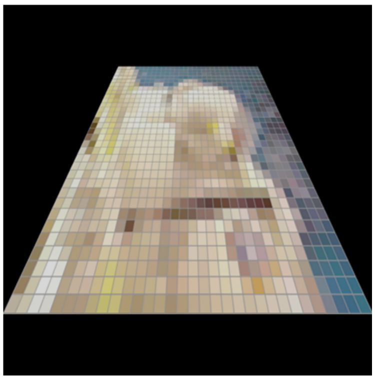
Moving-Window Transformations



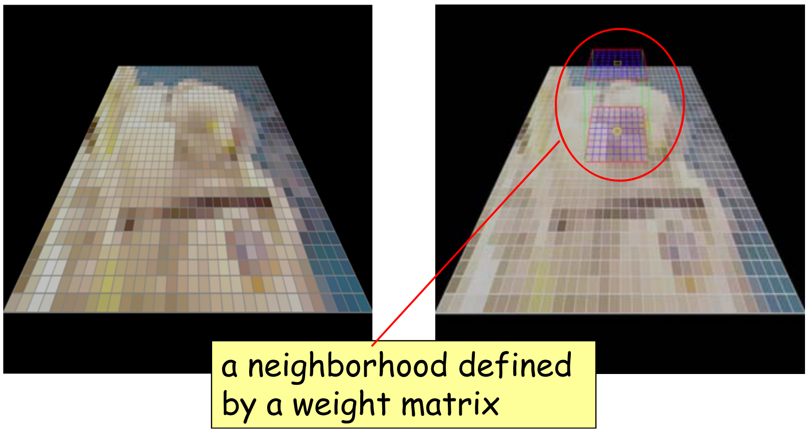
sample (average in the squares).

Moving-Window Transformations

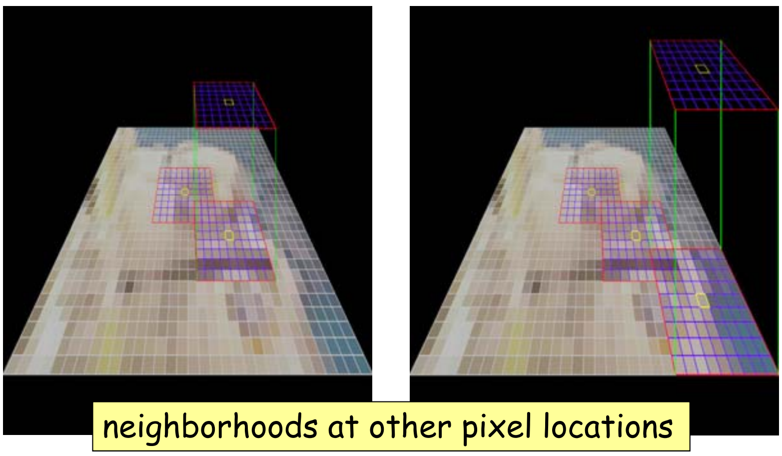
lets get some perspective on this



Moving-Window Transformations

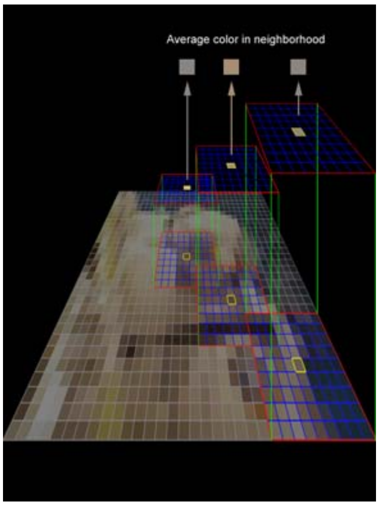


Moving-Window Transformations

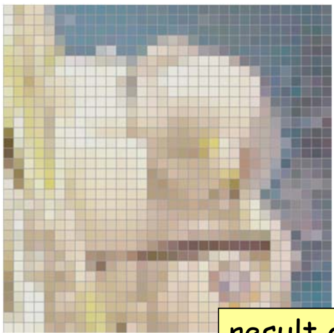


Linear Moving-Window Transformations
(*i.e.* convolution)

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*:

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

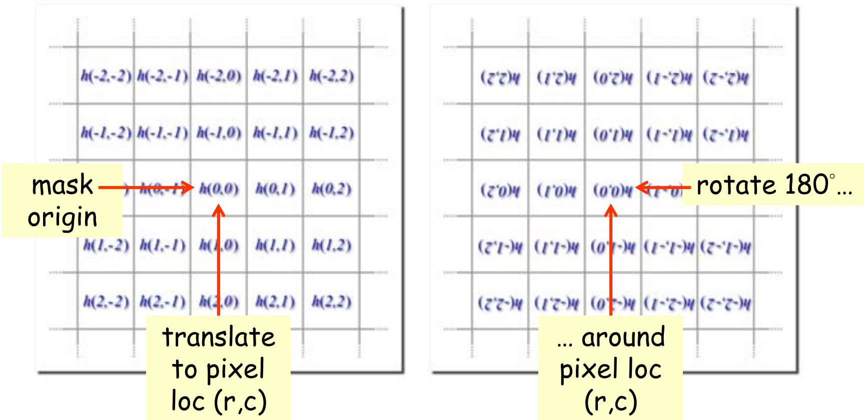
for an ideal, Euclidean image, or for a digital image:

$$J(r,c) = [I * h](r,c) = \sum_{\rho=-\infty}^{\infty}\sum_{\kappa=-\infty}^{\infty} I(r-\rho, c-\kappa)h(\rho,\kappa)$$

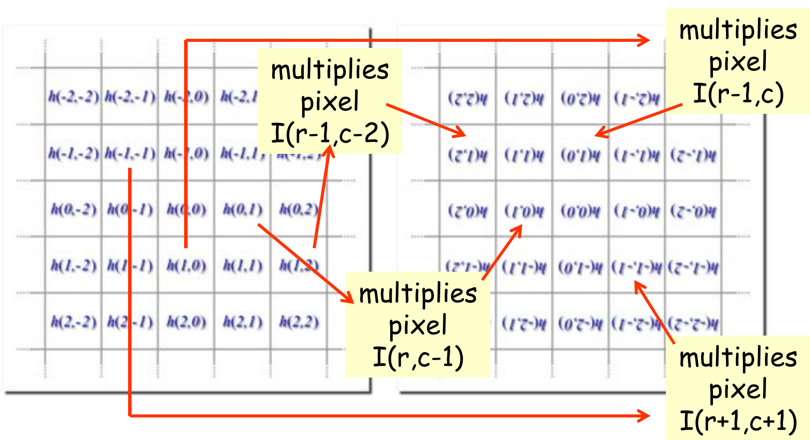
Convolution Mask (Weight Matrix)

- The object, $h(\rho,\kappa)$, 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 180^o. (See slide 22).
- The sum of those products is the value of pixel (r,c) in the output image

