MAEG 5720:
Computer Vision in Practice

Lecture 3:

Image Filtering

Dr. Terry Chang

2021-2022

Semester 1

香港中文大學
The Chinese University of Hong Kong

Department of Mechanical and Automation Engineering
機械與自動化工程學系

Last Lecture

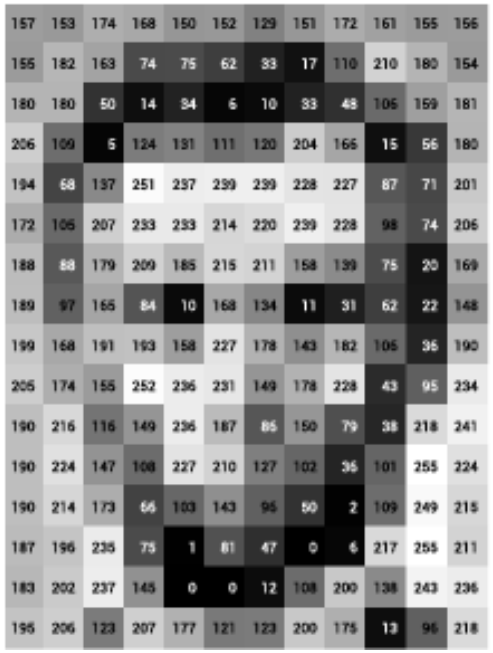
- What is a camera?
- Image is a 2D rectilinear array of pixels
- Pinhole Camera
- Size of aperture: large vs small
- Lens:
  - depth of field
  - Field of view
- Affine transformation
- Homogenous Coordinates
- Homography

Today’s Agenda

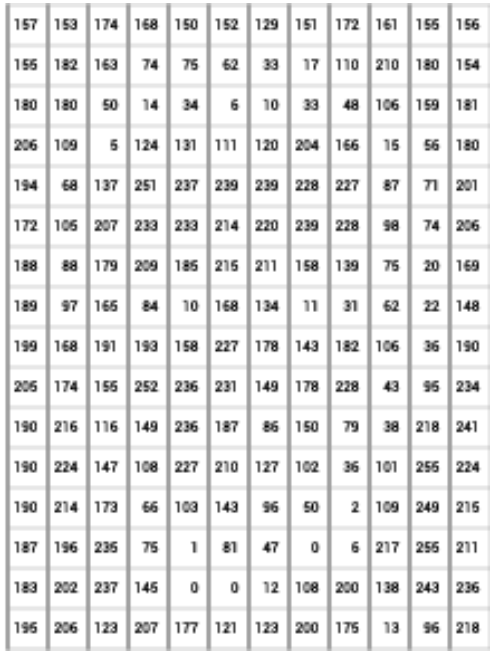
- Image as functions
- Filters in Spatial domain
- Filters in Frequency Domain
- Filters for Template Matching
- Image Pyramid

Images as functions

- An image is a *2D array* of pixels
- For ***gray*** image, pixel value is ***intensity [0,255]***



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

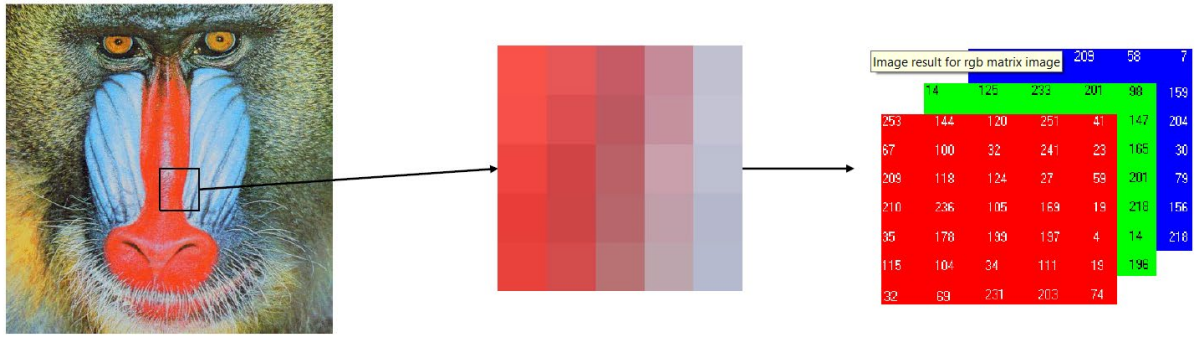


157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Source: http://techundred.com/how-snapchat-filter-work/

Images as functions

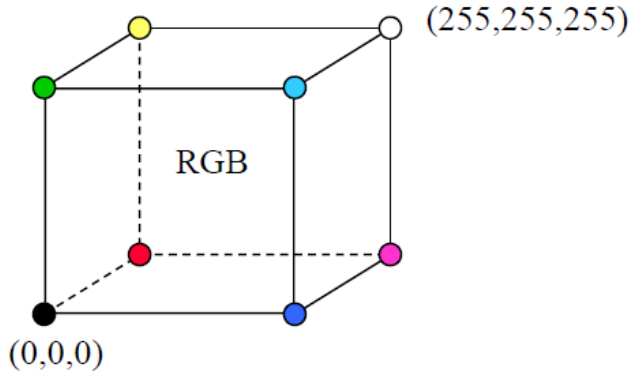
- An image is a 2D array of pixels
- For Color Image
  - RGB: [R,G,B]
  - HSV: [H, S, V]
    - Hue,
    - Saturation,
    - Value



Source: R. Dockter

RGB-Color

- Usually we specify the levels of R,G, and B in the range [0, 255]
- 8 bit, per pixel, per channel



$$(256)3=16{,}777{,}216 Colors$$

Images as functions

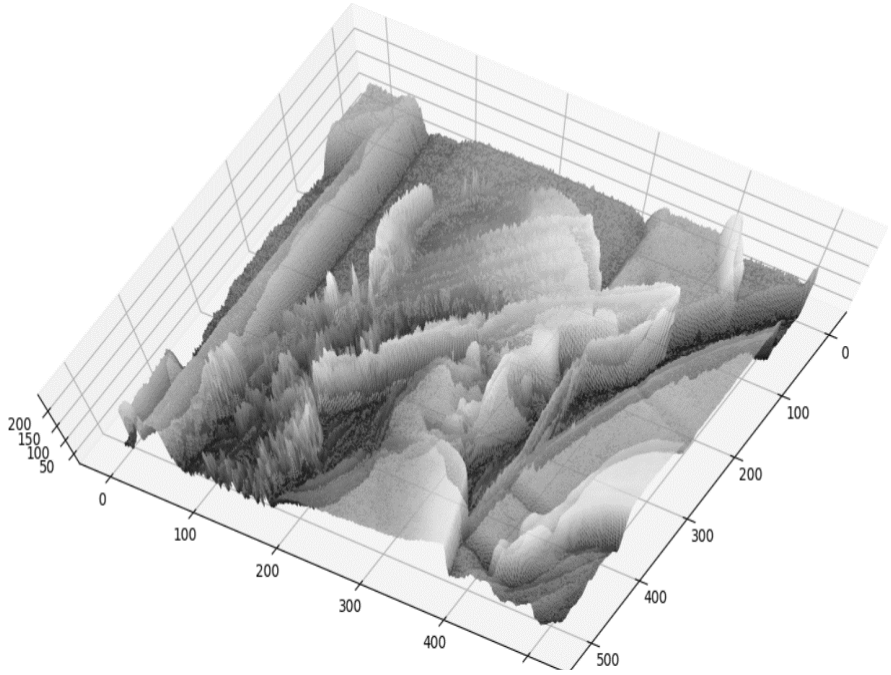
- An Image is a function f
  - $f: \mathbb{R}^2 \rightarrow \mathbb{R}$
  - $f(x, y)$ gives the intensity at position (x, y)
  - We expect the image only defined over a finite rectangular range
  - $f: [a, b] \times [c, d] \rightarrow [0,255]$
- A color image is just three functions represented as a vector-valued function
  - $f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$

Digital Image

- In computer vision, we operate on digital (discrete) Images:
  - Sample the 2D space on regular rid
  - Quantize to nearest integer

i →
j ↓

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218



Images as discrete functions

- $f[i,j] = \begin{bmatrix} \ddots & & \vdots & & \\ & f[-1,-1] & f[0,-1] & f[1,-1] & \\ \cdots & f[-1,0] & f[0,0] & f[1,0] & \cdots \\ & f[-1,1] & f[0,1] & f[1,1] & \\ & & \vdots & & \ddots \end{bmatrix}$

Today's Agenda

- Image as functions
- Filters in Spatial domain
  - Box filter
  - Edge/Derivative filter
  - Gaussian filter
- Filters in Frequency Domain
- Template Matching
- Image Pyramid

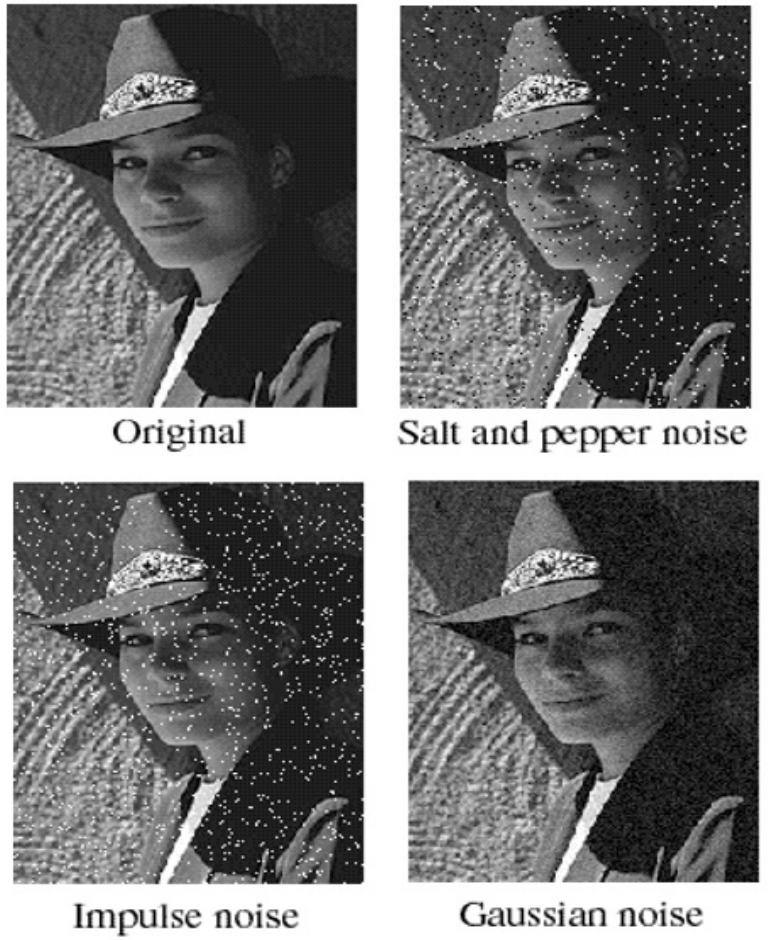
Image Filtering

- Why we need image filters?
  - Real Images are ***noisy***
- Filtering method
  - Form a new image by ***combination*** of ***original*** pixel values
- Goals
  - Enhance the image properties
    - ***De-noising***, ***Sharpening***
  - Extract useful information
    - ***Edges***, ***corners***, ***template matching***

$$f[i,j] = \begin{bmatrix} \ddots & & \vdots & & \\ & f[-1,-1] & f[0,-1] & f[1,-1] & \\ \cdots & f[-1,0] & f[0,0] & f[1,0] & \cdots \\ & f[-1,1] & f[0,1] & f[1,1] & \\ & & \vdots & & \ddots \end{bmatrix}$$

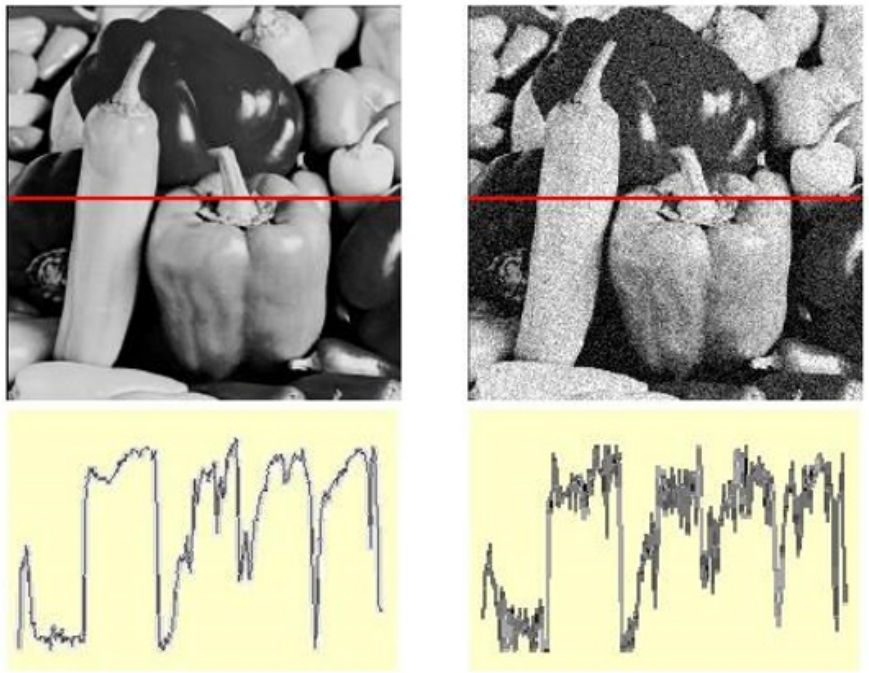
Common types of noise

- ***Salt and pepper noise***
  - Random occurrences of *black and white* pixels
- ***Impulse noise***:
  - Random occurrences of *white* pixel
- ***Gaussian noise***
  - Variations in intensity drawn from a *Gaussian normal distribution*



Source: S. Seitz

Images are noisy



$$f(x,y) = \hat{f}(x,y) + \eta(x,y)$$

image Ideal image noise

```
clear all; close all; clc;

%Read the image
input_image = imread('lena.jpg');
noisy_image
=imnoise(input_image,'gaussian');

figure, imshow(input_image); title('Input
Image');
figure, imshow(noisy_image); title('Noisy
Image');
```

Fig: M. Hebert

How to reduce the noise?

- Applying filters!

- Filtering method
  - Form a new image by combination of original pixel values

- Assumptions:
  - We expect pixels likes their neighbors
  - Noise independent of pixel locations

Moving Average Filter

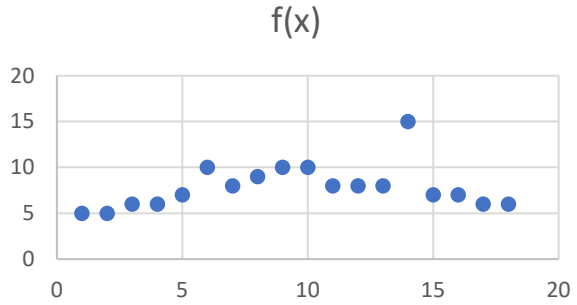
- In 1-D, For any *k*, say *k=1*

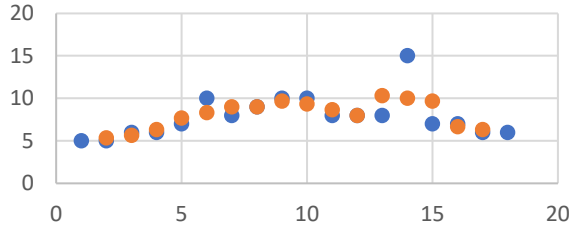
$$g(i) = \sum_k h(k) f(i-k)$$

- $g(i)$ is the moving average of *f*
- Effect: smoothing!

f(x)
5
5
6
6
7
10
8
9
10
10
8
8
8
15
7
7
6
6

g(x)
5.3
5.7
6.3
7.7
8.3
9.0
9.0
9.7
9.3
8.7
8.0
10.3
10.0
9.7
6.7
6.3





In 2D- Box filter

$h[\cdot,\cdot]$

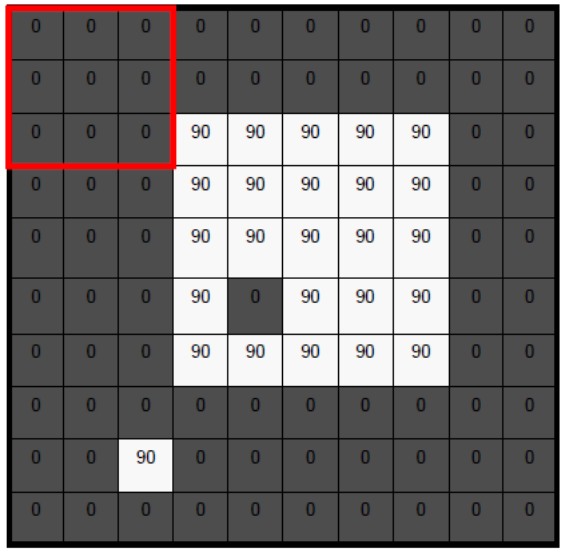
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

```
%Read the filter size
pad_size=input('Enter Averaging Mask size: ');

%create the box filter
average_filter=ones(pad_size,pad_size);
average_filter(:)/(pad_size*pad_size);
```

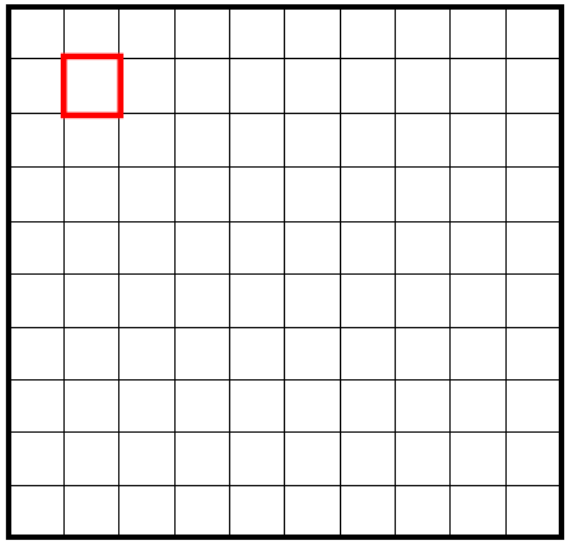
Slide credit: David Lowe (UBC)

Image Filtering

$f[.,.]$



$h[.,.]$



$h[\cdot,\cdot]$

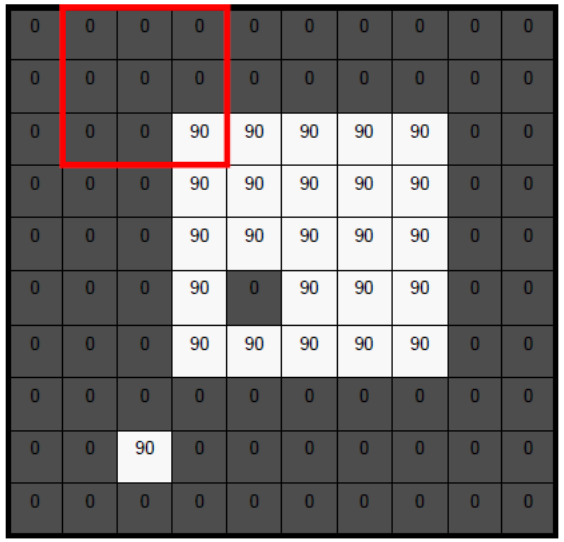
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

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

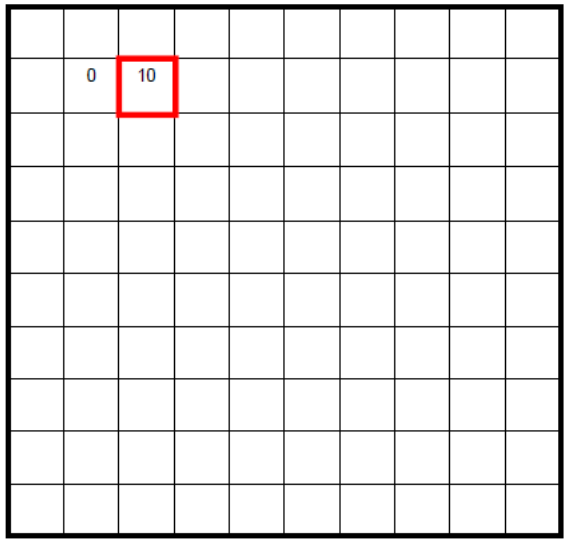
Credit: S. Seitz

Image Filtering

$f[.,.]$



$h[.,.]$



$h[\cdot,\cdot]$

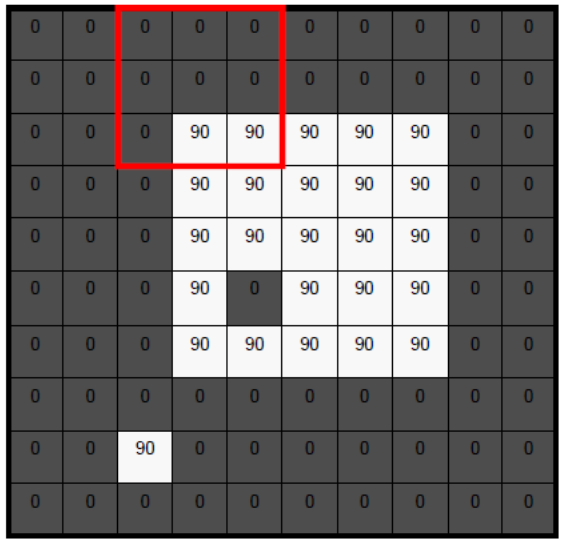
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

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

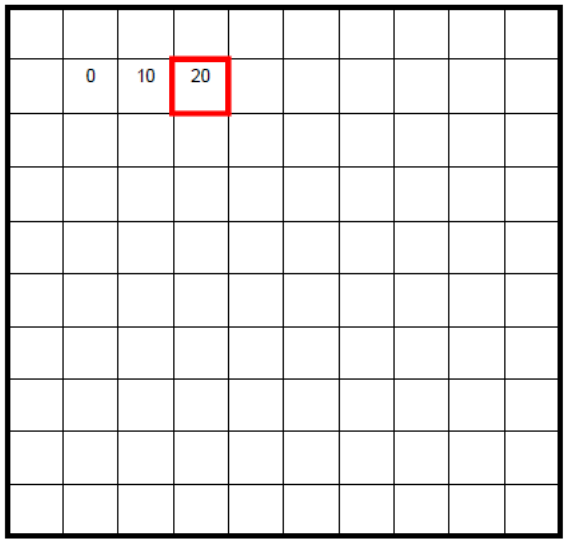
Credit: S. Seitz

Image Filtering

$f[.,.]$



$h[.,.]$



$h[\cdot,\cdot]$

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

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

Credit: S. Seitz

Image Filtering

$f[.,.]$

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					

$h[\cdot,\cdot]$

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

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

Credit: S. Seitz

Image Filtering

$f[.,.]$

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				

$h[\cdot,\cdot]$

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

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

Credit: S. Seitz

Image Filtering

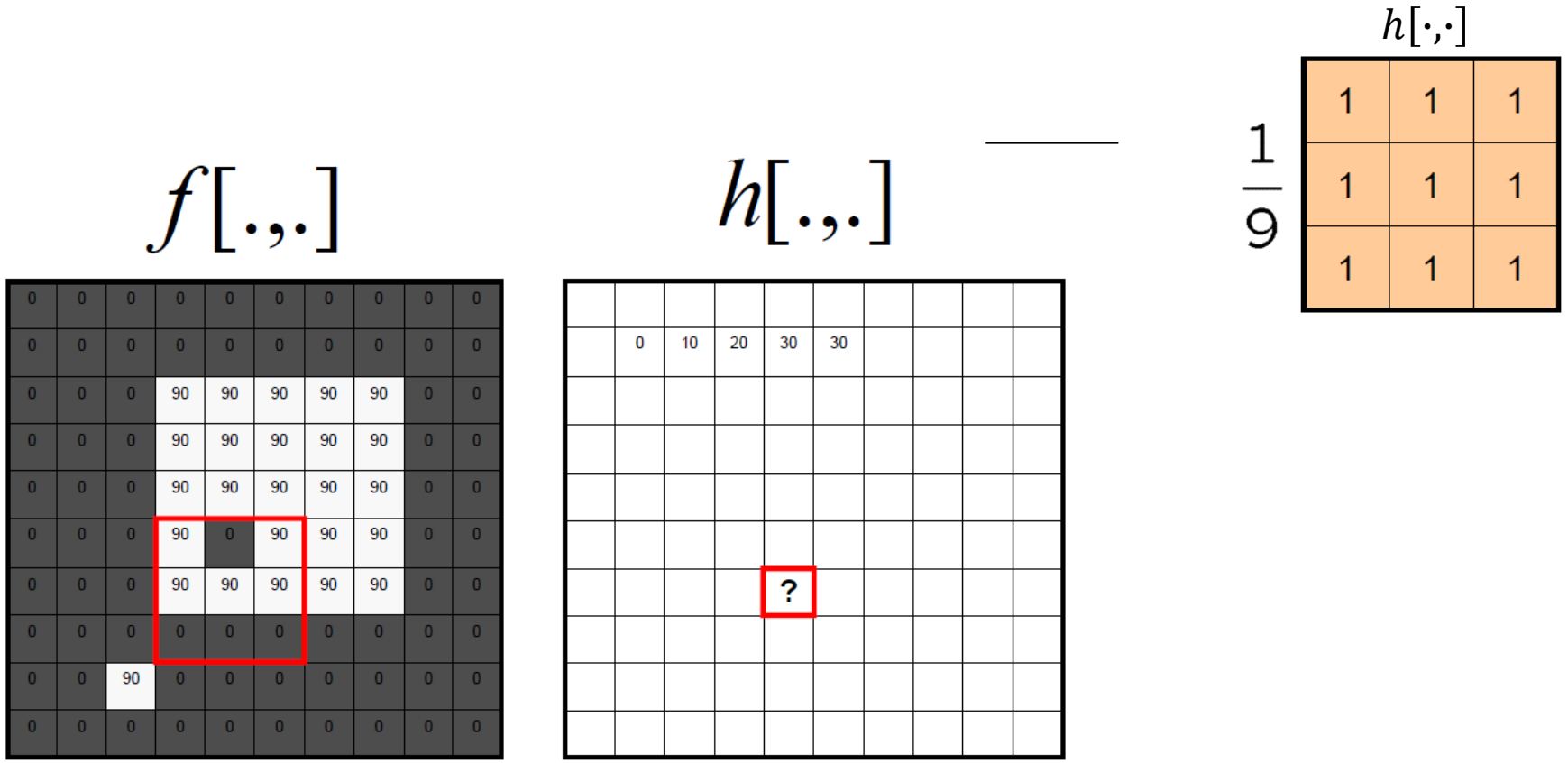
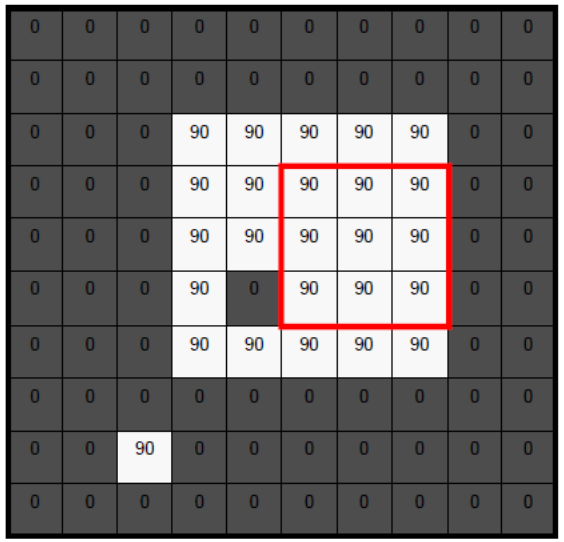
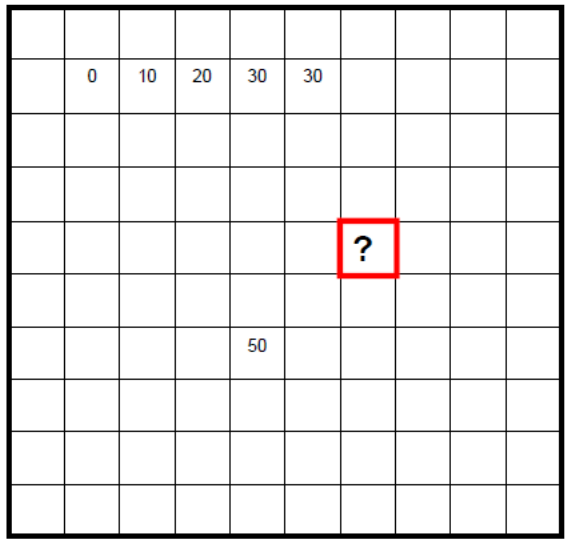


Image Filtering

$f[.,.]$



$h[.,.]$



$h[\cdot,\cdot]$

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

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

Credit: S. Seitz

Image Filtering

$f[.,.]$

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	

$h[\cdot,\cdot]$

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

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

Credit: S. Seitz

Box Filter

- What does it do?
  - Replaces each pixel with an average of its neighborhood
- Effect?
  - Smoothing

$h[\cdot,\cdot]$

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

Smoothing with box filter

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

MATLAB Code

```
%convert uint8 to double
padded_image =double(padded_image);
for r=1+pad_size:Row-pad_size
    for c=1+pad_size:Col-pad_size
        sum(1:3) = 0;
        original_value = [padded_image(r,c,1), padded_image(r,c,2), padded_image(r,c,3)];

        for j=-pad_size:pad_size                    %filter index starts from 1
            for i =-pad_size:pad_size
                sum(1) = sum(1) + average_filter(pad_size+1+j,pad_size+1+i)*padded_image(r+j,c+i,1);
                sum(2) = sum(2) + average_filter(pad_size+1+j,pad_size+1+i)*padded_image(r+j,c+1,2);
                sum(3) = sum(3) + average_filter(pad_size+1+j,pad_size+1+i)*padded_image(r+j,c+1,3);
            end
        end
        Averaged_image(r,c,:)=sum(:);
    end
end
figure, imshow(uint8(Averaged_image)); title('Averaged Image');
```

Other Filters

Original

0	0	0
0	1	0
0	0	0

?

Source: D. Lowe

Other Filters

Original

0	0	0
0	1	0
0	0	0

Filtered
(no change)

Source: D. Lowe

Other Filters

Original

0	0	0
0	0	1
0	0	0

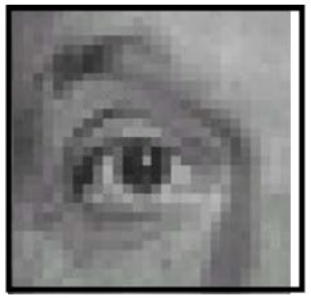
?

Source: D. Lowe

Other Filters

Original

0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

Source: D. Lowe

Combination of Filters

Original

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

(Note that filter sums to 1)

? What does smoothing take away?

