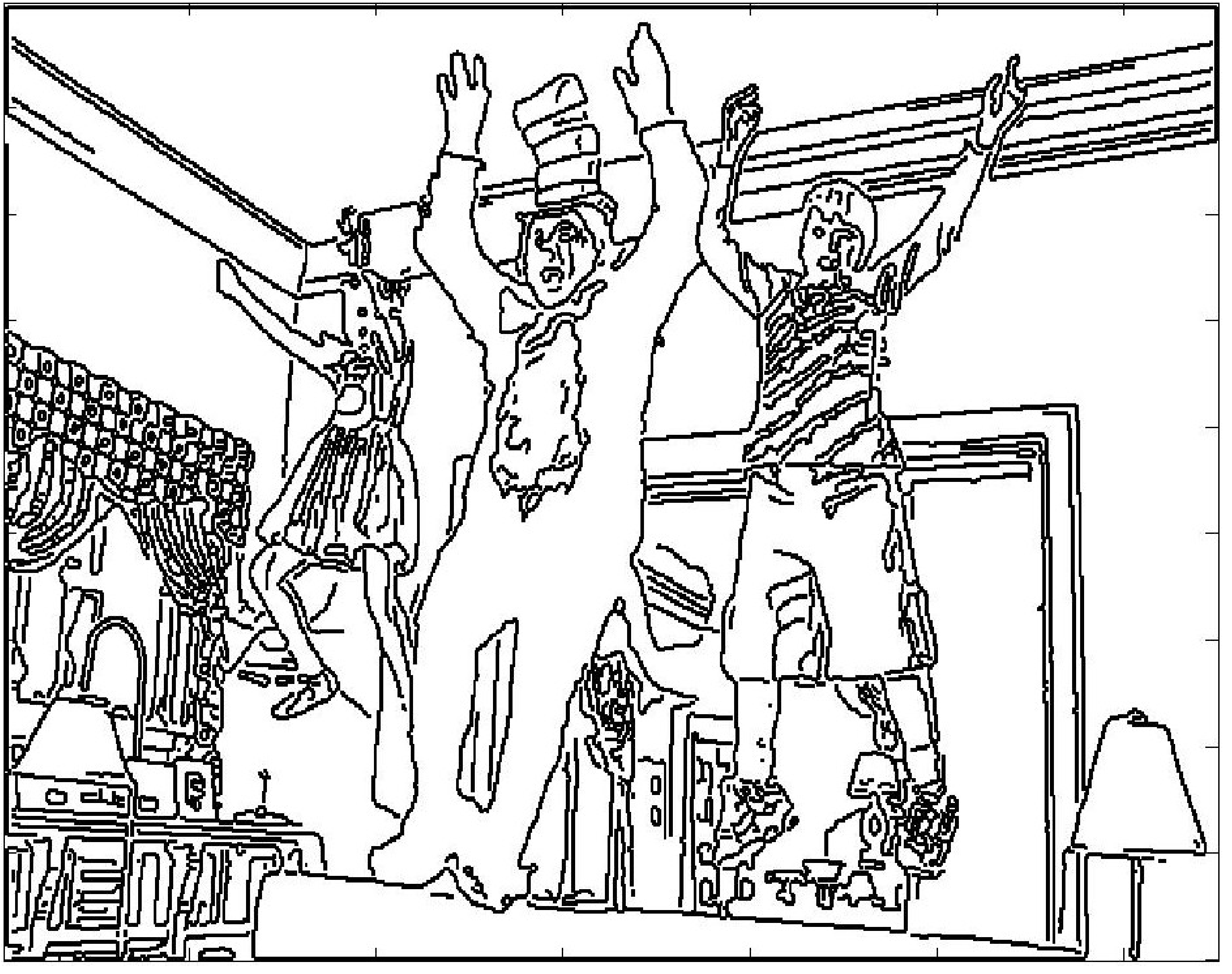
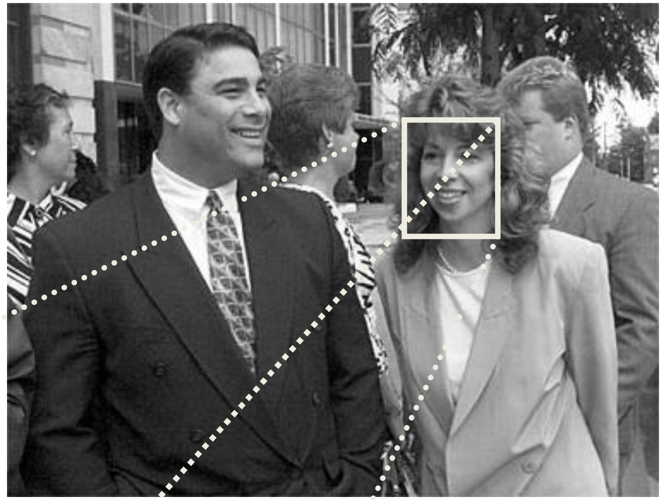
Image Filtering

```
>> Ig = 0.5*(I(:,:,1)+I(:,:,2));
>> [bw,thresh] = edge(Ig,'canny'); imagesc(bw); colormap(gray)
```



In a computer…
an image is a 2 dimensional table of numbers, a 2D matrix



j

i

121	121	118	111	. . .	21
134	136	137	132	. . .	23
133	131	136	136	. . .	25
136	145	148	151	. . .	34
137	140	147	149	. . .	54
. . .	. . .	. . .	. . .	. . .	. . .
231	233	243	244	. . .	179

I(i,j) is the sensor value at location x = i, y = j

I(2,1) = 134
I(3,4) = 136

Any 2D matrix can be seen as an image

Examples:

(r,g,b) =

(255,255,251)

(222,15,7)

(0,0,0)

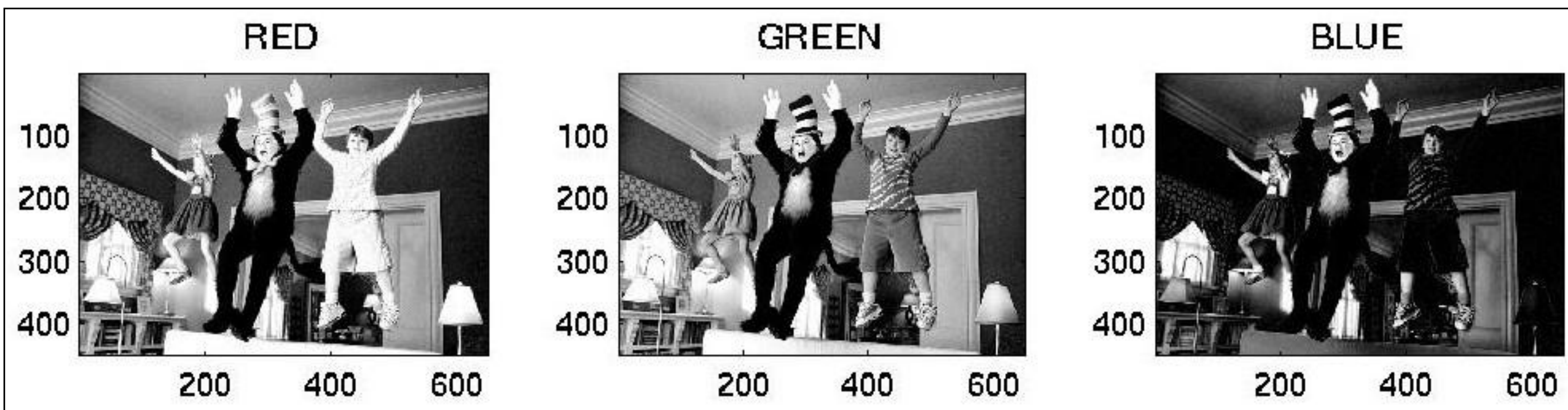
(89,120,1)

(246,99,0)

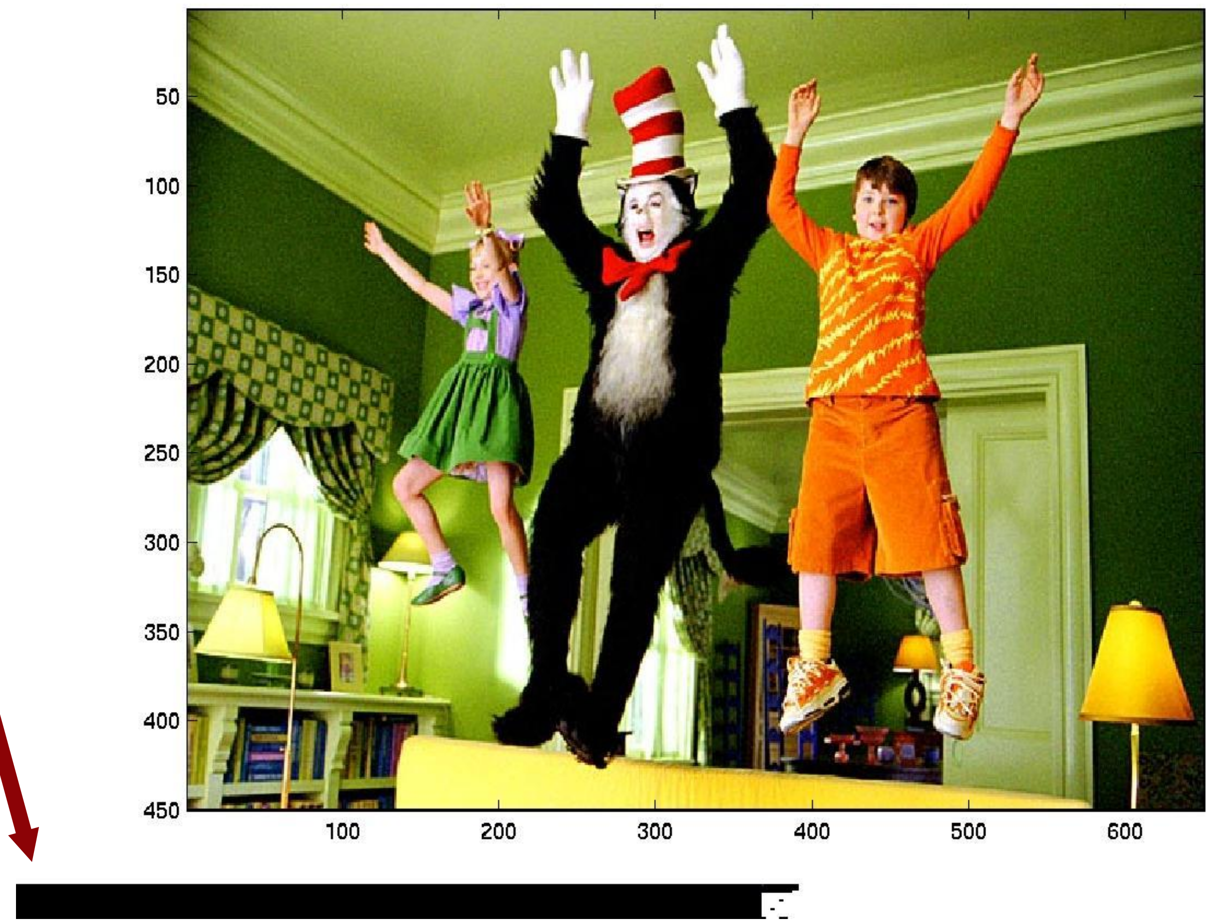
(19,37,87)

(255,255,115)

Brightness = 0.5*(R+G)



```
>> I = imread(image_file);
>> figure(1); image(I);
>> pixval on;
```



```
I = double(I);
Ig = 0.5*(I(:,:,1) + I(:,:,2));
figure(2); imagesc(Ig);
Colormap(gray);
```

Linear functions

- Simplest: linear filtering.
  - Replace each pixel by a linear combination of its neighbors.
- The prescription for the linear combination is called the “convolution kernel”.

10	5	3
4	5	1
1	1	7

Local image data

0	0	0
0	0.5	0
0	1	0.5

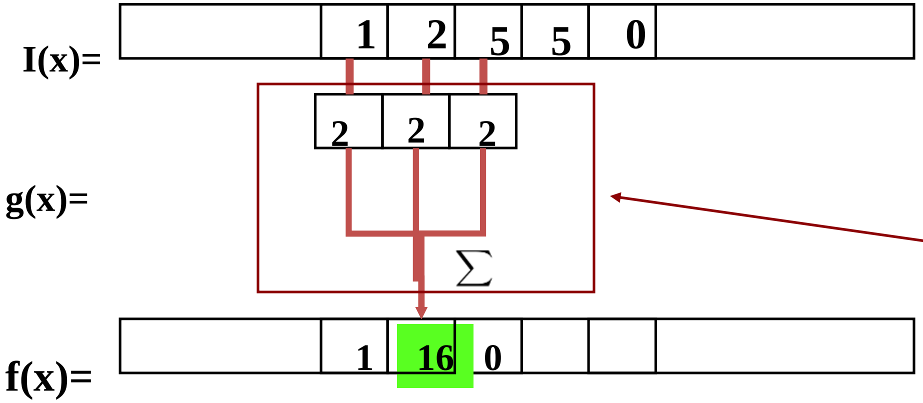
kernel

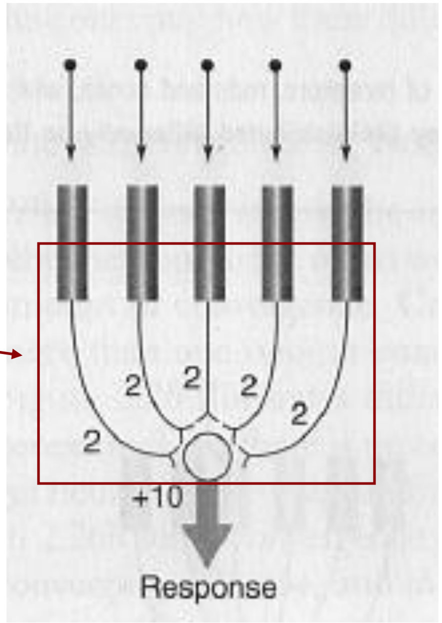
	7	

Modified image data

(Freeman)

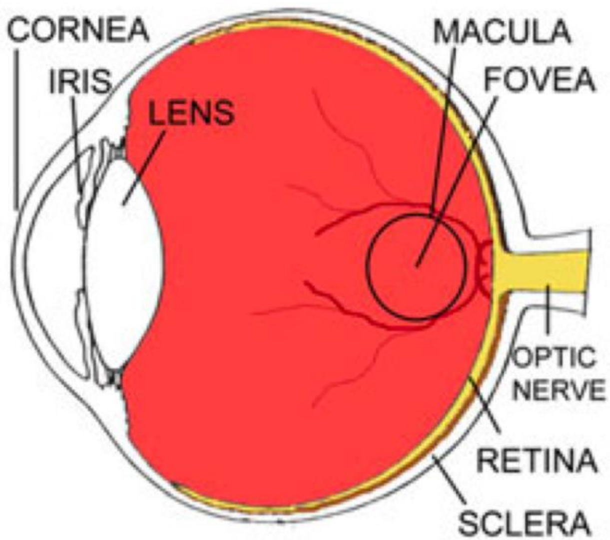
Simple Neural Network

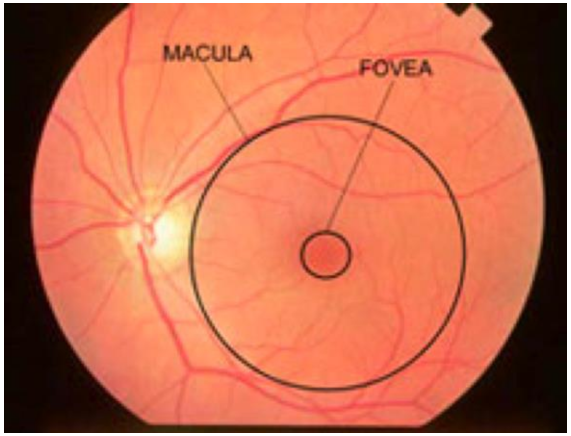




This leads to parallel computation

Anatomy of eye

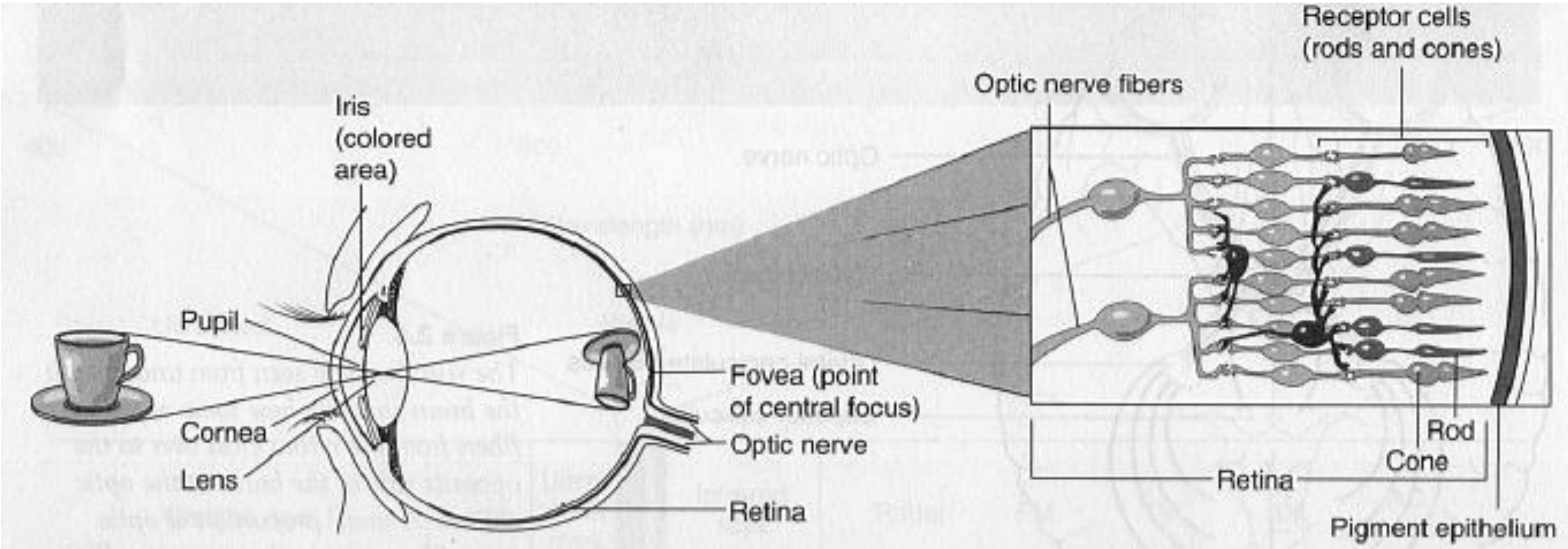




The macula is the center of vision (and retina) and the fovea (FAZ) is the focal point approximately only 0.4mm in diameter. Reading, driving, etc. is all performed here.

