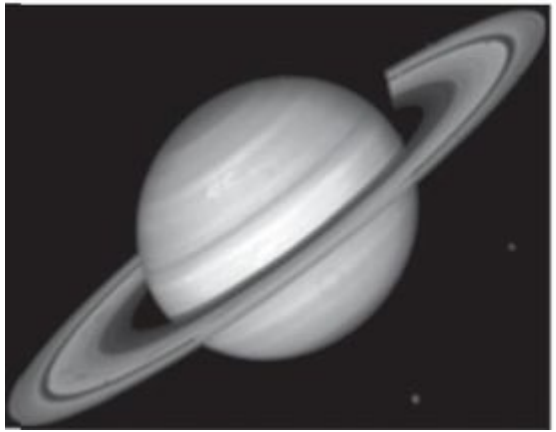
Computer Vision

Image Filtering

Dr. Pratik Mazumder



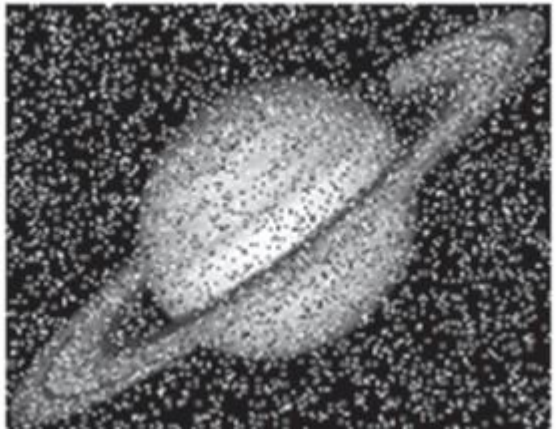


Image Noise

- Image noise refers to random variations of brightness or color information in images often caused by
    - imperfections in the imaging system,
        - Camera Electronics
        - Quality of Lens
    - environmental conditions,
        - Light Variations
        - Surface Reflectance
    - transmission errors
- Unwanted noise has to be removed or filtered out.

Image Noise

- $I_{original}(x,y)$: true pixel value at (x,y)
- n(x,y) : noise at (x,y)
- $I_{observed}(x,y) = I_{original}(x,y) + n(x,y)$ {additive noise}

Image Noise

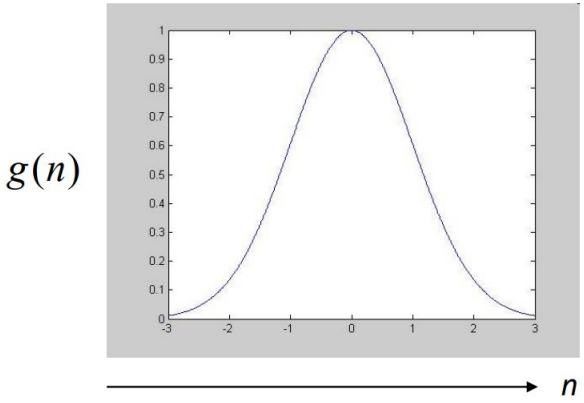
- $I_{original}(x,y)$: true pixel value at (x,y)
- n(x,y) : noise at (x,y)
- $I_{observed}(x,y) = I_{original}(x,y) * n(x,y)$ {multiplicative noise}

Noise

Noise can be assumed to be from a distribution, e.g., a Gaussian $\rho(n) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(n-\mu)^2/(2\sigma^2)}$

Gaussian distribution with mean 0

$$n(x,y) \approx g(n) = e^{\frac{-n^2}{2\sigma^2}}$$



Probability Distribution
n is a random variable

Salt and pepper noise

Pixels are randomly made black or white with a uniform probability distribution.

➔

Salt-pepper

DeNoise

FILTERING

Image filtering

Computes a function of the local neighborhood at each position in the image

Enhance images

- Denoise, resize, increase contrast, etc.
- Extract information from images
- Texture, edges, distinctive points, etc.
- Detect patterns
- Template matching

Image filtering

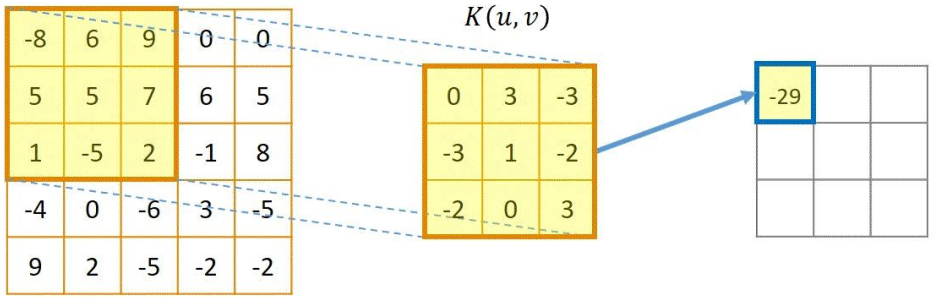
Computes a function of the local neighborhood at each position in the image.

h=output f=filter I=image

$$h[m,n] = \sum_{k,l} f[k,l]\, I[m+k, n+l]$$

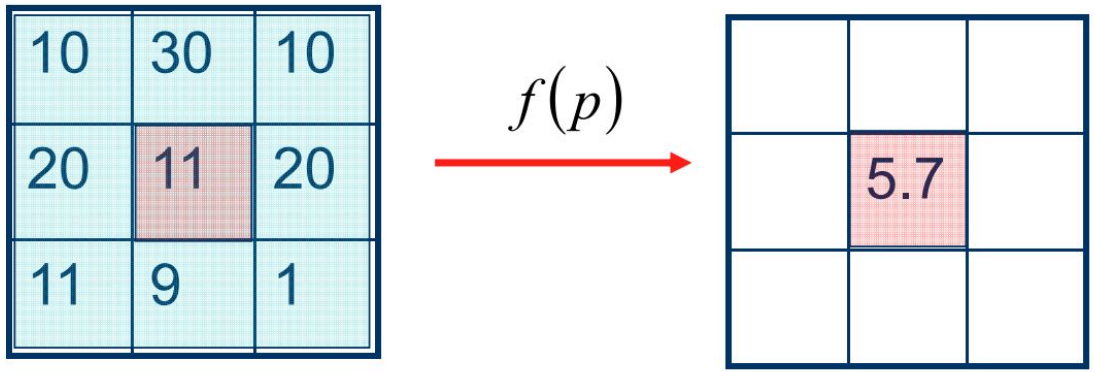
2d coords=k,l 2d coords=m,n

[ ] [ ] [ ]



