

Filtering

Let's Take An Image



Let's Fix Things

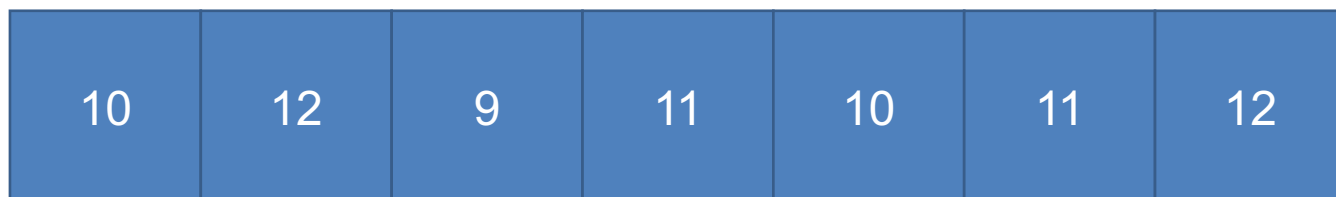
- We have noise in our image
- Let's replace each pixel with a *weighted* average of its neighborhood
- Weights are *filter kernel*

	Out	

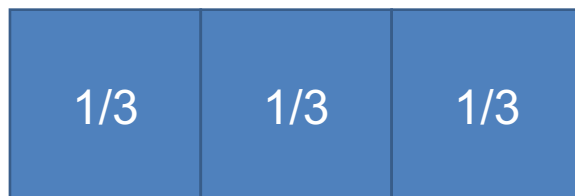
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

1D Case

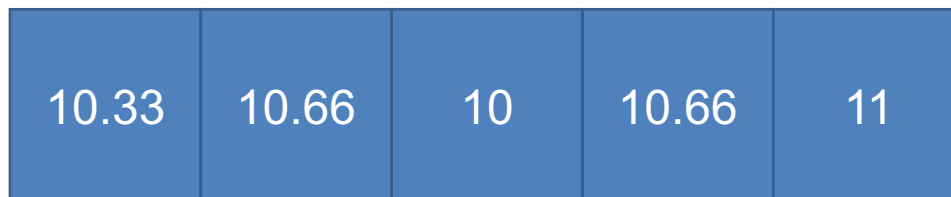
Signal



Filter



Output



Applying a Linear Filter

Input

I11	I12	I13	I14	I15	I16
I21	I22	I23	I24	I25	I26
I31	I32	I33	I34	I35	I36
I41	I42	I43	I44	I45	I46
I51	I52	I53	I54	I55	I56

Filter

F11	F12	F13
F21	F22	F23
F31	F32	F33

Output

O11	O12	O13	O14
O21	O22	O23	O24
O31	O32	O33	O34

Applying a Linear Filter

Input & Filter

F11	F12	F13	I14	I15	I16
F21	F22	F23	I24	I25	I26
F31	F32	F33	I34	I35	I36
I41	I42	I43	I44	I45	I46
I51	I52	I53	I54	I55	I56

Output

O11

$$O_{11} = I_{11} * F_{11} + I_{12} * F_{12} + \dots + I_{33} * F_{33}$$

Applying a Linear Filter

Input & Filter

I11	F11	F12	F13	I15	I16
I21	F21	F22	F23	I25	I26
I31	F31	F32	F33	I35	I36
I41	I42	I43	I44	I45	I46
I51	I52	I53	I54	I55	I56

Output

O11	O12
-----	-----

$$O12 = I12 * F11 + I13 * F12 + \dots + I34 * F33$$

Applying a Linear Filter

Input

I11	I12	I13	I14	I15	I16
I21	I22	I23	I24	I25	I26
I31	I32	I33	I34	I35	I36
I41	I42	I43	I44	I45	I46
I51	I52	I53	I54	I55	I56

Filter

F11	F12	F13
F21	F22	F23
F31	F32	F33

Output

**How many times can we apply a
3x3 filter to a 5x6 image?**

Applying a Linear Filter

Input

I11	I12	I13	I14	I15	I16
I21	I22	I23	I24	I25	I26
I31	I32	I33	I34	I35	I36
I41	I42	I43	I44	I45	I46
I51	I52	I53	I54	I55	I56

Filter

F11	F12	F13
F21	F22	F23
F31	F32	F33

Output

O11	O12	O13	O14
O21	O22	O23	O24
O31	O32	O33	O34

$$O_{ij} = I_{ij} * F_{11} + I_{i(j+1)} * F_{12} + \dots + I_{(i+2)(j+2)} * F_{33}$$

