Image filtering

Olac Fuentes
Computer Science Department
University of Texas at El Paso

Image filtering

- Modify the pixels in an image based on some function of a local neighborhood of each pixel

10	5	3
4	5	1
1	1	7

Local image data

Some function →

	7	

Modified image data

Source: L. Zhang

Image filtering

Why?

- Reduce noise
- Make important information more salient
- Find simple patterns

Source: L. Zhang

Cross-correlation

Let F be the image, H be the kernel (of size 2k+1 x 2k+1), and G be the output image

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

This is called a **cross-correlation** operation:

$$G = H \otimes F$$

Convolution

- Same as cross-correlation, except that the kernel is "flipped" (horizontally and vertically)

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

This is called a **convolution** operation:

$$G = H * F$$

- Convolution is **commutative** and **associative**

Cross-correlation

```
def correlate2d(image,filt):
    # For convolution, we would need to do filt = filt[::-1,::-1]
    r,c = filt.shape
    result = np.zeros((image.shape[0]-r + 1, image.shape[1]-c + 1))
    for i in range(result.shape[0]):
        for j in range(result.shape[1]):
            result[i,j] = np.sum(image[i:i+r,j:j+c]*filt)#  Dot product of region and filter
    return result
```

Cross-correlation

In practice, we will use the much more efficient implementation provided by Scipy

```
from scipy import signal
image_f = signal.correlate2d(image,f,mode='valid')
```

This code will produce the same results as the one in the previous slide

By changing the mode parameter, we can obtain slightly different results

mode = 'same' returns an array of the same size as *image.* To do this it 'pads' image with extra rows and columns (with value 0, by default)

Cross-correlation

scipy.signal.correlate2d

scipy.signal.correlate2d(*in1*, *in2*, *mode='full'*, *boundary='fill'*, *fillvalue=0*) [source]

Cross-correlate two 2-dimensional arrays.

Cross correlate *in1* and *in2* with output size determined by *mode*, and boundary conditions determined by *boundary* and *fillvalue*.

Parameters:

in1 : *array_like*

First input.

in2 : *array_like*

Second input. Should have the same number of dimensions as *in1*.

mode : *str {'full', 'valid', 'same'}, optional*

A string indicating the size of the output:

`full`

The output is the full discrete linear cross-correlation of the inputs. (Default)

`valid`

The output consists only of those elements that do not rely on the zero-padding. In 'valid' mode, either *in1* or *in2* must be at least as large as the other in every dimension.

`same`

The output is the same size as *in1*, centered with respect to the 'full' output.

Cross-correlation

$$\begin{array}{|c|c|}\hline -1 & 1 \\ \hline -1 & 1 \\ \hline \end{array} \otimes \begin{array}{|c|c|c|c|c|}\hline 3 & 4 & 5 & 6 & 0 \\ \hline 2 & 1 & 6 & 1 & 3 \\ \hline 5 & 4 & 5 & 4 & 5 \\ \hline 3 & 7 & 7 & 4 & 2 \\ \hline 6 & 4 & 8 & 6 & 3 \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline 0 & & & \\ \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array}$$