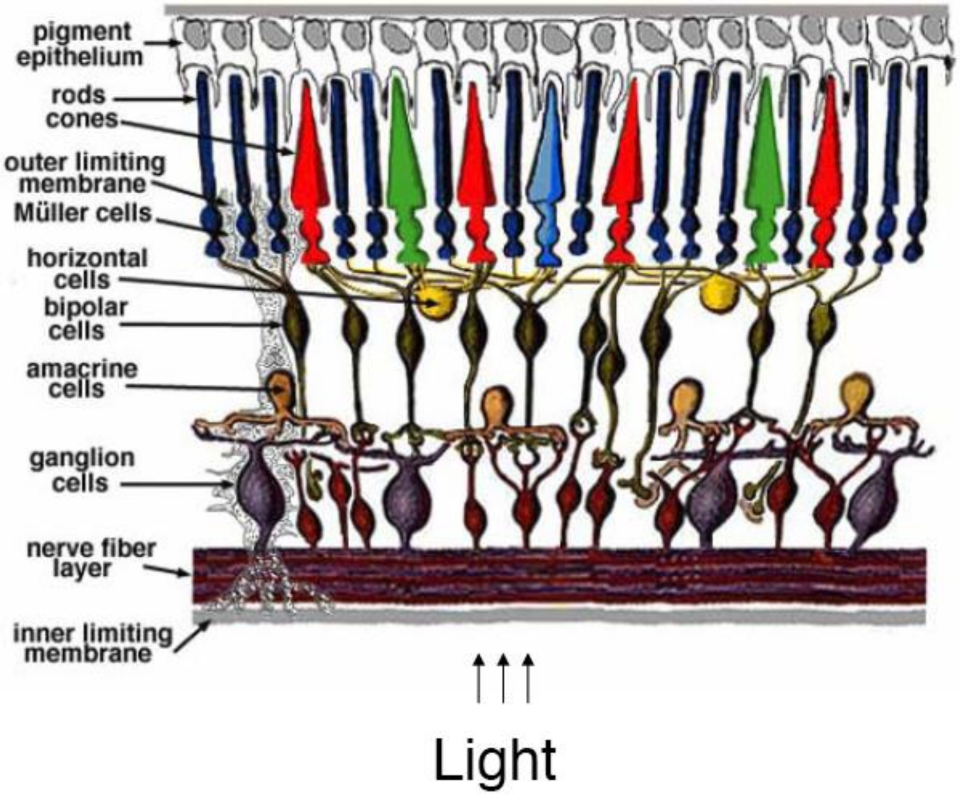
http://www.retinatexas.com/Anatomy_eng.html

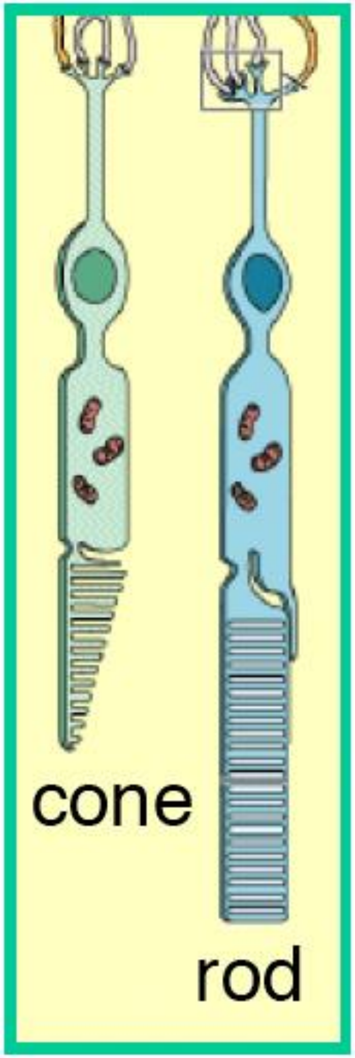
Photo-sensors

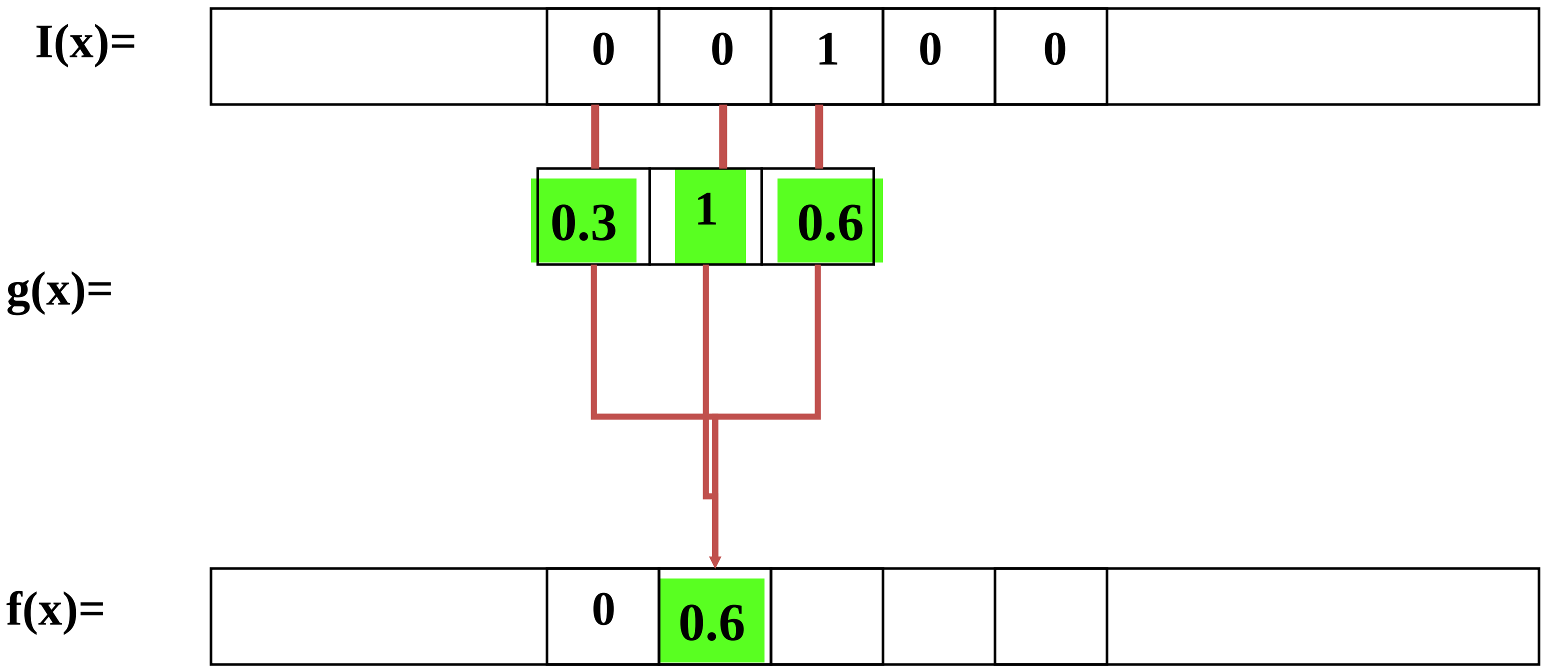


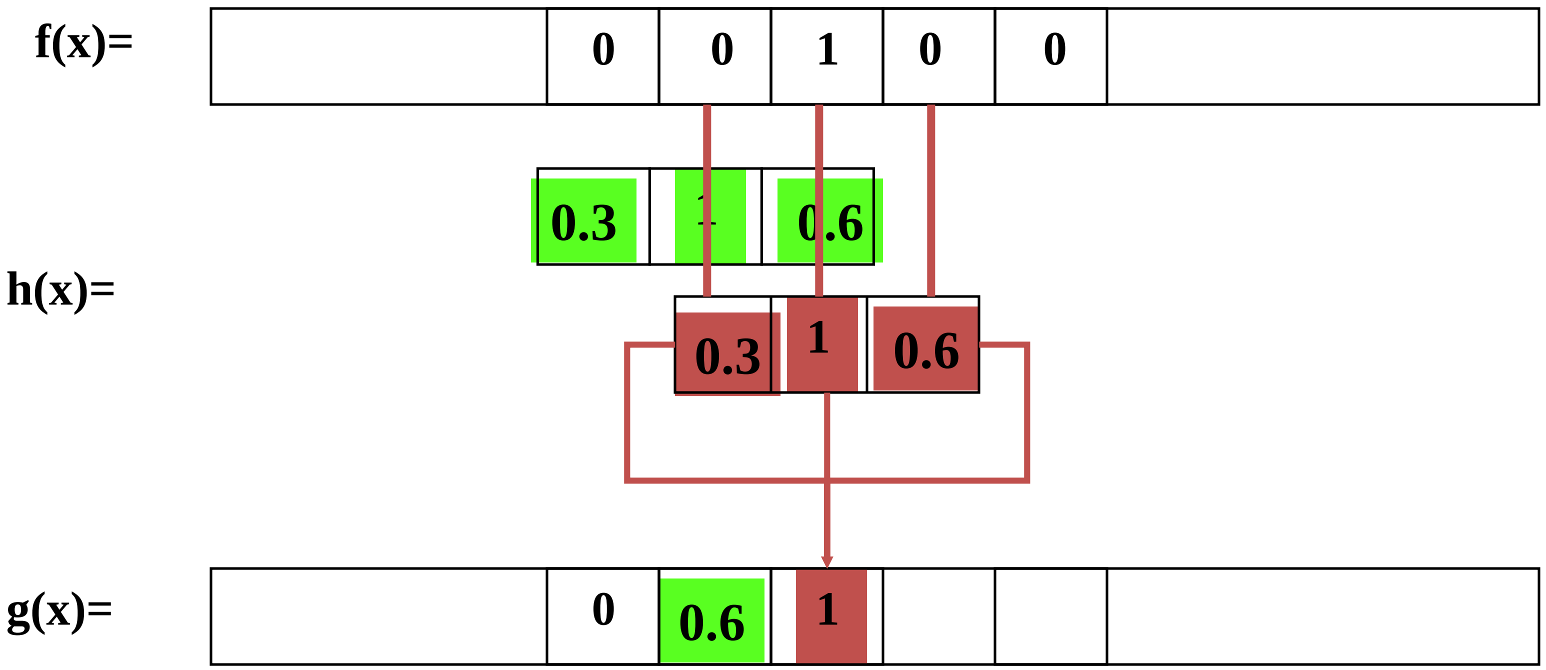
1) **Light pass through retina cells, excites Rod and Cone**
2) **Cone: color(spectral) sensitive, R,G,B, 6 Million**
3) **Rod: more photo sensitive, peak at 580nm(yellow), 120 M**
4) **What happens if you miss one type of Cone cells?**

Retina up-close

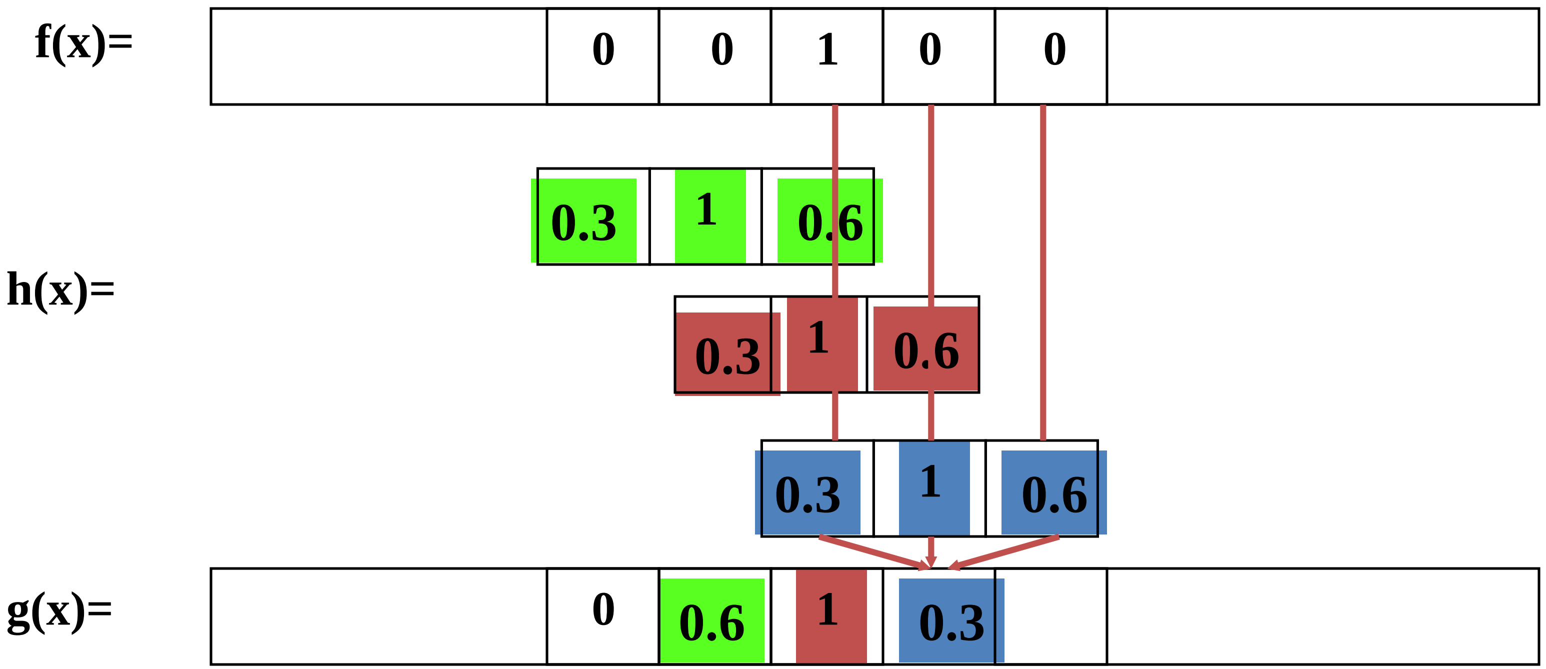




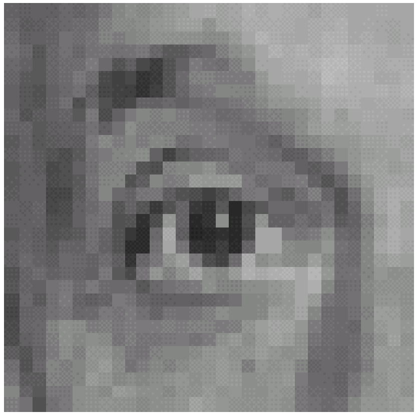

I(x)=
0
0
1
0
0
0.3
1
0.6
g(x)=
f(x)=
0
0.6



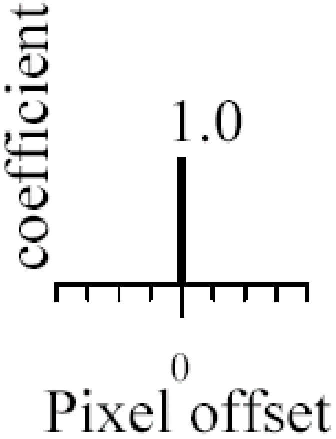
f(x)=
0
0
1
0
0
h(x)=
0.3
1
0.6
0.3
1
0.6
g(x)=
0
0.6
1


f(x)=
0
0
1
0
0
h(x)=
0.3
1
0.6
0.3
1
0.6
0.3
1
0.6
g(x)=
0
0.6
1
0.3

Linear filtering (warm-up slide)



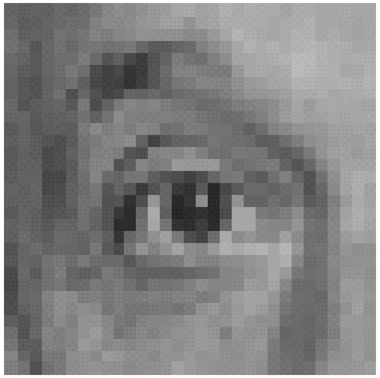
original



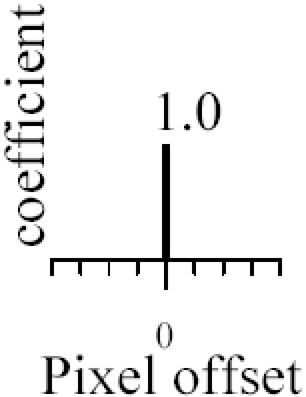
?

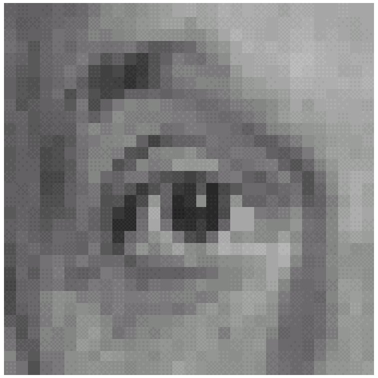
(Following examples taken from B. Freeman)

Linear filtering (warm-up slide)



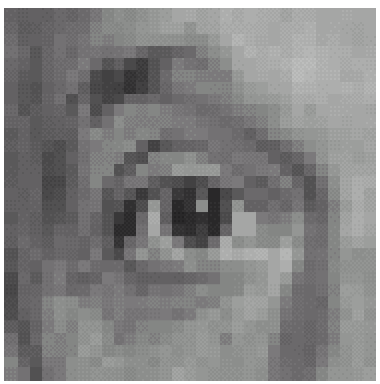
original



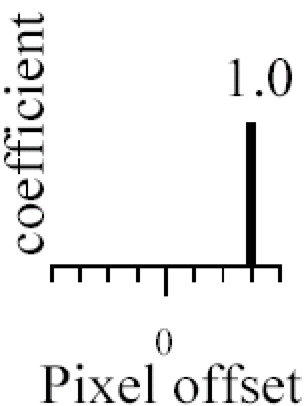


Filtered
(no change)

Linear filtering

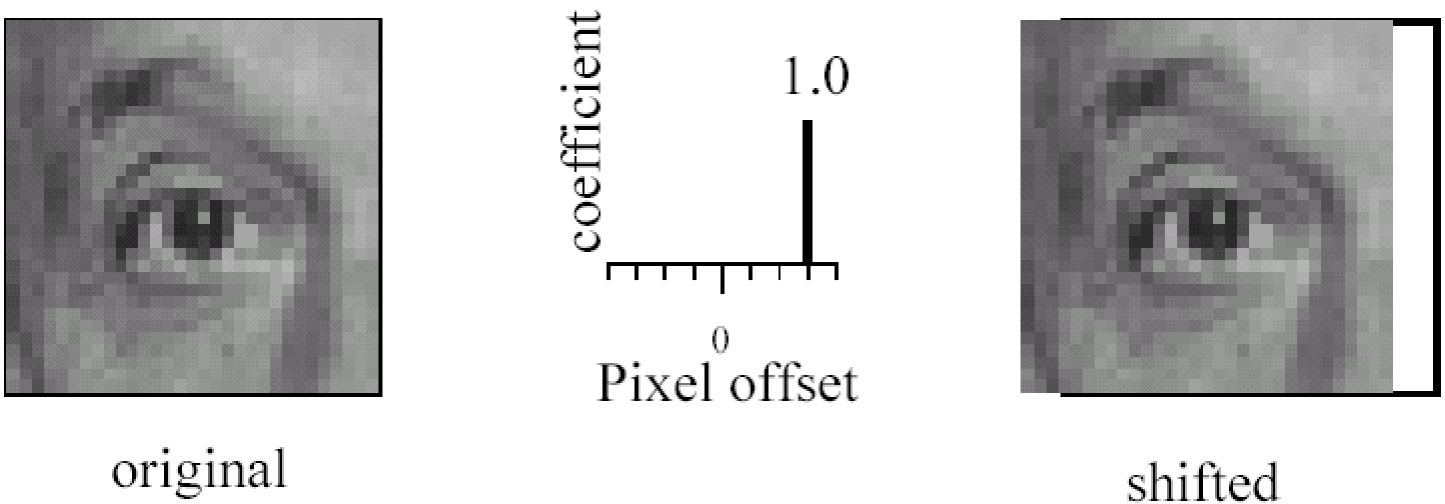


original

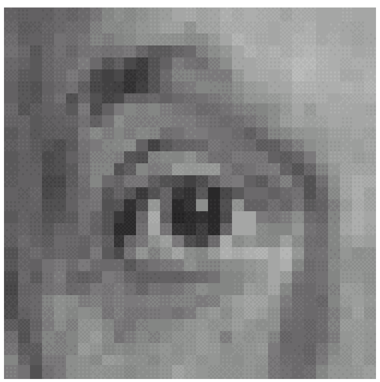


?