Filtering

- Output value at a pixel is based on some function of the neighborhood

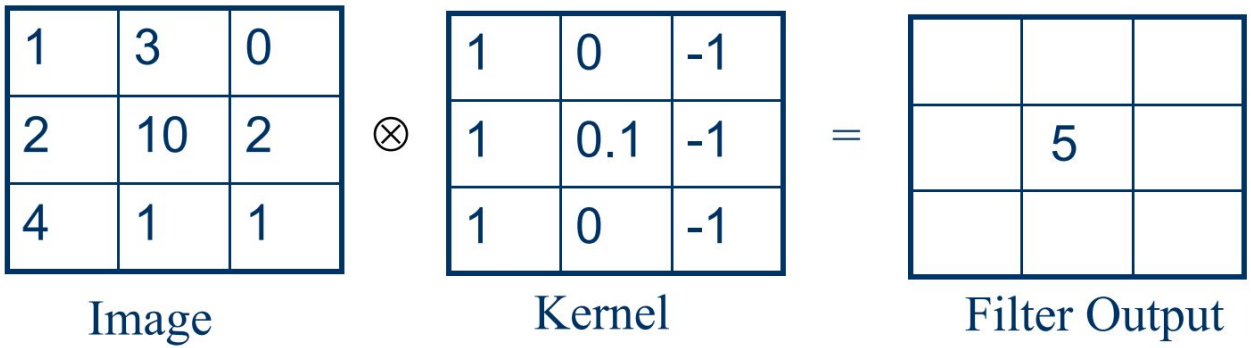


Linear Filtering

- Replace each pixel by a linear combination of its neighbors (and possibly itself).
- The combination is determined by the filter’s kernel.
- The same kernel is shifted to all pixel locations so that all pixels use the same linear combination of their neighbors.
- Linear filtering is shift-invariant
  - The filter will produce a similar output for similar neighbours
  - Can be considered as a pattern identifier - **USEFUL**

Linear Filtering

- The output is the linear combination of the neighborhood pixels



Convolution and Correlation

Definition of filtering as convolution:

notice the flip

$$(f * I)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j)I(x - i, y - j)$$

Definition of filtering as correlation:

notice the lack of a flip

$$(f * I)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j)I(x + i, y + j)$$

Convolution and Correlation

Convolution is associative.
F ✶ (G ✶ I) = (F ✶ G) ✶ I

Convolution is commutative.

What about correlation?