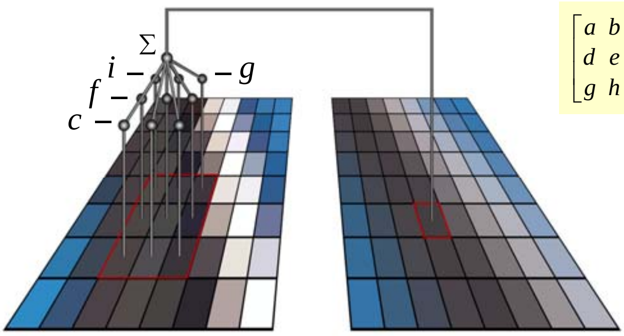
Convolution Masks: Moving Window



Convolution Masks: Moving Window



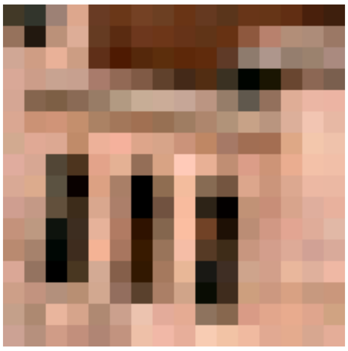
Convolution by Moving Window



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

Another example

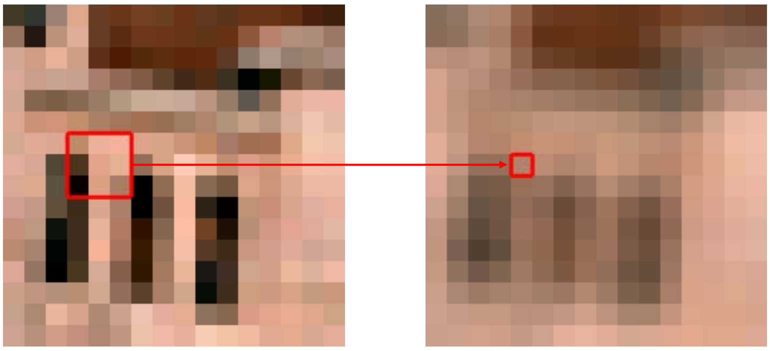
Moving Window Transform: Example



original

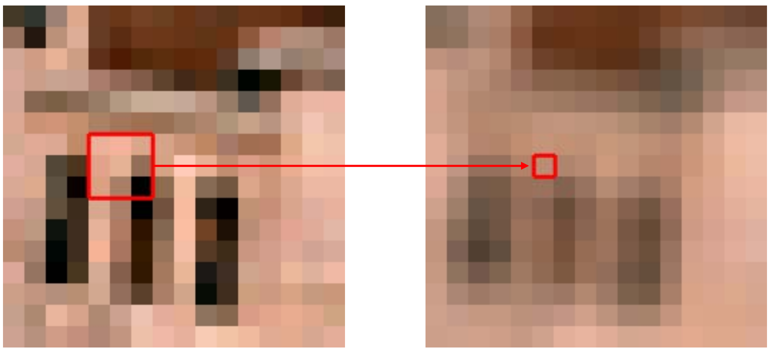
3x3 average

Moving Window Transform: Example



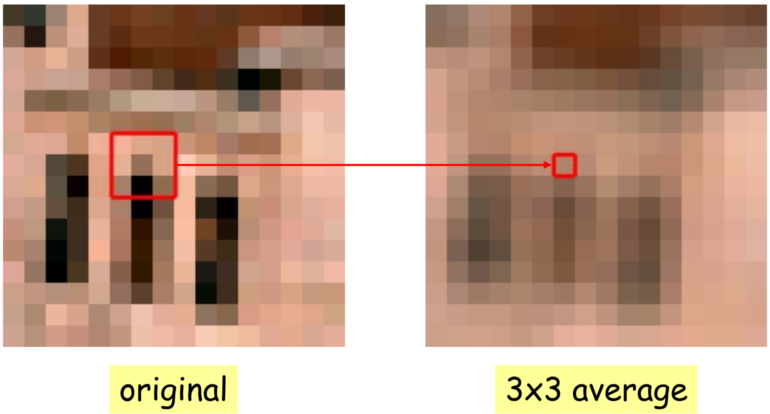
original 3x3 average

Moving Window Transform: Example

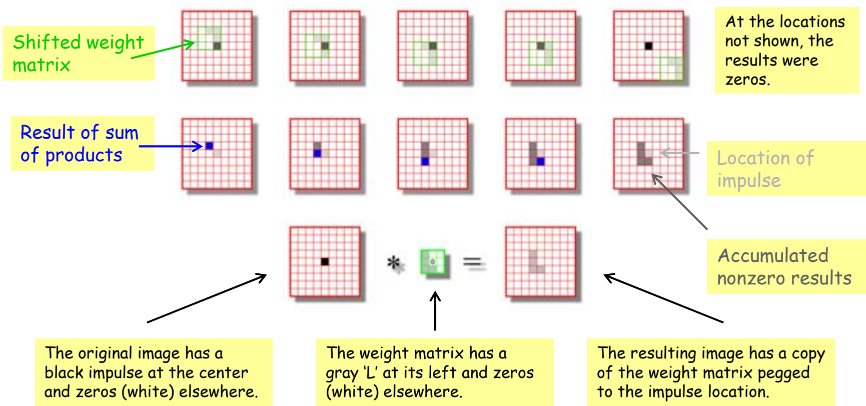


original 3x3 average

Moving Window Transform: Example



Convolution by Rotating and Shifting the Weight Matrix



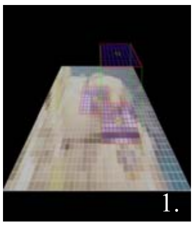
Symmetric Weight Matrix

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°.

Three ways to compute a convolution

1. Moving window transform as just shown.
2. Shift multiply add.
3. Fourier transform.



Shift-Multiply-Add Approach

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

Convolution by an Impulse

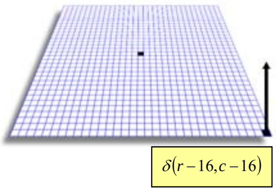
An *impulse* is a digital image, that has a single pixel with value 1; all others have value zero. An impulse at location (ρ, χ) is represented by:

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

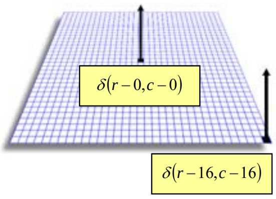
If an image is convolved with an impulse at location (ρ, χ), the image is shifted in location down by r pixels and to the right by χ pixels.

