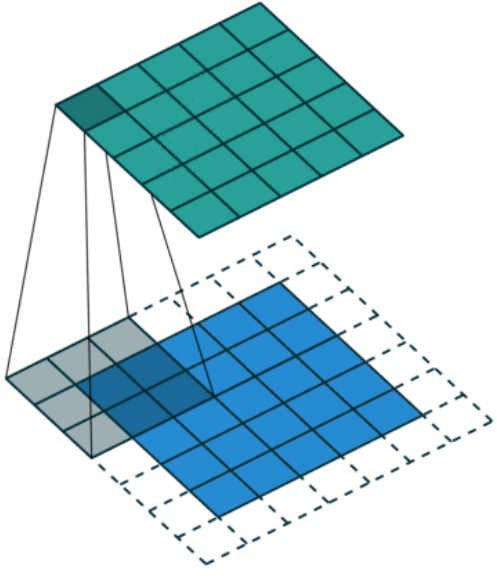
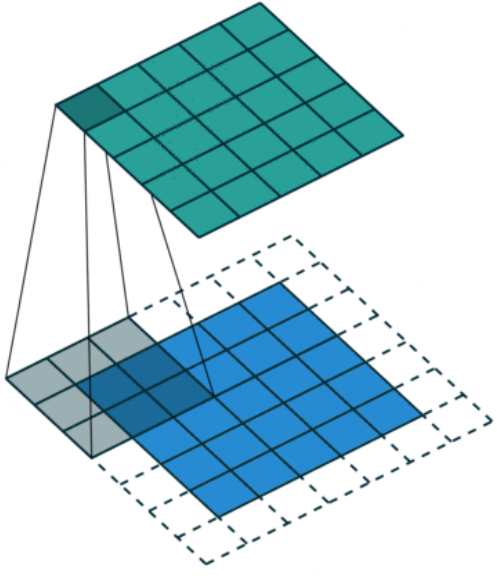
CSCI 497P/597P: Computer Vision



Lecture 3: Convolution and Filtering

CSCI 497P/597P: Computer Vision



Lecture 3: Convolution and Filtering

Goals

- My attempt to be transparent about what I want you to get out of a lecture.
- Helpful for me as well as you.

Goals - Up to this point

- Get a feel for the wide range of problems that fall under the computer vision umbrella.
- Understand how grayscale and color images are represented:
  - On a computer
  - In math
- Know the (informal) definition of image noise.
- Know how to filter (v) an image by cross-correlating it with a given filter (n)/kernel/weights

Goals (today)

- Know how to handle image borders when filtering:
  - output sizes: full / same / valid
  - out-of-bounds values: zeros, reflection, replication
- Understand the distinction between cross-correlation and convolution.
- Know the properties of cross-correlation and convolution:
  - Linearity; shift-invariance; associativity ; commutativity (convolution only)
- Understand the design of several common image filters:
  - Box blur and Gaussian blur
  - Sharpening

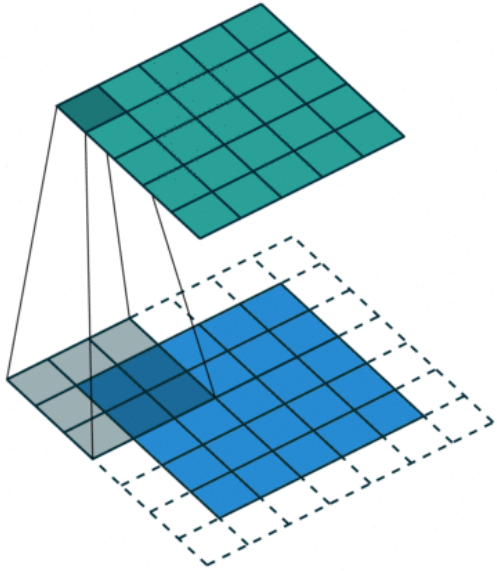
Computing Cross-Correlation

$$g = f \otimes w$$

output image

input image

weights, or filter, or kernel



```
for x = 0 to w:
  for y = 0 to h:
    for i in -k to k:
      for j in -k to k:
        out[x,y] += w[i,j] * in[x+i, y+j]
```

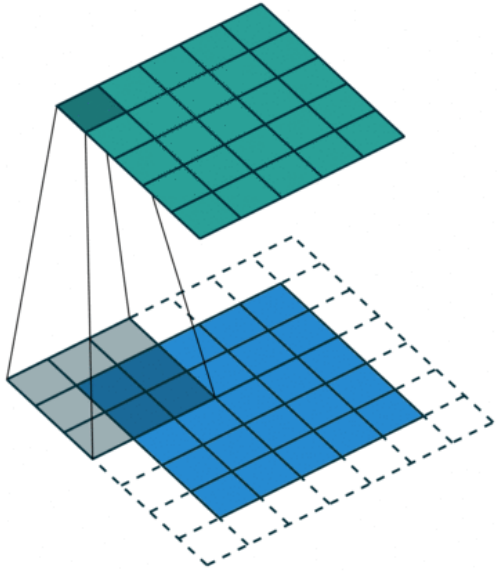
Computing Cross-Correlation

$$g = f \otimes w$$

output image

input image

weights, or filter, or kernel

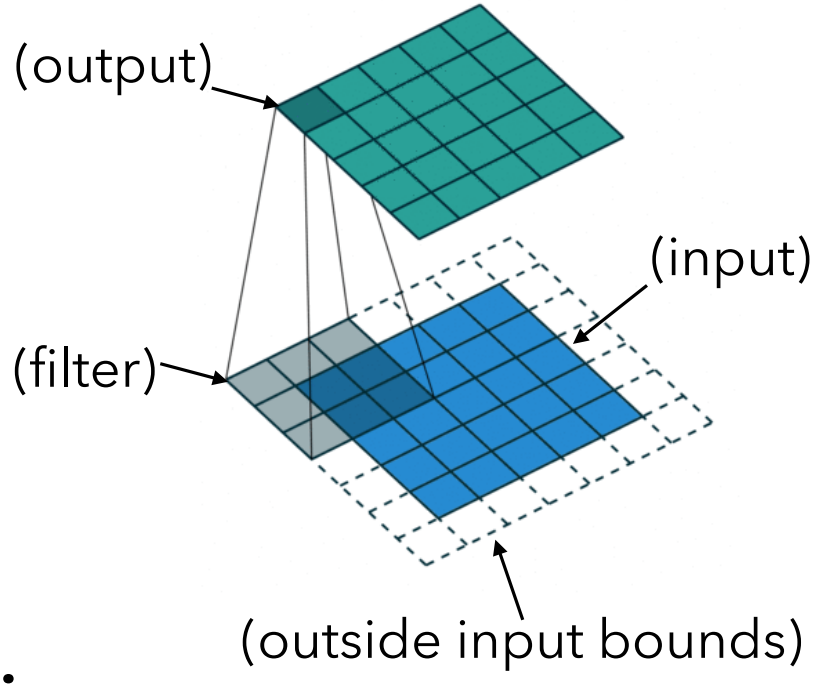


```
for x = 0 to w:
  for y = 0 to h:
    for i in -k to k:
      for j in -k to k:
        out[x,y] += w[i,j] * in[x+i, y+j]
```