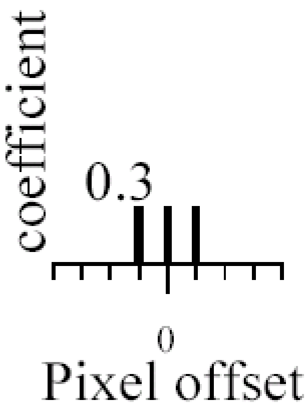
shift



Linear filtering

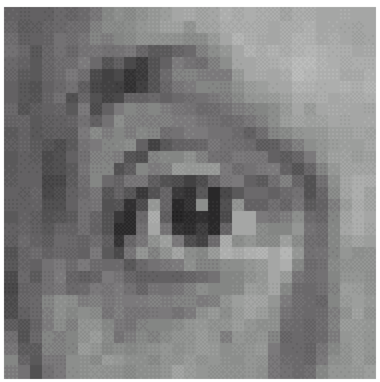


original

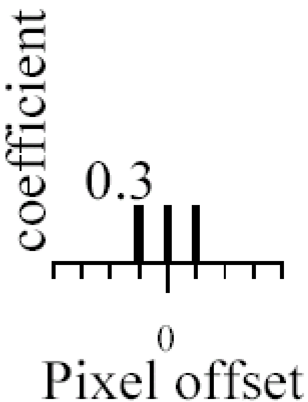


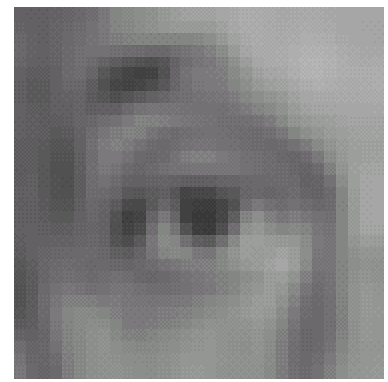
?

Blurring



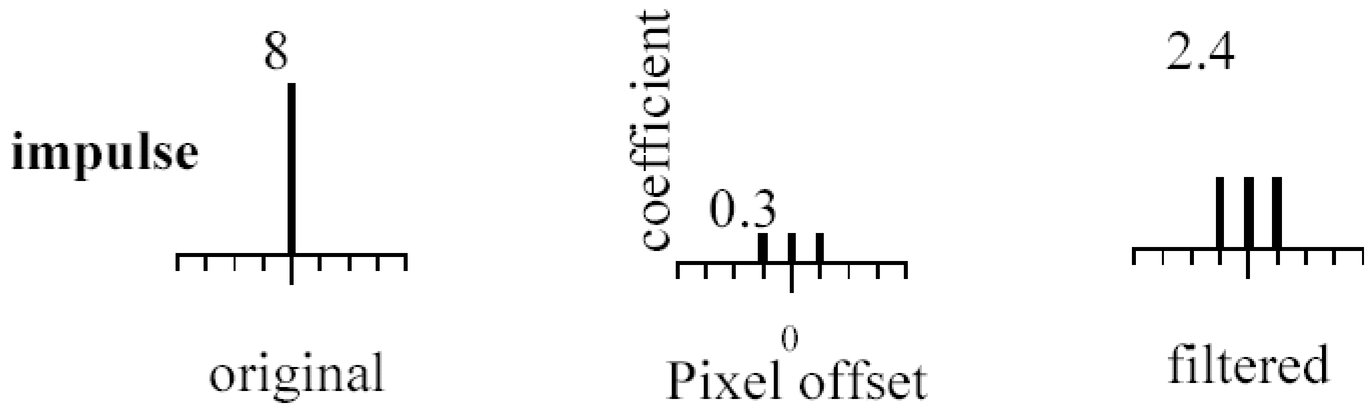
original



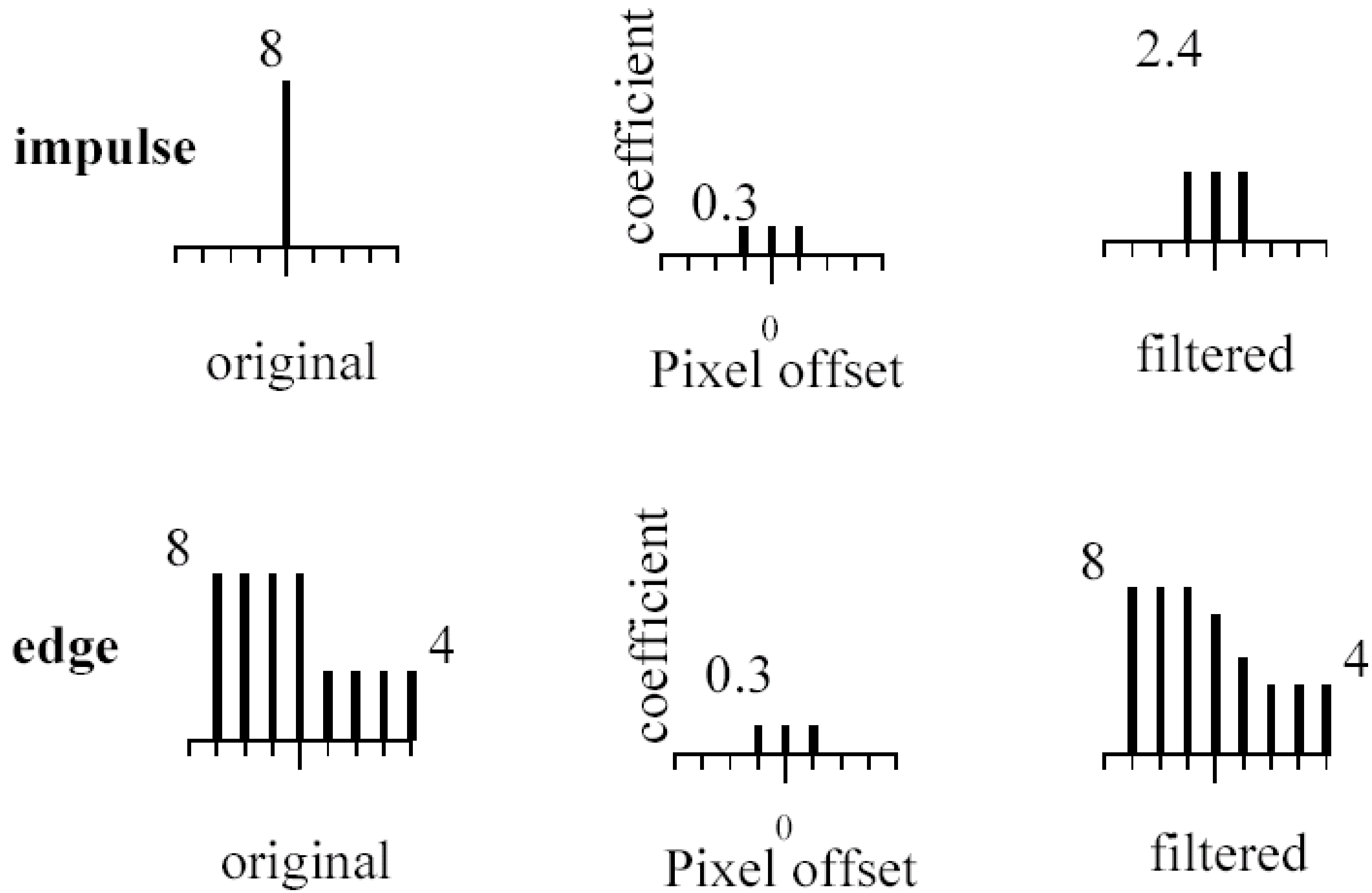


Blurred (filter applied in both dimensions).

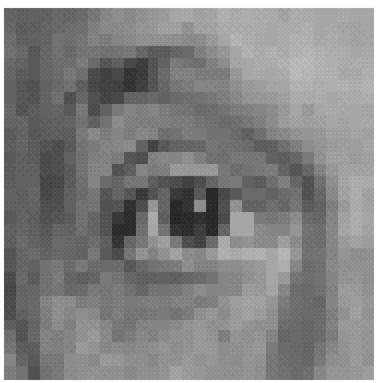
Blur examples



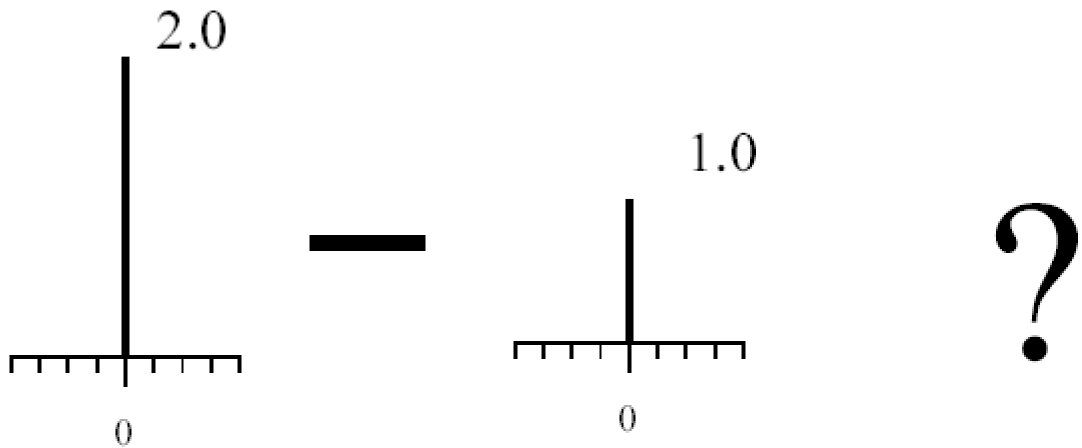
Blur examples



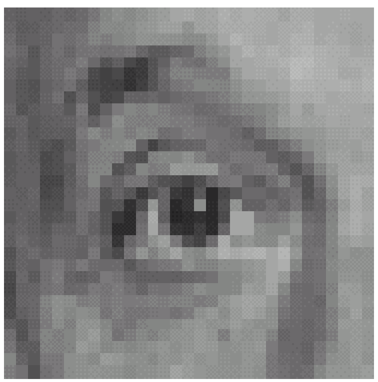
Linear filtering (warm-up slide)



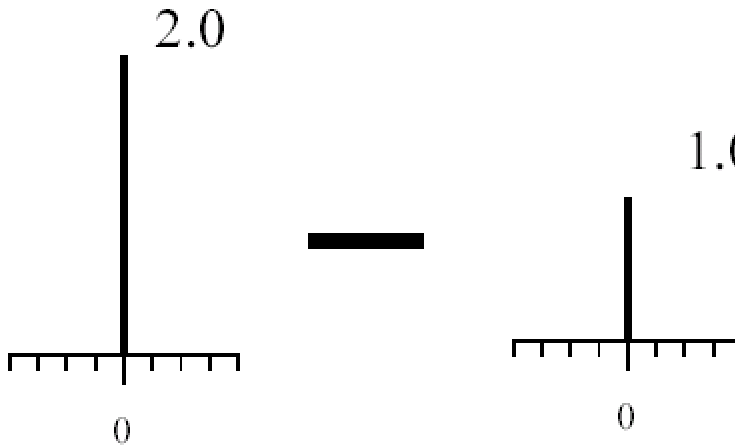
original

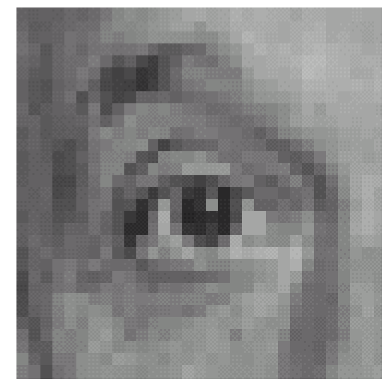


Linear filtering (no change)



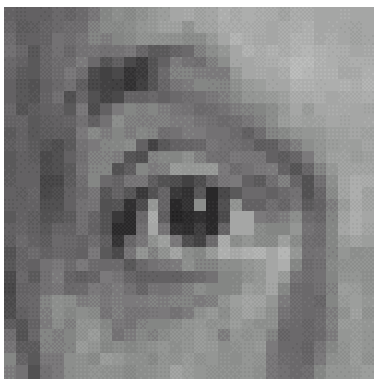
original



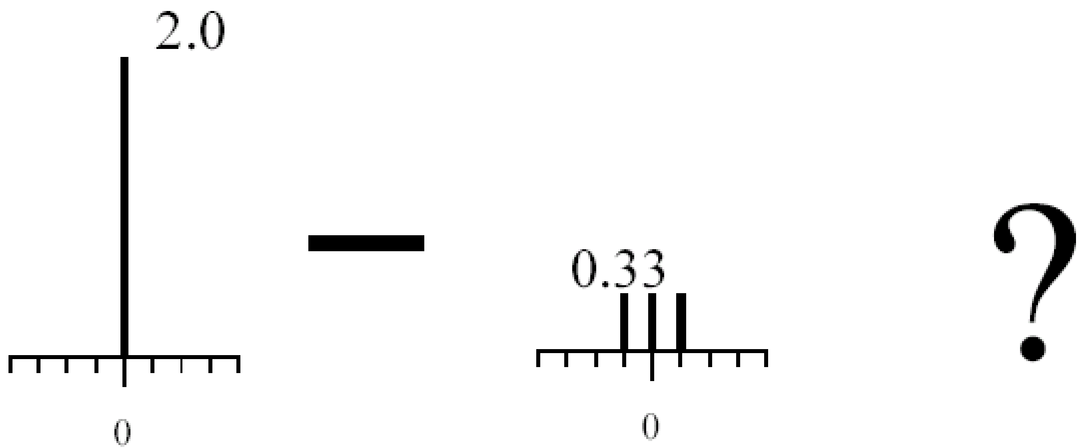


Filtered
(no change)

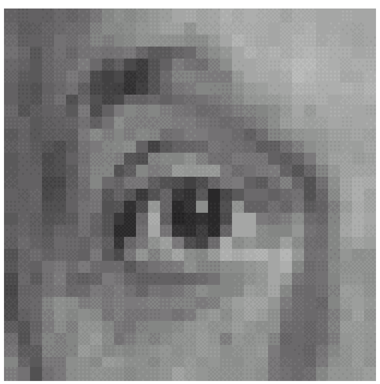
Linear filtering



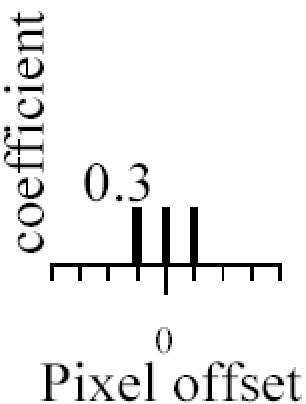
original

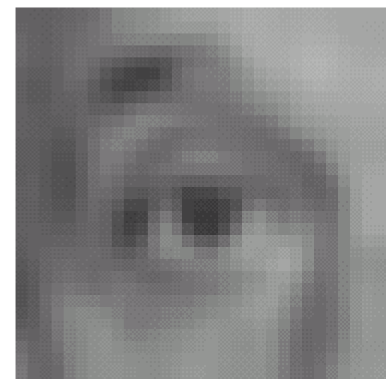


(remember blurring)



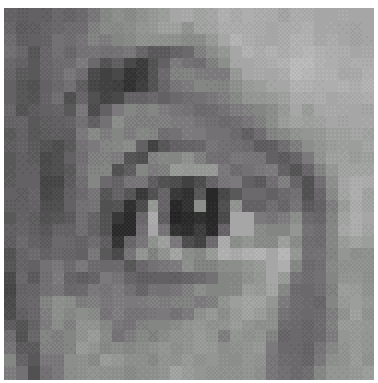
original



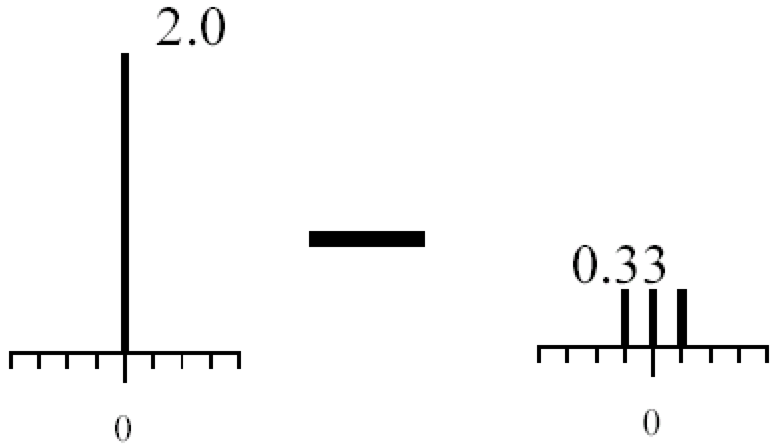


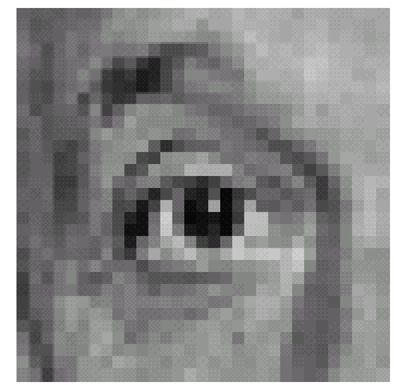
Blurred (filter applied in both dimensions).

Sharpening



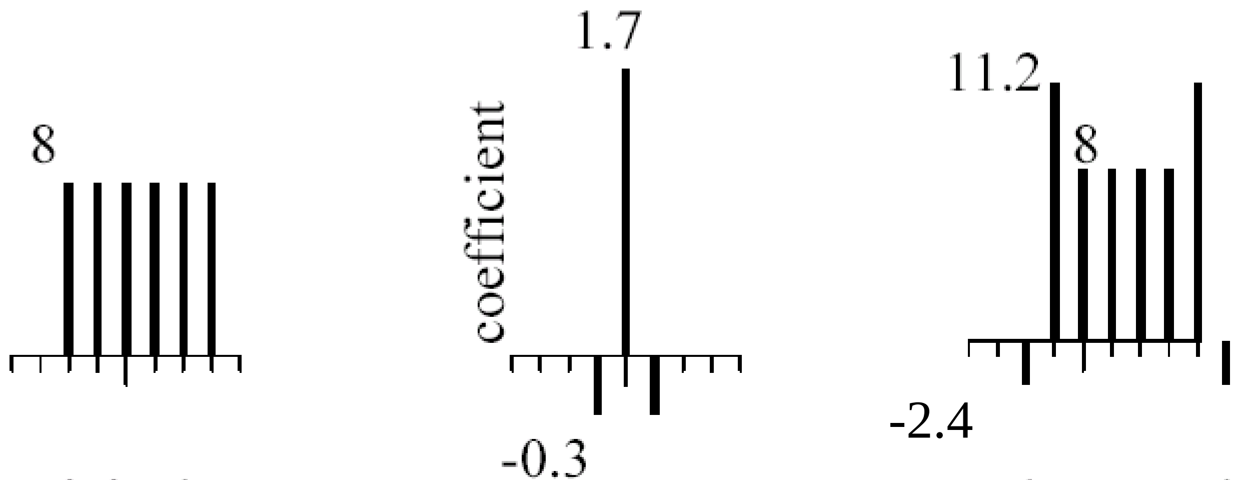
original





Sharpened
original

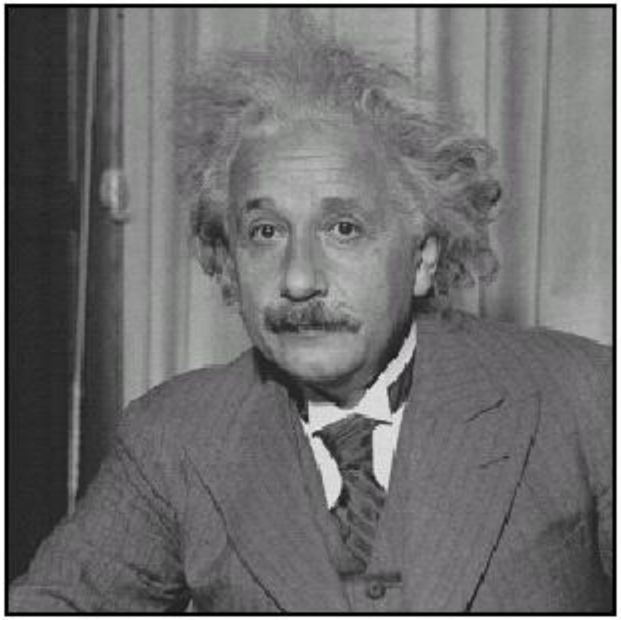
Sharpening example



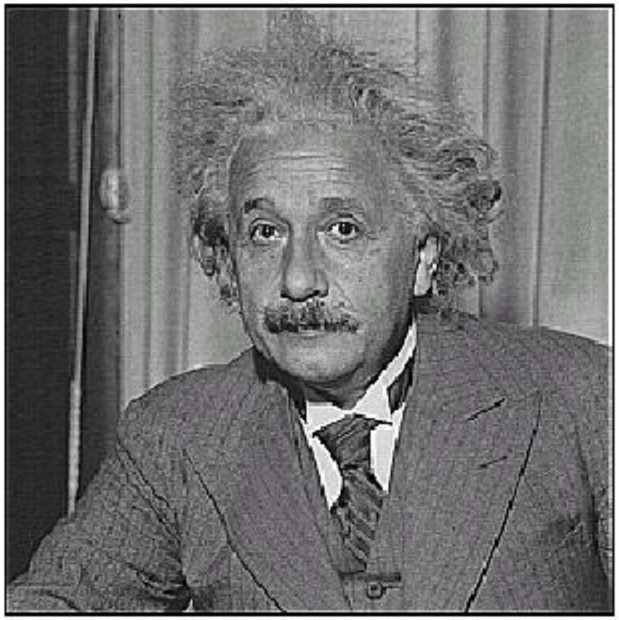
original

Sharpened
(differences are
accentuated; constant
areas are left untouched).