Source: D. Lowe

Other Filters

Original

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

Sharpening filter

- Accentuates differences with local average

Source: D. Lowe

Sharpening

Original

Sharpened

Review - Derivative 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 $|\nabla f(x, y)| = \sqrt{f_x^2 + f_y^2}$
- Gradient Vector orientation $\theta(x, y) = \tan^{-1}(\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}})$

Image Derivative

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

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

- Image is discrete function
  - $\frac{df}{dx} = \frac{f(x) - f(x-1)}{1} = f'(x) = f_x$

$$f[i,j] = \begin{bmatrix} \ddots & & \vdots & & \\ & f[-1,-1] & f[0,-1] & f[1,-1] & \\ \cdots & f[-1,0] & f[0,0] & f[1,0] & \cdots \\ & f[-1,1] & f[0,1] & f[1,1] & \\ & & \vdots & & \ddots \end{bmatrix}$$

Discrete Derivative

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

Example

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

- Derivative Mask
  - Backward difference [-1 1]
  - Forward difference[1 -1]
  - Centre Difference[-1 0 1]

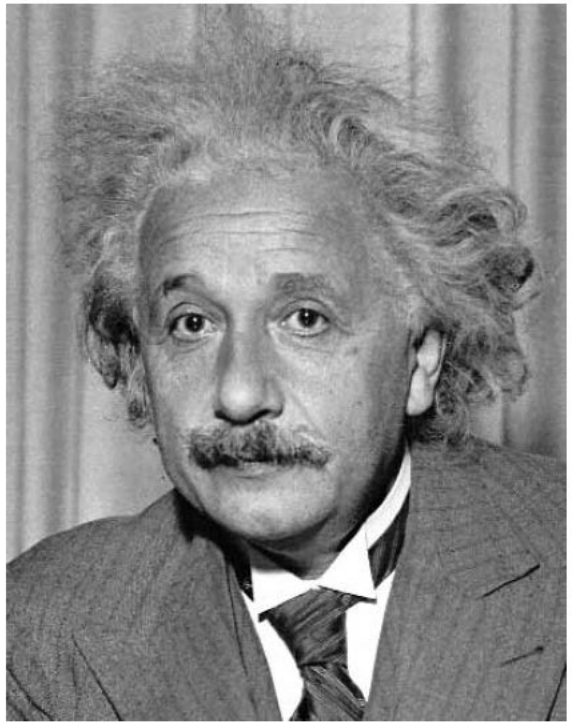
Derivative of Images (2D)

- Derivative filters

$$h_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad h_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

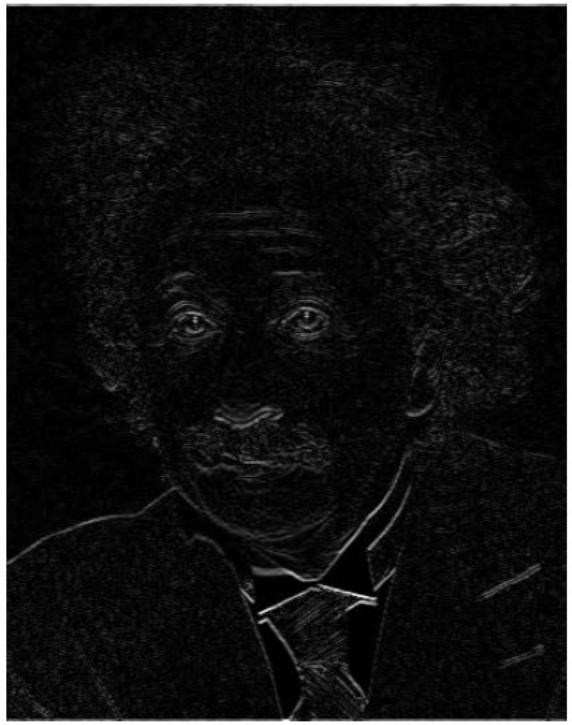
$$F = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \qquad F_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Derivative - Edge Filter



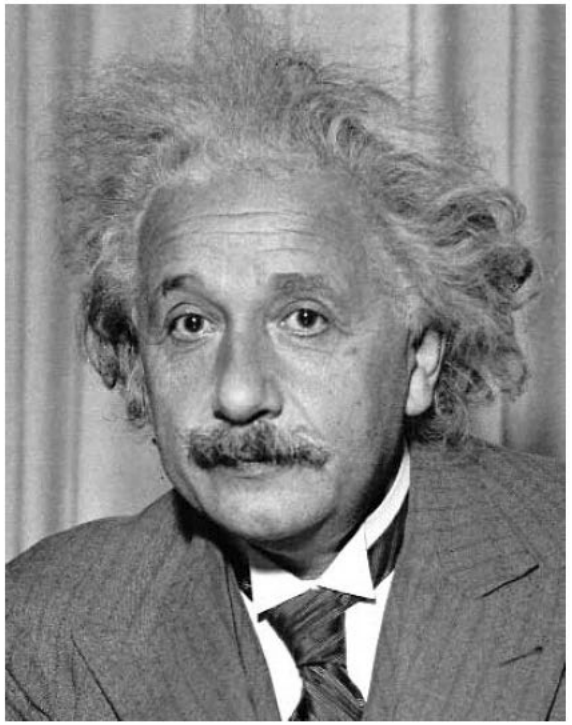
1	2	1
0	0	0
-1	-2	-1

Sobel



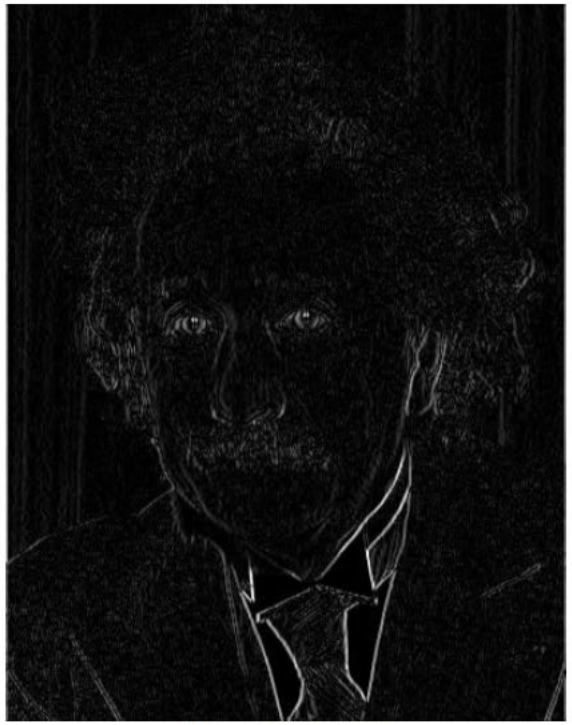
Horizontal Edge
(absolute value)

Derivative - Edge Filter



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

Sobel



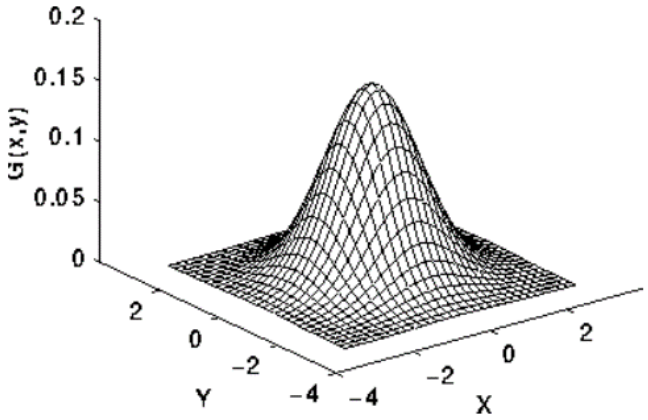
Vertical Edge
(absolute value)

Summary of Derivative/Edge Filter

- Find the intensity changes of the image in x and y directions
- Useful for detecting edges

Important Filter - Gaussian Filter

- Gaussian kernel
  - $g(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\left(\frac{i^2+j^2}{\sigma^2}\right)}$



$\frac{1}{273}$

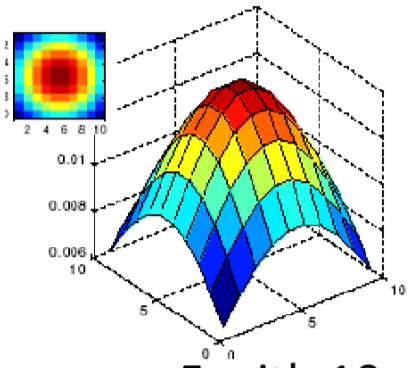
1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

5x5 kernel , σ=1

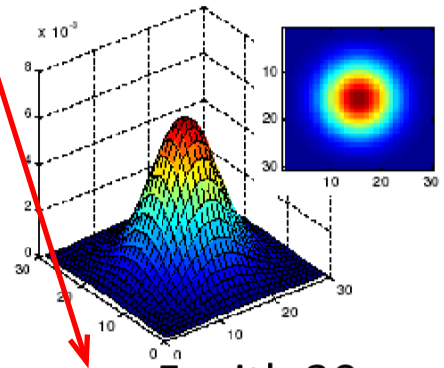
https://homepages.inf.ed.ac.uk

Gaussian filters

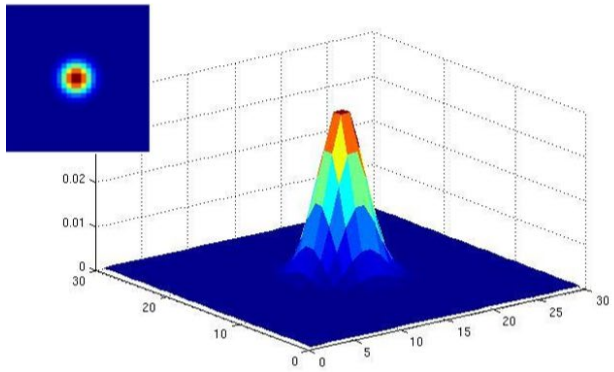
- What parameters do we have?
  - Size of Kernel – determines the support of Gaussian function
  - Variance of Gaussian – determines extent of smoothing



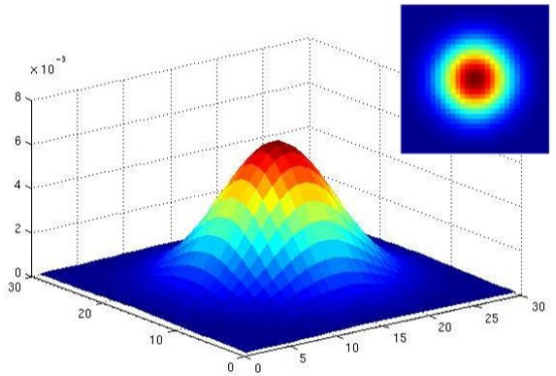
σ = 5 with 10 x 10 kernel



σ = 5 with 30 x 30 kernel



σ = 2 with 30 x 30 kernel



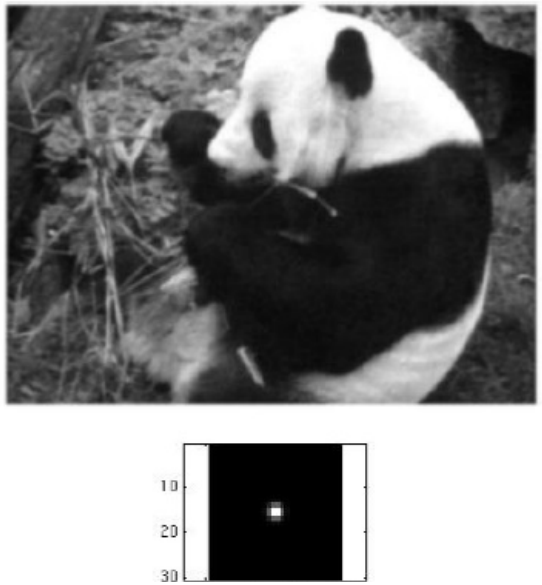
σ = 5 with 30 x 30 kernel

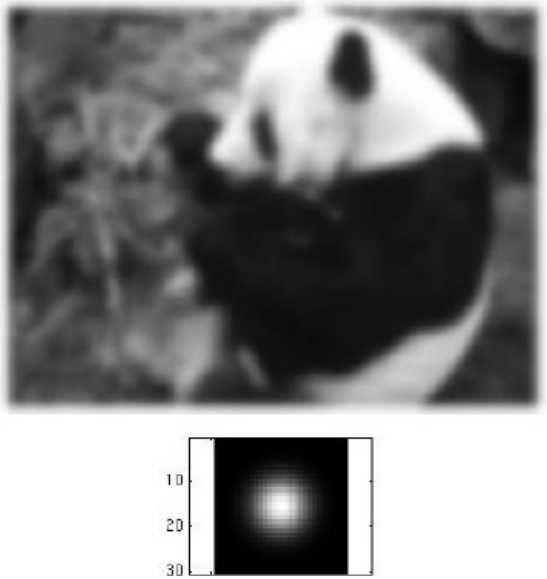
Gaussian Smoothing

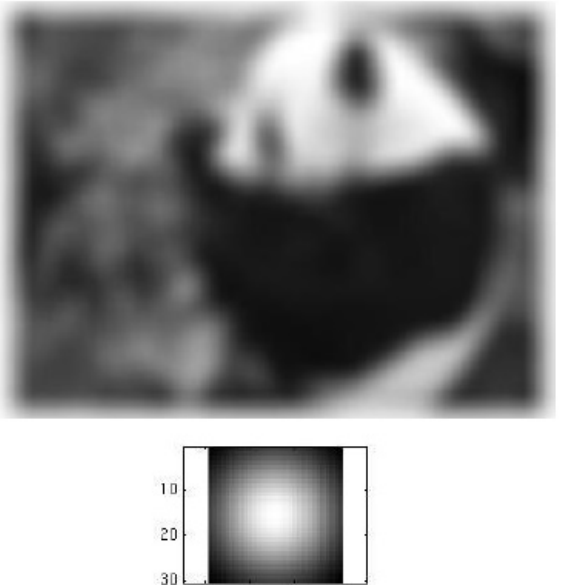
https://michaelbach.de/ot/cog-blureffects/index.html

Gaussian filter smoothing

Increasing σ



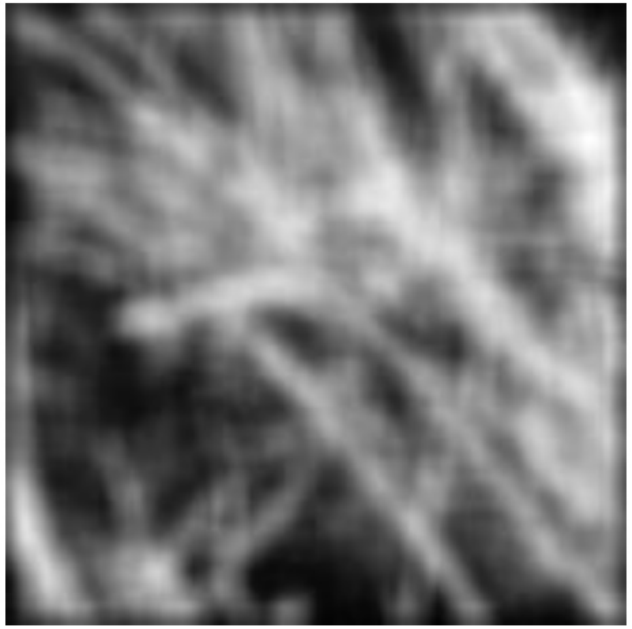




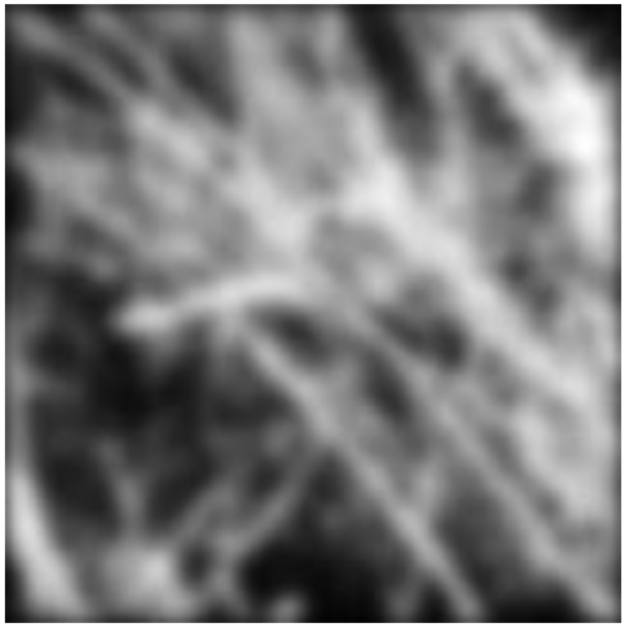
R. Dockter

Gaussian VS Box Filter

Original Image



box filter



Gaussian filter

Gaussian filters properties

- Remove “high-frequency” components from image (low-pass filter)
  - Image become more smooth
  - Most common natural model (used everywhere)

- 2D Gaussian filters can be separated into 2 1-D Gaussian filters

Source: K. Grauman

Important properties of linear filters

- Box filter, edge and Gaussian filters are linear filters

- Linearity:
  - $\text{filter}\,(f_1 + f_2) = \text{filter}(f_1)$+filter($f_2$)

- Shift invariance: same behavior regardless of pixel location
  - Filter(shift(f)) = shift(filter(f))

- Any linear and shift-invariant operator can be represented as a convolution

Source: S. Lazebnik

Convolutions

- All Linear filters can be represented by convolution

- What is convolution?

Recall - Correlation

- $f \otimes h = \sum_k \sum_l f(k,l) h(i+k, j+l)$

- $f = Image$
- $h = kernel$

f				h		
f_1	f_2	f_3		h_1	h_2	h_3
f_4	f_5	f_6	$\otimes$	h_4	h_5	H_6
f_7	f_8	f_9		h_7	h_8	h_9

$$f \otimes h = f_1 h_1 + f_2 h_2 + f_3 h_3 + f_4 h_4 + f_5 h_5 + f_6 h_6 + f_7 h_7 + f_8 h_8 + f_9 h_9$$

Source: M. Shah

Convolution

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

- $f = Image$
- $h = kernel$

h

h_1	h_2	h_3
h_4	h_5	H_6
h_7	h_8	h_9

flip x →

h_3	h_2	h_1
h_6	h_5	h_4
h_9	H_8	h_7

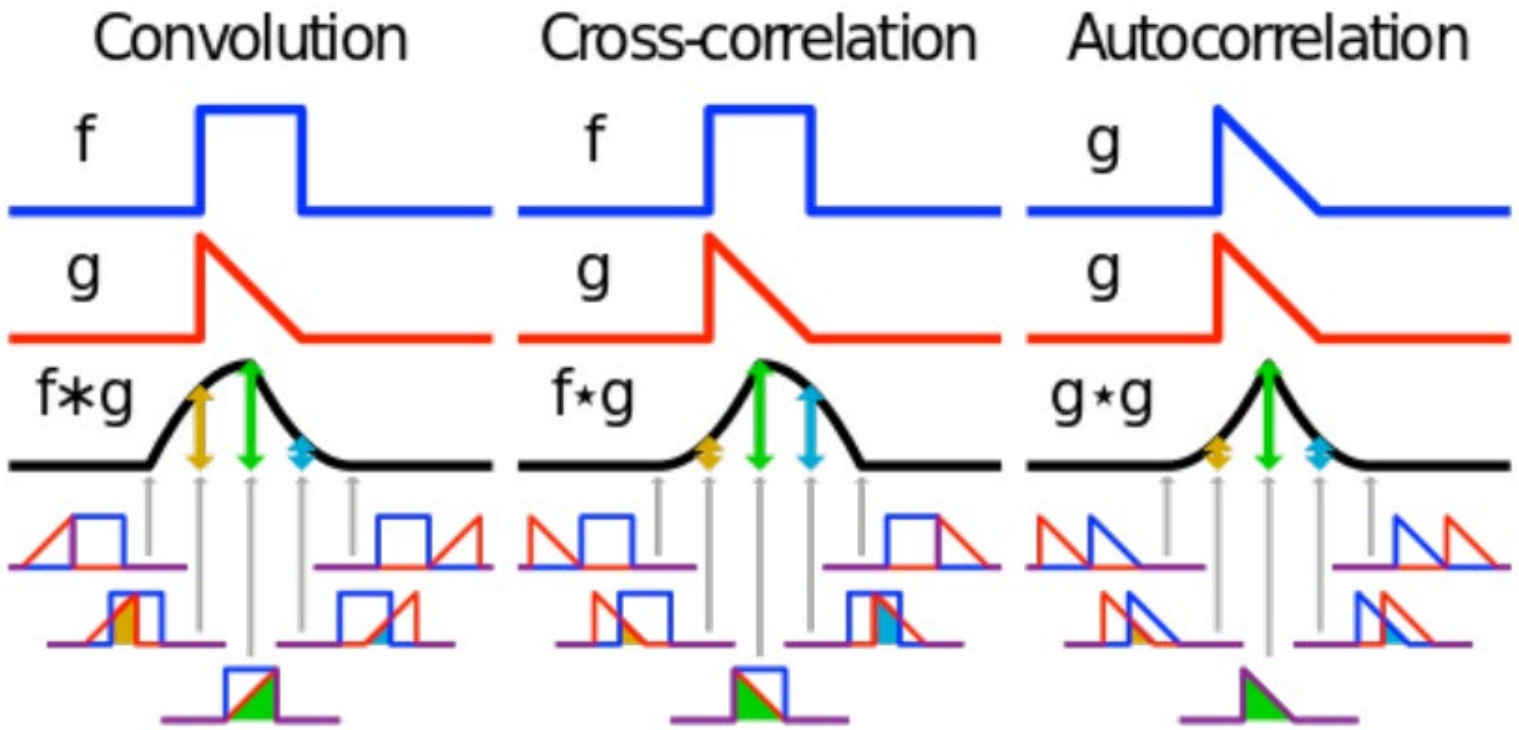
flip y →

h_9	h_8	h_7
h_6	h_5	h_4
h_3	h_2	h_1

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

Source: M. Shah

Convolution VS Cross Correlation



Properties about convolution

- Commutative property:
  - $a * b = b * a$
- Associative property:
  - $a * (b * c) = (a * b) * c$
- Distributive property:
  - $a * (b + c) = (a * b) + (a * c)$
- Scalar factor out:
  - $ka * b = a * kb = k(a * b)$

Source: S. Lazebnik

Back to Gaussian Filter

- Convolution with itself is another Gaussian
  - So repeat smoothing with small-width kernel get same result as larger-width kernel
  - Convolving two times with Gaussian kernel of width σ is same as convolving once with kernel of width $\sqrt{2}\sigma$
- Fourier Transform of a Gaussian is Gaussian
- Separable Kernel
  - Factor into product of two 1D Gaussians

Source: K. Grauman

Separability of Gaussian filter

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

$$= \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{y^2}{2\sigma^2}}\right)$$

- The 2D Gaussian can be expressed as the product of two functions, one in x and the other in y
- In this case, two functions are 1D Gaussian

Source: D. Lowe

Separability example

- $y * h = \begin{bmatrix} 10 & 20 & 30 \\ 40 & 50 & 60 \\ 70 & 80 & 90 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$

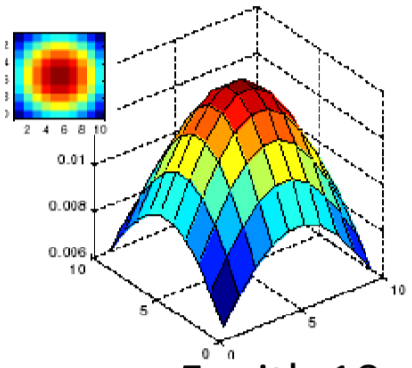
- The 2D convolution mask can be separated into 2 1-D mask
- $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$

- Using 2D convolution
  - $y[1,1] = 10 \times 1 + 20 \times 2 + \cdots + 90 \times 1 = 800$

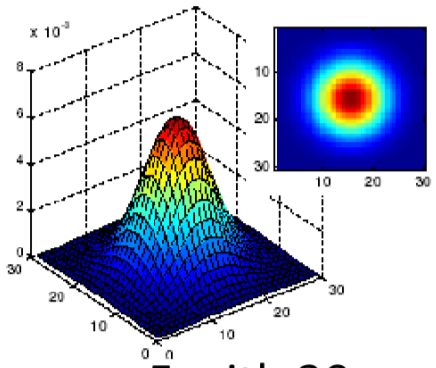
- Using 1D convolution separately
  - $\begin{bmatrix} 10 & 20 & 30 \\ 40 & 50 & 60 \\ 70 & 80 & 90 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 160 & 200 & 240 \end{bmatrix}$
- Apply another 1D convolution
- $\begin{bmatrix} 160 & 200 & 240 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$
  - $=800$

Practical matter for Gaussian Filter

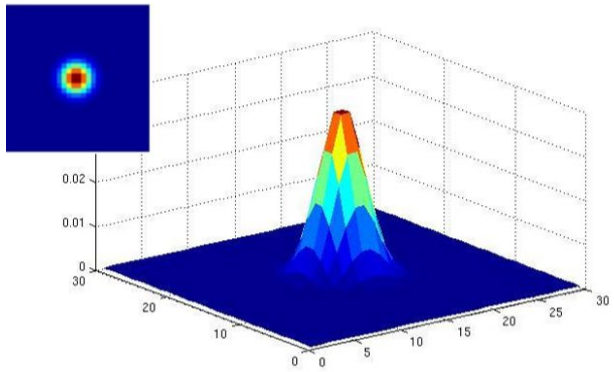
- How big should the filter be?
  - Values at edges should be near zero
  - Rule of thumb for Gaussian: set filter half-width to about 3σ



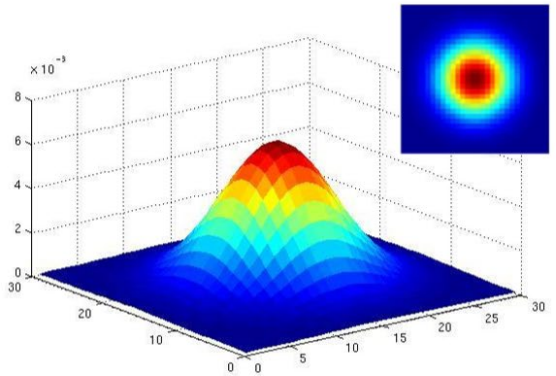
σ = 5 with 10
x 10 kernel



σ = 5 with 30
x 30 kernel



σ = 2 with 30
x 30 kernel



σ = 5 with 30
x 30 kernel

Edge Effect

- Computer will only convolve finite support signals.
- What happens at the edge?
  - Need to extrapolate
  - Methods
    - Zero “padding”
    - Edge value replication
    - Mirror extension
    - More….

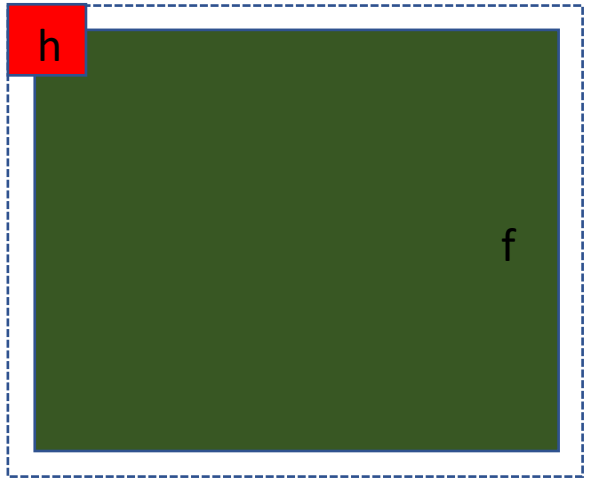


Image Border

Zero

Edge Value replication

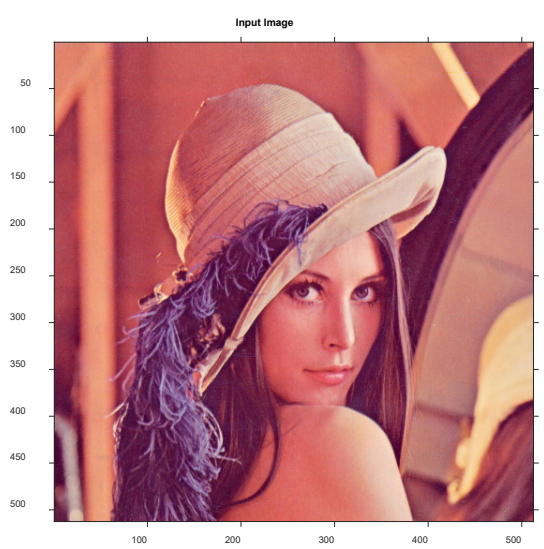
Periodic

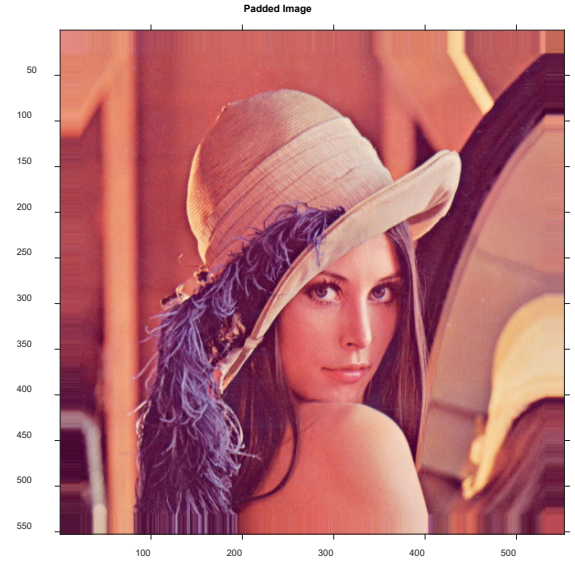
Reflected

Slide: R Dockter

MATLAB Padding Example

```
clear all; close all; clc;

pad_size=50;

input_image = imread('lena.jpg');

figure, imshow(input_image); title('Input Image');


% Adding Padding

% 0 | numeric scalar | 'circular' | 'replicate' | 'symmetric'
| string scalar | character vector | missing

padded_image = padarray(input_image, [pad_size,pad_size],
'replicate');

figure, imshow(padded_image); title('Padded Image');


% Remove Padding

[Row,Col,S] = size(padded_image);

Unpadded_image = zeros(Row-2*pad_size,Col-2*pad_size,S);


%Cast to unsigned integer for image display.

Unpadded_image = uint8(padded_image(pad_size+1:Row-
2*pad_size,pad_size+1:Col-2*pad_size,1:3));

figure, imshow(Unpadded_image); title('UnPadded Image');
```





padval — Pad value

0 | numeric

scalar | 'circular' | 'replicate' | 'symmetric' | string scalar | character vector | missing

Summary of Linear Filters

- Linear Filtering is sum of dot product at each position
  - Smoothing, Sharpening, Translating
- Linear Filters can be represented as 2D convolution
- Be aware of the filter size and edge effect.

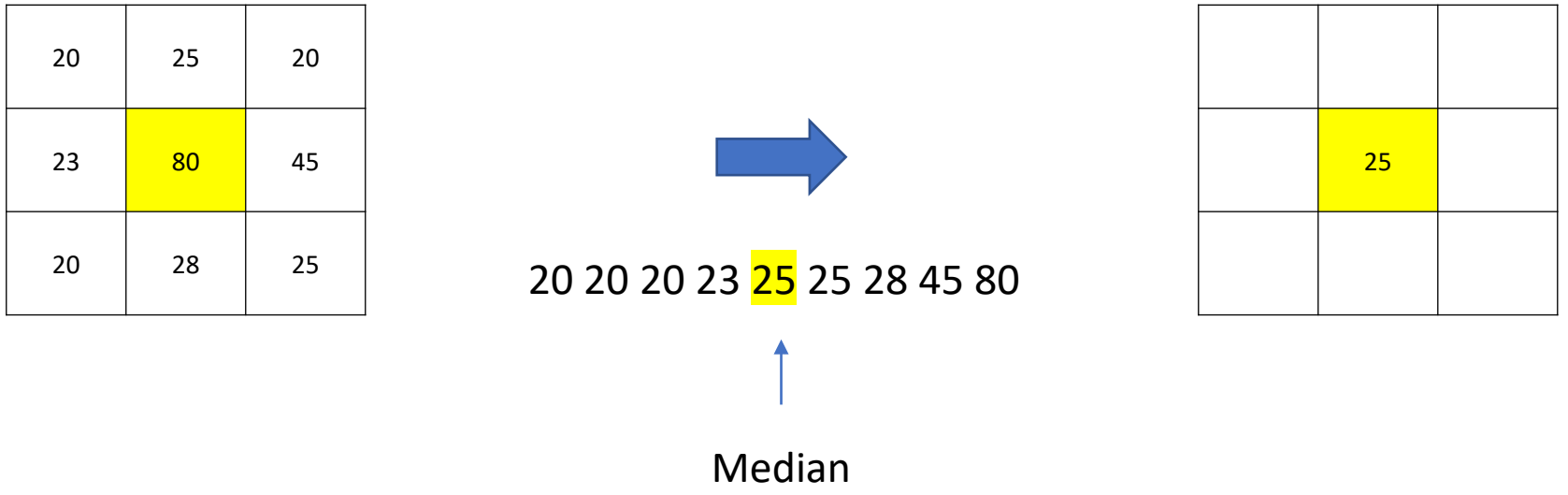
Other Non-linear Filters-Median filters

- A Median Filter operates over a window by selecting the median intensity in the window
- What advantage does a median filter have over a mean filter?
- Is a median filter a kind of convolution?