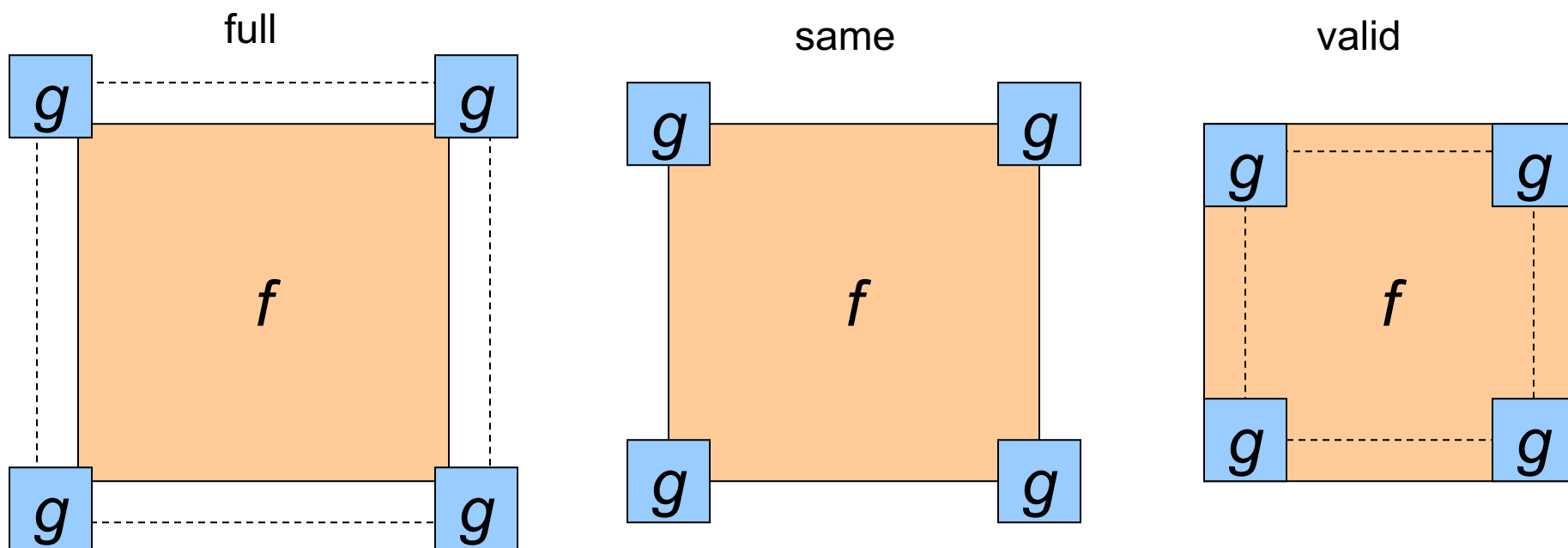
Painful Details – Edge Cases

Convolution doesn't keep the whole image.

Suppose f is the image and g the filter.

Full – any part of g touches f . **Same** – same size as f ;

Valid – only when filter doesn't fall off edge.

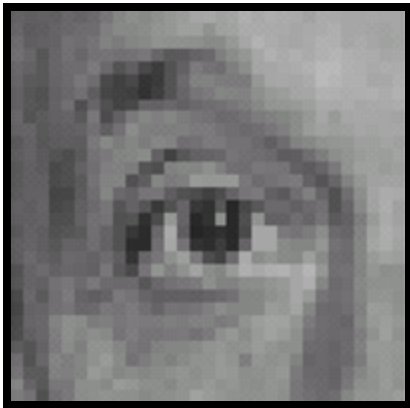


Dealing with edges

- If the filter window falls off the edge of the image, we need to pad the image
 - Zero pad (or clip filter)
 - Wrap around
 - Copy edge
 - Reflect across edge



