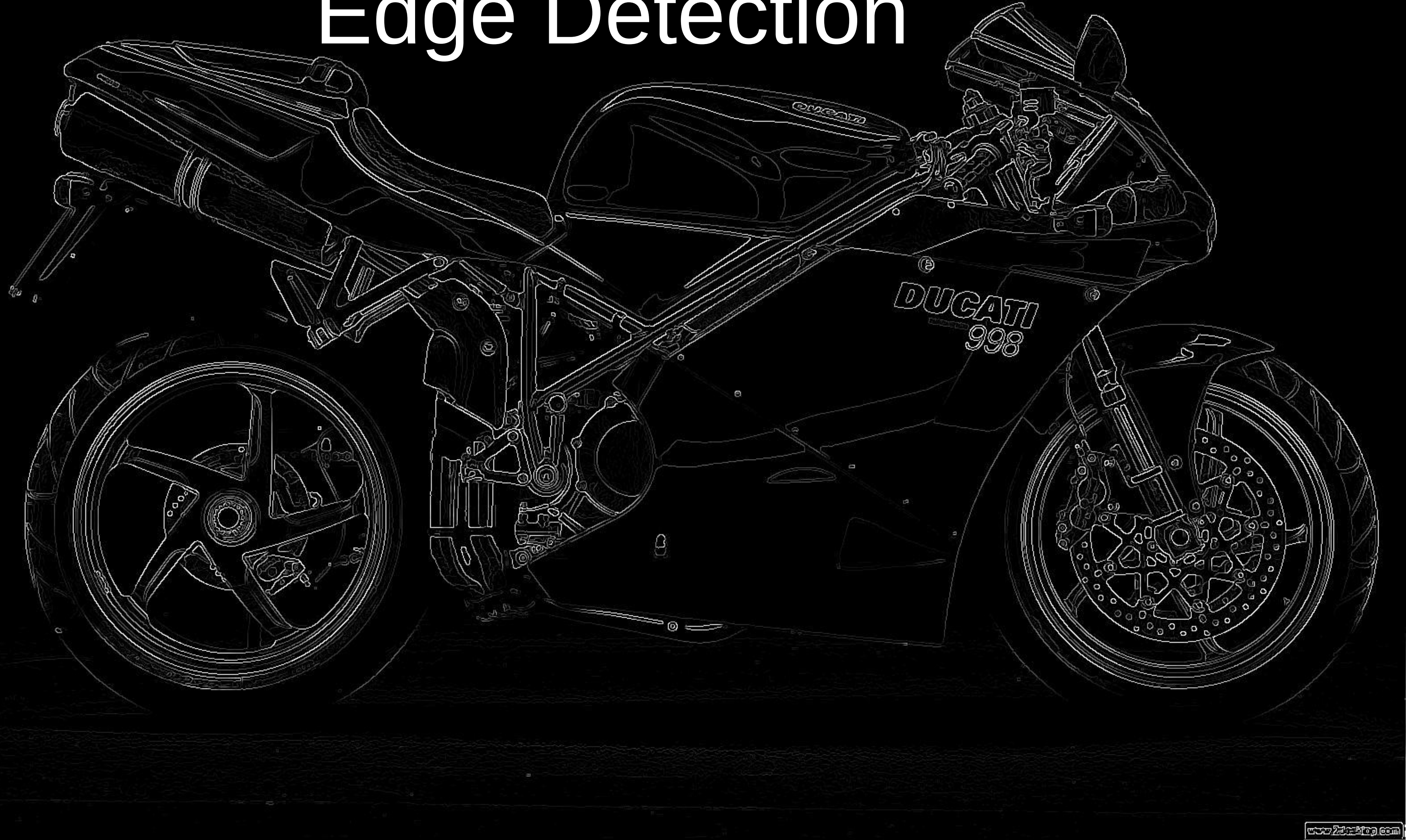
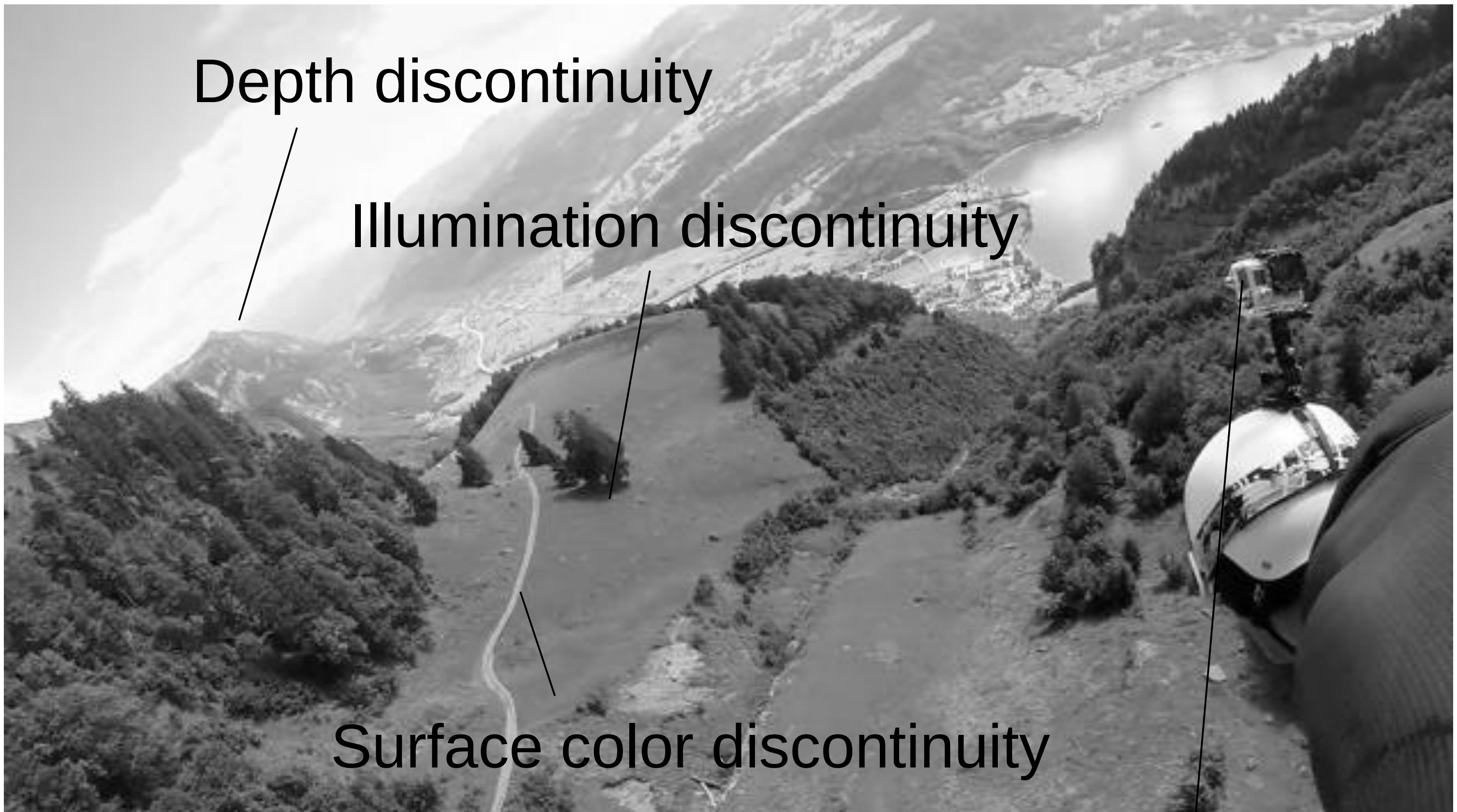
Image Convolution and Edge Detection

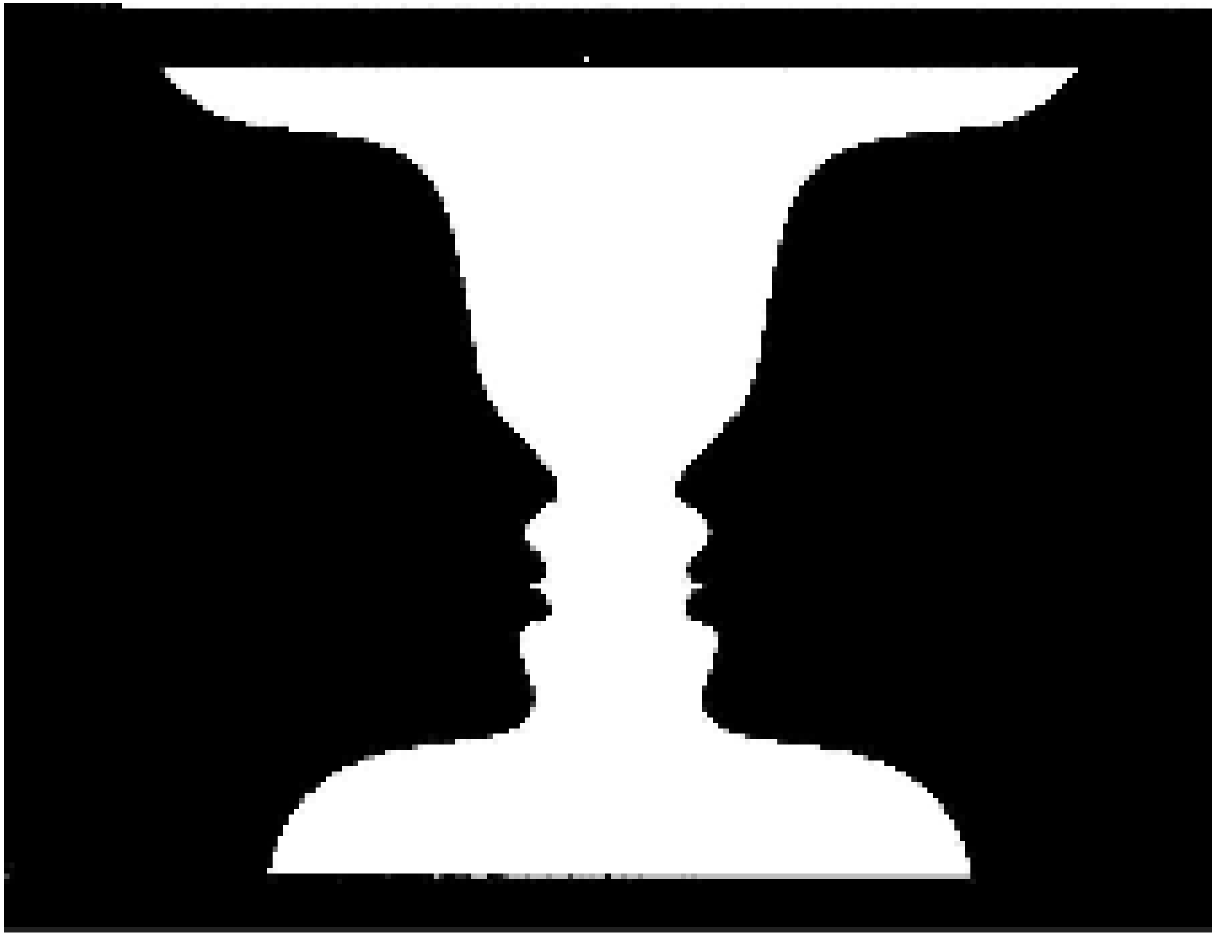


Edge Formation Factors



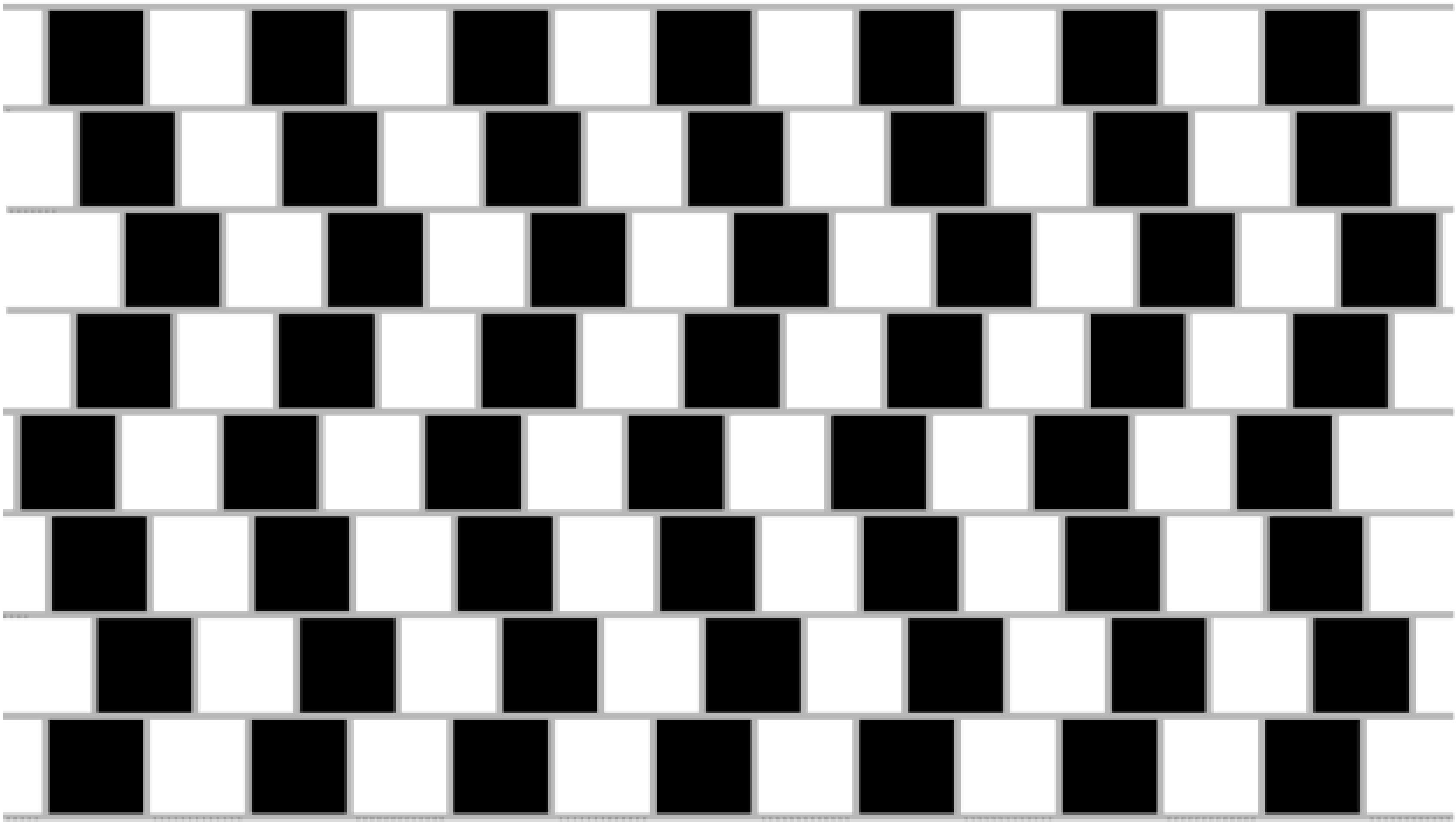
What’s in front?