$$(f * I)(x, y) = \sum_{i,j} f(i, j) I(x - i, y - j)$$

$$(I * f)(x, y) = \sum_{i,j} I(i, j) f(x - i, y - j)$$

Let u = x - i, v = y - j

Therefore, i = x - u, j = y - v

$$(I * f)(x, y) = \sum_{u,v} I(x - u, y - v) f(u, v)$$

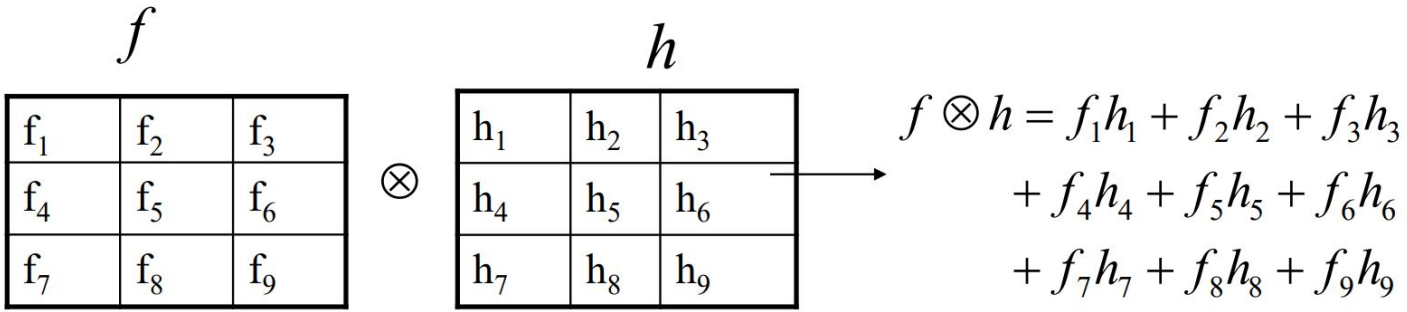
= identical to $(f * I)(x, y)$

Correlation

$$f \otimes h = \sum_k \sum_l f(k,l)h(k,l)$$

$f = \text{Image}$

$h = \text{Kernel}$

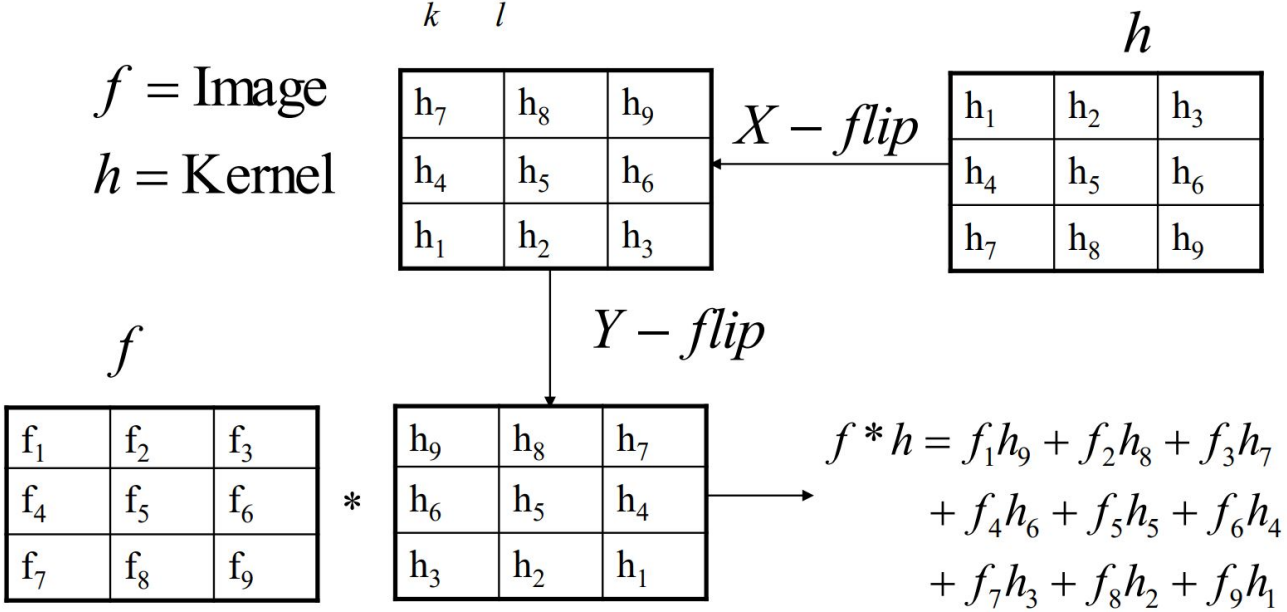


Convolution

$$f * h = \sum_k \sum_l f(k,l)h(-k,-l)$$

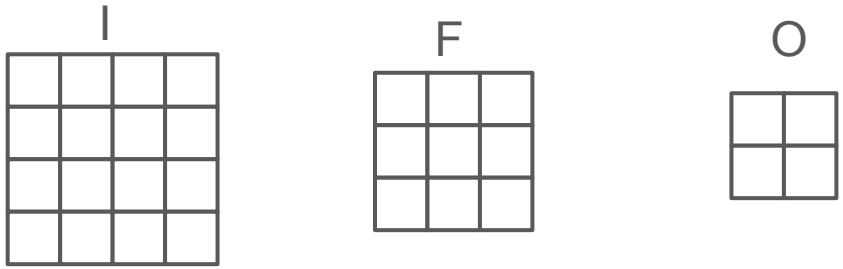
$f = \text{Image}$

$h = \text{Kernel}$



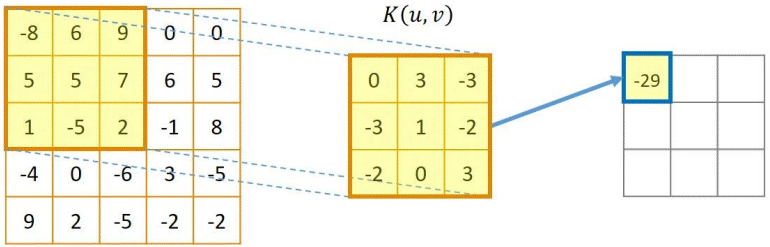
$$f * h = f_1h_9 + f_2h_8 + f_3h_7 + f_4h_6 + f_5h_5 + f_6h_4 + f_7h_3 + f_8h_2 + f_9h_1$$

Filtering

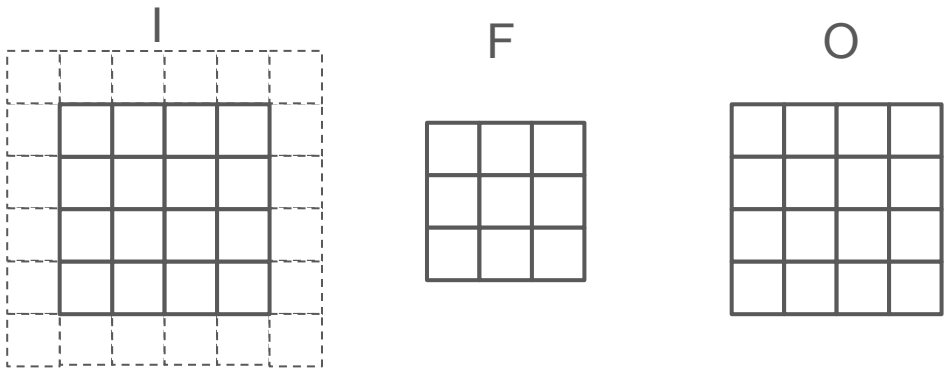


Output Height $O_H = \lfloor (I_H - F + 2P)/S \rfloor + 1$

Output Width $O_w = \lfloor (I_W - F + 2P)/S \rfloor + 1$



Filtering

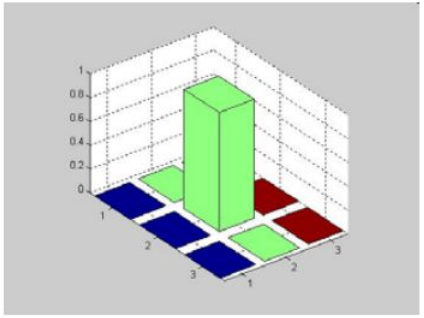


Output Height $O_H = \lfloor (I_H - F + 2P)/S \rfloor + 1$

Output Width $O_w = \lfloor (I_W - F + 2P)/S \rfloor + 1$

- Pad with a constant
- Pad with reflection

Filtering Examples



*

0	0	0
0	1	0
0	0	0

=

Box Filter

- also known as the 2D rect filter
- also known as the square mean filter

kernel $g[\cdot,\cdot] = \frac{1}{9}$

1	1	1
1	1	1
1	1	1

- replaces pixel with local average
- has smoothing (blurring) effect

Box Filter

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

$$h[m,n] = \sum_{l} g[k,l]\, f[m+k,n+l]$$

S.

Box Filter

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10							

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

Box Filter

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

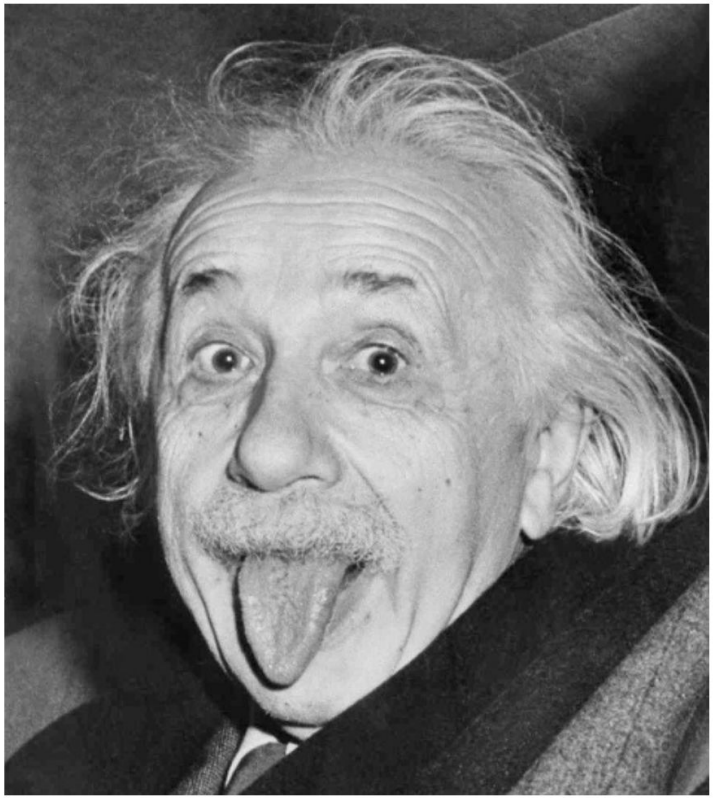
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