Practice with Linear Filters

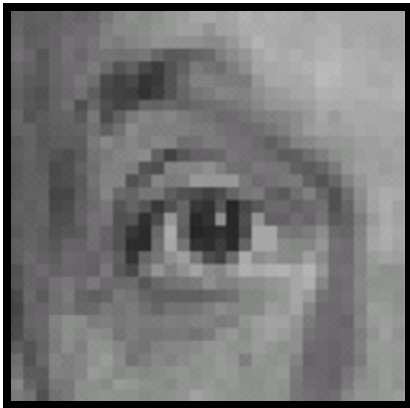


Original

0	0	0
0	1	0
0	0	0

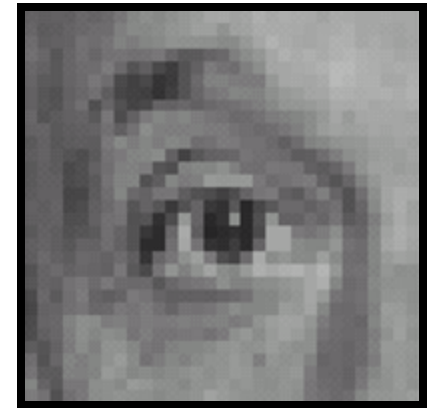
?

Practice with Linear Filters



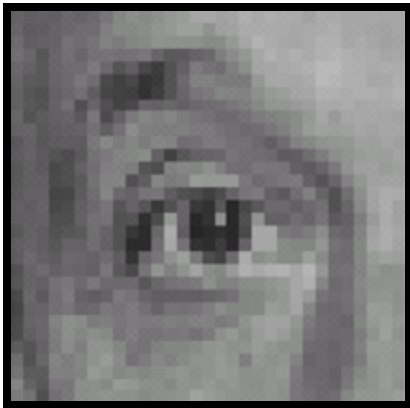
Original

0	0	0
0	1	0
0	0	0



The Same!

Practice with Linear Filters

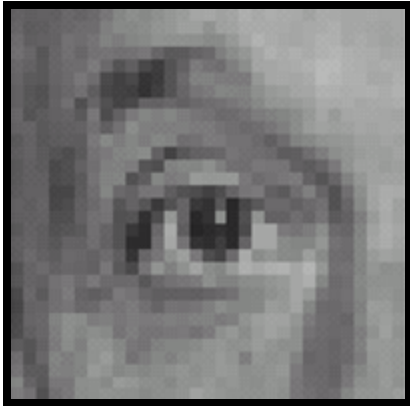


Original

0	0	0
0	0	1
0	0	0

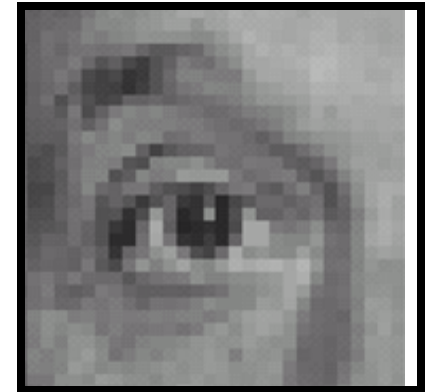
?

Practice with Linear Filters



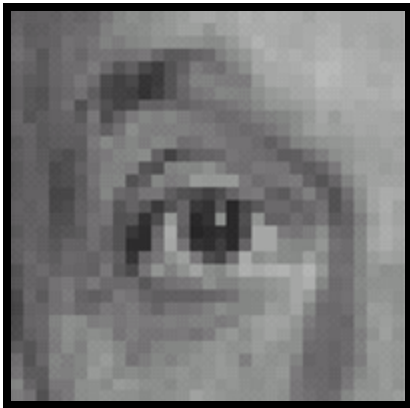
Original

0	0	0
0	0	1
0	0	0



Shifted
LEFT
1 pixel

Practice with Linear Filters

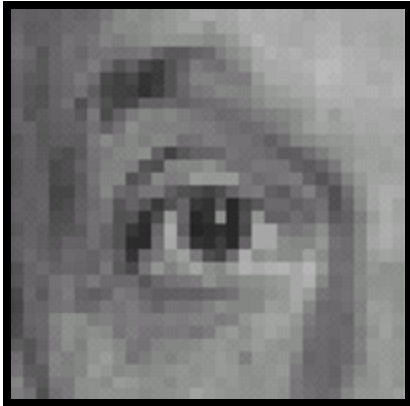


Original

0	1	0
0	0	0
0	0	0

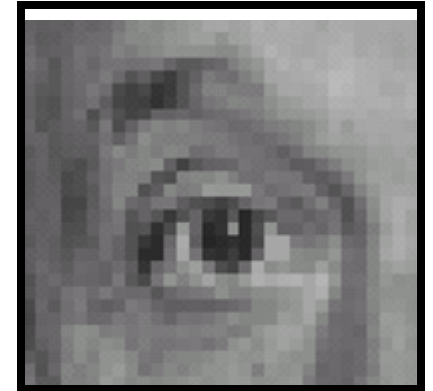
?