Questions remain

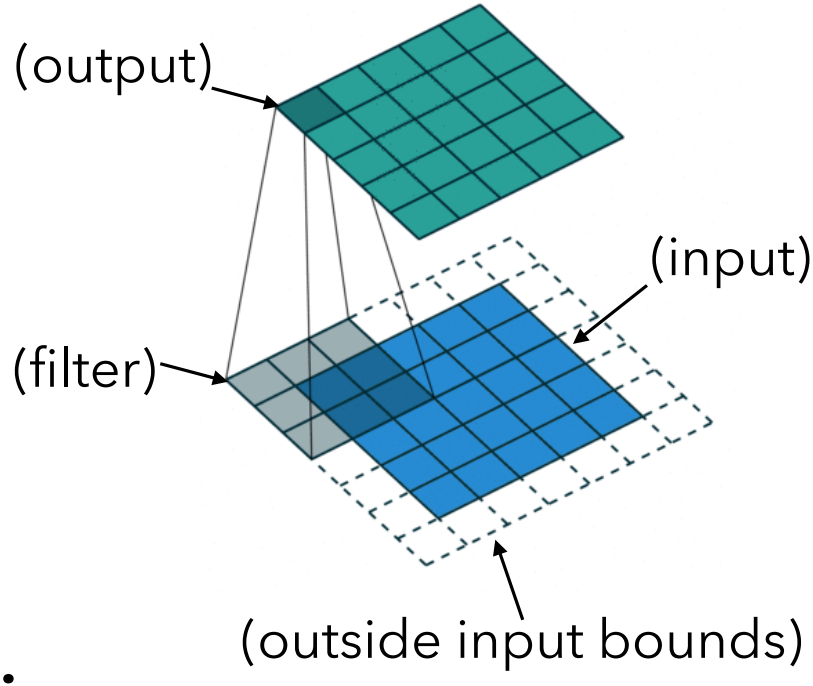
- What happens at the edges?
- What properties does this operator have?
- What can and can't this operator do?

What happens at the border?




```
for x = 0 to w:
  for y = 0 to h:
    for i in -k to k:
      for j in -k to k:
        out[x,y] += w[i,j] * in[x+i, y+j]
```

What happens at the border?

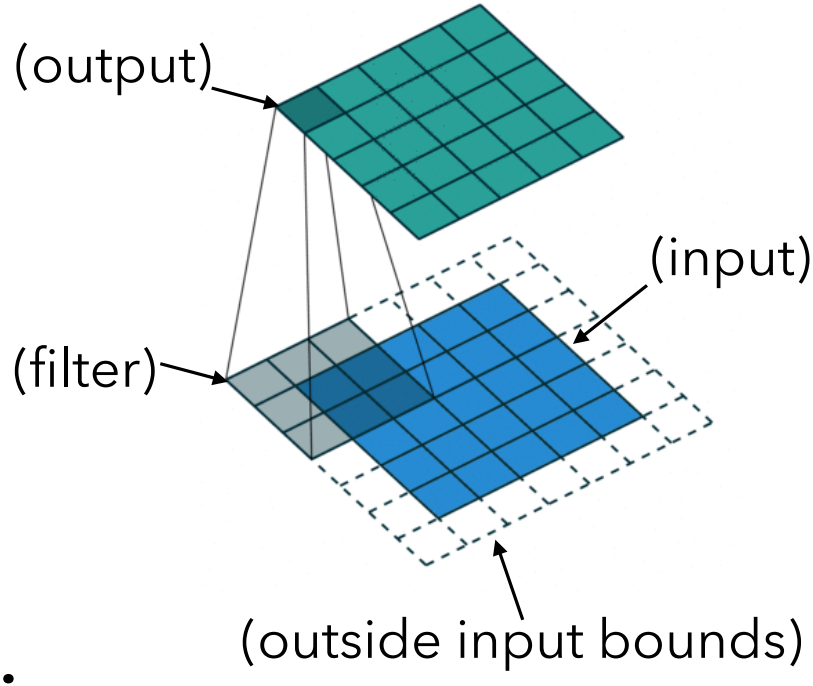



```
for x = 0 to w:
  for y = 0 to h:
    for i in -k to k:
      for j in -k to k:
        out[x,y] += w[i,j] * in[x+i, y+j]
```

What happens at the border?

Two questions:

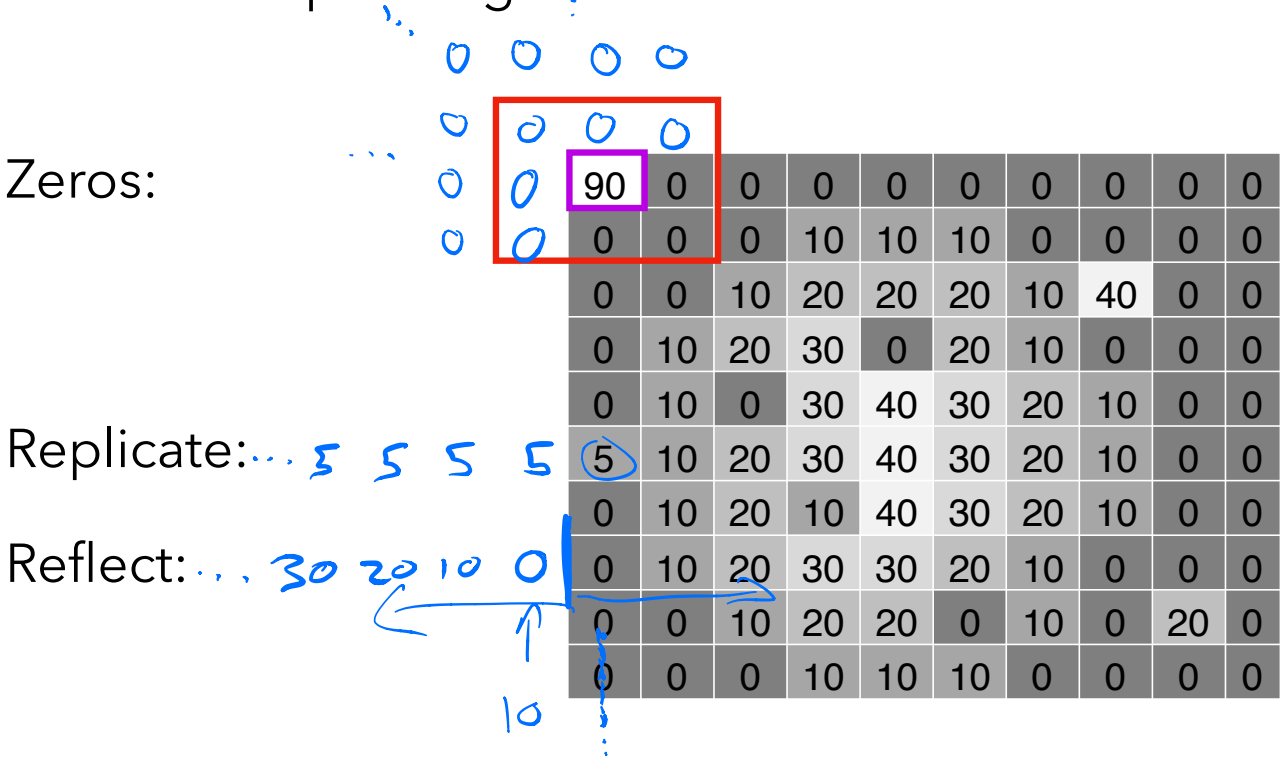
- What are the values outside the input image boundaries?
- What's the size of the output?




```
for x = 0 to w:
  for y = 0 to h:
    for i in -k to k:
      for j in -k to k:
        out[x,y] += w[i,j] * in[x+i, y+j]
```