Cross-correlation

-1	1
-1	1

$\otimes$

3	4	5	6	0
2	1	6	1	3
5	4	5	4	5
3	7	7	4	2
6	4	8	6	3

=

0	6		

Cross-correlation

-1	1
-1	1

$\otimes$

3	4	5	6	0
2	1	6	1	3
5	4	5	4	5
3	7	7	4	2
6	4	8	6	3

=

0	6	-4	

Cross-correlation

-1	1
-1	1

$\otimes$

3	4	5	6	0
2	1	6	1	3
5	4	5	4	5
3	7	7	4	2
6	4	8	6	3

=

0	6	-4	-4

Cross-correlation

-1	1
-1	1

$\otimes$

3	4	5	6	0
2	1	6	1	3
5	4	5	4	5
3	7	7	4	2
6	4	8	6	3

=

0	6	-4	-4
-2			

Cross-correlation

-1	1
-1	1

⊗

3	4	5	6	0
2	1	6	1	3
5	4	5	4	5
3	7	7	4	2
6	4	8	6	3

=

0	6	-4	-4
-2	6		

Cross-correlation

-1	1
-1	1

$\otimes$

3	4	5	6	0
2	1	6	1	3
5	4	5	4	5
3	7	7	4	2
6	4	8	6	3

=

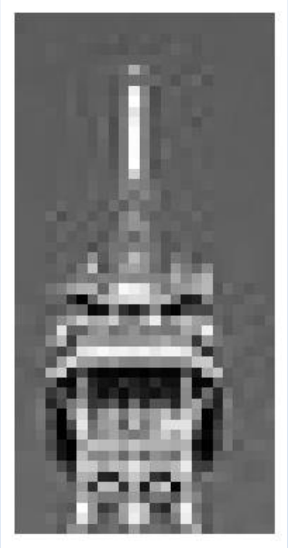
0	6	-4	-4
-2	6	-6	3
3	1	-4	-1
2	4	-5	-5

Cross-correlation

$\otimes$

0	0	0
0	1	0
0	0	0