What it is?

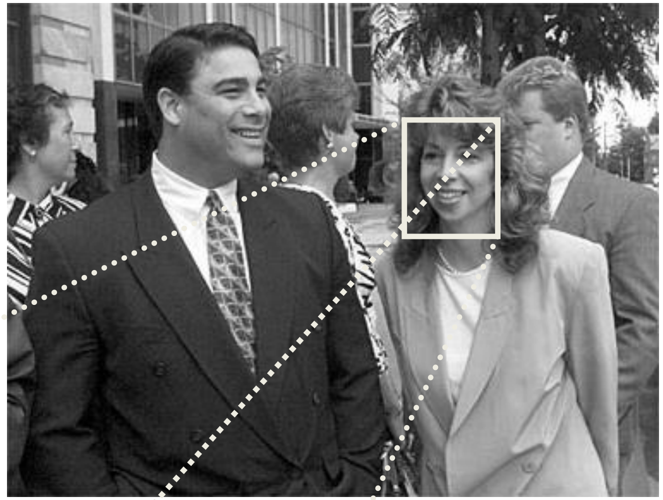


http://www.cfar.umd.edu/~fer/optical/index.html

The rows of black and white squares are all parallel.
The vertical zigzag patterns disrupt our horizontal perception.



Cornelia Fermüller



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

Any 2D matrix can be seen as an image

Linear Filtering

Image I

200	130	20
255	100	10
200	100	30

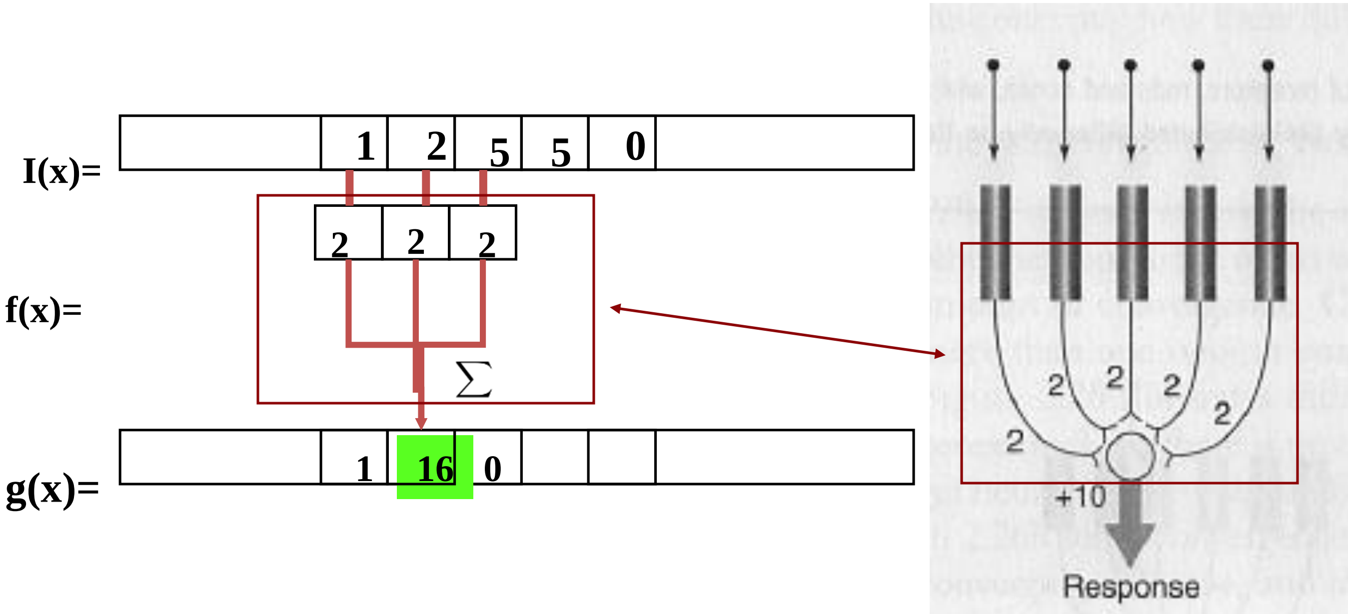
Kernel f

0.5	0	0
0	1.5	0.5
0	-0.5	0

	205	

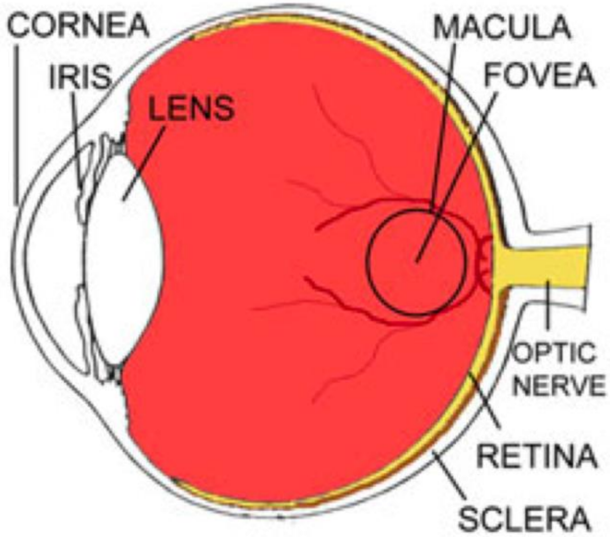
Image *filtering:* Replacing each pixel value by weighted average of its neighbors

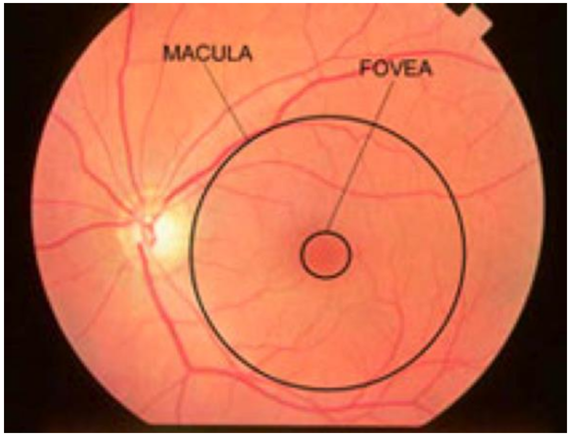
Simple Neural Network



This leads to parallel computation

Anatomy of eye

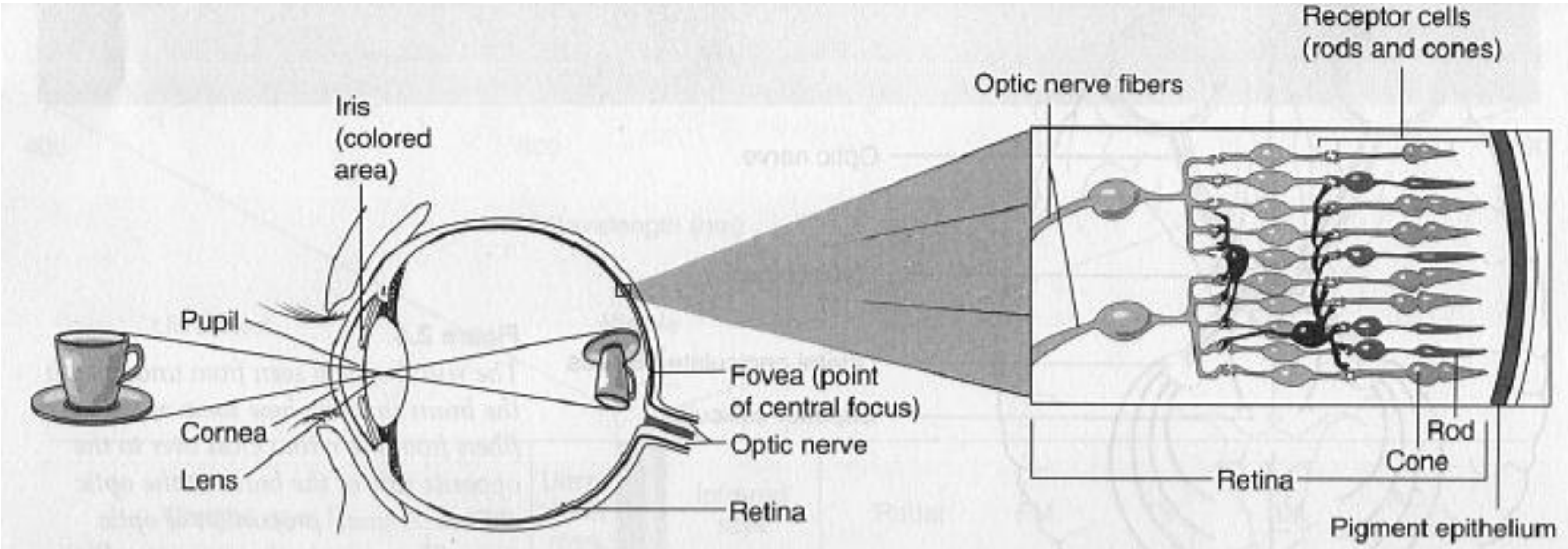




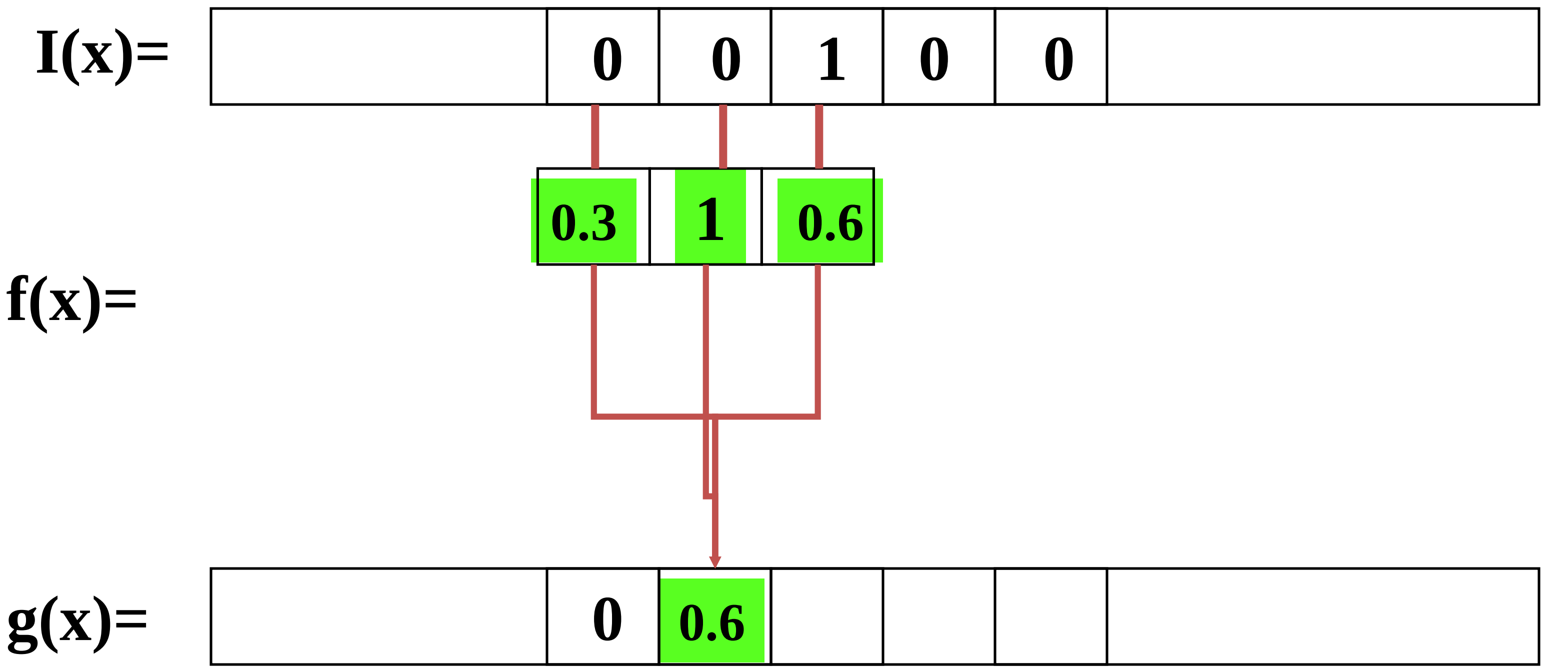
The macula is the center of vision (and retina) and the fovea (FAZ) is the focal point approximately only 0.4mm in diameter. Reading, driving, etc. is all performed here.