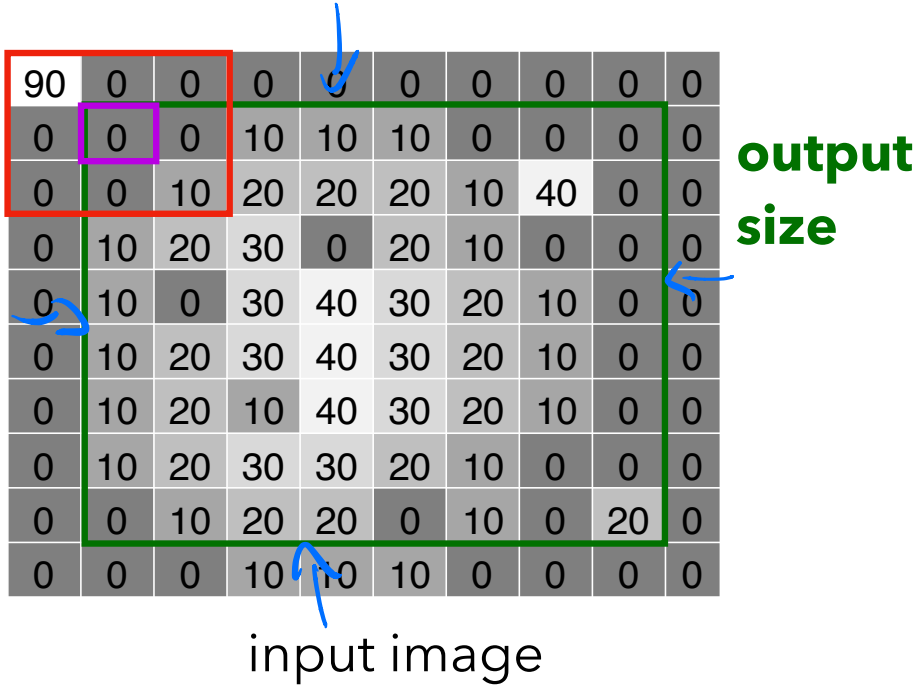
Handling Edges - Values

Possible "padding modes":



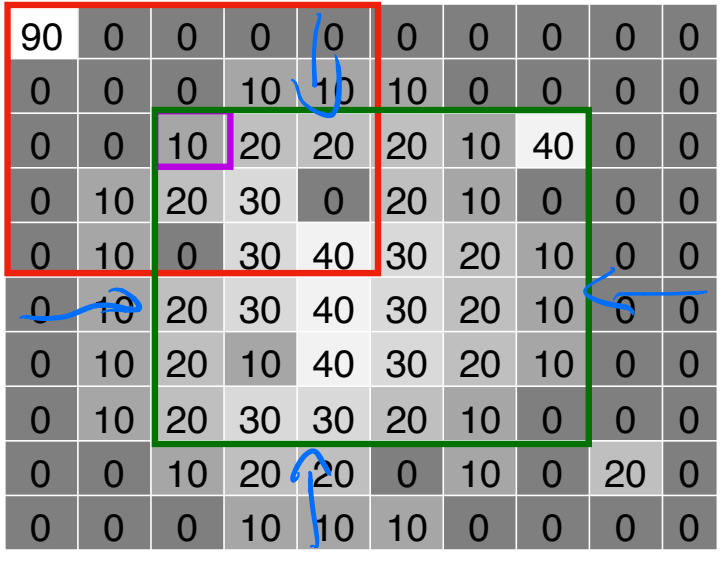
Handling Edges - Sizes

"Valid" (3x3)



Handling Edges

"Valid" (5x5)

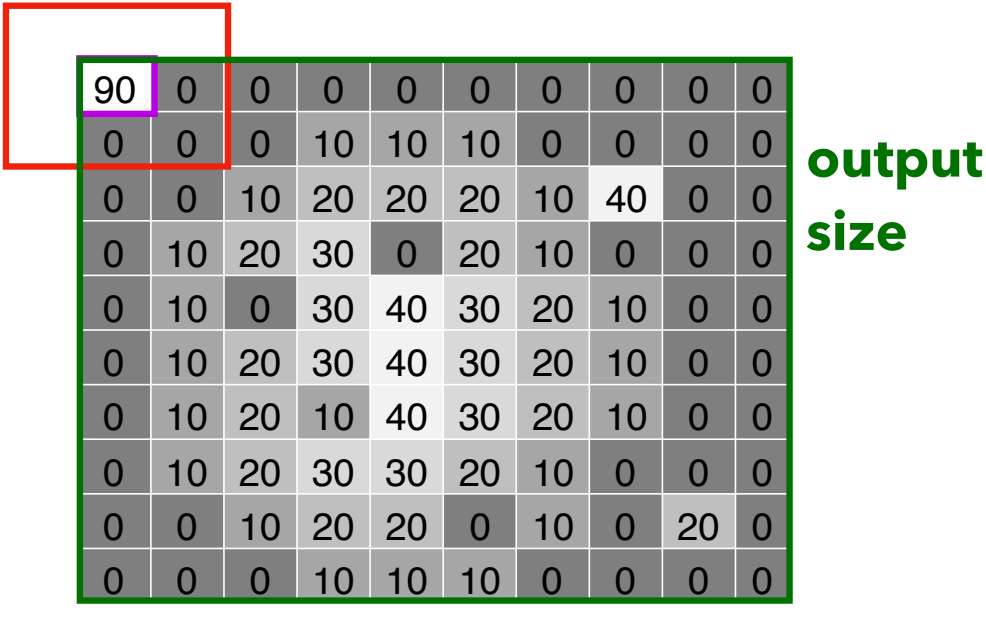


input image

output size

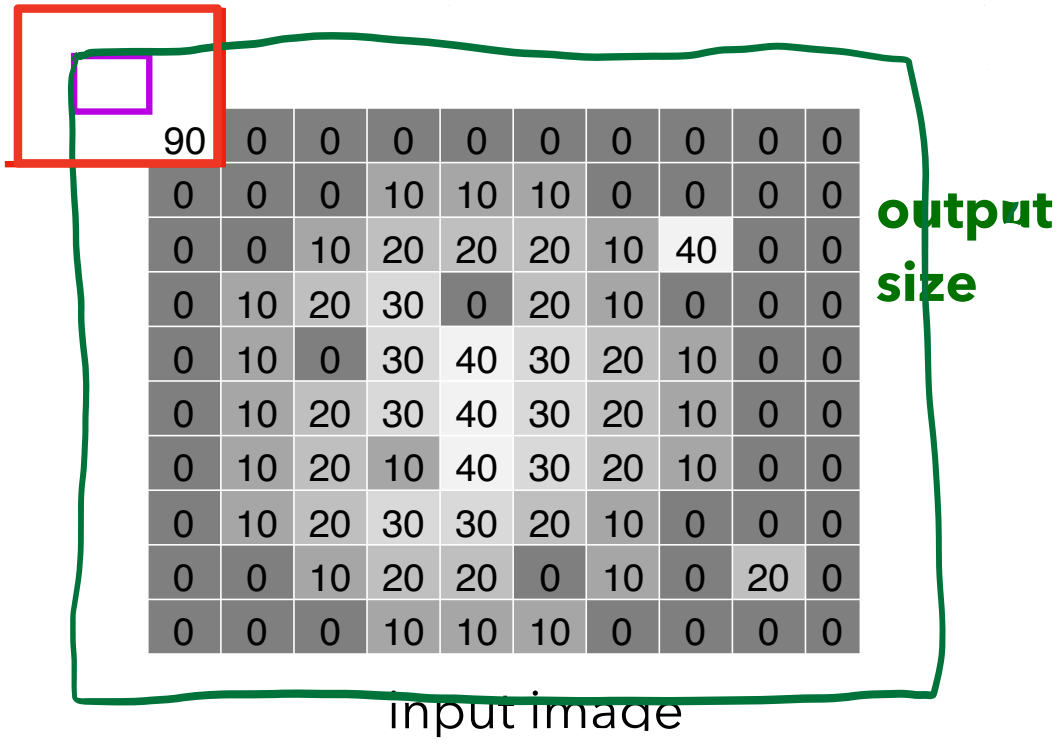
Handling Edges

"Same"

