Digital Image Processing (CSE/ECE 478)

Lecture 6 : Spatial Filters (Part 2)

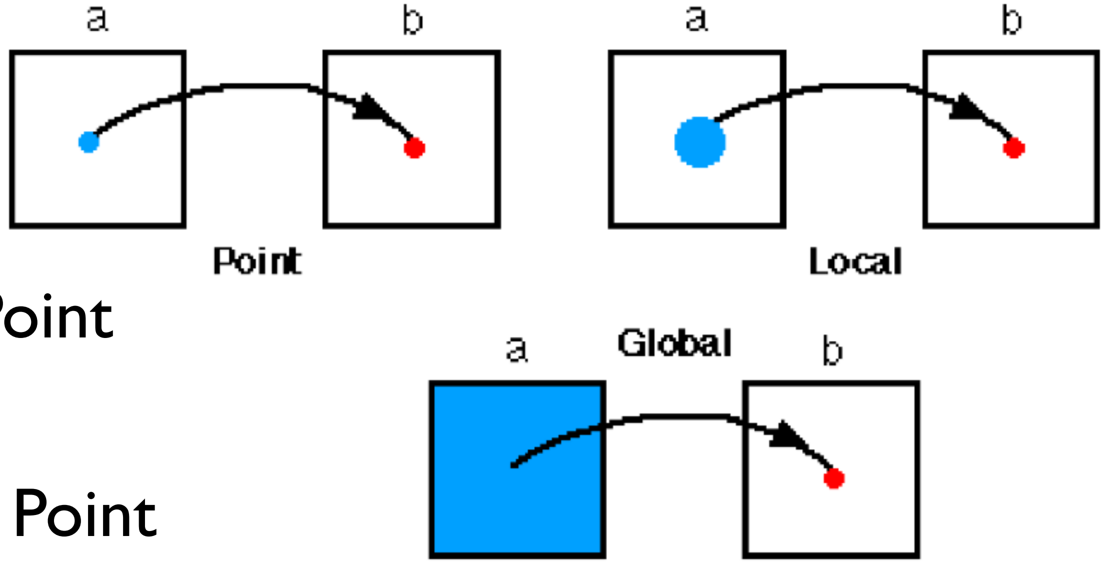
Ravi Kiran

Rajvi Shah

Recap …

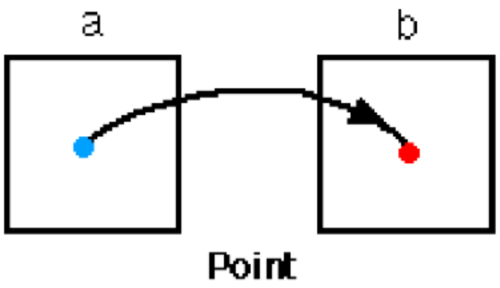
Spatial Domain Processing

- Manipulating Pixels Directly in Spatial Domain
- Point to Point
- Neighborhood to Point
- Global Attribute to Point

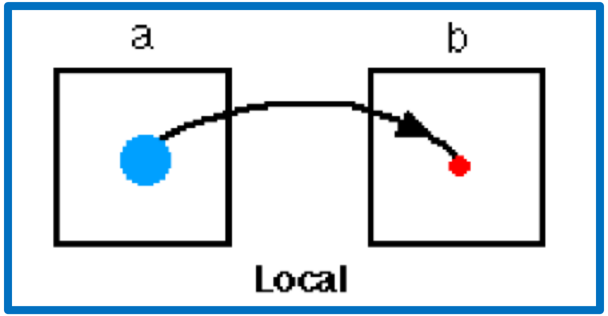


Spatial Domain Processing

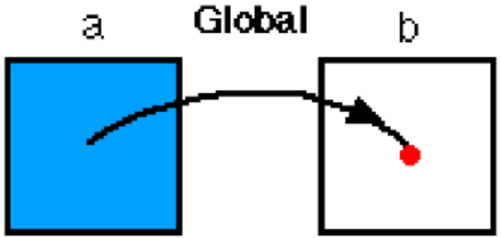
- Manipulating Pixels Directly in Spatial Domain
- Point to Point



- **Neighborhood to Point**

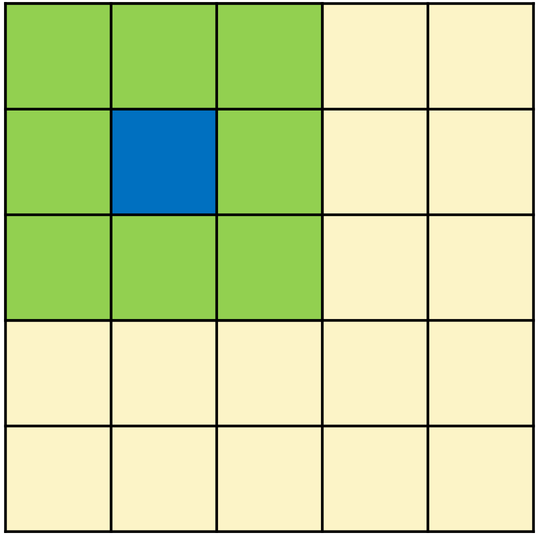


- Global Attribute to Point



Smoothing as Averaging

I



H → Mask

Kernel

Filter

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

$$I'(u,v) \leftarrow \frac{1}{9} \cdot \sum_{j=-1}^{1} \sum_{i=-1}^{1} I(u+i, v+j)$$

$$I'(u,v) \leftarrow \sum_{j=-1}^{1} \sum_{i=-1}^{1} I(u+i, v+j) \boxed{\bullet\ H(i,j)}$$

Sharpening Filter

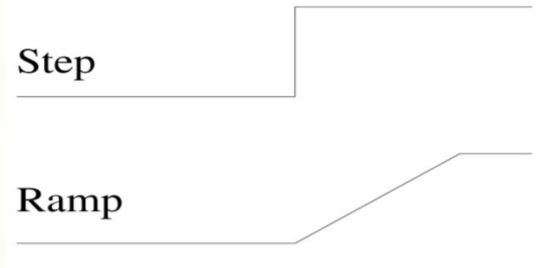
- Objective of sharpening is to highlight fine detail in an image or to enhance detail that has been blurred.
- Smoothing → Averaging → Summation → Integration
- Sharpening → Difference

1-D Derivatives

- First Derivative

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

  - Zero in flat segments
  - Nonzero at the onset of a step or ramp
  - Nonzero along ramps



- Second Derivative

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x).$$

  - Zero in flat areas;
  - Nonzero at the onset and end of a gray-level step or ramp;
  - Zero along ramps of constant slope

Image Derivatives

f(x-1,y)	f(x,y)	f(x+1,y)

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

f(x,y-1)
f(x,y)
f(x,y+1)

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

Laplacian Filter

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial x^2}$$

$$\nabla^2 f = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

0	1	0
1	-4	1
0	1	0

Laplacian Filter

0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

a	b
c	d

FIGURE 3.37
(a) Filter mask used to implement Eq. (3.6-6). (b) Mask used to implement an extension of this equation that includes the diagonal terms. (c) and (d) Two other implementations of the Laplacian found frequently in practice.

