19CSE435: Computer Vision
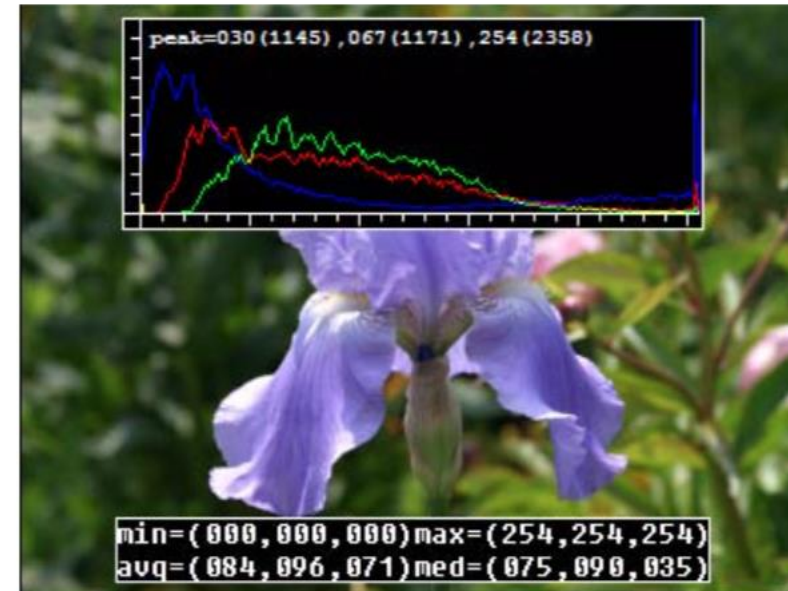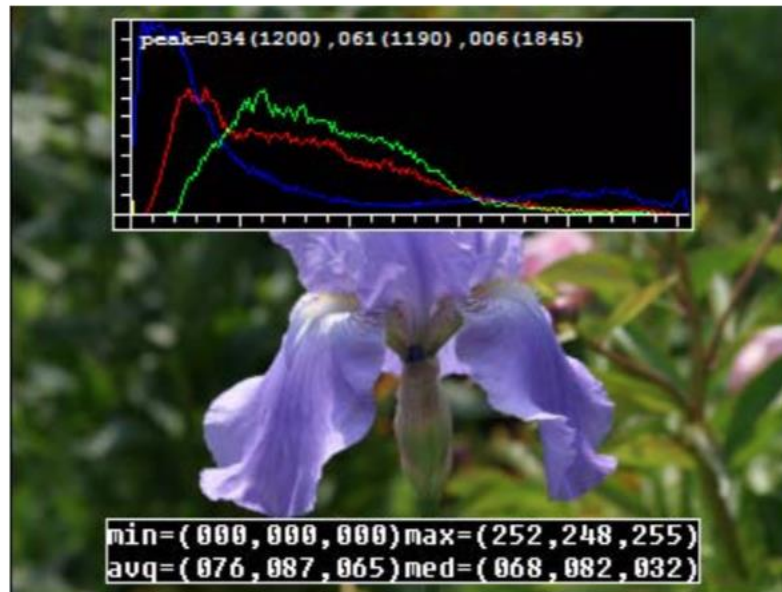
Image processing

3.1.1 Pixel transforms

- **Contrast**
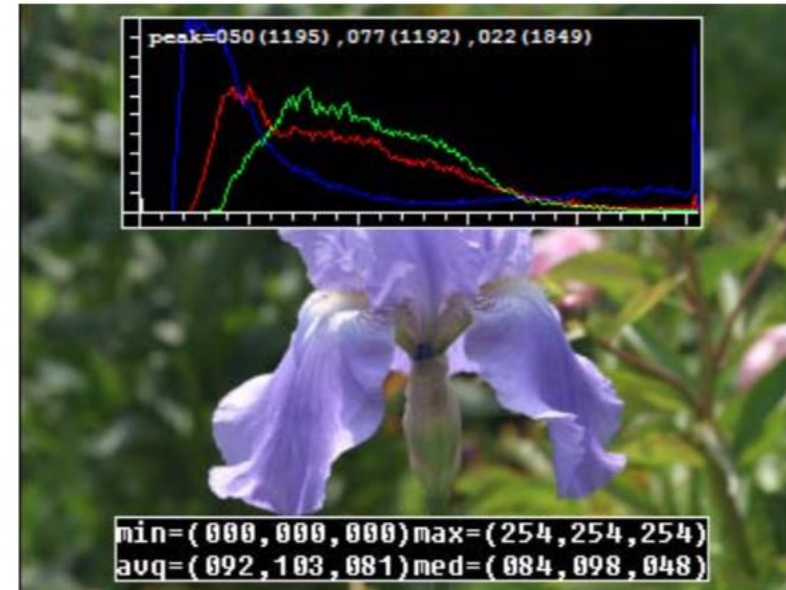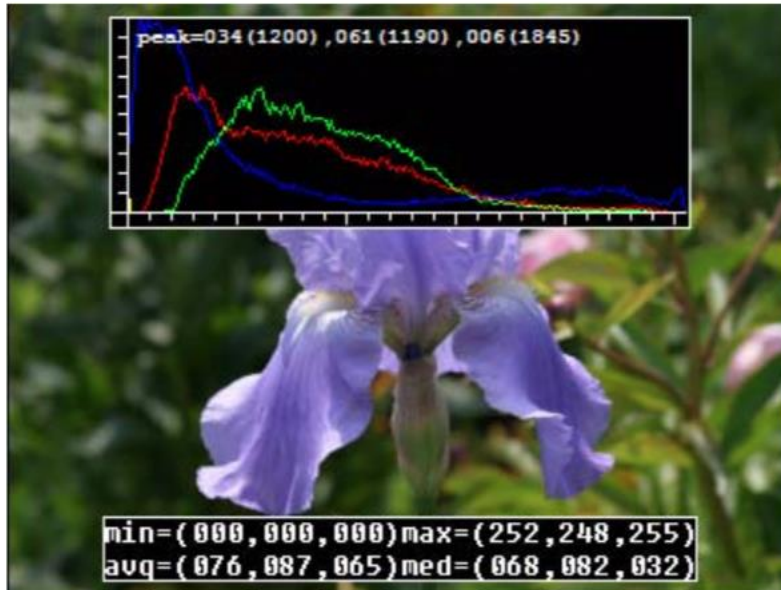- **Brightness**
- **Gamma**
- **Histogram equalization**
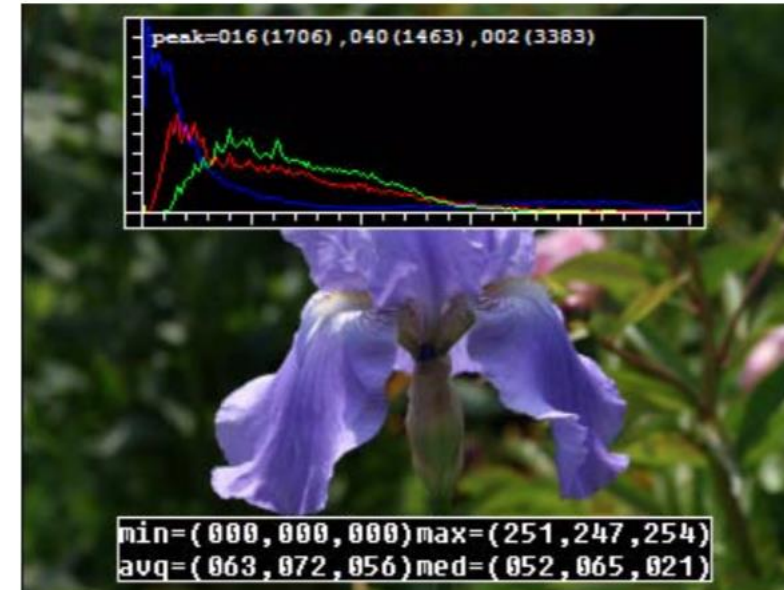- **Arithmetic**
- **Compositing**