=

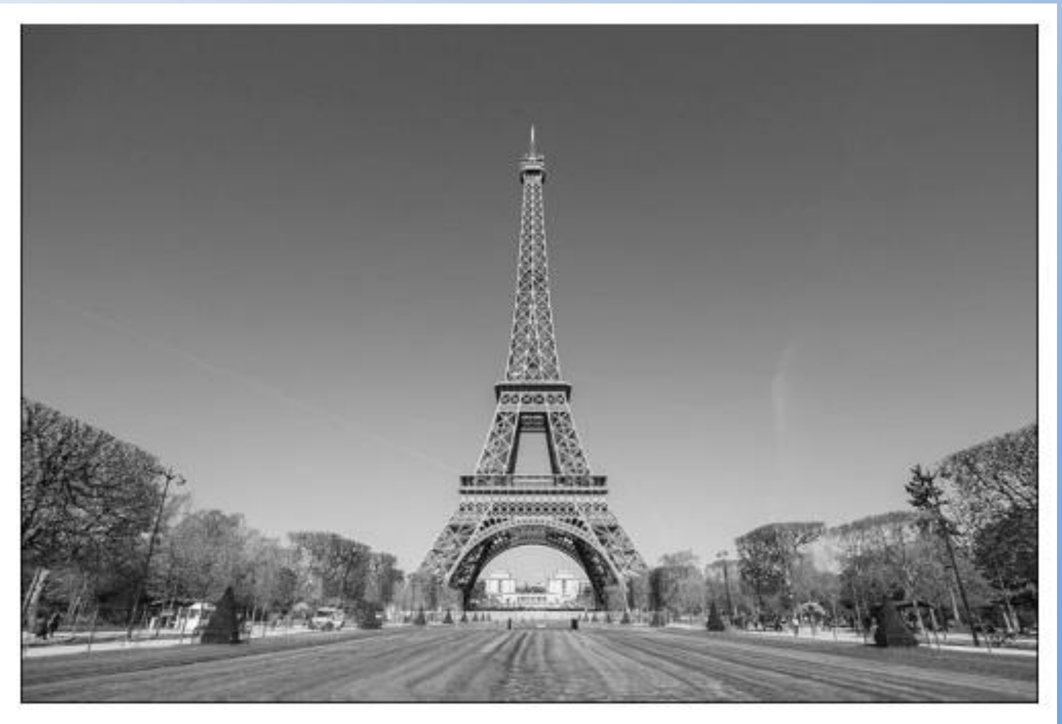


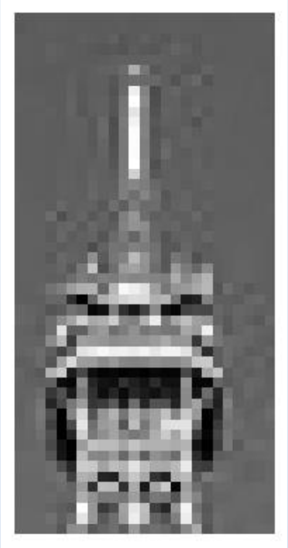
Cross-correlation

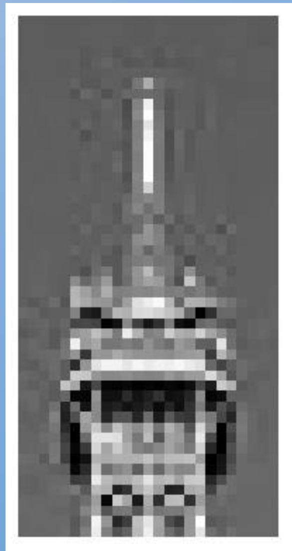
$\otimes$

0	0	0
0	1	0
0	0	0

=





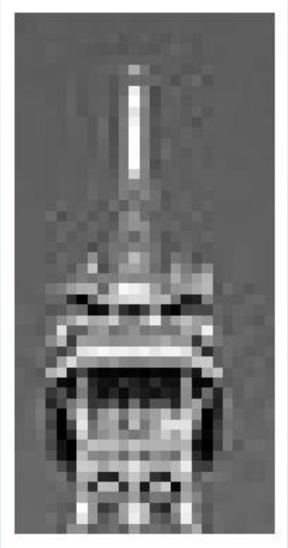


Cross-correlation

$\otimes$

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

=

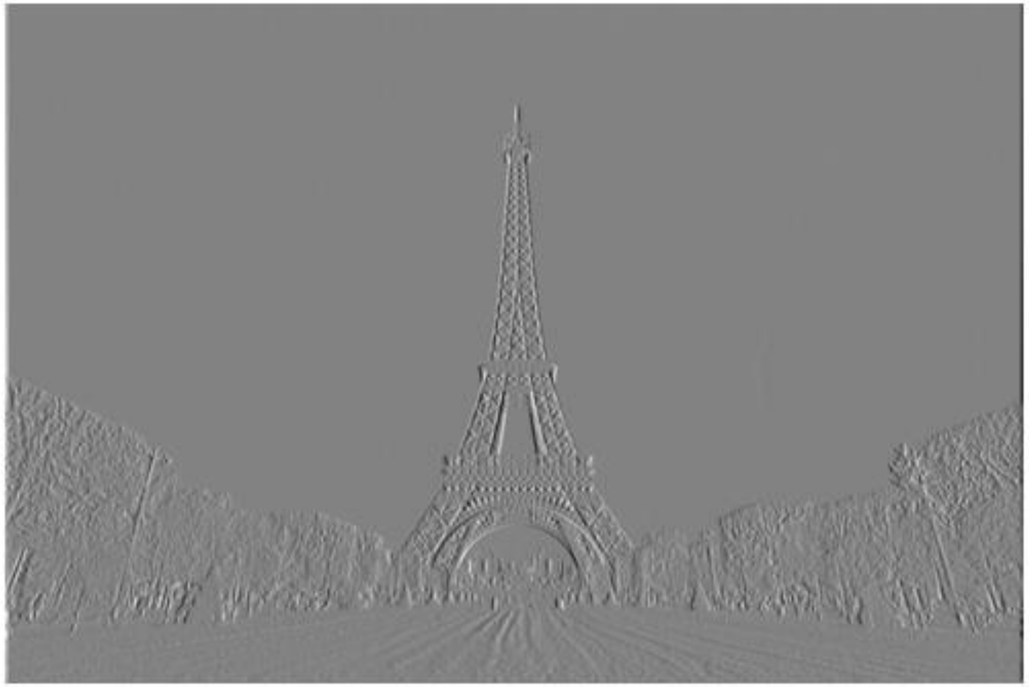


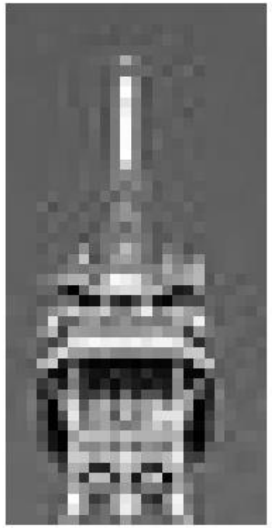
Cross-correlation

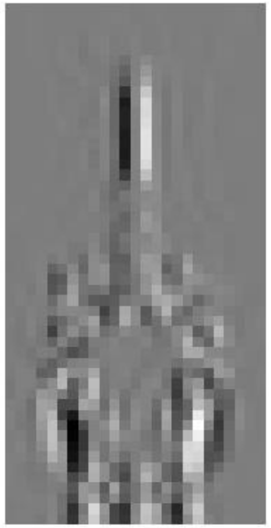
$\otimes$

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

$=$





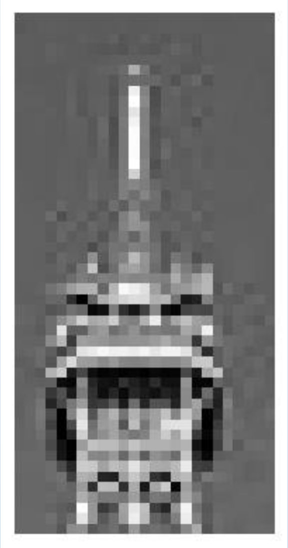


Cross-correlation

$\otimes$

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

=

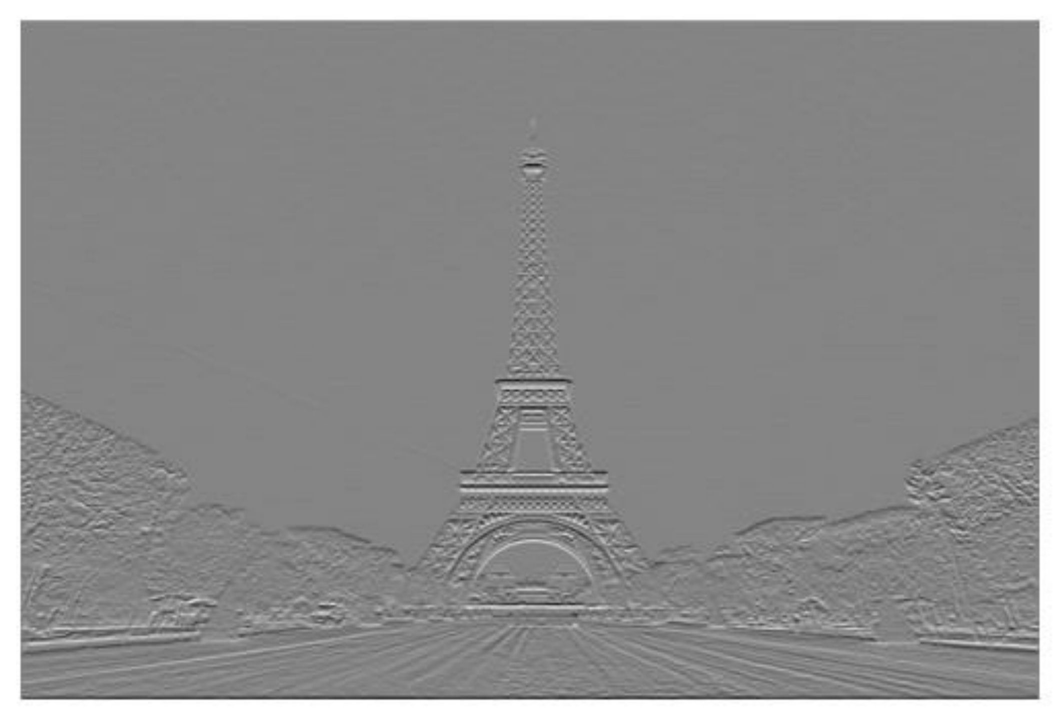


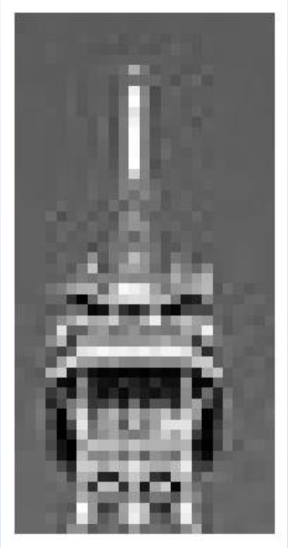