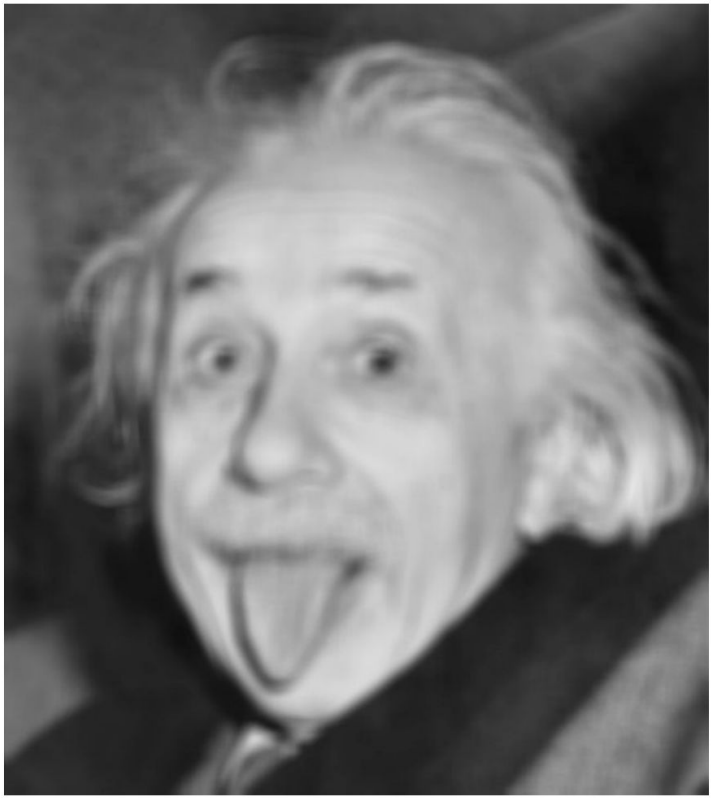
$h[.,.]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

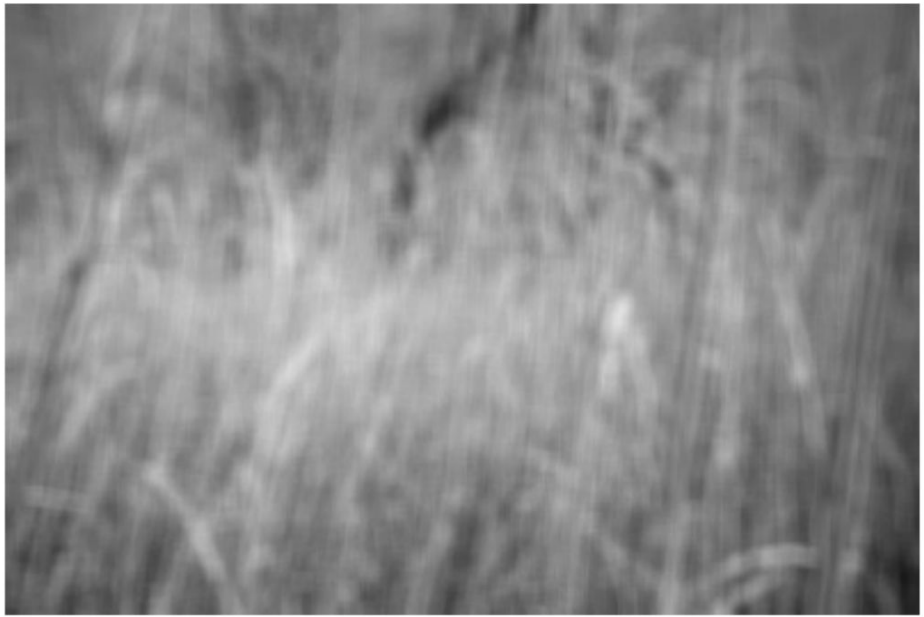
$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

Box Filter

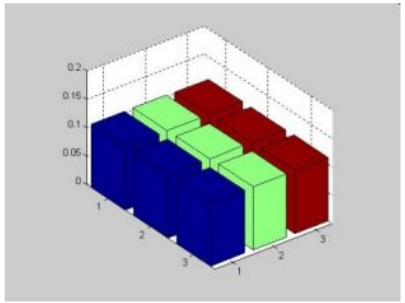




Box Filter

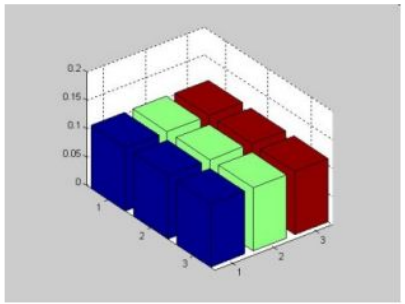


Filtering Examples



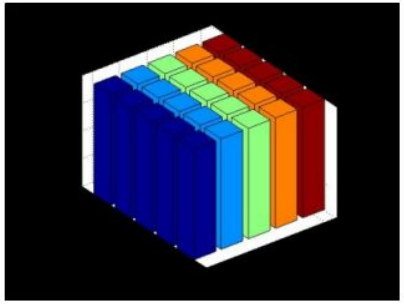
$$* \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$

Filtering Examples



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

Filtering Examples

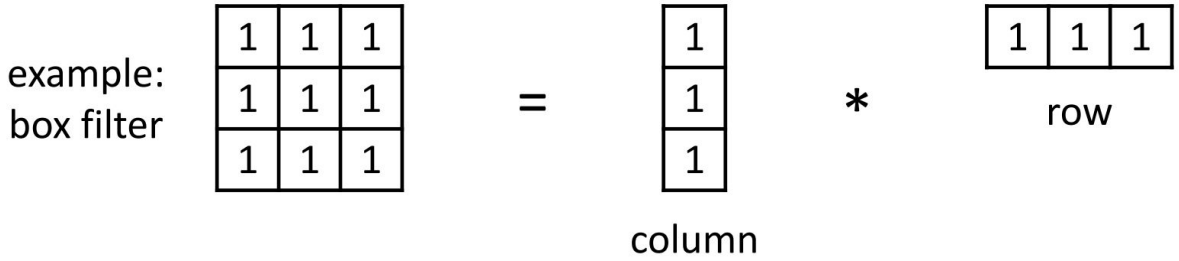


Any Change?

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

Separable filters

- A 2D filter is separable if it can be written as the product of a “column” and a “row”.



Background: Rank of a Matrix

- The maximum number of linearly independent row vectors of a matrix A, or
- The maximum number of linearly independent column vectors of a matrix A.