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

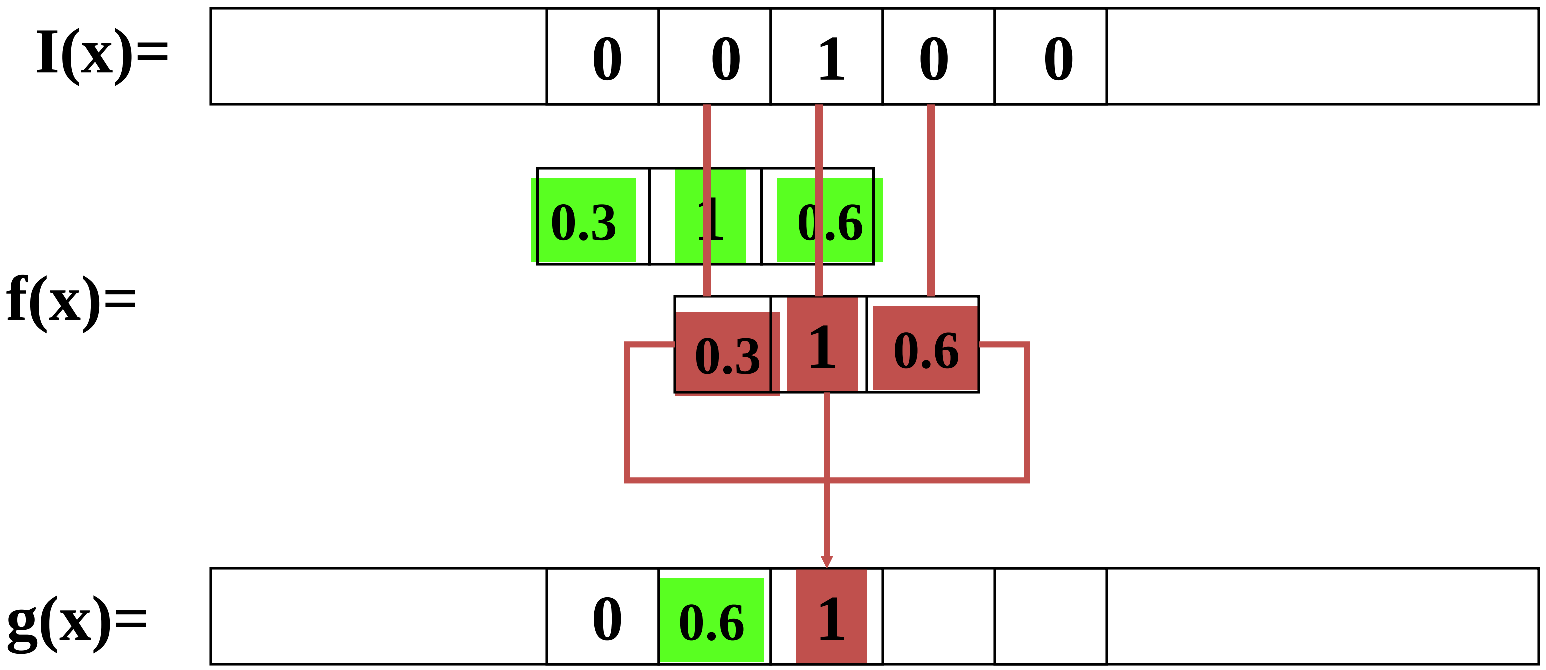
Photo-sensors

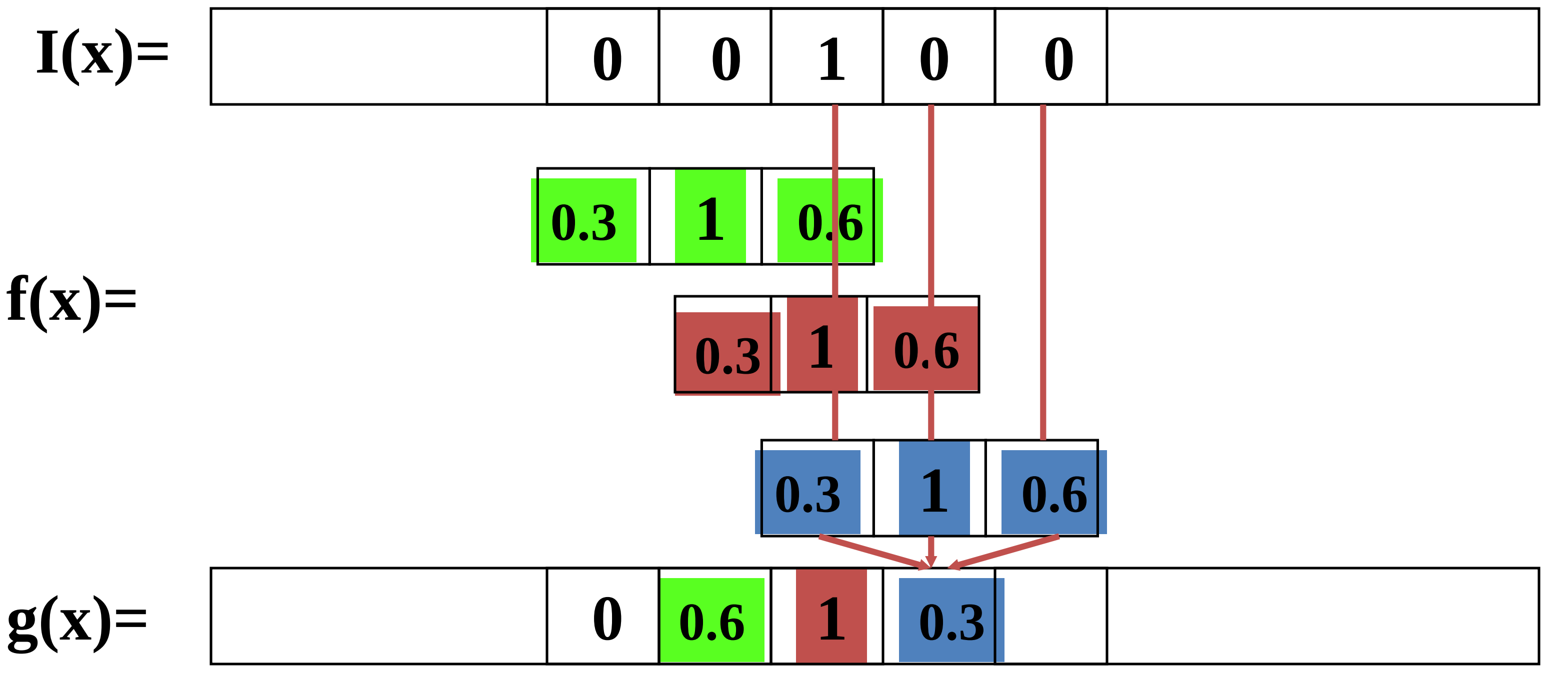


1) **Light pass through retina cells, excites Rod and Cone**
2) **Cone: color(spectral) sensitive, R,G,B, 6 Million**
3) **Rod: more photo sensitive, peak at 580nm(yellow), 120 M**
4) **What happens if you miss one type of Cone cells?**



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


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

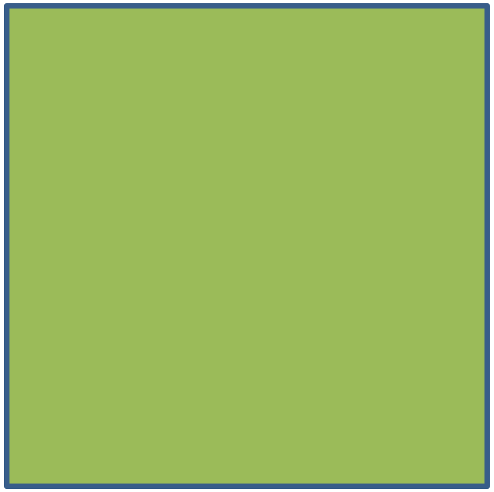

I(x)=
0
0
1
0
0
f(x)=
0.3
1
0.6
0.3
1
0.6
0.3
1
0.6
g(x)=
0
0.6
1
0.3

Linear Filter

Image

0	0	0
0	1	0
0	0	0

Kernel



Linear Filter

Image

0	0	0
0	1	0
0	0	0

Kernel

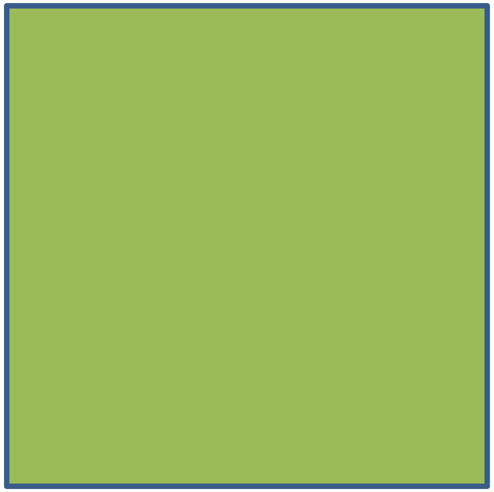
Image no change

Linear Filter

Image

0	0	0
1	0	0
0	0	0

Kernel



Linear Filter

Image

0	0	0
1	0	0
0	0	0

Kernel

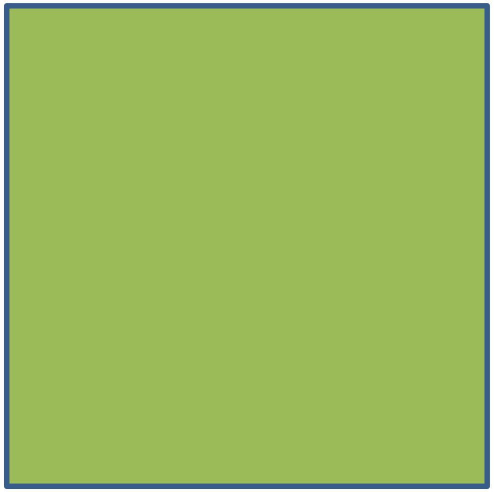
Image Shifted

Linear Filter

Image

0	0	0
0.3	0.3	0.3
0	0	0

Kernel



Linear Filter

Image

0	0	0
0.3	0.3	0.3
0	0	0

Kernel

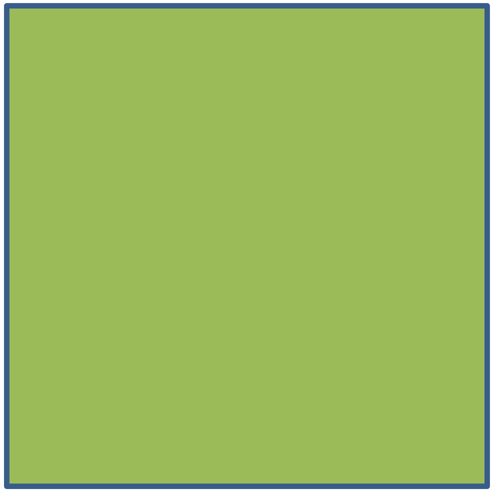
Blurred
(horizontal)

Linear Filter

Image

0	0.3	0
0	0.3	0
0	0.3	0

Kernel

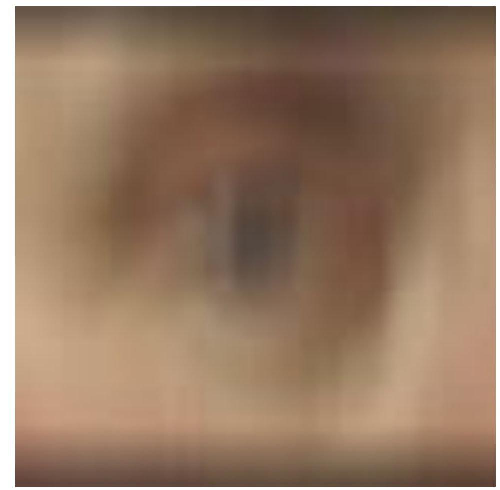


Linear Filter

Image

0	0.3	0
0	0.3	0
0	0.3	0

Kernel



Blurred
(vertical)

Linear Filter

Image

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 \\ 0.3 & 0.3 & 0.3 \\ 0 & 0 & 0 \end{bmatrix}$$