Implementation

$$g(x,y) = \begin{cases} f(x,y) - \nabla^2 f(x,y) & \textit{If the center coefficient is negative} \\ f(x,y) + \nabla^2 f(x,y) & \textit{If the center coefficient is positive} \end{cases}$$

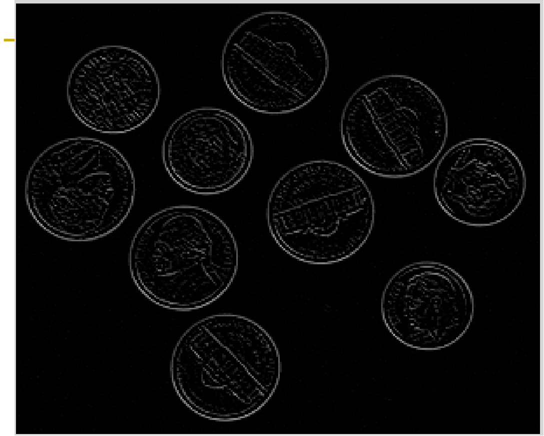
Where *f(x,y)* is the original image

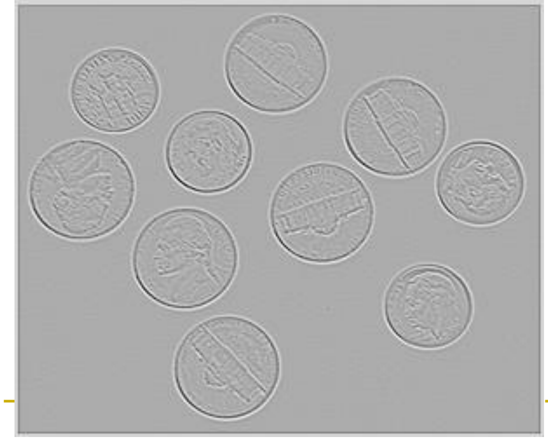
$\nabla^2 f(x,y)$ is Laplacian filtered image

g(x,y) is the sharpen image

sharpened = range_normalize(input + filtered)

Laplacian Filters





Unsharp Masking (and Highboost Filtering)

- **High boost filter**: amplify input image, then subtract a lowpass image

$$Highboost = A\ Original - Lowpass$$

$$= (A-1)\ Original + Original - Lowpass$$

$$= (A-1)\ Original + Highpass$$

(A-1) 

+ 

= 

Unsharp Masking / Highboost Filtering

A>=1
w = 9A-1

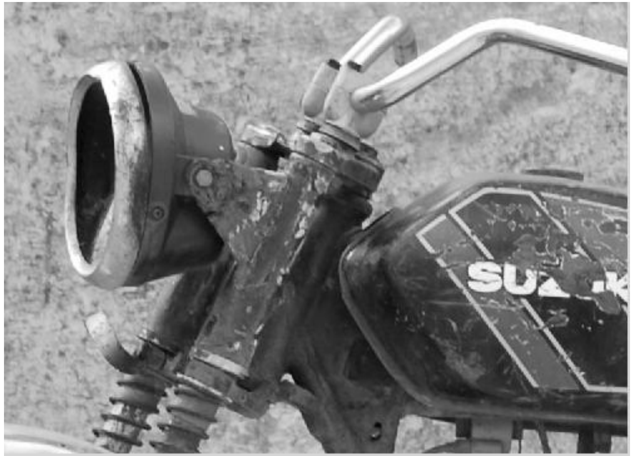
-1	-1	-1
-1	w	-1
-1	-1	-1

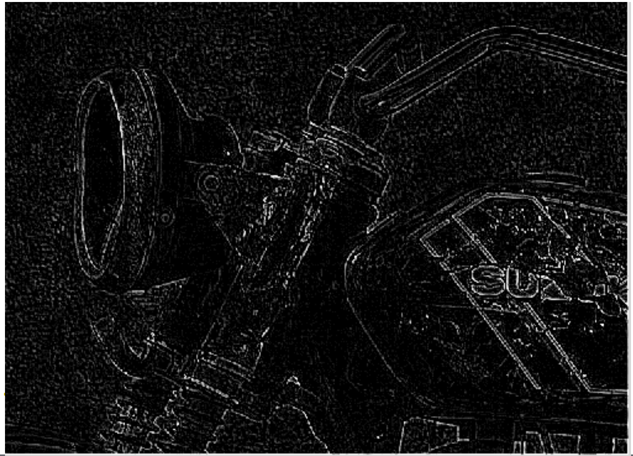
A=2
w = 17

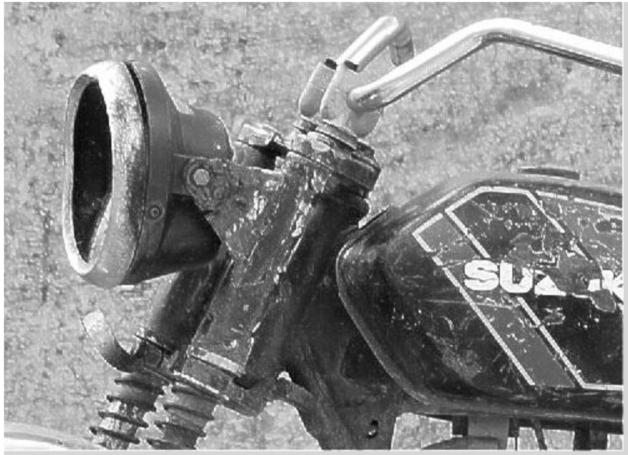
-1	-1	-1
-1	17	-1
-1	-1	-1

- If **A=1**, we get unsharp masking.
- If **A>1**, part of the original image is added back to the high pass filtered image.

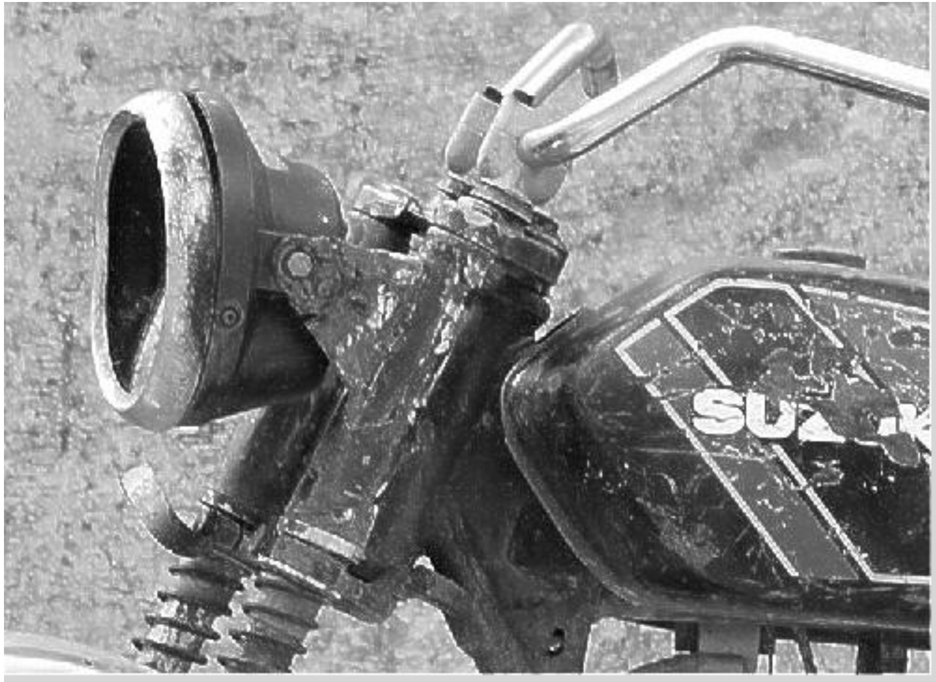
Unsharp Masking (and Highboost Filtering)







Unsharp Masking (and Highboost Filtering)