$$[I * \delta(r-\rho, c-\chi)](r,c) = I(r-\rho, c-\chi).$$

Convolution by an Impulse

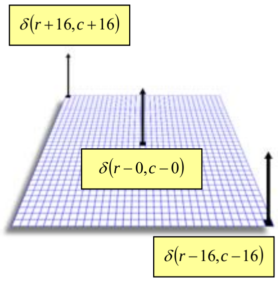


Convolution by Two Impulses



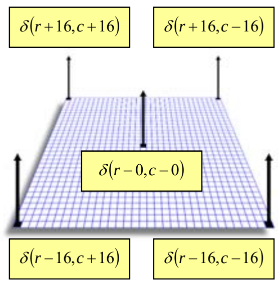
Sum times 1/2

Convolution by Three Impulses



Sum times 1/3

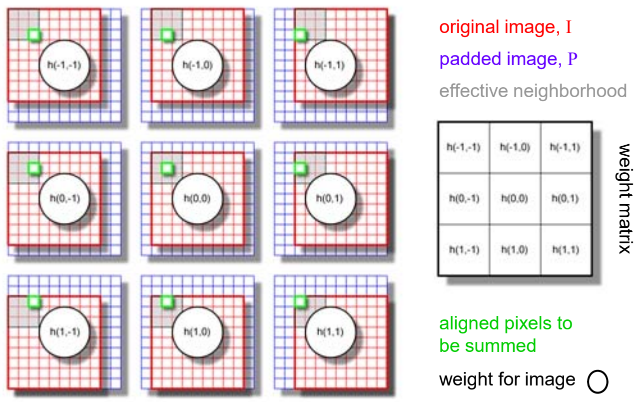
Convolution by Five Impulses



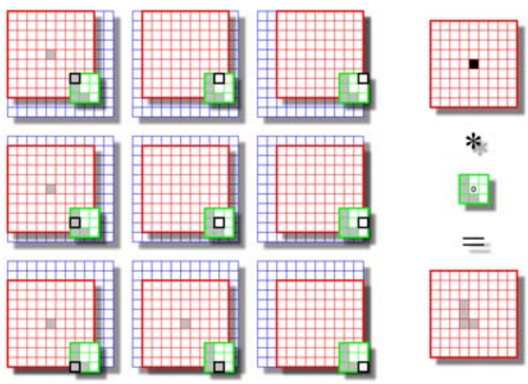
Sum times 1/5

Convolution by Copying, Multiplying, and Shifting the Image

For each element $h(r_h,c_h)$ in weight matrix, *h*, image *I* is copied into a zero-padded image, *P*, starting at (r_h,c_h). Each *P* is multiplied by the corresponding weight, $h(r_h,c_h)$. All the *P* images are summed pixel-wise then divided by the sum of the elements of *h*. The result is cropped out of the center of the accumulated *P*'s.



Convolution by Copying, Multiplying, and Shifting 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.

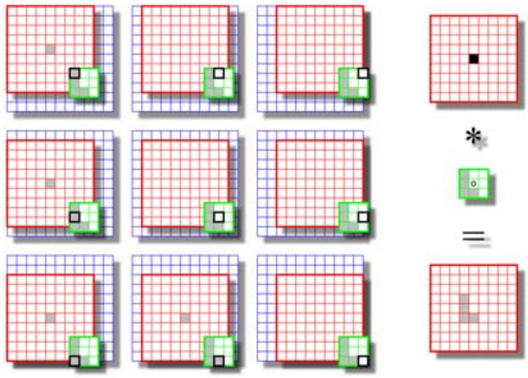
original image, I
effective neighborhood
padded image, P

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 black square (here, white = 0) before being accumulated (pixelwise) in the padded image

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

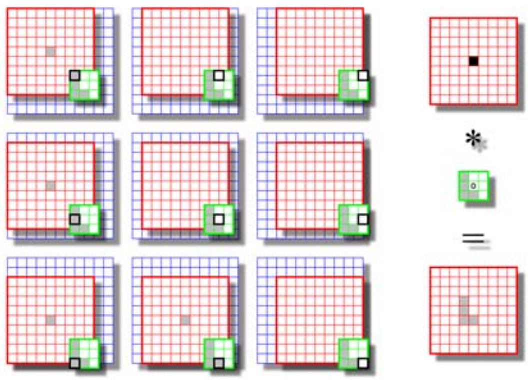


In this image, only the pixel in the center is nonzero so only it shows a result when the image is multiplied by a nonzero value

Convolution by Copying, Multiplying, and Shifting the Image

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

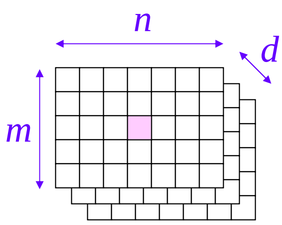
The position of the black square relative the center of the weight matrix indicates the shift of the original image relative to the middle of the padded image.