Practice with Linear Filters



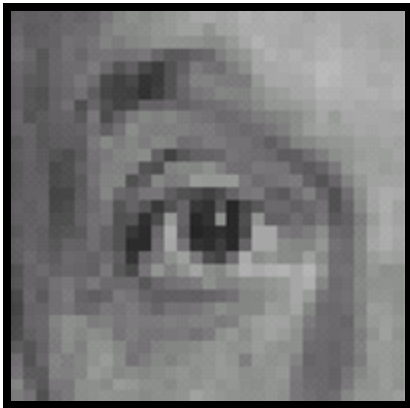
Original

0	1	0
0	0	0
0	0	0



Shifted
DOWN
1 pixel

Practice with Linear Filters

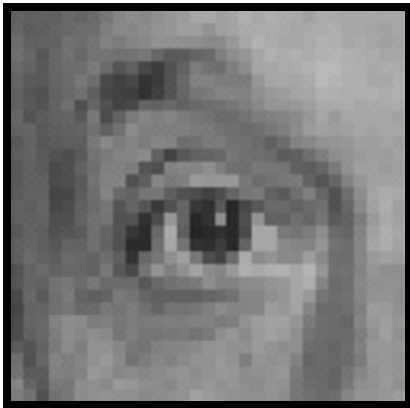


Original

$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$

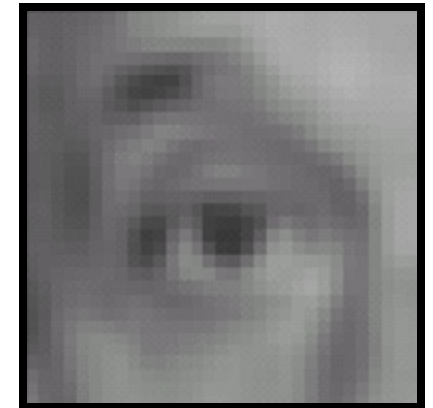
?

Practice with Linear Filters



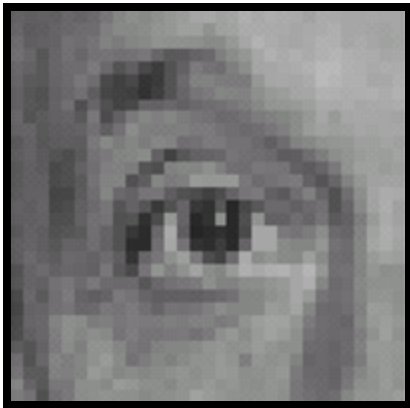
Original

$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$



Blur
(Box Filter)

Practice with Linear Filters



Original

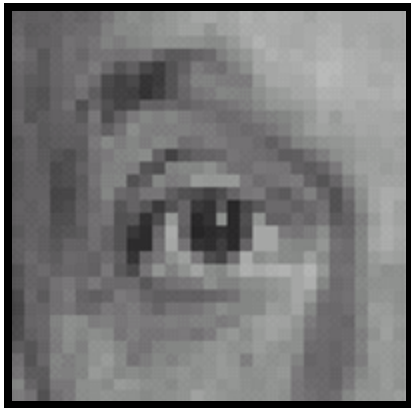
0	0	0
0	2	0
0	0	0

-

$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$

?