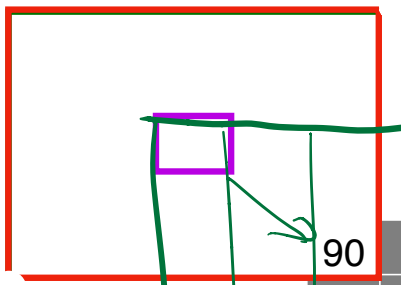


Handling Edges

"Full" (3x3)



Handling Edges



"Full" (5x5)

90	0	0	0	0	0	0	0	0	0
0	0	0	10	10	10	0	0	0	0
0	0	10	20	20	20	10	40	0	0
0	10	20	30	0	20	10	0	0	0
0	10	0	30	40	30	20	10	0	0
0	10	20	30	40	30	20	10	0	0
0	10	20	10	40	30	20	10	0	0
0	10	20	30	30	20	10	0	0	0
0	0	10	20	20	0	10	0	20	0
0	0	0	10	10	10	0	0	0	0

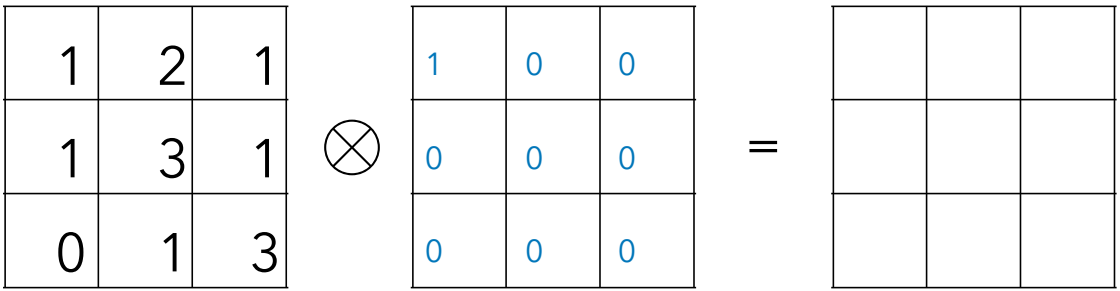
output size

input image

Let's try one.

Cross-correlate the image *f* with the kernel *w*.

Use "same" output size, with zero-padding for out-of-bounds values.



The result is a cropped copy of the input that is:

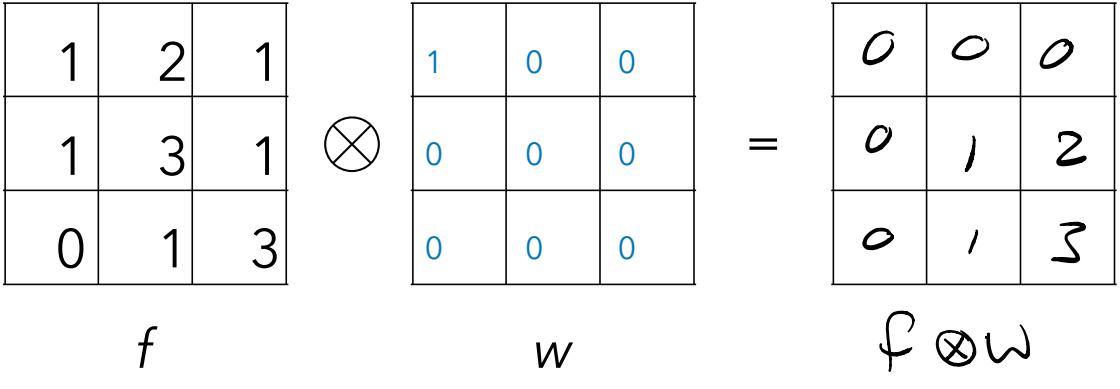
A. shifted left and up by one pixel

B. shifted down and right by one pixel

Let's try one.

Cross-correlate the image *f* with the kernel *w*.

Use "same" output size, with zero-padding for out-of-bounds values.



Cross-correlation vs Convolution

- Cross-correlation: $g = f \otimes w$

last time

$$g(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i,j) f(x+i, y+j)$$

- Convolution: $g = f * w$

$$g(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i,j) f(x-i, y-j)$$

These are related: $f \otimes w = f * \text{flip}\updownarrow(\text{flip}\leftrightarrow(w))$

(not the same as transpose)

Cross-correlation and convolution: Properties

Shift invariance (both) $f(x,y) \otimes w = [f(x-s,y) \otimes w](x-s,y)$

img: f kernels: w, v

Linearity (both) $(f \otimes aw) + (f \otimes bv) = f \otimes (aw + bv)$

Commutativity (conv only)

$$f * w = w * f$$

Associativity (conv only)

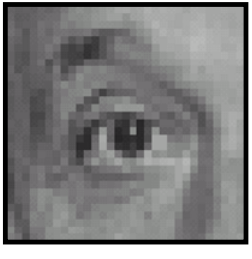
$$(f * w) * v = f * (w * v)$$

2 filters became 1! ↓

+ efficiency
+ derive new filters

What can we do with this?

What can we do with this?

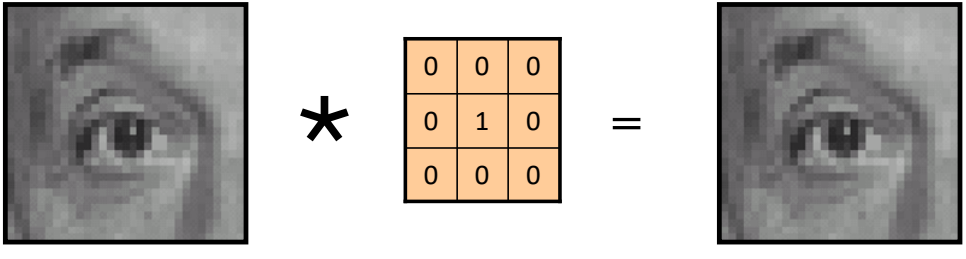


$\otimes$

0	0	0
0	1	0
0	0	0

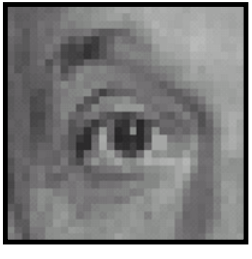
=

What can we do with this?



Identity filter: output = input

What can we do with this?



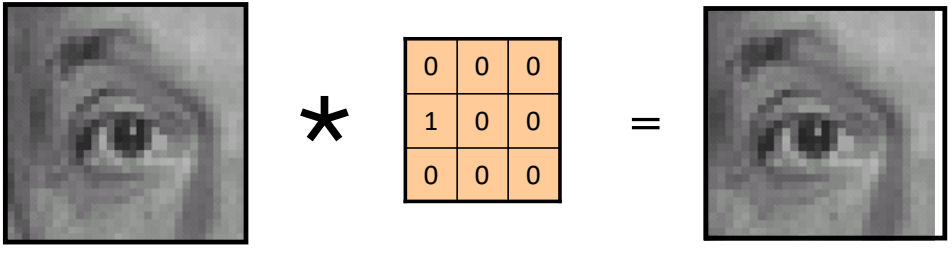
$*$

0	0	0
1	0	0
0	0	0

$=$

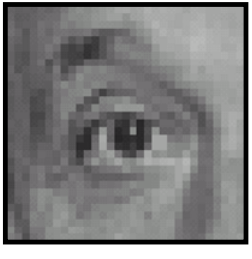
Convolution flips filter first

What can we do with this?



left shift

What can we do with this?