Sharpening



before



after

Image filtering

$$g[m,n] = \sum_{k,l} I(m+k, n+l) * f(k,l)$$

Output Image

Input Image

Kernal Image

Image filtering

$$g[m,n] = \sum_{k,l} I(m+k, n+l) * f(k,l)$$

Image I 8x8

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Kernel f 3x3

1	2	3
4	5	6
7	8	9

Output g

28	39	39	39	39	39	39	24
33	45	45	45	45	45	45	27
33	45	45	45	45	45	45	27
16	21	21	21	21	21	21	12
5	6	6	6	6	6	6	3
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Same position

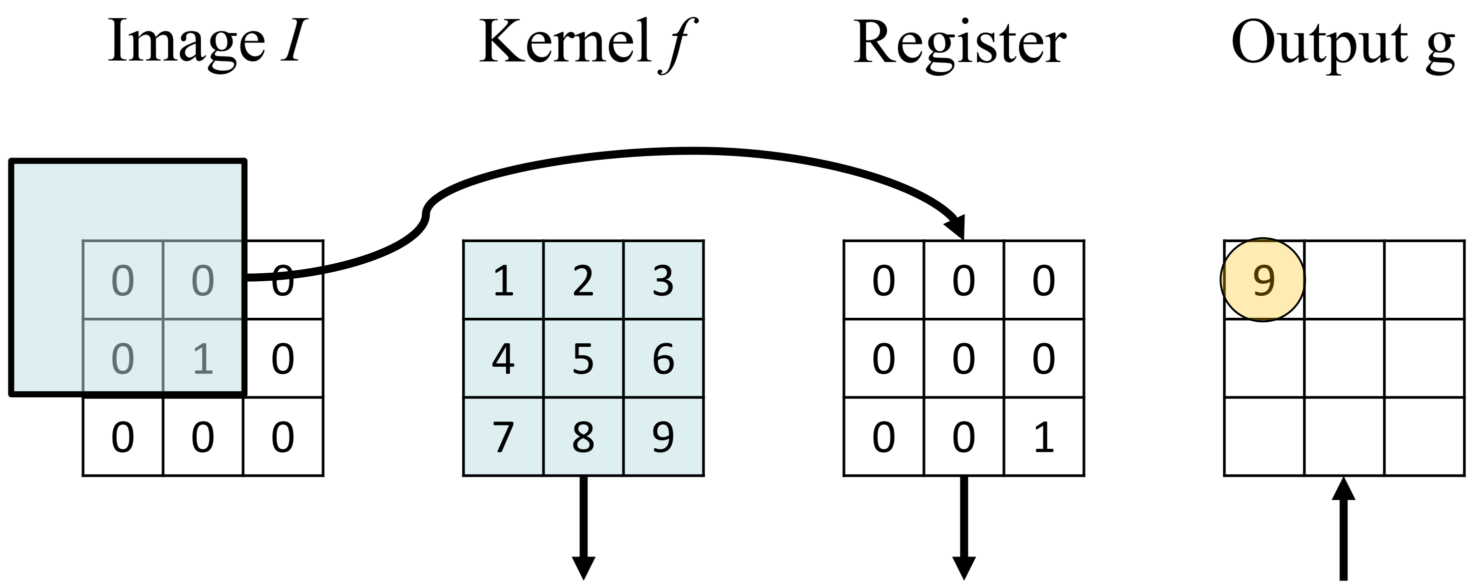
Register

a	b	c
d	e	f
g	h	i

Loop over every pixel (m,n)