Also saw

- Non-linear Filters
  - min, max, median
  - Introduction to bilateral Filter

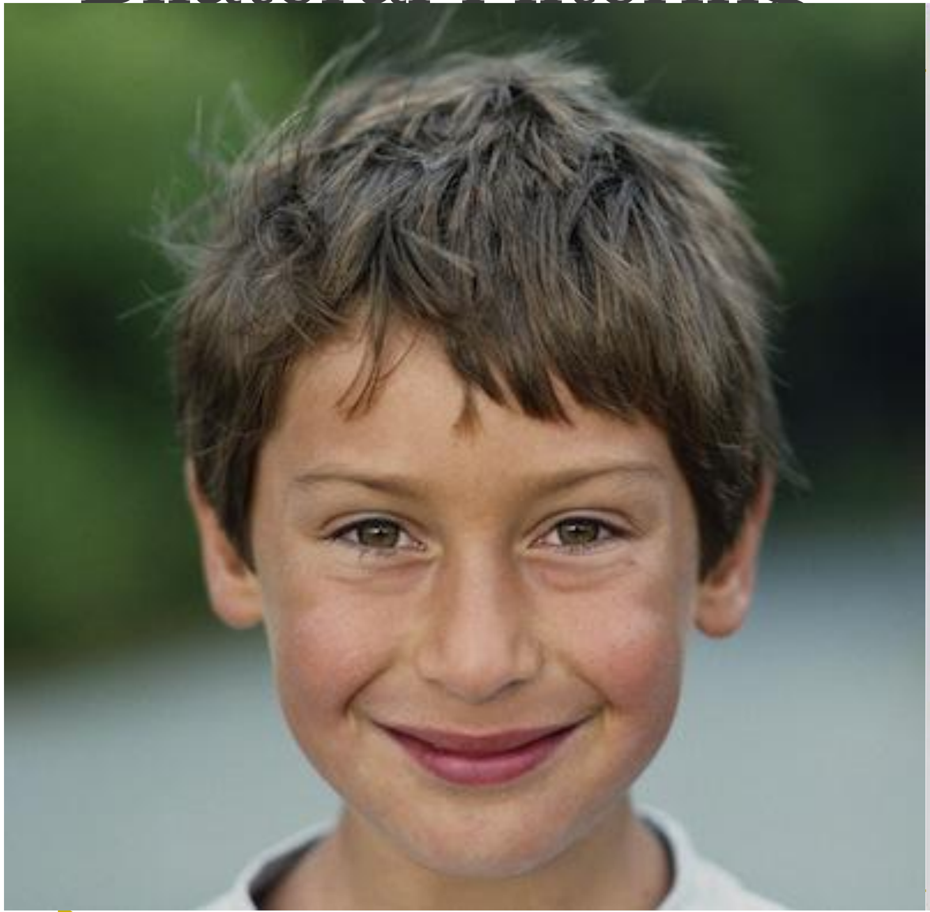
Continued …

Bilateral Filtering



Original image taken from cs.cityu.edu.hk

Bilateral Filtering



Original image from mfullywoodco.hol.es

Bilateral Filter

$$g(i,j) = \frac{\sum_{k,l} f(k,l)w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}. \tag{3.34}$$

Bilateral Filter

$$g(i,j) = \frac{\sum_{k,l} f(k,l) w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}. \tag{3.34}$$

The weighting coefficient $w(i,j,k,l)$ depends on the product of a *domain kernel* (Figure 3.19c),

and a data-dependent *range kernel* (Figure 3.19d),

Bilateral Filter

$$g(i,j) = \frac{\sum_{k,l} f(k,l) w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}. \tag{3.34}$$

The weighting coefficient $w(i,j,k,l)$ depends on the product of a *domain kernel* (Figure 3.19c),

$$d(i,j,k,l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2}\right), \tag{3.35}$$

and a data-dependent *range kernel* (Figure 3.19d),

$$r(i,j,k,l) = \exp\left(-\frac{\|f(i,j) - f(k,l)\|^2}{2\sigma_r^2}\right). \tag{3.36}$$

Bilateral Filter

$$g(i,j) = \frac{\sum_{k,l} f(k,l) w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}. \tag{3.34}$$

The weighting coefficient $w(i,j,k,l)$ depends on the product of a *domain kernel* (Figure 3.19c),

$$d(i,j,k,l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2}\right), \tag{3.35}$$

and a data-dependent *range kernel* (Figure 3.19d),

$$r(i,j,k,l) = \exp\left(-\frac{\|f(i,j) - f(k,l)\|^2}{2\sigma_r^2}\right). \tag{3.36}$$

When multiplied together, these yield the data-dependent *bilateral weight function*

$$w(i,j,k,l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2} - \frac{\|f(i,j) - f(k,l)\|^2}{2\sigma_r^2}\right). \tag{3.37}$$

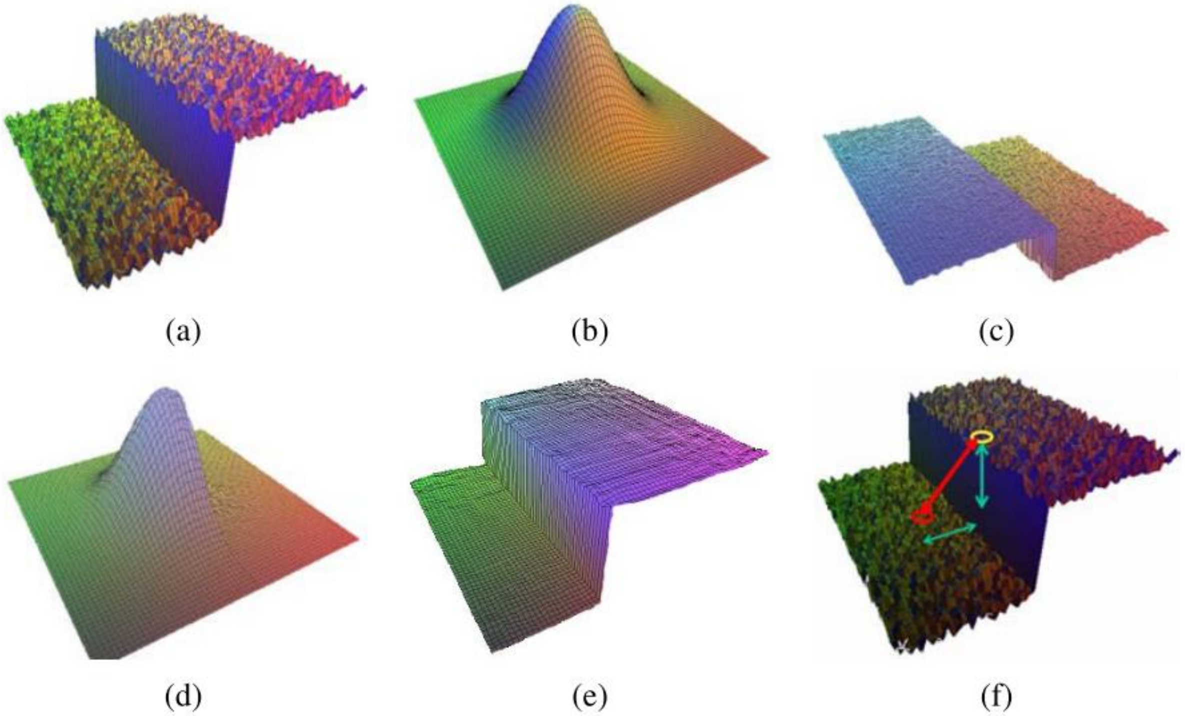
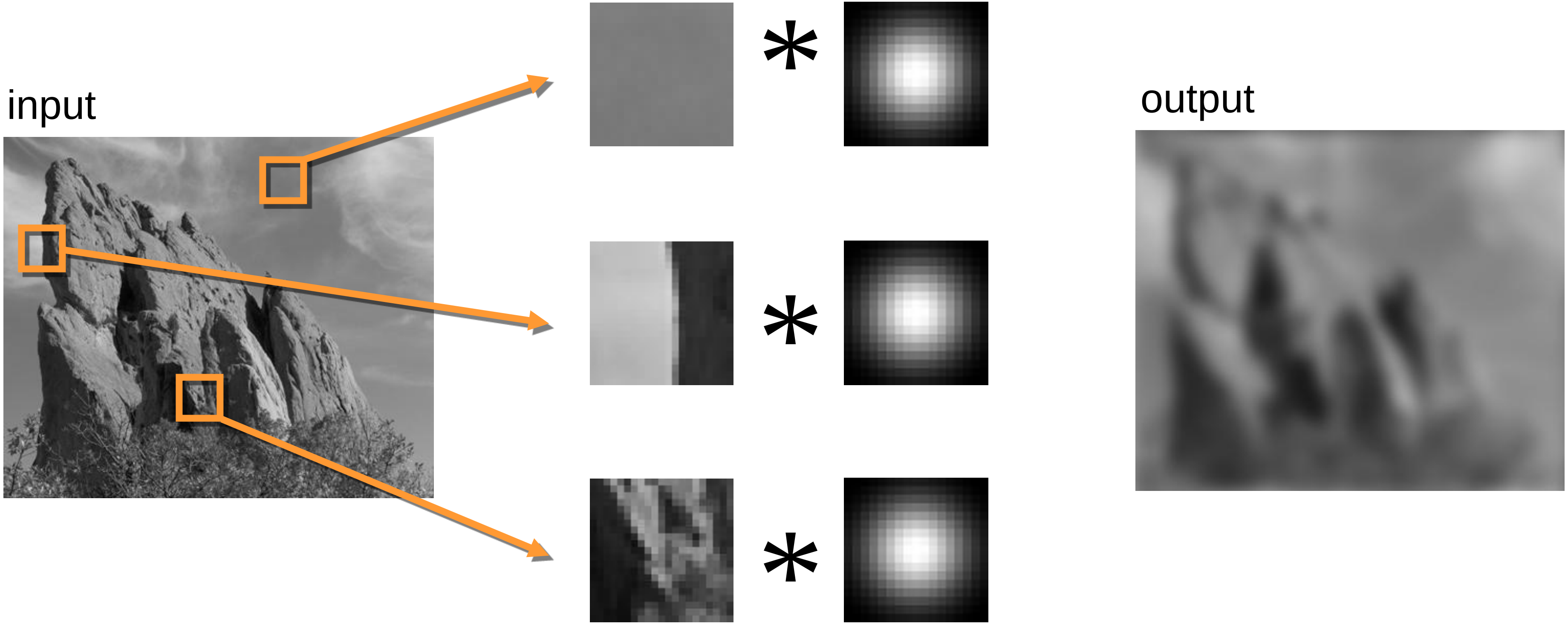