Filter
Kernel

Linear Filter

Image

0	0	0
0	2	0
0	0	0

\-

0	0	0
0.3	0.3	0.3
0	0	0

Kernel Image

\-

Linear Filter

0	0	0
0	2	0
0	0	0

-

0	0	0
0.3	0.3	0.3
0	0	0

Kernel Image

Image

-

Image filtering

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

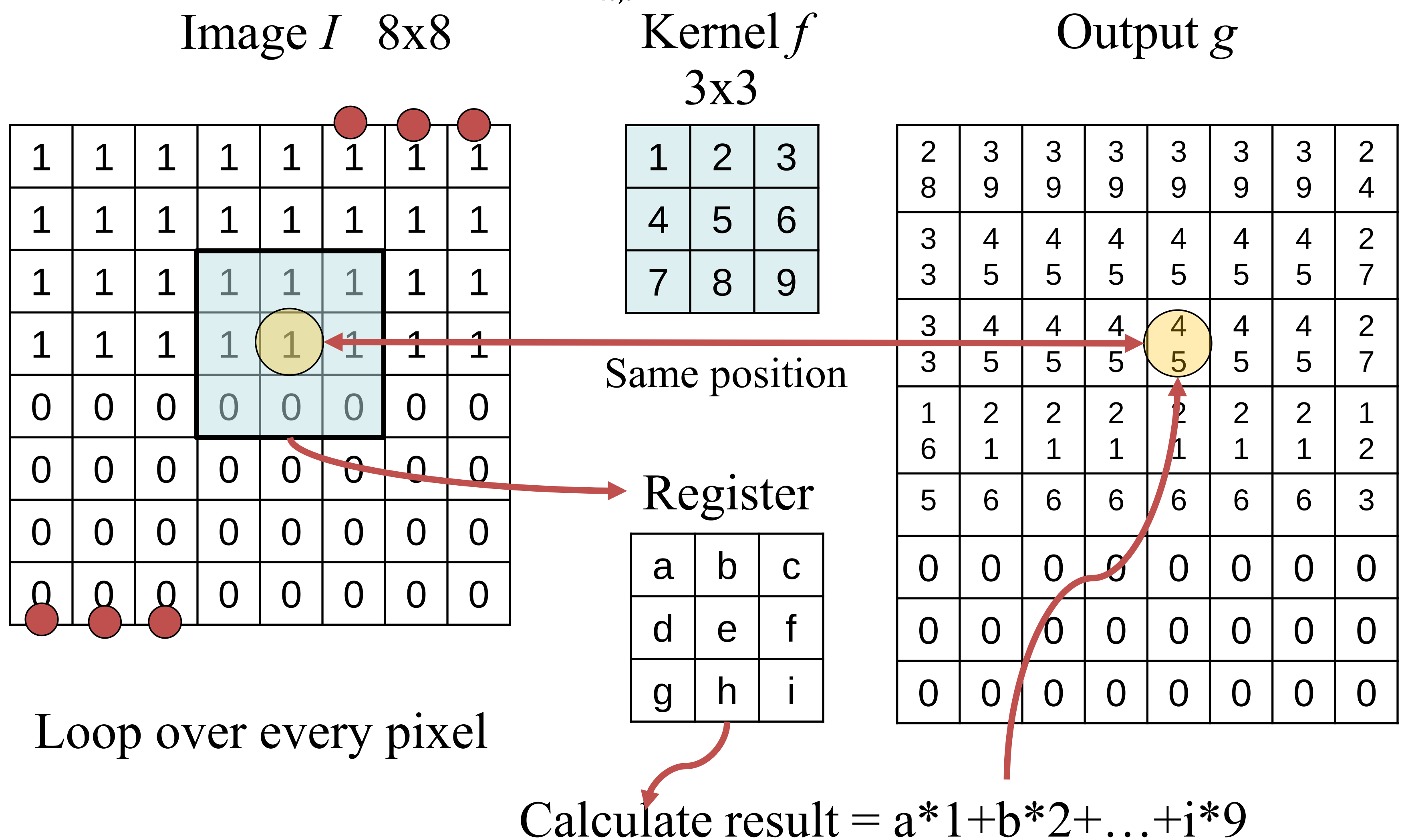
Output Image

Input Image

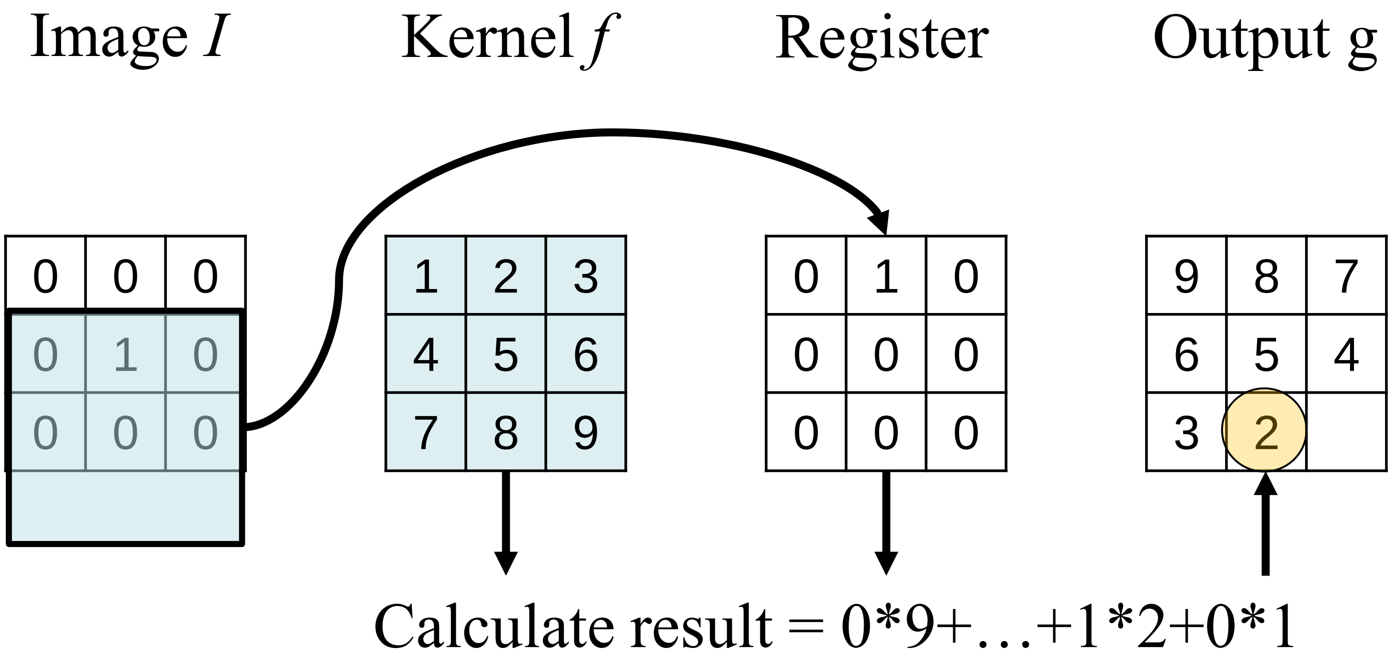
Kernal Image

Image filtering

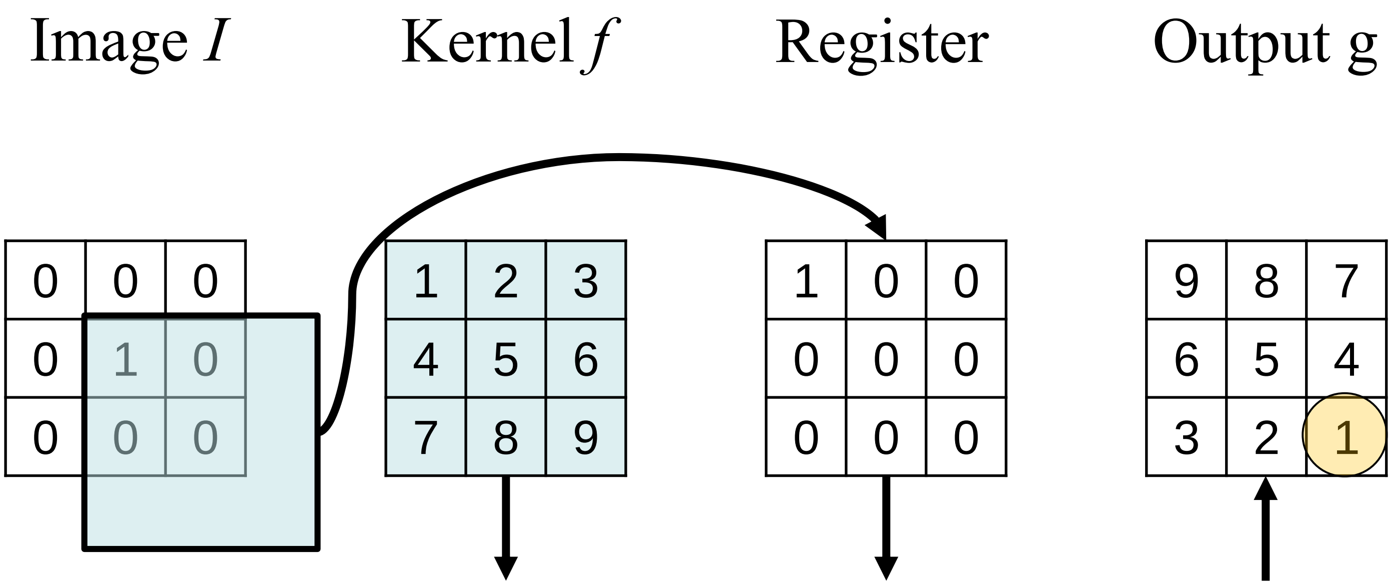
$$g[m,n] = \sum_{k,l} I(m+k, n+l) * f(k,l)$$



Special case: impulse function



Special case: impulse function



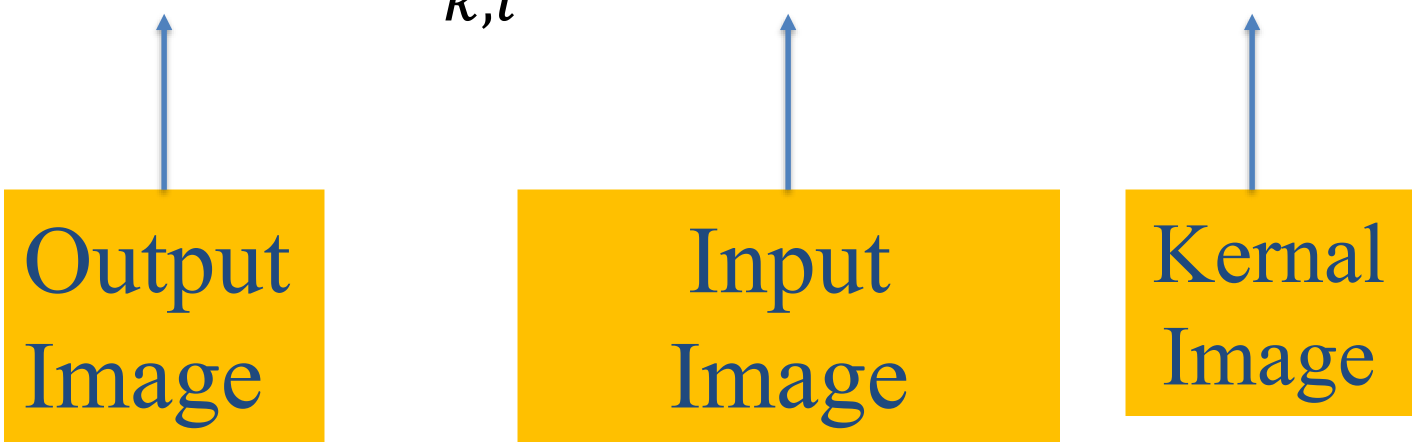
Calculate result = 0*9+…0*2+1*1

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

Convolution $I \otimes f$

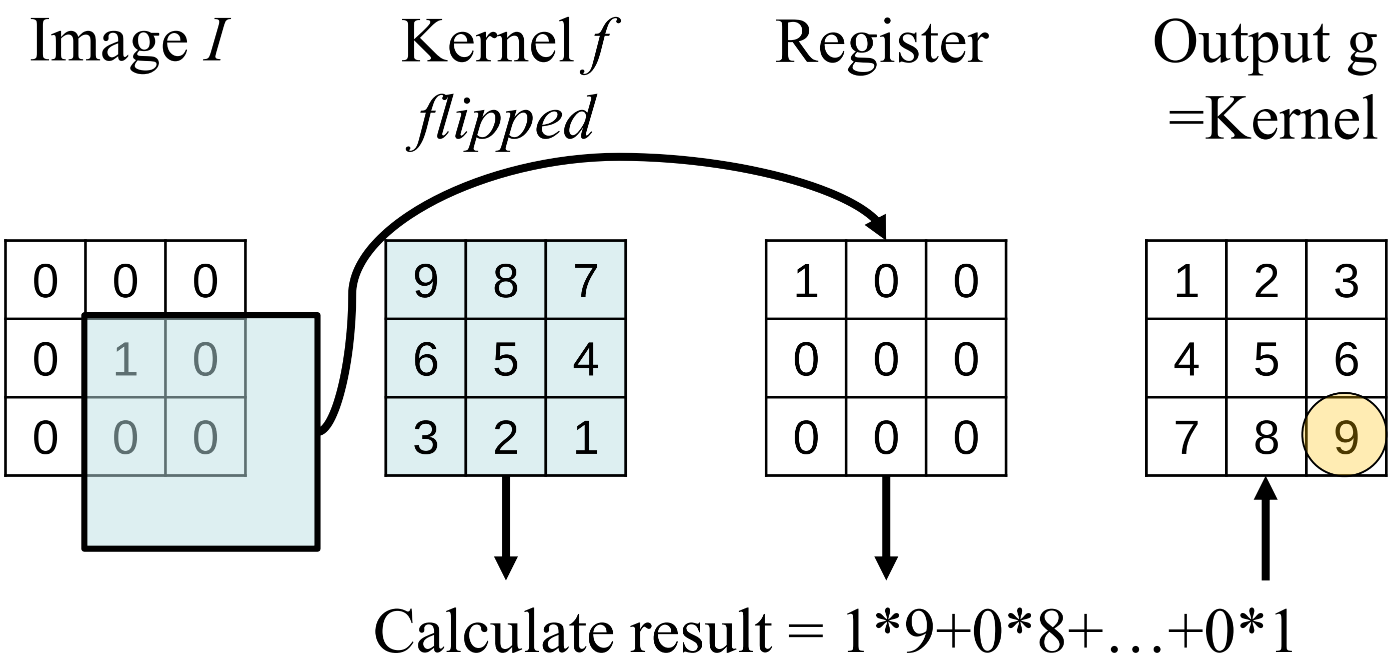
- Let I be an Signal(image), Convolution kernel f,