Contrast

- $g(x) = a\ f(x),\ a=1.1$

Brightness

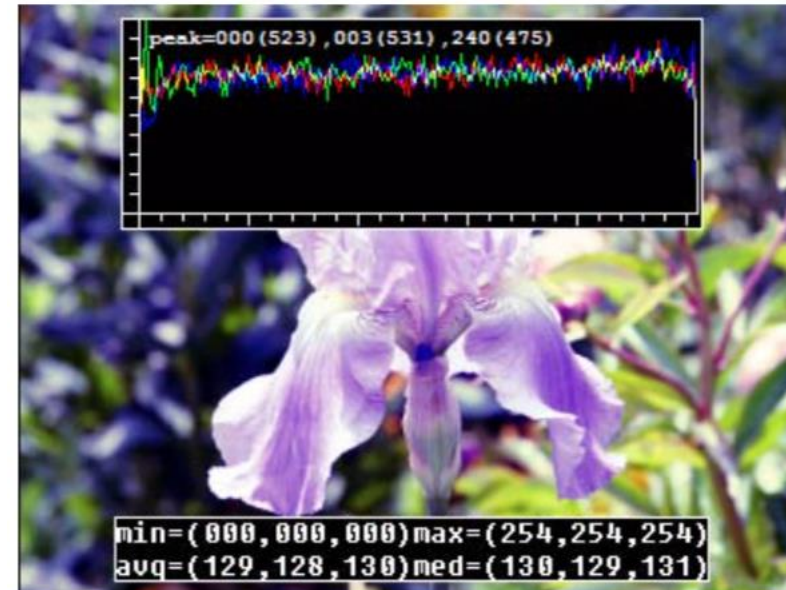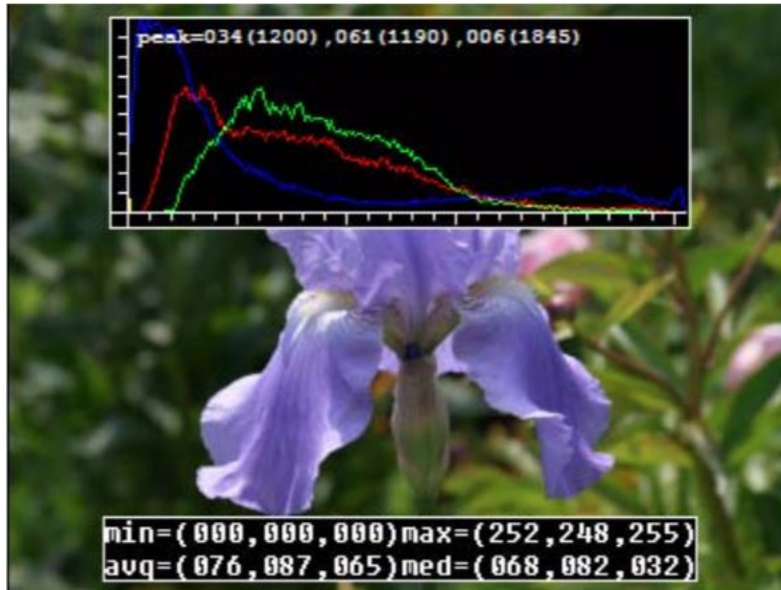- $g(x) = f(x) + b, b = 16$

Gamma correction
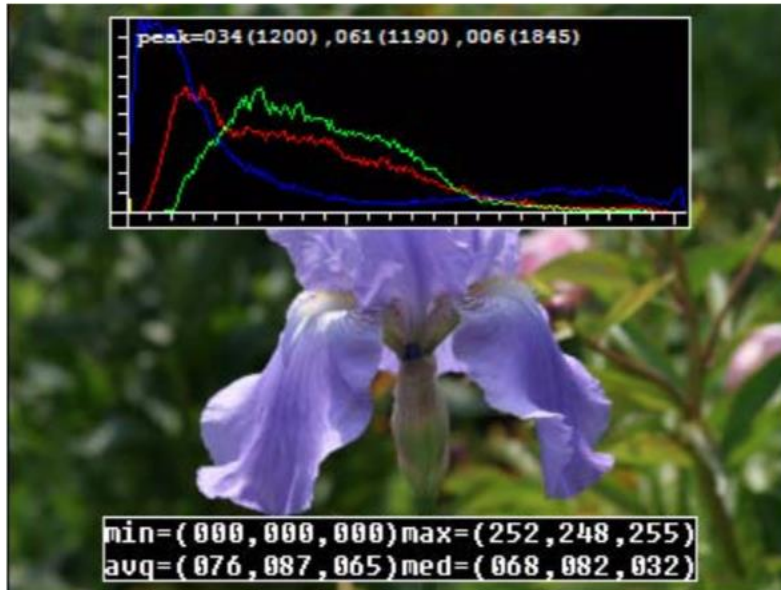
$$g(\boldsymbol{x}) = [f(\boldsymbol{x})]^{1/\gamma}$$

- gamma = 1.2

Histogram Equalization

- **Non-linear transform to make histogram flat**

- **Still a per-pixel operation g(x) = h(f(x))**

Point-Process: Pixel/Point Arithmetic

120	122	140	142	143
121	120	141	144	147
122	121	144	146	11
125	121	144	145	10
126	121	145	147	13

+

120	122	140	142	143
121	80	40	144	10
122	81	40	0	151
125	80	40	0	152
126	70	40	0	153

=

240	244	280	284	286
121	200	181	288	157
122	202	184	146	162
125	201	184	145	164
126	191	185	147	166

120	122	140	142	143
121	120	141	144	147
122	121	144	146	11
125	121	144	145	10
126	121	145	147	13

-

120	122	140	142	143
121	80	40	144	10
122	81	40	0	151
125	80	40	0	152
126	70	40	0	153