Figure 3.20 Bilateral filtering (Durand and Dorsey 2002) © 2002 ACM: (a) noisy step edge input; (b) domain filter (Gaussian); (c) range filter (similarity to center pixel value); (d) bilateral filter; (e) filtered step edge output; (f) 3D distance between pixels.

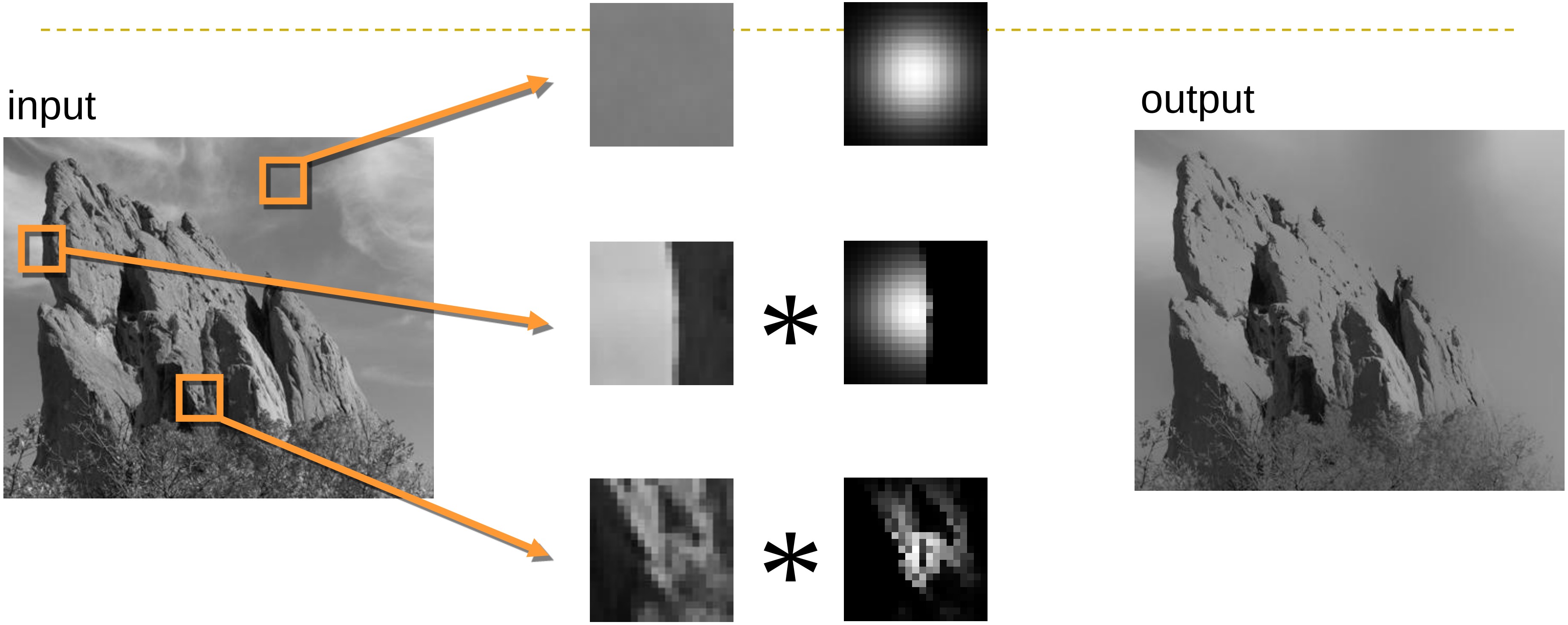
Usual Gaussian Filtering



Same Gaussian kernel everywhere.

Source: Sylvain Paris

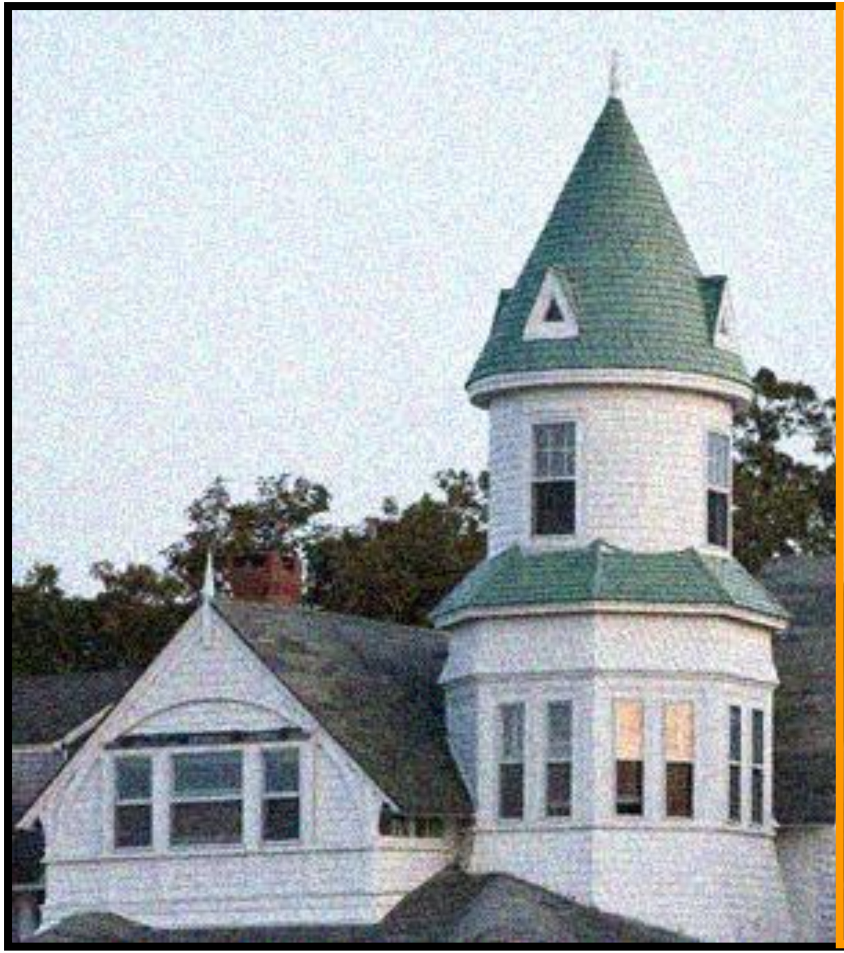
Bilateral Filtering



The kernel shape depends on the image content.