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



Convolution

- Convolution is filtering with kernel flipped



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 $I \otimes f$

e	f	0
h	i	0
0	0	0

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

Convolution is associative

I

a	b	c
d	e	f
g	h	i

f

1	0	0
0	0	0
0	0	0

$I \otimes f$

e	f	0
h	i	0
0	0	0

f

1	0	0
0	0	0
0	0	0

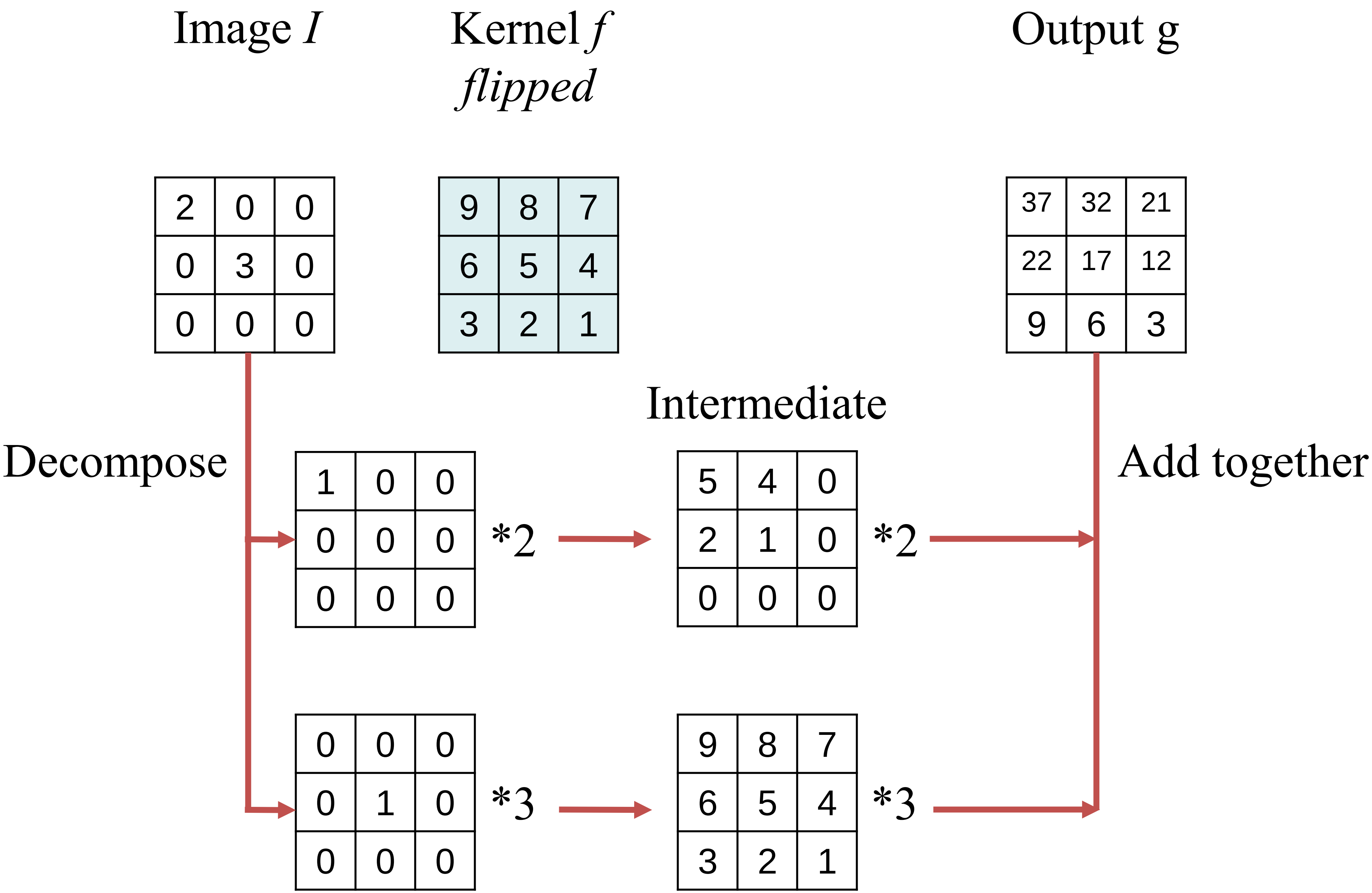
I

a	b	c
d	e	f
g	h	i

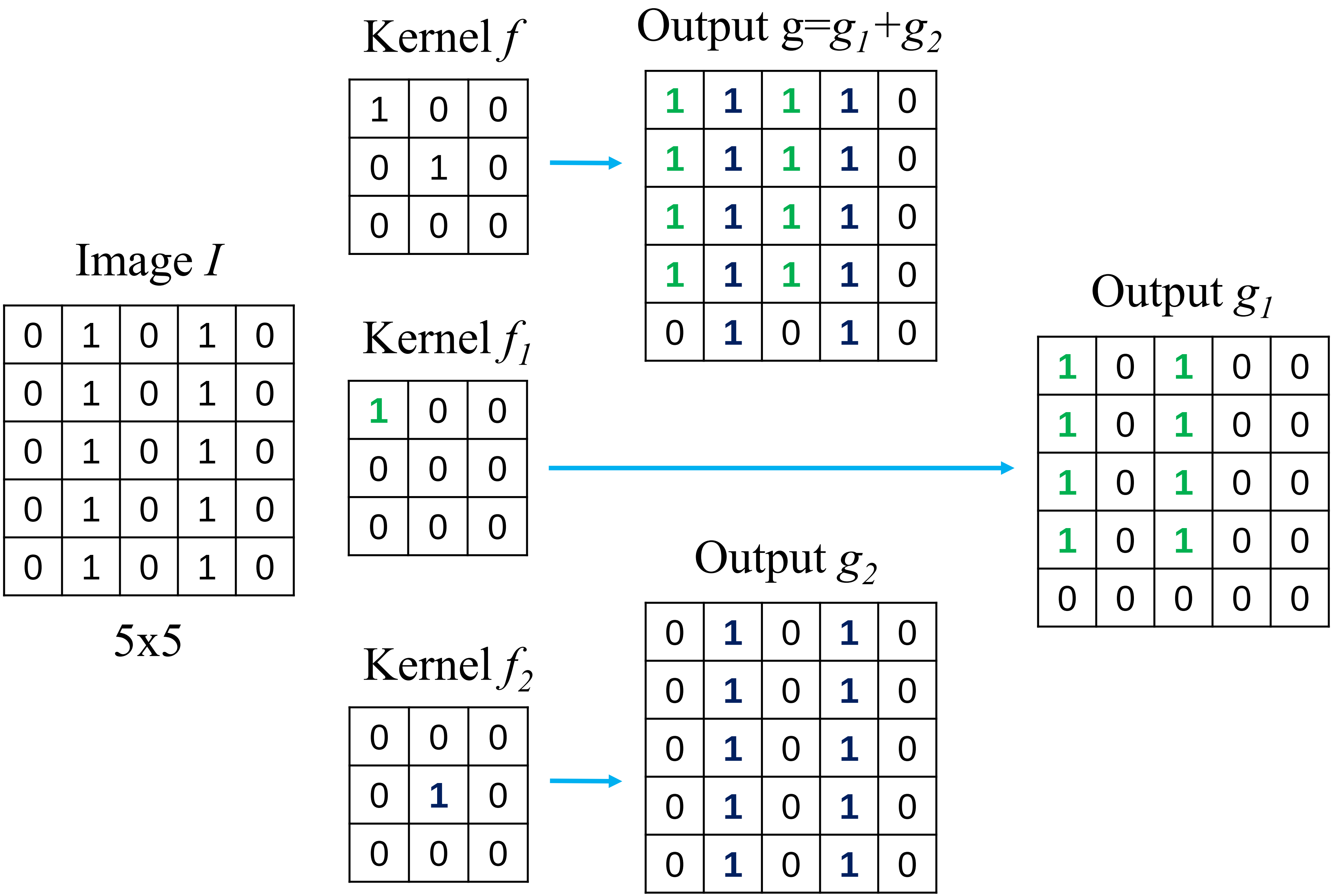
$\mathrm{f} \otimes I$

e	f	0
h	i	0
0	0	0

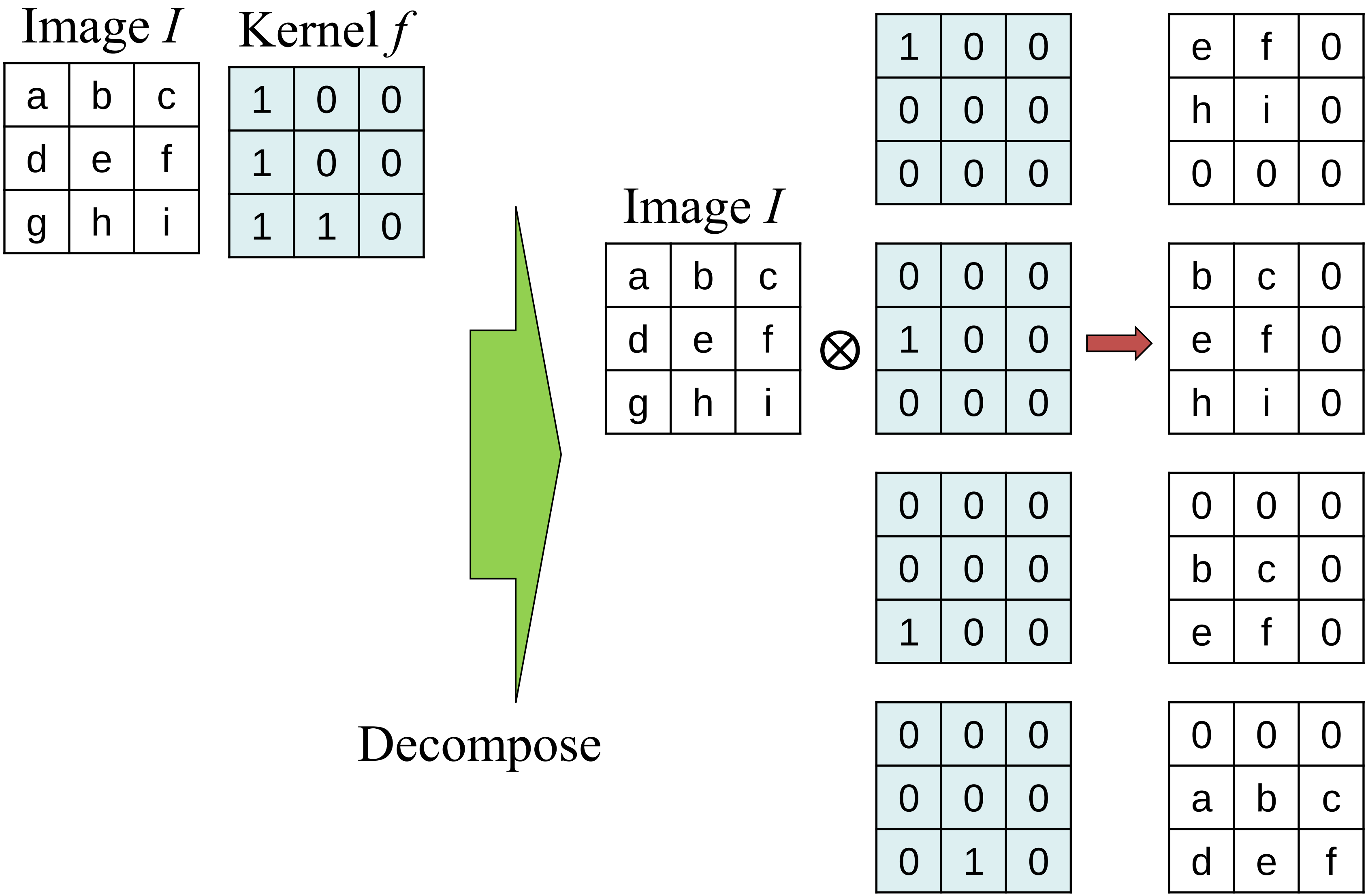
Linear independence



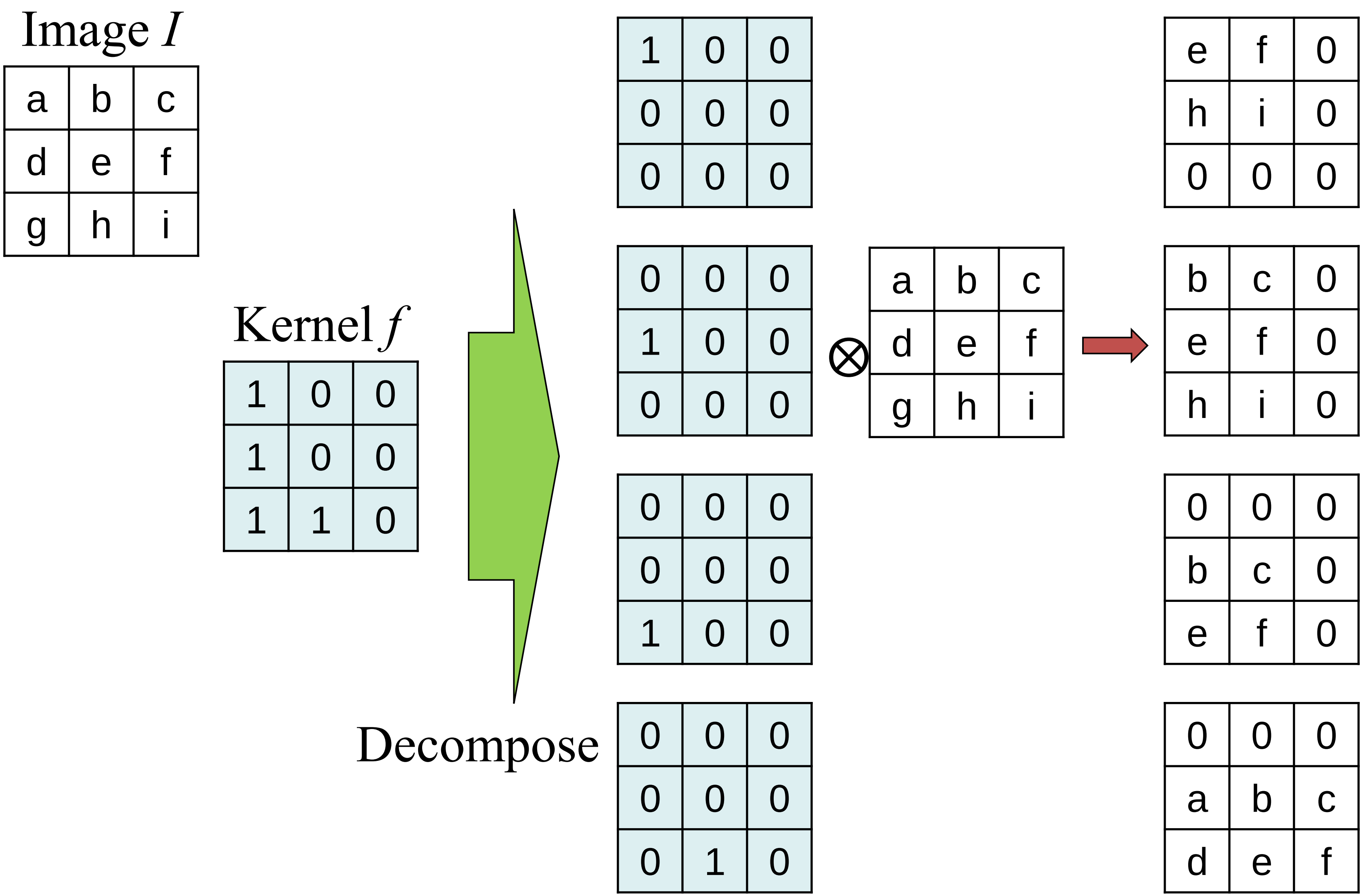
Linear independence



- Convolution has commutative property $I \otimes f$



- Convolution has commutative property $f \otimes I$

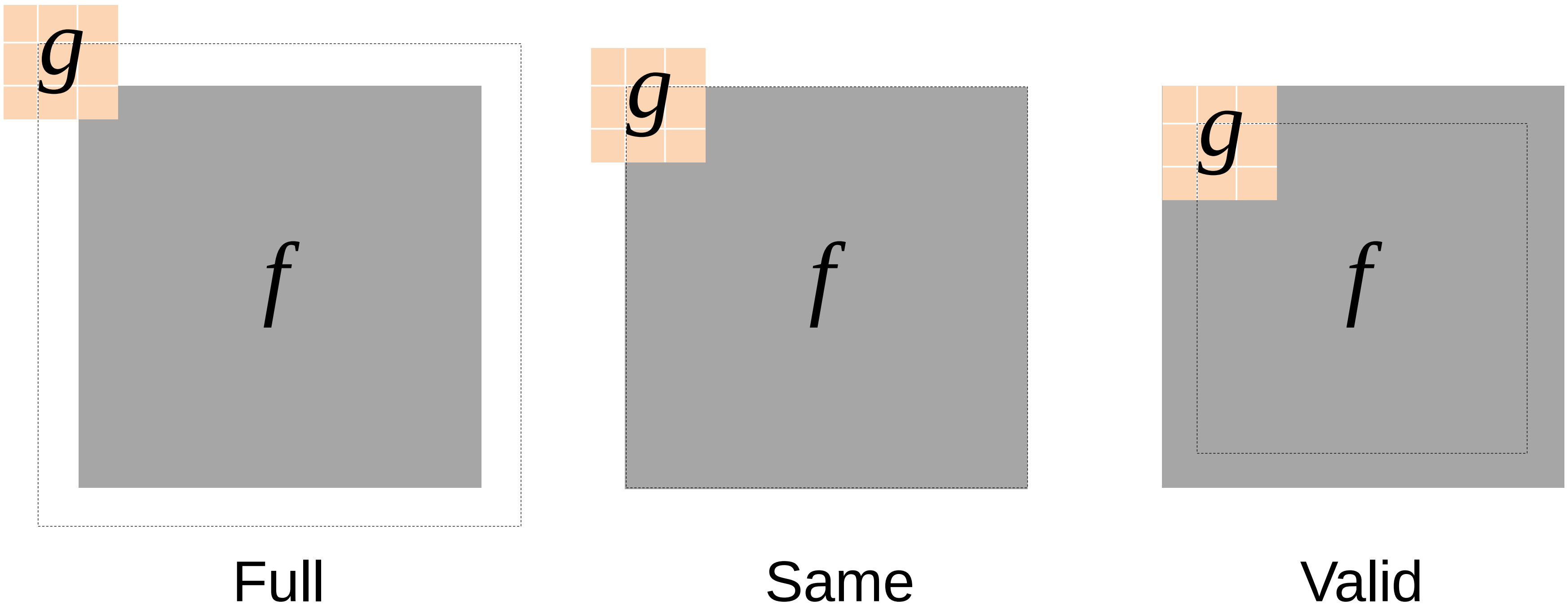


Proof of Commutative property

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

Output Size of Image Convolution

$f \otimes g$



filter2(g, f, shape) in MATLAB

Full: output_size = f_size + g_size – 1

Same: output_size = f_size

Valid: output_size = f_size – (g_size – 1)