Practice with Linear Filters



Original

0	0	0
0	2	0
0	0	0

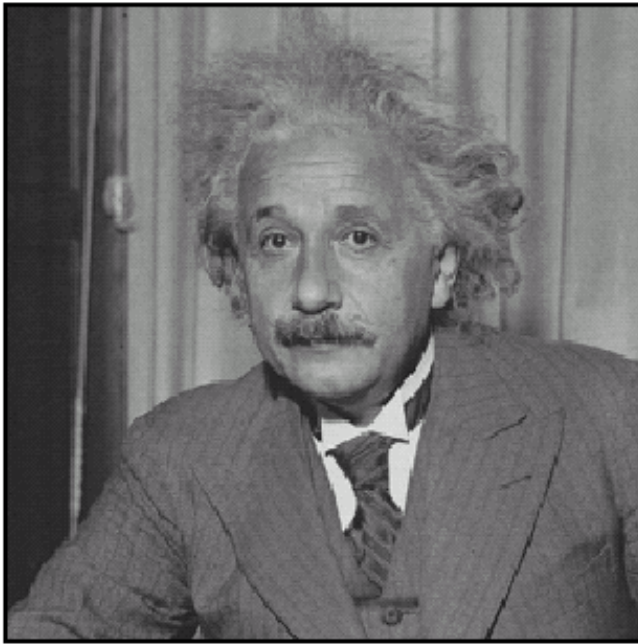
-

$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$

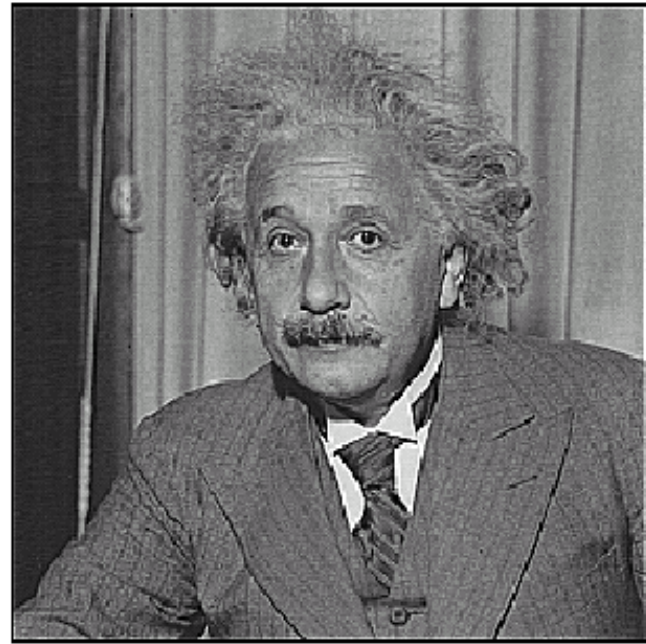


Sharpened
(Accentuates
difference from
local average)

Sharpening



before



after

Properties – Linear

Assume: I image f1, f2 filters

Linear: $\text{apply}(I, f1+f2) = \text{apply}(I, f1) + \text{apply}(I, f2)$

I is a white box on black, and f1, f2 are rectangles

$$A\left(\begin{array}{|c|} \hline \blacksquare \\ \hline \end{array}, \begin{array}{|c|} \hline \blacksquare \\ \hline \end{array} + \begin{array}{|c|} \hline \blacksquare \\ \hline \end{array}\right) = A\left(\begin{array}{|c|} \hline \blacksquare \\ \hline \end{array}, \begin{array}{|c|} \hline \square \\ \hline \end{array}\right) = \begin{array}{|c|} \hline \blacksquare \\ \hline \end{array}$$

$$A\left(\begin{array}{|c|} \hline \blacksquare \\ \hline \end{array}, \begin{array}{|c|} \hline \blacksquare \\ \hline \end{array}\right) + A\left(\begin{array}{|c|} \hline \blacksquare \\ \hline \end{array}, \begin{array}{|c|} \hline \blacksquare \\ \hline \end{array}\right) = \begin{array}{|c|} \hline \blacksquare \\ \hline \end{array} + \begin{array}{|c|} \hline \blacksquare \\ \hline \end{array} = \begin{array}{|c|} \hline \blacksquare \\ \hline \end{array}$$