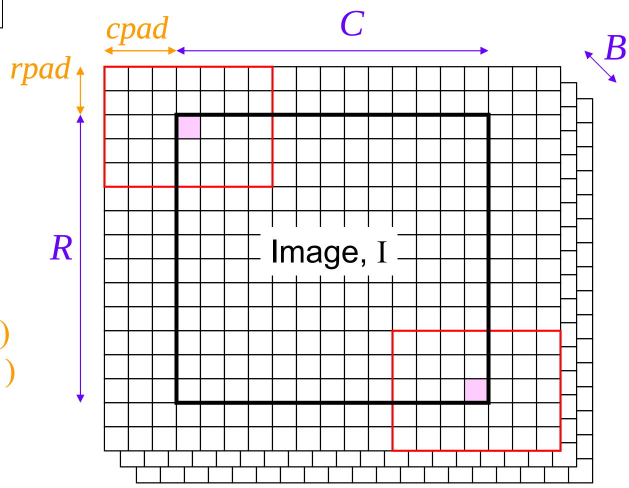


In this image, only the pixel in the center is nonzero so only it shows a result when the image is multiplied by a nonzero value

Zero Padding an Image for Convolution: Variable Names.

weight matrix, *h*

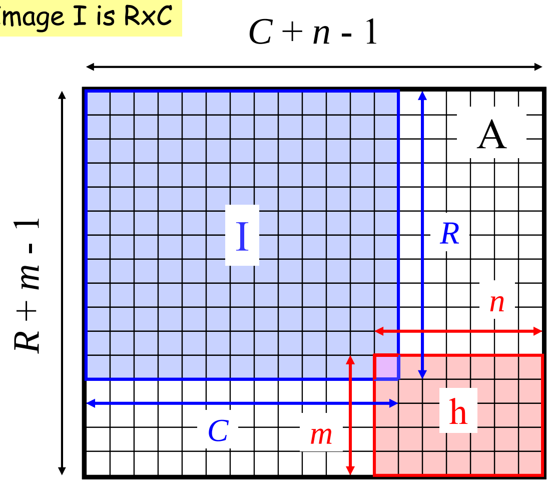




cpad = floor(*n* / 2)
rpad = floor(*m* / 2)
hcorig = *cpad* + 1
hrorig = *rpad* + 1

Convolution by Copying and Shifting the Image

Image I is RxC



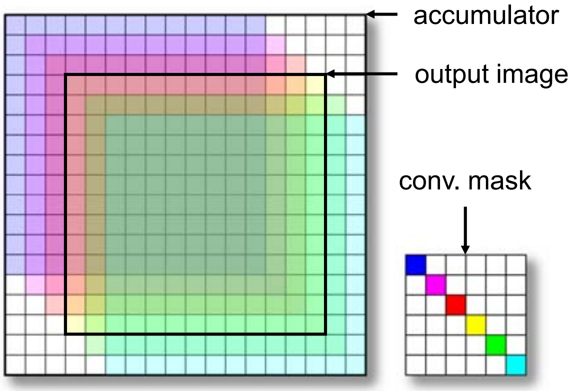
To use the image shift-multiply-accumulate algorithm, create an accumulator image, A, that is R+m-1 rows by C+n-1 columns

Convolution by Copying, Multiplying, and Shifting the Image

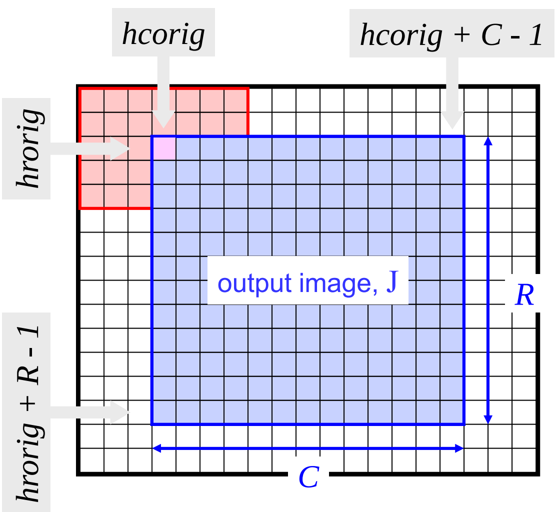
13x13 image convolved by 6x6 mask.