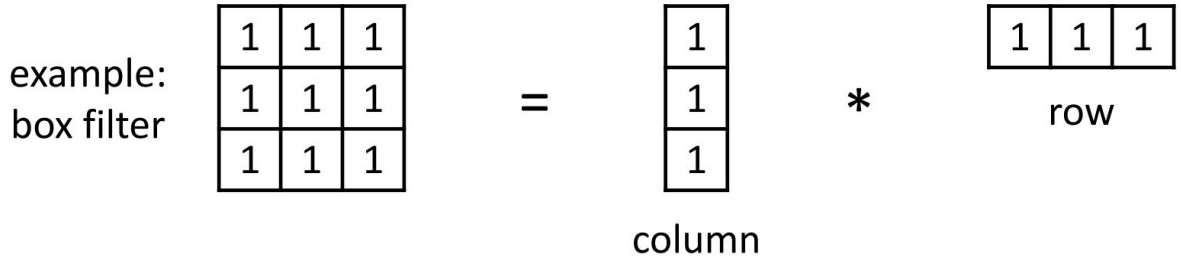
- Check each row vector
  - If a row vector cannot be written as a linear combination of the other row vectors
    - Add 1 to the number of linearly independent row vectors.

- A matrix is of **full rank** if its rank is the same as its smaller dimension,
  - i.e., Suppose $A \in R^{nxm}$ and $n<m$. If $rank(A)=n$, then it is full rank.

- A matrix that is not full rank is **rank deficient,** and the rank deficiency is the difference between its smaller dimension and the rank.
  - e.g. if $rank(A)<n$ in the above example

Separable filters

- A 2D filter is separable if it can be written as the product of a “column” and a “row”.



- What is the rank of this filter matrix?
  Rank = 1

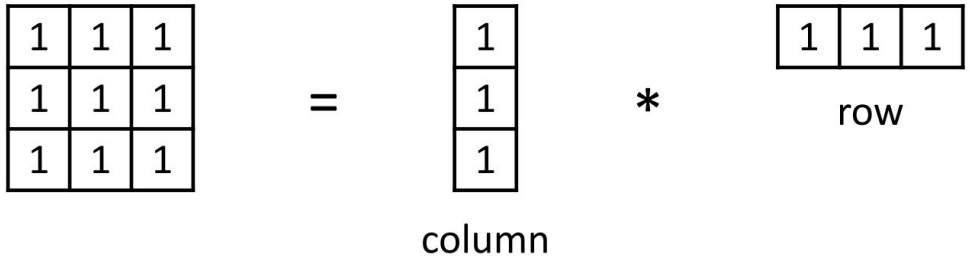
- Why is this important?
  A rank-1 matrix can be represented as the outer product of a column vector and a row vector.

- 2D convolution with a separable filter is equivalent to two 1D convolutions (with the “column” and “row” filters)

Separable filters

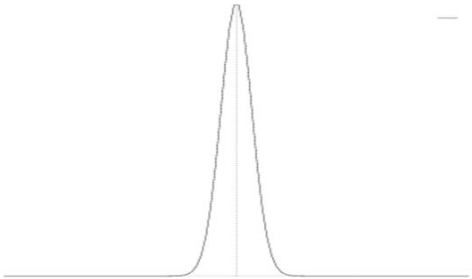
- A 2D filter is separable if it can be written as the product of a “column” and a “row”.

example:
box filter



- If the image has M x M pixels and the filter kernel has size N x N:
  - What is the cost of convolution with a non-separable filter?
    - ≈ M^2 x N^2
  - What is the cost of convolution with a separable filter?
    - ≈ 2 x M^2 x N

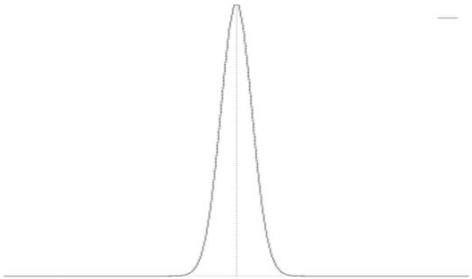
Gaussian Filter

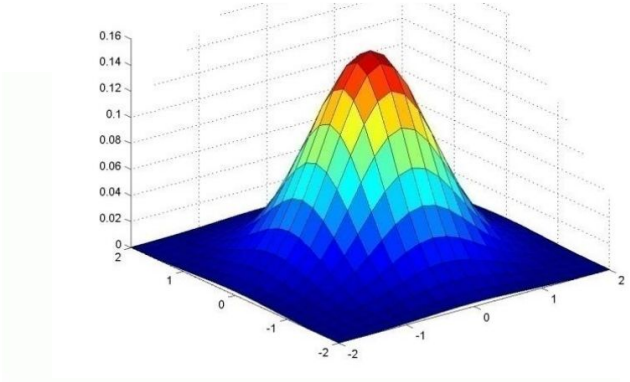


$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)}$$

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/(2\sigma^2)}$$

Gaussian Filter



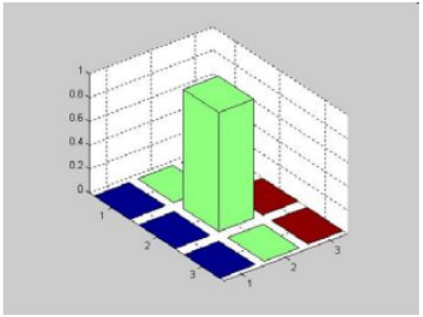


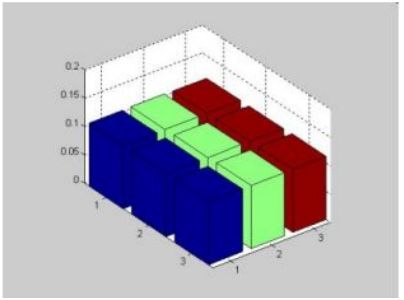
$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)}$$

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/(2\sigma^2)}$$

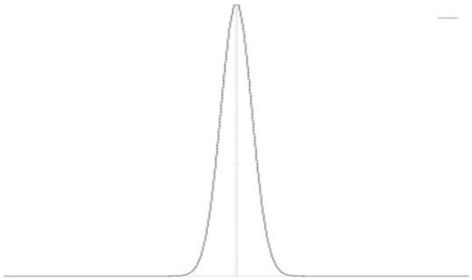
$$g(x) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Previously seen Filters