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

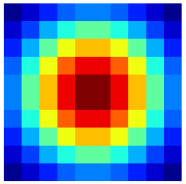
3x3

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

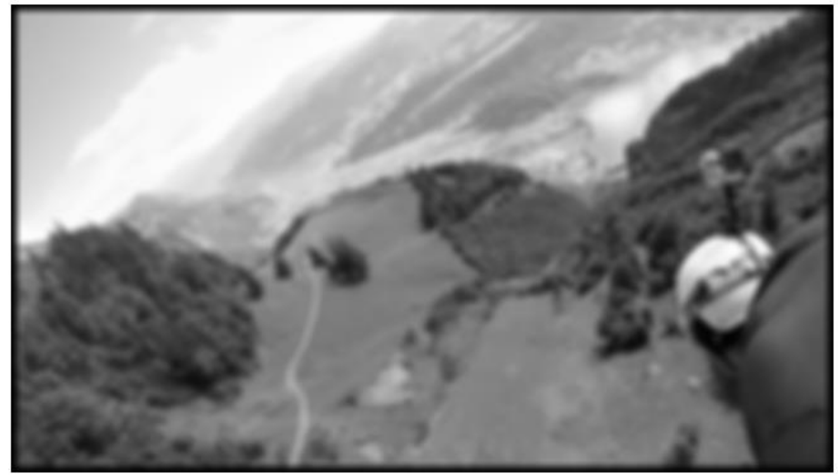
10x10

Output Size of Image Convolution



g : 10 x 10 Gaussian kernel

f : 640 x 360 resolution



Full

filter2(g, f, shape) in MATLAB

Full: output_size = f_size + g_size – 1

```
>> full = filter2(g, im, 'full');
>> size(full)

ans =

   369   649
```

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

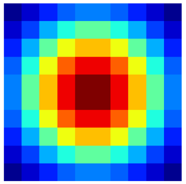
3x3

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

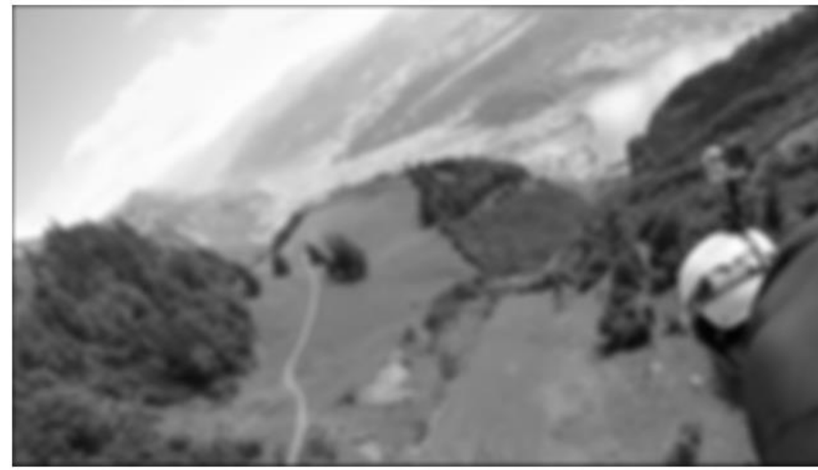
8x8

Output Size of Image Convolution



g : 10 x 10 Gaussian kernel

f : 640 x 360 resolution



Same

filter2(g, f, shape) in MATLAB

Full: output_size = f_size + g_size – 1

Same: output_size = f_size

```
>>same = filter2(g, im, 'same');
>> size(same)

ans =

   360   640
```

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

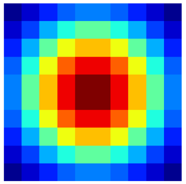
3x3

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

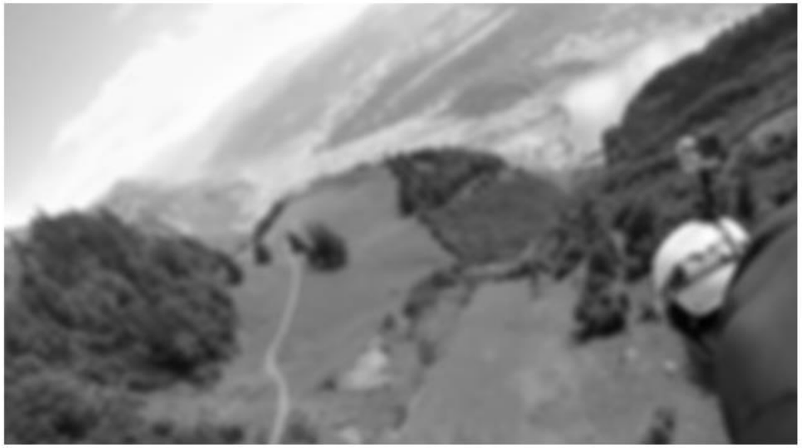
6x6

Output Size of Image Convolution



g : 10 x 10 Gaussian kernel

f : 640 x 360 resolution



Valid

filter2(g, f, shape) in MATLAB

Full: output_size = f_size + g_size – 1

Same: output_size = f_size

Valid: output_size = f_size – (g_size – 1)

```
>> valid = filter2(g, im, 'valid');
>> size(valid)

ans =

   351   631
```

Image Boundary Effect

The filter window falls off at the edge of image.

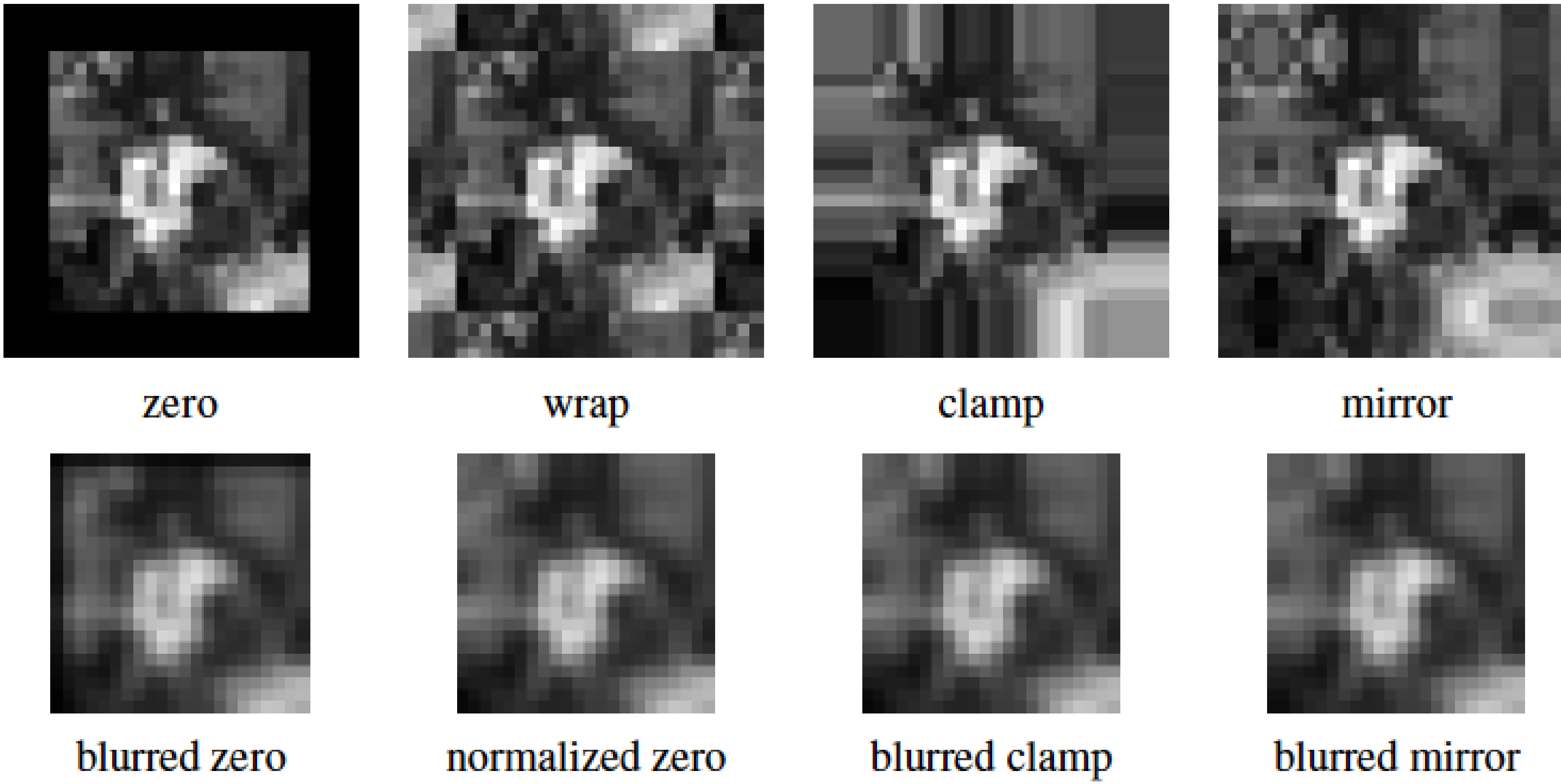

zero
wrap
clamp
mirror
blurred zero
normalized zero
blurred clamp
blurred mirror

Image Extrapolation (Mirroring)

Code

```
J = imread('image.bmp');
figure; imshow(J);
```

J

Image Extrapolation (Mirroring)

Code

```
boarder = 40;
[nr,nc,nb] = size(J);
J_big = zeros(nr+2*boarder, nc + 2*boarder,nb);
J_big(boarder+1:boarder+nr,boarder+1:boarder+nc,:) = J;
```

J_big

Image Extrapolation (Mirroring)

Code

```
for i=1:border,
    for j=1:border,
        J_big(i,j,:) = J(border-i+1,border-j+1,:);
    end
end
```

Mirroring with respect to the boarder

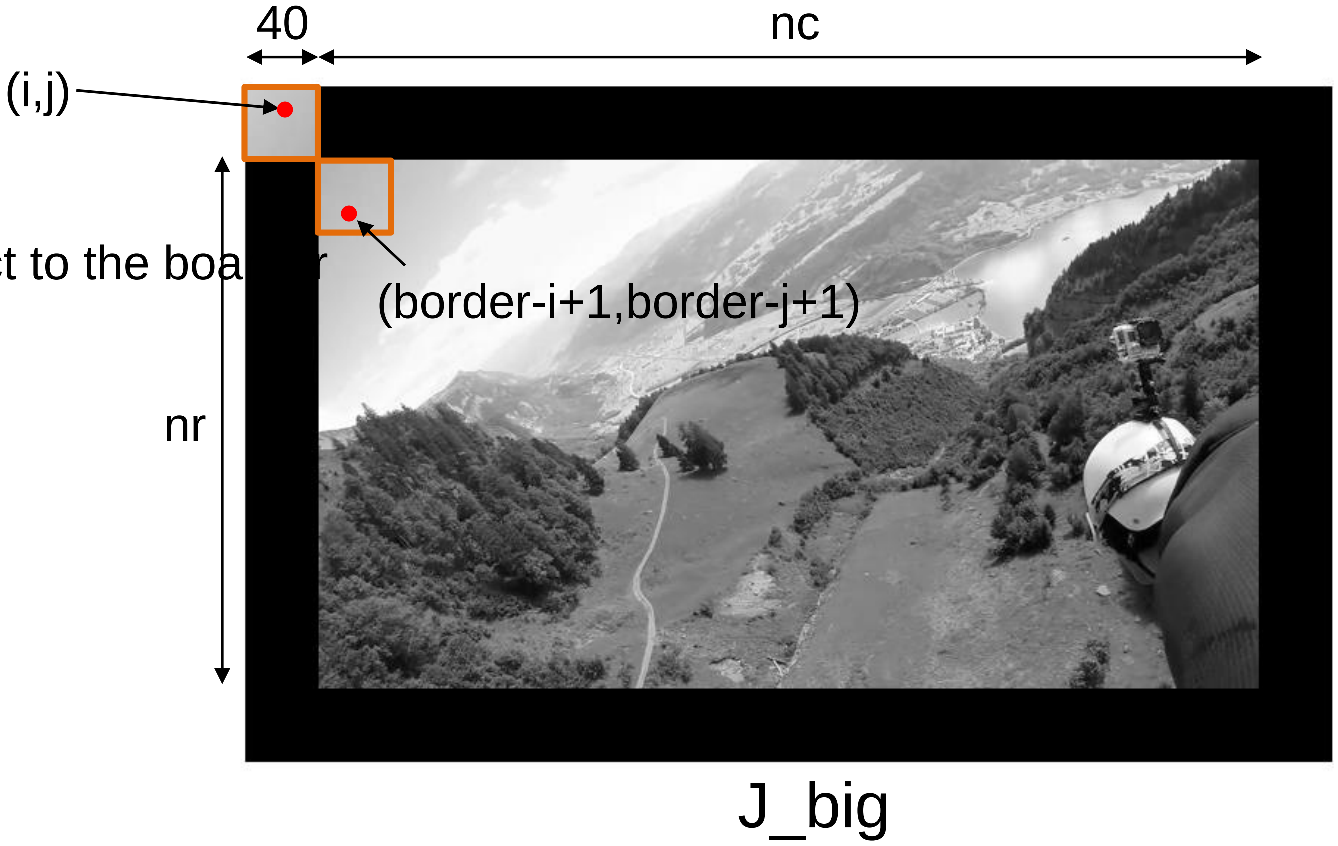


Image Extrapolation (Mirroring)

Code

```
for i=1:boarder,
   for j=border+1:border+nc,
          J_big(i,j,:) = J(border-i+1,j-border,:);
    end
end
```