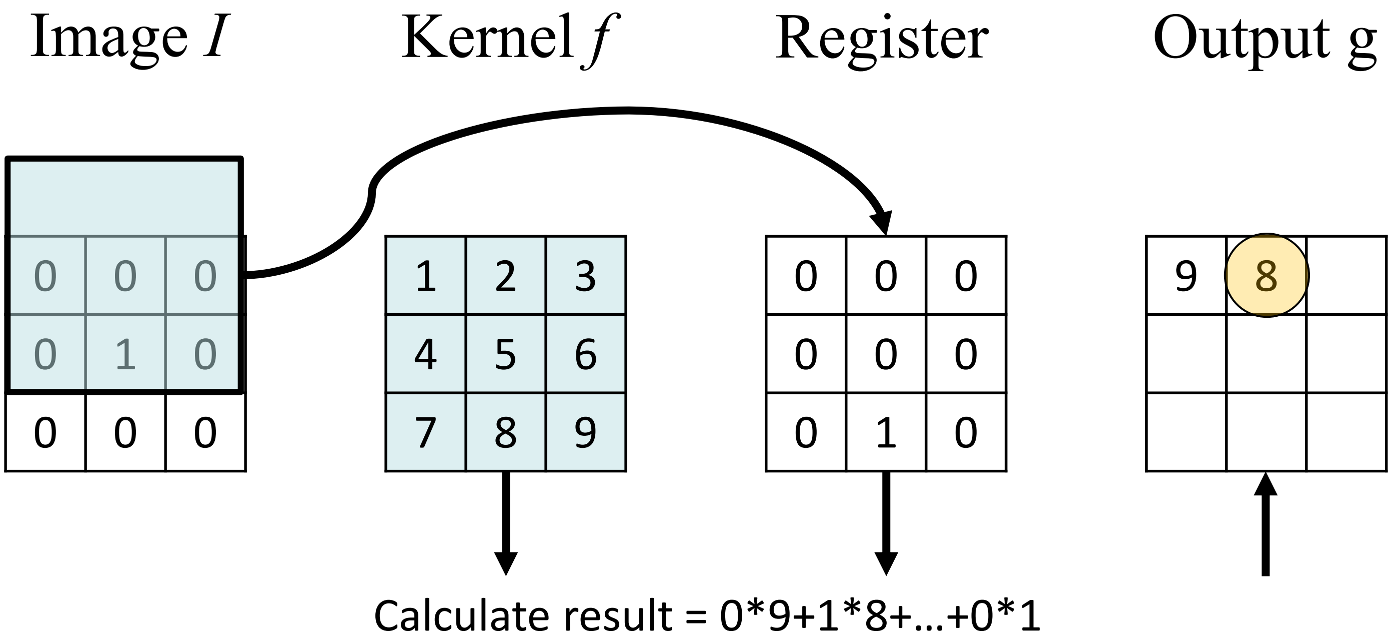
Calculate result = a*1+b*2+...+i*9

Special case: impulse function

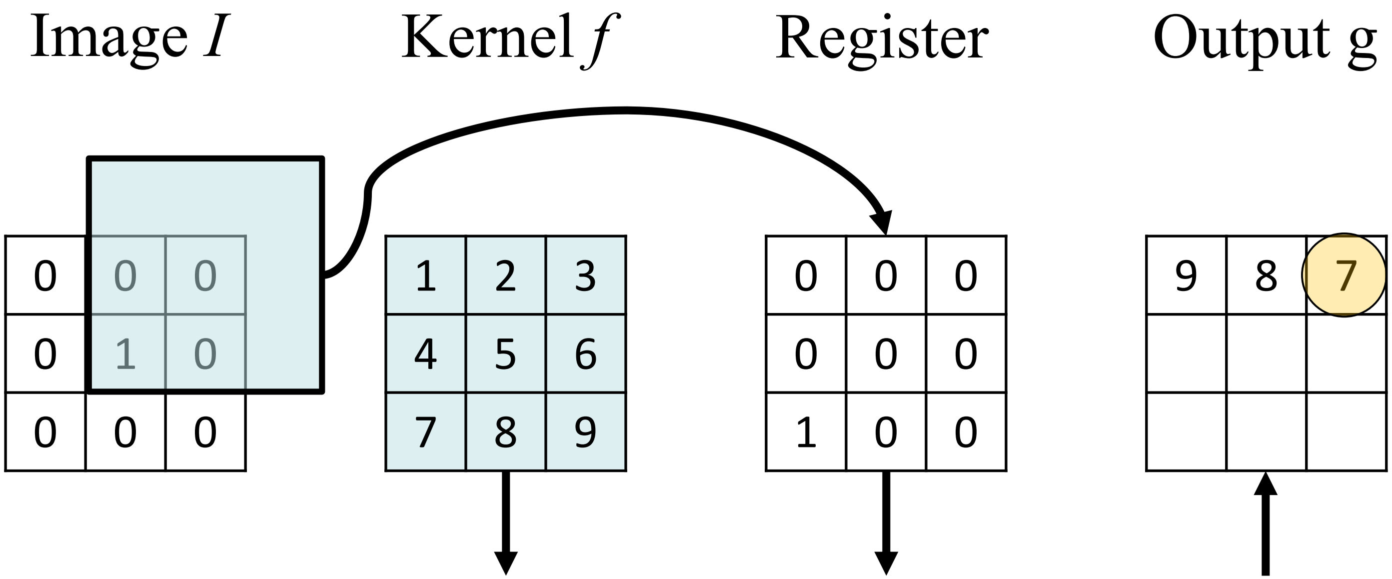


Calculate result = 1*9+0*8+…+0*1

Special case: impulse function

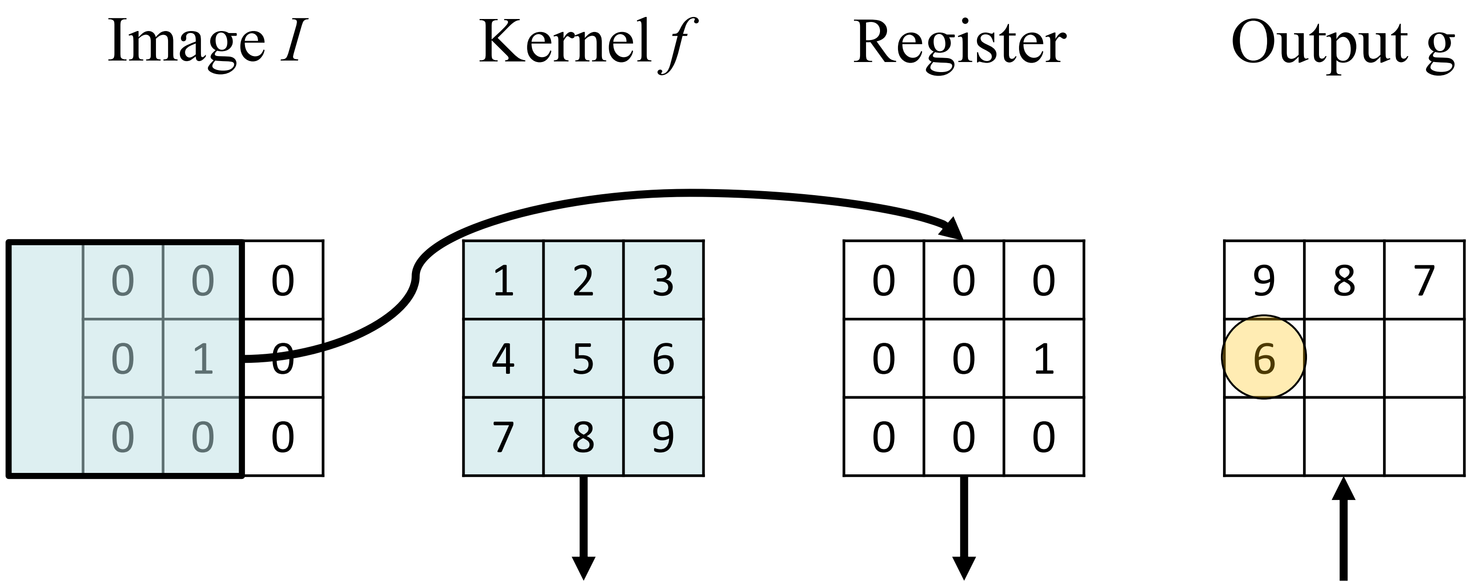


Special case: impulse function



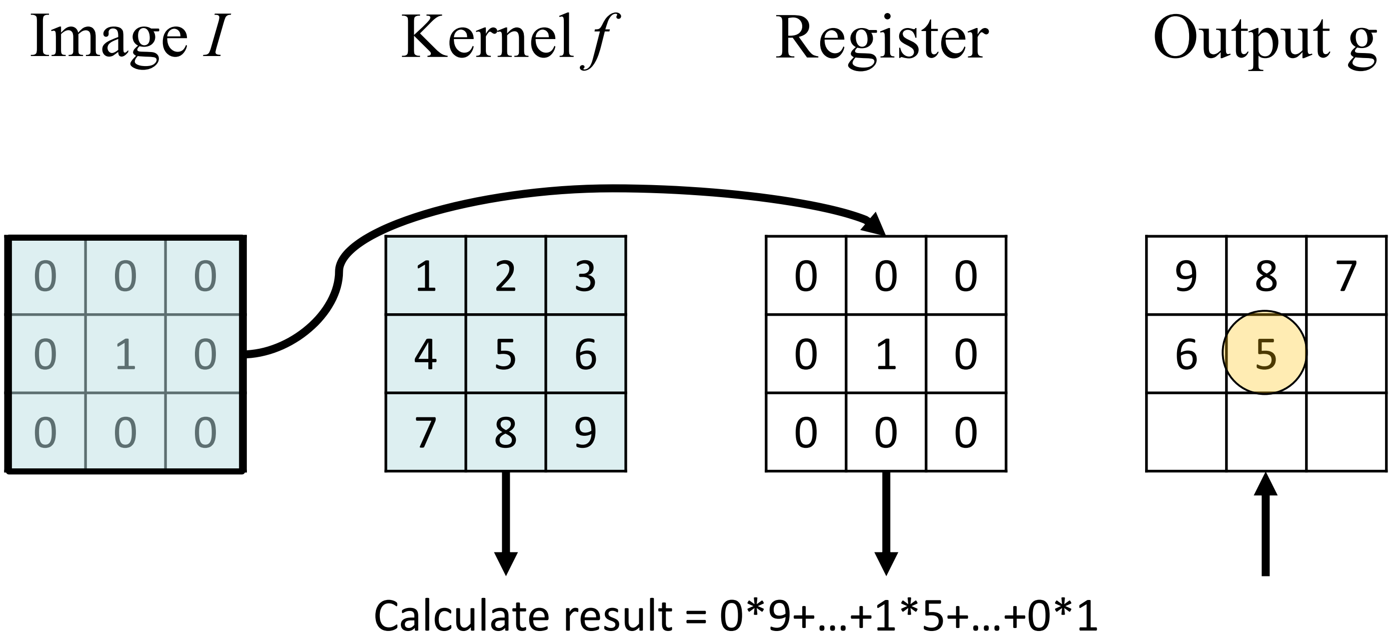
Calculate result = 0*9+…+1*7+…+0*1

Special case: impulse function

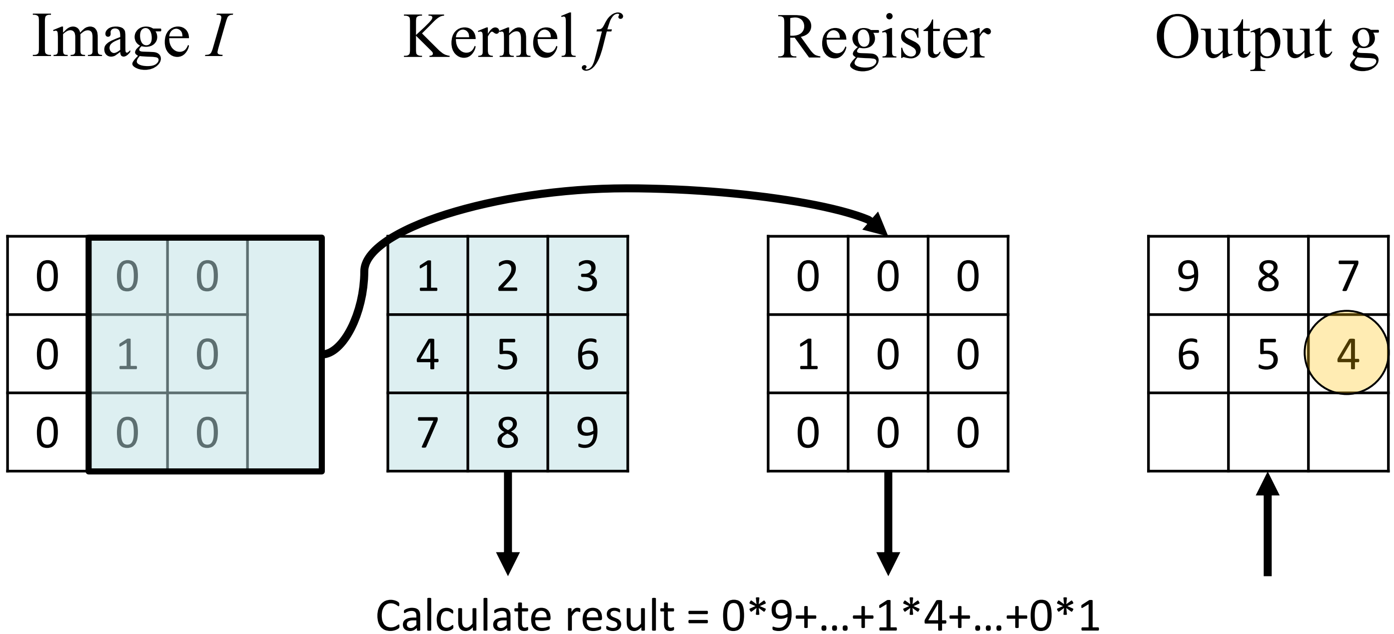


Calculate result = 0*9+…+1*6+…+0*1

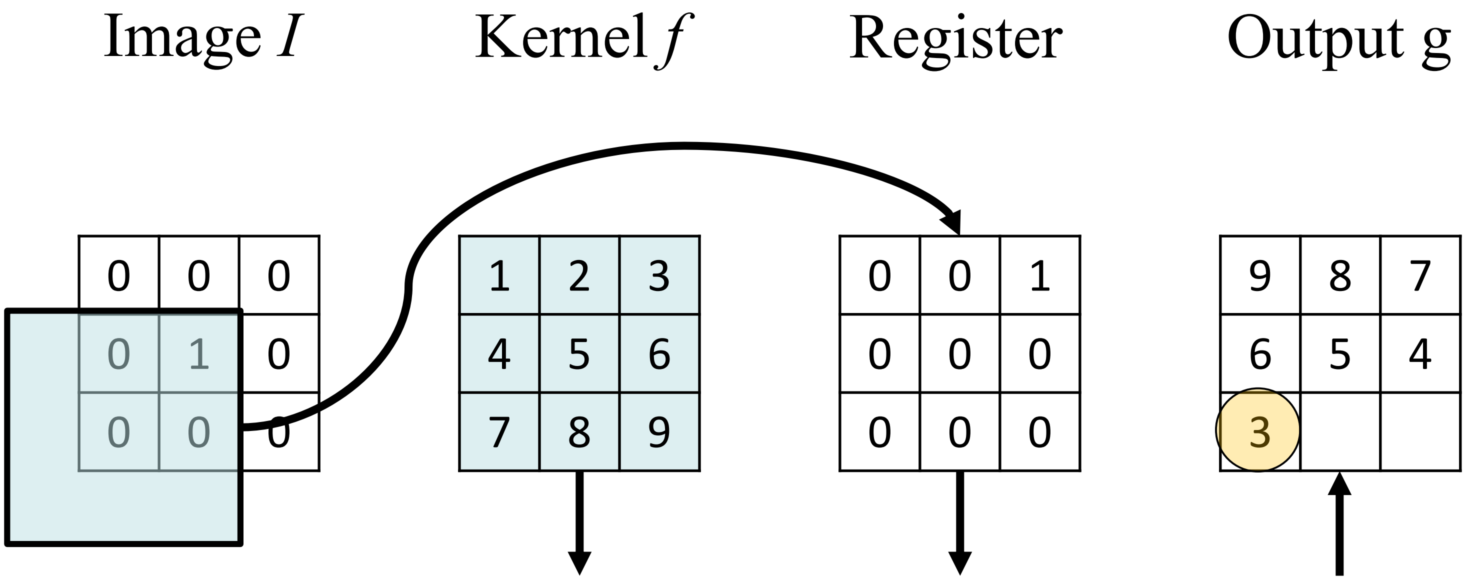
Special case: impulse function



Special case: impulse function

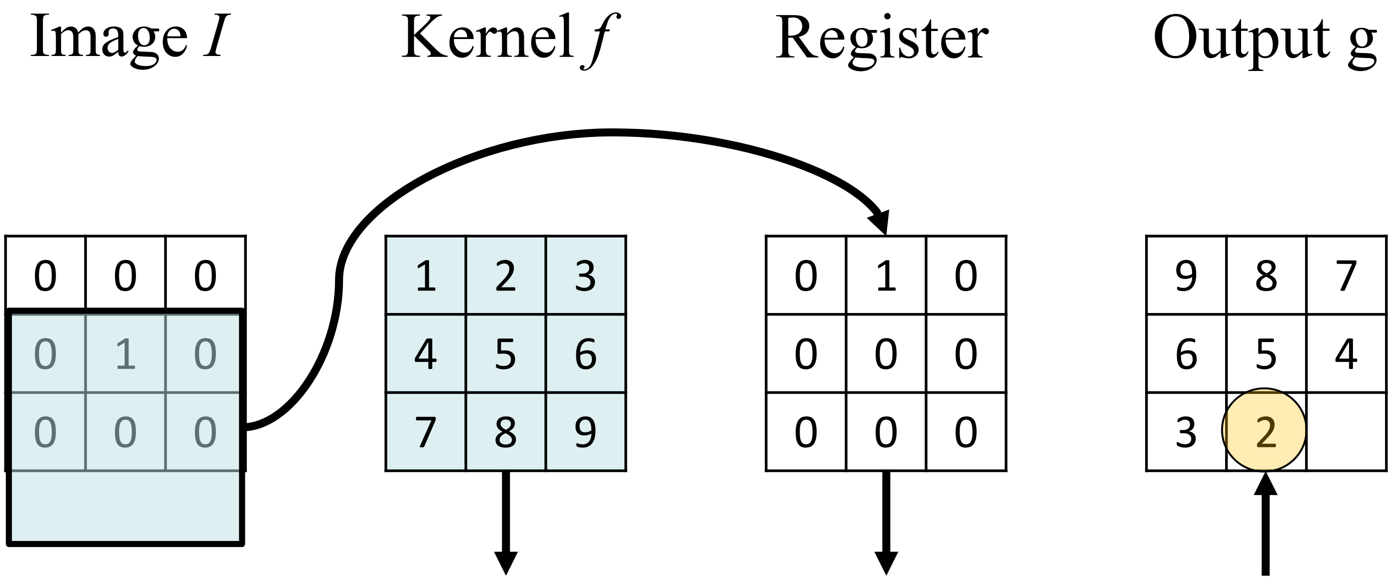


Special case: impulse function



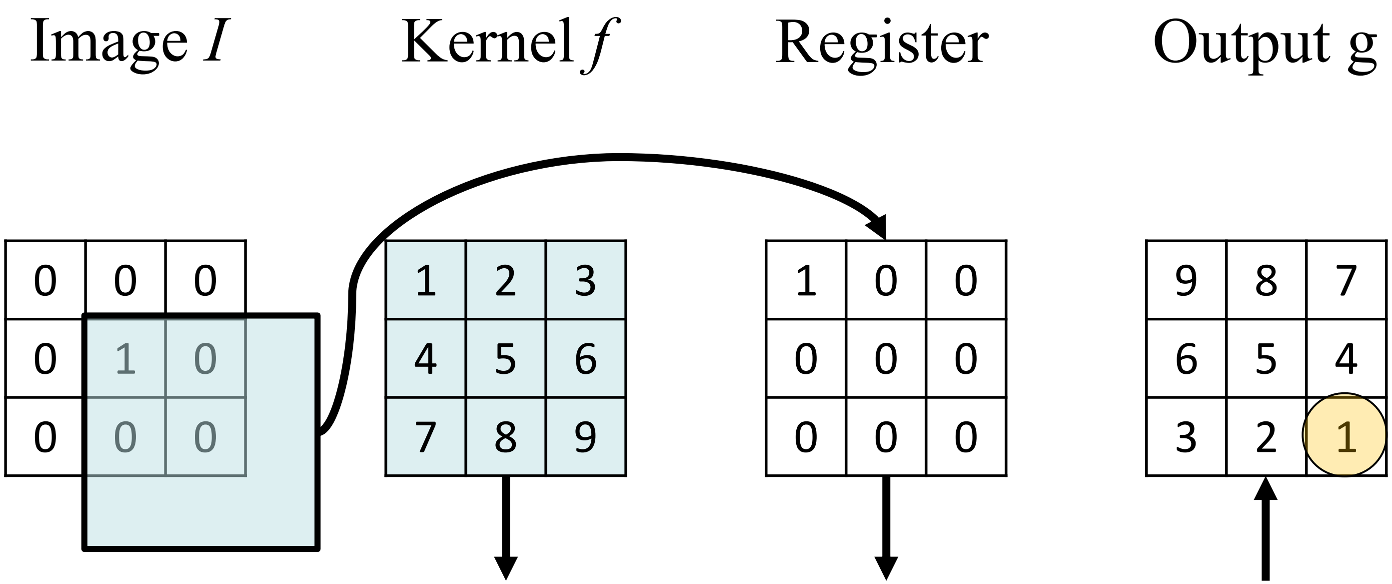
Calculate result = 0*9+…+1*3+…+0*1

Special case: impulse function



Calculate result = 0*9+…+1*2+0*1

Special case: impulse function



Calculate result = 0*9+…0*2+1*1

<Note> The output is the kernel flipped left-right, up-down!

Convolution

- Let I be an Signal(image), Convolution kernel f,

$$g[m,n] = I \otimes f = \sum_{k,l} I(m-k, n-l) * f(k,l)$$

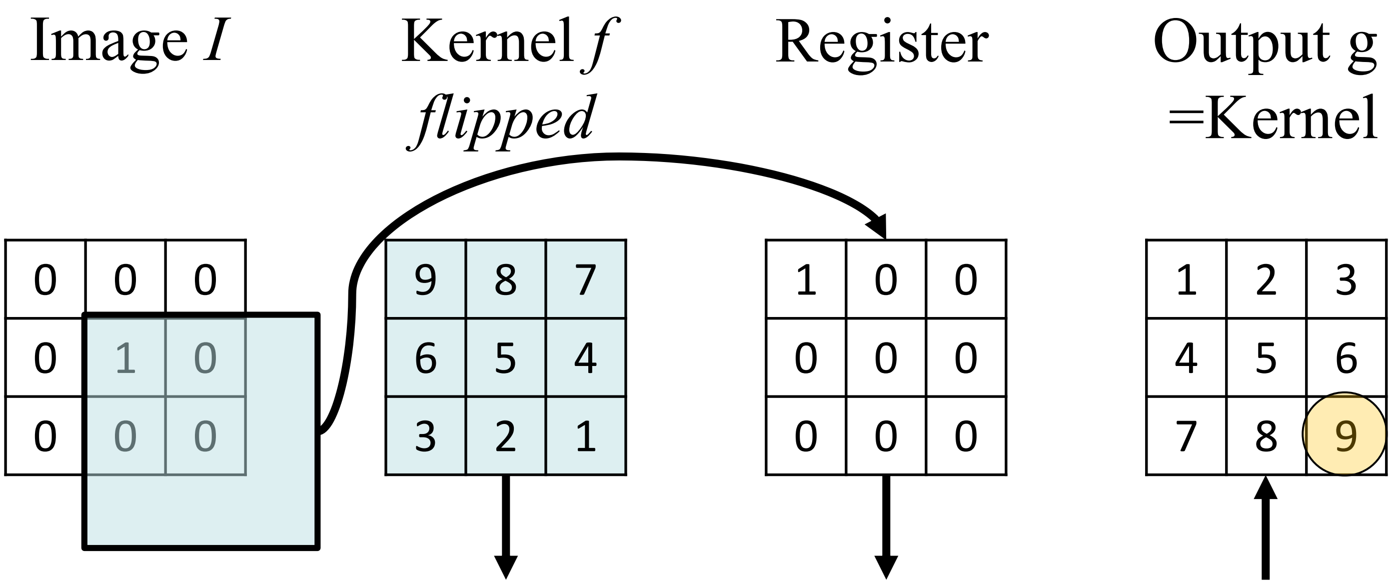
Output Image

Input Image

Kernal Image

Convolution

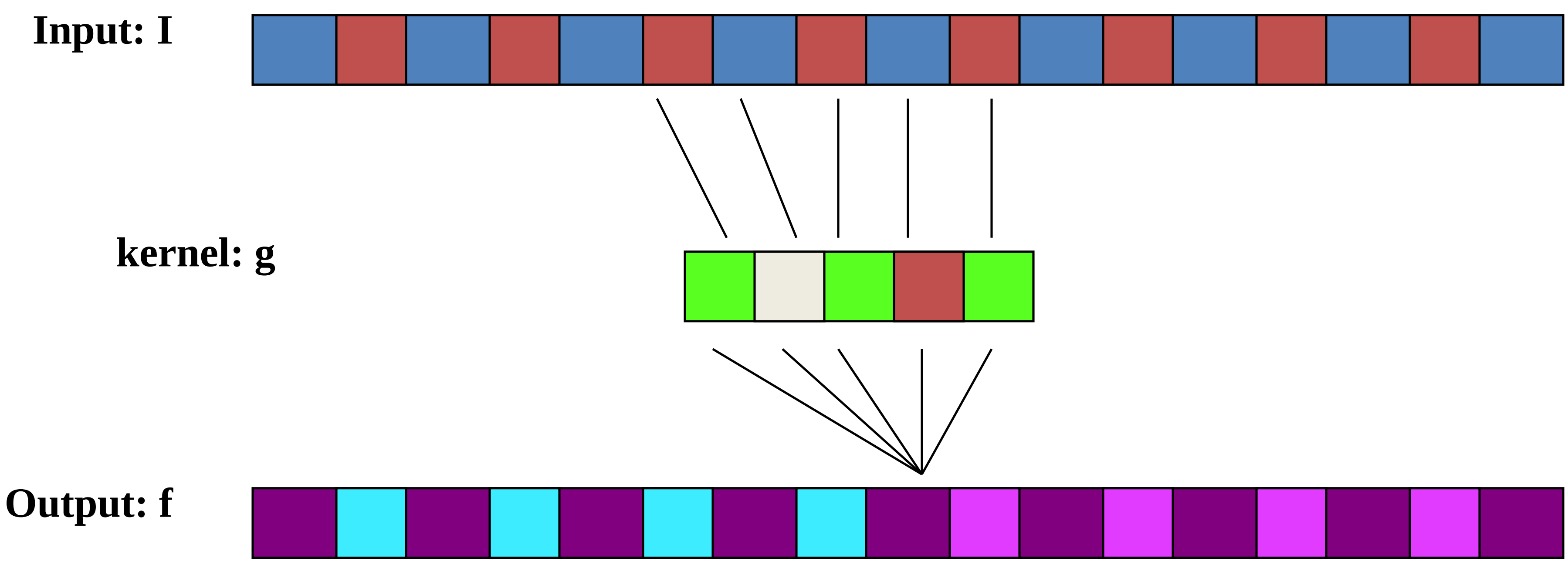
- $g[m,n] = I \otimes f = \sum_{k,l} I(m-k, n-l) * f(k,l)$
- Convolution is filtering with kernel flipped



Calculate result = 1*9+0*8+…+0*1

Image Convolution:

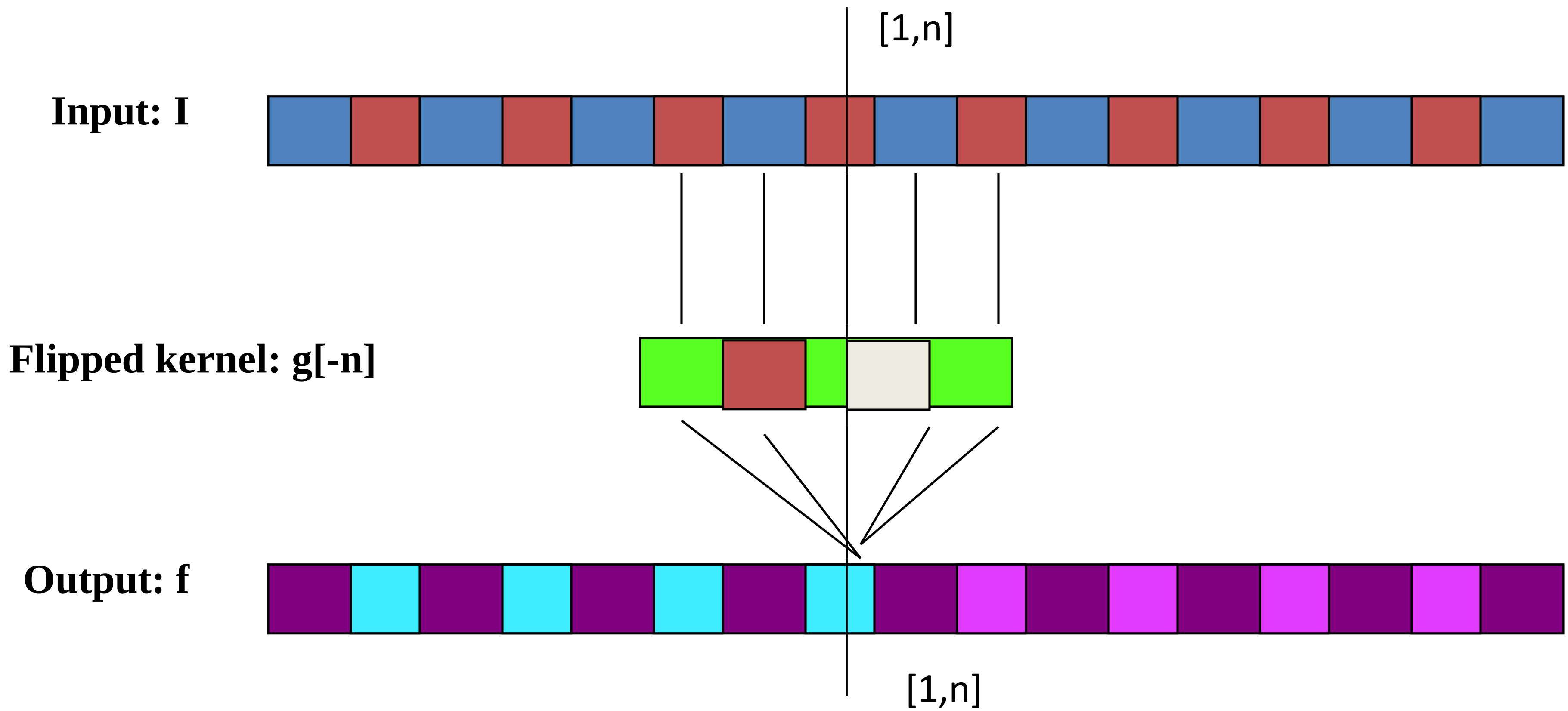
$$f[m,n] = I \otimes g = \sum_{k,l} I[m-k, n-l]\, g[k,l]$$