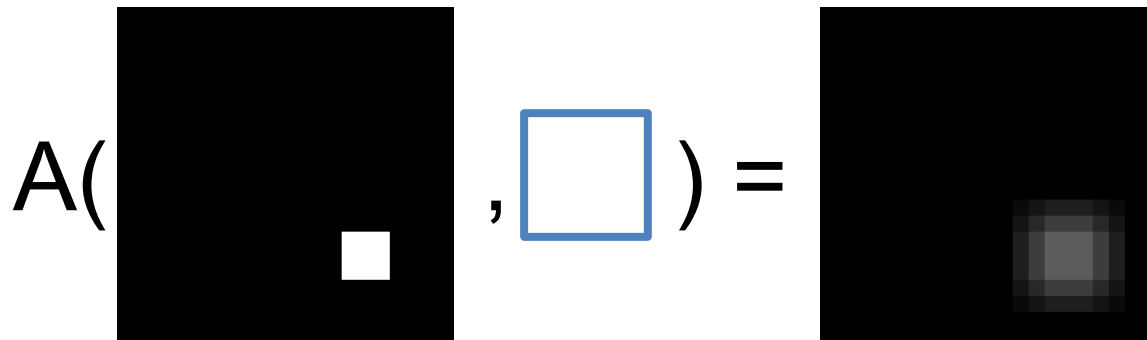
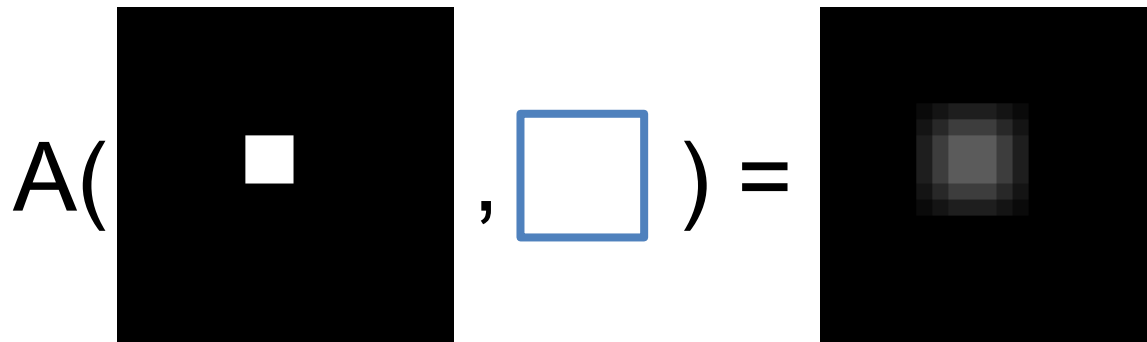
Note: I am showing filters un-normalized and blown up. They're a smaller box filter (i.e., each entry is $1/(\text{size}^2)$)

Properties – Shift-Invariant

Assume: I image, f filter

Shift-invariant: $\text{shift}(\text{apply}(I,f)) = \text{apply}(\text{shift}(I),f)$

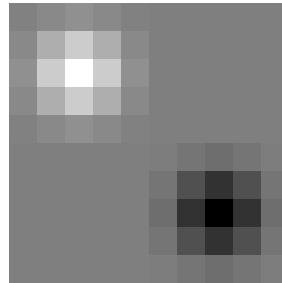
Intuitively: only depends on filter neighborhood



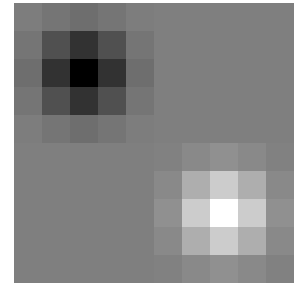
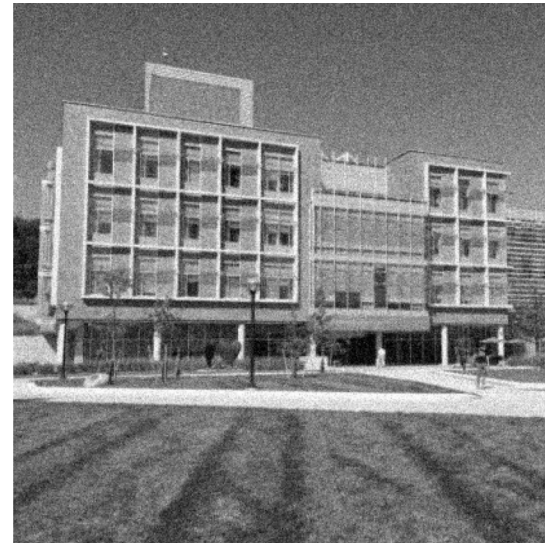
Painful Details – Signal Processing

Often called “convolution”. *Actually* cross-correlation.

Cross-Correlation
(Original Orientation)