Cross-correlation

$\otimes$

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

=



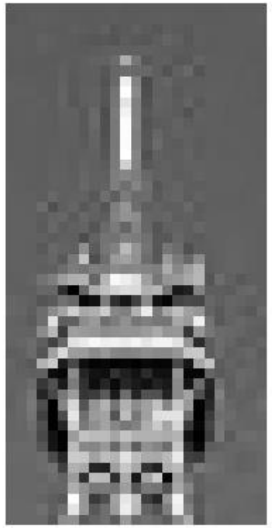


Cross-correlation

$\otimes$

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

=

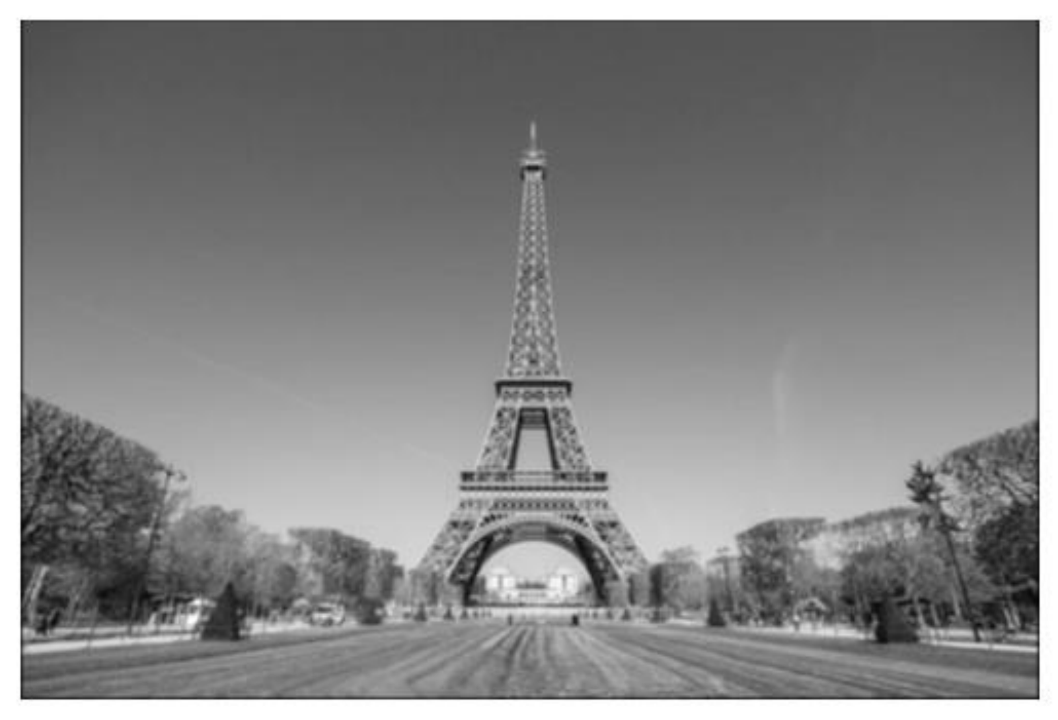


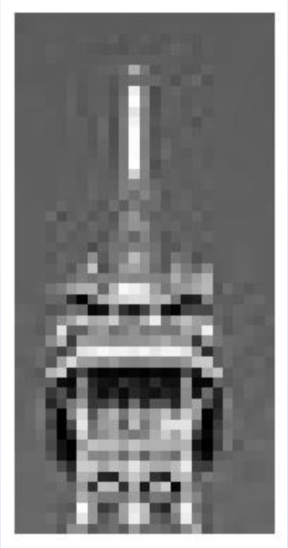
Cross-correlation

⊗

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

=



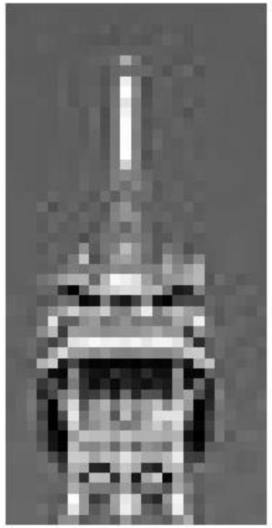


Cross-correlation

$\otimes$

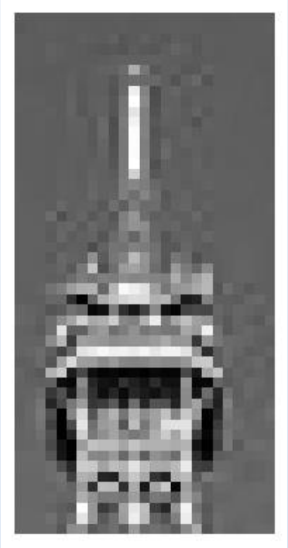
-1/9	-1/9	-1/9
-1/9	17/9	-1/9
-1/9	-1/9	-1/9

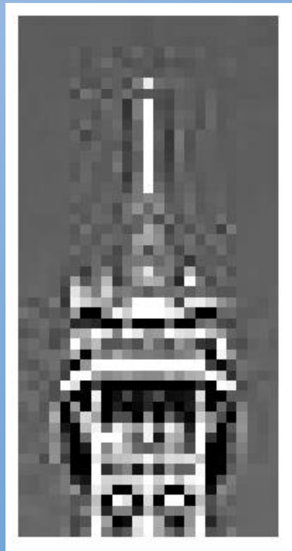
=



Cross-correlation

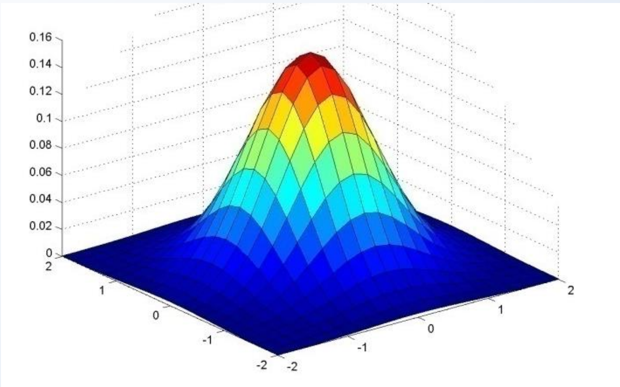
$$\otimes \begin{bmatrix} -1/9 & -1/9 & -1/9 \\ -1/9 & 17/9 & -1/9 \\ -1/9 & -1/9 & -1/9 \end{bmatrix} =$$