Slide by steve Seitz

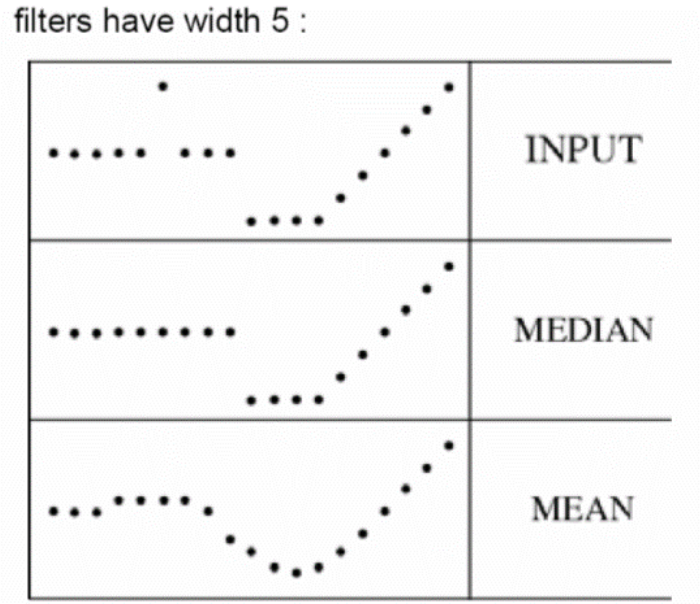
Median Filter

- Median filter operates over a window by selecting the median intensity in the windows



Median Filter

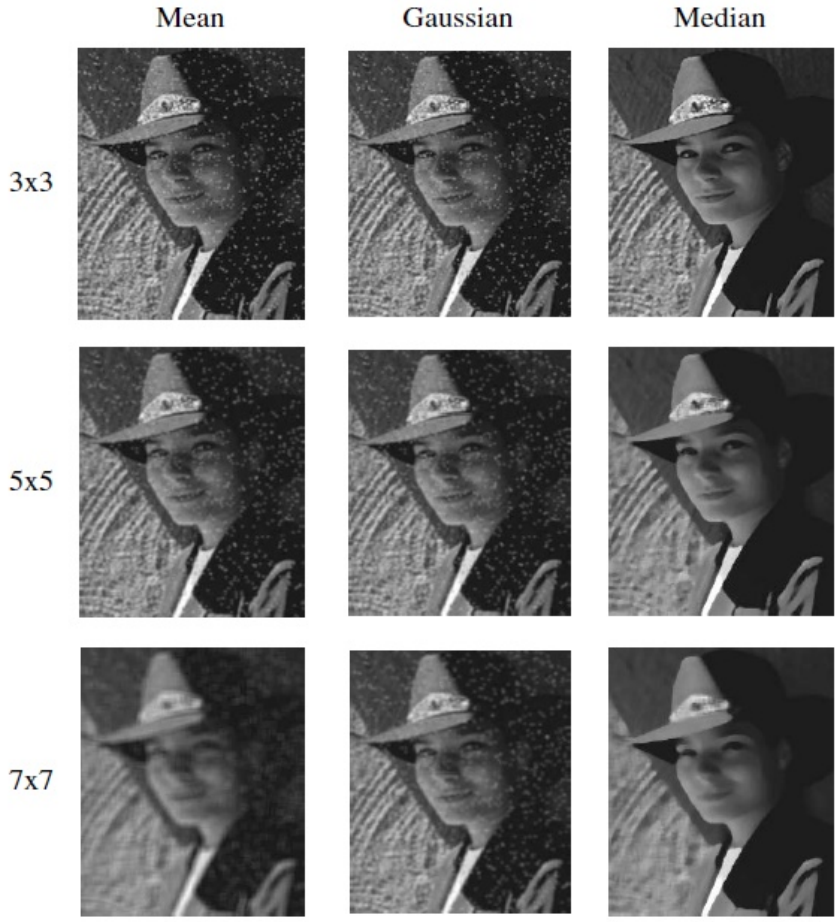
- Advantages of Media Filter over Gaussian
  - Edge preserving



Source: K. Grauman

Image Denoising

Comparison: Salt and Pepper Noise



Slide by Steve Seitz

Other Non-linear Filters

- Simple image threshold

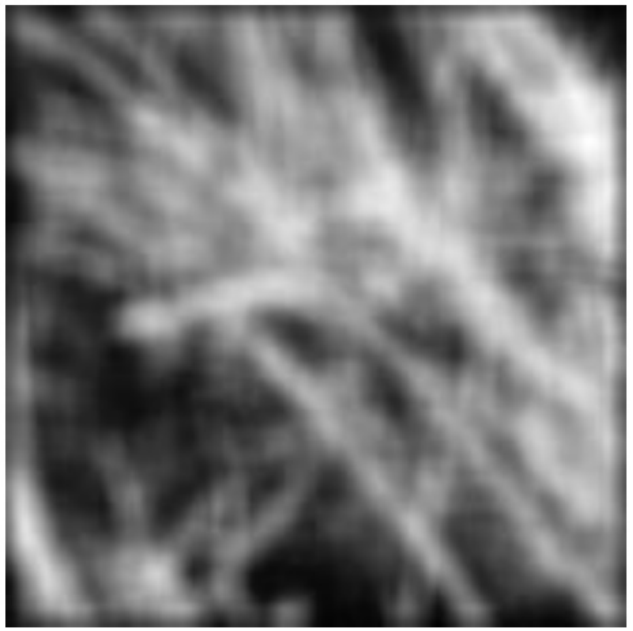
- $g[n,m] = \begin{cases} 255, f[n,m] > threshold \\ 0, otherwise \end{cases}$

Today's Agenda

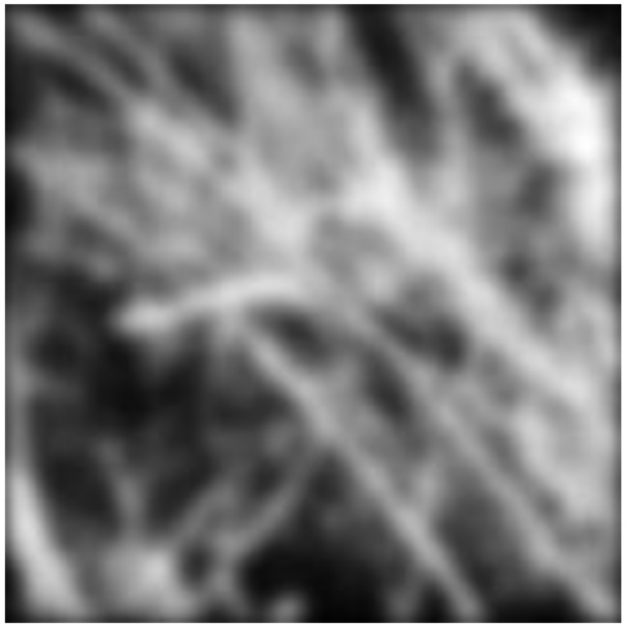
- Image as functions
- Filters in Spatial domain
- Filters in Frequency Domain
- Template Matching
- Image Pyramid

Why does Gaussian give a nice smooth image than square filter ?

Original Image



box filter



Gaussian filter

Gaussian is a low-pass filter

Why does a lower resolution image still make sense to us? What do we lose?

Image: https://www.flickr.com/photos/igorms/136916757/in/photostream/

Slide: Hoiem

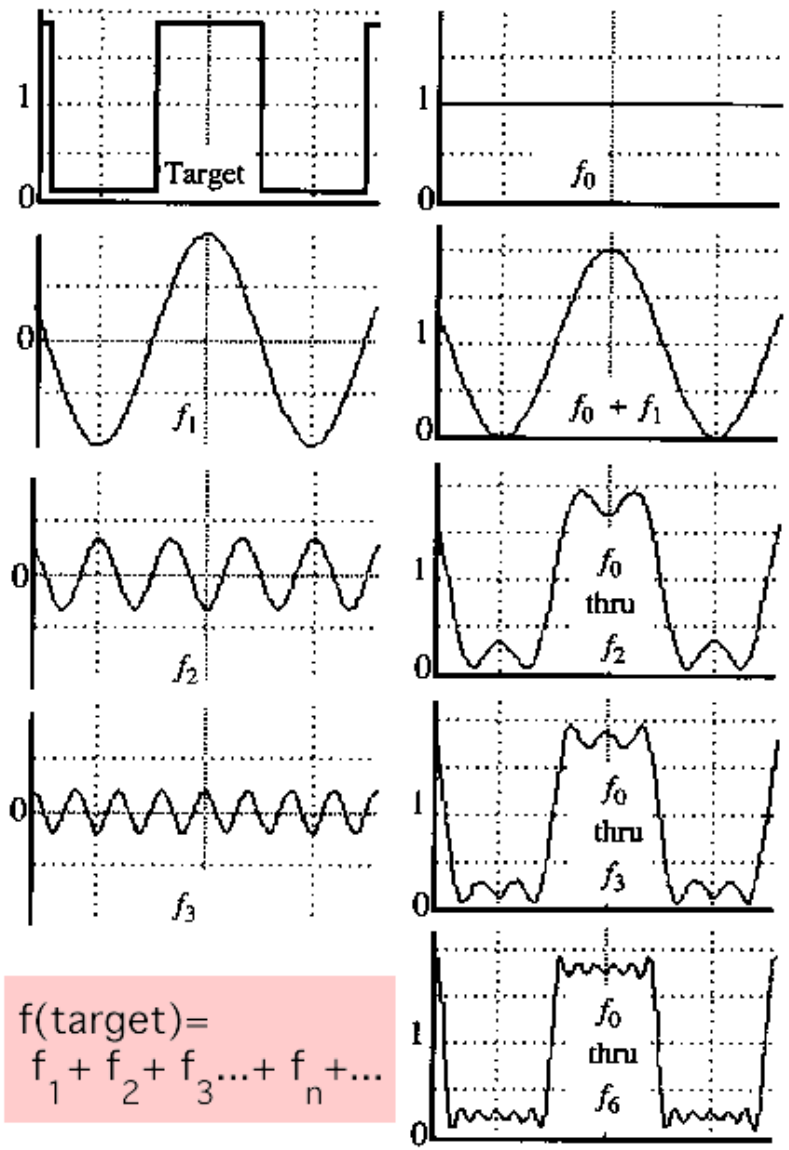
How to Represent Signals?

- Option 1: Taylor series represents any function using polynomials
  - $f(x) = f(a) + f'(a)(x-a) + \frac{f"(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n$
- Polynomials are not the best – unstable and not very physically meaningful
- Easier to talk about "signals" in terms of its "frequencies" (how fast/often signals change, etc).

Source: S. Narasimhan

A Sum of Sinusoids

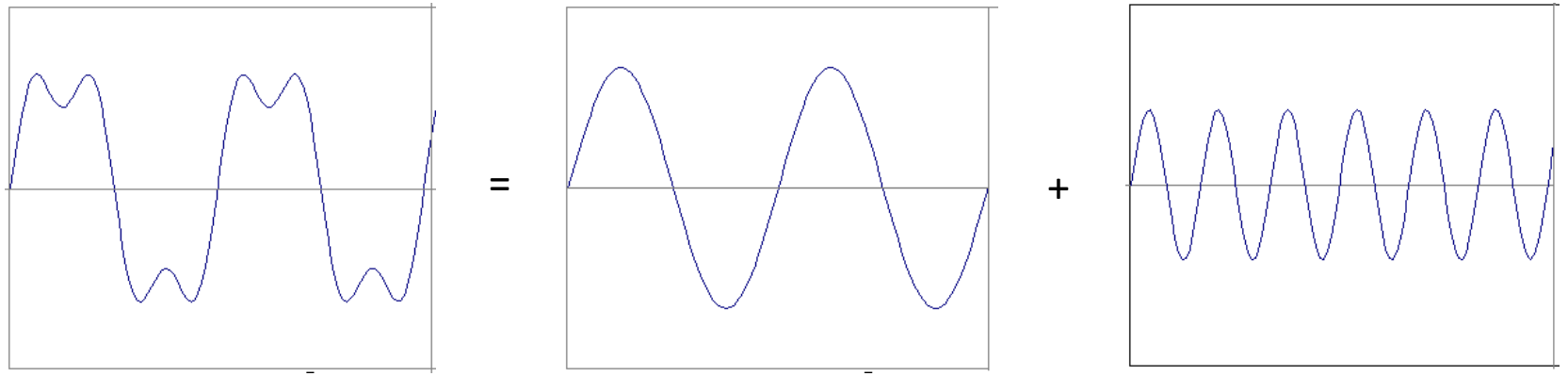
- Our building block
  - $A\sin(\omega x + \phi)$
- Add enough of them to get any signal $f(x)$ you want.

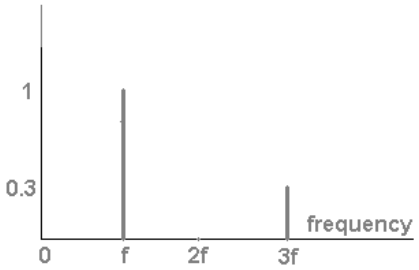


Slides: S. Narasimhan

Frequency Spectra

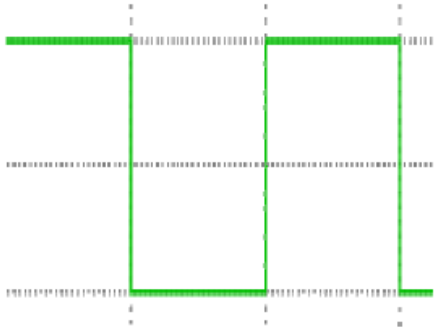
- Example



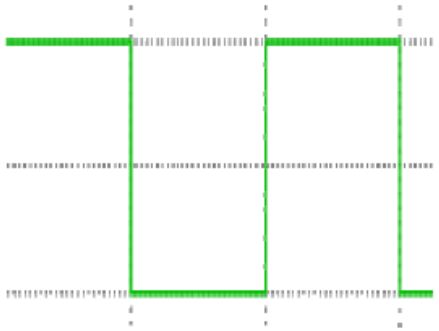


Slides: Efros

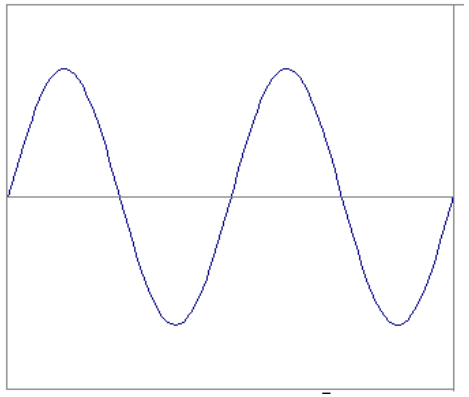
Frequency Spectra



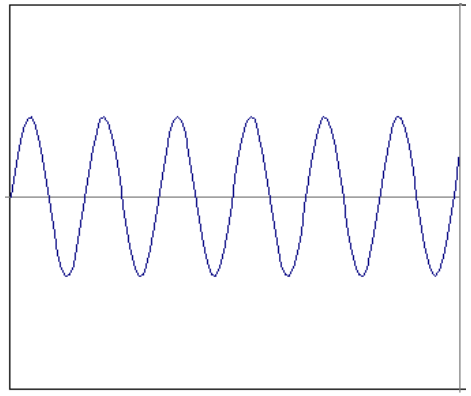
Frequency Spectra



=

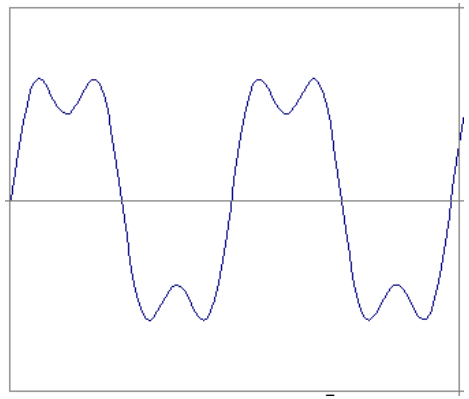


+



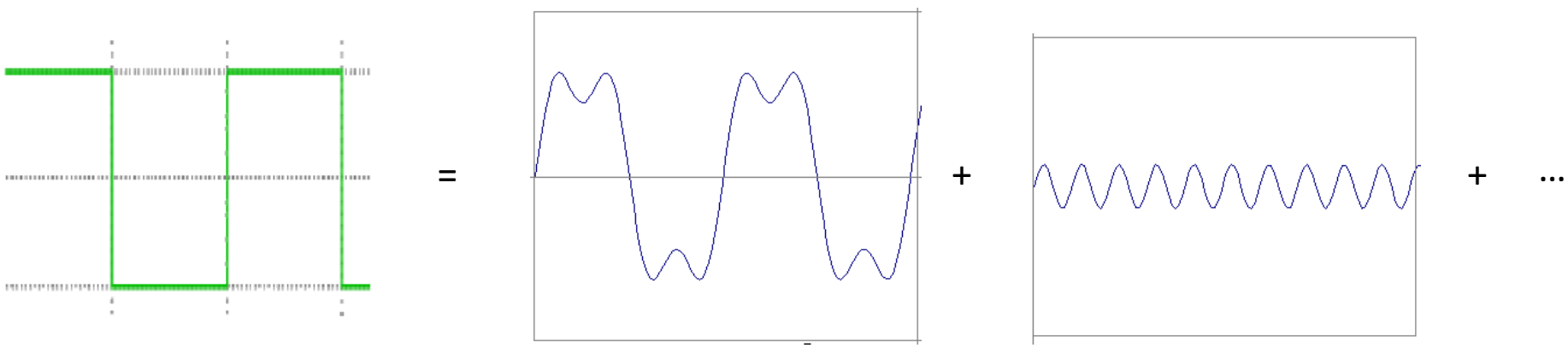
+ …

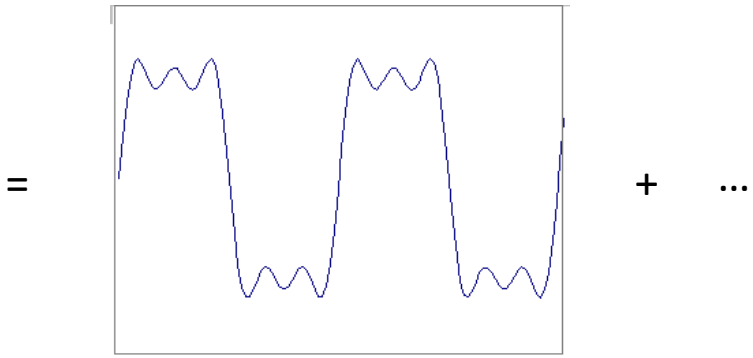
=



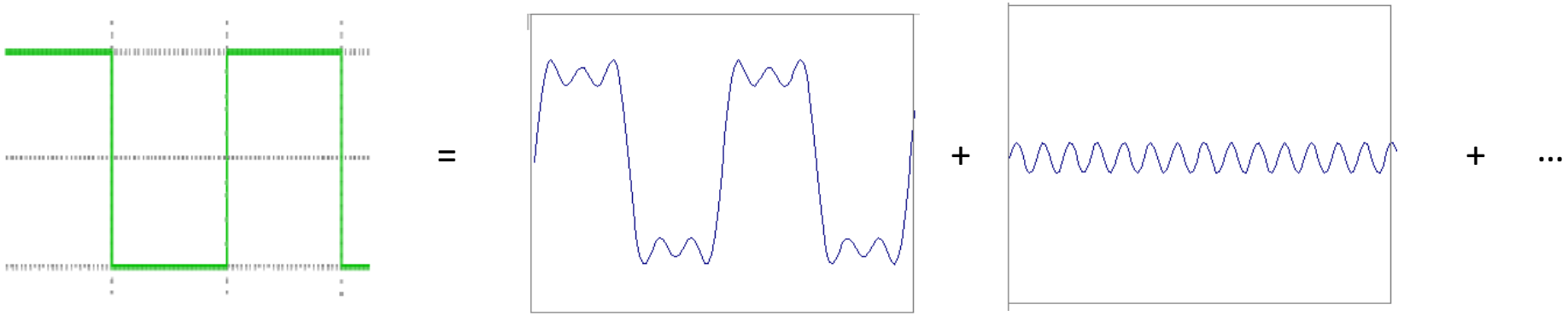
+ …

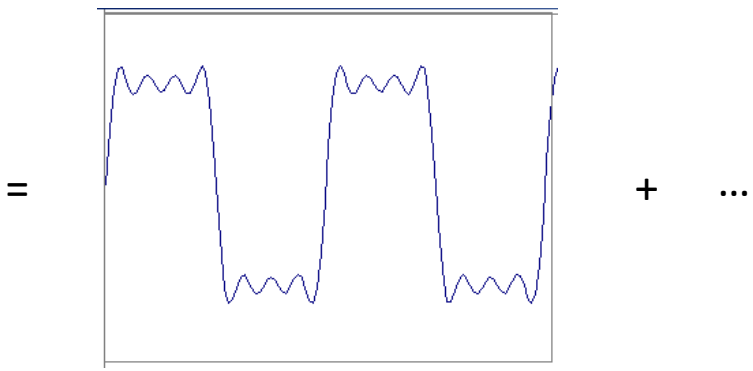
Frequency Spectra



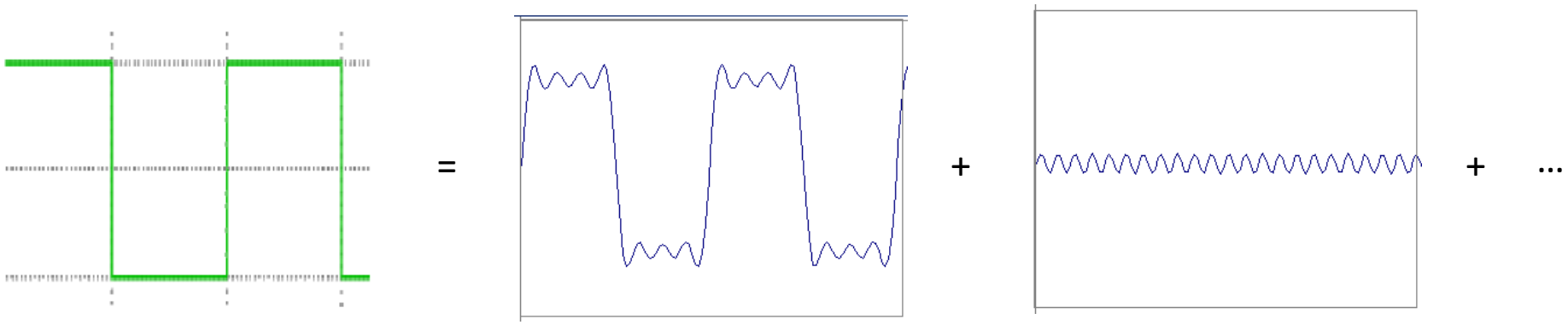


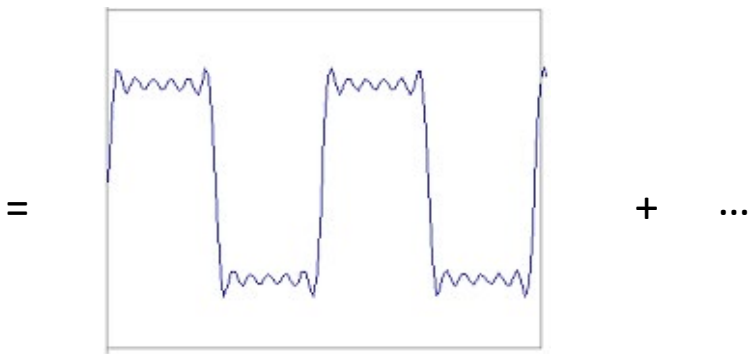
Frequency Spectra



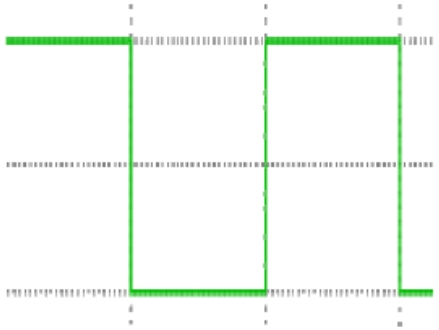


Frequency Spectra



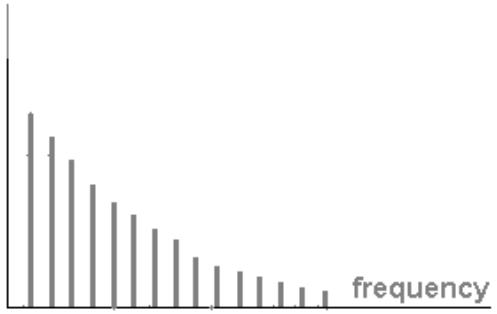


Frequency Spectra



=

$$A \sum_{k=1}^{\infty} \frac{1}{k} \sin(2\pi k t)$$



Fourier Transform

- Represent the signal as an infinite weighted sum of an infinite number of sinusoids

  - $F(u) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi ux} dx$

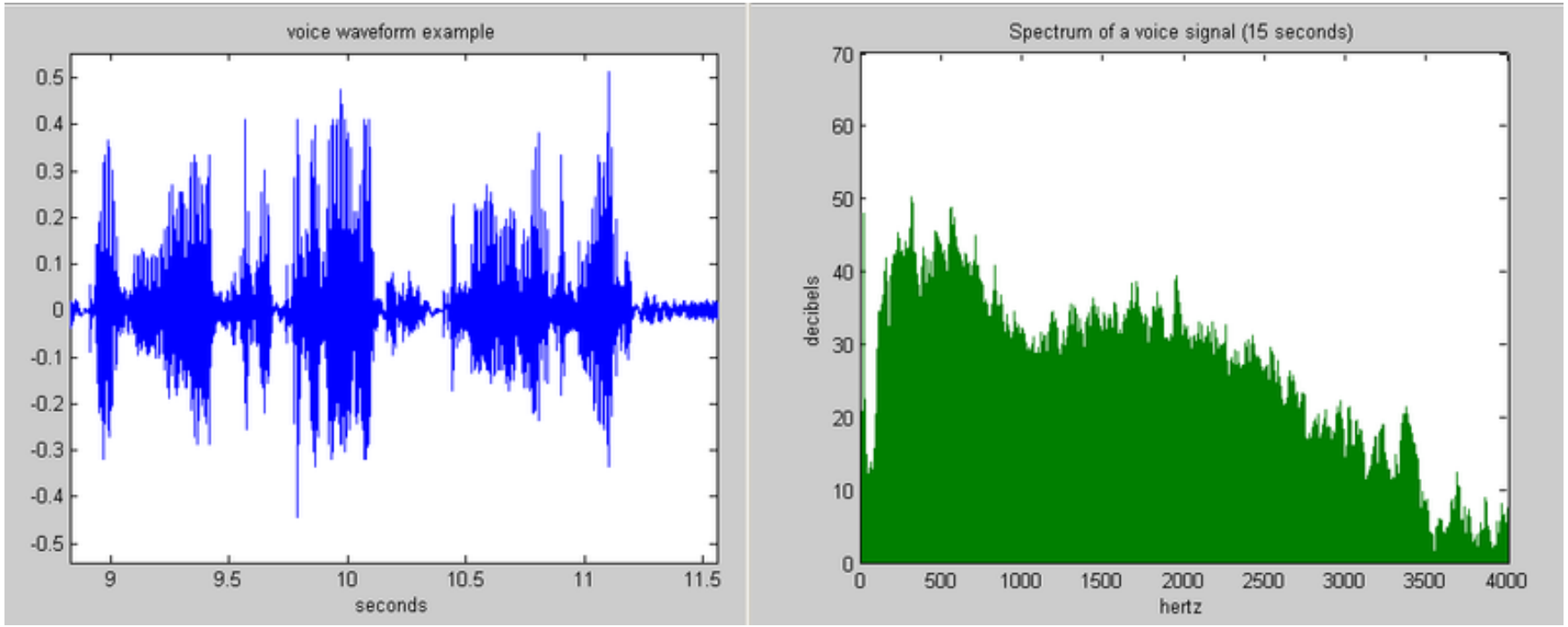
Where $e^{ik} = cosk + isink$ $\quad\quad$ $i = \sqrt{-1}$

Inverse Fourier Transform

- $f(x) = \int_{-\infty}^{\infty} F(u) e^{i2\pi ux} dx$

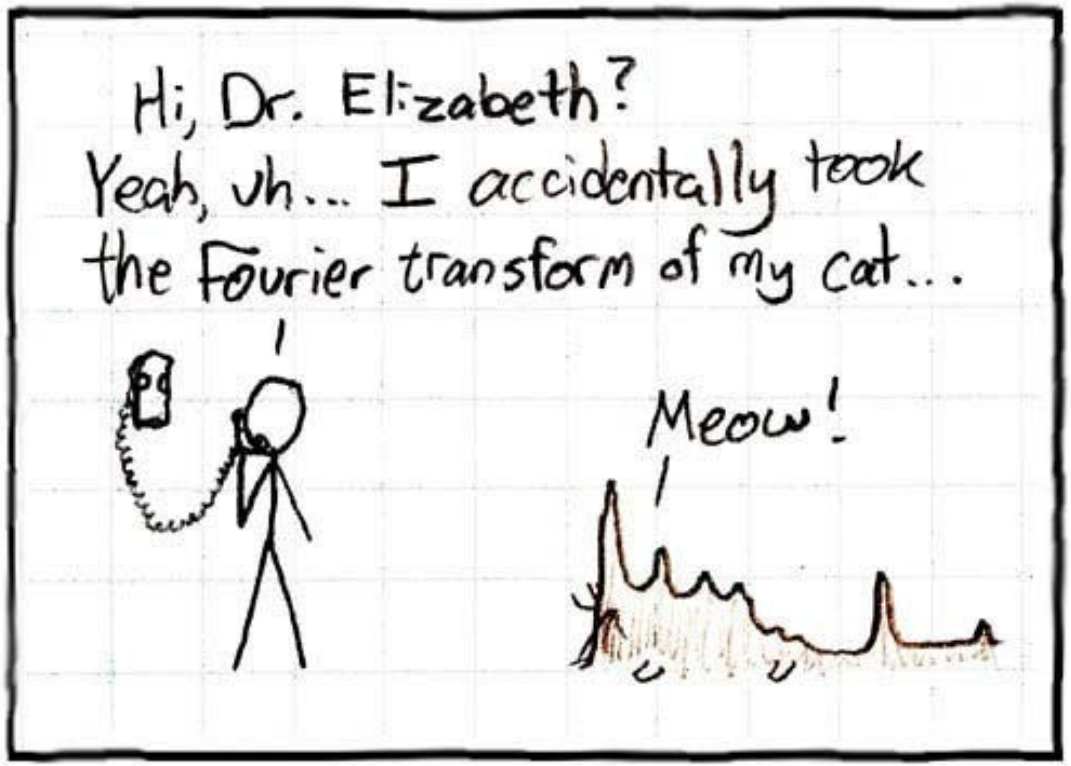
Example: Music

- We think of music in terms of frequencies at different magnitudes



Other signals

- We can also think of all kinds of other signals the same way



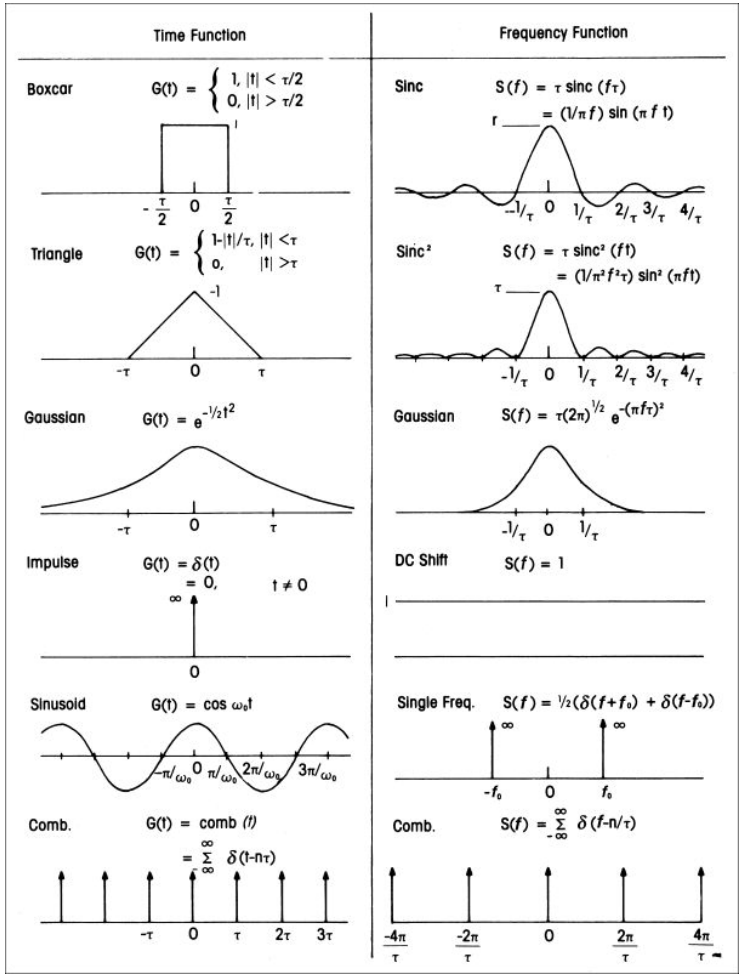
Source: xkcd.com

Jean Baptiste Joseph Fourier (1768-1830)