=

0	0	0	0	0
0	40	101	0	137
0	40	104	146	-140
0	40	104	145	-142
0	191	185	147	-140

Pixel/Point Arithmetic: An Example

Image 1

−

Image 2

=

Image 1 - Image 2
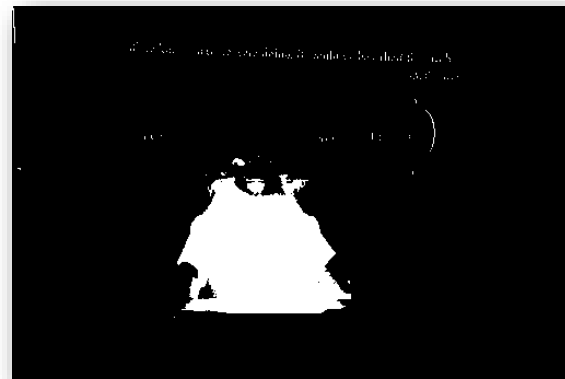
Binary(Image 1 - Image 2)

Matte: an alpha image

(1-a)B
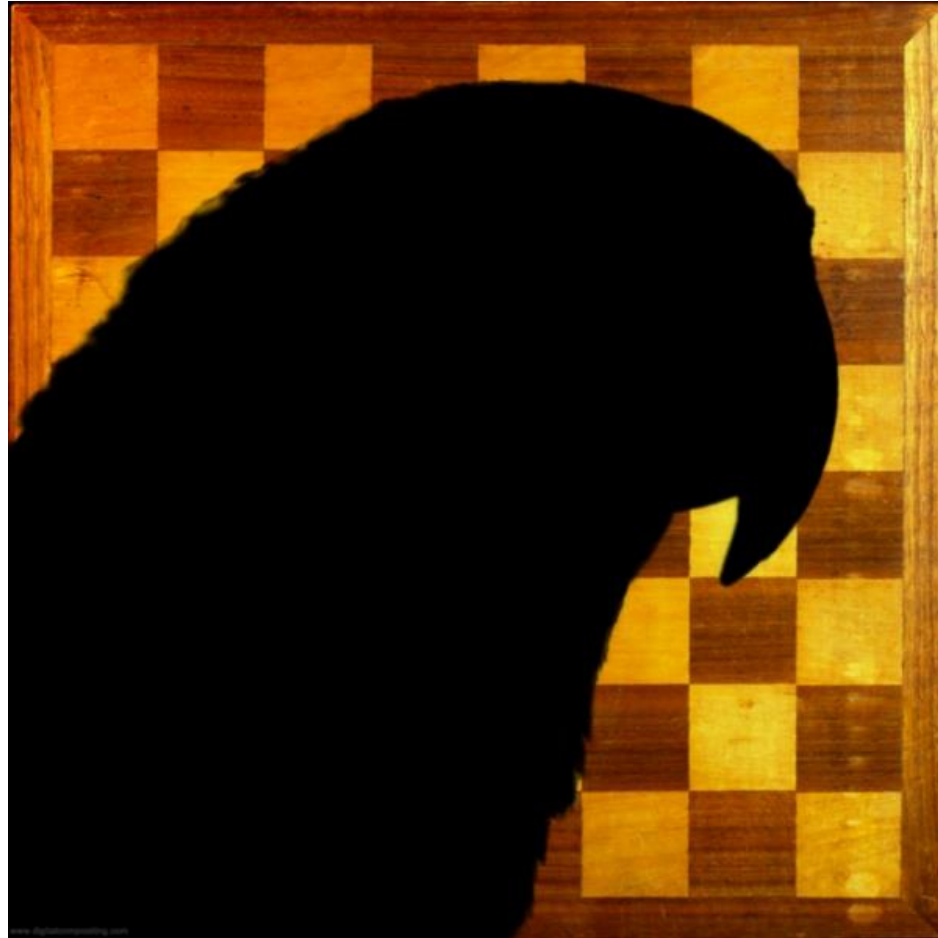
KeyMix: aF + (1-a)B
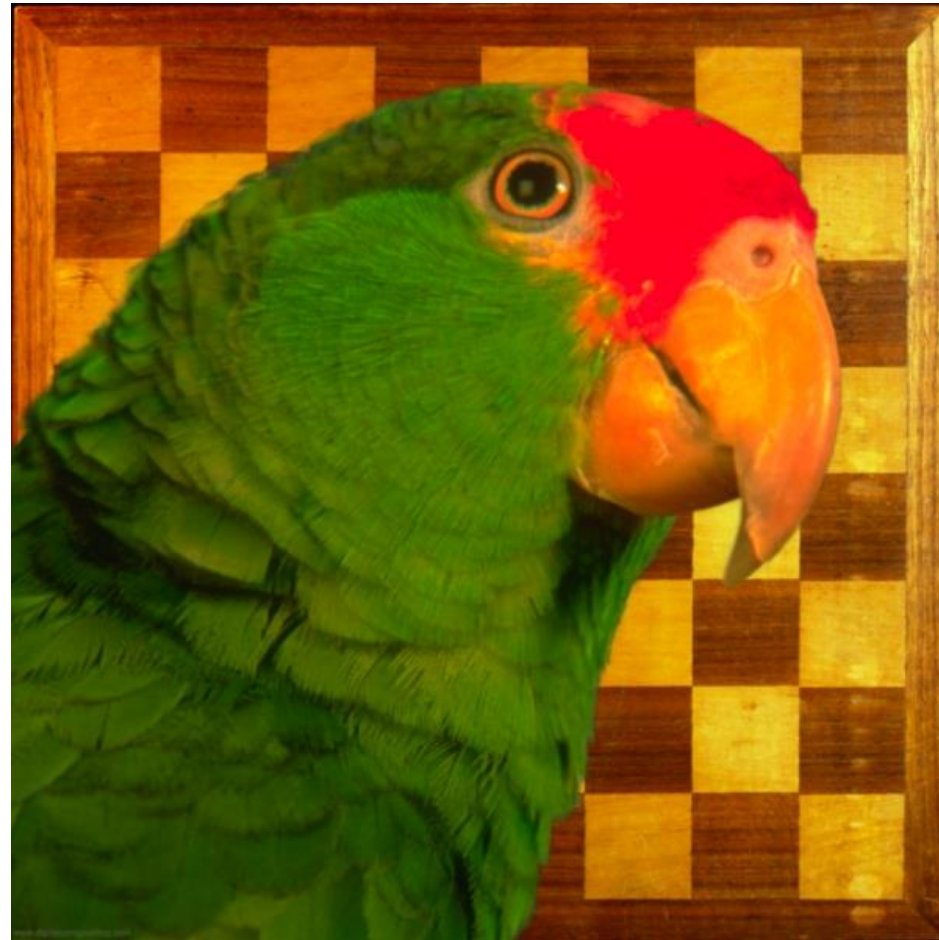
Premultiplied RGBA Images
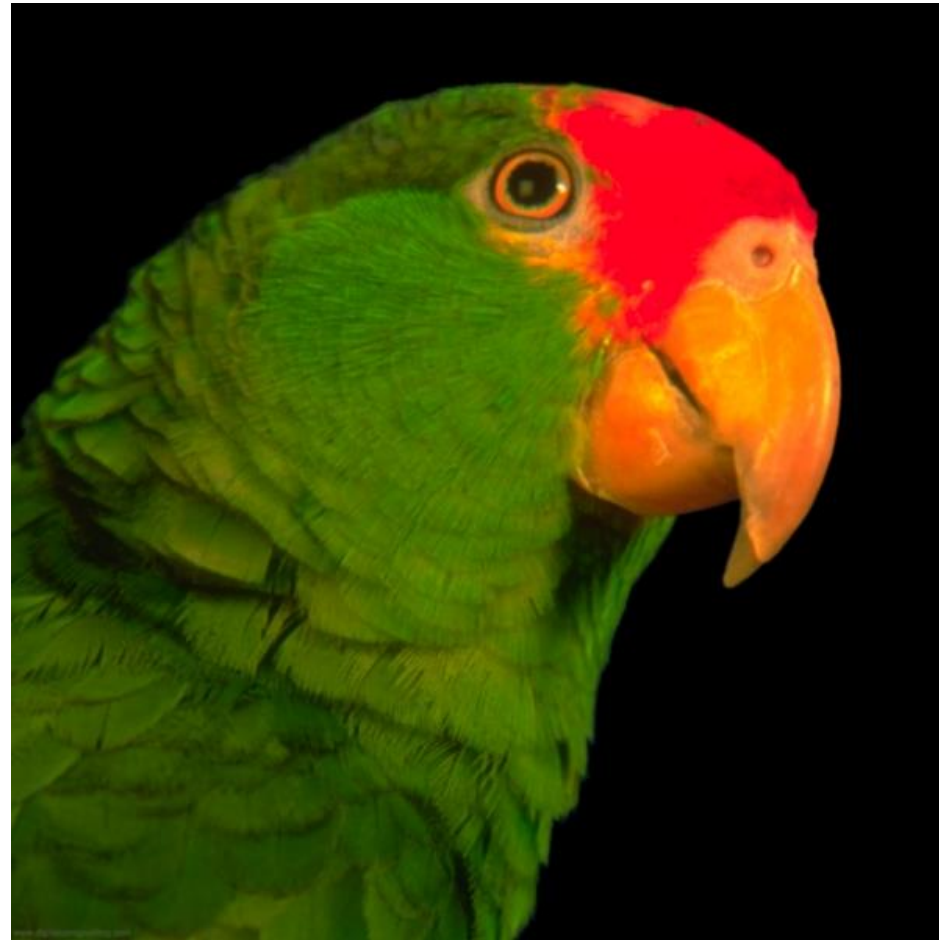
Over: F + (1-a)B
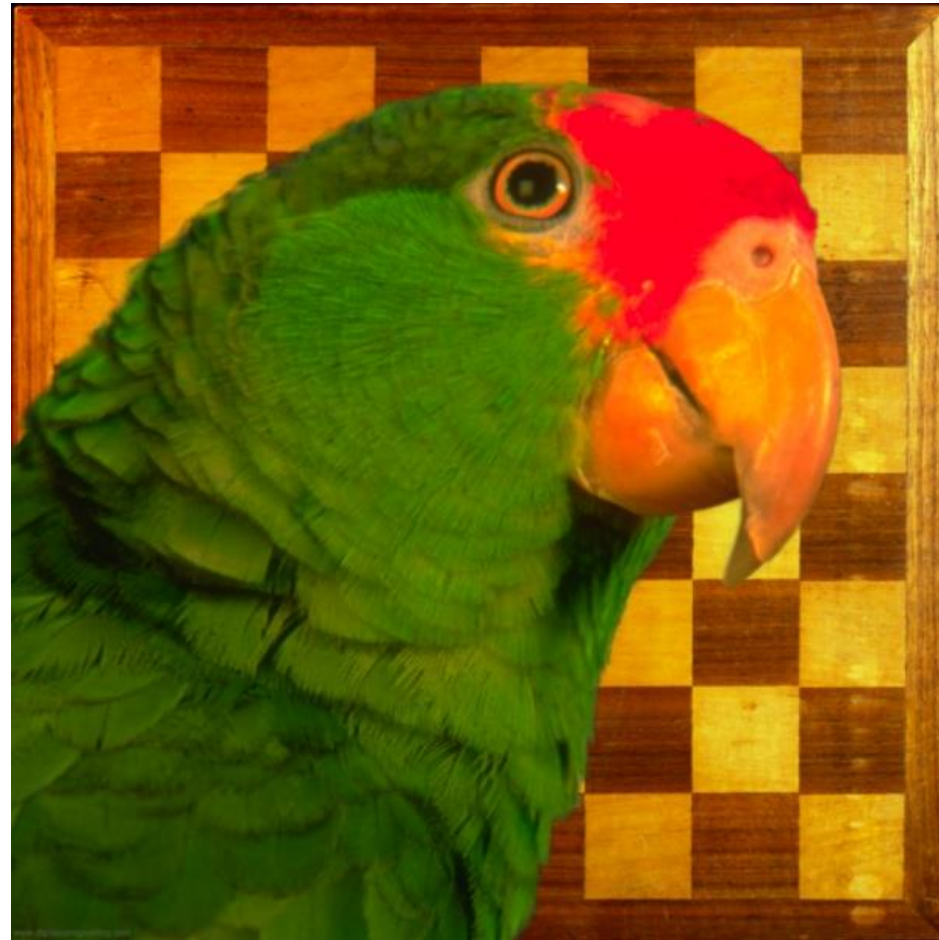
Image filtering