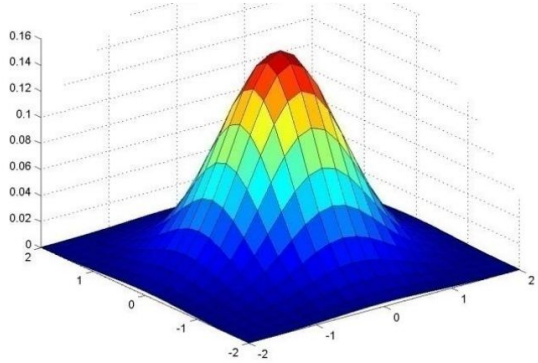


Gaussian Filter

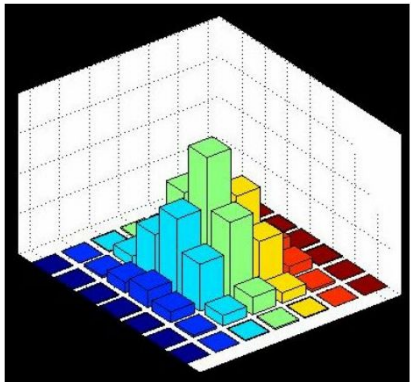


$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)}$$

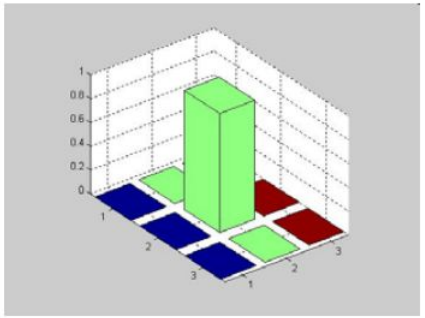
$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/(2\sigma^2)}$$

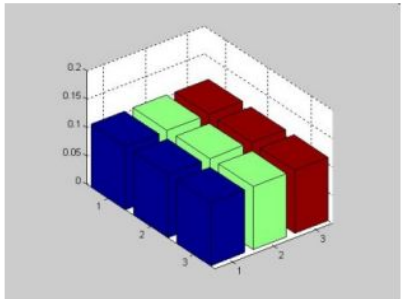


$$g(x) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



Previously seen Filters



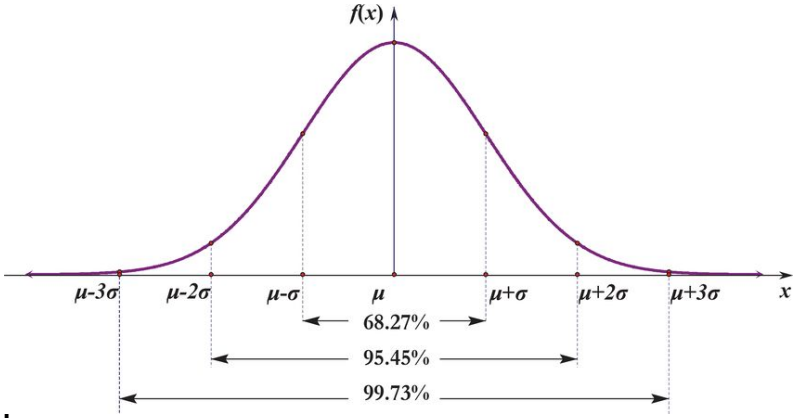


Gaussian Filter

- kernel values sampled from the 2D Gaussian function

$$g(x) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- **Weight falls off with distance from center pixel**
- Theoretically infinite, in practice truncated to some maximum distance - usually at 2 to 3σ
- **3 sigma rule** - for normally distributed data, almost all observed data will fall within three standard deviations of the mean or average



Gaussian Filter

kernel $\frac{1}{16}$

1	2	1
2	4	2
1	2	1

Gaussian Filter

Is this a separable filter?

kernel $\frac{1}{16}$

1	2	1
2	4	2
1	2	1

Gaussian Filter

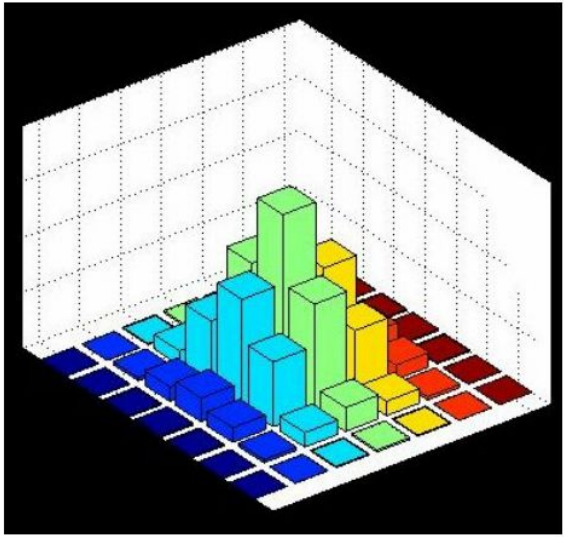
Is this a separable filter? Yes!