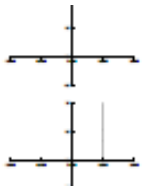
- Had crazy idea (1807):
  - Any periodic function can be rewritten as a weighted sum of Sines and Cosines of different frequencies
  - Don't believe it?
    - Neither did Lagrange, Laplace, Poisson and other big wigs
    - Not translated into English until 1878
  - But it's true!
    - Called Fourier Series
    - Possibly the greatest tool used in Engineering

Source: S. Narasimhan

Fourier Transform Pairs



impulse $\delta(x) \Leftrightarrow 1$



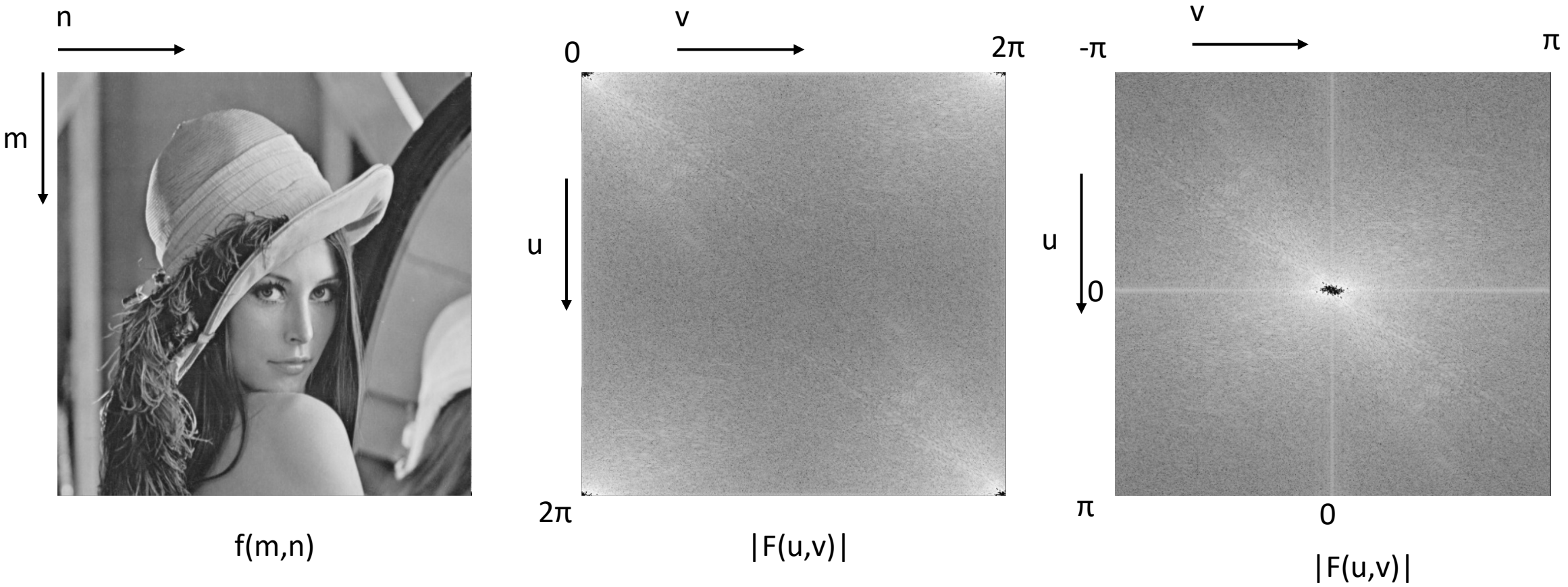
shifted impulse $\delta(x-u) \Leftrightarrow e^{-j\omega u}$

https://wiki.seg.org

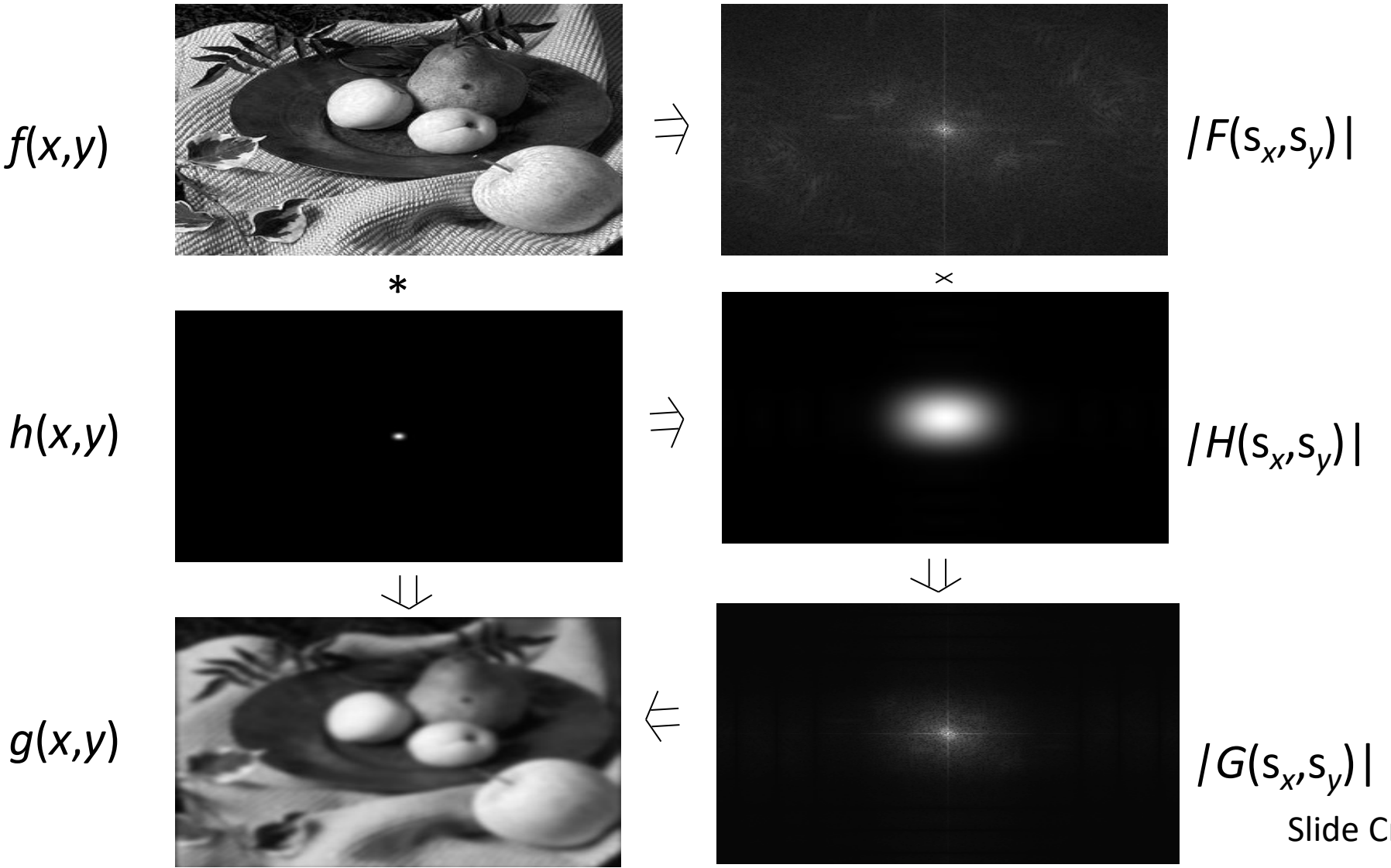
Fourier Transform and Convolution

- The Fourier transform of the convolution of two function is the product of their Fourier transforms
  - $F[g * h] = F[g]F[h]$
- Convolution in spatial domain is equivalent to multiplication in frequency domain
  - $g * h = F^{-1}[F[g]F[h]]$

Fourier Transform of Images

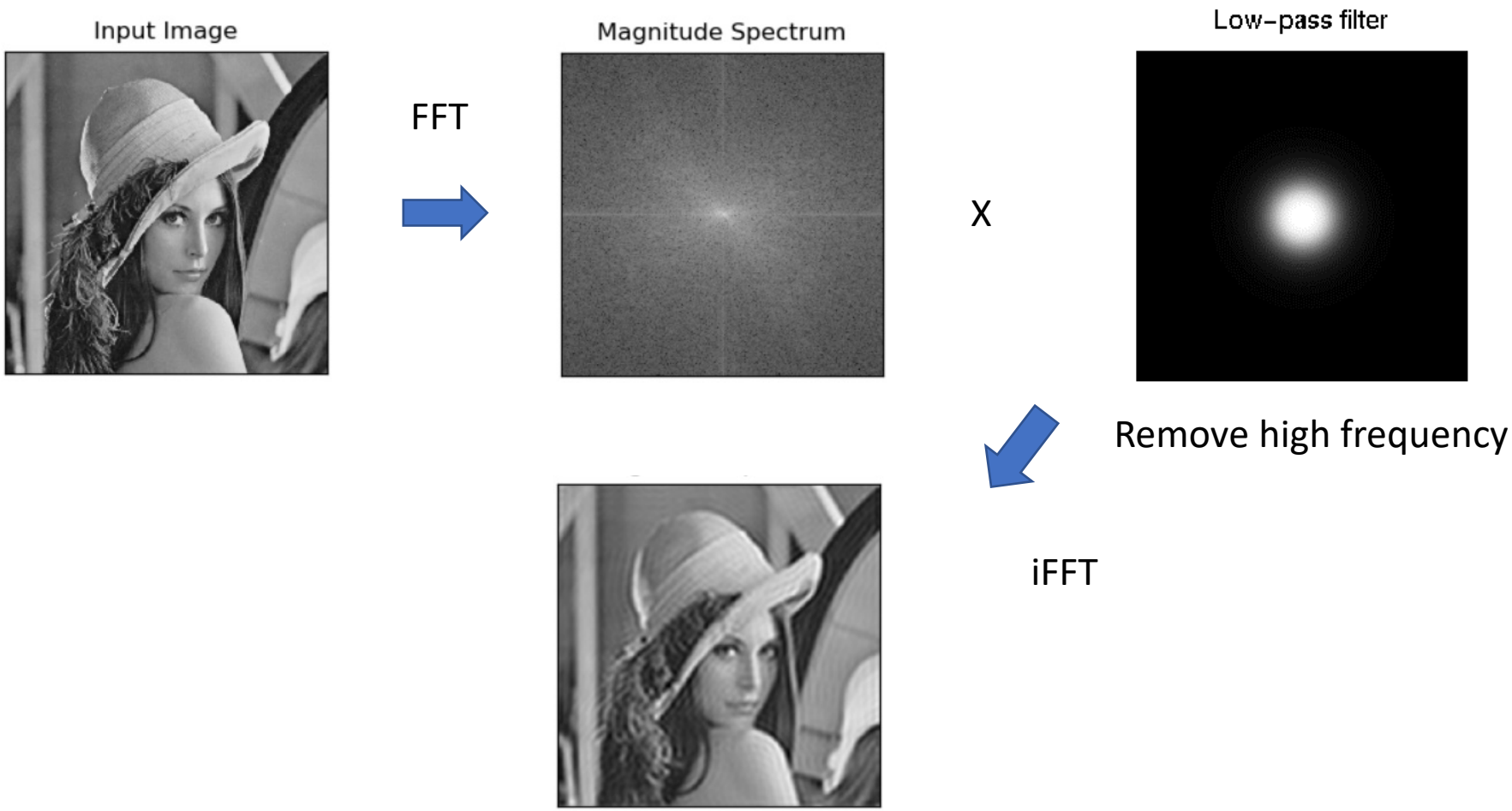


Convolution is Multiplication in Fourier Domain



Slide Credit: S Narasimhan

Low-pass Filtering

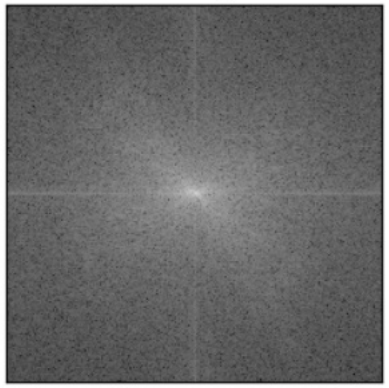


High-Pass Filtering

Input Image

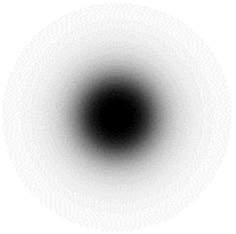
FFT

Magnitude Spectrum



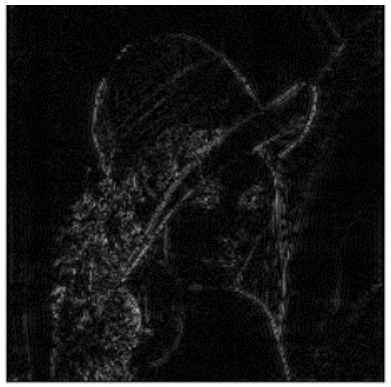
X

High–pass filter

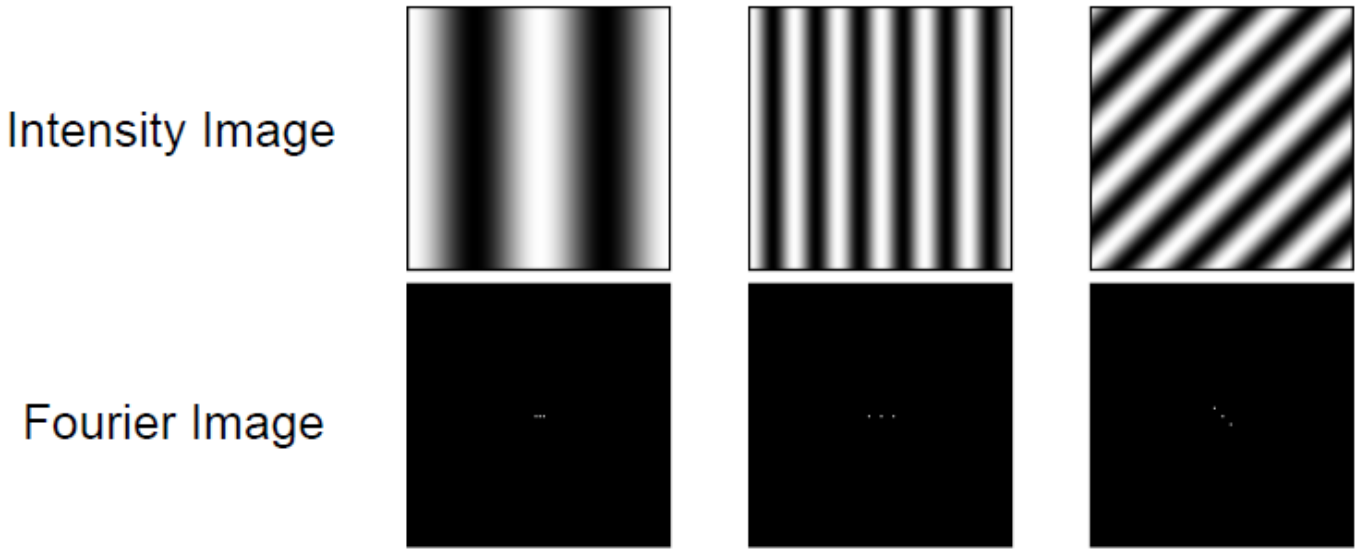


Remove low frequency

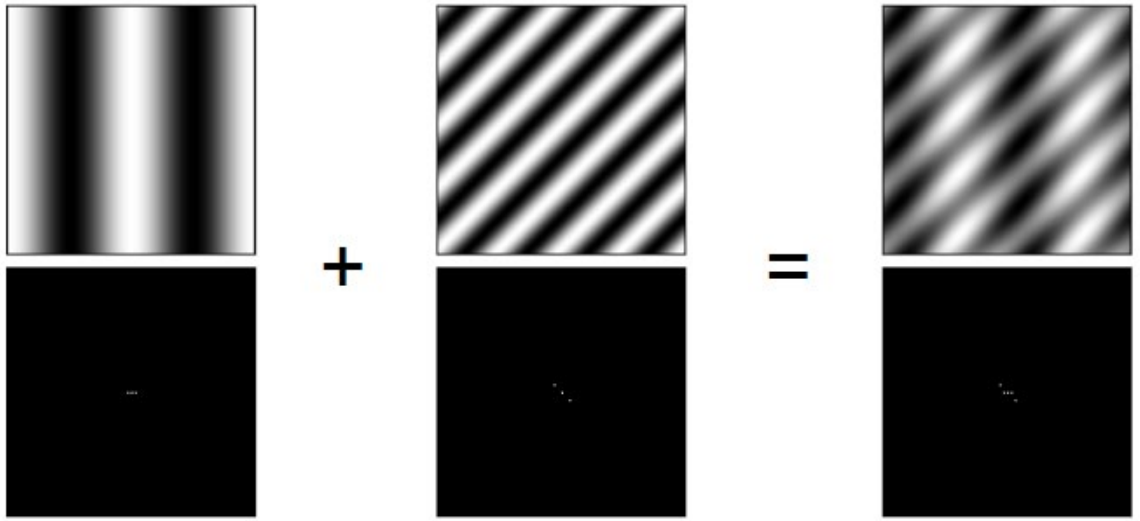
Inverse FFT



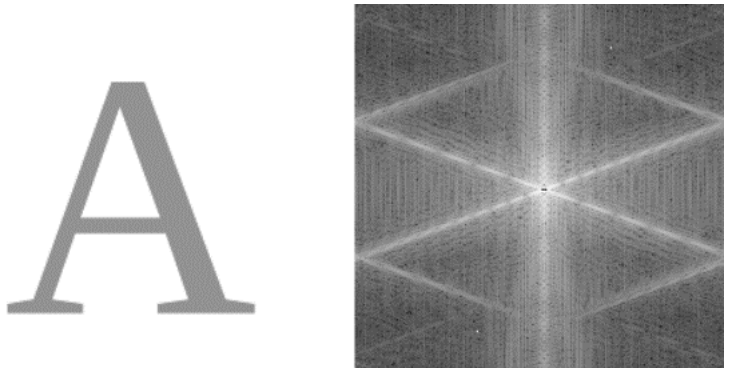
Fourier Analysis in images

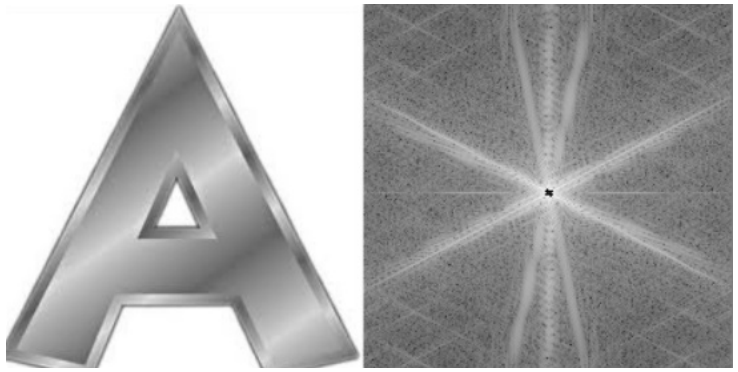


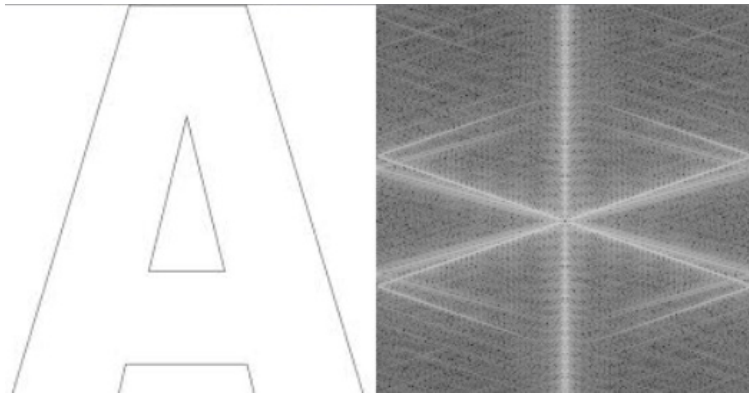
Signals can be composed

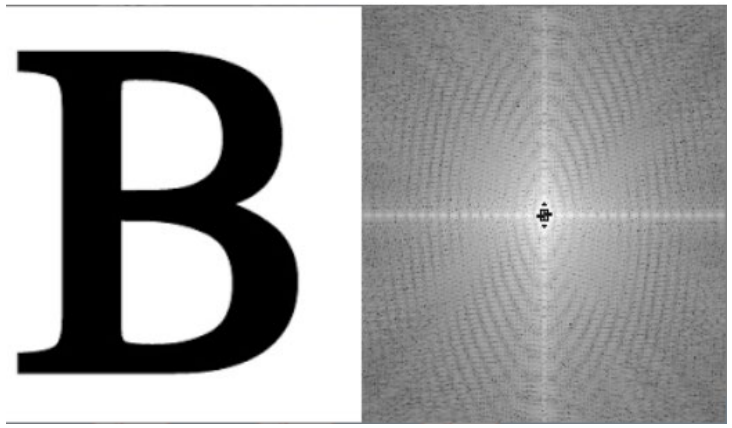


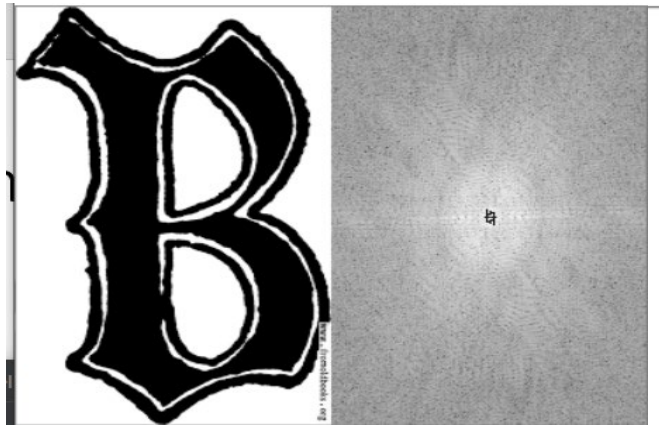
Fourier Transform of Similar Images











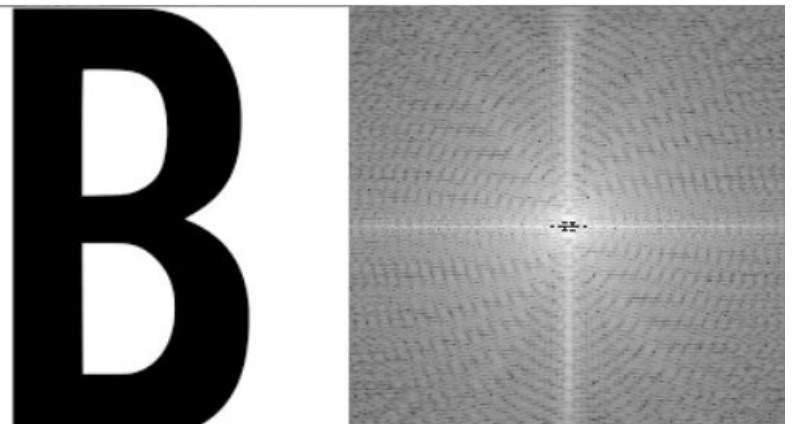
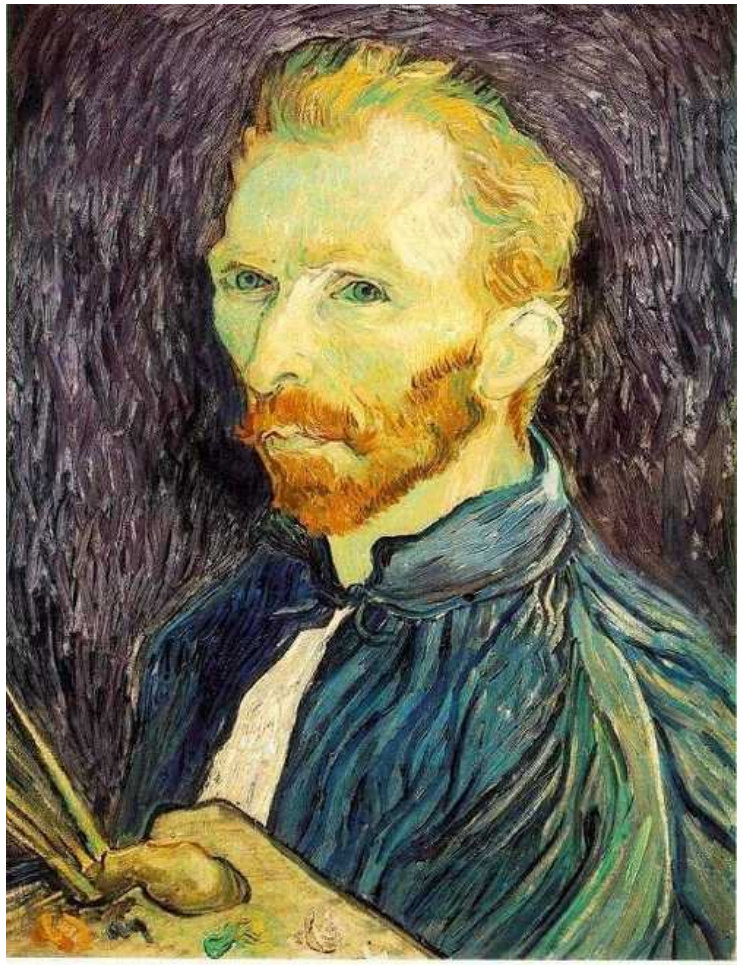
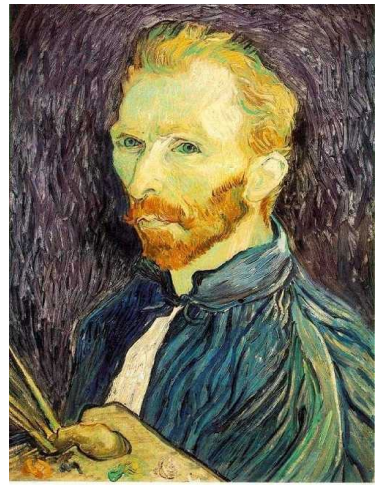


Image Subsampling

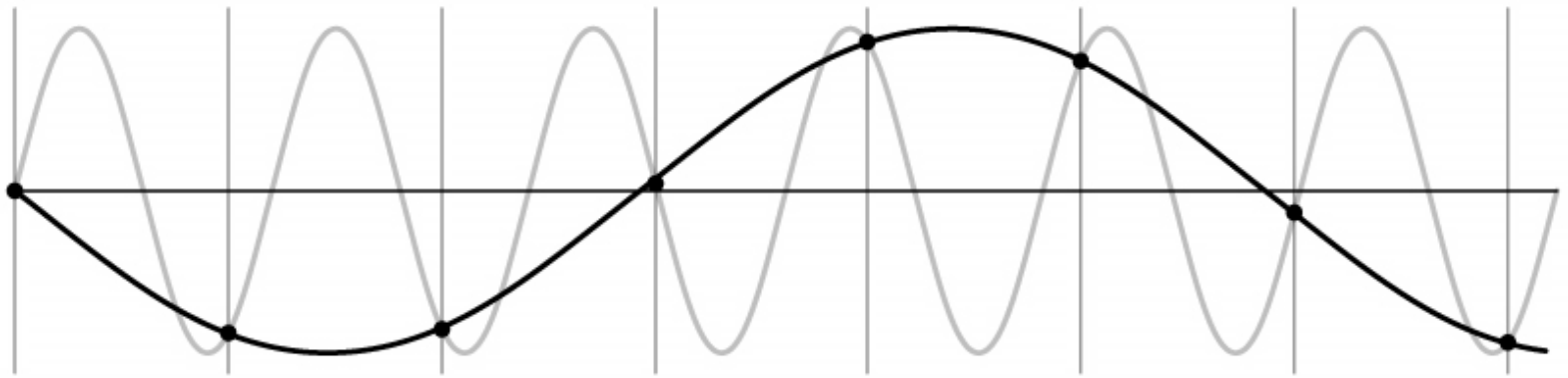




Subsampling by factor of 2 is done by throw away every other row and column to generate ½ size image

Aliasing problem

- 1D Sinewave example



Source: S. Marschner

Aliasing problem

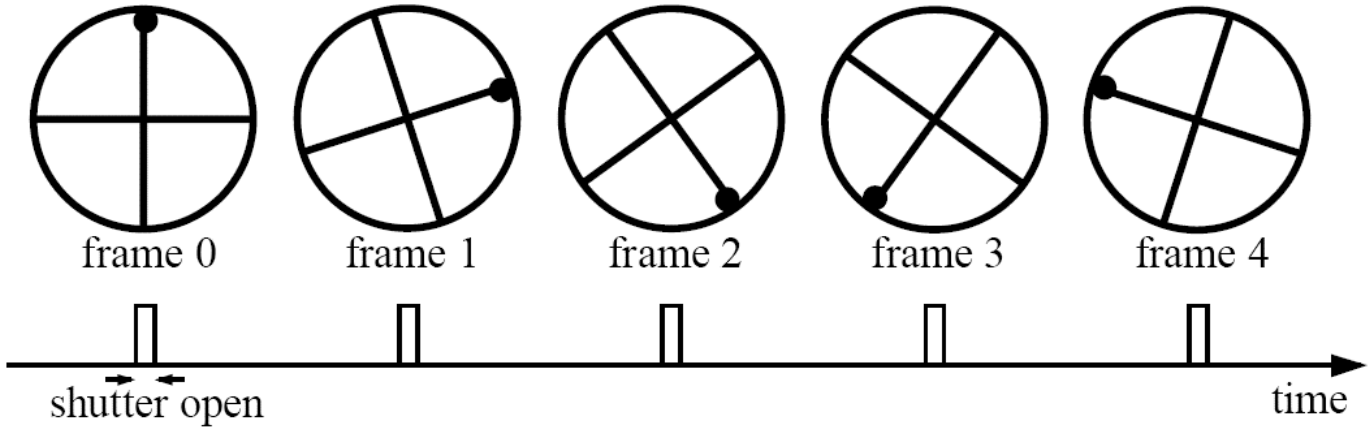
- Sub-sampling may be dangerous
- Characteristic error may appear:
  - “Wagon wheel rolling the wrong way in movies”
  - “Checkerboards disintegrate in ray tracing”
  - “Striped shirts look funny on color television

Source: D. Forsyth

Aliasing problem

Imagine a spoked wheel moving to the right (rotating clockwise). Mark wheel with dot so we can see what's happening.