- **Image filtering: compute function of local neighborhood at each position**

- **Really important!**
  - Enhance images
    - Denoise, resize, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching
  - Deep Convolutional Networks

Example: box filter

$$g[\cdot,\cdot]$$

$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1

Slide credit: David Lowe (UBC)

Image filtering

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

1	1	1
1	1	1
1	1	1

$$f[.,.]$$

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

$$h[.,.]$$

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

Credit: S. Seitz

Image filtering

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

1	1	1
1	1	1
1	1	1

$$f[.,.]$$

$$h[.,.]$$

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

	0	10							

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

Image filtering

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

1	1	1
1	1	1
1	1	1

$$f[.,.]$$

$$h[.,.]$$

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

	0	10	20						

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

Credit: S. Seitz

Image filtering

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

1	1	1
1	1	1
1	1	1

$$f[.,.]$$

$$h[.,.]$$

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

		0	10	20	30				

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

Credit: S. Seitz

Image filtering

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

1	1	1
1	1	1
1	1	1

$$f[\cdot,\cdot]$$

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

$$h[\cdot,\cdot]$$

	0	10	20	30	30				

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

Credit: S. Seitz

Image filtering

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

1	1	1
1	1	1
1	1	1

$$f[.,.]$$

$$h[.,.]$$

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

	0	10	20	30	30				
				?					

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

Credit: S. Seitz

Image filtering

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

1	1	1
1	1	1
1	1	1

$$f[.,.]$$

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

$$h[.,.]$$

	0	10	20	30	30				
					?				
			50						

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

Credit: S. Seitz

Image filtering

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

1	1	1
1	1	1
1	1	1

$$f[.,.]$$

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

$$h[.,.]$$

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

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

Credit: S. Seitz

Box Filter

What does it do?