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

$$g'[k,l] = g[-k,-l]$$

$$= \sum_{k,l} I[m+k, n+l]\, g'[k,l]$$



Image

0	**0**	**0**	**0**	**0**
0	1	0	0	0
0	1	0	0	0
0	1	0	1	0
0	0	0	1	0

Kernel

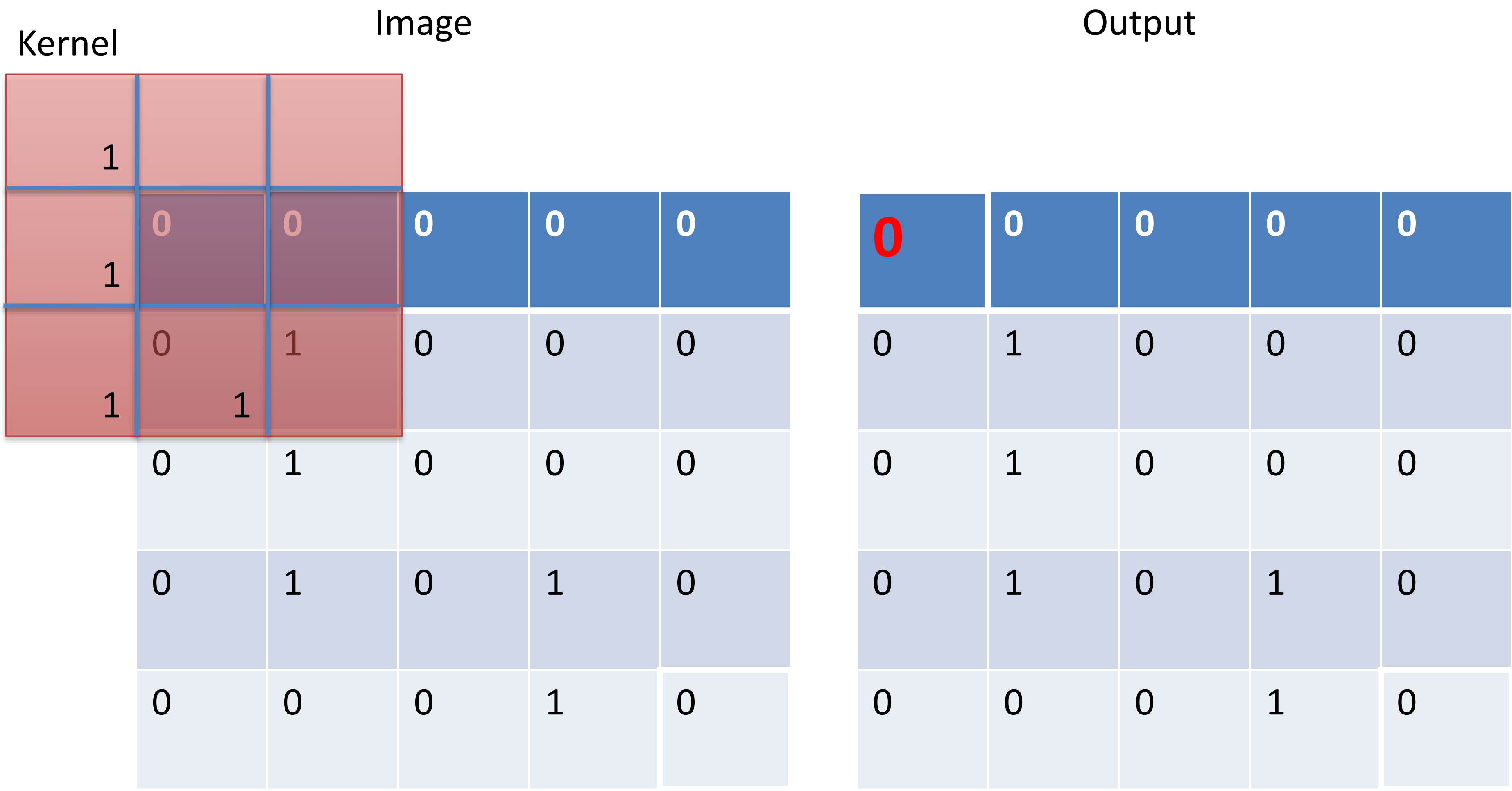
	1	1
		1
		1

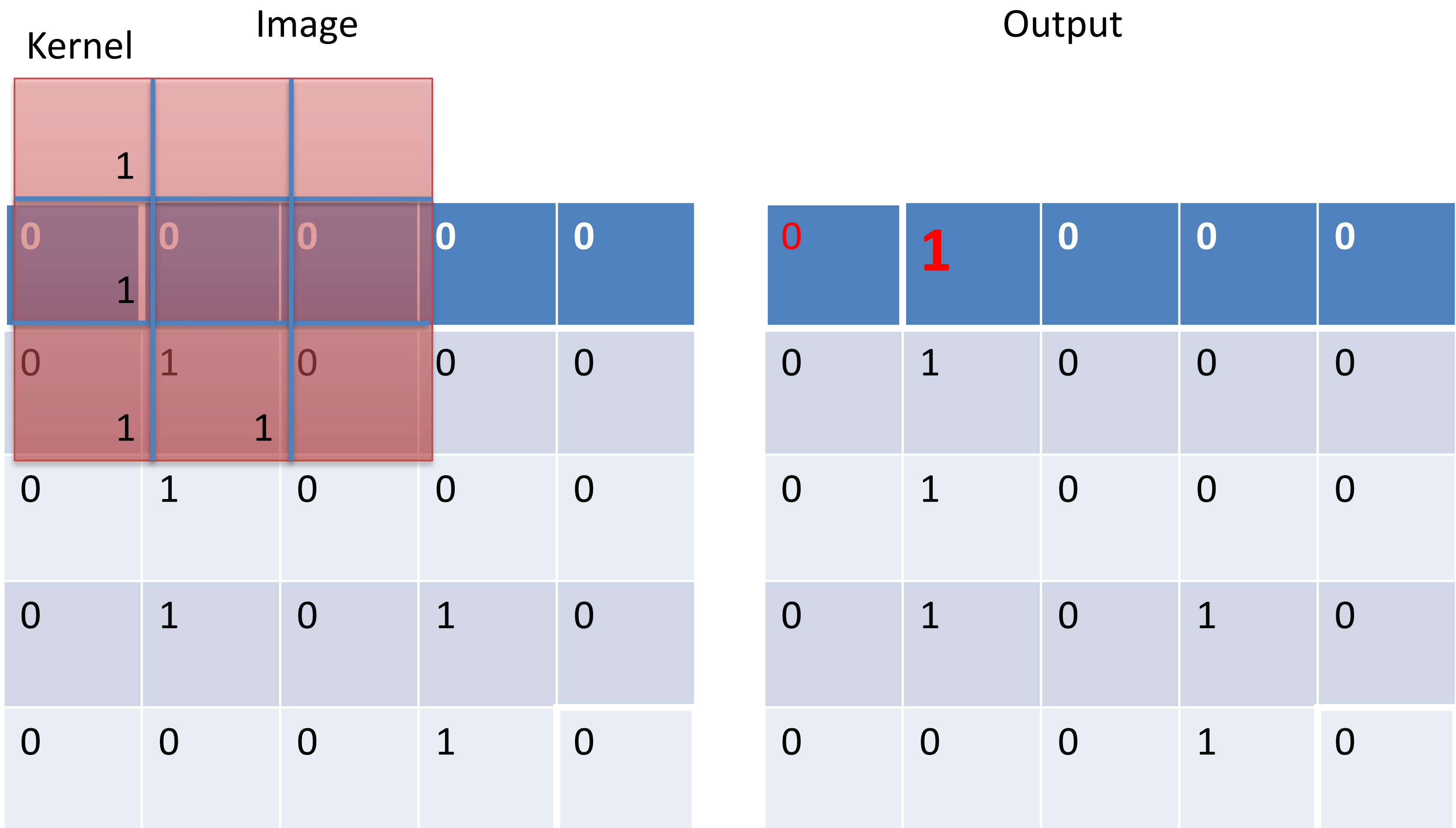
Image

0	0	0	0	0
0	1	0	0	0
0	1	0	0	0
0	1	0	1	0
0	0	0	1	0

Flipped Kernel

1		
1		
1	1	


Kernel
Image
Output
1
1 0 0 0 0 0
1 0 1 1 0 0 0
0 1 0 0 0
0 1 0 1 0
0 0 0 1 0
0 0 0 0 0
0 1 0 0 0
0 1 0 0 0
0 1 0 1 0
0 0 0 1 0



Kernel
Image
Output
1
0 0 0 0 0
1
0 1 0 0 0
1 1
0 1 0 0 0
0 1 0 1 0
0 0 0 1 0
0 1 0 0 0
0 1 0 0 0
0 1 0 0 0
0 1 0 1 0
0 0 0 1 0

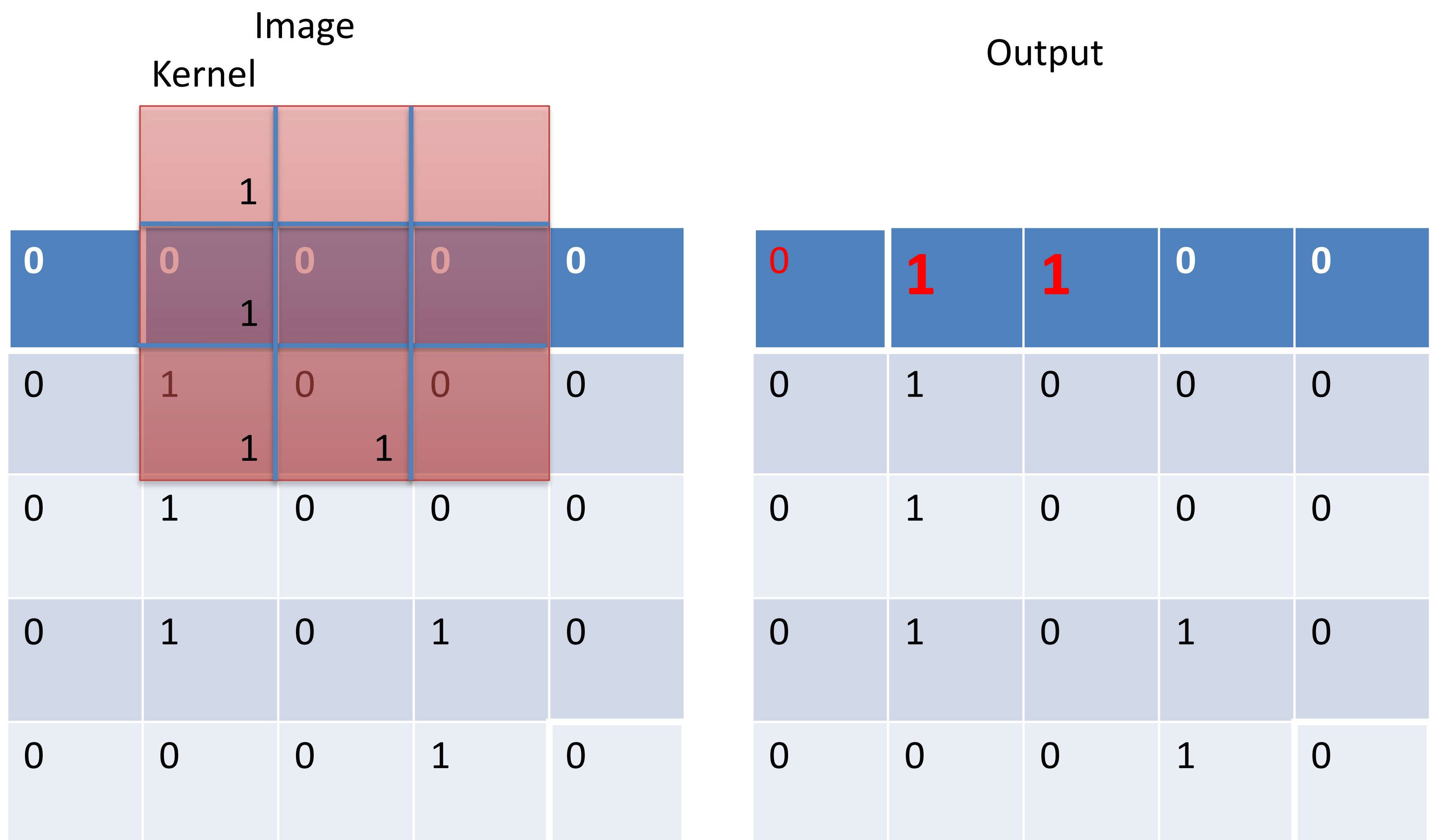


Image
Kernel
Output
1
0 0 0 0 0
1
0 1 0 0 0
1 1
0 1 0 0 0
0 1 0 1 0
0 0 0 1 0
0 1 1 0 0
0 1 0 0 0
0 1 0 0 0
0 1 0 1 0
0 0 0 1 0

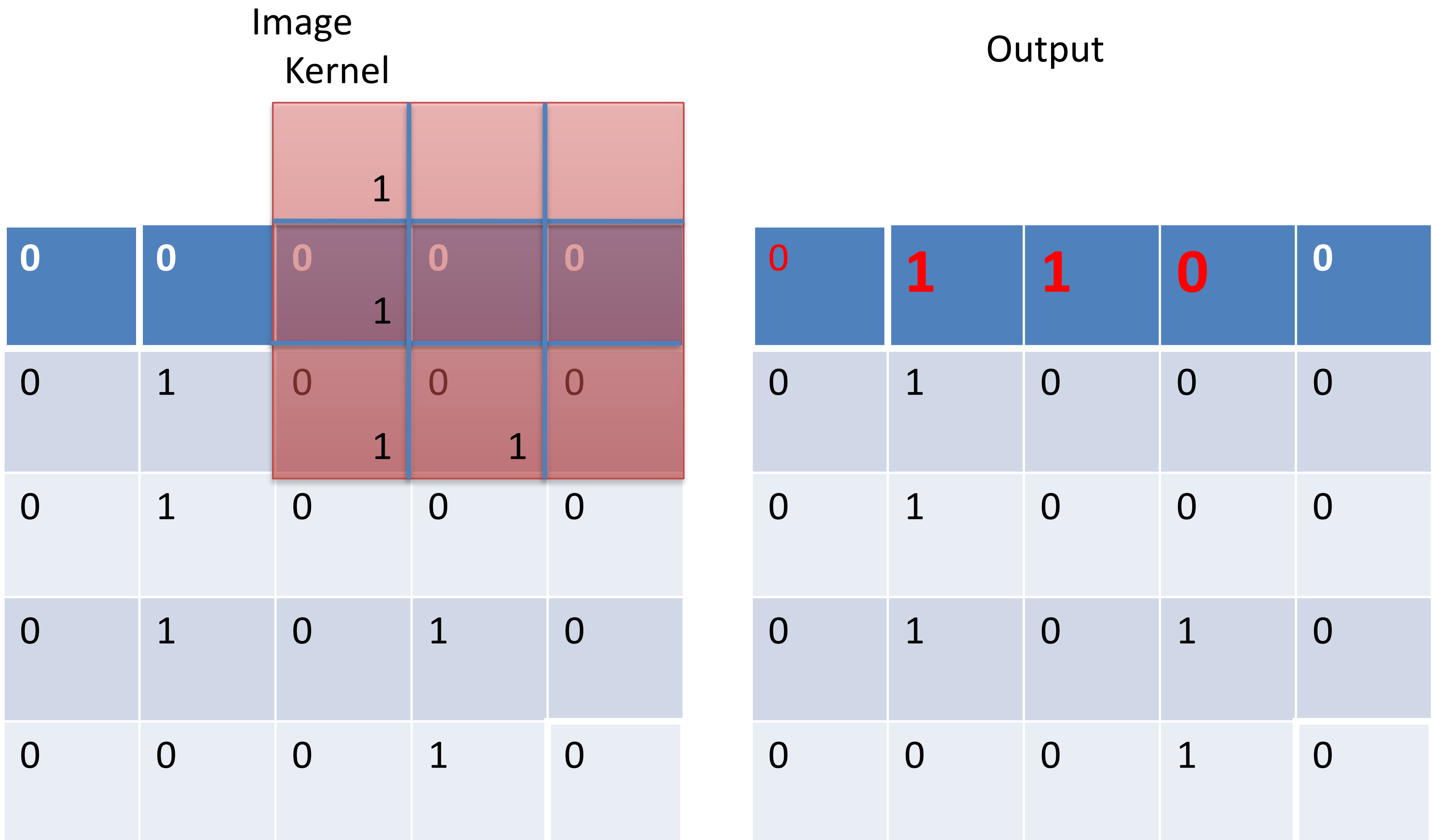


Image
Kernel
Output
1
0 0 0 0 0
1
0 1 0 0 0
1 1
0 1 0 0 0
0 1 0 1 0
0 0 0 1 0
0 1 1 0 0
0 1 0 0 0
0 1 0 0 0
0 1 0 1 0
0 0 0 1 0

Image

Kernel

0	0	0	0	0
0	1	0	0	0
0	1	0	0	0
0	1	0	1	0
0	0	0	1	0

1		
1		
1	1	

Output

0	1	1	0	0
0	1	2	0	0
0	1	0	0	0
0	1	0	1	0
0	0	0	1	0

Image

Kernel

0	0	0	0	0
0	1	0	0	0
0	1	0	0	0
0	1	0	1	0
0	0	0	1	0

1		
1		
1	1	

Output

0	1	1	0	0
0	1	2	0	0
0	1	0	0	0
0	1	0	1	0
0	0	0	1	0

Image

0 Kernel	0	0	0	0
0 1	1	0	0	0
0 1	1	0	0	0
0 1	1 1	0	1	0
0	0	0	1	0

Output

0	**1**	**1**	**0**	**0**
0	1	2	0	0
0	1	0	0	0
0	1	0	1	0
0	0	0	1	0

Image

0	0 Kernel	0	0	0
0	1 1	0	0	0
0	1 1	0	0	0
0	1 1	0 1	1	0
0	0	0	1	0

Output

0	**1**	**1**	**0**	**0**
0	1	2	0	0
0	1	3	0	0
0	1	0	1	0
0	0	0	1	0

Image

0	0	0 Kernel	0	0
0	1	0 1	0	0
0	1	0 1	0	0
0	1	0 1	1 1	0
0	0	0	1	0

Output

0	1	1	0	0
0	1	2	0	0
0	1	3	1	0
0	1	0	1	0
0	0	0	1	0

Image

0	0	0	0	0
0	1	0	0	0
0	1	0	0	0
0	1	0	1	0
0	0	0	1	0

Kernel

1		
1		
1	1	

Output

0	1	1	0	0
0	1	2	0	0
0	1	3	1	1
0	1	0	1	0
0	0	0	1	0

Image

0	0	0	0	0
0	1	0	0	0
0	1	0	0	0
0	1	0	1	0
0	0	0	1	0

Kernel

1		
1		
1	1	

Output

0	1	1	0	0
0	1	2	0	0
0	1	3	1	1
0	0	2	1	2
0	0	1	1	0

Image

0	0	0	0	0
0	1	0	0	0
0	1	0	0	0
0	1	0	1	0
0	0	0	1	0

Kernel

1		
1		
1	1	

Output

0	1	1	0	0
0	1	2	0	0
0	1	3	1	1
0	0	2	1	2
0	0	1	0	2

Kernel

	1	1
		1
		1

Image

0	0	0	0	0
0	1	0	0	0
0	1	0	0	0
0	1	0	1	0
0	0	0	1	0

Ouput

0	1	1	0	0
0	1	2	0	0
0	1	3	1	1
0	0	2	1	2
0	0	1	0	2

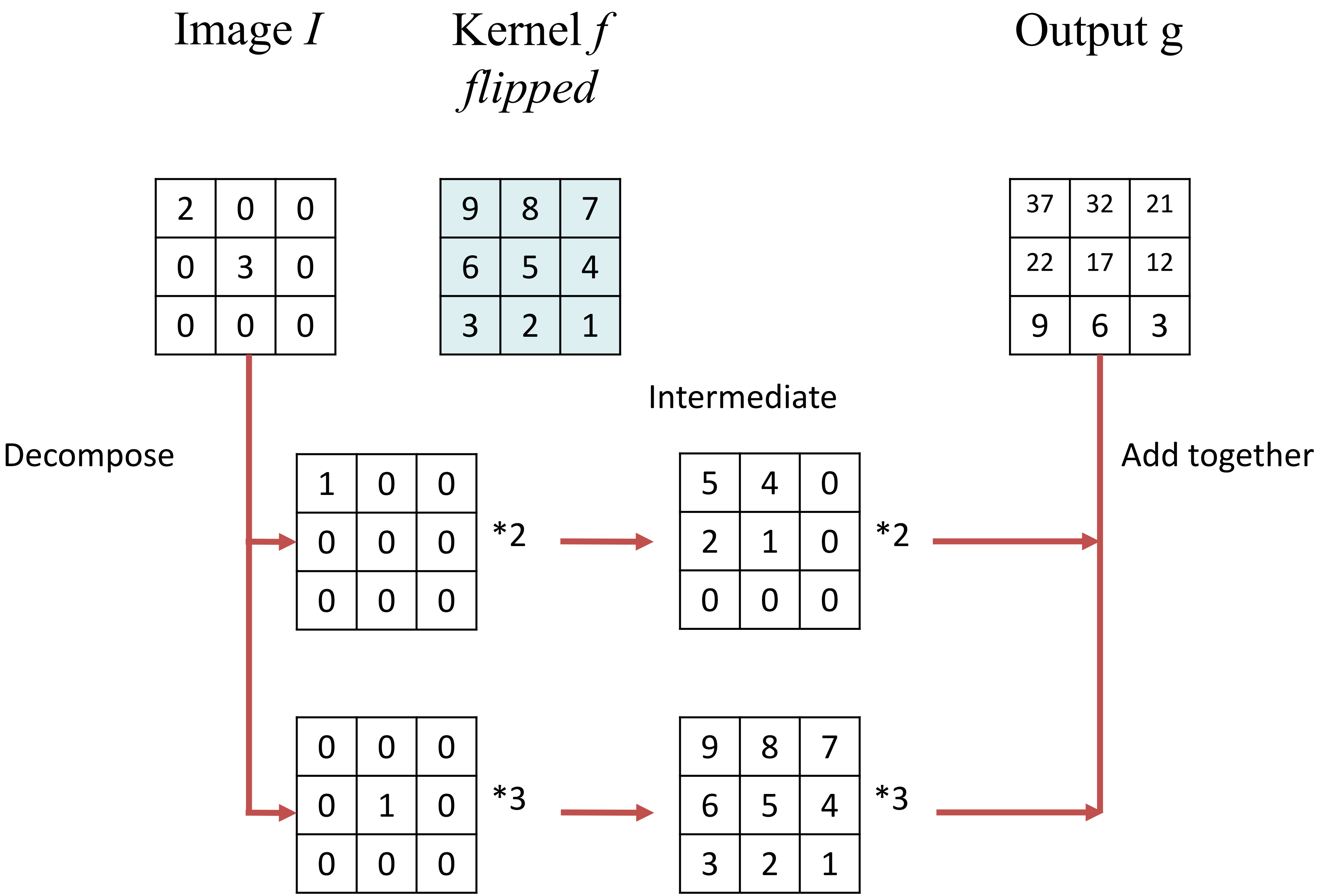


Image I
Kernel f flipped
Output g
Intermediate
Decompose
Add together
*2
*3
*2
*3