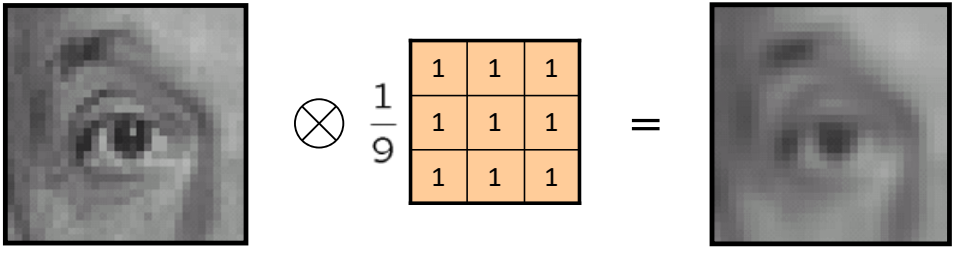


$\otimes \frac{1}{9}$

1	1	1
1	1	1
1	1	1

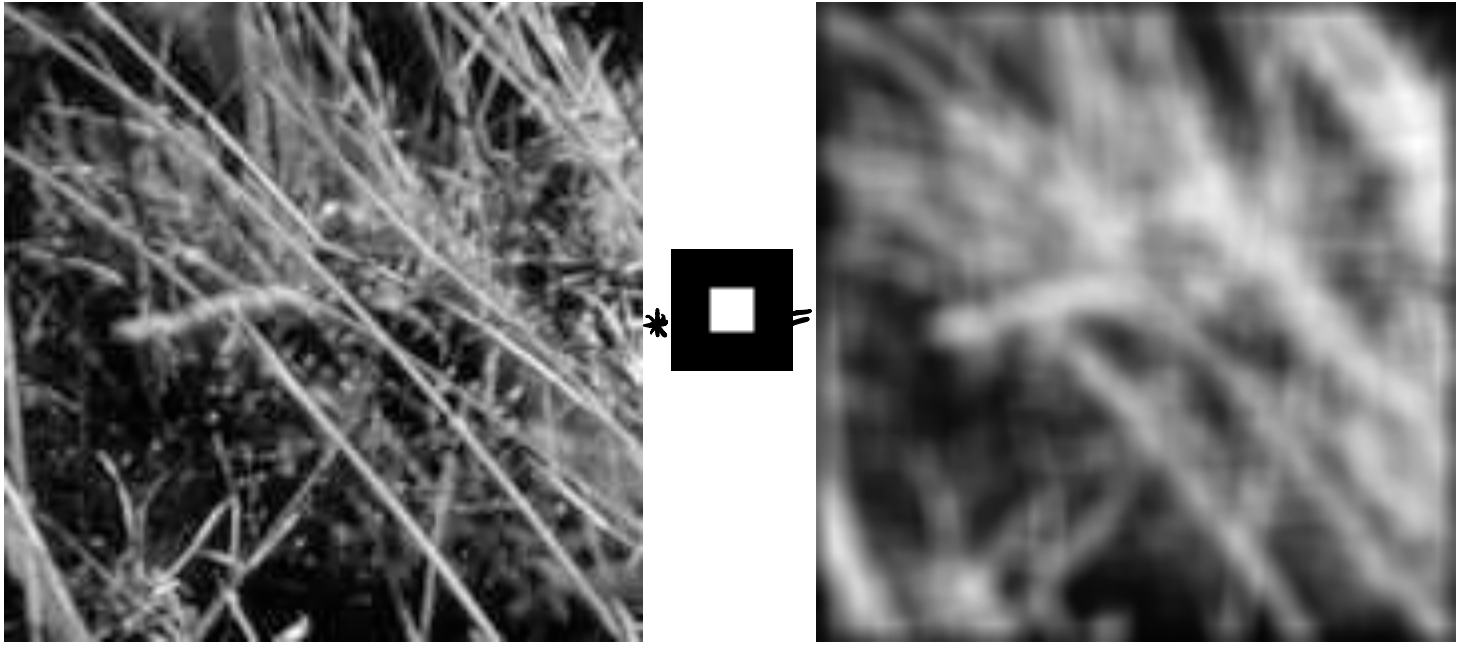
=

What can we do with this?