- Replaces each pixel with an average of its neighborhood

- Achieve smoothing effect (remove sharp features)

$$g[\cdot,\cdot]$$

$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1

Slide credit: David Lowe (UBC)

Smoothing with box filter
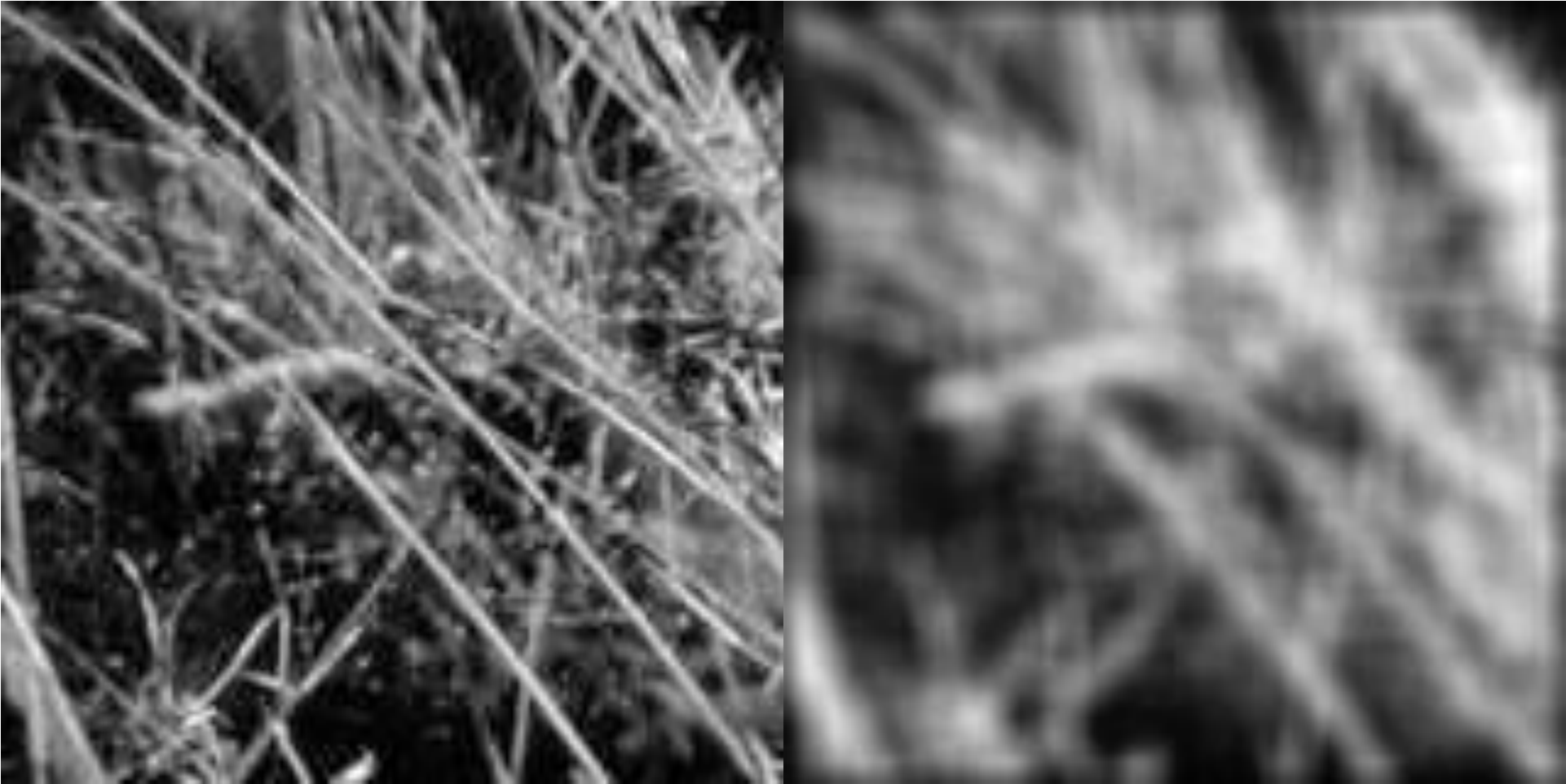
Original

0	0	0
0	1	0
0	0	0

?

Practice with linear filters

0	0	0
0	1	0
0	0	0

Original

Filtered
(no change)

Source: D. Lowe

Original

0	0	0
0	0	1
0	0	0

?

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

Shifted left
By 1 pixel

Source: D. Lowe

Original

0	0	0
0	2	0
0	0	0

-

$\dfrac{1}{9}$

1	1	1
1	1	1
1	1	1

?

(Note that filter sums to 1)

Source: D. Lowe

Practice with linear filters

Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$
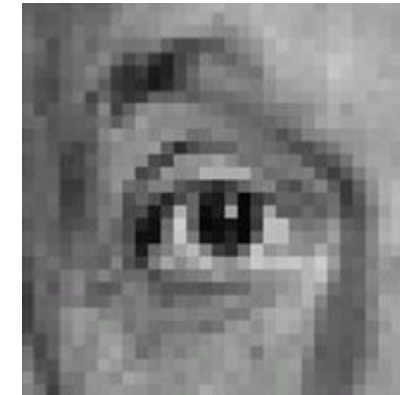
Sharpening filter
- Accentuates differences with local average