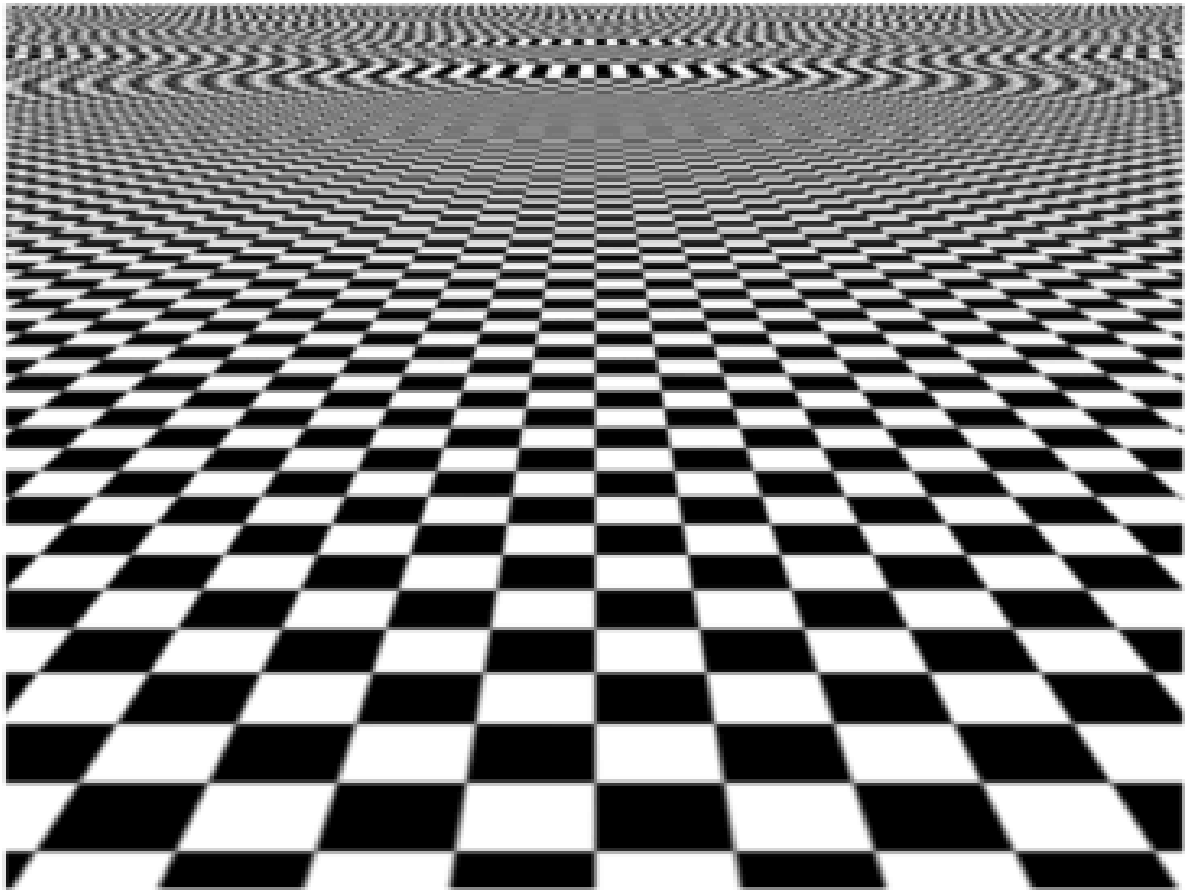
If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):



Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)

Slide by Steve Seitz

Aliasing in graphics



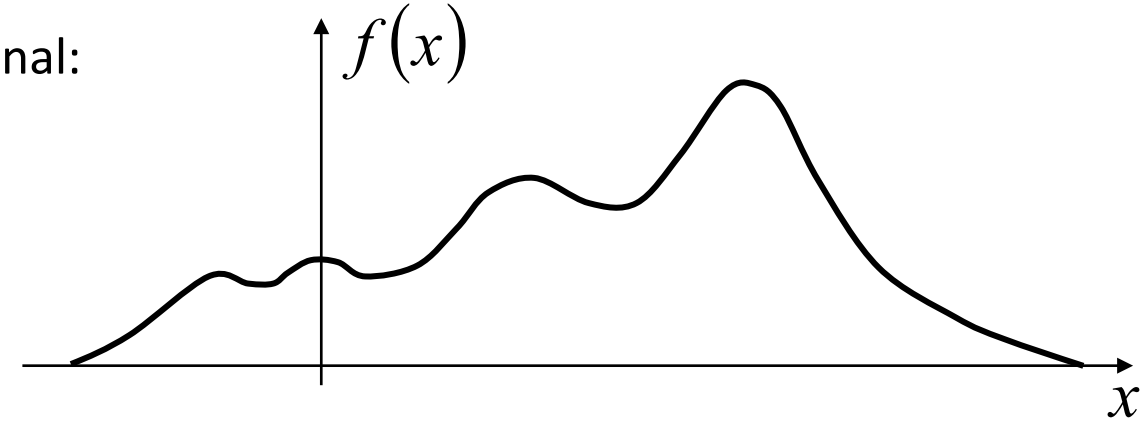
https://www.imgtec.com

Aliasing in image

https://wenzelcreative.com

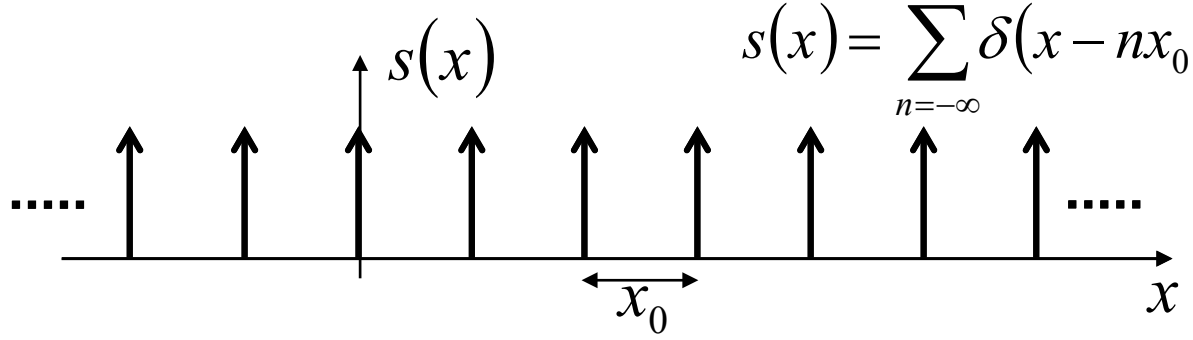
Sampling Theorem

Continuous signal:



Shah function (Impulse train):

$$s(x) = \sum_{n=-\infty}^{\infty} \delta(x - nx_0)$$



Sampled function:

$$f_s(x) = f(x)s(x) = f(x)\sum_{n=-\infty}^{\infty} \delta(x - nx_0)$$

Slide Credit: S Narasimhan

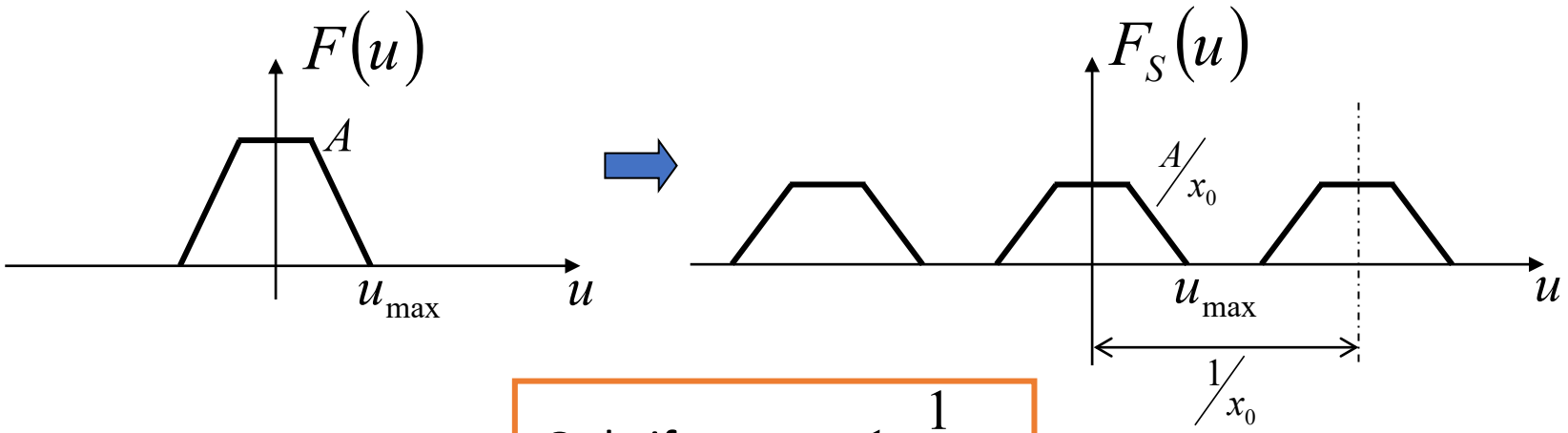
Sampling Theorem

Sampled function:

$$f_s(x) = f(x)s(x) = f(x)\sum_{n=-\infty}^{\infty}\delta(x - nx_0)$$

Sampling frequency $\frac{1}{x_0}$

$$F_S(u) = F(u) * S(u) = F(u) * \frac{1}{x_0}\sum_{n=-\infty}^{\infty}\delta\left(u - \frac{n}{x_0}\right)$$

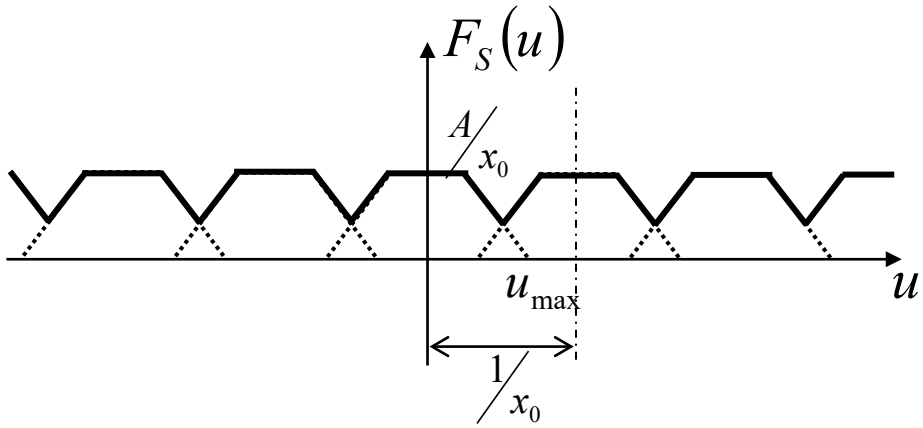


Only if $u_{\max} \le \frac{1}{2x_0}$

Slide Credit: S Narasimhan

Nyquist Theorem

If $u_{\max} > \frac{1}{2x_0}$



Aliasing

When can we recover $F(u)$ from $F_S(u)$?

Only if $u_{\max} \leq \frac{1}{2x_0}$ (Nyquist Frequency)

We can use

$$C(u) = \begin{cases} x_0 & |u| < 1/2x_0 \\ 0 & \text{otherwise} \end{cases}$$

Then $F(u) = F_S(u)C(u)$ and $f(x) = \text{IFT}[F(u)]$

Sampling frequency must be greater than $2u_{\max}$

Slide Credit: S Narasimhan

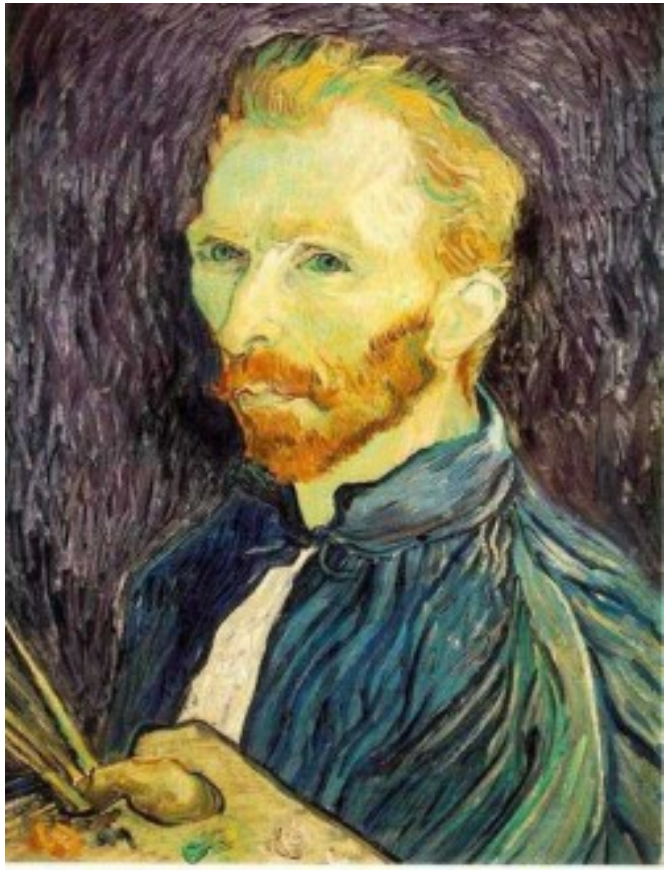
Anti-aliasing

- How to avoid aliasing?

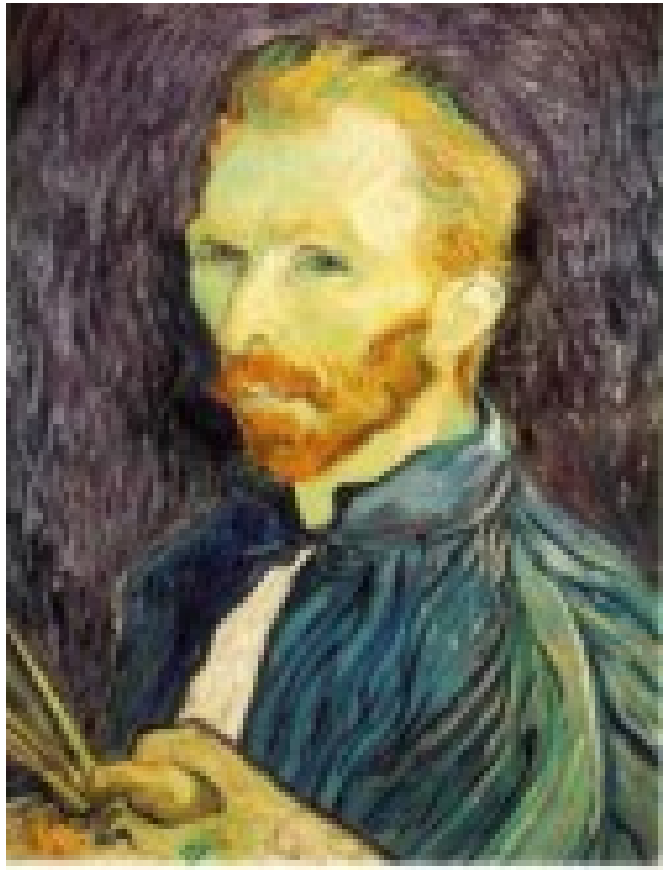
- Sampling more
  - For general image, need to sample at least twice the rate of the highest frequency component

- Get rid of all frequency that are greater than half the new sampling frequency
  - To subsample, remove high frequency components
  - Apply Gaussian low pass filter.
  - Will lose detail.
  - Better than aliasing

Sub-Sampling with Gaussian Pre-Filtering



Gaussian 1/2

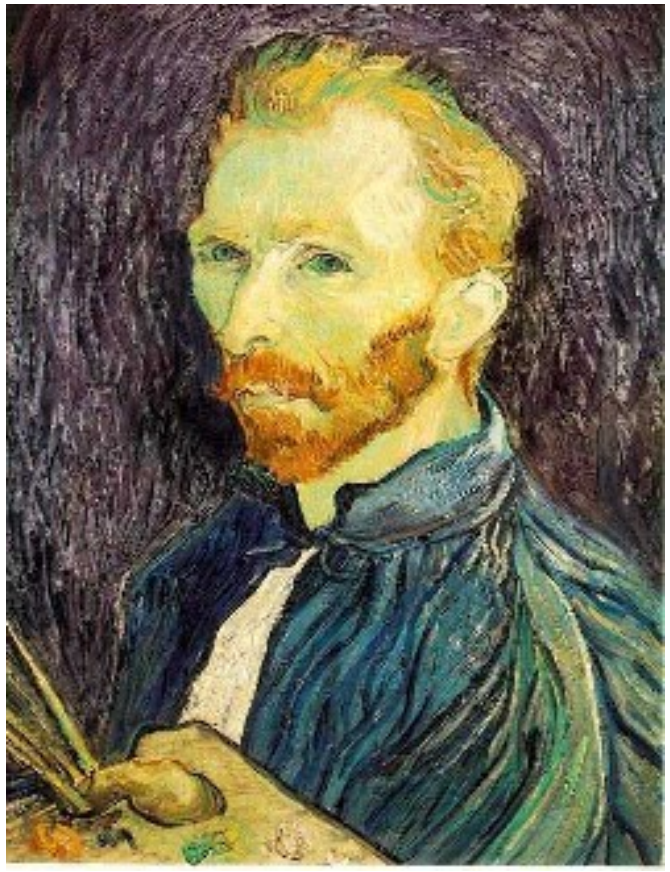


G 1/4

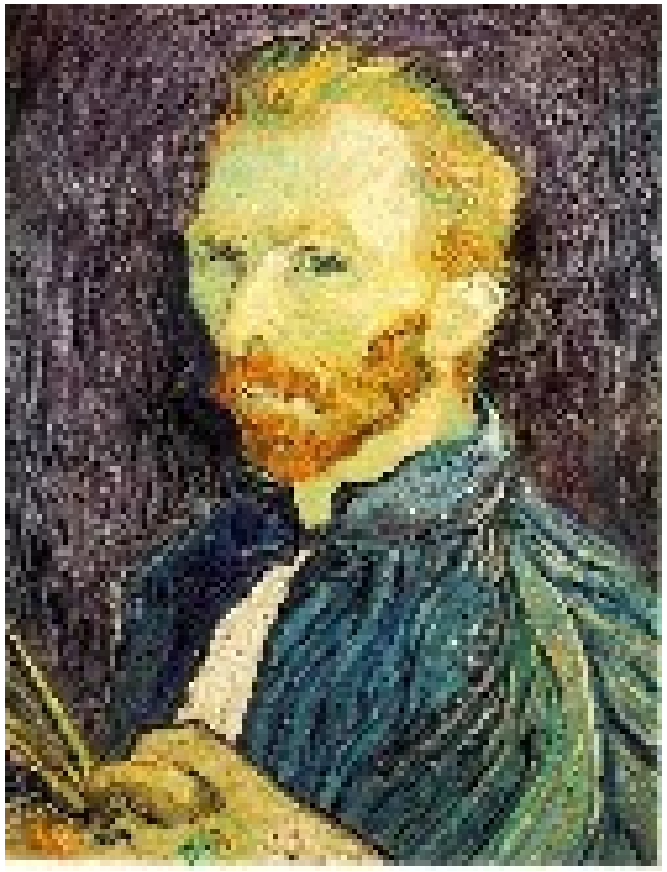
G 1/8

Slide by Steve Seitz

Subsampling without Gaussian pre-filtering



1/2



1/4 (2x zoom)

1/8 (4x zoom)

Slide by Steve Seitz

Summary

- Image can be transformed to frequency domain by Fourier Transform
- Convolutions in spatial domain is equivalent to multiplication in frequency domain
- We can apply filters effectively in frequency domain.

Today's Agenda

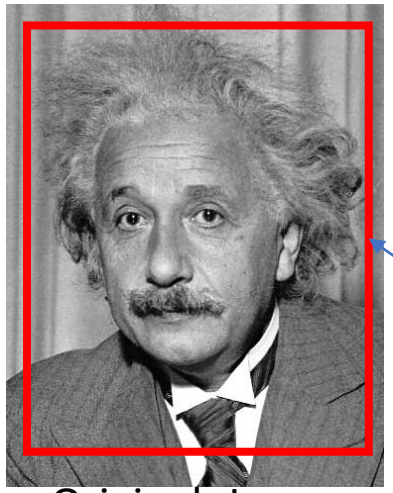
- Image as functions
- Filters in Spatial domain
- Filters in Frequency Domain
- Template Matching
- Image Pyramid

Templates and Image Pyramid

- Filtering is a way to match a template to the image
- Detection by template matching
- Coarse-to-fine registration

What is Template Matching?

- Basic Idea
  - Select a pattern (template) you wish to match
  - Move the pattern over the search area on the image
  - Measure the different between the template and sub-image at the different position.
  - Record the position with highest similarity with the template



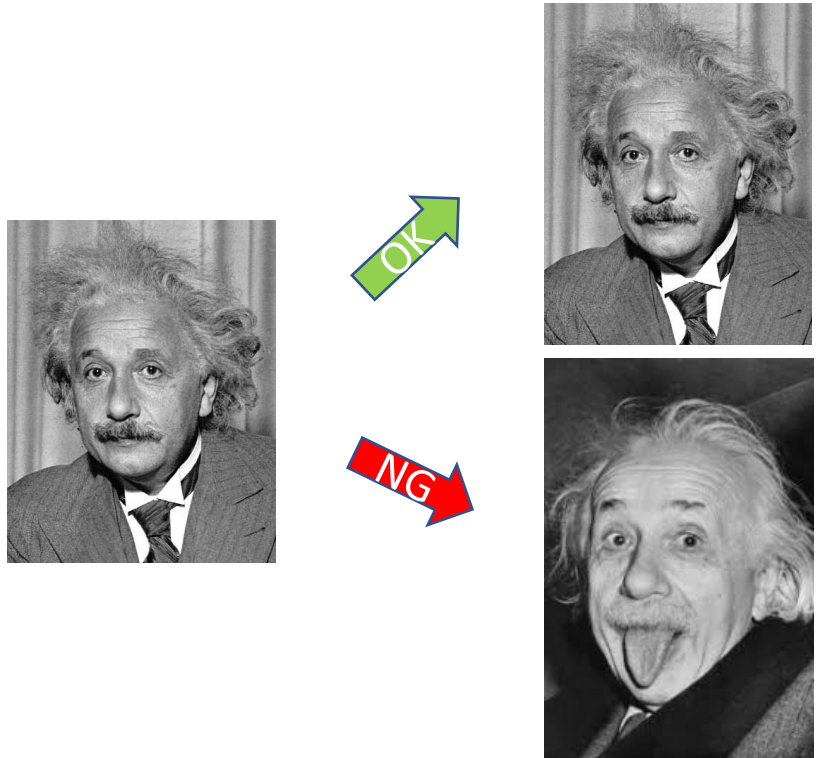
Original -Image

Sub-Image

Template

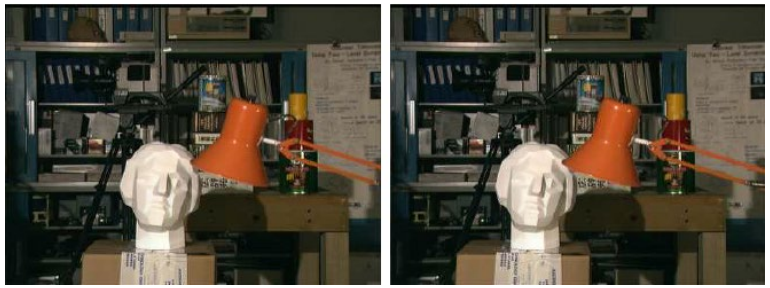
Why Template Matching?

- How to tell if 2 images are the same?
  - Pixel by pixel comparison?
    - Only applicable if and only if the two pictures are taken from the same angle, some lighting condition and etc…

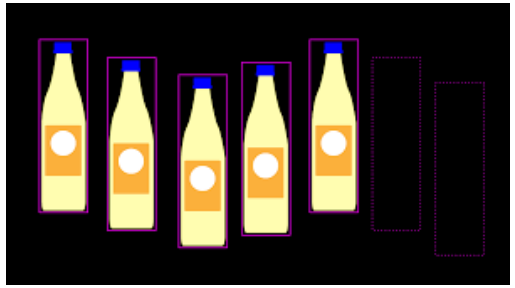


Template Matching

- Better approach: Template matching
  - Identify similar sub-images (template) within 2 images.
- Applications
  - Stereo Image Matching
  - Pattern Searching
  - Object Tracking



Stereo Image Pairs



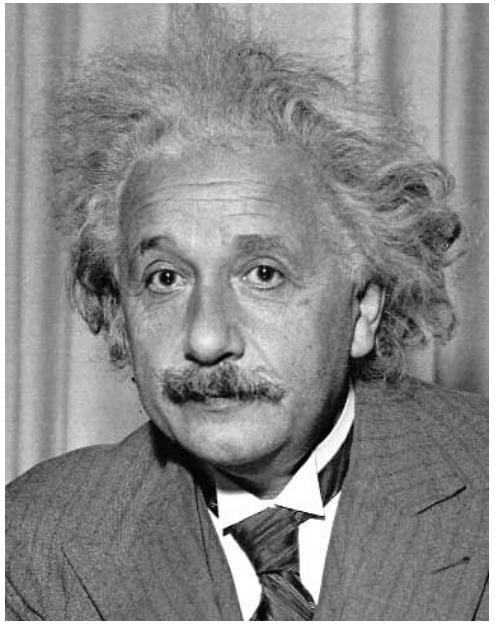
Patterns Searching

Object Tracking

How Template Matching is done?

- Let f be the template with size l,k
- Swipe the template on image pixel (m,n) in the sub-image.
- Calculate the distance function for position (m,n)
- What kind of distance function we have?

Template f



Original Image g

Distance between Image Patterns

- Different methods are available to measure the distance between the reference image(template) and corresponding sub image I

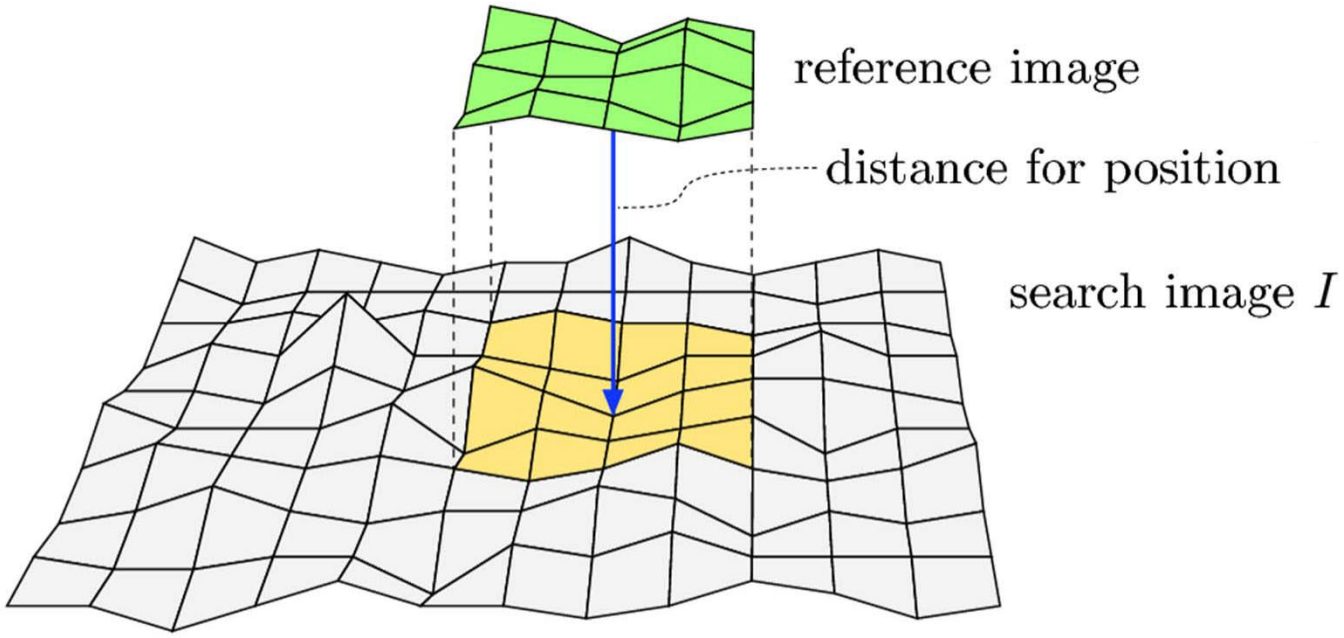
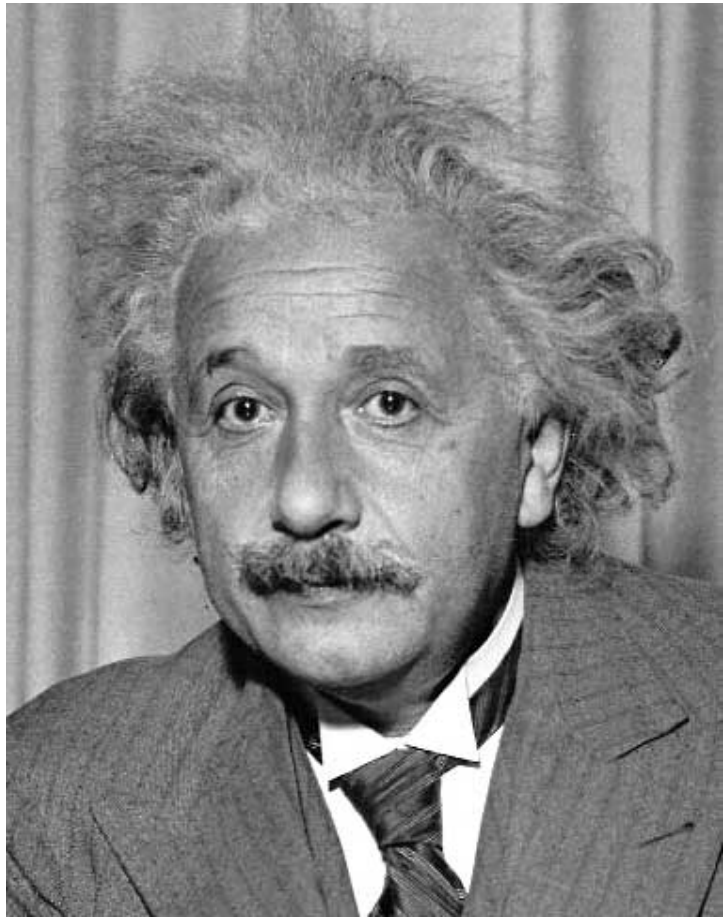


Image Credit: E. Agu

Template Matching

- Goal: To find  in the image

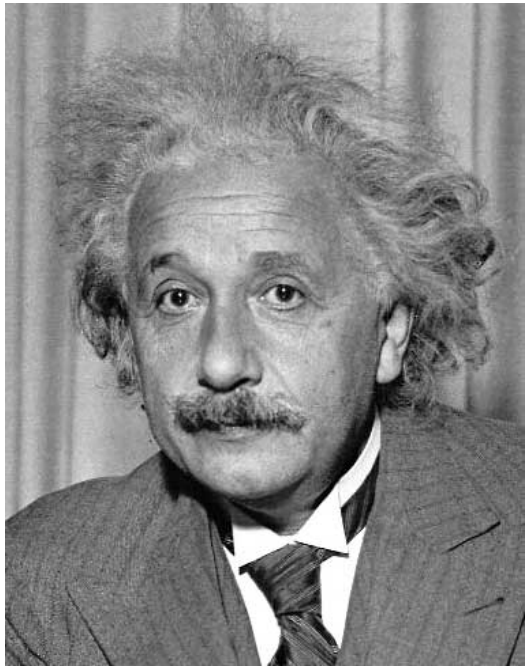
- Methods (distance function)
  - Correlation
  - Zero-mean correlation
  - Sum of Square Difference
  - Normalized Cross-Correlation



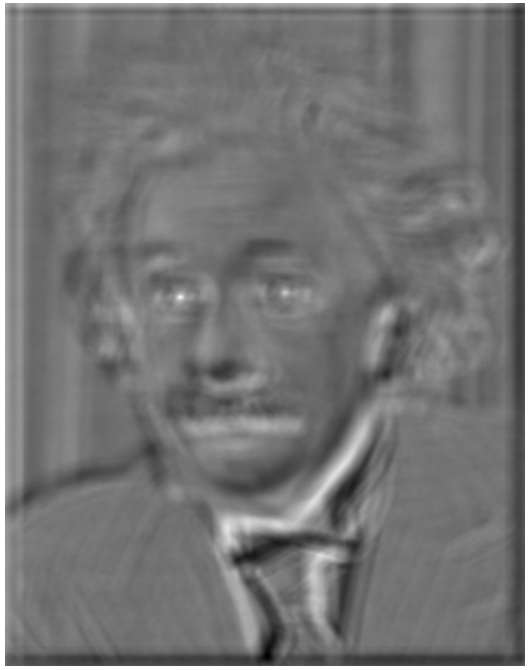
Matching with filters

- Goal: To find  in the image
- Method 1: filter the image with zero-mean eye patch
  - $h[m,n] = \sum_{k,l}(f[k,l] - \bar{f})(g[m+k, n+l])$