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

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

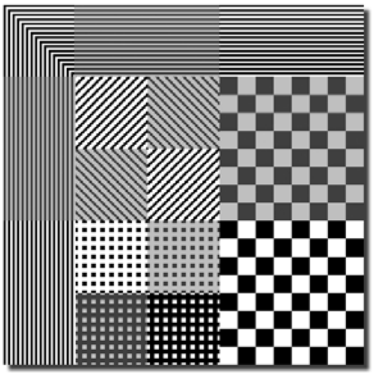


Convolution by Copying and Shifting the Image



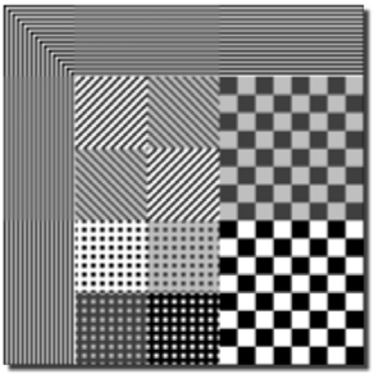
When done, copy the output image from the accumulator starting at (hrorig, hcorig) and ending at (hrorig+R-1, hcorig+C-1)

Convolution Examples: Original Images



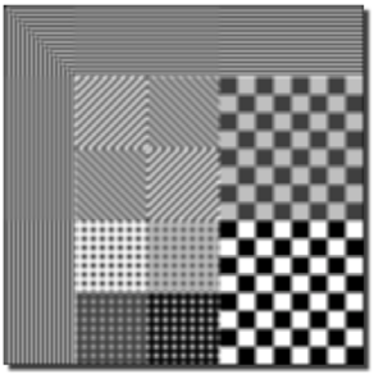
$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}$$

Convolution Examples: 3×3 Blur



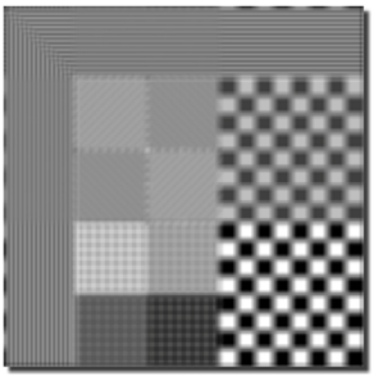
$$\frac{1}{25}\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}$$

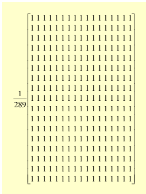
Convolution Examples: 5×5 Blur



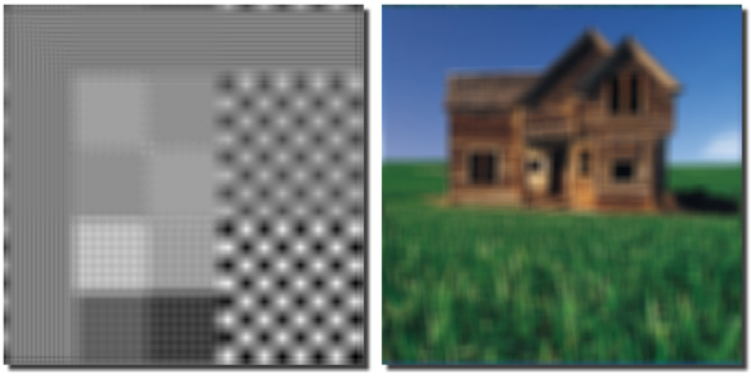
$$\frac{1}{81}\begin{bmatrix}1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&1&1\end{bmatrix}$$