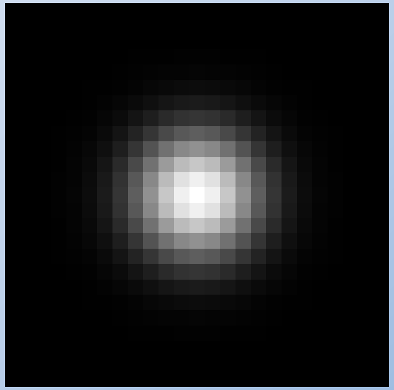




Gaussian Filters

Gaussian filters simulate natural blur in images

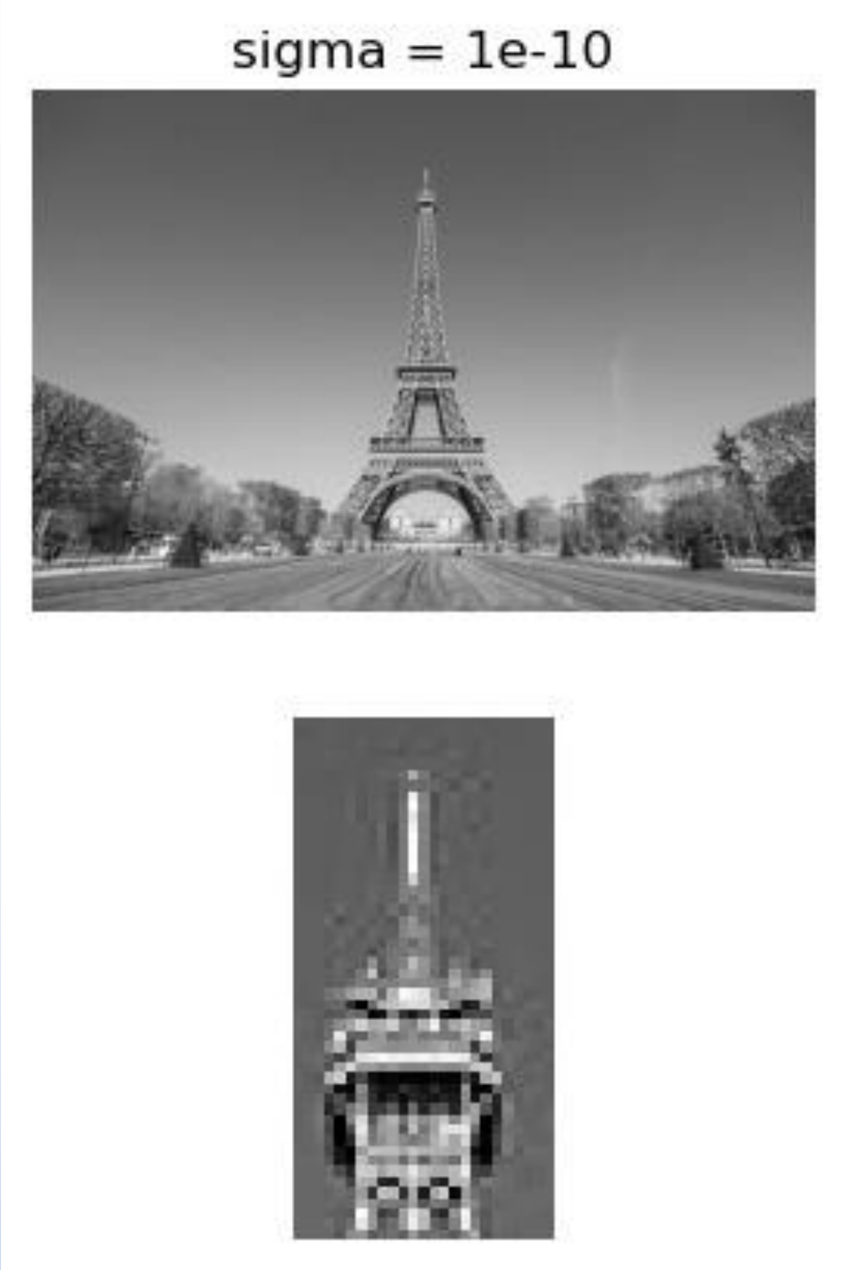


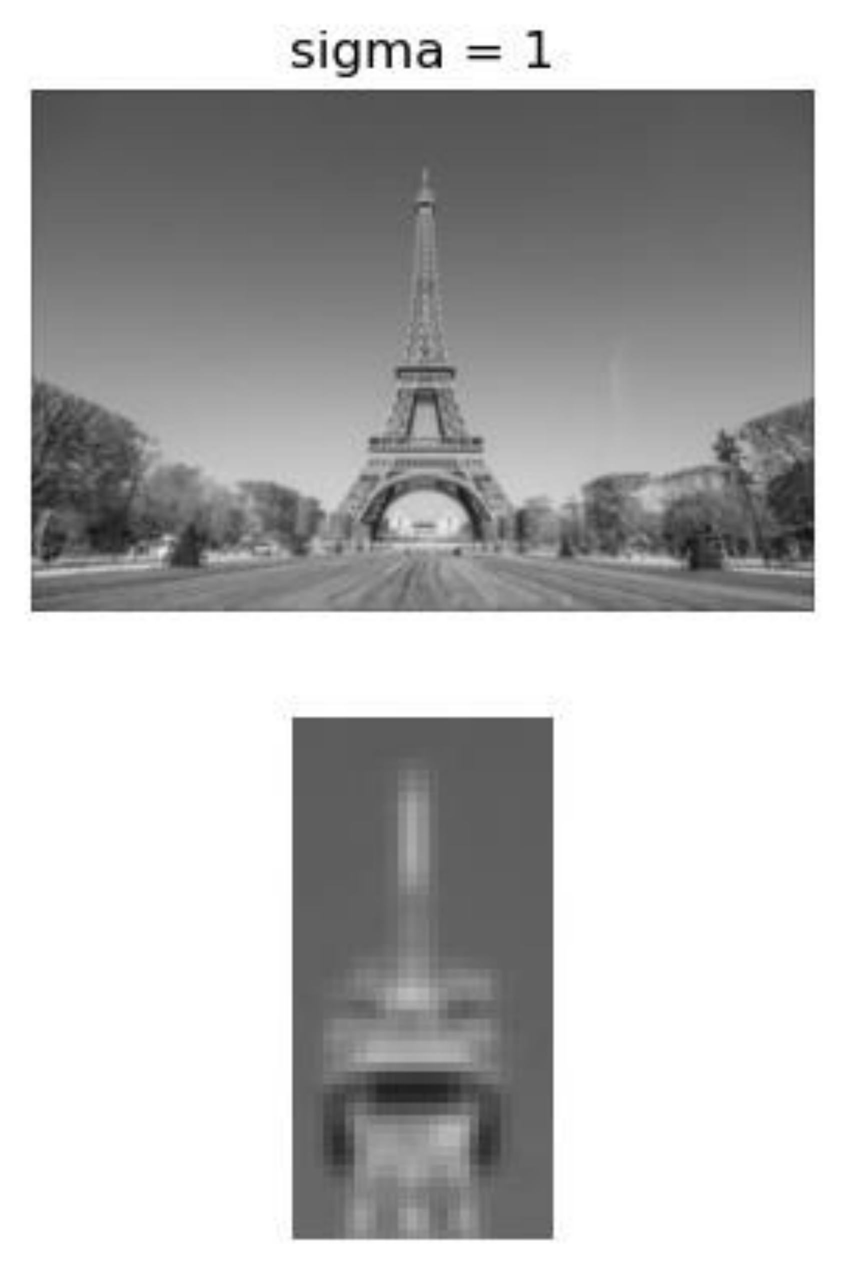


$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

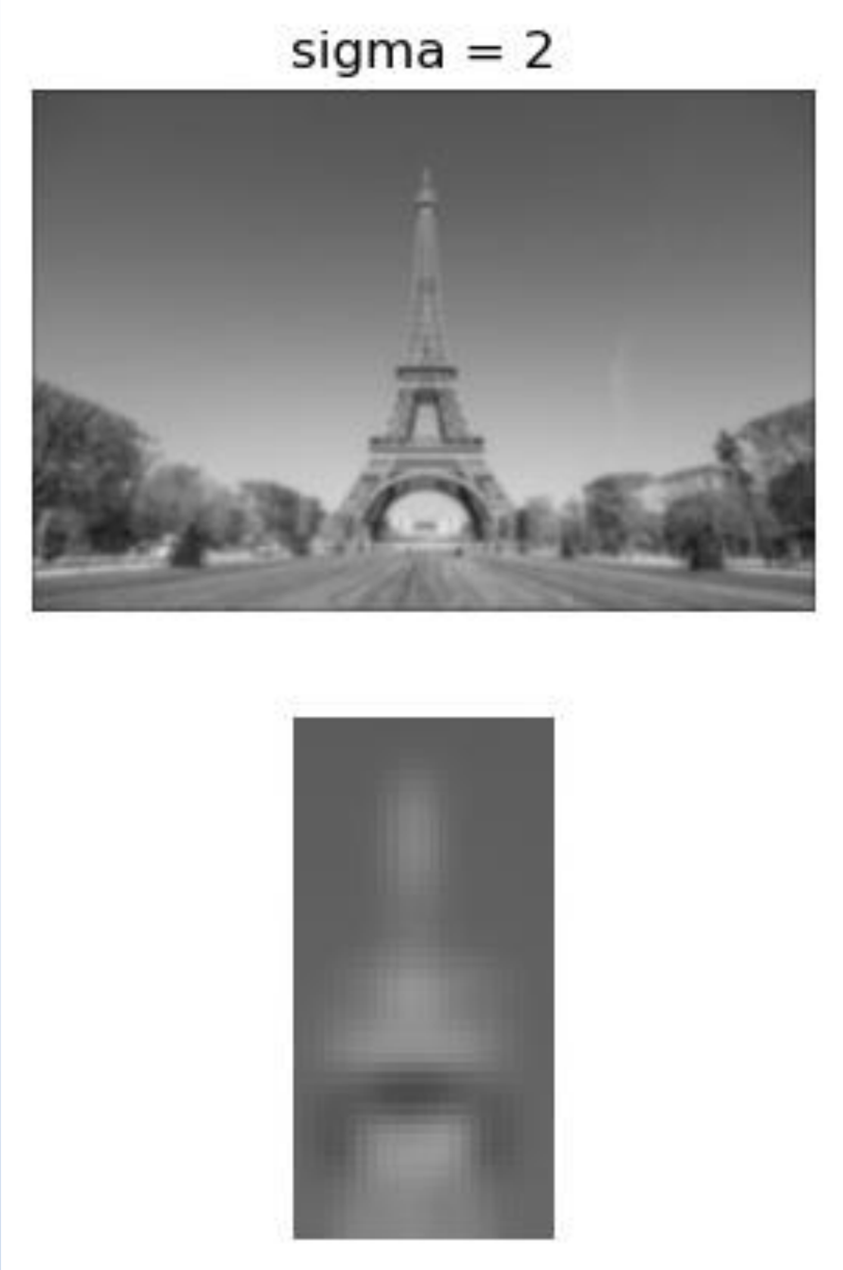
Source: C. Rasmussen

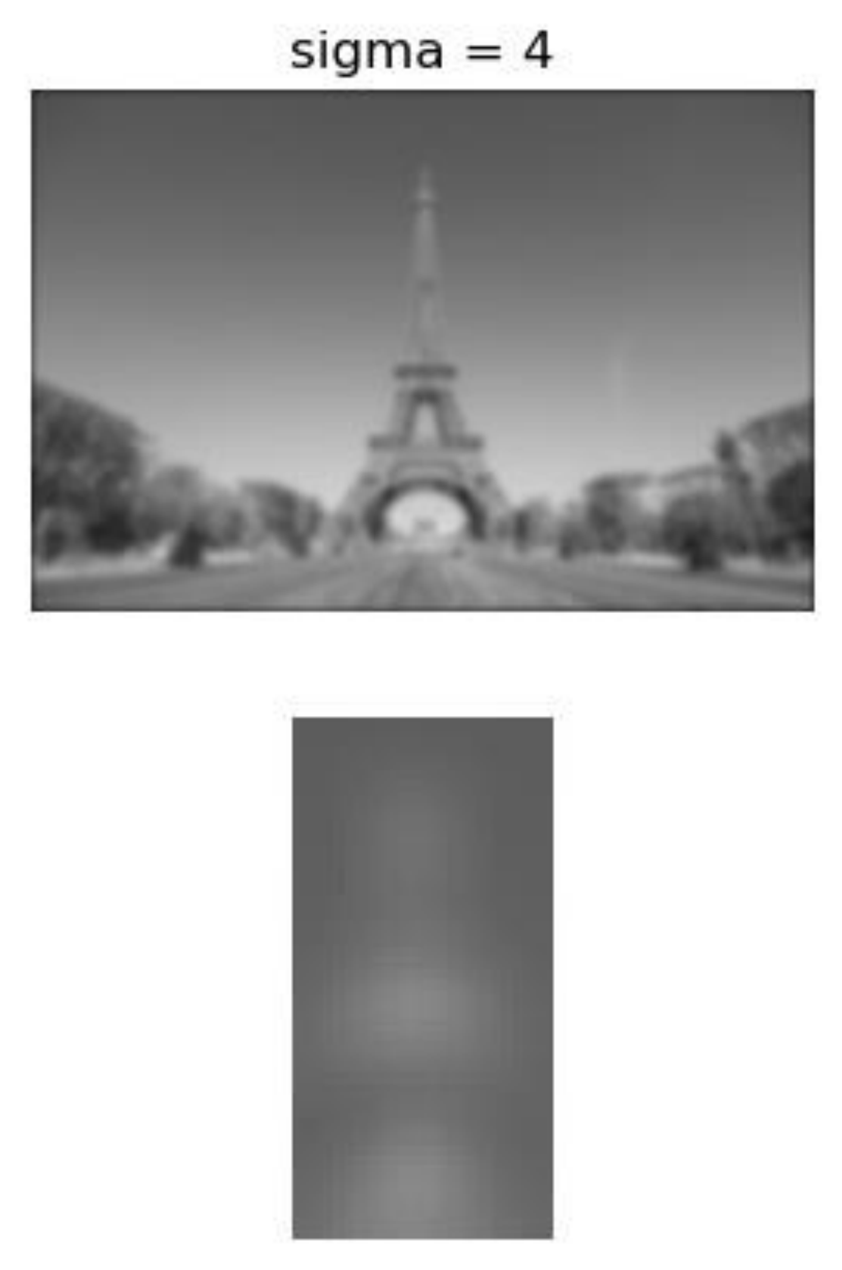
Cross-correlation



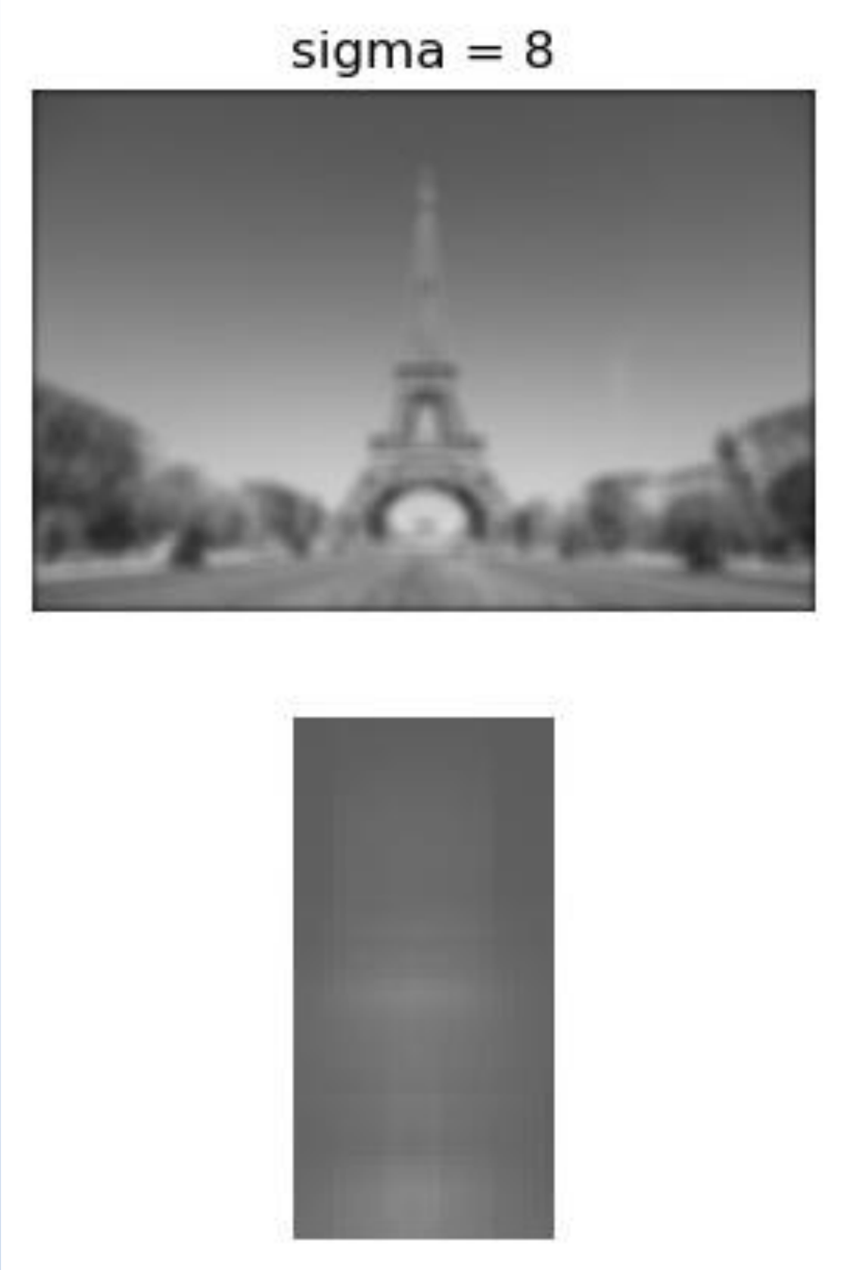


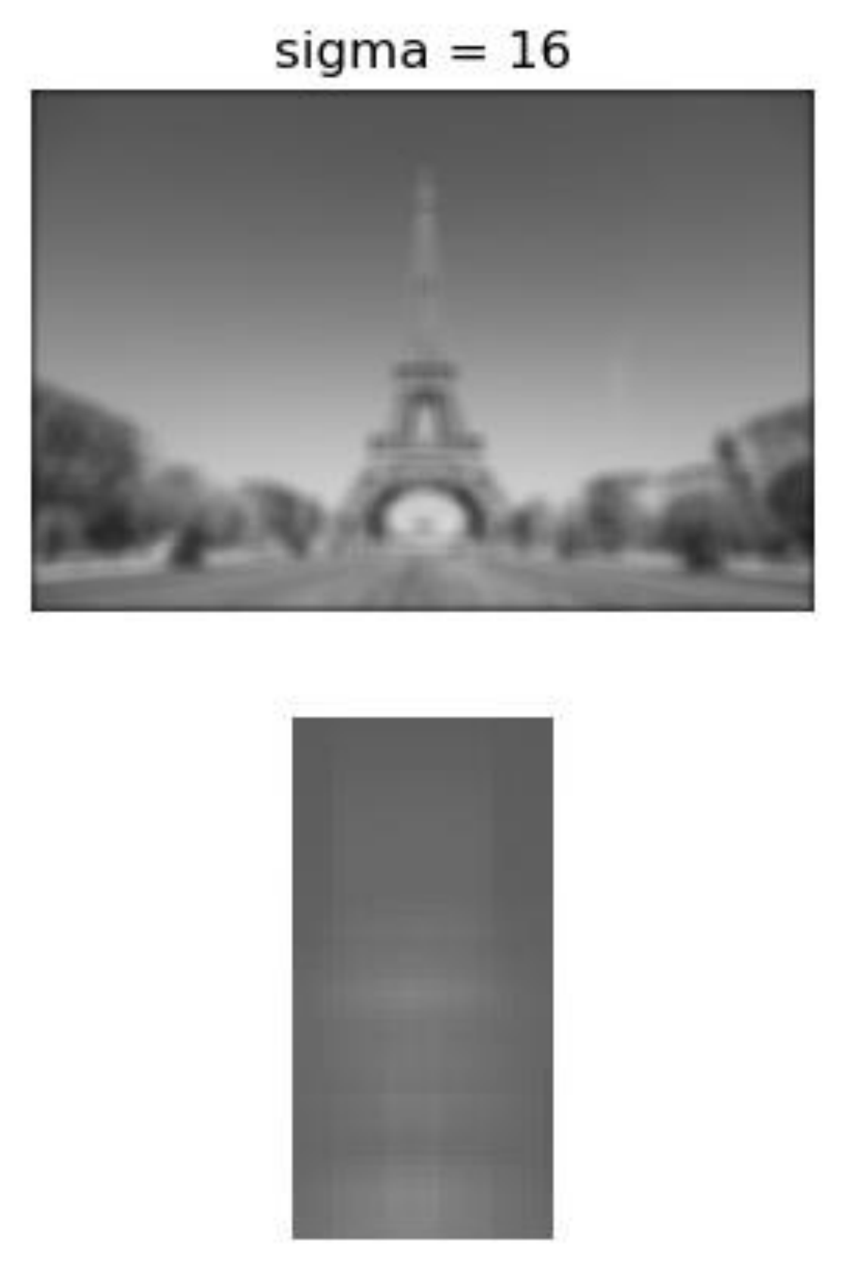
Cross-correlation





Cross-correlation





Gaussian filters for color images

We simply filter each channel separately

Gaussian filters for color images

We simply filter each channel separately

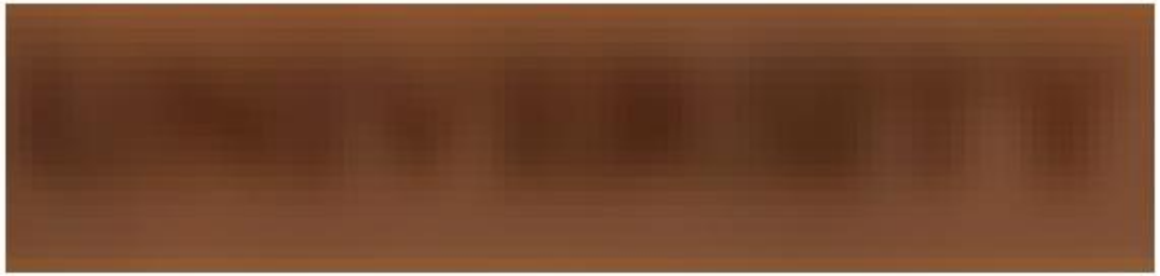
sigma = 1e-10