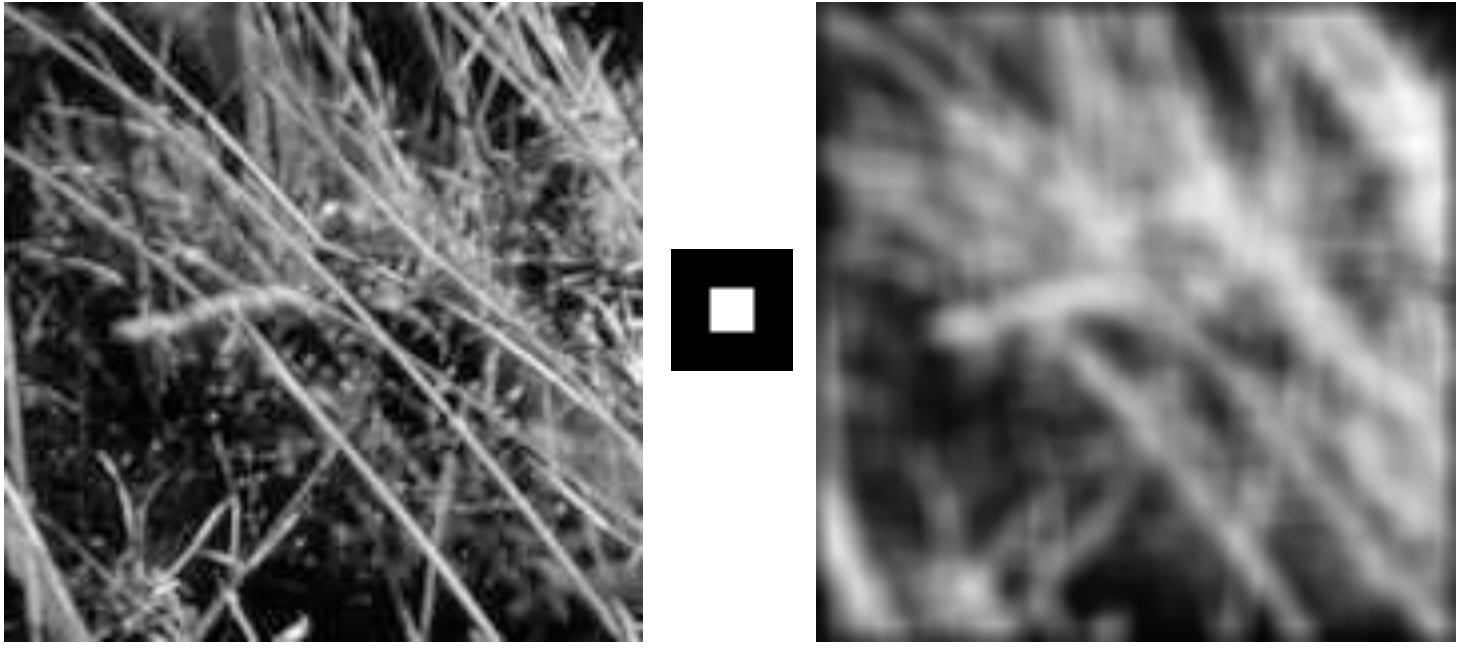


mean filter, or
box blur

Blurring: Another example

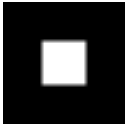


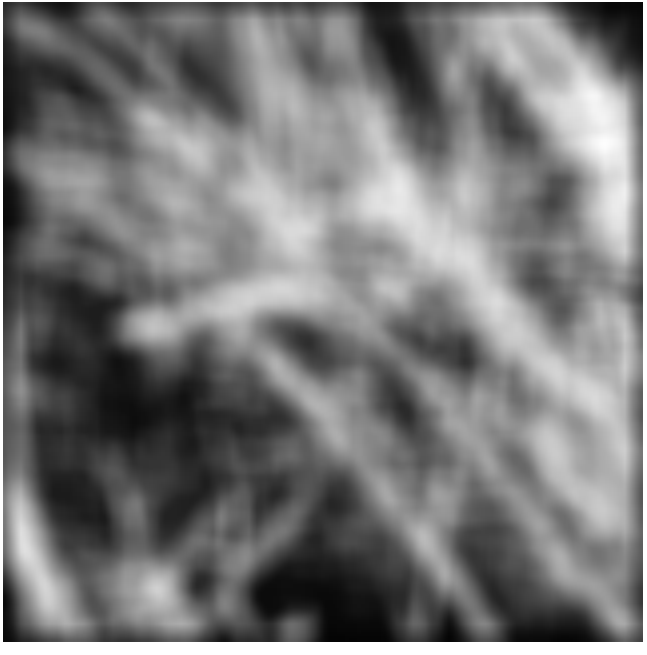
Blurring: Another example



Notice: lattice-like texture

Blurring: Another example



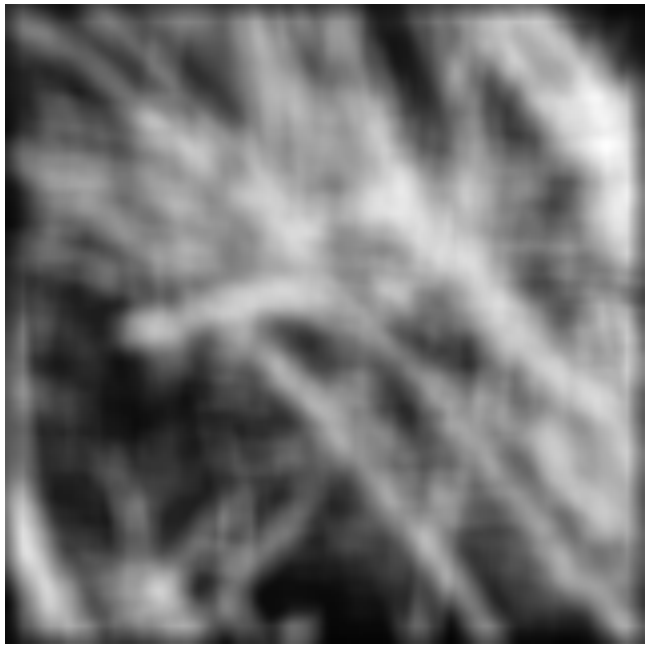


Notice: lattice-like texture

What motivated the mean filter?

nearby pixels have similar color

Blurring: Another example



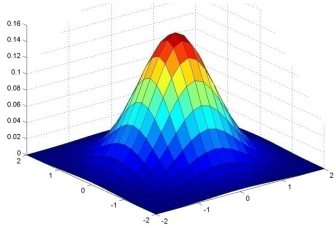
Notice: lattice-like texture

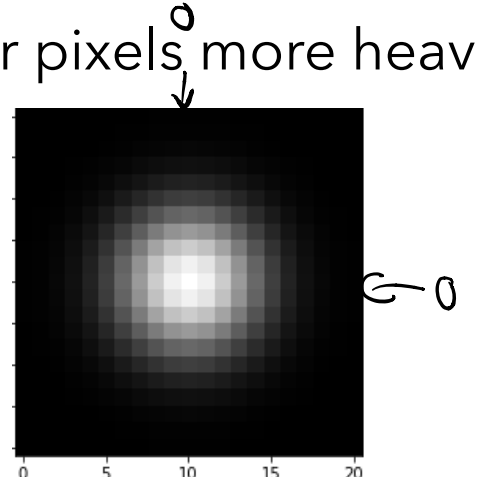
What motivated the mean filter?

Idea: the closer the pixel, the more likely it is to be similar

Gaussian Blur

- Idea: weight closer pixels more heavily using a Gaussian kernel:





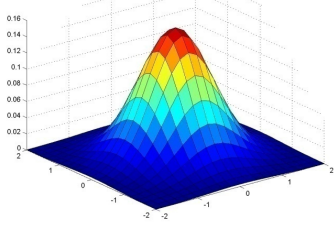
This is a bivariate (2D) Gaussian function:

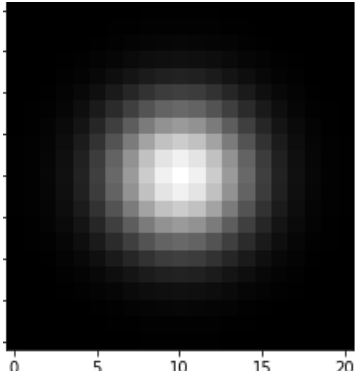
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

$\mu = (0,0)$

Gaussian Blur

- Idea: weight closer pixels more heavily using a Gaussian kernel:





$\frac{1}{16}$

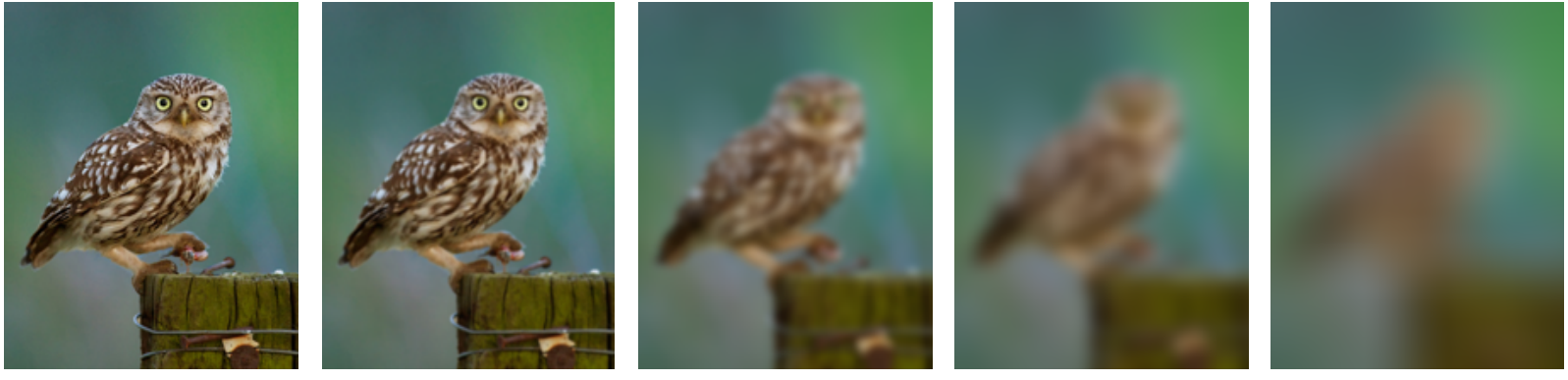
1	2	1
2	4	2
1	2	1

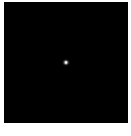
3x3 approximation

This is a bivariate (2D) Gaussian function:

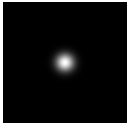
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Gaussian Filters