Mirroring with respect to the boa r

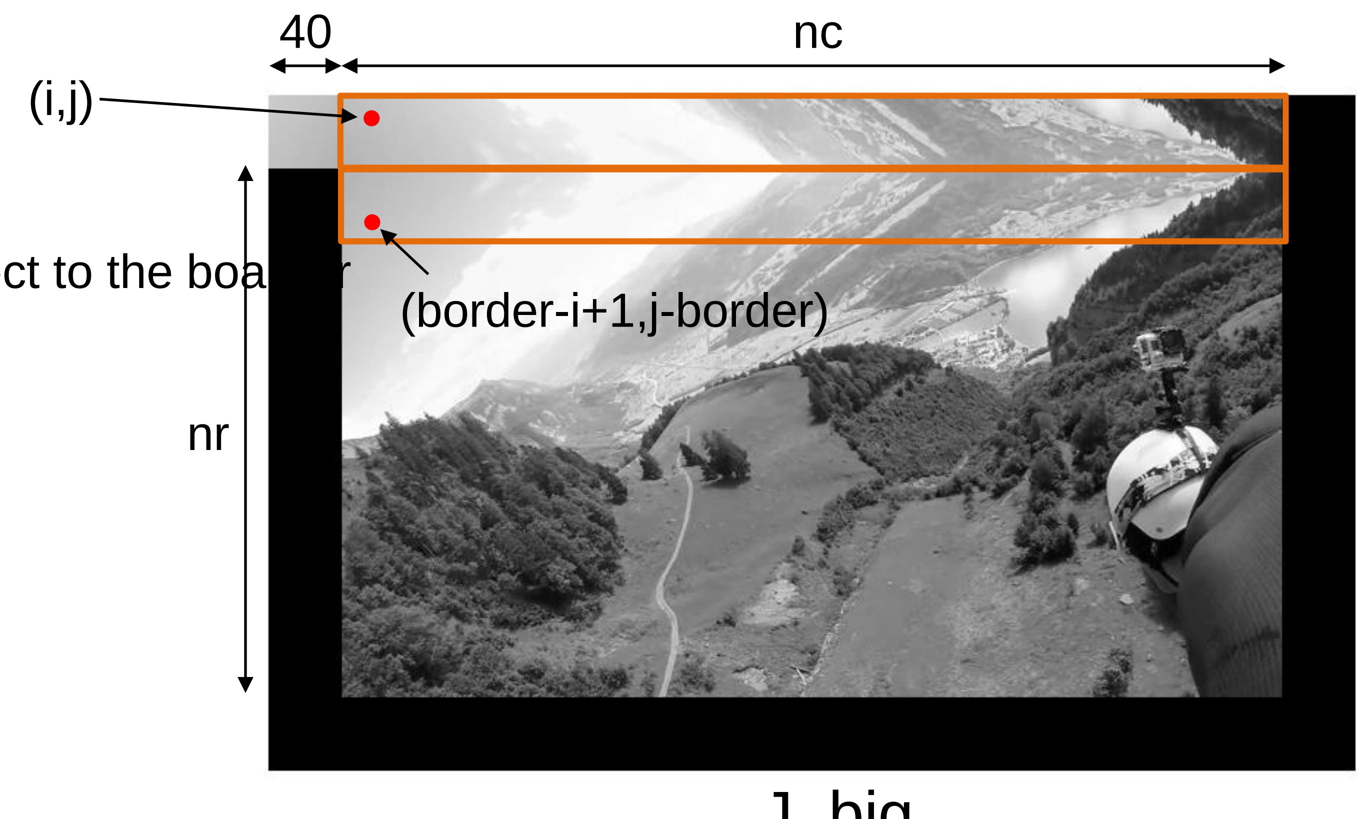


Image Extrapolation (Mirroring)

Code

```
for i=nr+border+1:border*2+nr,
    for j=border+1:border+nc,
        J_big(i,j,:) = J(2*nr-i+border+1,j-border,:);
    end
end
```

Mirroring with respect to the boa r

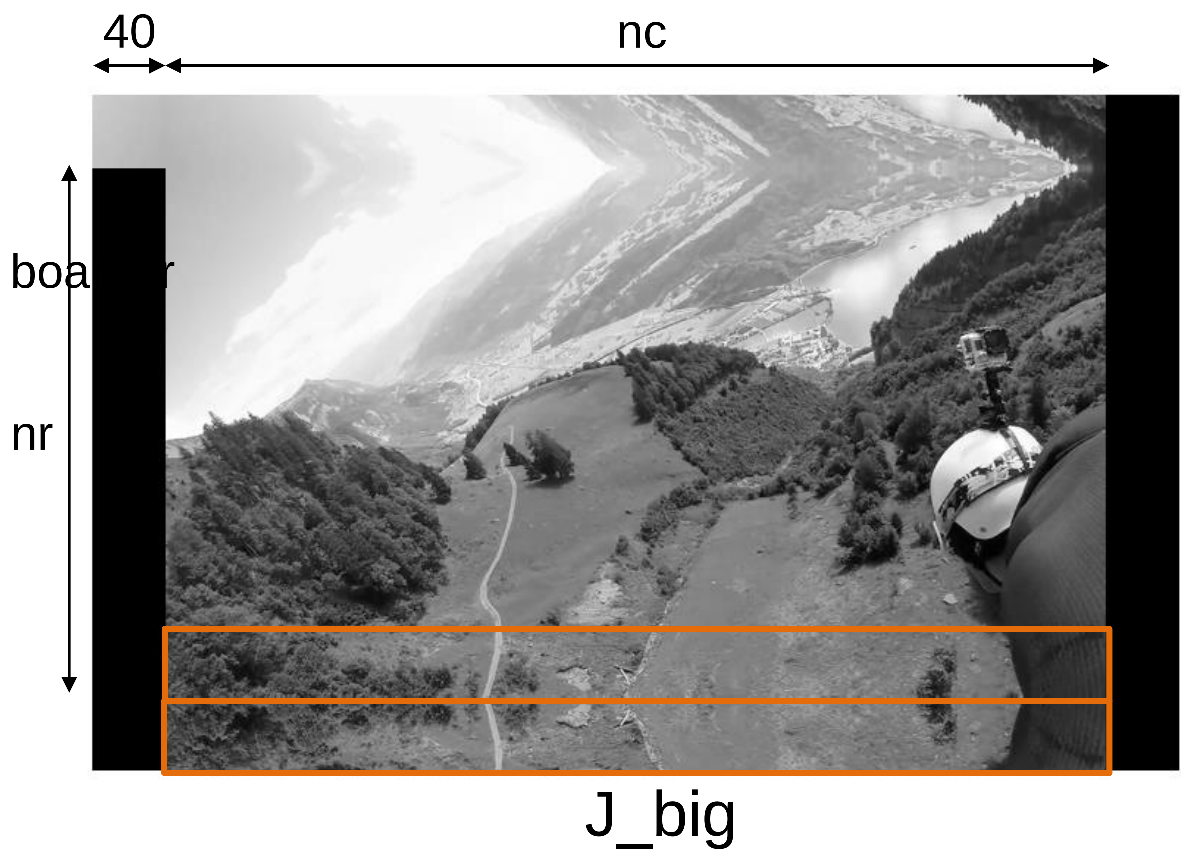
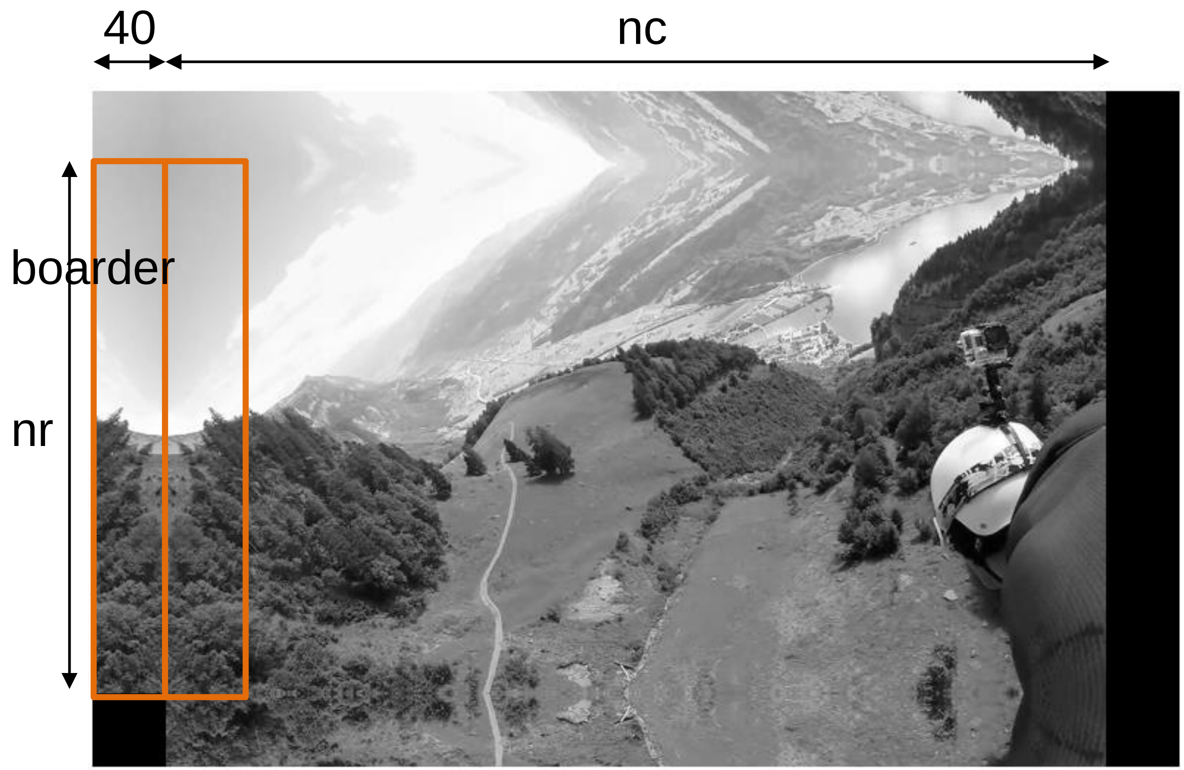


Image Extrapolation (Mirroring)

Code

```
for i=border+1:border+nr;
    for j=1:border,
        J_big(i,j,:) = J(i-border,border-j+1,:);
    end
end
```

Mirroring with respect to the boarder

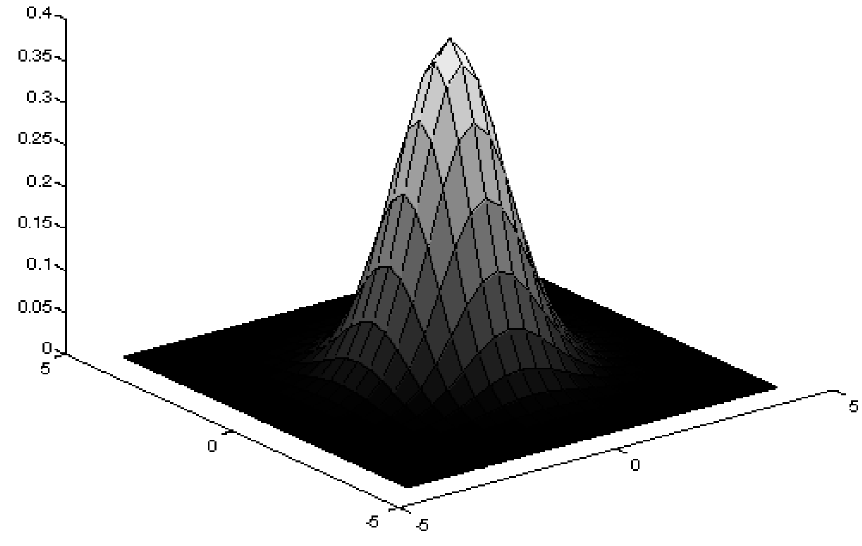


J_big

Examples of image operation as convolution

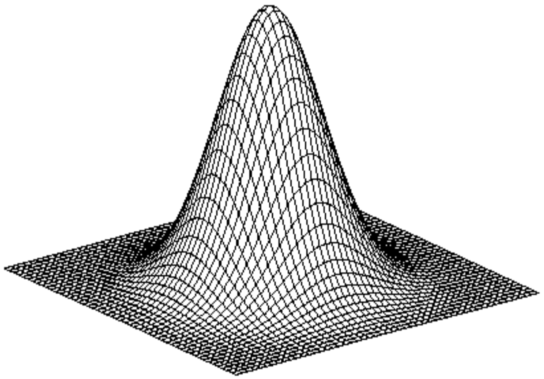
Gaussian Averaging

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



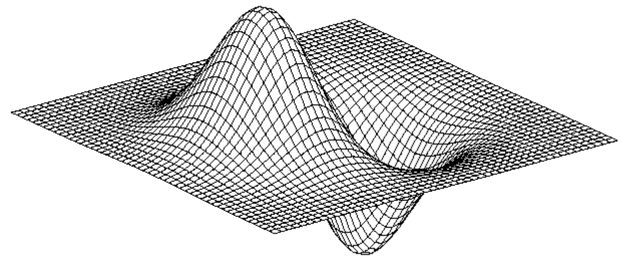
- A Gaussian gives a good model of a fuzzy blob

2D filters, more on this later…



Gaussian

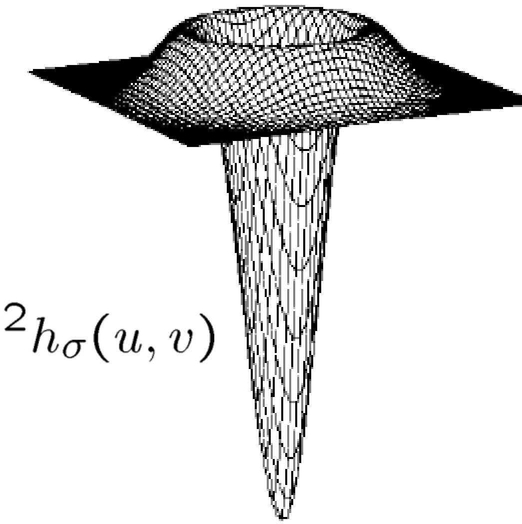
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

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

Laplacian of Gaussian

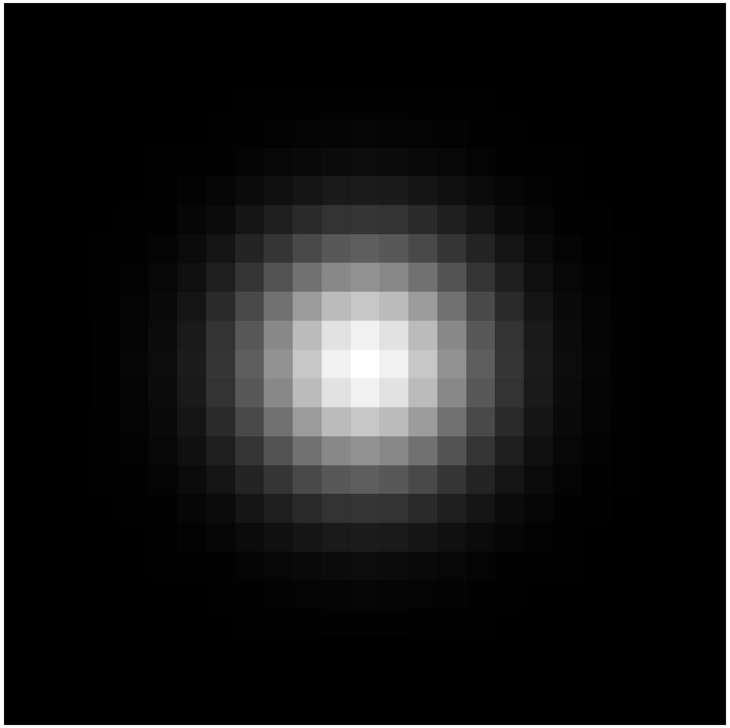


$$\nabla^2 h_\sigma(u,v)$$

- is the **Laplacian** operator:

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

An Isotropic Gaussian

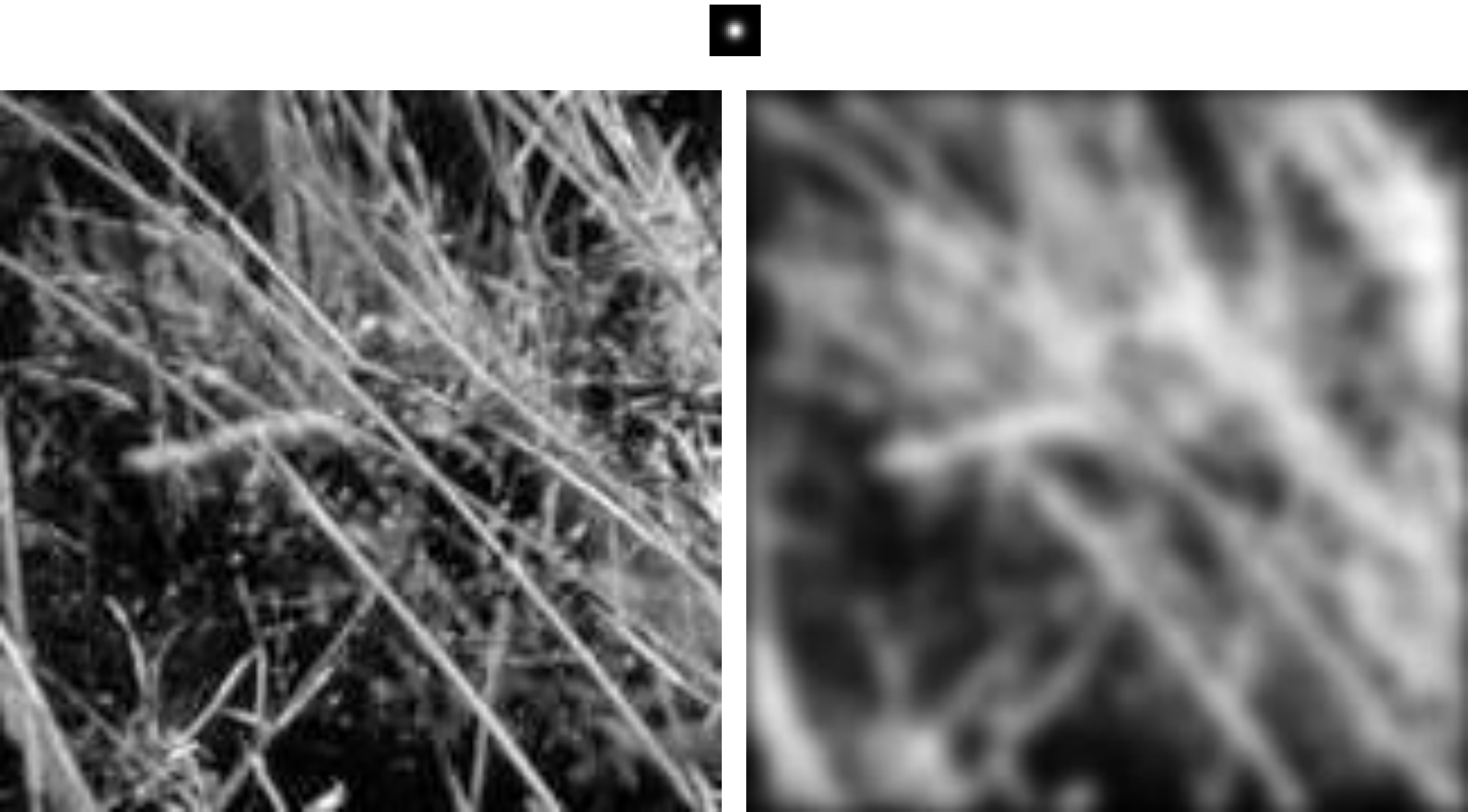


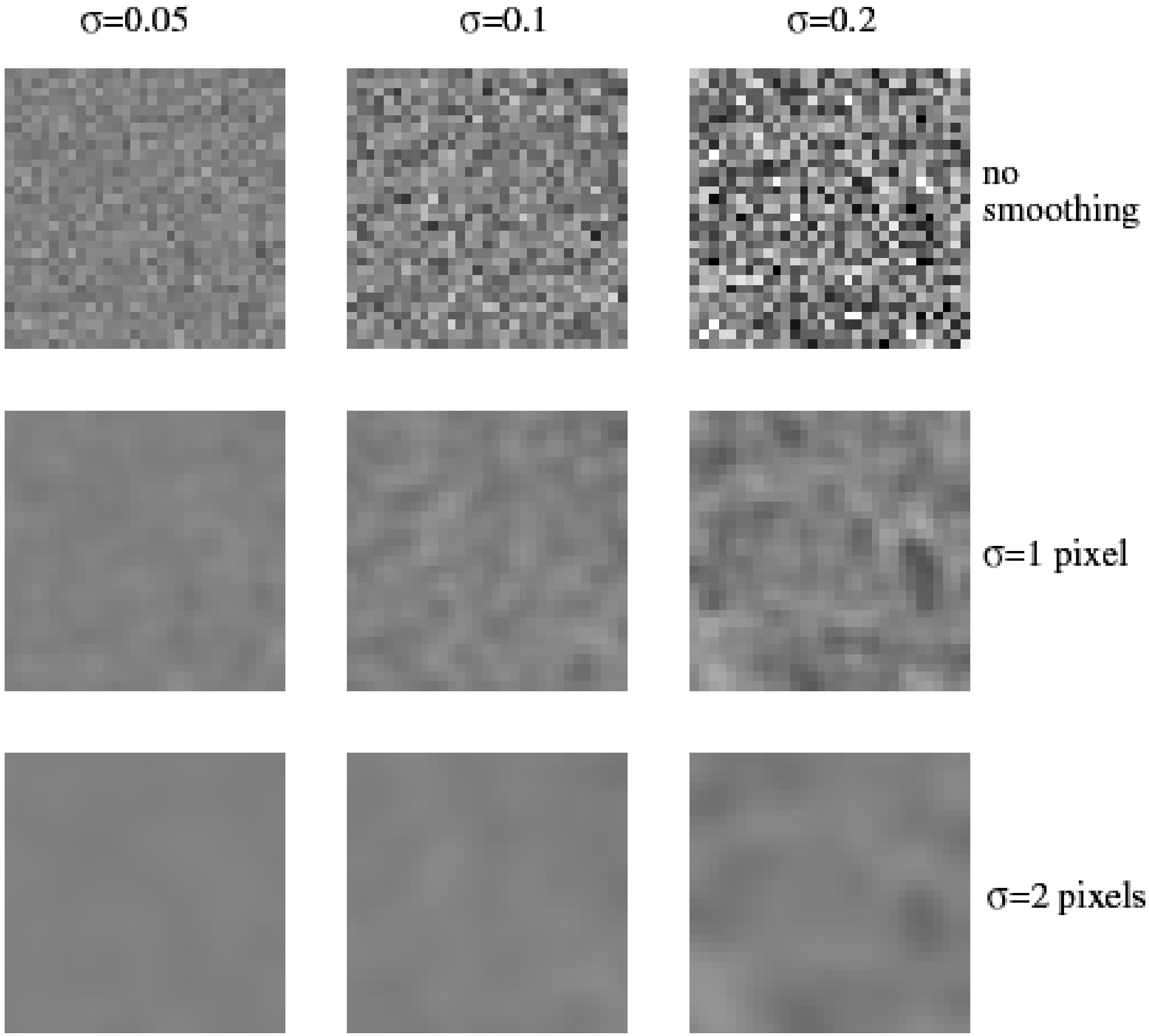
- The picture shows a smoothing kernel proportional to

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

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

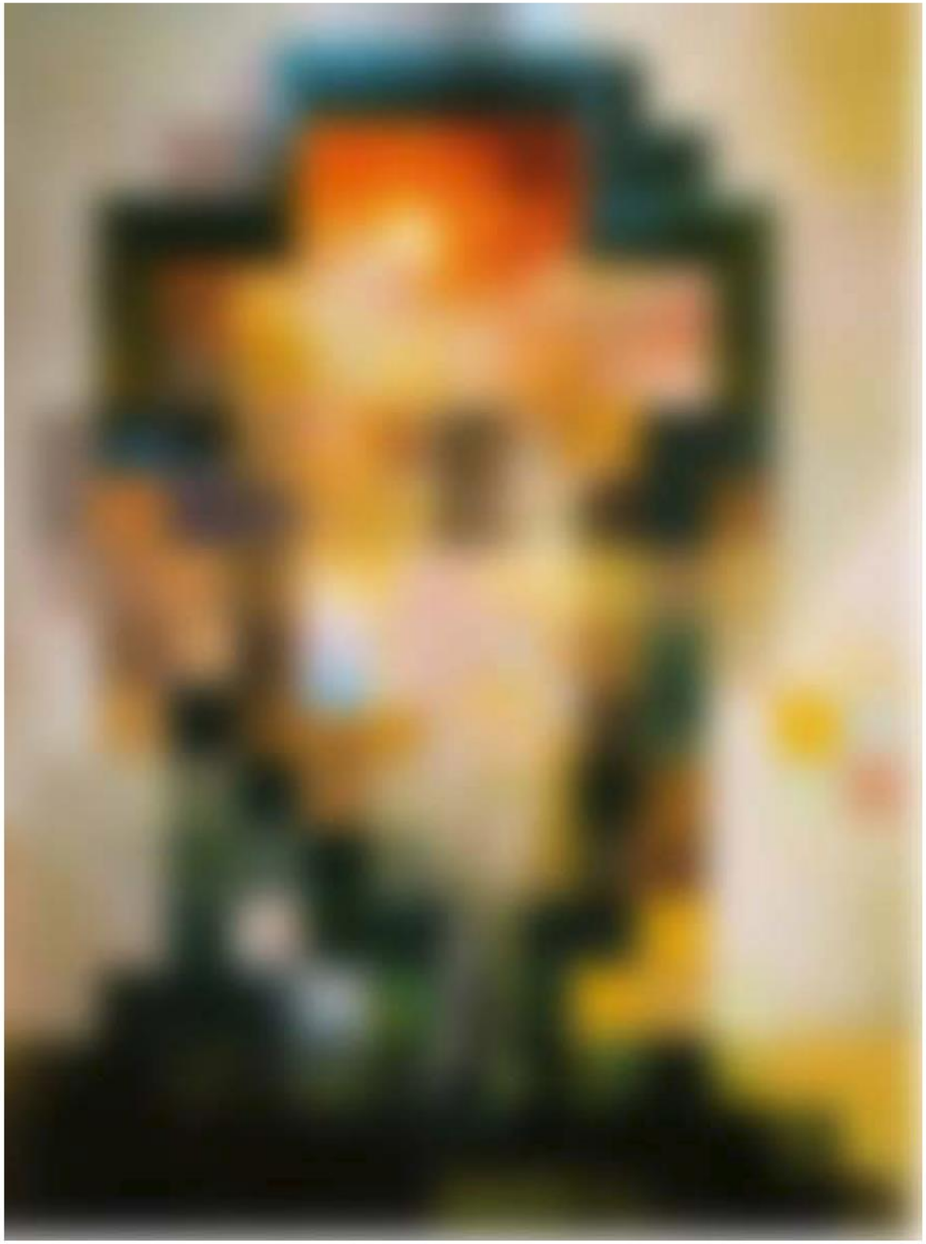
Smoothing with a Gaussian





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

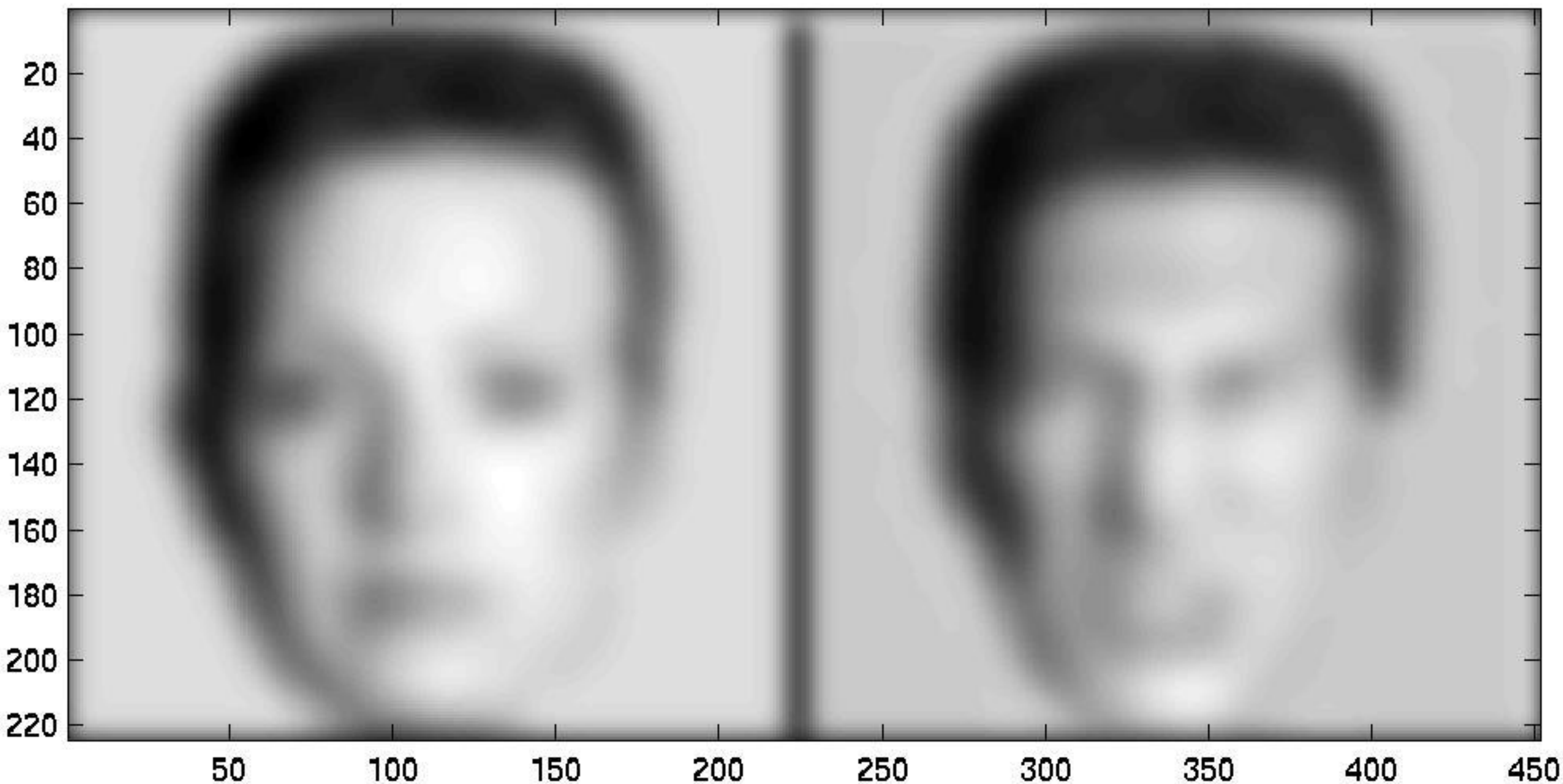
Salvador Dali, *"Gala Contemplating the Mediterranean Sea, which at 30 meters becomes the portrait of Abraham Lincoln*", 1976



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