Source: Sylvain Paris

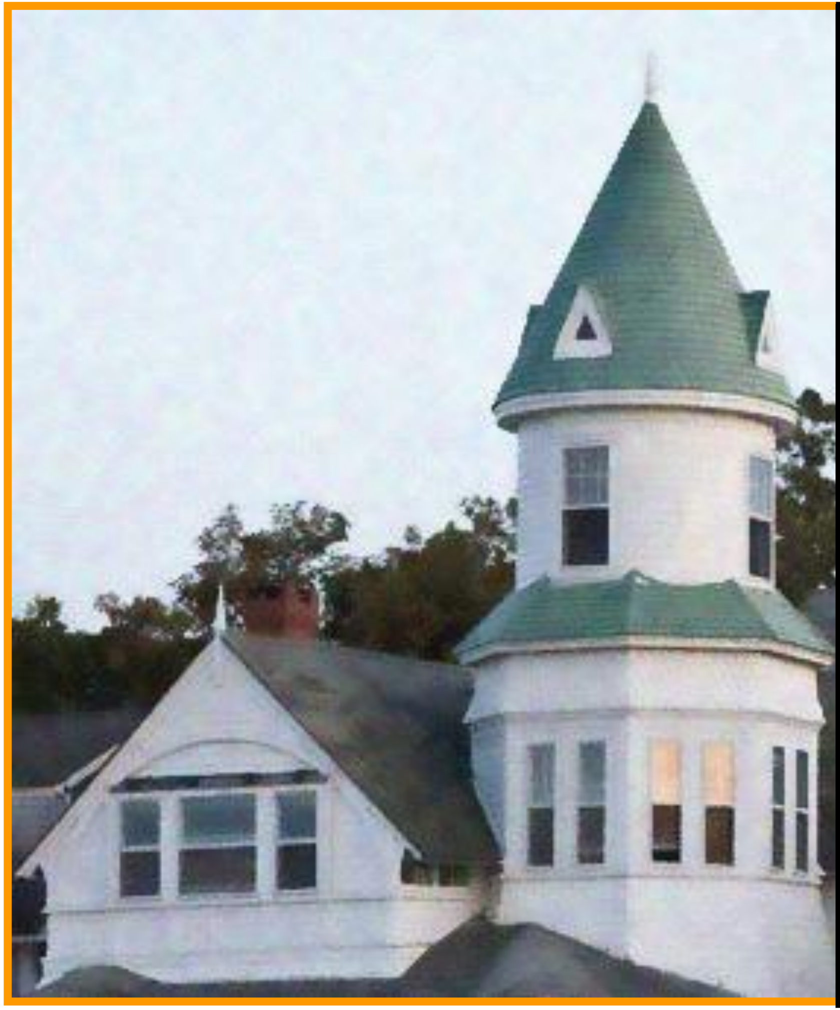
Noisy input



Bilateral filter 7x7 window

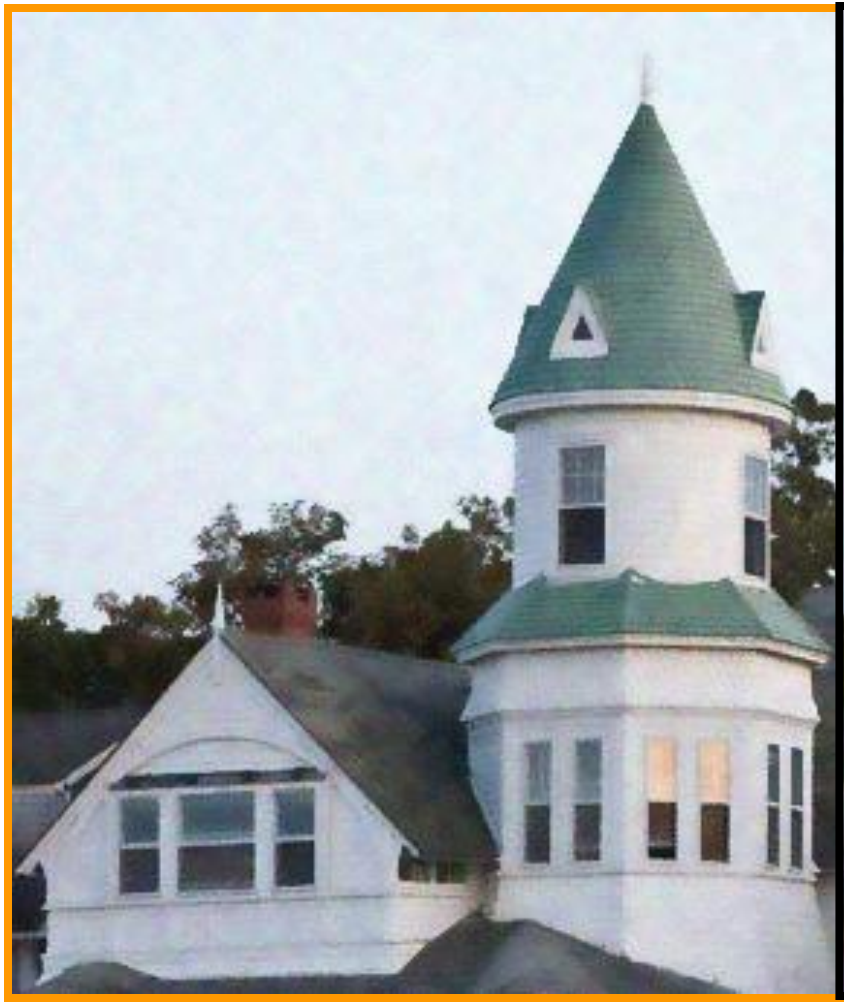
Source: Sylvain Paris

Bilateral filter



Median 3x3

Bilateral filter

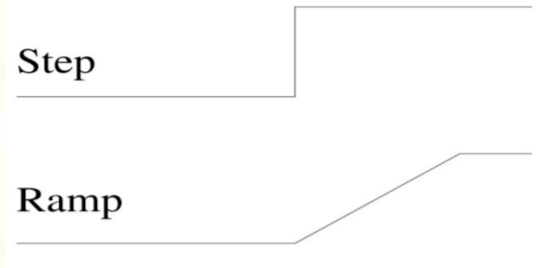


Median

More on Gradients

1-D Derivatives

- First Derivative
  $$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$
  - Zero in flat segments
  - Nonzero at the onset of a step or ramp
  - Nonzero along ramps



- Second Derivative
  $$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x).$$
  - Zero in flat areas;
  - Nonzero at the onset and end of a gray-level step or ramp;
  - Zero along ramps of constant slope

Image Derivatives

f(x-1,y)	*f(x,y)*	*f(x+1,y)*

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

f(x,y-1)
f(x,y)
f(x,y+1)

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

Laplacian Filter

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\nabla^2 f = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

0	1	0
1	-4	1
0	1	0

Laplacian Filter

(a)

0	1	0
1	−4	1
0	1	0

(b)

1	1	1
1	−8	1
1	1	1

(c)

0	−1	0
−1	4	−1
0	−1	0

(d)

−1	−1	−1
−1	8	−1
−1	−1	−1

a	b
c	d

FIGURE 3.37 (a) Filter mask used to implement Eq. (3.6-6). (b) Mask used to implement an extension of this equation that includes the diagonal terms. (c) and (d) Two other implementations of the Laplacian found frequently in practice.