kernel $\frac{1}{16}$

1	2	1
2	4	2
1	2	1

Filtering Gaussian

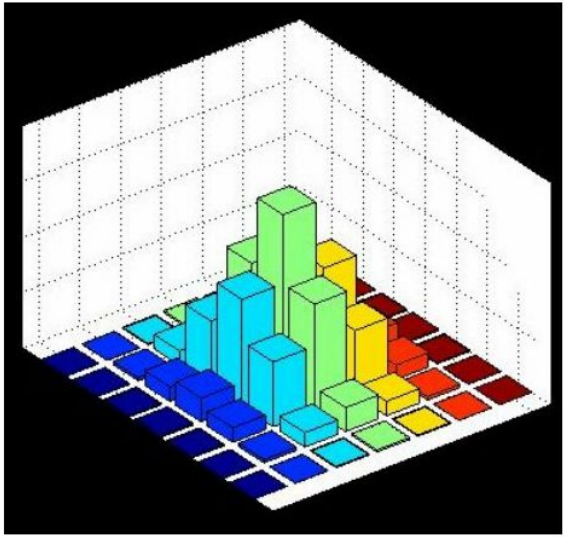
*



=

Filtering Gaussian

*



=

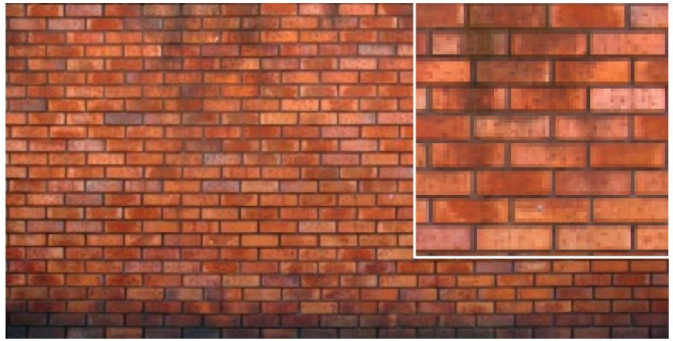
Gaussian smoothing

Filtering Examples

Gaussian Filter

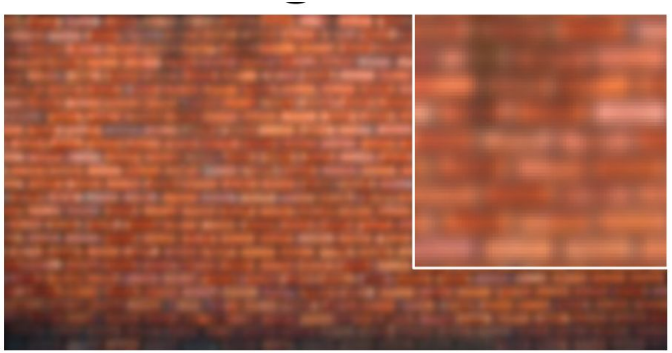
Box Filter

Filtering Examples

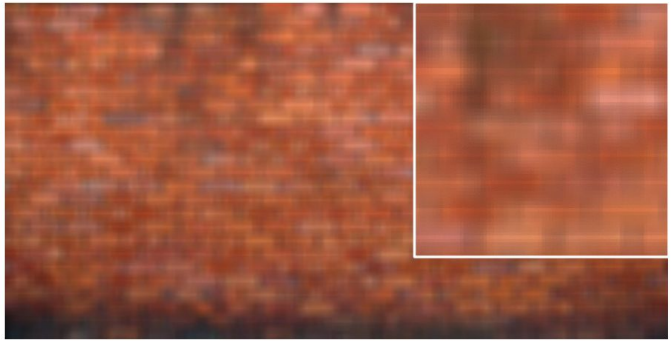


original

Which blur do you like better?



7x7 Gaussian



7x7 box

Noise Filtering

After additive
Gaussian Noise

After Averaging

After Gaussian Smoothing

Filtering Examples

input

filter

0	0	0
0	0	1
0	0	0

output

?

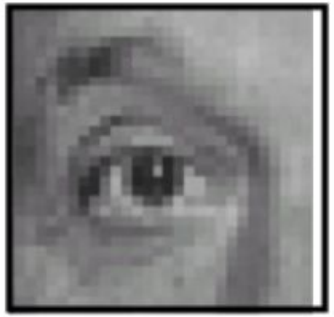
Filtering Examples

input

filter

0	0	0
0	0	1
0	0	0

output



shift to left by one

Filtering Examples

input

filter

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

output

?

Filtering Examples

input

filter

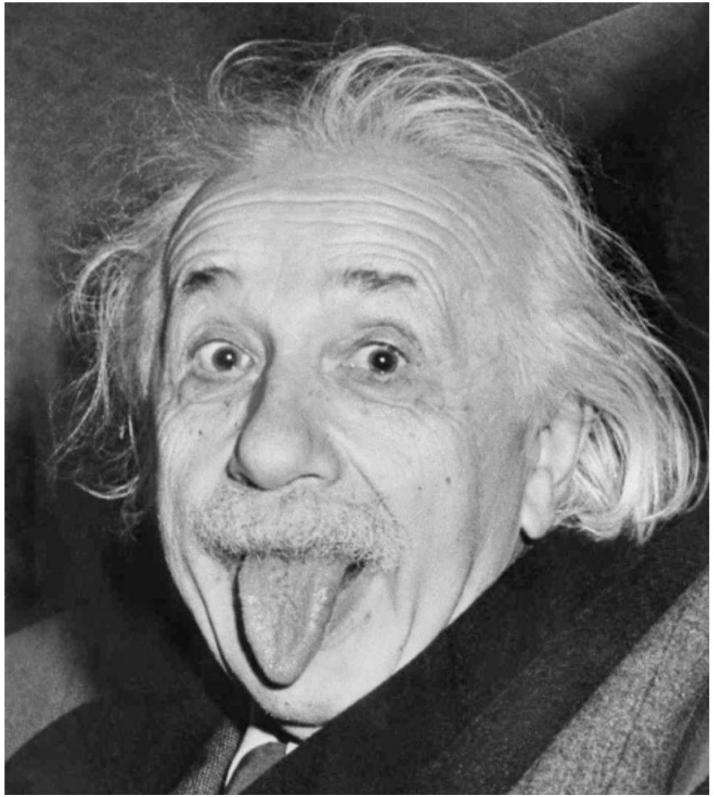
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

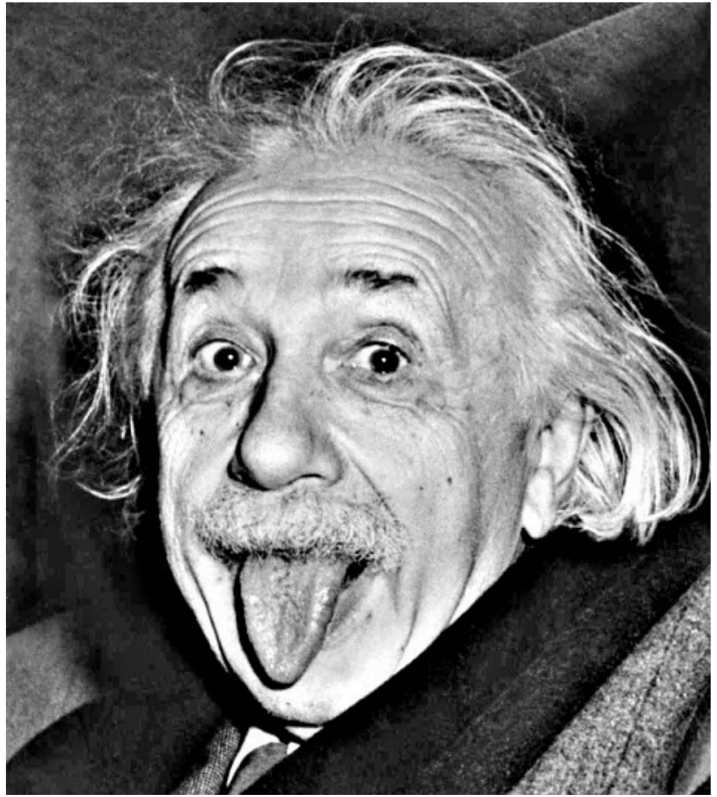
output

sharpening

- do nothing for flat areas
- stress intensity peaks

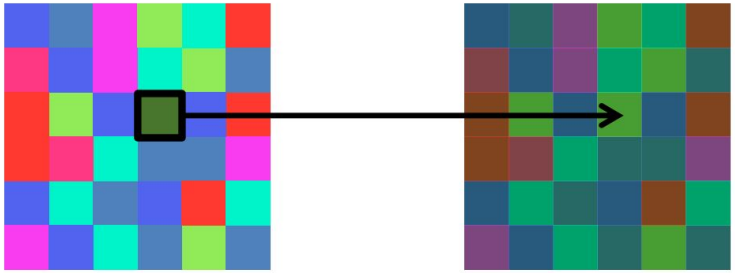
Sharpening Examples





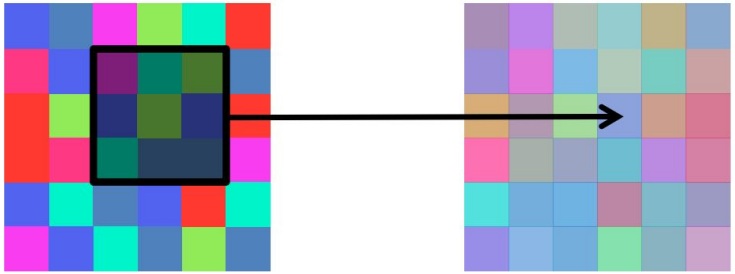
What types of image filtering can we do?