Source: D. Lowe

Sharpening

before   after

Source: D. Lowe
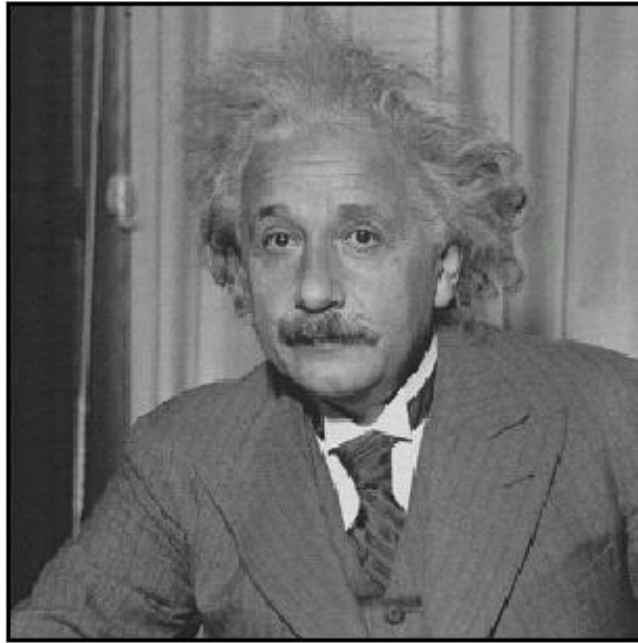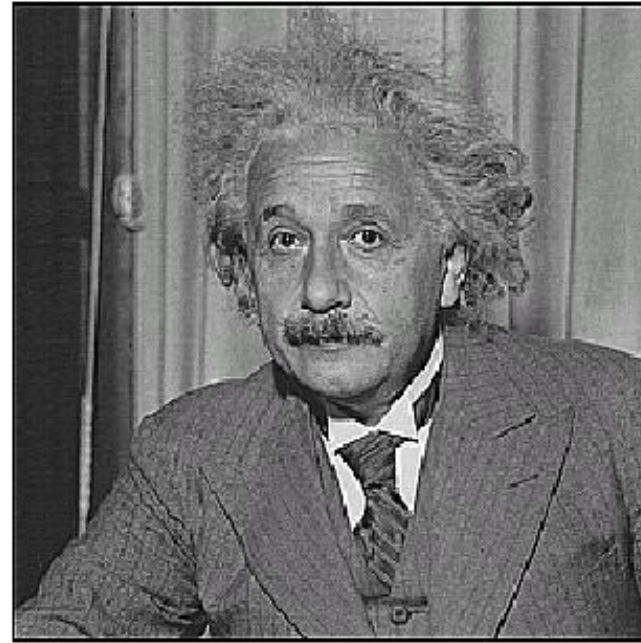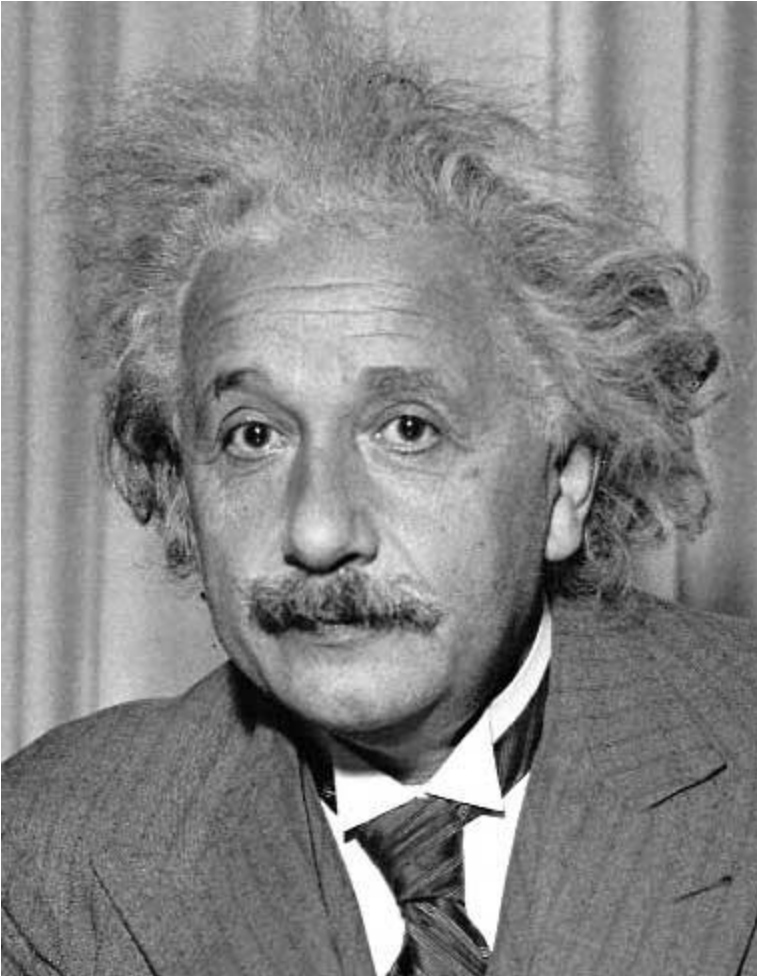
Other filters

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

Sobel
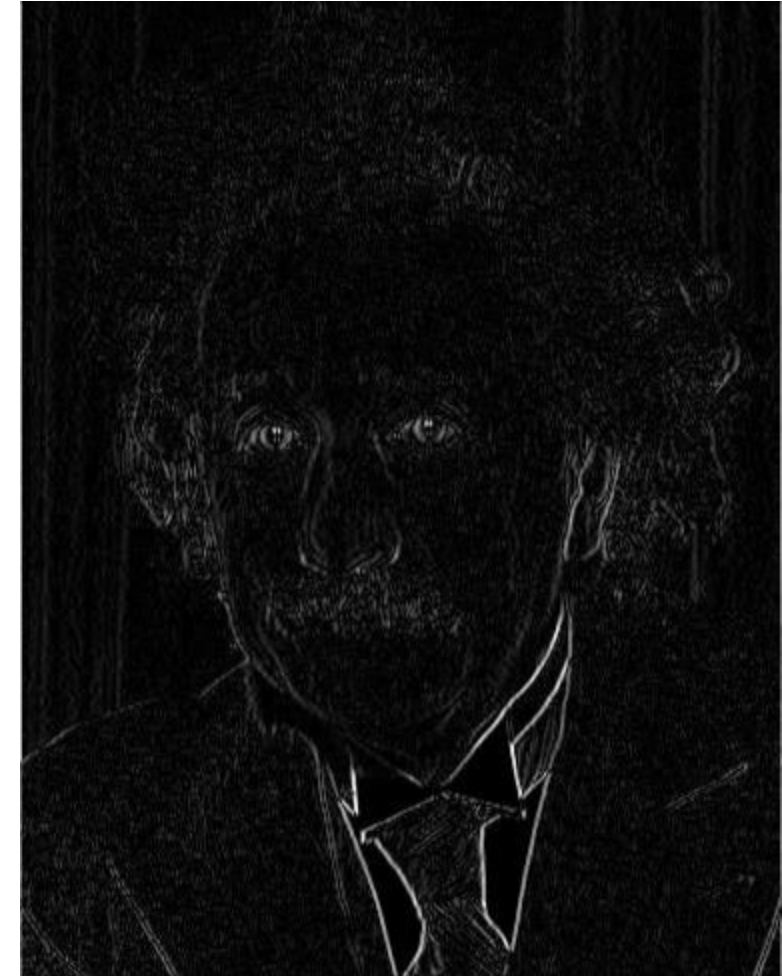
Vertical Edge
(absolute value)