Convolution Examples: 9×9 Blur

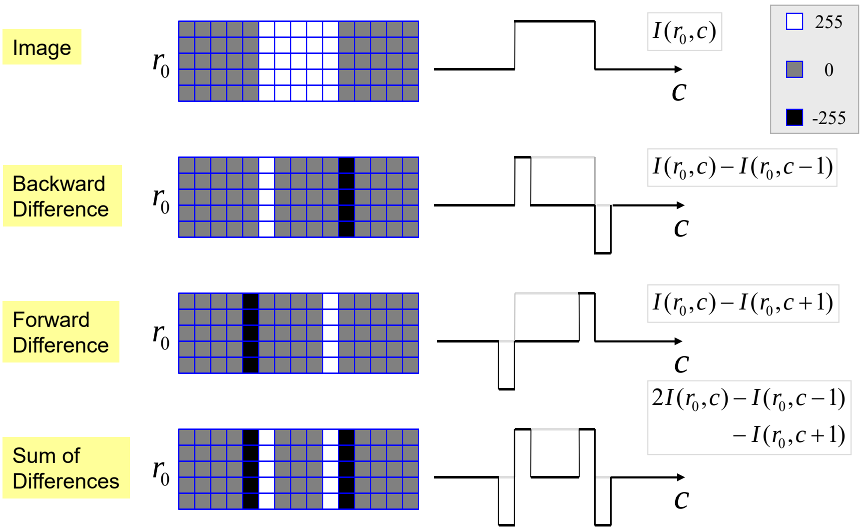




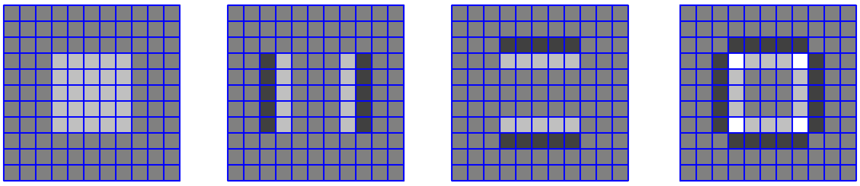
Convolution Examples: 17×17 Blur



Vertical Edge Detection



Symmetric Edge Detection

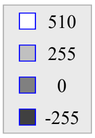


$I(r,c)$

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

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

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

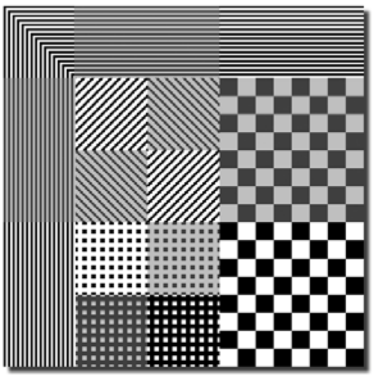


-1	2	-1

	-1	
	2	
	-1	

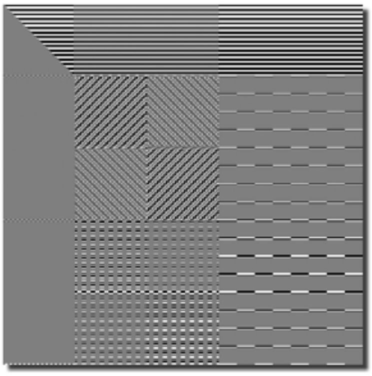
	-1	
-1	4	-1
	-1	

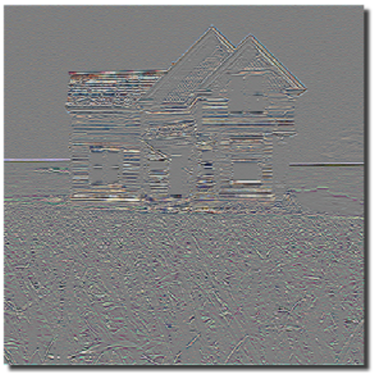
Convolution Examples: Original Images



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

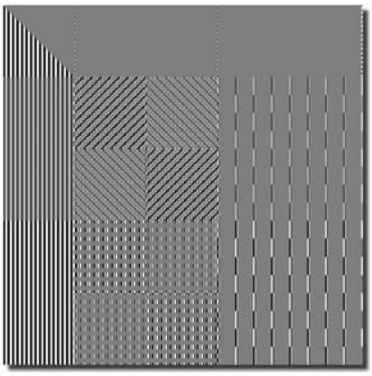
Convolution Examples: Vertical Difference

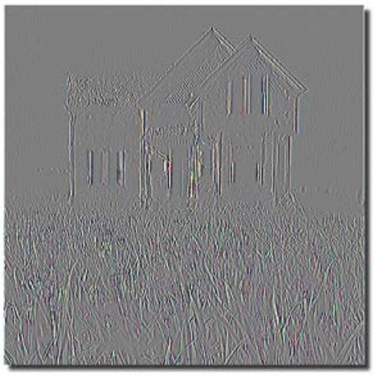




$$\begin{bmatrix}-1 & 2 & -1\end{bmatrix}$$

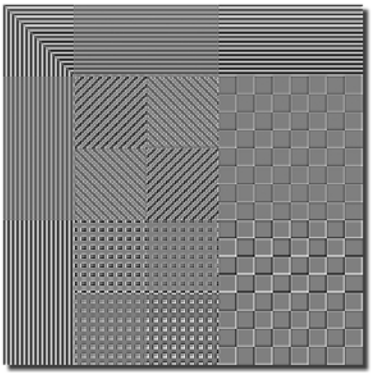
Convolution Examples: Horizontal Difference