Point Operation



point processing

Neighborhood Operation



“filtering”

Examples of point processing

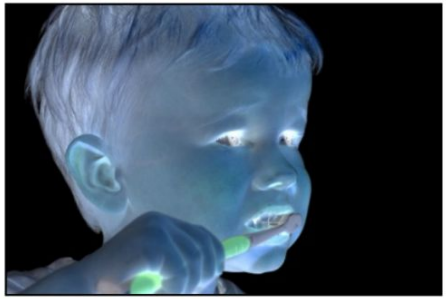
original

darken

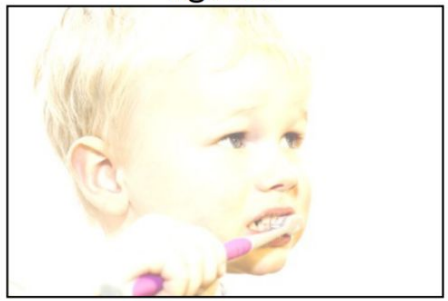
lower contrast

non-linear lower contrast

invert



lighten



raise contrast

non-linear raise contrast

Examples of point processing

original

x

darken

$x - 128$

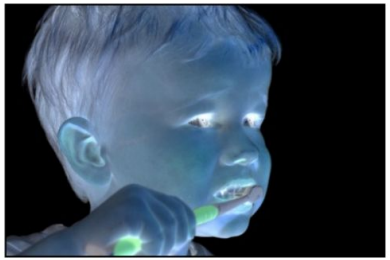
lower contrast

$\frac{x}{2}$

non-linear lower contrast

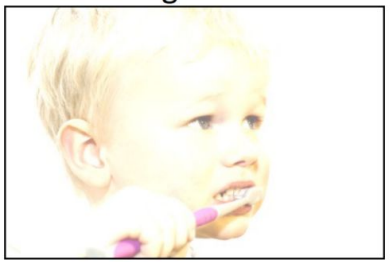
$\left(\frac{x}{255}\right)^{1/3} \times 255$

invert



$255 - x$

lighten



$x + 128$

raise contrast

$x \times 2$

non-linear raise contrast

$\left(\frac{x}{255}\right)^{2} \times 255$

Image Derivatives and Averages

- Derivative: Rate of change
  - Speed is a rate of change of a distance
  - Acceleration is a rate of change of speed

- Average (Mean)
  - Dividing the sum of N values by N

Derivative

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x) = f_x$$

$v = \frac{ds}{dt}$ speed $\qquad a = \frac{dv}{dt}$ acceleration

Examples

$$y = x^2 + x^4$$

$$\frac{dy}{dx} = 2x + 4x^3$$

$$y = \sin x + e^{-x}$$

$$\frac{dy}{dx} = \cos x + (-1)e^{-x}$$

Discrete Derivative

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x)$$

$$\frac{df}{dx} = \frac{f(x) - f(x-1)}{1} = f'(x)$$

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$

Discrete Derivative

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$ Backward difference

$$\frac{df}{dx} = f(x+1) - f(x) = f'(x)$$ Forward difference

$$\frac{df}{dx} = f(x+1) - f(x-1) = f'(x)$$ Central difference

Example

$f(x) =$	10	15	10	10	25	20	20	20
$f'(x) =$	0	5	-5	0	15	-5	0	0
$f''(x) =$	0	5	-10	5	15	20	5	0

Example

$$
\begin{array}{lllllllll}
f(x) = & 10 & 15 & 10 & 10 & 25 & 20 & 20 & 20 \\
f'(x) = & 0 & 5 & -5 & 0 & 15 & -5 & 0 & 0 \\
f''(x) = & 0 & 5 & -10 & 5 & 15 & 20 & 5 & 0
\end{array}
$$

Derivative Masks

Backward difference	[-1 1]
Forward difference	[1 -1]
Central difference	[-1 0 1]

Derivatives in 2 Dimensions

Given function $f(x, y)$

Gradient vector $$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

Gradient magnitude $$\left|\nabla f(x, y)\right| = \sqrt{f_x^2 + f_y^2}$$

Gradient direction $$\theta = \tan^{-1} \frac{f_y}{f_x}$$

Derivatives of Images

Derivative masks

$$f_x \Rightarrow \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad f_y \Rightarrow \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \qquad I_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Derivatives of Images

Derivative masks

$$f_x \Rightarrow \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad f_y \Rightarrow \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \qquad I_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

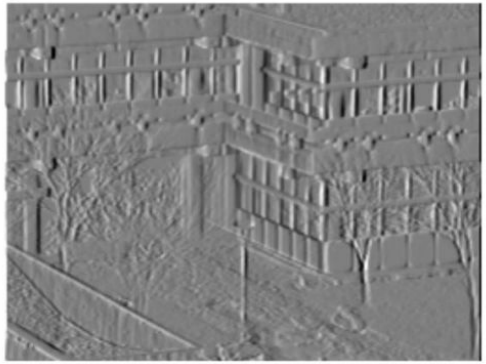
Derivatives of Images

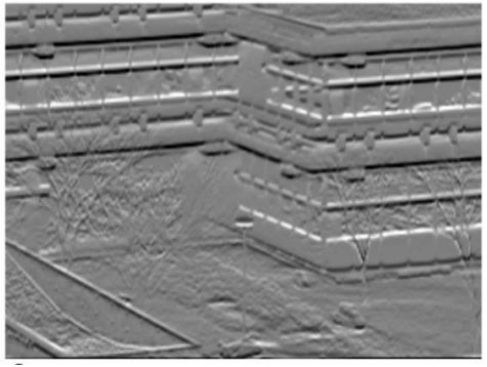
$$f_y \Rightarrow \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \qquad I_y = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Effect on Images

f_x





f_y

Detecting edges

How would you go about detecting edges in an image

✓ You take derivatives

How do you differentiate a discrete image (or any other discrete signal)?

✓ You use finite differences.