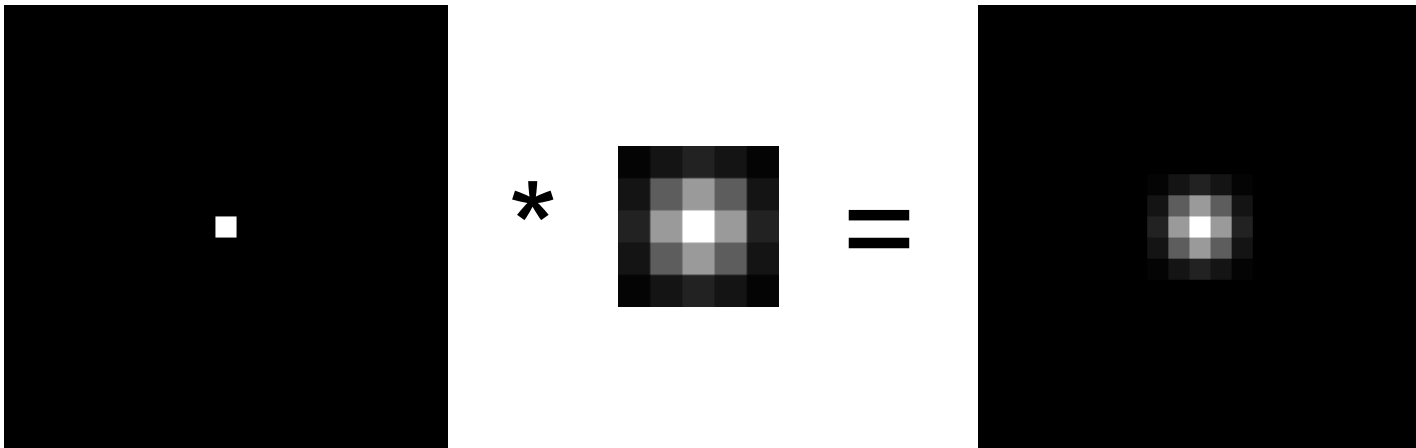


Convolution
(Flipped in x and y)



Properties of Convolution

- Any shift-invariant, linear operation is a convolution ($*$)
- Commutative: $f * g = g * f$
- Associative: $(f * g) * h = f * (g * h)$
- Distributes over $+$: $f * (g + h) = f * g + f * h$
- Scalars factor out: $kf * g = f * kg = k(f * g)$
- Identity (a single one with all zeros):



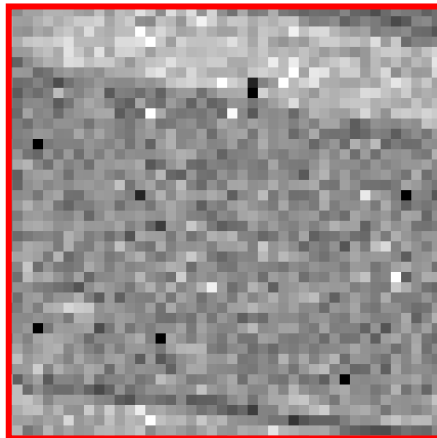
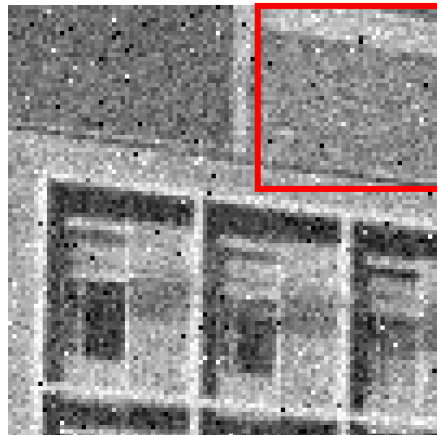
Questions?

- Nearly everything onwards is a convolution.
- This is important to get right.

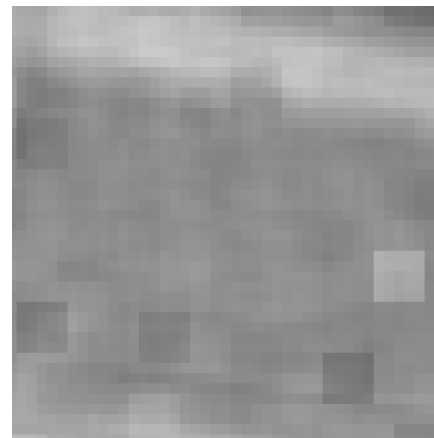
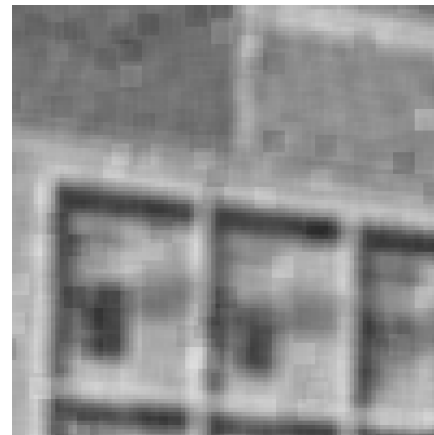
Smoothing With A Box

Intuition: if filter touches it, it gets a contribution.