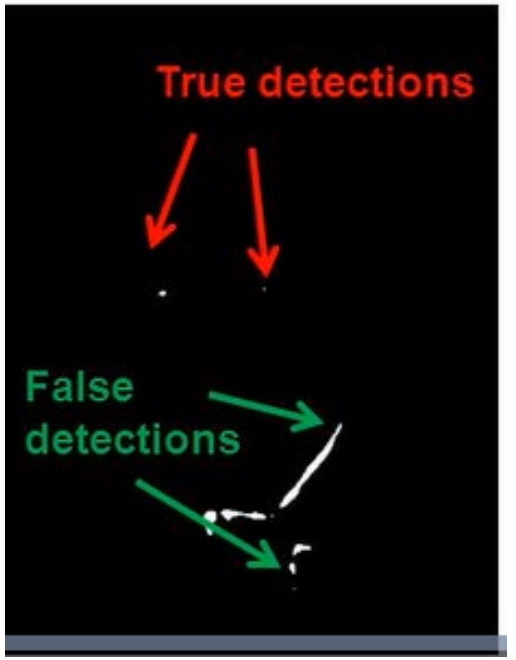
Mean of f



Input



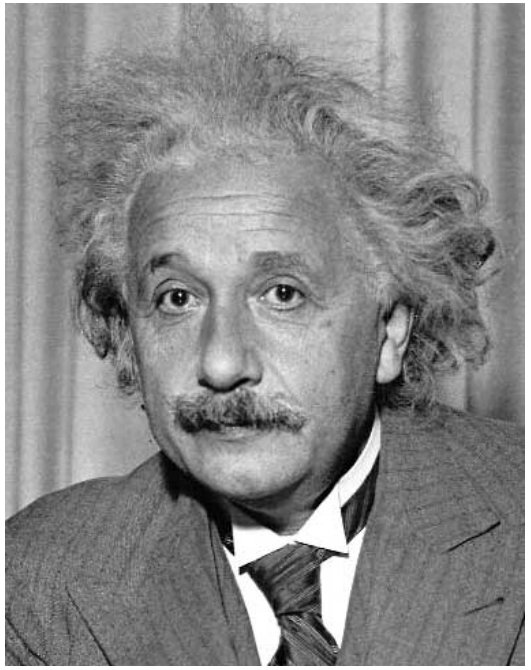
Filtered Image



Thresholded Image

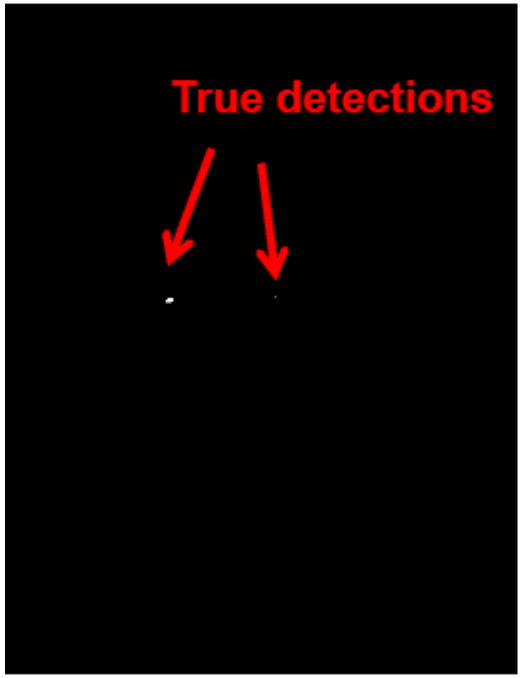
Matching with filters

- Goal: To find  in the image
- Method 2: SSD
  - $h[m,n] = \sum_{k,l}(g[k,l] - f[m+k,n+l])^2$



Input

Filtered Image

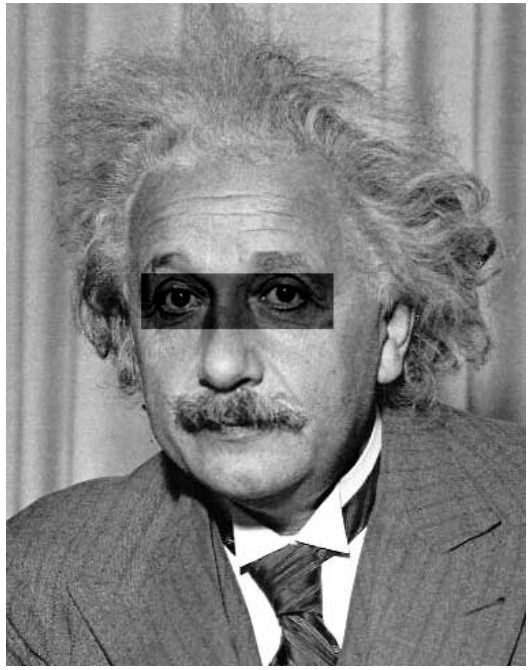


Thresholded Image

Matching with filters

- Goal: To find  in the image
- Method 2: SSD
  - $h[m,n] = \sum_{k,l}(g[k,l] - f[m+k,n+l])^2$

What's the potential downside of SSD?



Input

Filtered Image

Matching with filters

- Goal: To find  in the image
- Method 3: Normalized cross-correlation

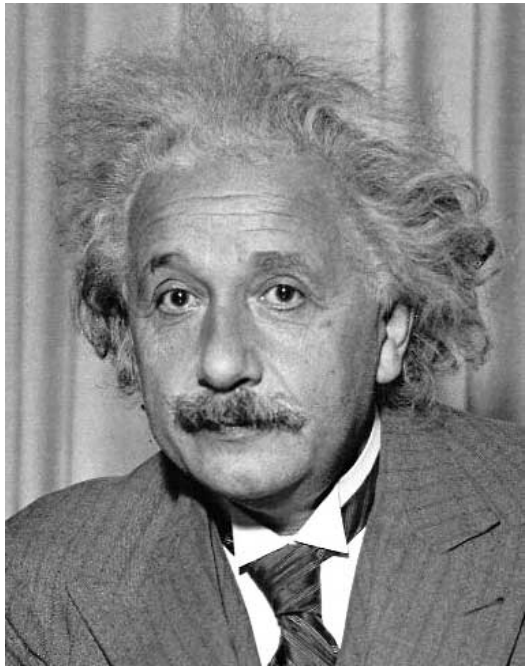
Mean template

Mean Image Patch

- $$h[m,n] = \frac{\sum_{k,l}(g[k,l]-\bar{g})(f[m-k,n-l]-\overline{f_{m,n}})}{(\sum_{k,l}(g[k,l]-\bar{g})^2\sum_{k,l}(f[m-k,n-l]-\overline{f_{m,n}})^2)^{0.5}}$$

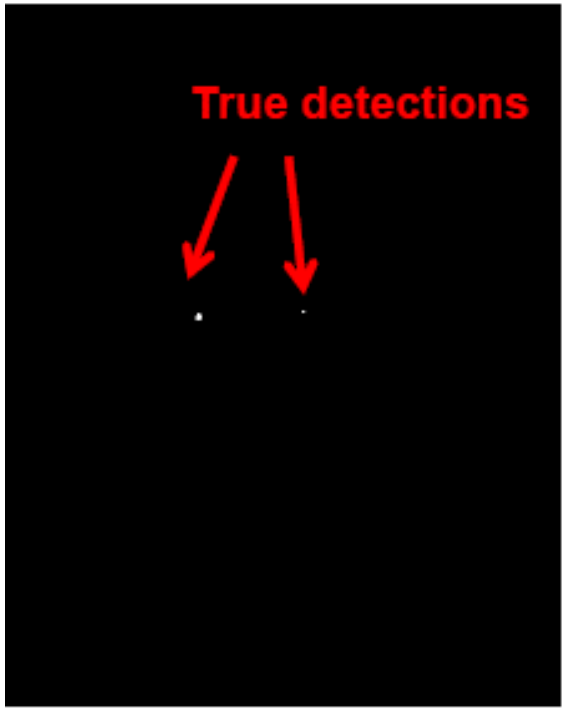
Matching with filters

- Goal: To find  in the image
- Method 3: Normalized cross-correlation



Input

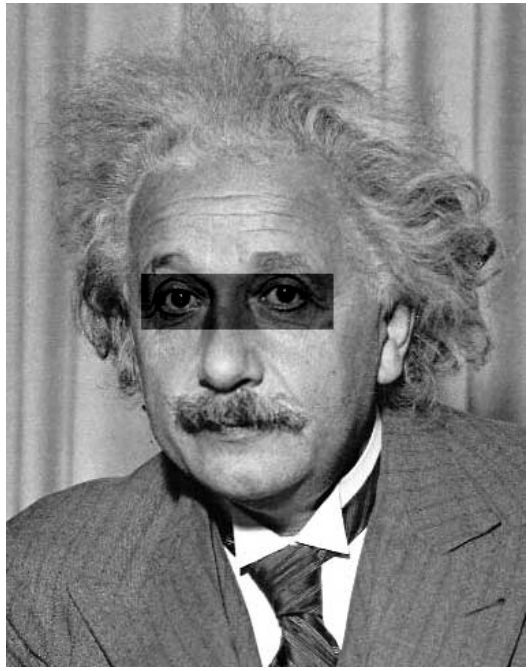
Normalized X-Correlation



Thresholded Image

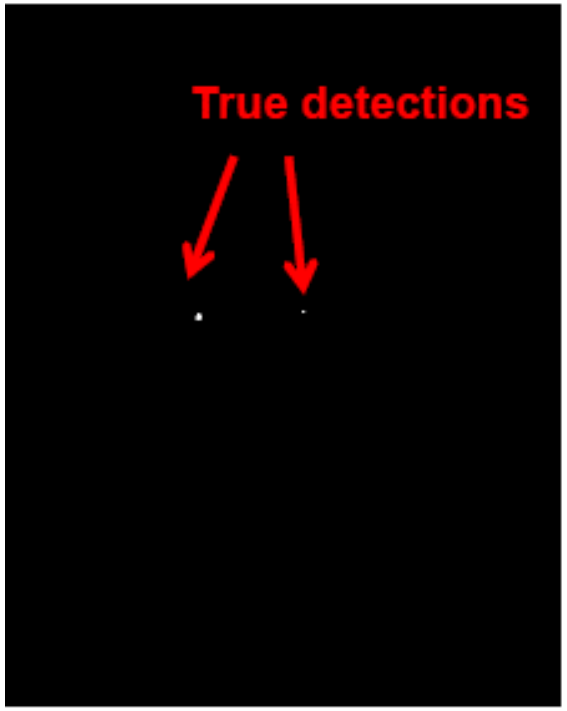
Matching with filters

- Goal: To find  in the image
- Method 3: Normalized cross-correlation



Input

Normalized X-Correlation



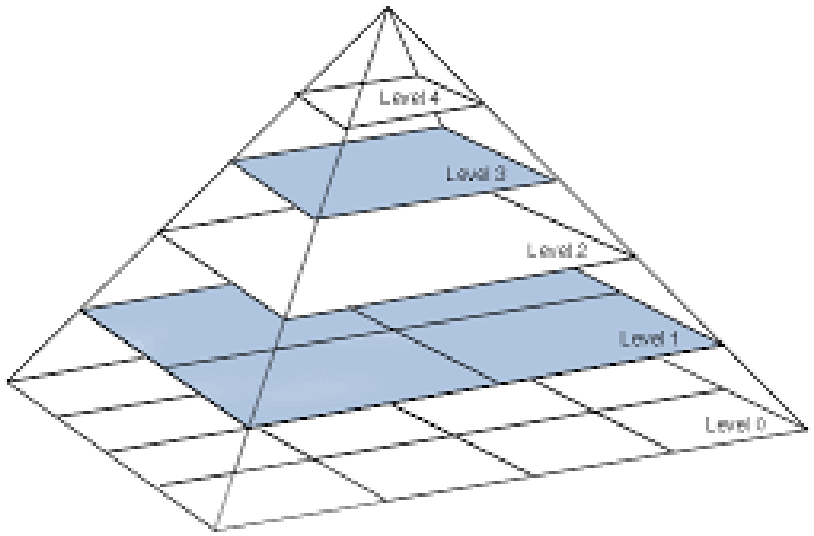
Thresholded Image

What is the best method to use?

- Depends
  - SSD: faster, sensitive to overall intensity
  - Normalized cross-correlation: Slower, invariant to local average intensity and contrast
  - But Neither these are representative of modern recognition

What if we want to find larger or smaller eyes?

Answer – Image Pyramid

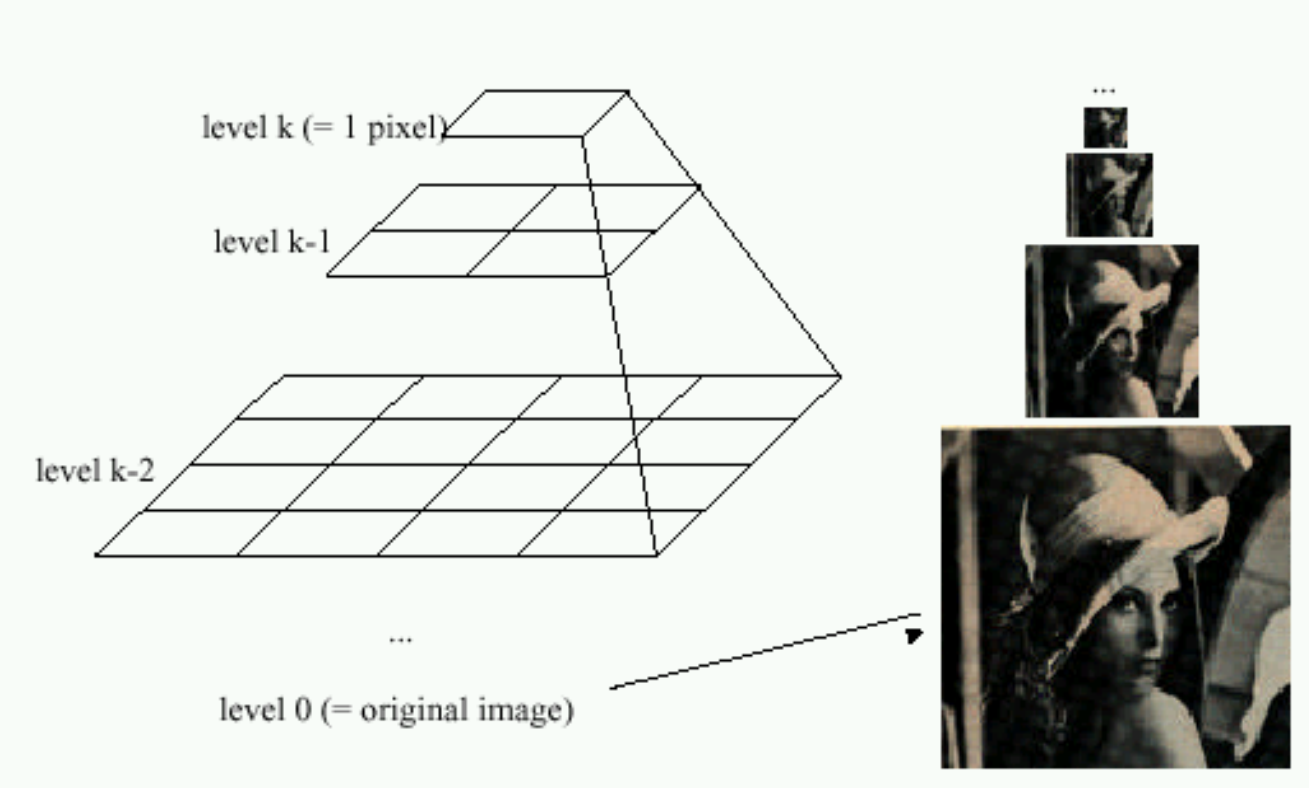


Today’s Agenda

- Image as functions
- Filters in Spatial domain
- Filters in Frequency Domain
- Template Matching
- Image Pyramid Representation

Gaussian pyramids [Burt and Adelson, 1983]

- Idea: Represent $N \times N$ image as a "Pyramid" of $1 \times 1, 2 \times 2, \ldots, N \times N$



- In computer graphics, a *mip map* [Williams, 1983]

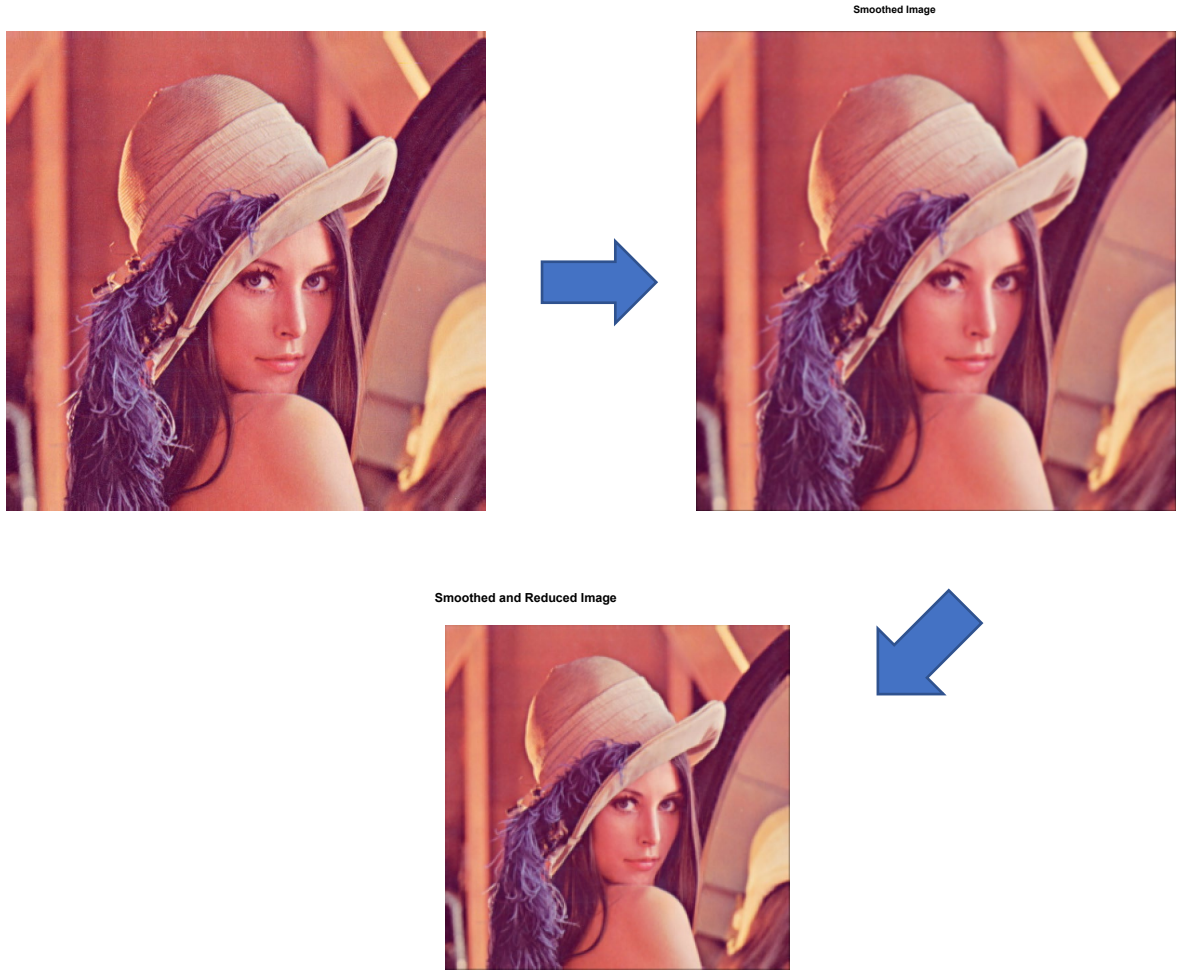
Source: S. Seitz

The Gaussian Pyramid

- The representation is based on ***2 basic operations***

- ***Smoothing:***
- Smooth the image with smoothing filter

- ***Down sampling:***
- Reduce the image size by a factor of 2 after each smoothing



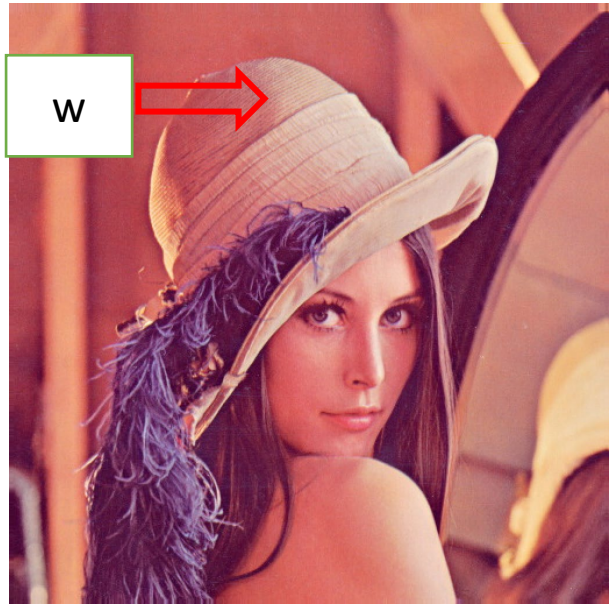
Operation 1: Smooth Image

Original Photos g_0

$$g_1 = w * g_0$$

5x5 filter

g_0



Slide the box filter over the image

$$g_1(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) g_0(2i+m, 2j+n)$$

Operation 1: Smooth Image

Original Photos g_0

$$g_2 = w * g_1$$
$$= (w * w) * g_0$$

Slide the box filter over the image

$$g_2(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) g_1(2i+m, 2j+n)$$

Operation 1: Smooth Image

Original Photos g_0

$$g_3 = w * g_2$$
$$= (w * w * w) * g_0$$

Slide the box filter over the image

$$g_3(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) g_2(2i+m, 2j+n)$$

Operation 1: Smooth Image

Original Photos g_0

$$g_3 = w * g_2$$
$$= (w * w * w) * g_0$$

g_3
Smoothed Image

Slide the box filter over the image

$$g_4(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) g_3(2i+m, 2j+n)$$

How to select the filter?

1. w always has 5 elements
2. w is symmetric about centre
3. $w = \begin{bmatrix} c \\ b \\ a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \frac{1}{4} - \frac{a}{2} \\ \frac{1}{4} \\ a \\ \frac{1}{4} \\ \frac{1}{4} - \frac{a}{2} \end{bmatrix}$
4. Sum of all elements equal to one

$$\sum_{m=-2}^{2} w(m) = 1$$

4. Equal contribution

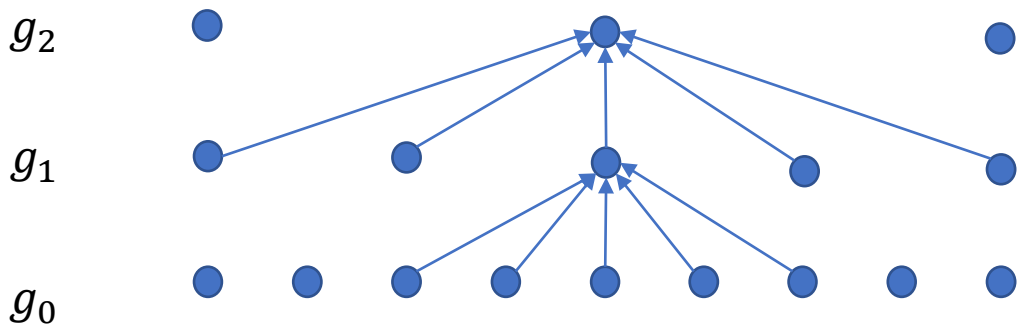
$$a + 2c = 2b = 1/2$$

Gaussian Pyramid (In –D)

- $g_l(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) g_{l-1}(2i+m, 2j+n)$

$level\ l$

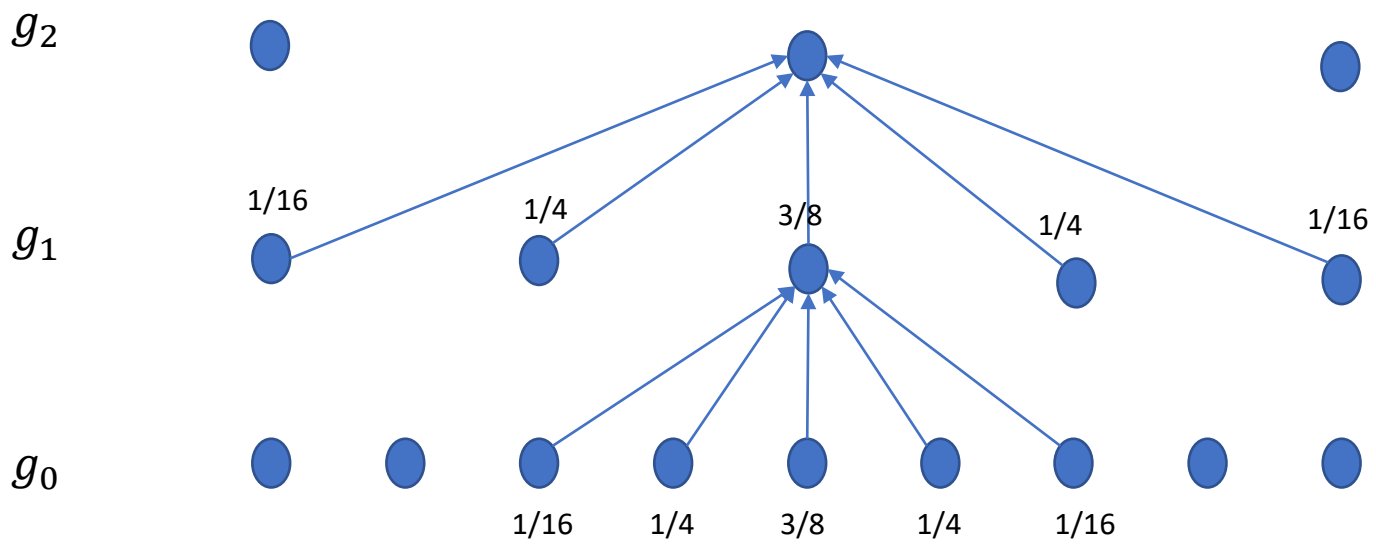
- Reduce to 1D
  - $g_l(i) = \sum_{m=-2}^{2} w(m) g_{l-1}(2i+m)$
  - $w(m)$ is convolution mask



$g_l = Reduce[g_{l-1}]$

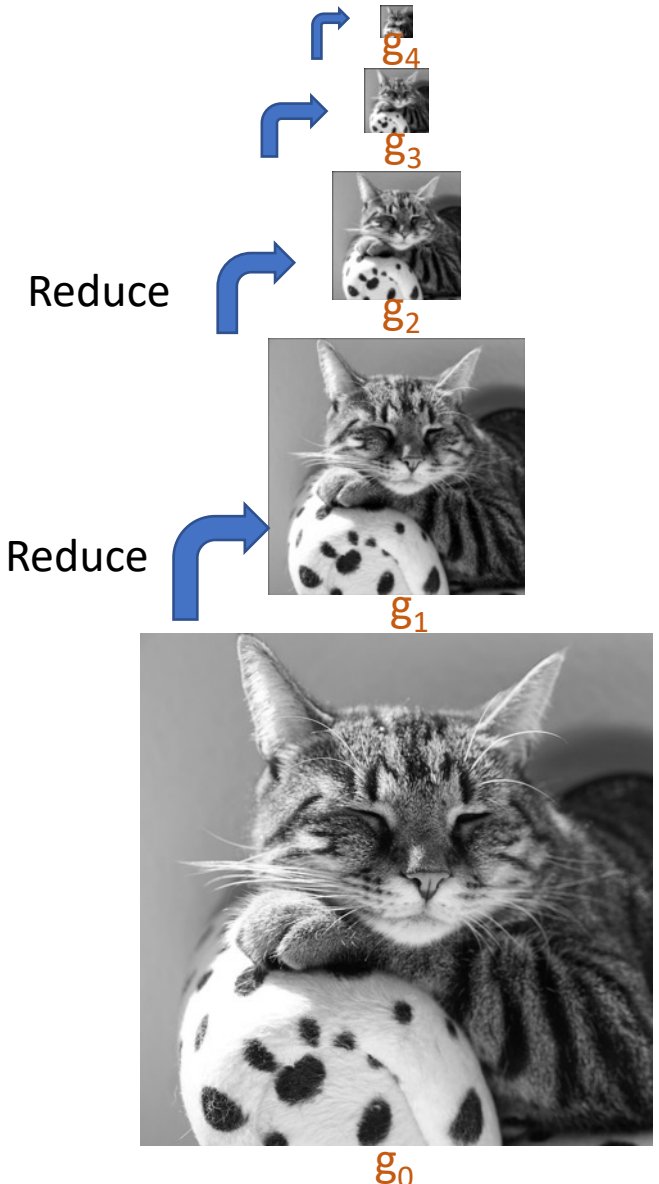
Mask

- Usually we select $a \in [0.3, 0.6]$

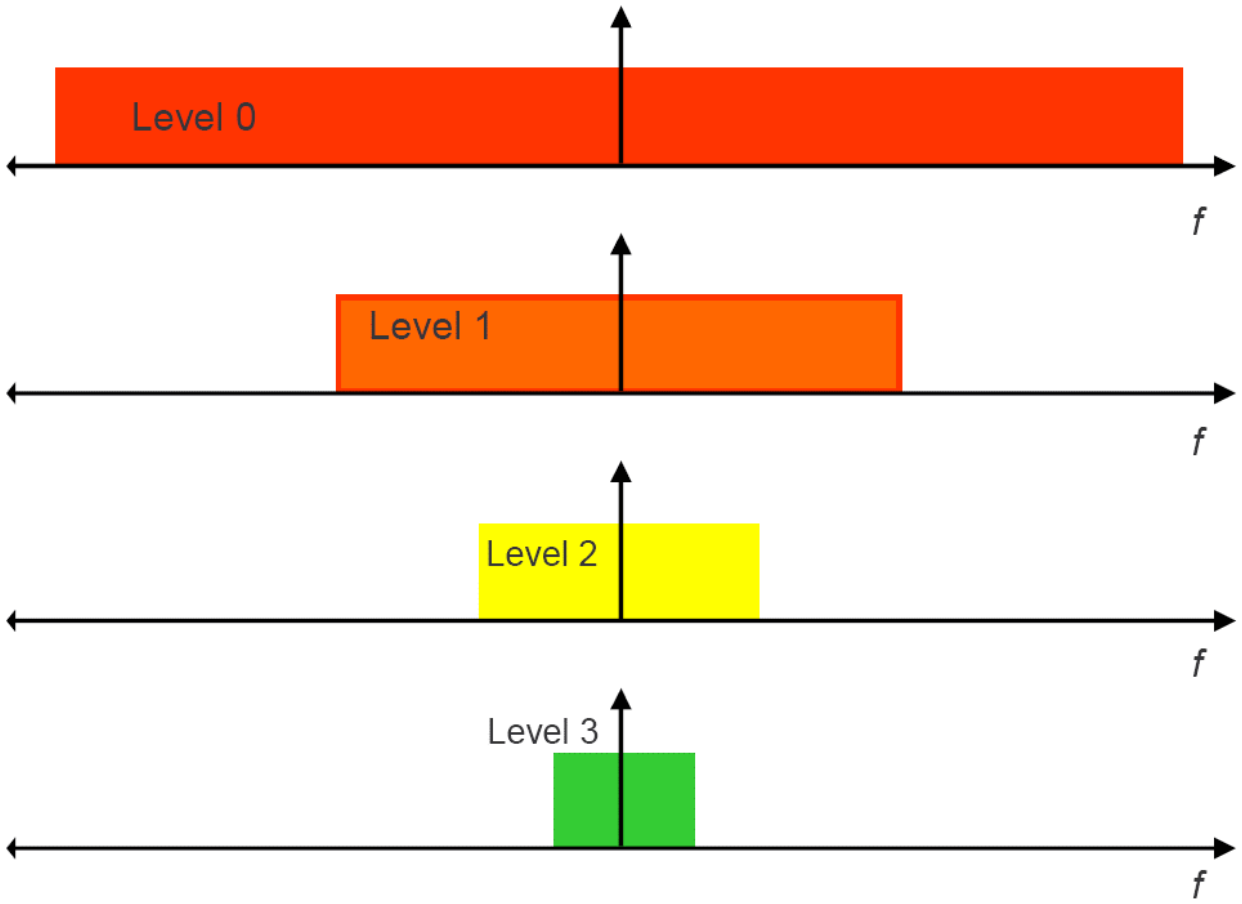


$$w = \begin{bmatrix} c \\ b \\ a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \frac{1}{4} - \frac{a}{2} \\ \frac{1}{4} \\ a \\ \frac{1}{4} \\ \frac{1}{4} - \frac{a}{2} \end{bmatrix}$$