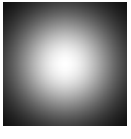


σ = 1 pixel



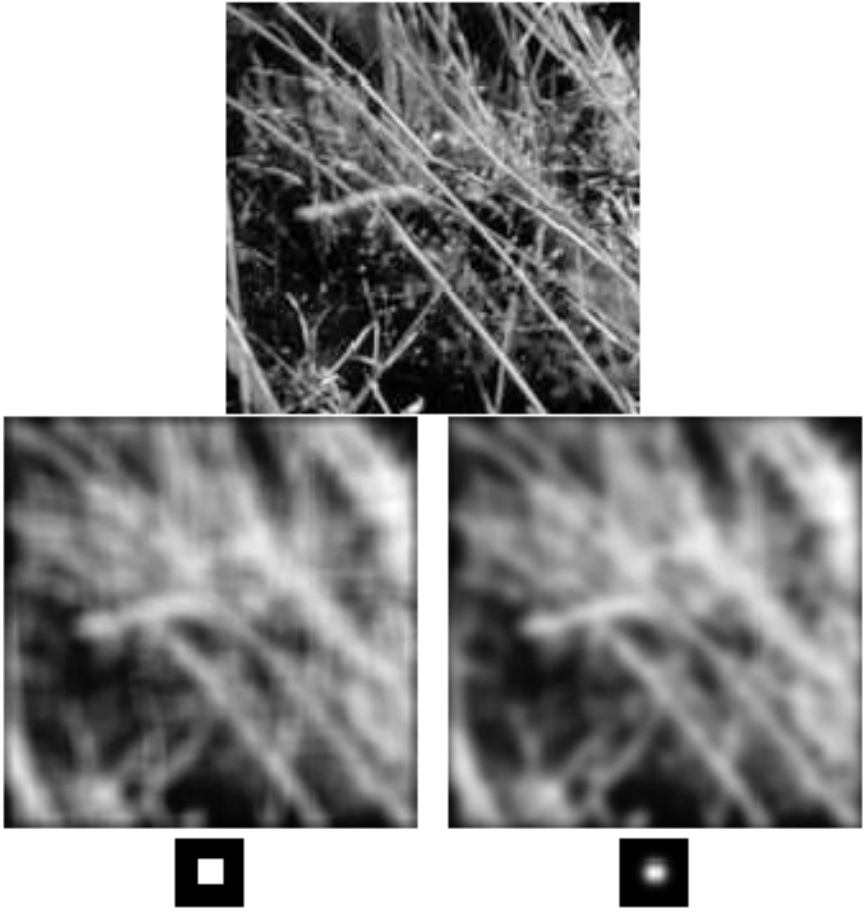
σ = 5 pixels

σ = 10 pixels



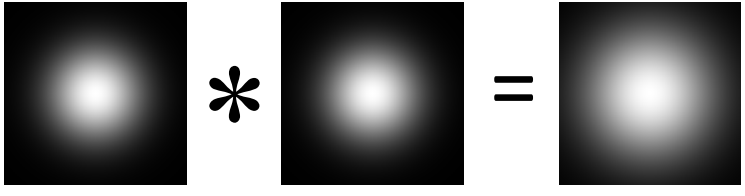
σ = 30 pixels

Mean vs. Gaussian



Composing Filters

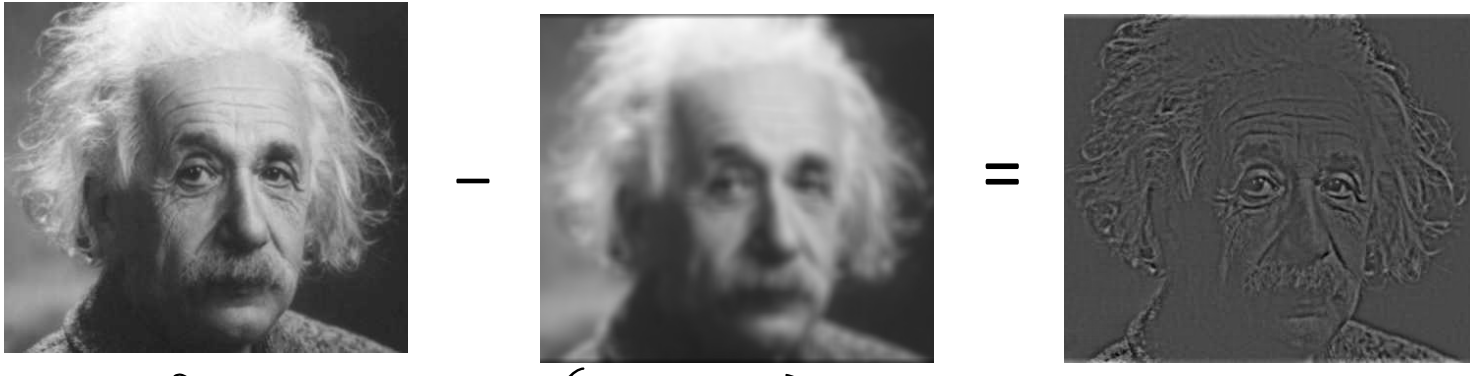
- Recall associativity:



$$G_\sigma * G_\sigma = G_{\sqrt{2}\sigma}$$

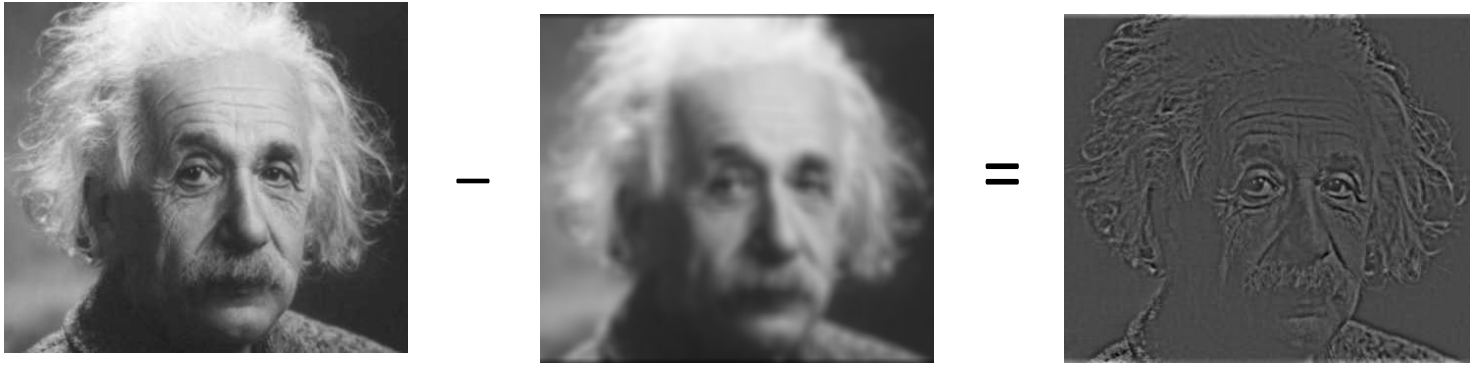
Sharpening!?

- What gets removed when we blur?



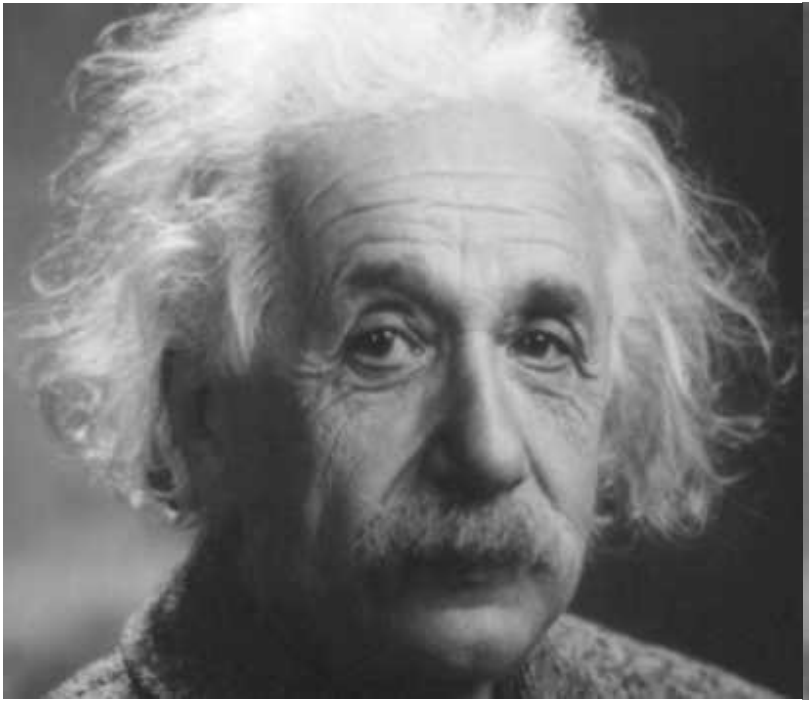
Sharpening!?