Gaussian filters for color images

sigma = 1

Gaussian filters for color images

sigma = 2



Gaussian filters for color images

sigma = 3

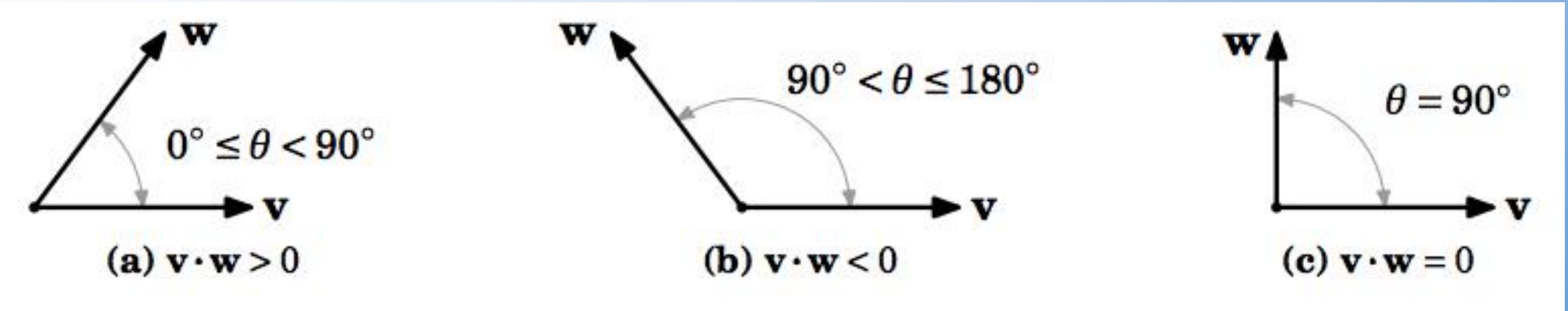
Gaussian filters for color images

sigma = 6

Gaussian filters for color images

Pattern Matching

The similarity of two vectors can be measured by the angle they form:



v and *w* are similar

$\cos\theta > 0$

v and *w* are very different

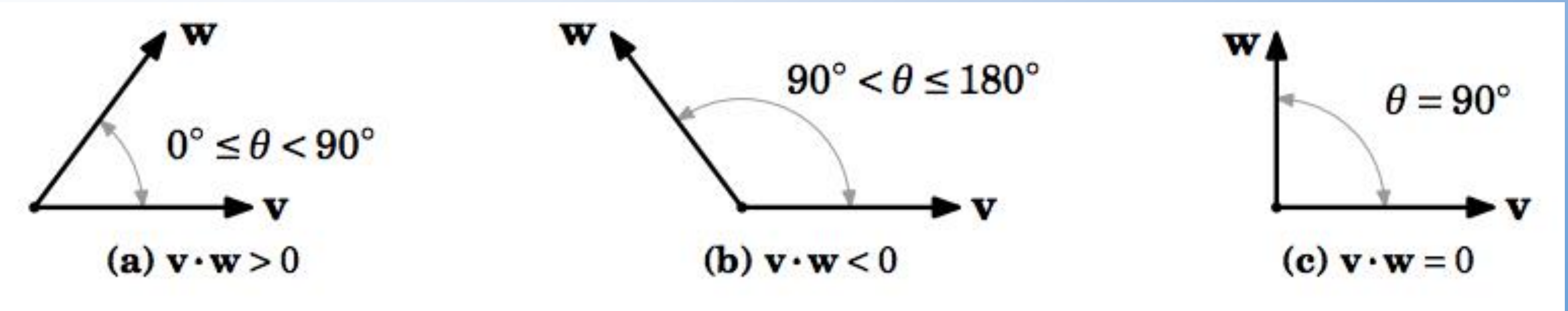
$\cos\theta < 0$

v and *w* are somewhat different

$\cos\theta = 0$

Pattern Matching

The similarity of two vectors can be measured by the angle they form:



$\cos \theta = \dfrac{v \cdot w}{|v|\,|w|}$ where: $v \cdot w = \sum_{k=0}^{n-1} vk\ wk$

$$|v| = \sqrt{\sum_{k=0}^{n-1} vk^{\,2}}$$