Other filters
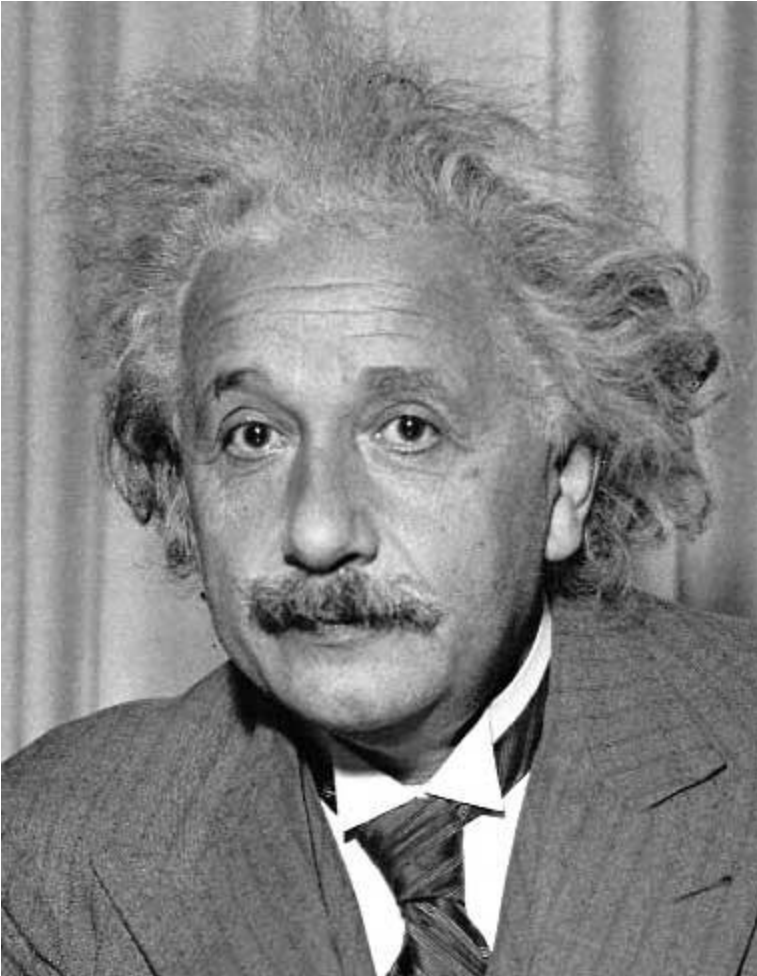
1	2	1
0	0	0
-1	-2	-1

Sobel
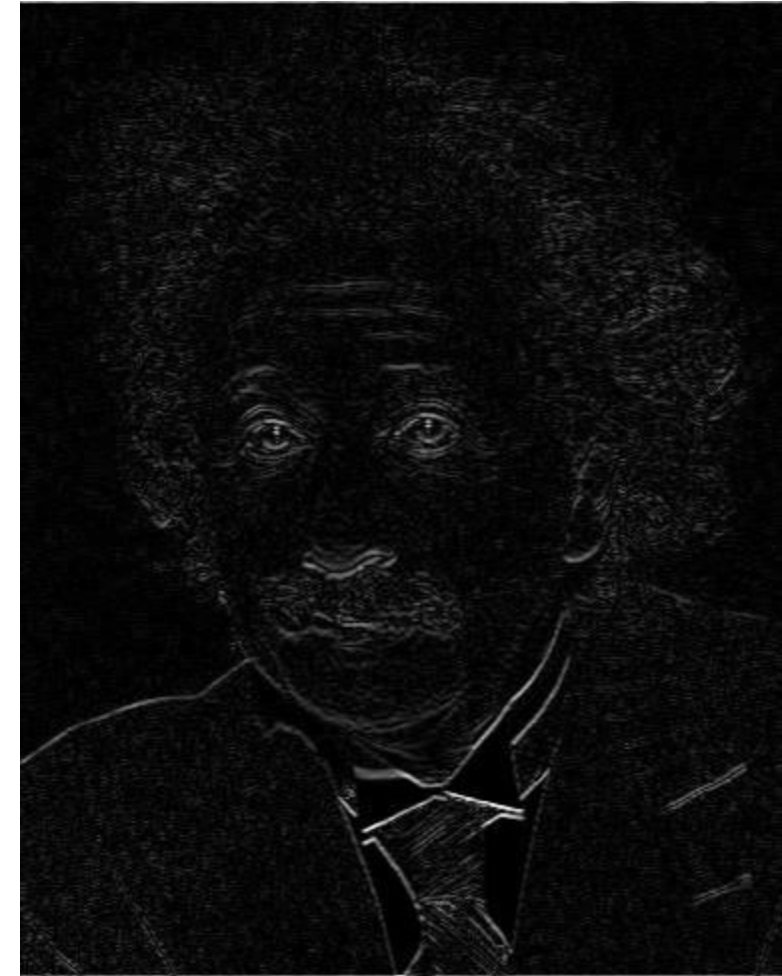
Horizontal Edge
(absolute value)

Filtering vs. Convolution

- **2d filtering** `f=filter` `I=image`
  - `h=filter2(f,I);` or `h=imfilter(I,f);`

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

- **2d convolution**
  - `h=conv2(f,I);`

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

Key properties of linear filters

Linearity:

`imfilter(I, f`$_1$` + f`$_2$`) =`

   `imfilter(I,f`$_1$`) + imfilter(I,f`$_2$`)`

Shift invariance: same behavior regardless of pixel location

`imfilter(I,shift(f)) = shift(imfilter(I,f))`

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

More properties

- **Commutative: $a * b = b * a$**
    - Conceptually no difference between filter and signal
    - But particular filtering implementations might break this equality

- **Associative: $a * (b * c) = (a * b) * c$**
    - Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
    - This is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$

- **Distributes over addition: $a * (b + c) = (a * b) + (a * c)$**

- **Scalars factor out: $ka * b = a * kb = k (a * b)$**

- **Identity: unit impulse $e = [0, 0, 1, 0, 0]$, $a * e = a$**

Important filter: Gaussian

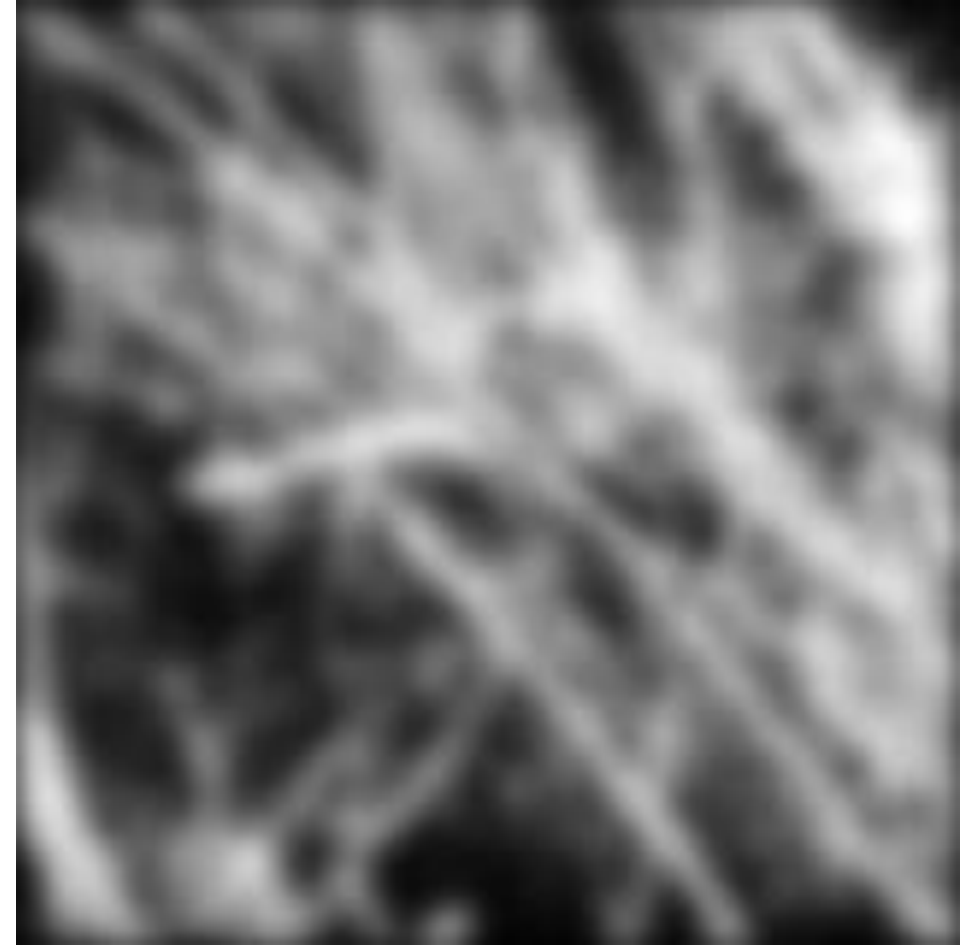- **Weight contributions of neighboring pixels by nearness**
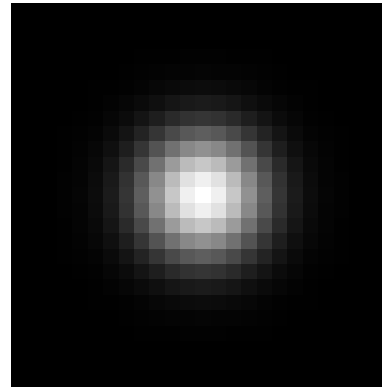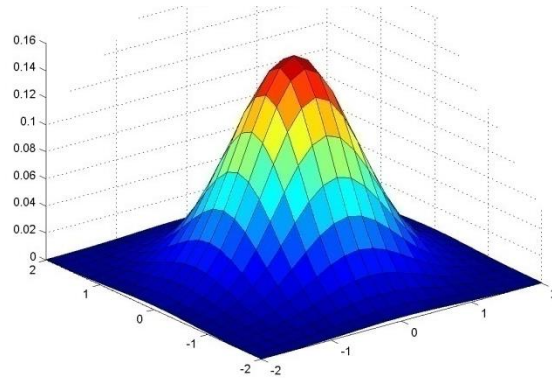
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, σ = 1