- Convolution has commutative property $f \otimes I$

Image I

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

Image I^{flip}

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

Kernel f

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

Decompose

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} e & f & 0 \\ h & i & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \bullet \begin{bmatrix} i & h & g \\ f & e & d \\ c & b & a \end{bmatrix} \rightarrow \begin{bmatrix} b & c & 0 \\ e & f & 0 \\ h & i & 0 \end{bmatrix}$$

Image I^{flip}

(Flipped)

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 0 \\ b & c & 0 \\ e & f & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 0 \\ a & b & c \\ d & e & f \end{bmatrix}$$

Impulse functions shift images

Image *I*

a	b	c
d	e	f
g	h	i

Kernel *f*

1	0	0
0	0	0
0	0	0

Kernel *f'*

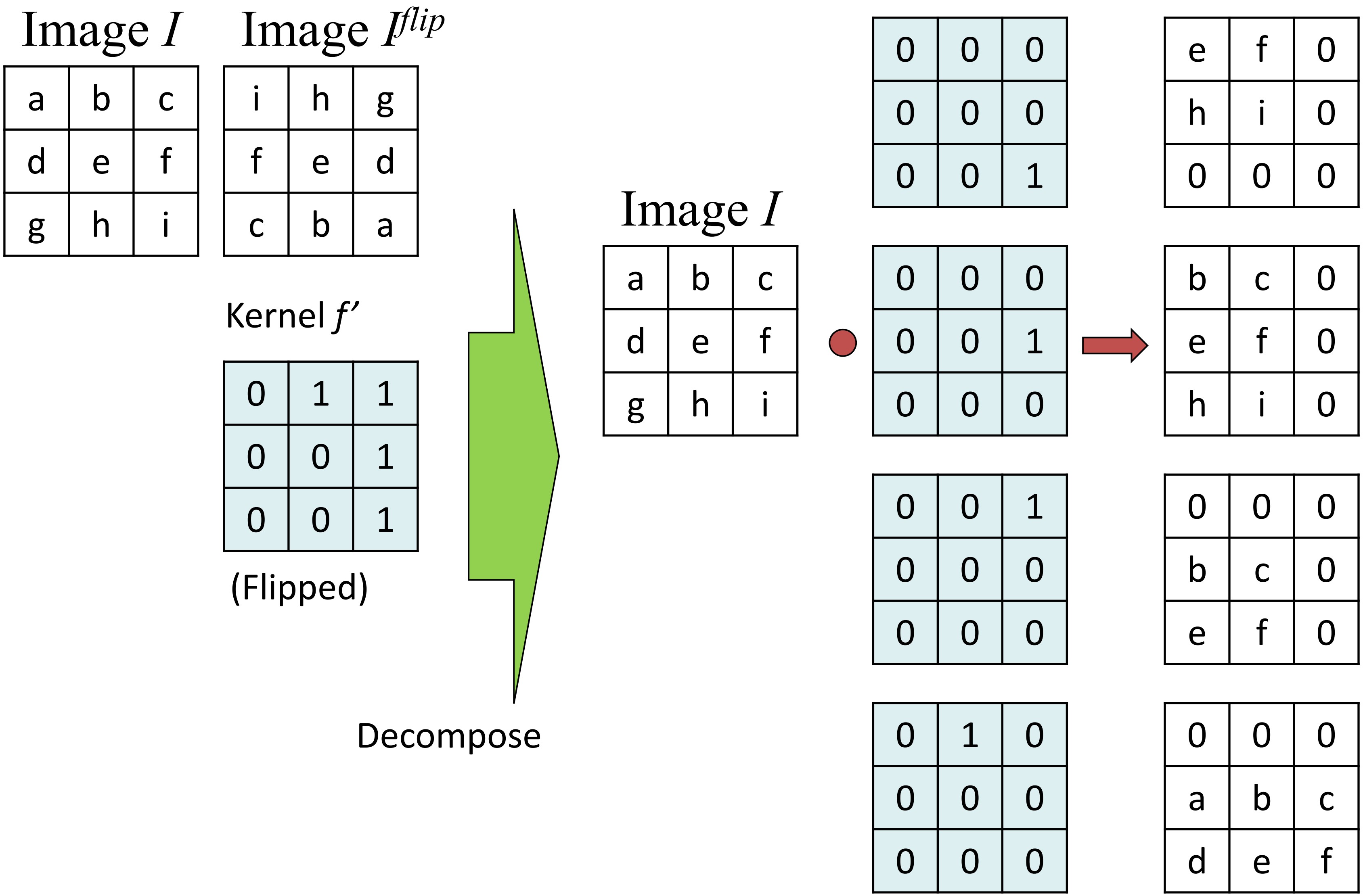
0	0	0
0	0	0
0	0	1

Result

e	f	0
h	i	0
0	0	0

- In this case the resulting image shifted to the upper left

- Convolution has commutative property $I \otimes f$



Proof of Commutative property

- $g[m,n] = I \otimes f = f \otimes I$
- $g[m,n] = I \otimes f = \sum_{k,l} I(m-k, n-l) * f(k,l)$
- Let $k' = m - k, l' = n - l,$
  then $k = m - k', l = n - l'$
- $g[m,n] = \sum_{k\prime,l\prime} I(k', l') * f(m-k', m-l') = f \otimes I$

2D visualization of convolution (full)

Image I

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

8x8

Kernel f

1	2	3
4	5	6
7	8	9

3x3

Output g

1	3	6	6	6	6	6	6	5	3
5	12	21	21	21	21	21	21	16	9
12	27	45	45	45	45	45	45	33	18
12	27	45	45	45	45	45	45	33	18
11	24	39	39	39	39	39	39	28	15
7	15	24	24	24	24	24	24	17	9
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

10x10

2D visualization of convolution (same)

Image I

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

8x8

Kernel f

1	2	3
4	5	6
7	8	9

3x3

Output g

12	21	21	21	21	21	21	16
27	45	45	45	45	45	45	33
27	45	45	45	45	45	45	33
24	39	39	39	39	39	39	28
15	24	24	24	24	24	24	17
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

8x8

2D visualization of convolution (valid)

Image I

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

8x8

Kernel f

1	2	3
4	5	6
7	8	9

3x3

Output g

45	45	45	45	45	45
45	45	45	45	45	45
39	39	39	39	39	39
24	24	24	24	24	24
0	0	0	0	0	0
0	0	0	0	0	0

6x6

Linear independence

Image I

0	1	0	1	0
0	1	0	1	0
0	1	0	1	0
0	1	0	1	0
0	1	0	1	0

5x5

Kernel f

1	0	0
0	1	0
0	0	0

Output $g=g_1+g_2$

1	1	1	1	0
1	1	1	1	0
1	1	1	1	0
1	1	1	1	0
0	1	0	1	0

Kernel f_1

1	0	0
0	0	0
0	0	0

Output g_1

1	0	1	0	0
1	0	1	0	0
1	0	1	0	0
1	0	1	0	0
0	0	0	0	0

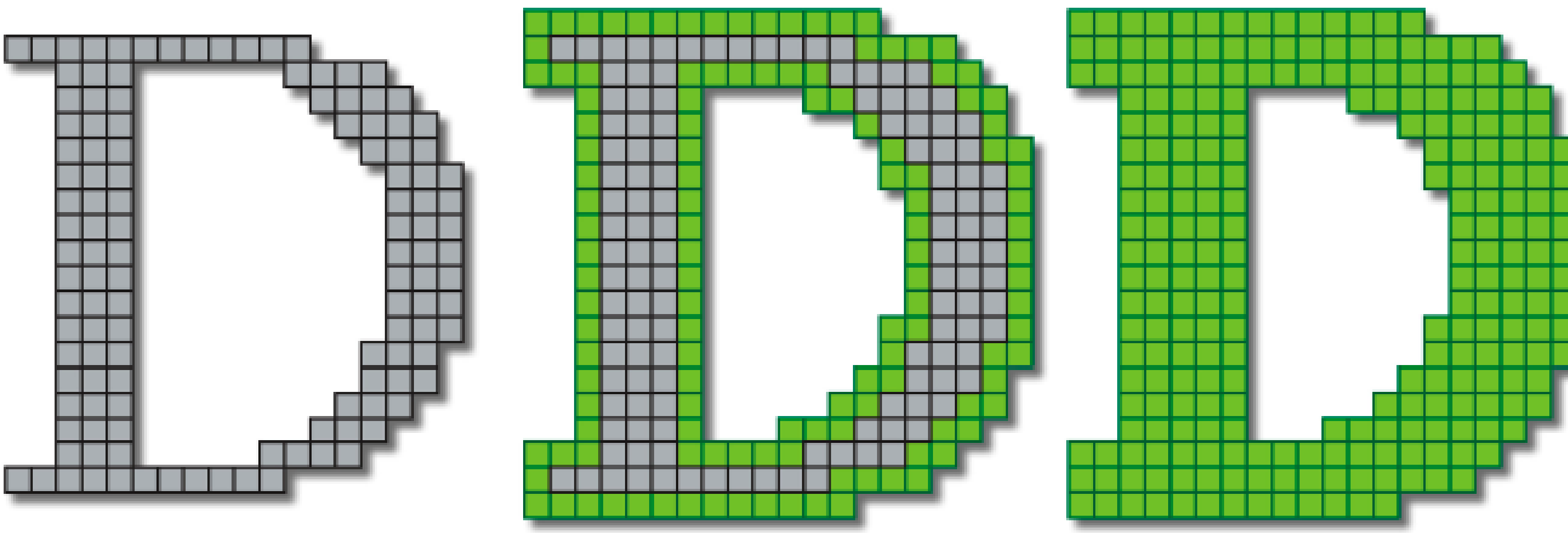
Kernel f_2

0	0	0
0	1	0
0	0	0

Output g_2

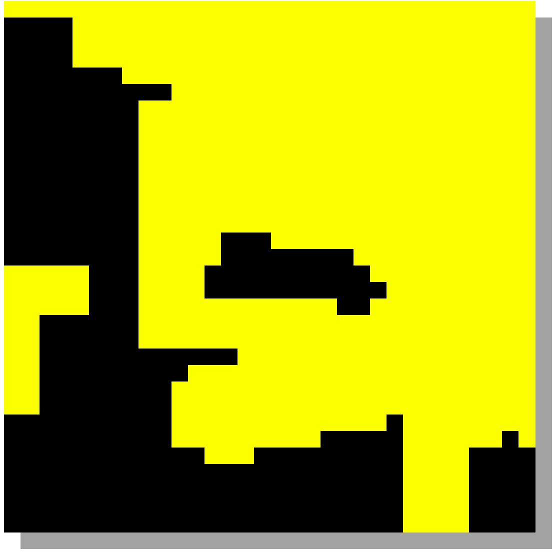
0	1	0	1	0
0	1	0	1	0
0	1	0	1	0
0	1	0	1	0
0	1	0	1	0

Dilation

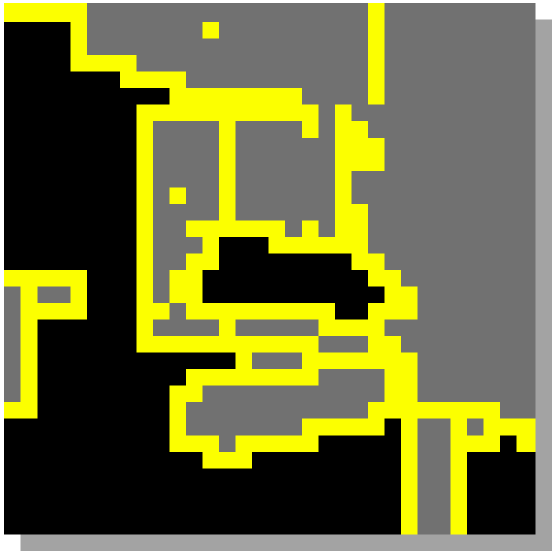


Dilation

The locus of pixels $\mathbf{p} \in S_{\mathrm{p}}$ such that $(\check{Z}+\mathbf{p}) \cap \mathrm{I} \neq \varnothing$.



dilated image

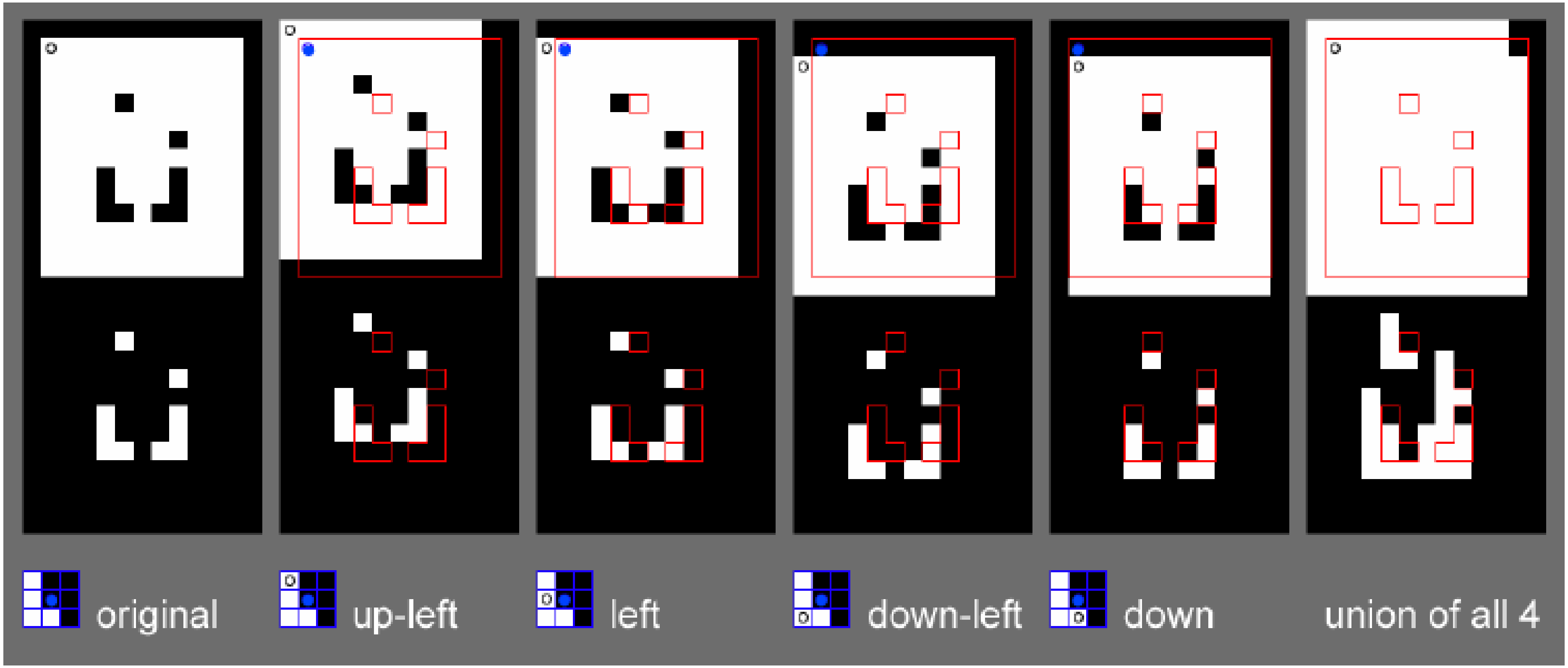


original / dilation

original image

SE = Z_8

Dilation through Image Shifting



Examples of image operation as convolution

Average Filter

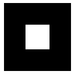
- Mask with positive entries, that sum 1.
- Replaces each pixel with an average of its neighborhood.
- If all weights are equal, it is called a BOX filter.

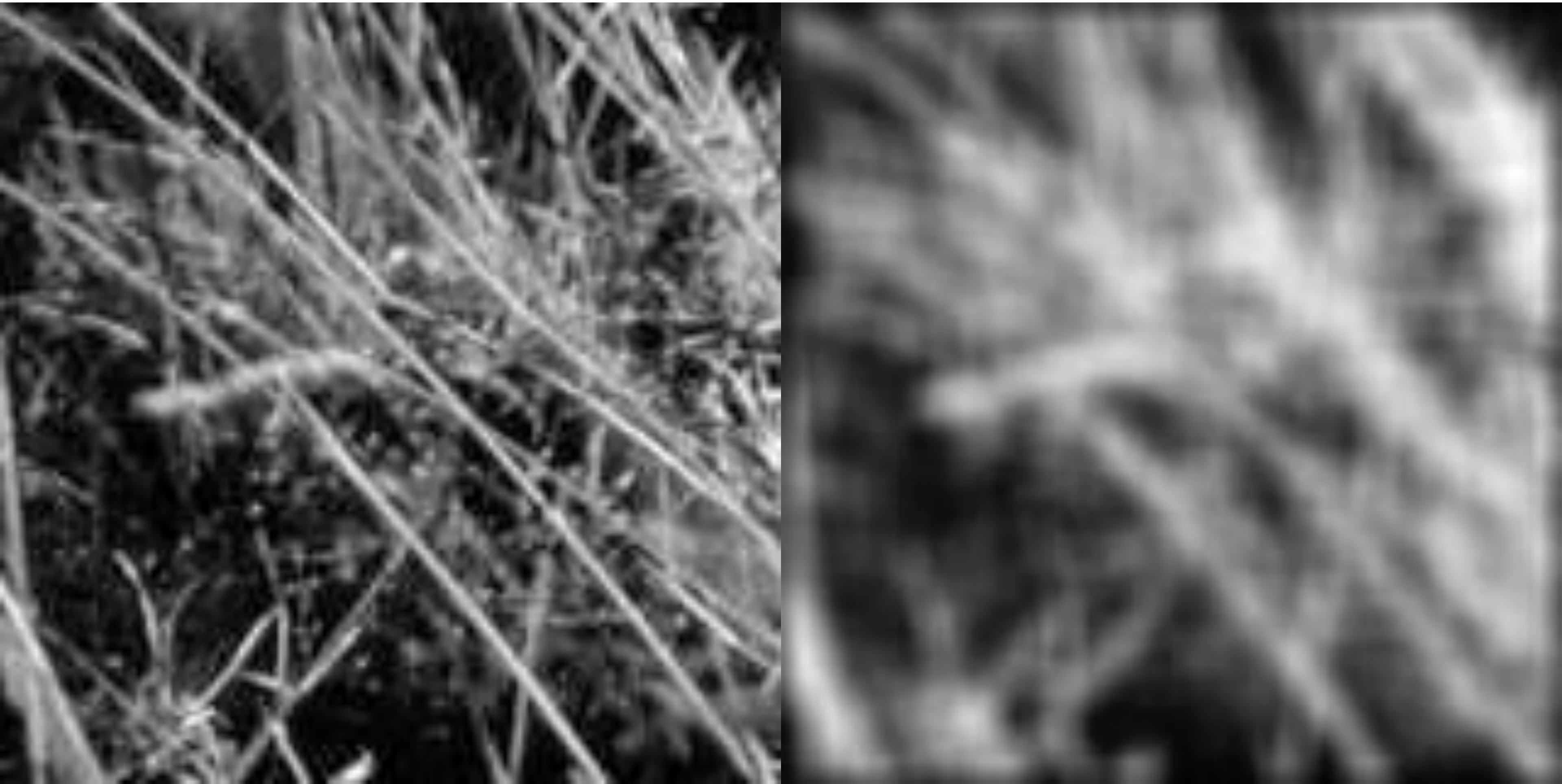
F

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

(Camps)

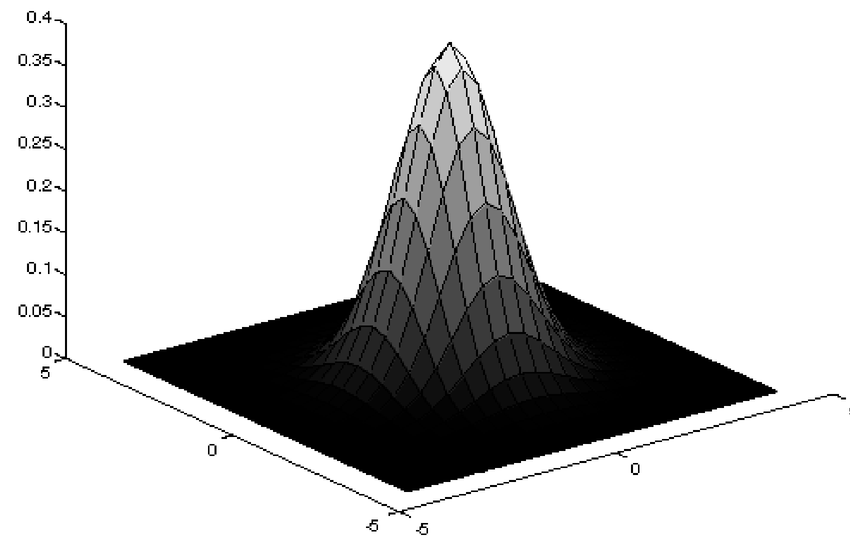
Example 1: Smoothing by Averaging





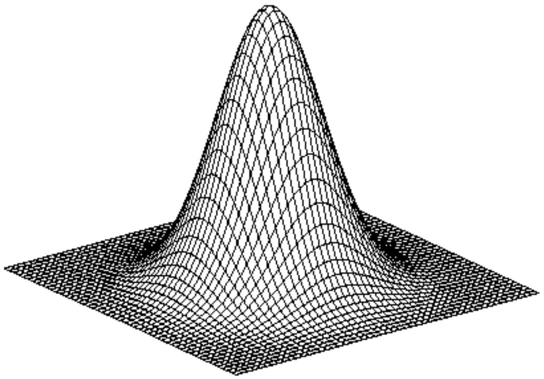
Gaussian Averaging

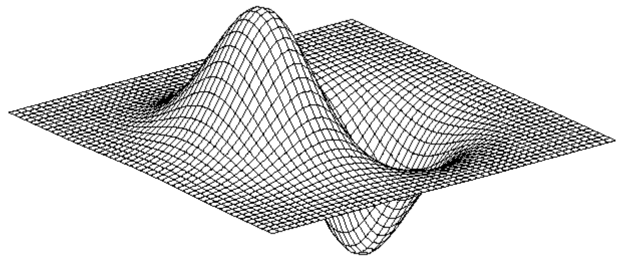
- Rotationally symmetric.
- Weights nearby pixels more than distant ones.
  - This makes sense as probabalistic inference.