Gradient Operators

- Based on First Derivative

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$magnitude(\nabla f) = \sqrt{G_x^2 + G_y^2} \approx |G_x| + |G_y|$$

$$orientation(\nabla f) = \tan^{-1}\left({G_y}/{G_x} \right)$$

Gradient Implementation

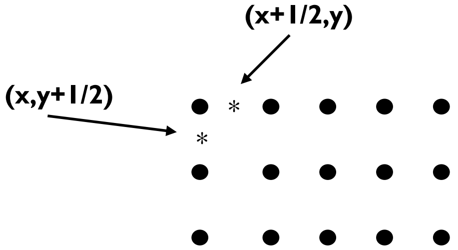
- We can implement $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ using masks:

-1	1

1
-1

good approximation at (x+1/2,y)

good approximation at (x,y+1/2)



- A different approximation of the gradient:

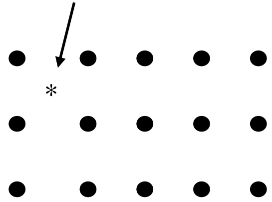
$$\frac{\partial f}{\partial x}(x, y) = f(x, y) - f(x+1, y+1)$$

$$\frac{\partial f}{\partial y}(x, y) = f(x+1, y) - f(x, y+1),$$

−1	0
0	1

0	−1
1	0

good approximation (x+1/2,y+1/2)



Gradient Operators

- Sobel Operator

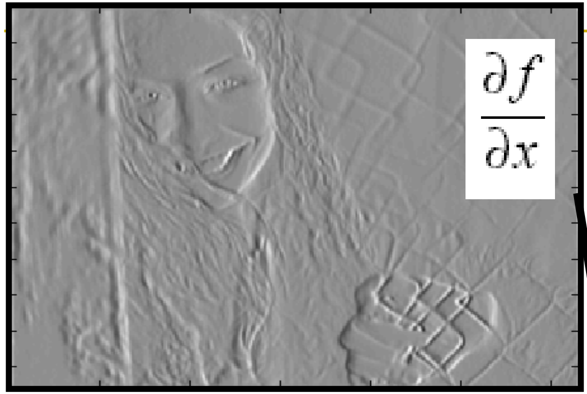
$$\frac{\partial f}{\partial x}$$

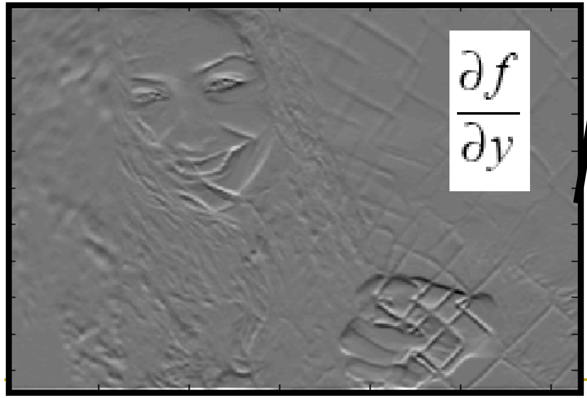
-1	0	1
-2	0	2
-1	0	1

$$\frac{\partial f}{\partial y}$$

-1	-2	-1
0	0	0
1	2	1

Example

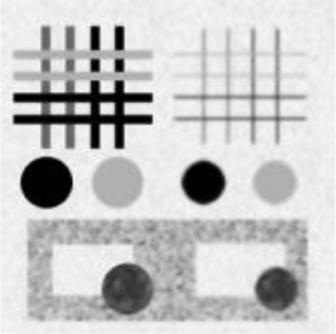




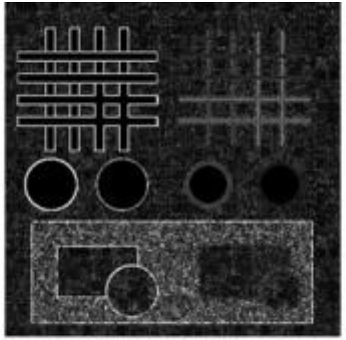
Gradient Magnitude

$$\sqrt{\frac{\partial f}{\partial x}^2 + \frac{\partial f}{\partial y}^2}$$

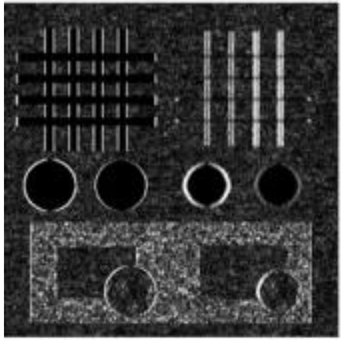
Gradient vs. Laplacian



Original

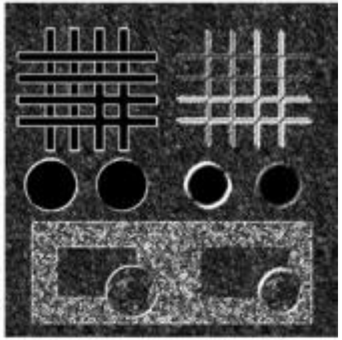


Laplacian



Sobel X

Sobel Y

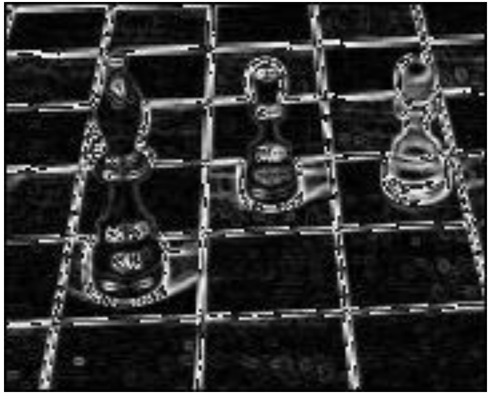


Sobel X+Y

Gradient vs. Laplacian

Laplacian

Sobel

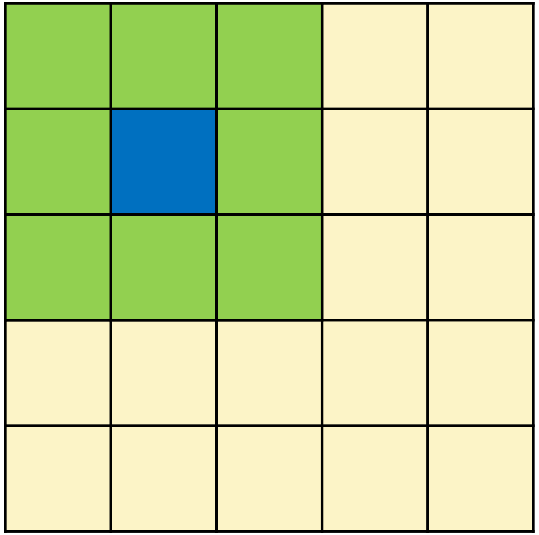


Other Important Filters

- Laplacian of Gaussian
  - Noise Suppression
- Difference of Gaussian
  - Band-pass

Remember this?

I



H → Mask

Kernel

Filter

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

$$I'(u,v) \leftarrow \frac{1}{9} \cdot \sum_{j=-1}^{1} \sum_{i=-1}^{1} I(u+i, v+j)$$

$$I'(u,v) \leftarrow \sum_{j=-1}^{1} \sum_{i=-1}^{1} I(u+i, v+j)$$

- *H(i,j)*

Linear Filtering : Generalization

$$I'(u,v) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} I(u+i, v+j) \cdot H(i,j)$$

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} f(x+s, y+t) \cdot w(s,t)$$

Linear Filtering : Generalization

$$g(x,y)=\sum_{s=-a}^{a}\sum_{t=-b}^{b} f(x+s,y+t)\cdot w(s,t)$$

1-D Case:

$$g(x)=\sum_{s=-a}^{a} f(x+s)\cdot w(s)$$

Linear Filtering – 1D

$$g(x) = \sum_{s=-a}^{a} f(x+s) \cdot w(s)$$

Origin f w

(a) 0 0 0 1 0 0 0 0   1 2 3 2 8

(b) 0 0 0 1 0 0 0 0
1 2 3 2 8
Starting position alignment

Zero padding

(c) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8

(d) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
Position after one shift

(e) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
Position after four shifts