Gaussian pyramid

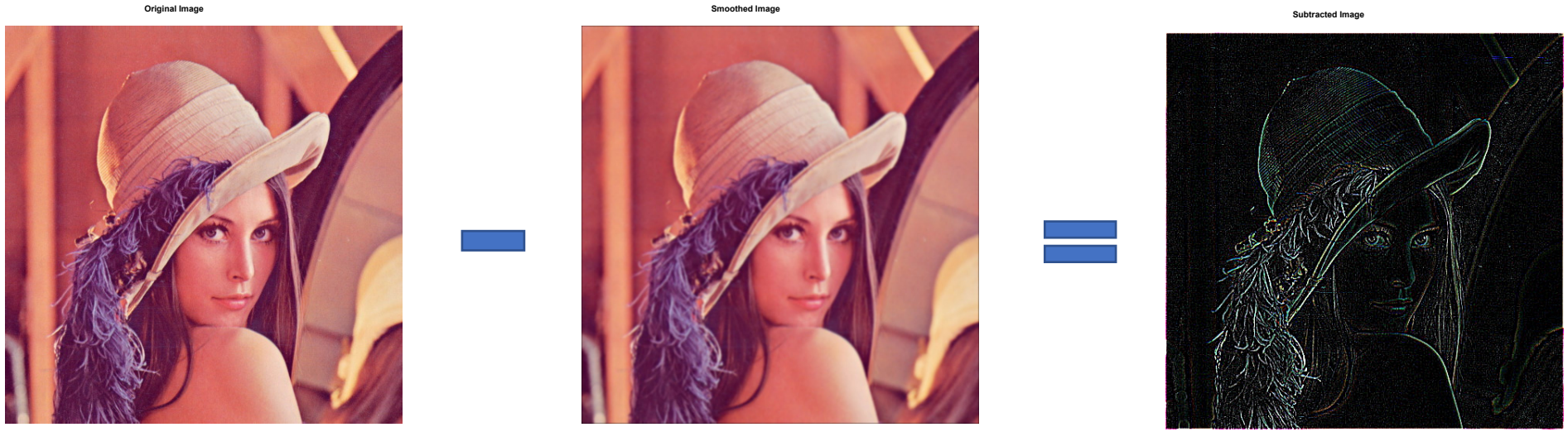


Gaussian pyramid frequency composition

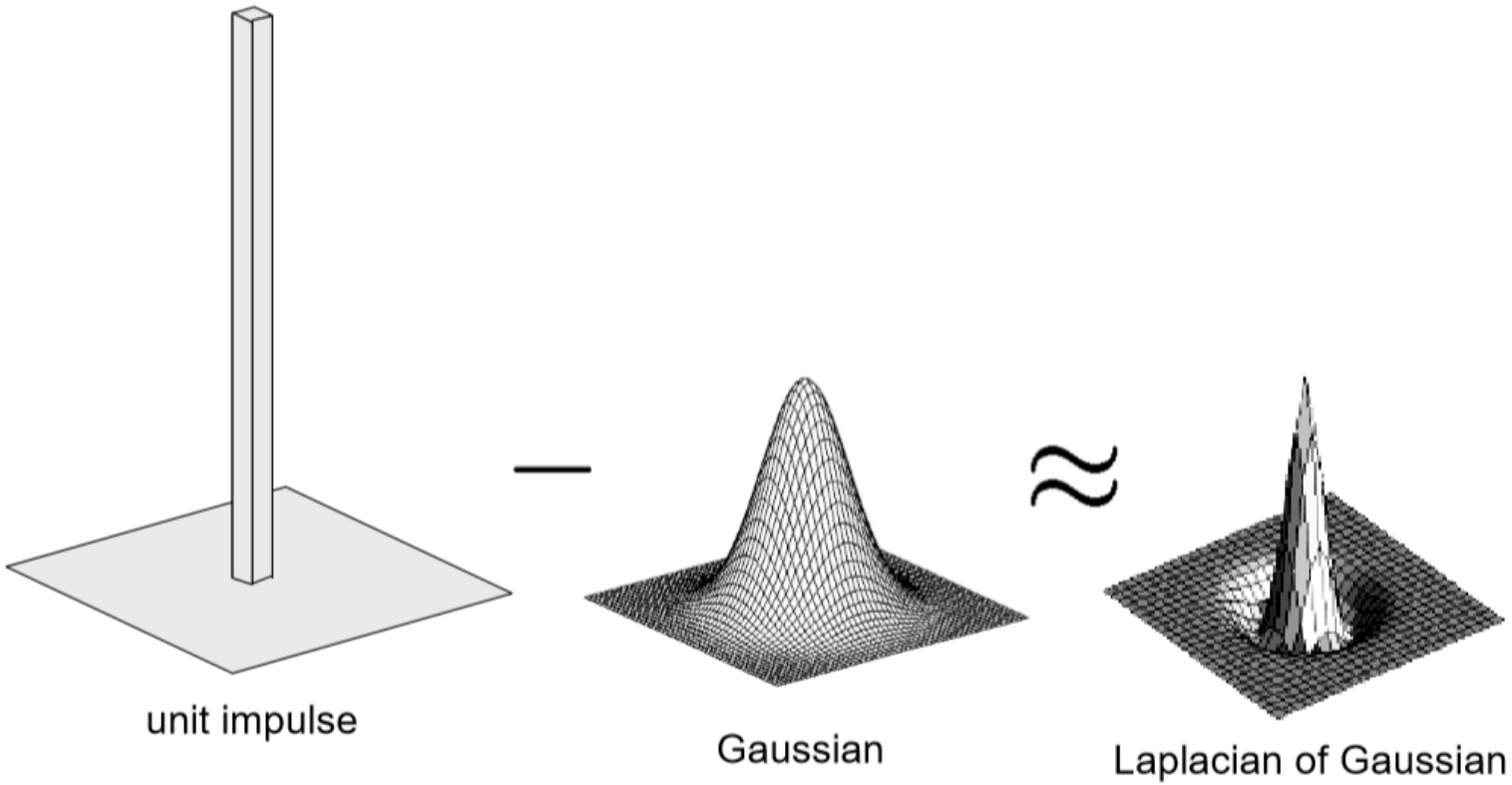


Slide adopted from S. Narasimhan

What is smoothing take away?



Laplacian of Gaussian (LOG)



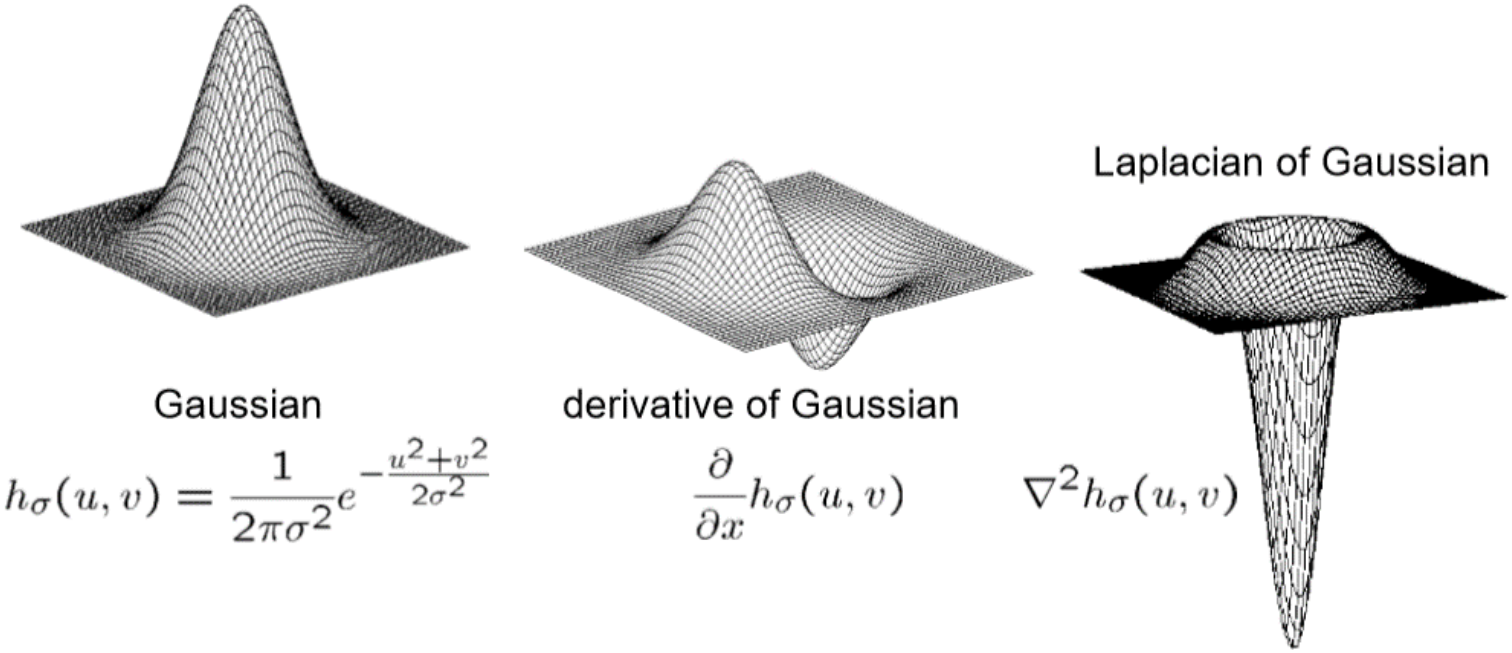
Source: Lazebnik

Laplacian of Gaussian (LOG)

$$\nabla^2 G_\sigma(x,y) = \frac{\partial^2 G_\sigma(x,y)}{\partial x^2} + \frac{\partial^2 G_\sigma(x,y)}{\partial y^2}$$

- Second derivative of Gaussian
- Output of convolution is Laplacian of image
- Zero-crossing corresponds to edges

LOG - 2D Edge Detection Filters



Gaussian

$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

derivative of Gaussian

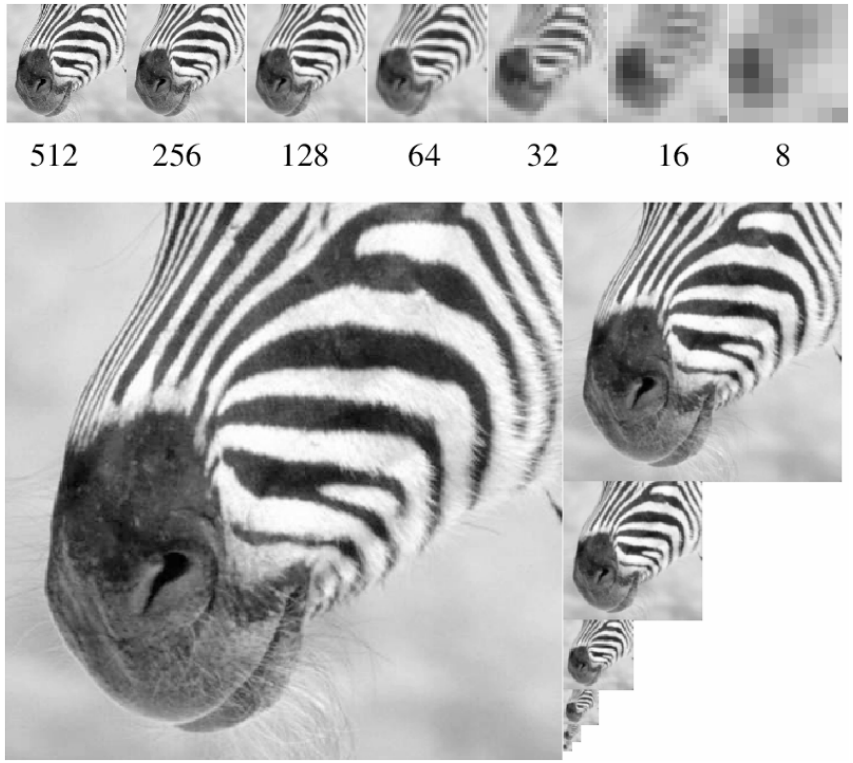
$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

$$\nabla^2 h_\sigma(u,v)$$

$$\nabla^2 G_\sigma(x,y) = \frac{\partial^2 G_\sigma(x,y)}{\partial x^2} + \frac{\partial^2 G_\sigma(x,y)}{\partial y^2}$$

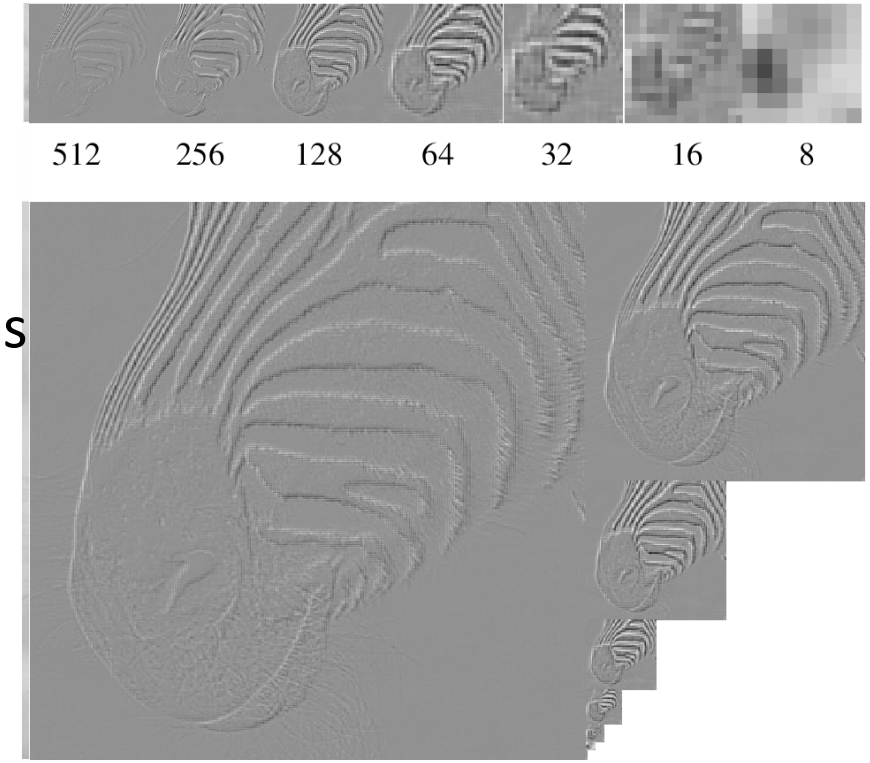
Laplacian Pyramid

- Idea: Rather than store the smoothed images, store only the difference between level g_l and g_{l+1}



Laplacian Pyramid

- Idea: Rather than store the smoothed images, store only the difference between level g_l and g_{l+1}



Similar to edge detected images
Most pixel are zero
Good for image compression

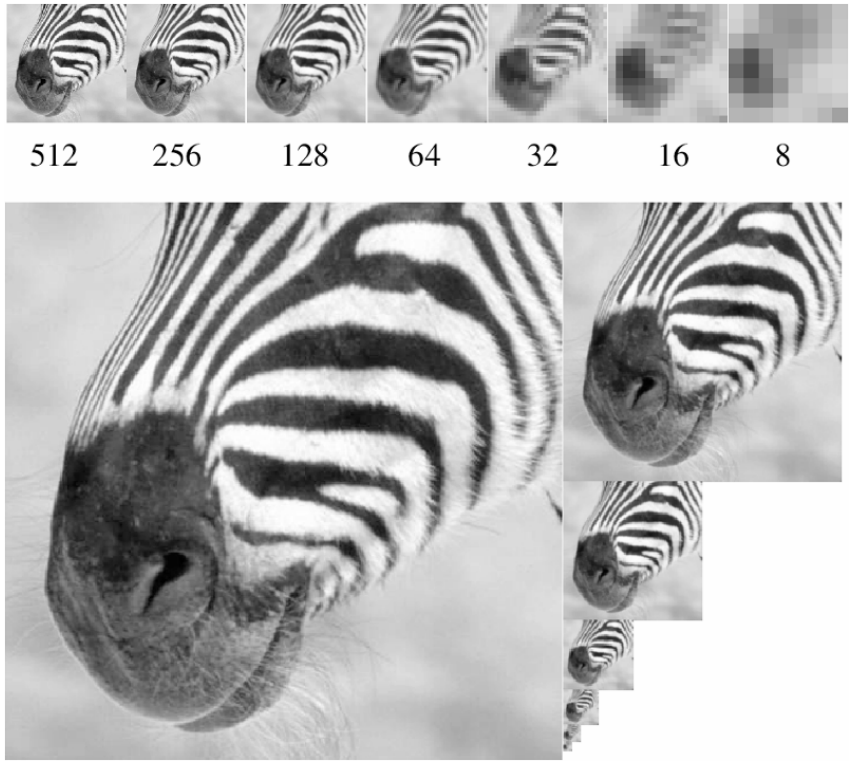
Laplacian Pyramid

- Idea: Rather than store the smoothed images, store only the difference between level g_l and g_{l+1}

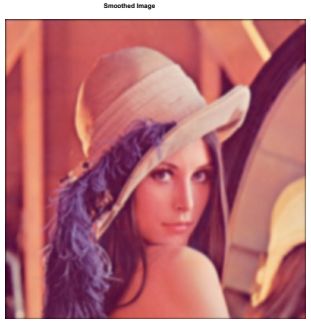
To do this, we need to compare the adjacent level of Gaussian pyramid g_l, g_{l+1}

But g_l, g_{l+1} do not have the same size!

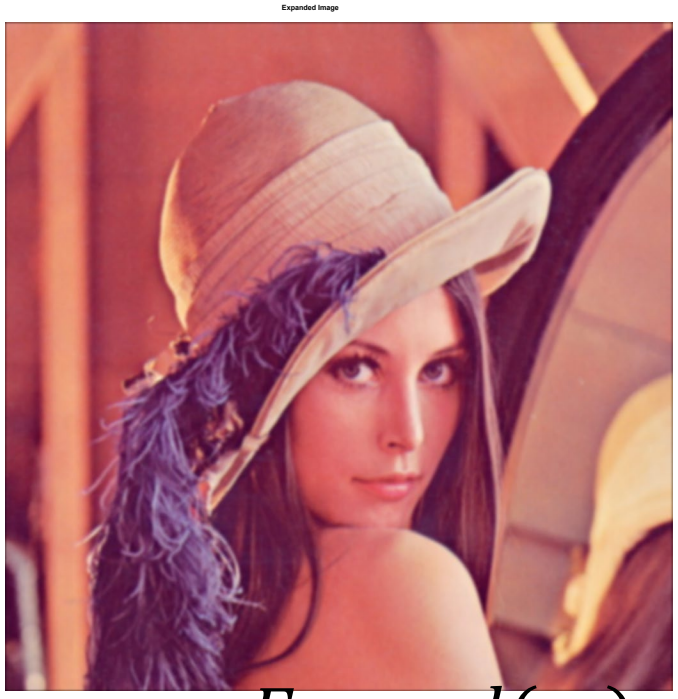
Expand g_{l+1} to make it equal size to g_l



The expand function



g_l



$Expand(g_l)$

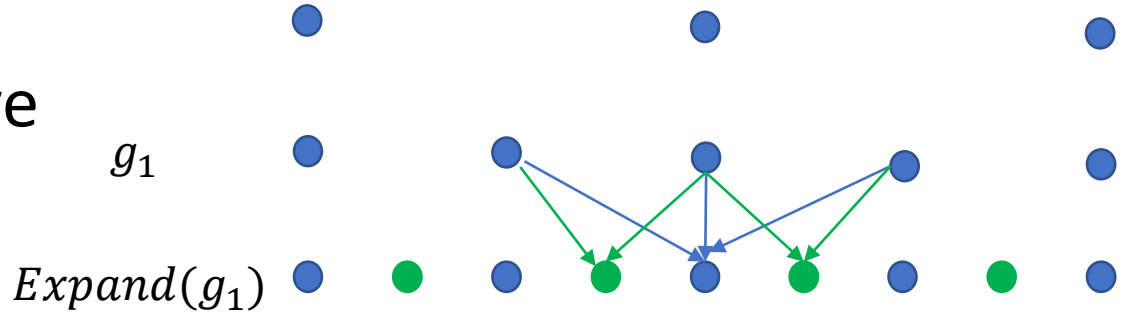
- $Expand(g_l) = 4\sum_{m=-2}^{2}\sum_{n=-2}^{2} w(m,n) \cdot g_l(\frac{i-m}{2}, \frac{j-n}{2})$

How to Expand?

- Criteria:
- *Even pixels (orange)* receive contribution from 3 pixels above them
- Total weight $(2c + a = \frac{1}{2})$
- *Odd pixels(green)* receive contribution from 2 pixels above them
- Total weight $(2b = \frac{1}{2})$

In 1D

$$Expand\big(g_l(i)\big) = 2\sum_{m=-2}^{2} w(m)\, g_l\left(\frac{i-m}{2}\right)$$

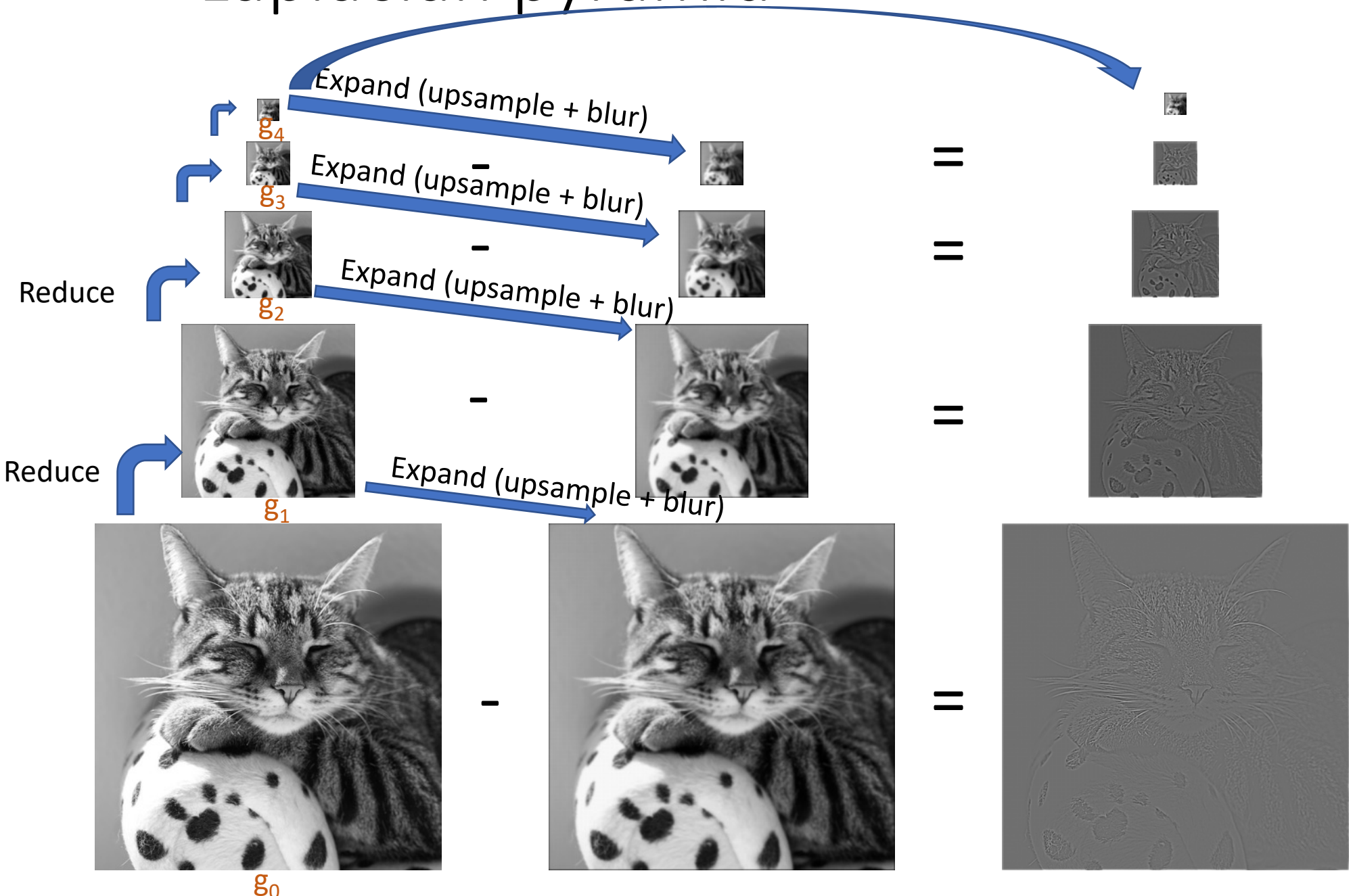


$$g_l = Expand[g_{l-1}]$$

Computing Laplacian Pyramid

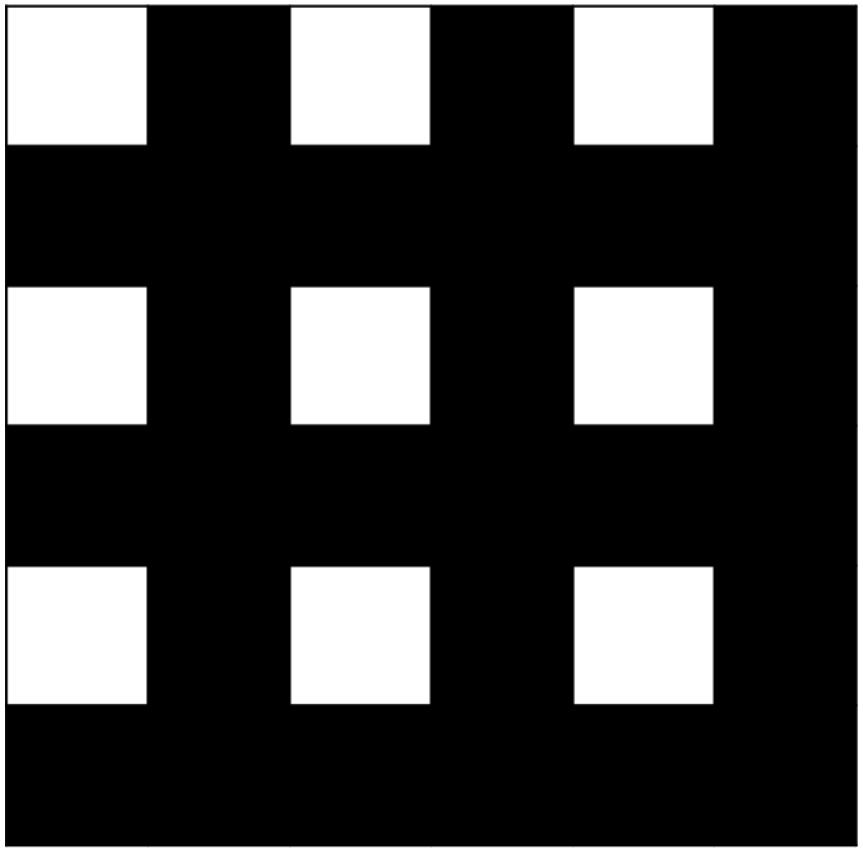
- Compute Gaussian Pyramid
  - g_1, g_2, g_3, g_4
- Compute Laplacian pyramid
  - $L_1 = g_1 - EXPAND[g_2]$
  - $L_2 = g_2 - EXPAND[g_3]$
  - $L_3 = g_3 - EXPAND[g_4]$
  - $L_4 = g_4$

Laplacian pyramid



How to Up Sampling?

- Simply add zeros in between every pixel?
- Not a good solution!