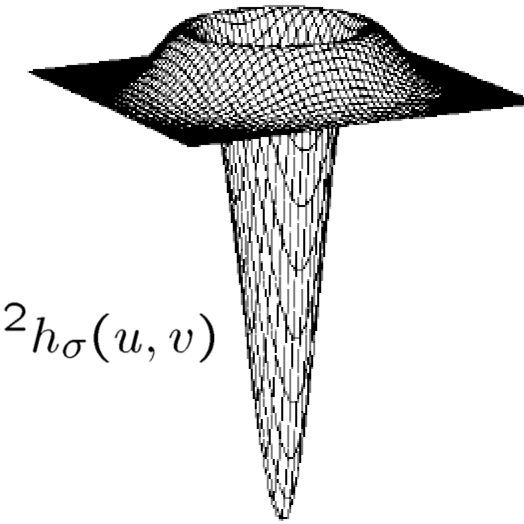
- A Gaussian gives a good model of a fuzzy blob

2D filters, more on this later…





Laplacian of Gaussian



Gaussian

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

derivative of Gaussian

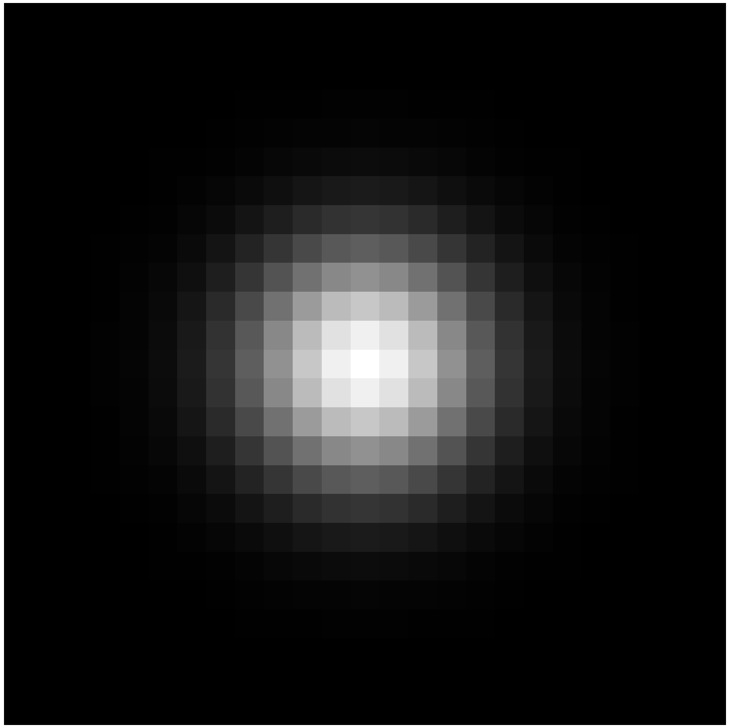
$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

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

- is the **Laplacian** operator:

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

An Isotropic Gaussian

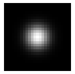


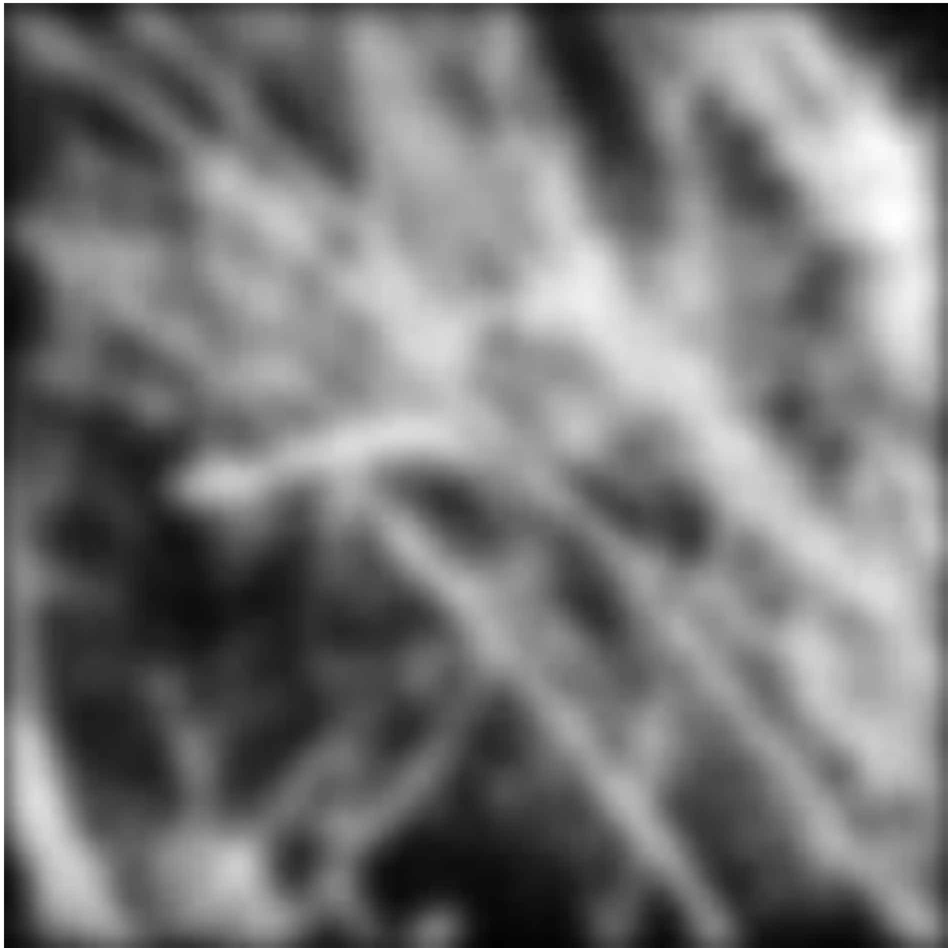
- The picture shows a smoothing kernel proportional to

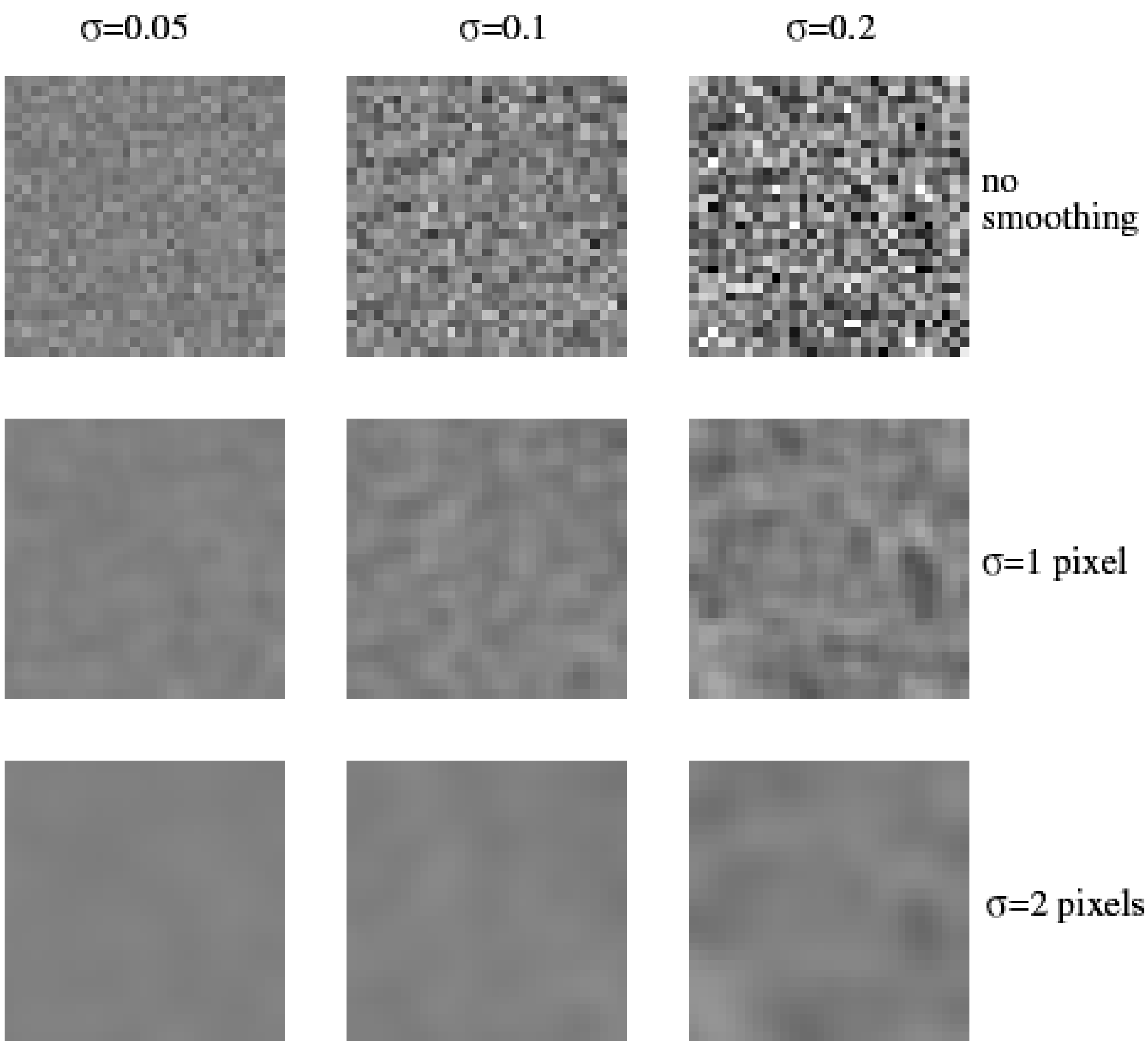
$$e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- (which is a reasonable model of a circularly symmetric fuzzy blob)

Smoothing with a Gaussian

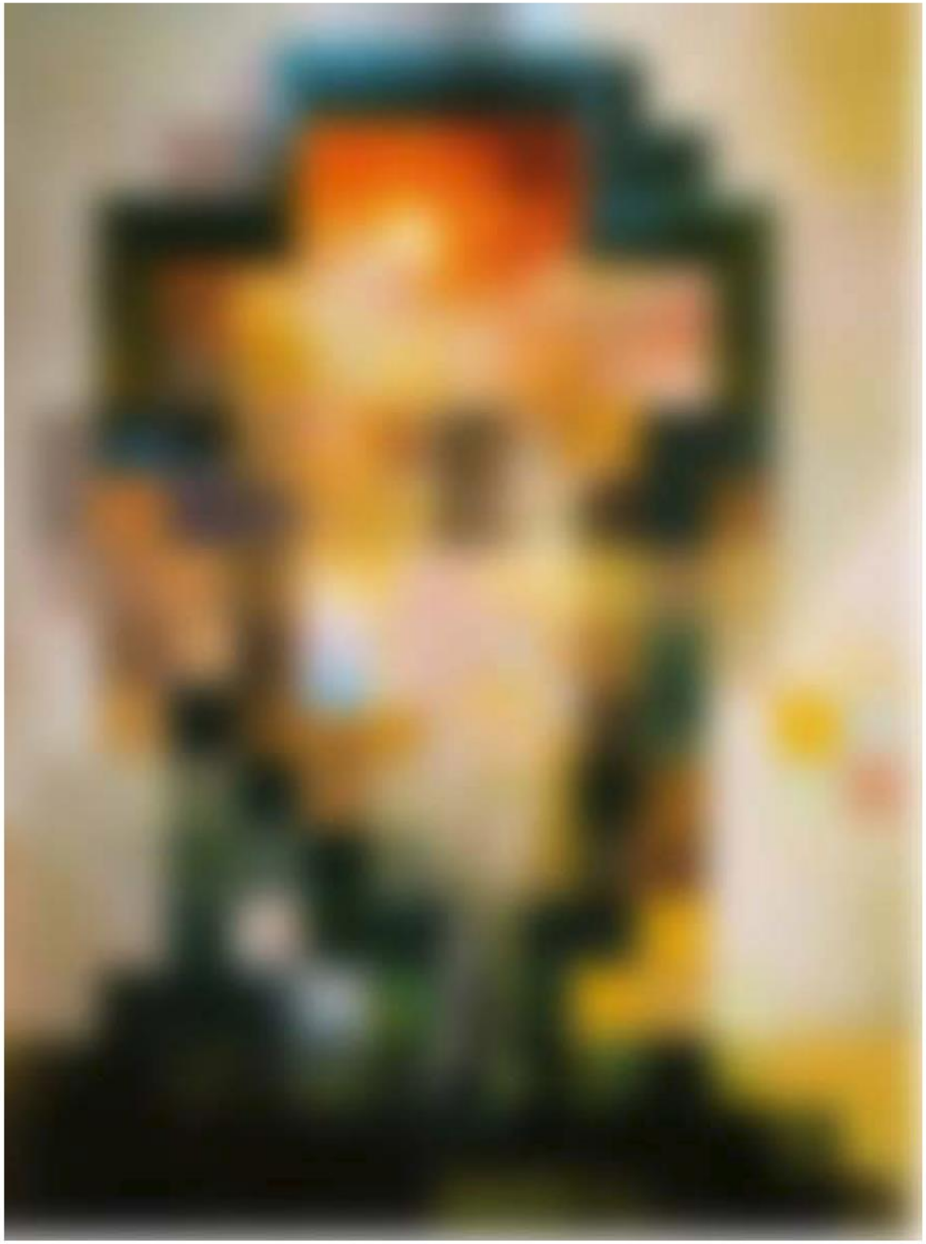






The effects of smoothing
Each row shows smoothing with gaussians of different width; each column shows different realizations of an image of gaussian noise.

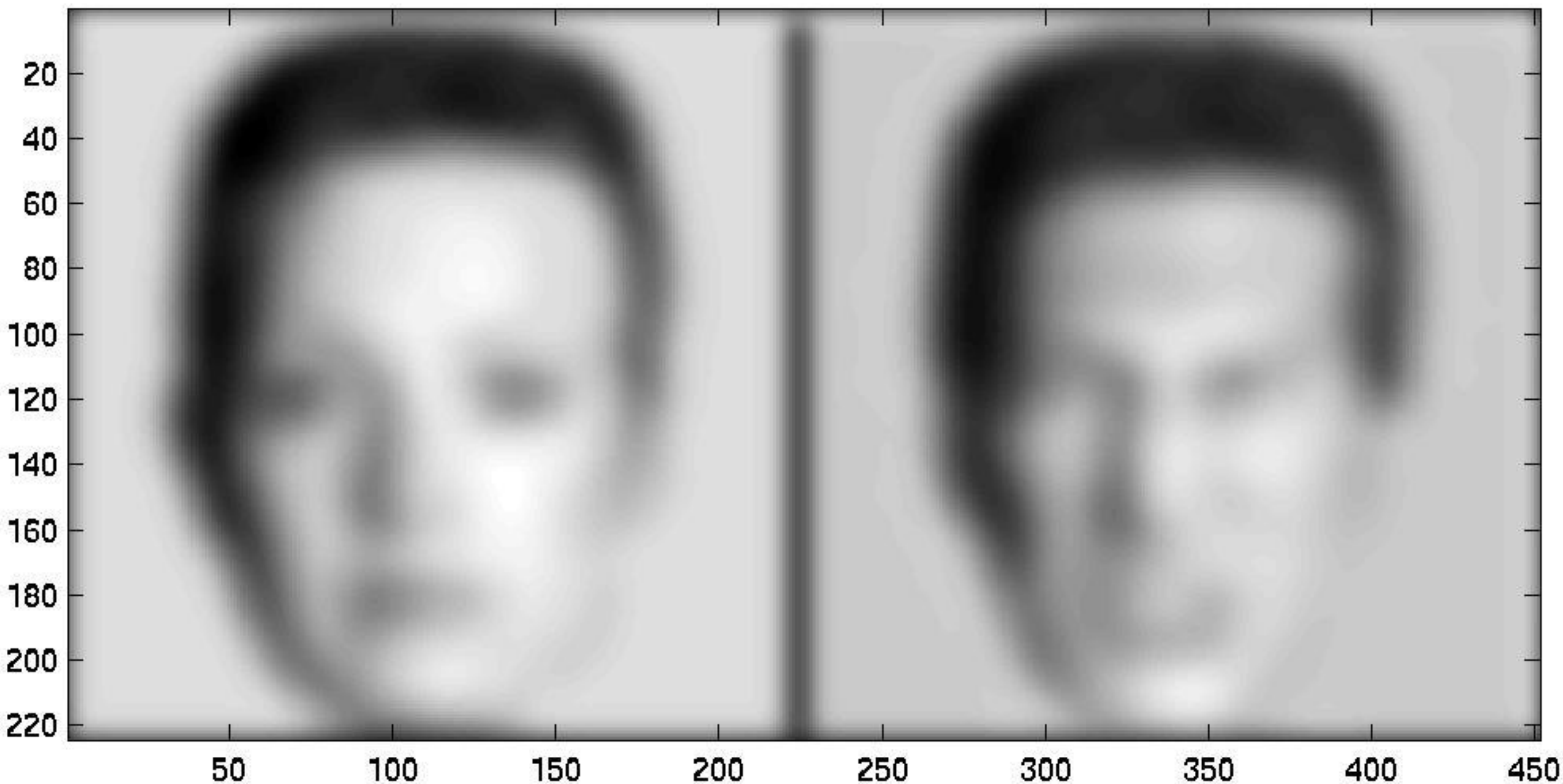
Salvador Dali, *"Gala Contemplating the Mediterranean Sea, which at 30 meters becomes the portrait of Abraham Lincoln"*, 1976



Salvador Dali, *"Gala Contemplating the Mediterranean Sea, which at 30 meters becomes the portrait of Abraham Lincoln"*, 1976

Image smoothing can remove noise, and also …

20
40
60
80
100
120
140
160
180
200
220
50
100
150
200
250
300
350
400
450


20
40
60
80
100
120
140
160
180
200
220
50
100
150
200
250
300
350
400
450