(f) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
Final position

Full result

(g) 0 0 0 8 2 3 2 1 0 0 0 0

Cropped result

(h) 0 8 2 3 2 1 0 0

Linear Filtering – 1D

$$g(x)=\sum_{s=-a}^{a} f(x-s)\cdot w(s)$$

Origin $\quad f \quad$ w rotated 180°

0 0 0 1 0 0 0 0 $\quad$ 8 2 3 2 1 $\quad$ (i)

$\quad$ 0 0 0 1 0 0 0 0 $\quad$ (j)

8 2 3 2 1

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 (k)

8 2 3 2 1

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 (l)

$\quad$ 8 2 3 2 1

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 (m)

$\quad$ 8 2 3 2 1

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 (n)

$\quad$ 8 2 3 2 1

Full result

0 0 0 1 2 3 2 8 0 0 0 0 $\quad$ (o)

Cropped result

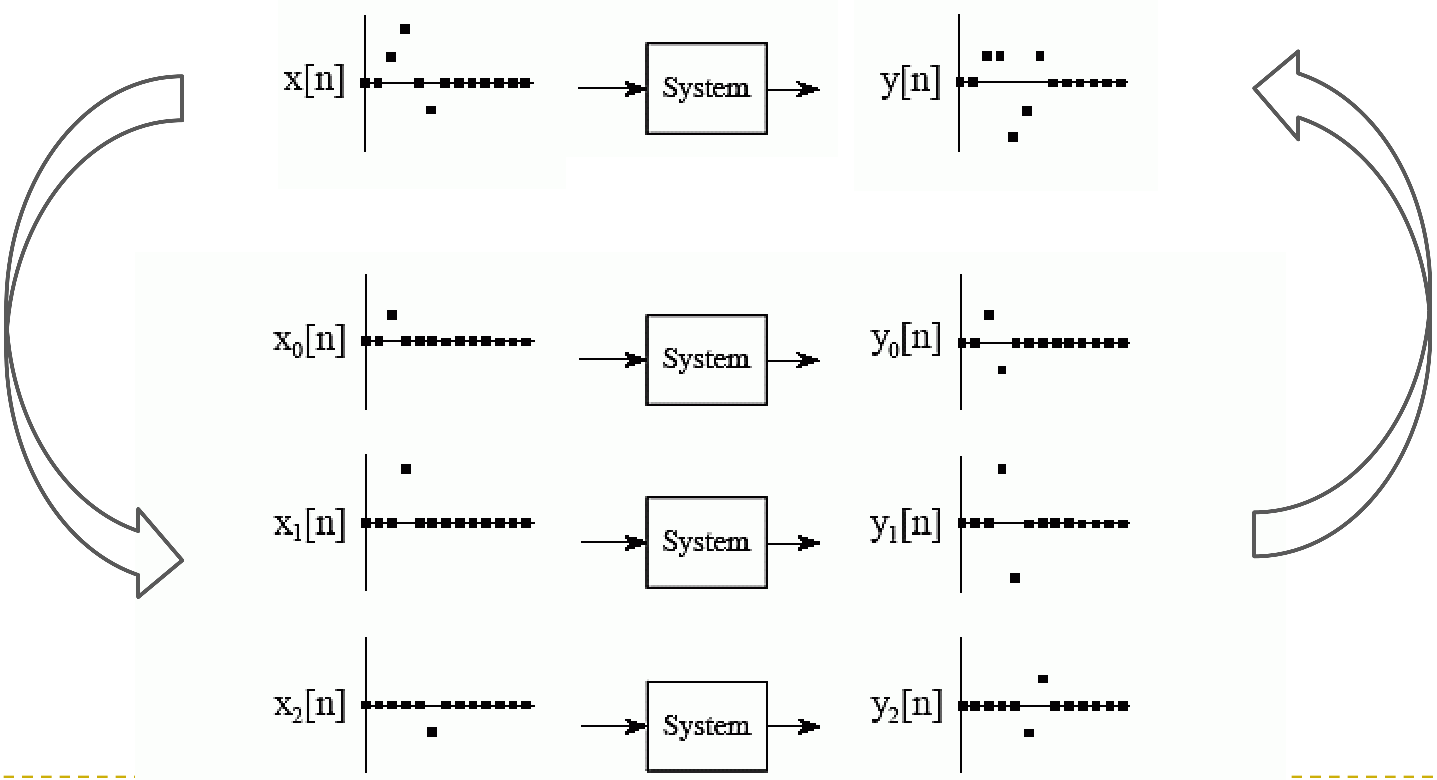
0 1 2 3 2 8 0 0 $\quad$ (p)

Linear Filtering – 1D

$$g(x) = \sum_{s=-a}^{a} f(x-s) \cdot w(s)$$

- f is impulse → g is same as w
- Impulse Response

Signal as weighted sum of shifted impulses



$$f(x)=\sum_{s=-a}^{a}\delta(x-s)\cdot f(s)$$

$$T[f(x)]=T[\sum_{s=-a}^{a}\delta(x-s)\cdot f(s)]$$

$$g(x)=\sum_{s=-a}^{a}T[\delta(x-s)]\cdot T[f(s)]$$

$$g(x)=\sum_{s=-a}^{a}w(x-s)\cdot f(s)\ =\sum_{s=-a}^{a}f(x-s)\cdot w(s)$$

Linear Filtering as convolution

$$g(x) = f * w = \sum_{s=-a}^{a} f(x-s) \cdot w(s)$$

$$g(x,y) = f * w = \sum_{s=-a}^{a} \sum_{t=-b}^{b} f(x+s, y+t) \cdot w(s,t)$$

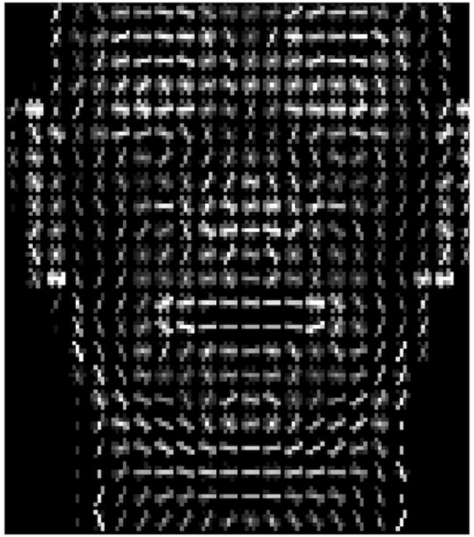
Time for a (de)tour!

Filters for Image Analysis

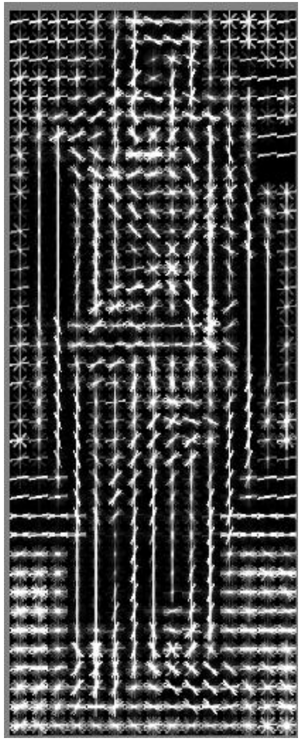
- We learned filters for image enhancement
- But also very important for image analysis tasks
- Fundamental tool for Feature based Image Representation
  - Necessary Computer Vision and Machine Learning