Pattern Matching

If we view an *n-by-n* pattern as a vector in a space of n^2 dimensions, the similarity of the pattern with every region in an image is given by the cosine of the angle that the pattern makes with every *n-by-n* region in the image.

$\otimes$

Example

P:

-1	1
-1	1

(size: r-by-c)

I:

3	4	5	6	0
2	1	6	1	3
5	4	5	4	5
3	7	7	4	2
6	4	8	6	3

C:

$$C[i,j] = \cos(P, I[i:i+r, j:j+c]) = \frac{P \cdot I[i:i+r, j:j+c]}{|P| \; |I[i:i+r, j:j+c]|}$$

$$\frac{[-1,1,-1,1] \cdot [3,4,2,1]}{|[-1,1,-1,1]| \; |[3,4,2,1]|}$$

Pattern Matching

P:

-1	1
-1	1

(size: r-by-c)

I:

3	4	5	6	0
2	1	6	1	3
5	4	5	4	5
3	7	7	4	2
6	4	8	6	3

C:

$$C[i,j] = \cos(P, I[i:i+r, j:j+c]) = \frac{P \cdot I[i:i+r, j:j+c]}{|P| \; |I[i:i+r, j:j+c]|}$$

$$C = \frac{P' \otimes I}{I_{mag}}$$

where $I_{mag}[i,j] = |I[i:i+r, j:j+c]|$

and $P' = \dfrac{P}{|P|}$

Pattern Matching

P:

-1	1
-1	1

(size: r-by-c)

I:

3	4	5	6	0
2	1	6	1	3
5	4	5	4	5
3	7	7	4	2
6	4	8	6	3

C:

$$C[i,j] = \cos(P, I[i:i+r, j:j+c]) = \frac{P \cdot I[i:i+r, j:j+c]}{|P| \; |I[i:i+r, j:j+c]|}$$

$$|P| = \sqrt{((-1)^2 + (1)^2 + (-1)^2 + (1)^2)} = \sqrt{4} = 2$$

P'=

-0.5	0.5
-0.5	0.5

I_{mag} =

5.48	8.83	9.9	6.78
6.78	8.83	8.83	7.14
9.95	11.79	10.3	7.81
10.49	13.34	12.85	8.06

$P' \otimes I$ =

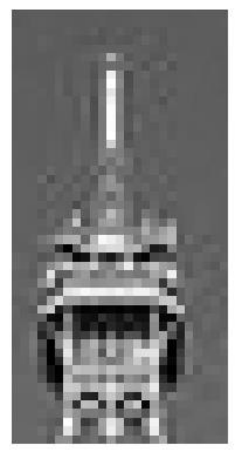
0	3	-2	-2
-1	3	-3	1.5
1.5	0.5	-2	-0.5
1	2	-2.5	-2.5

C =

0.00	0.34	-0.20	-0.30
-0.15	0.34	-0.34	0.21
0.15	0.04	-0.19	-0.06
0.10	0.15	-0.20	-0.31

Pattern Matching

P



I

C