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

Smoothing with Gaussian filter
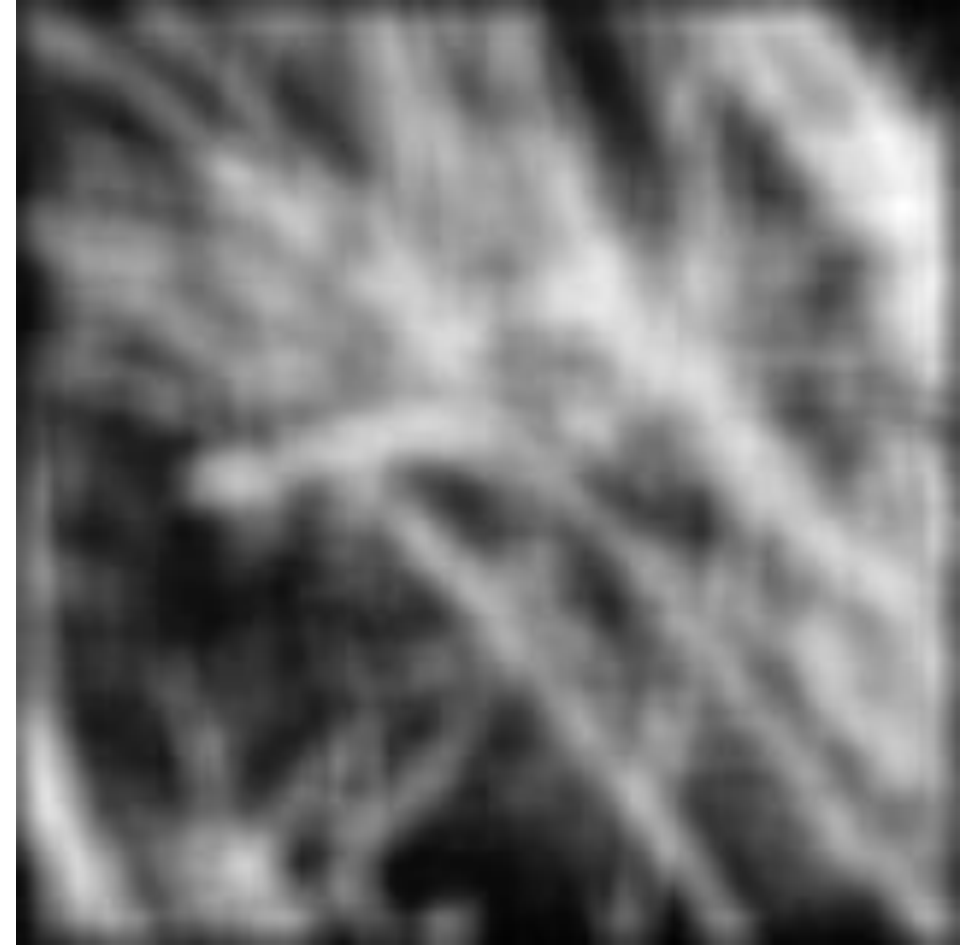
Gaussian filters

- **Remove "high-frequency" components from the image (low-pass filter)**

  - Images become more smooth
- **Convolution with self is another Gaussian**

  - So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
  - Convolving two times with Gaussian kernel of width σ is same as convolving once with kernel of width $\sigma\sqrt{2}$
- ***Separable* kernel**

  - Factors into product of two 1D Gaussians

Separability of the Gaussian filter

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2+y^2}{2\sigma^2}}$$

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

The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

In this case, the two functions are the (identical) 1D Gaussian

2D convolution
(center location only)

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix}$$

The filter factors
into a product of 1D
filters:

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Perform convolution
along rows:

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix} = \begin{bmatrix} & 11 & \\ & 18 & \\ & 18 & \end{bmatrix}$$

Followed by convolution
along the remaining column:

Source: K. Grauman

How big should the filter be?

- Values at edges should be near zero

- Rule of thumb for Gaussian: set filter half-width to about 3 σ

Practical matters

- **What about near the edge?**

  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Source: S. Marschner

- **Linear filtering is dot product at each position**

  - Not a matrix multiplication
  - Can smooth, sharpen, translate (among many other uses)

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

- **Be aware of details for filter size, extrapolation, cropping**
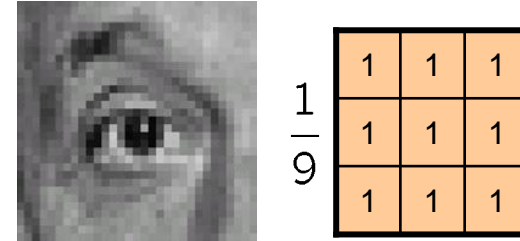
1. Write down a 3x3 filter that returns a positive value if the average value of the 4-adjacent neighbors is less than the center and a negative value otherwise

2. Write down a filter that will compute the gradient in the x-direction:

```
gradx(y,x) = im(y,x+1)-im(y,x) for each x, y
```

3. Fill in the blanks:

Filtering Operator

a) _ = D * B

b) A̅ = _ * _

c) F = D̅ * _

d) _ = D * D̅

A

B

E

G

C

F

H

I

D