Histogram of Colors

Visualizing Gradients

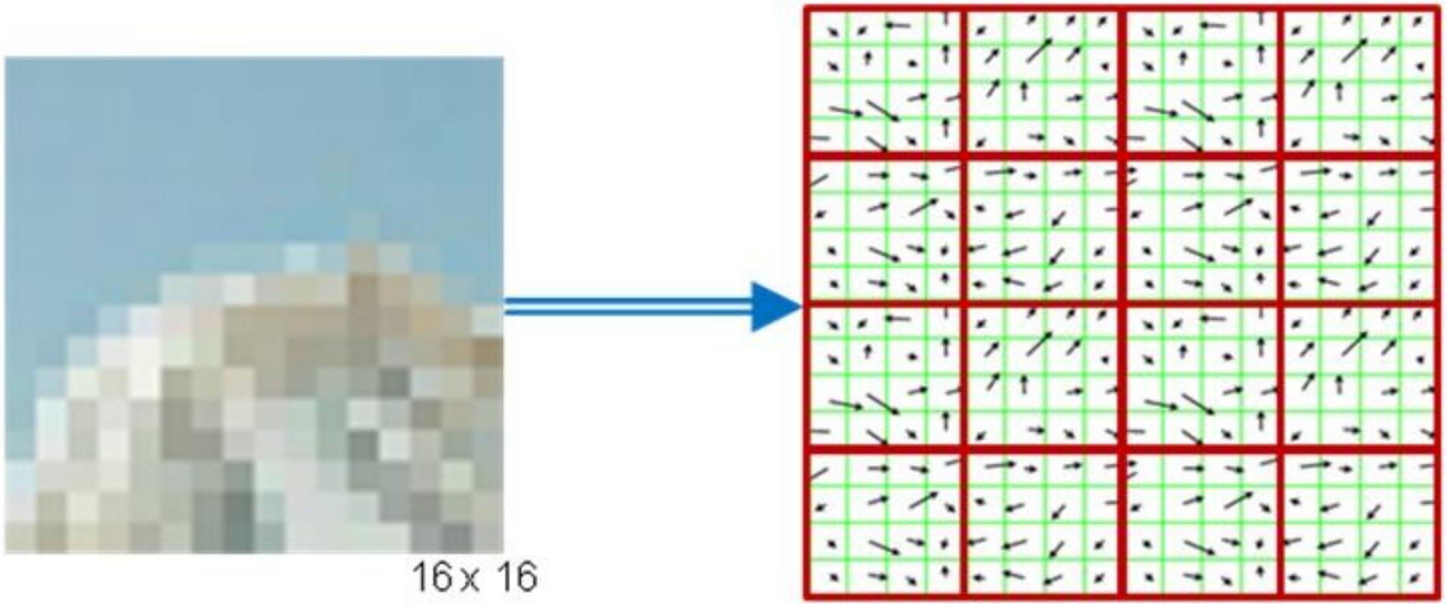


Face

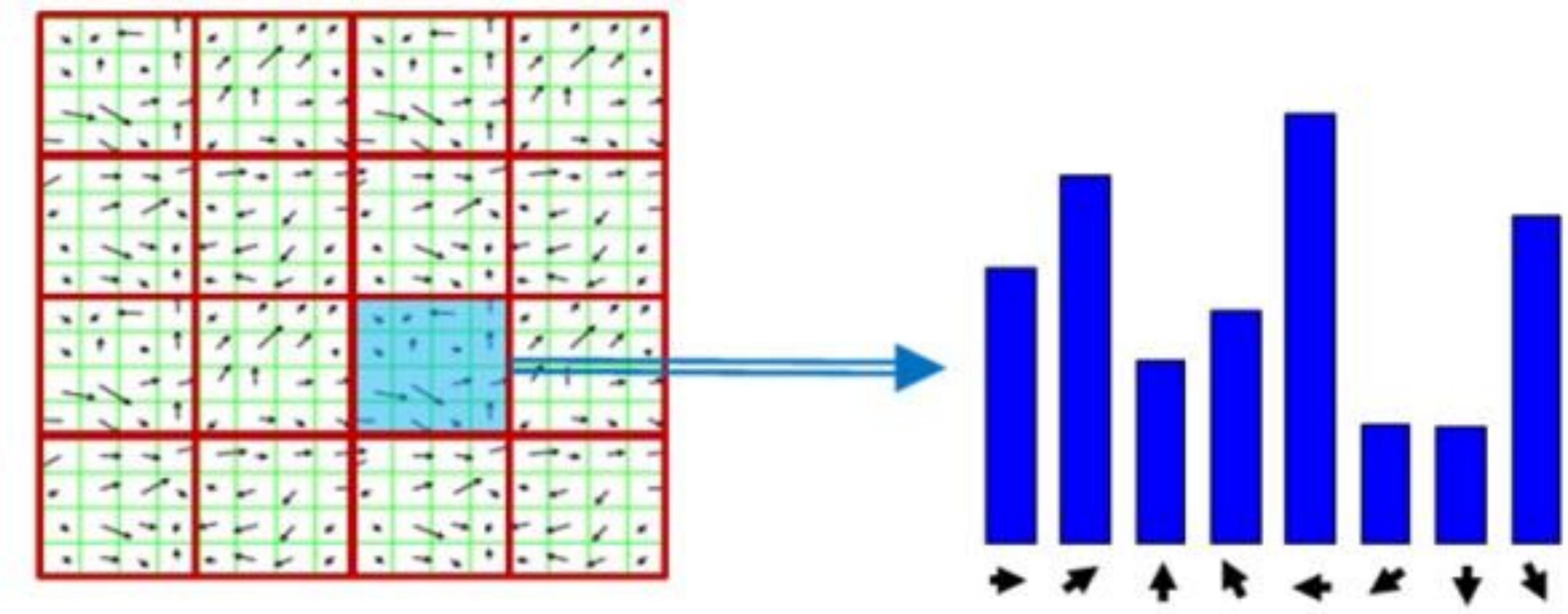


Person

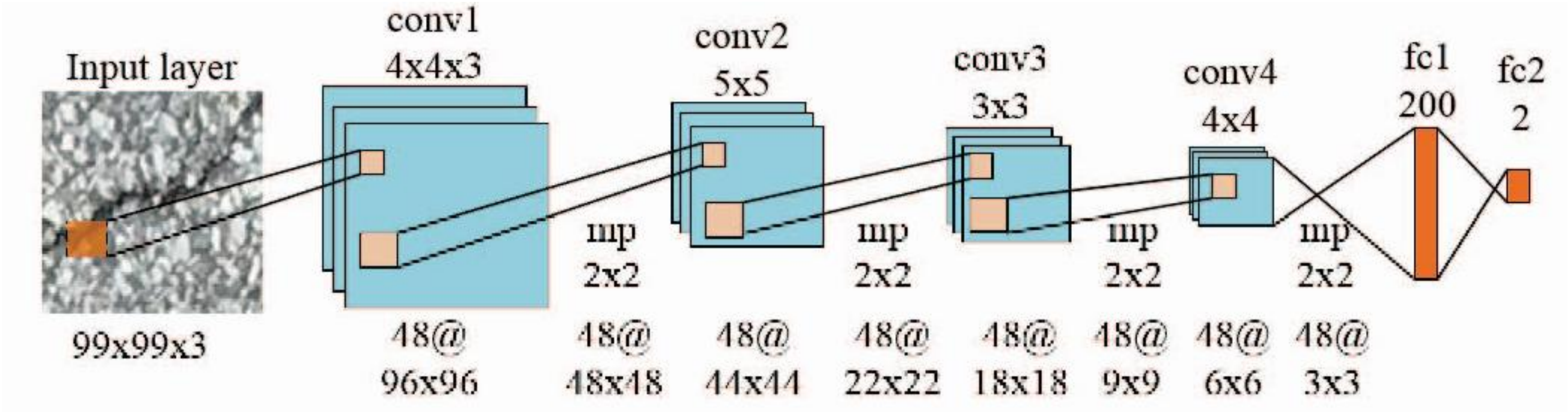
Gradient Orientations



Histogram of Gradient Orientations



Learning Filters



Central to Convolutional Neural Network

Summary

- Linear Filtering – moving a weight mask over the input image, multiplying weights with intensity values, and summing them up to produce output image
- Linear Filtering as Convolution
  - Part of larger LTI systems
- Nonlinear Filtering – min, max, median, bilateral (mask is data-dependent)

References

- GW Chapter – 3.4
- Szeliski Book : Computer Vision and Applications
  - Bilateral Filtering
- Prof. Bebis Slides
- Slides from Vineet Gandhi