- What gets removed when we blur?

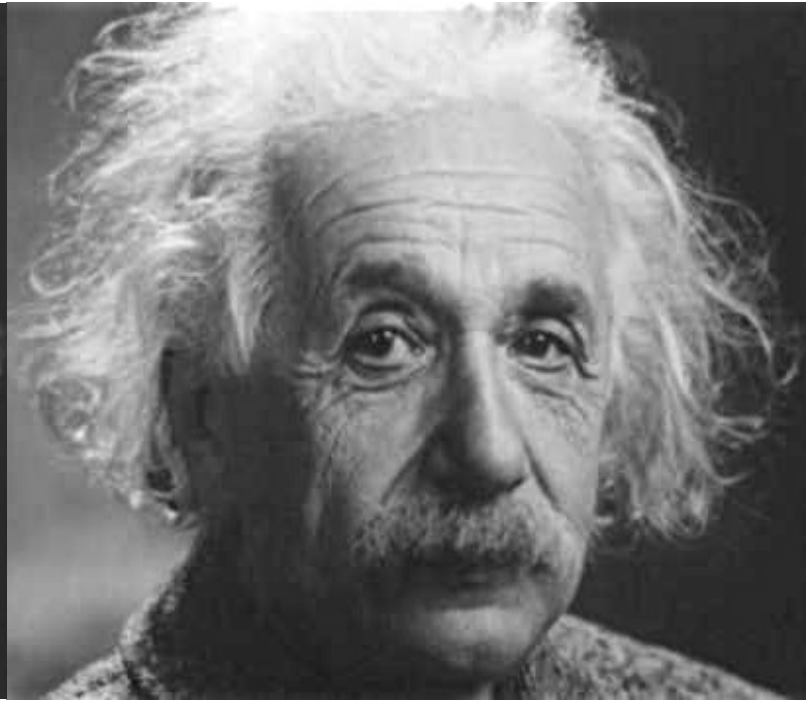


Sharpening

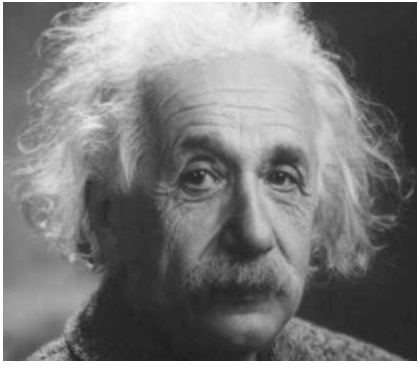
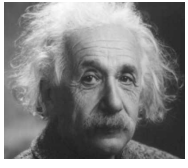
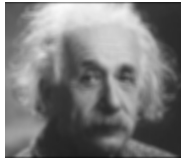
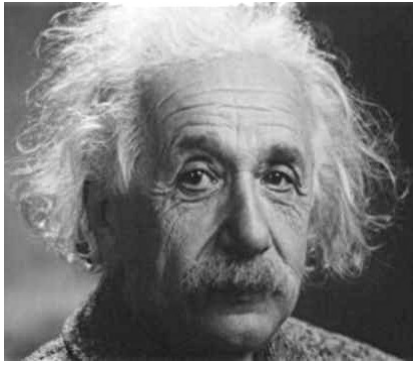
Before



After



Sharpening: once more with mathing.

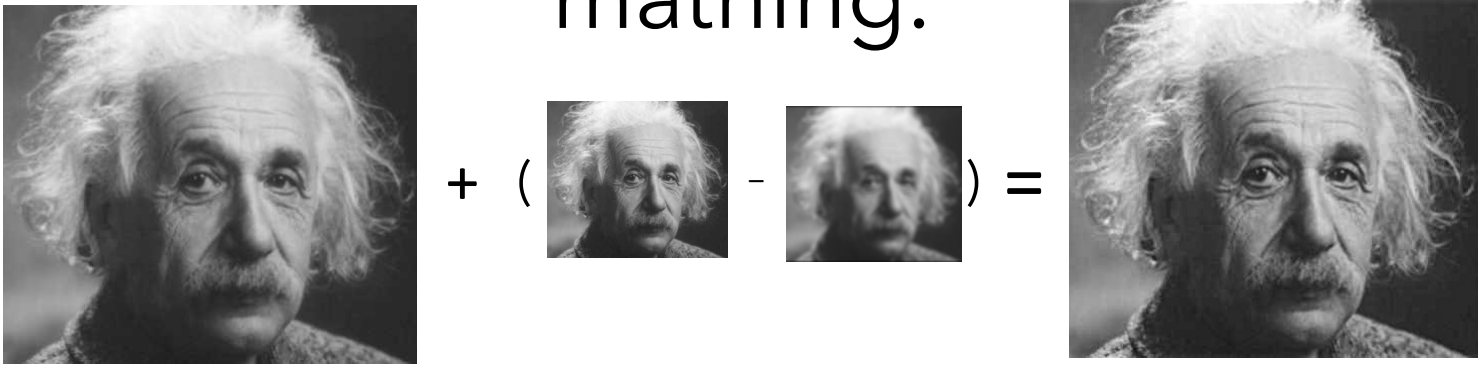
$$f + (f - [f * B])$$

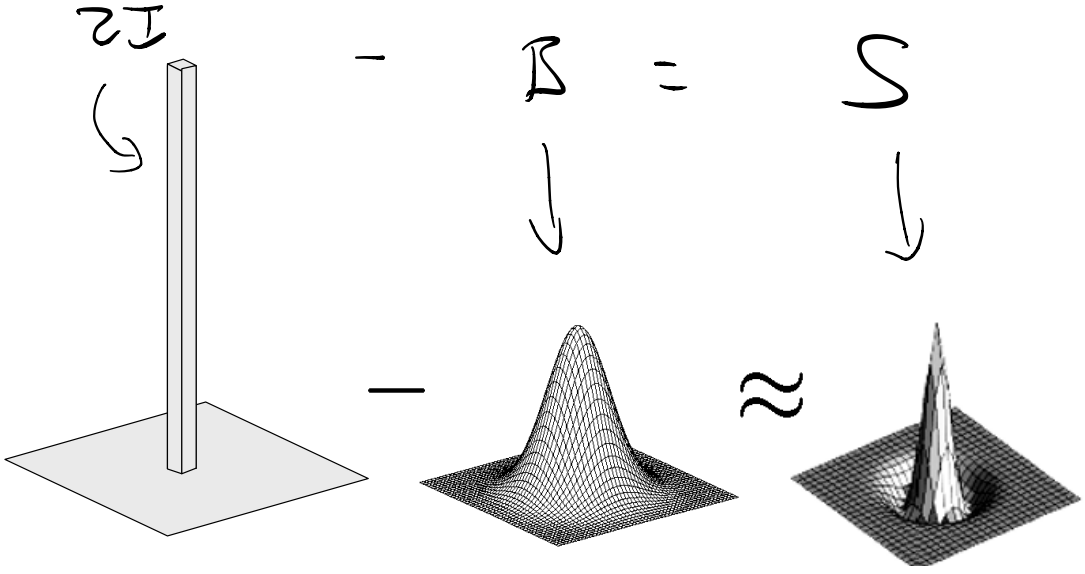
$$= 2f - [f * B]$$

$$= (f * 2I) - (f * B)$$

$$f * (2I - B)$$

Sharpening: once more with mathing.





Effects of Sharpening

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