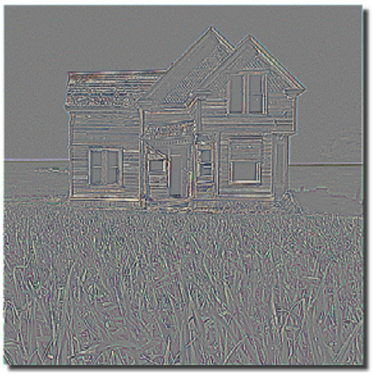




$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

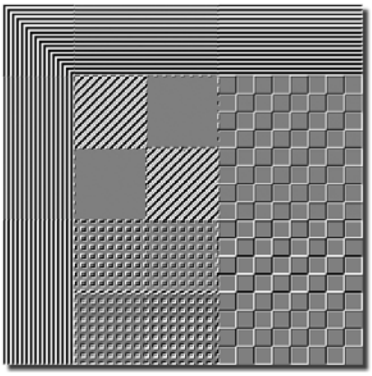
Convolution Examples: H + V Diff.

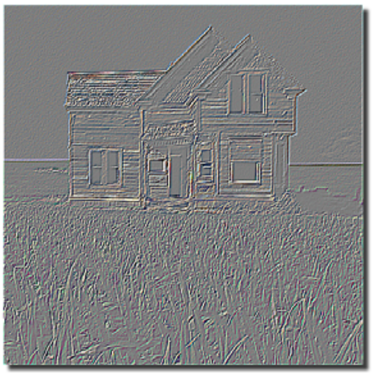




$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

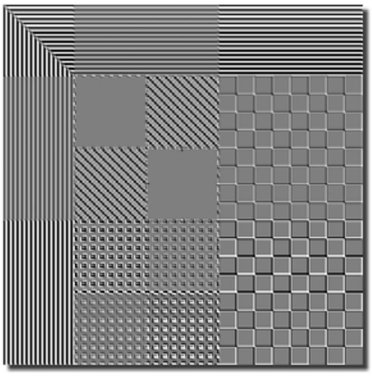
Convolution Examples: Diagonal Difference

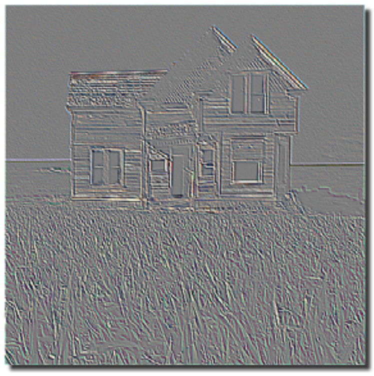




$$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 2 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

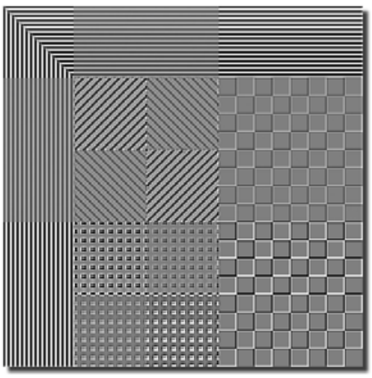
Convolution Examples: Diagonal Difference

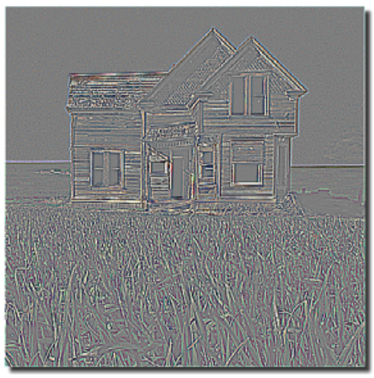




$$\begin{bmatrix} -1 & 0 & -1 \\ 0 & 4 & 0 \\ -1 & 0 & -1 \end{bmatrix}$$

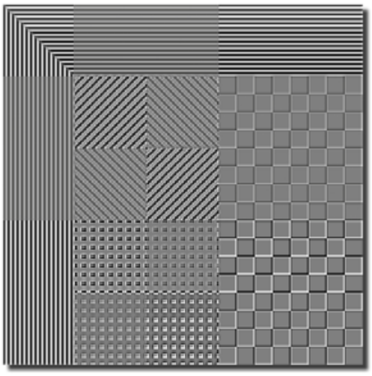
Convolution Examples: D + D Difference

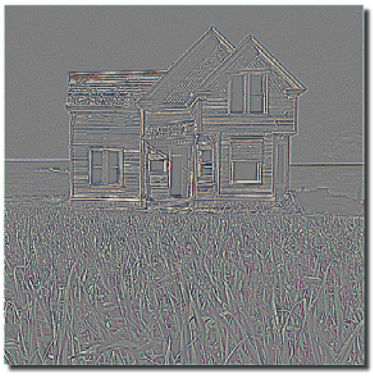




$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Convolution Examples: H + V + D Diff.





Convolution Examples: Original Images