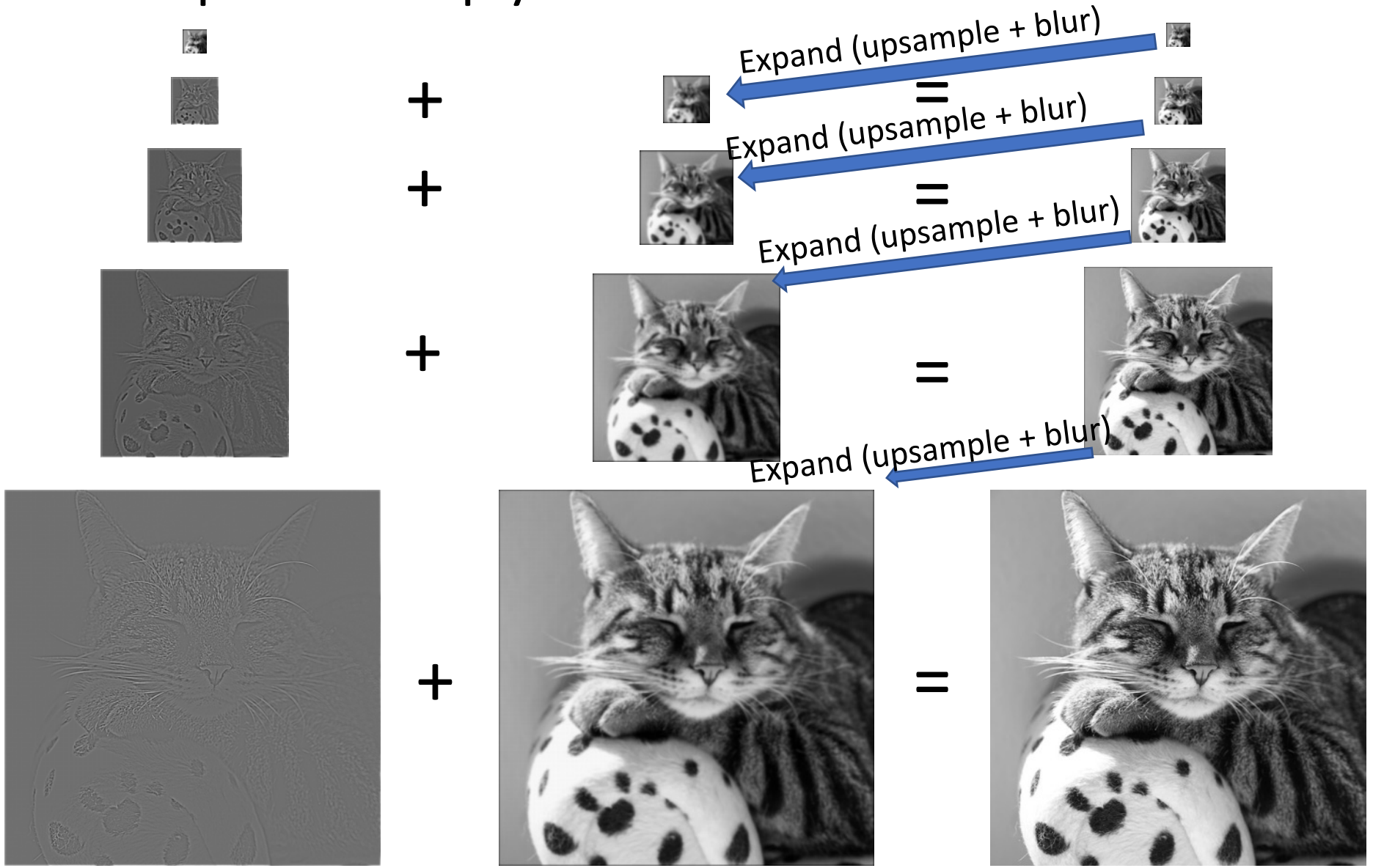


Reconstruct Image using Laplacian Pyramid

- Compute Gaussian pyramid from Laplacian pyramid
  - $g_4 = L_4$
  - $g_3 = L_3 + EXPAND[g_4]$
  - $g_2 = L_2 + EXPAND[g_3]$
  - $g_1 = L_1 + EXPAND[g_2]$

  - g_1 is reconstructed image

Reconstructing the image from a Laplacian pyramid



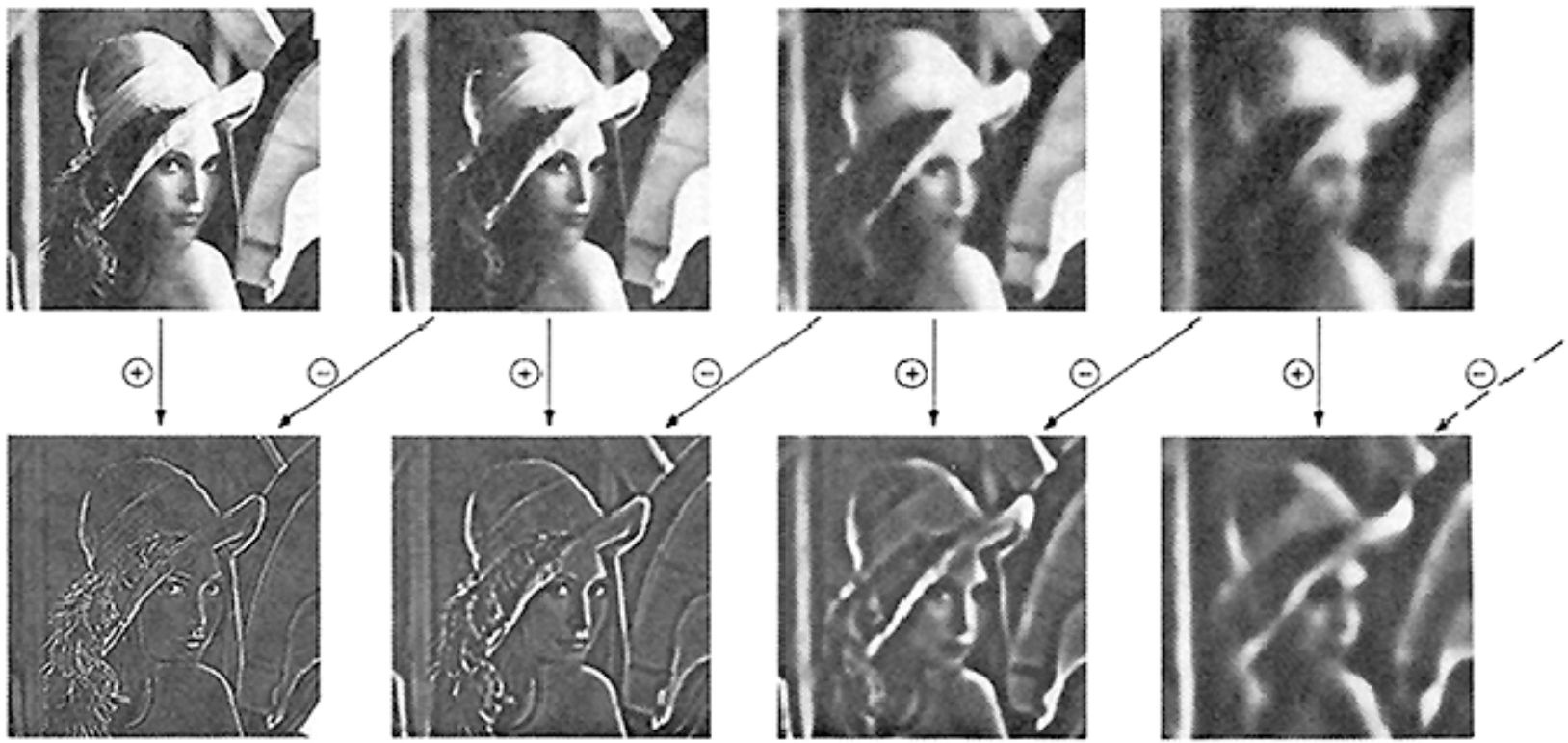
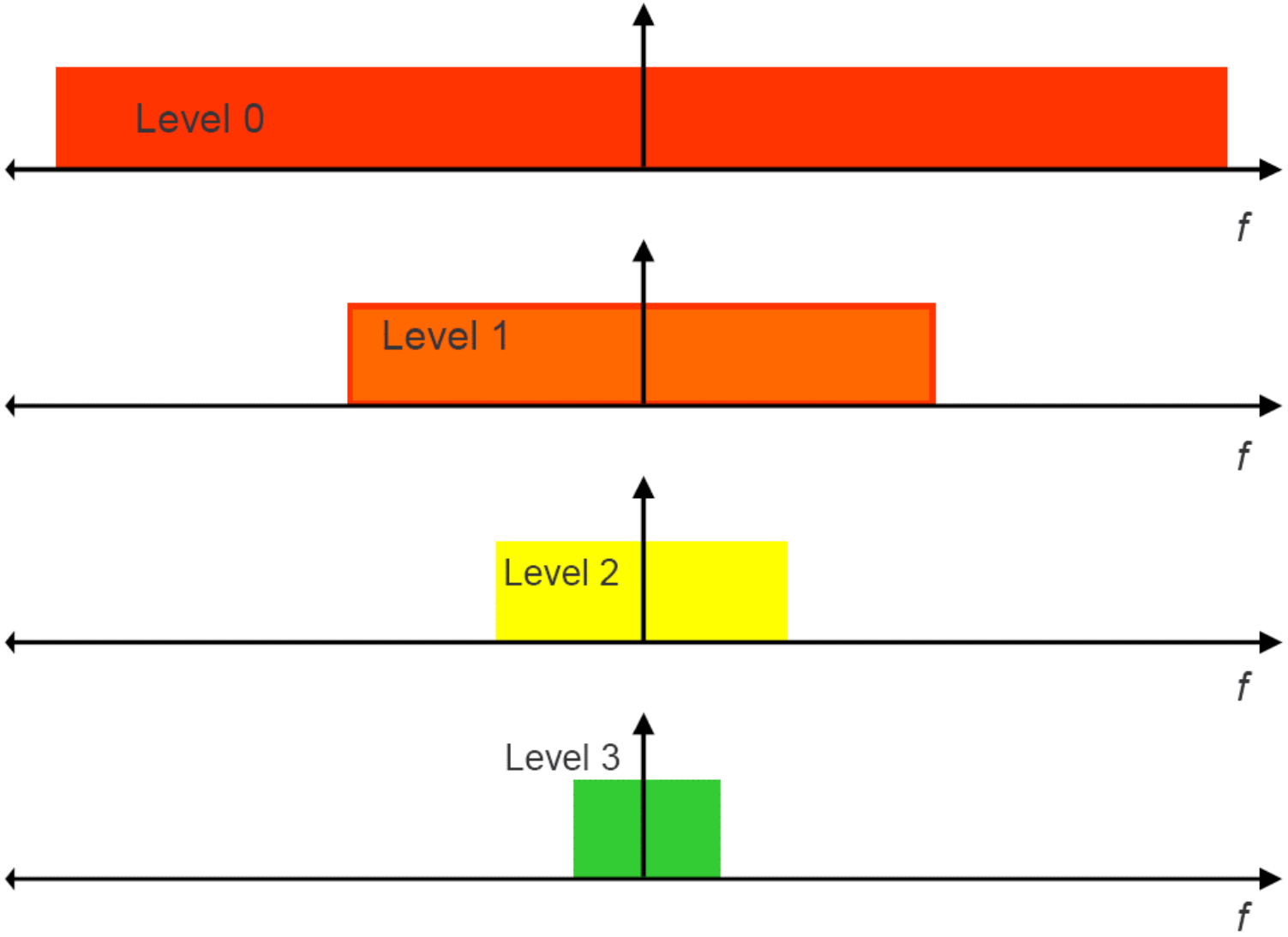


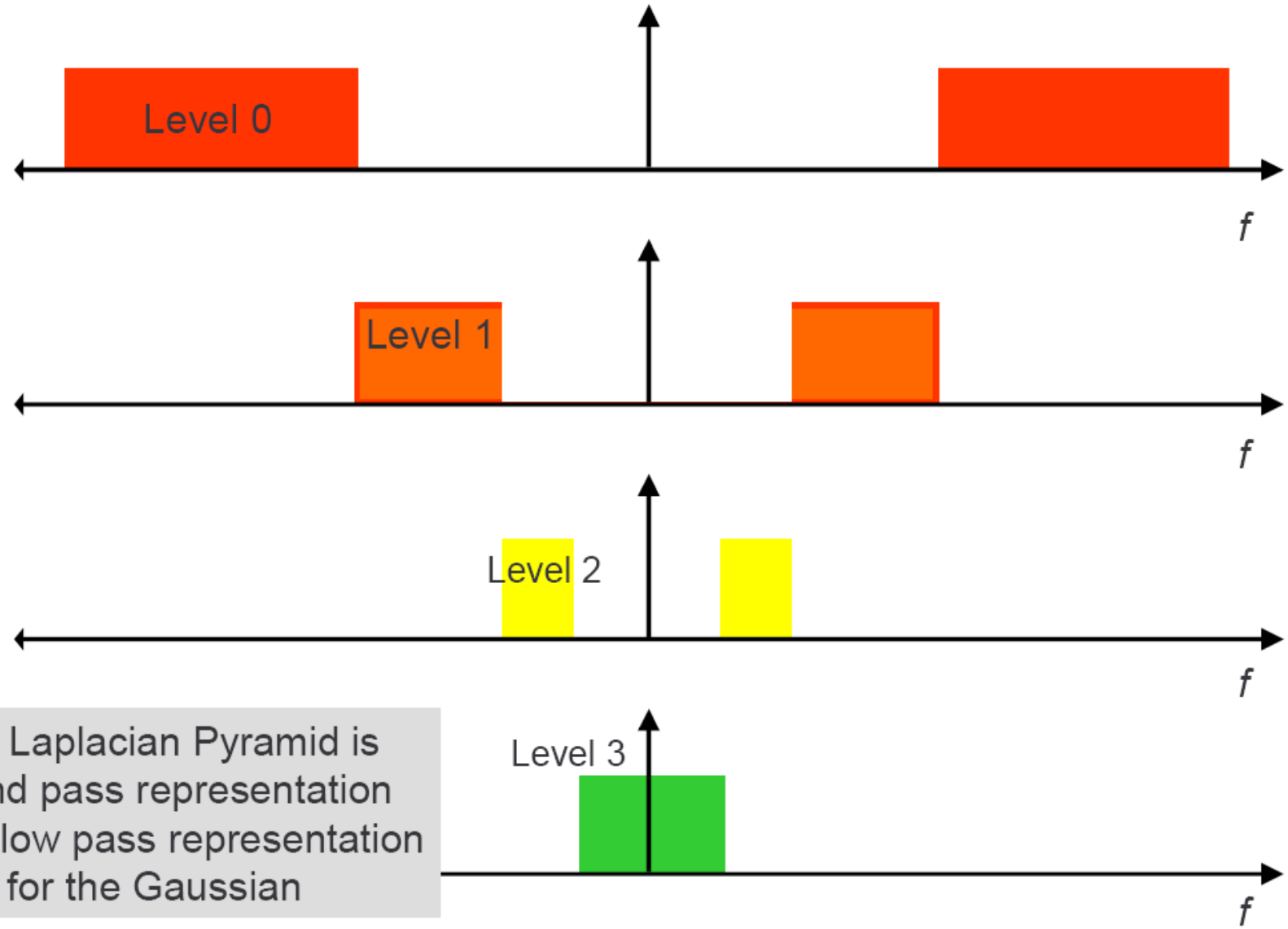
Fig 5. First four levels of the Gaussian and Laplacian pyramid. Gaussian images, upper row, were obtained by expanding pyramid arrays (Fig. 4) through Gaussian interpolation. Each level of the Laplacian pyramid is the difference between the corresponding and next higher levels of the Gaussian pyramid.

Gaussian pyramid frequency composition



Slide adopted from S. Narasimhan

Laplacian Pyramid Frequency Composition



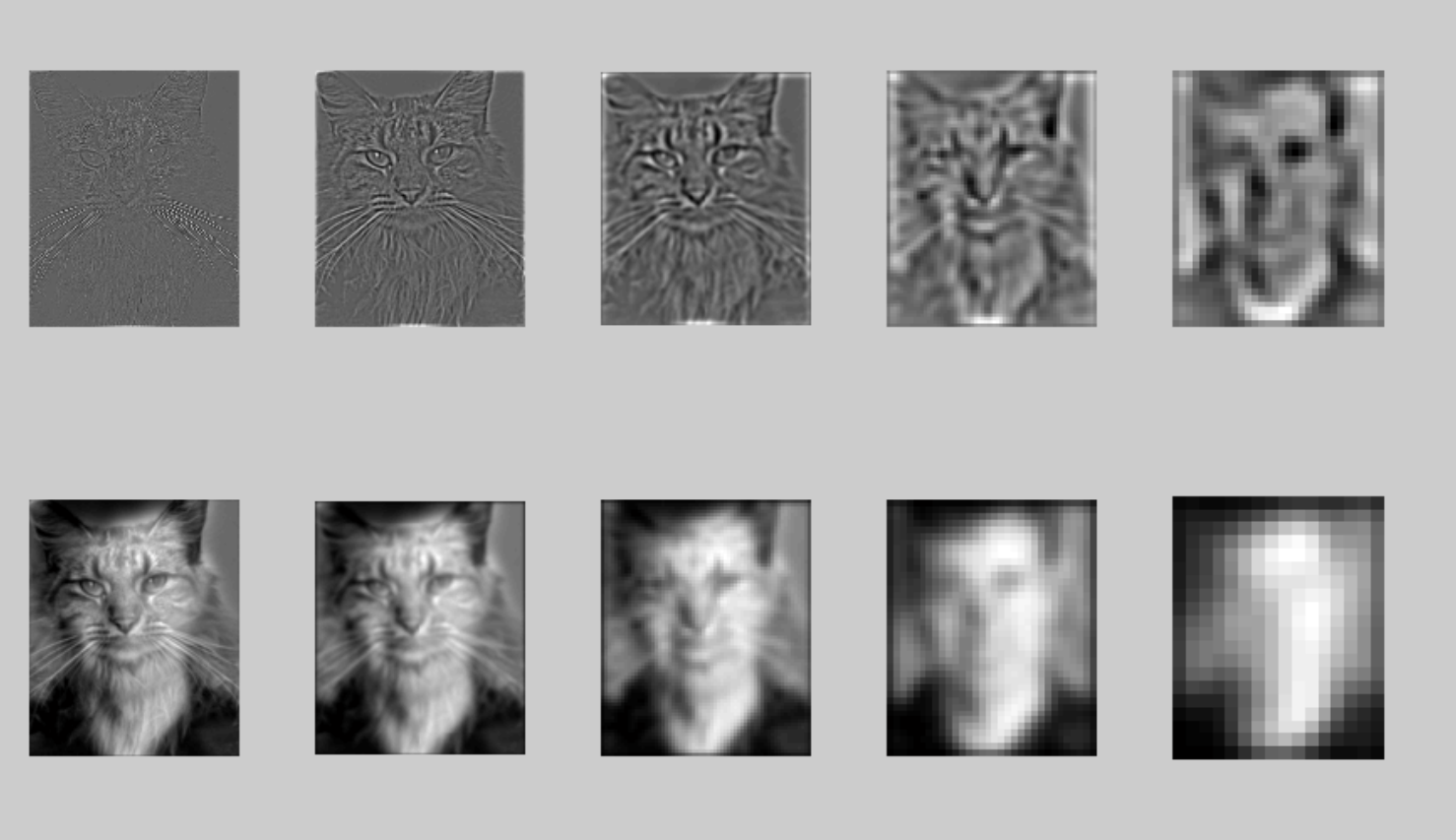
The Laplacian Pyramid is a band pass representation vice a low pass representation for the Gaussian

Source: Lazebnik

Applications of Pyramids

Hybrid Image

Hybrid Image in Laplacian Pyramid



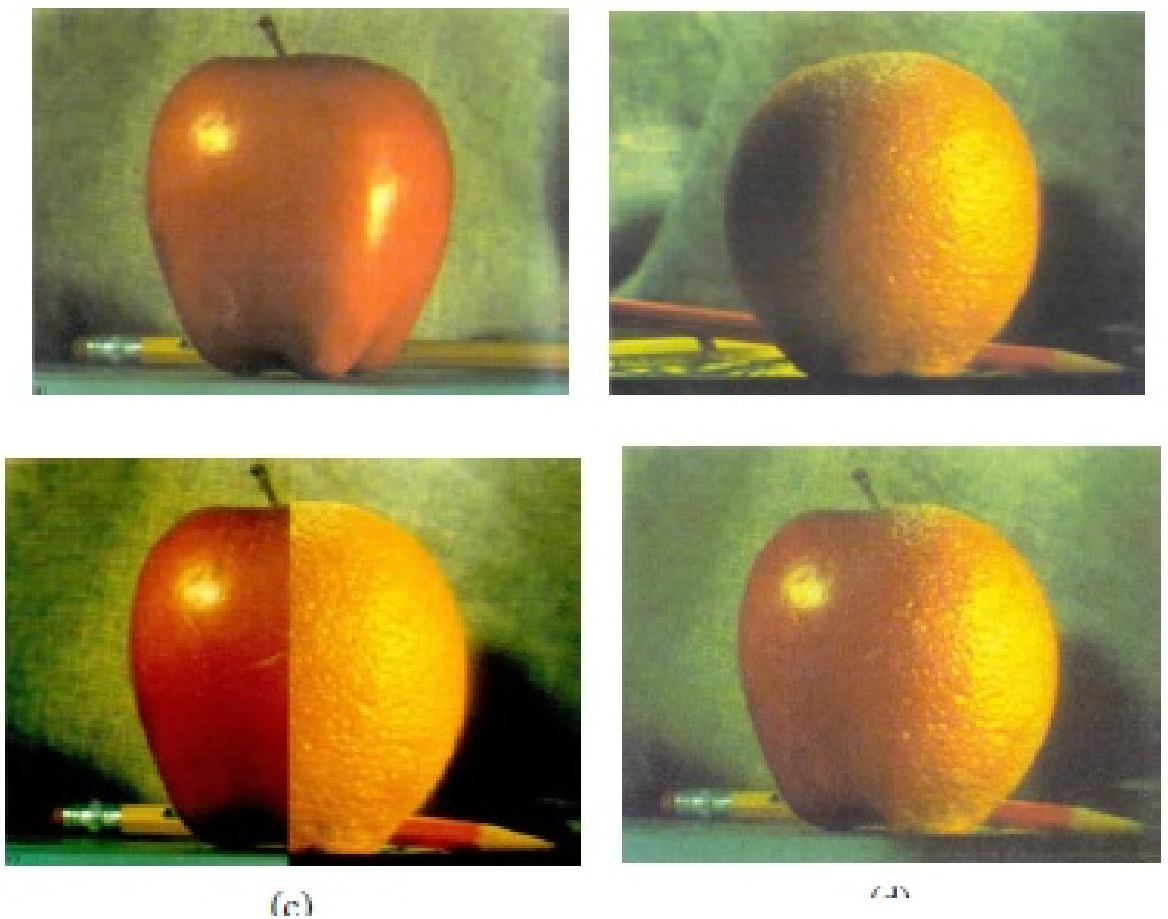
Fast Template Matching

Template

Search Region

Original Image

Image Blending



(c)

How to Blend 2 Image?

- Gaussian Pyramid g_{apple} and g_{orange} with the original image
- Compute L_{apple} and L_{orange}
- Combine left halve of L_{apple} and right halve of L_{orange}
- Reconstruct the combined image.

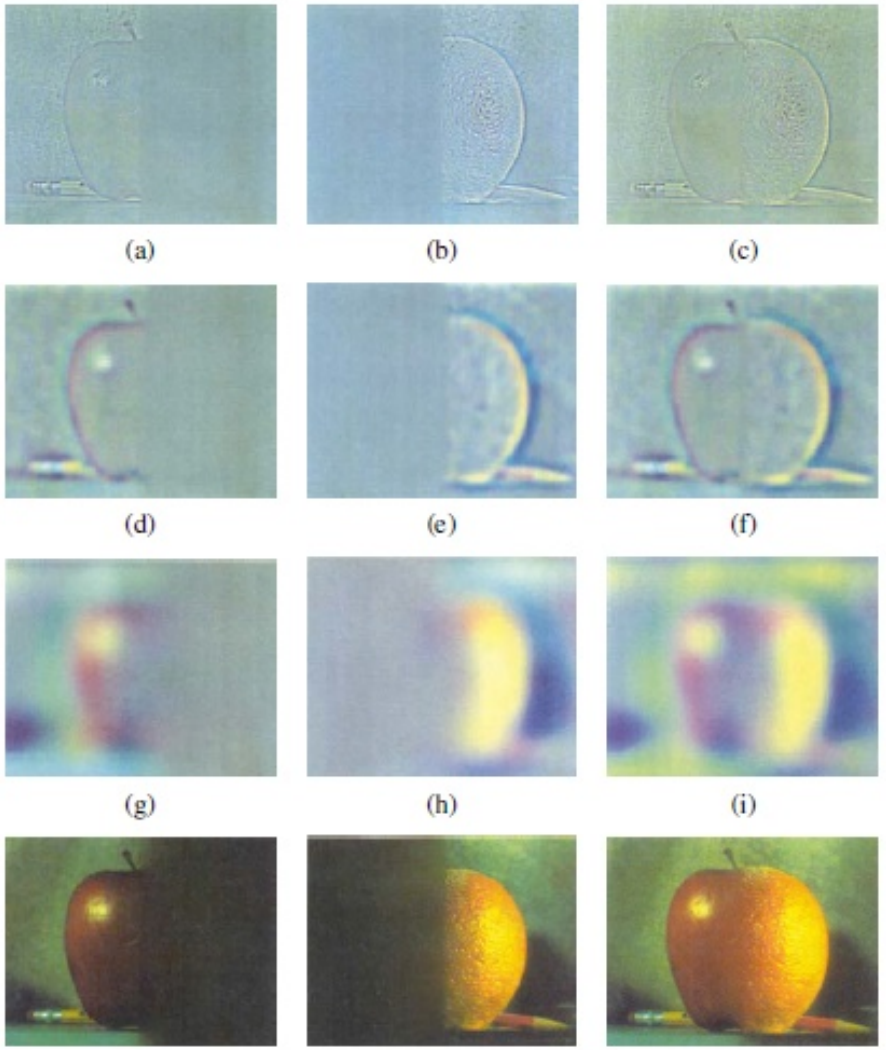
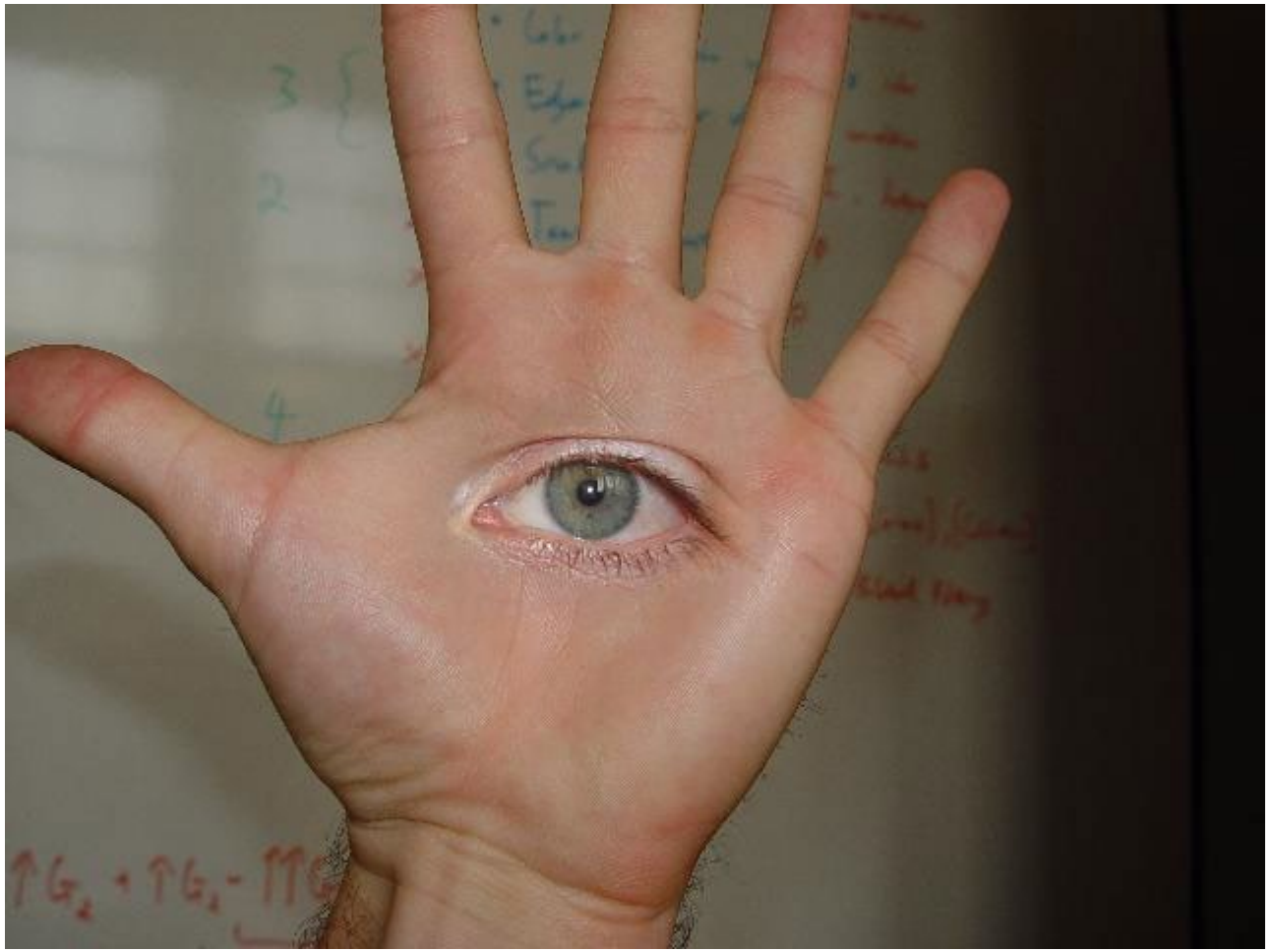


Image Blending

How to Blend 2 Image?



© prof. dmartin

Interactive Digital Photomontage

- Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, Michael Cohen, “Interactive Digital Photomontage”, SIGGRAPH 2004

Interactive Digital Photomontage

Aseem Agarwala[1] Mira Dontcheva[1] Maneesh Agrawala[2] Steven Drucker[2] Alex Colburn[2]
Brian Curless[1] David Salesin[1,2] Michael Cohen[2]

[1]University of Washington [2]Microsoft Research

Abstract

We describe an interactive, computer-assisted framework for combining parts of a set of photographs into a single composite picture, a process we call “digital photomontage.” Our framework makes use of two techniques primarily: graph-cut optimization, to choose good seams within the constituent images so that they can be combined as seamlessly as possible; and gradient-domain fusion, a process based on Poisson equations, to further reduce any remaining visible artifacts in the composite. Also central to the framework is a suite of interactive tools that allow the user to specify a variety of high-level image objectives, either globally across the image, or locally through a painting-style interface. Image objectives are applied independently at each pixel location and generally involve a function of the pixel values (such as “maximum contrast”) drawn from that same location in the set of source images. Typically, a user applies a series of image objectives iteratively in order to create a finished composite. The power of this framework lies in its generality; we show how it can be used for a wide variety of applications, including “selective composites” (for instance, group photos in which everyone looks their best), relighting, extended depth of field, panoramic stitching, clean-plate production, stroboscopic visualization of movement, and time-lapse mosaics.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques; I.4.9 [Image Processing and Computer Vision]: Applications

Keywords: Interactive image editing, image compositing, user-guided optimization

1 Introduction

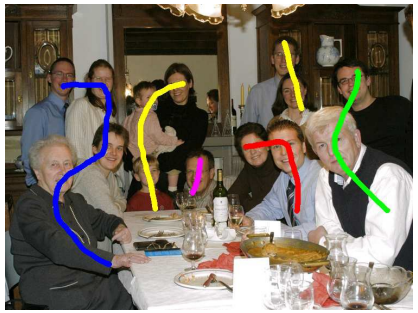
In this paper, we look at how digital photography can be used to create photographic images that more accurately convey our subjective impressions — or go beyond them, providing visualizations or a greater degree of artistic expression. Our approach is to utilize multiple photos of a scene, taken with a digital camera, in which some aspect of the scene or camera parameters varies with each photo. (A film camera could also be used, but digital photography makes the process of taking large numbers of exposures particularly easy and inexpensive.) These photographs are then pieced together, via an interactive system, to create a single photograph that better conveys the photographer’s subjective perception of the scene. We call this process *digital photomontage*, after the traditional process of combining parts of a variety of photographs to form a composite picture, known as *photomontage*.

The primary technical challenges of photomontage are 1) to choose good seams between parts of the various images so that they can be joined with as few visible artifacts as possible; and 2) to reduce any remaining artifacts through a process that fuses the image regions. To this end, our approach makes use of (and combines) two techniques: *graph-cut optimization* [Boykov et al. 2001], which we use to find the best possible seams along which to cut the various source images; and *gradient-domain fusion* (based on Poisson equations [Pérez et al. 2003; Fattal et al. 2002]), which we use to reduce or remove any visible artifacts that might remain after the image seams are joined.

This paper makes contributions in a number of areas. Within the graph-cut optimization, one must develop appropriate cost functions that will lead to desired optima. We introduce several new cost functions that extend the applicability of graph-cuts to a number of new applications (listed below). We also demonstrate a user interface that is designed to encourage the user to treat a stack of images as a single, three-dimensional entity, and to explore and find the best parts of the stack to combine into a single composite. The interface allows the user to create a composite by painting with high-level goals; rather then requiring the user to manually select specific sources, the system automatically selects and combines

Source images

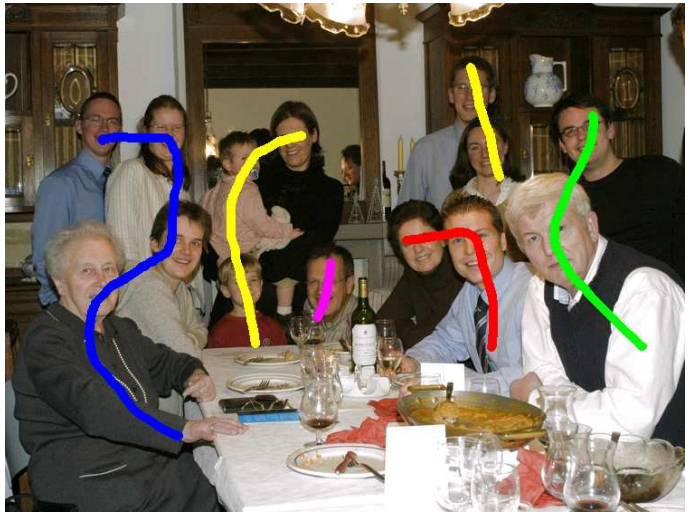
Brush strokes



Computed labeling

Composite

Brush strokes



Computed labeling

Today we learnt

- Application of filters
  - Image Smoothing
  - Template matching (SSD) or Normalized Cross-correlation
  - Downsampling
    - Smoothing and then downsampling
  - Pyramid
    - Gaussian pyramid
      - Image sub-sampling
      - Coarse-to-fine search, multi-scale detection (next class)
    - Laplacian pyramid
      - Image compression
      - Image Blending

Next Class

- Edge Detection
  - Gradient Detector
  - Laplacian of Gaussian (Marr-Hildreth)
  - Gradient of Gaussian (Canny)
- Interest Point Detection
- Scale-invariant Region Selection
- SIFT/SURF/ORB/HOG