Input



Box Filter



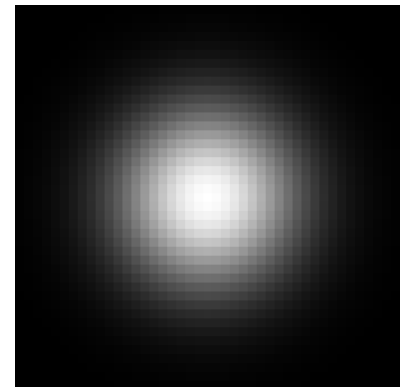
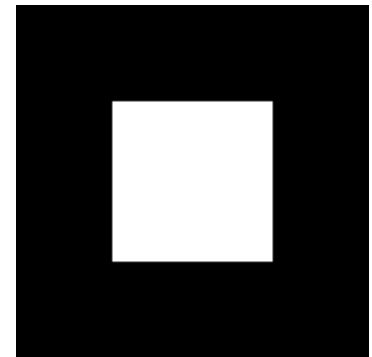
Solution – Weighted Combination

Intuition: weight contributions according to closeness to center.

$$Filter_{ij} \propto 1$$

What's this?

$$Filter_{ij} \propto \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

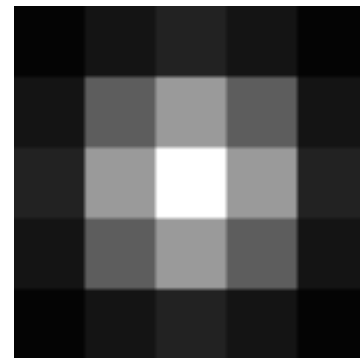


Recognize the Filter?

It's a Gaussian!

$$Filter_{ij} \propto \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

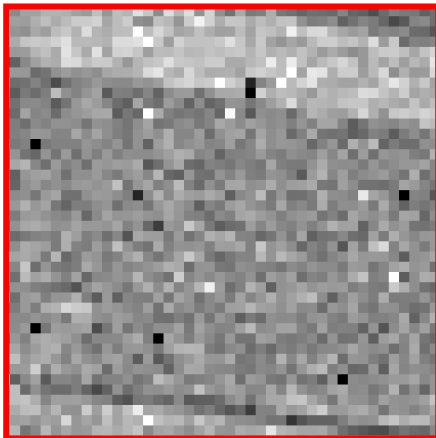
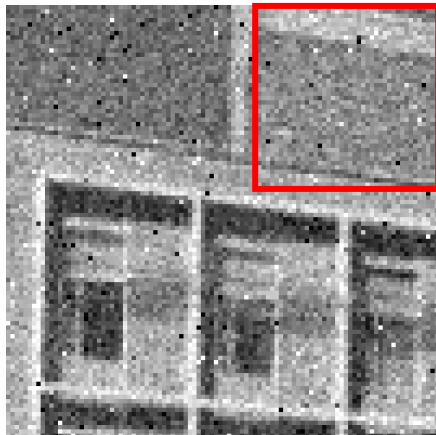
0.003	0.013	0.022	0.013	0.003
0.013	0.060	0.098	0.060	0.013
0.022	0.098	0.162	0.098	0.022
0.013	0.060	0.098	0.060	0.013
0.003	0.013	0.022	0.013	0.003



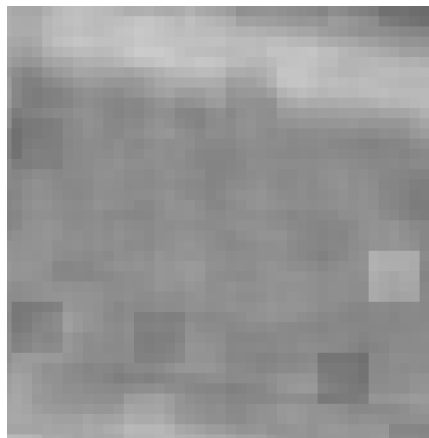
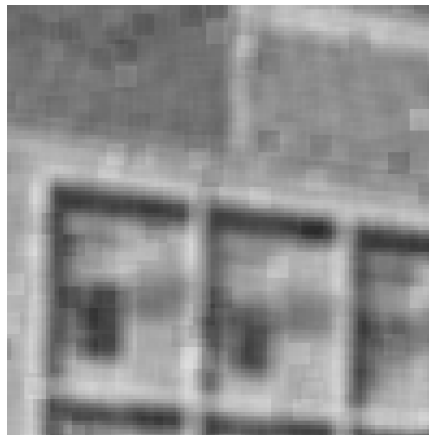
Smoothing With A Box & Gauss

Still have some speckles, but it's not a big box

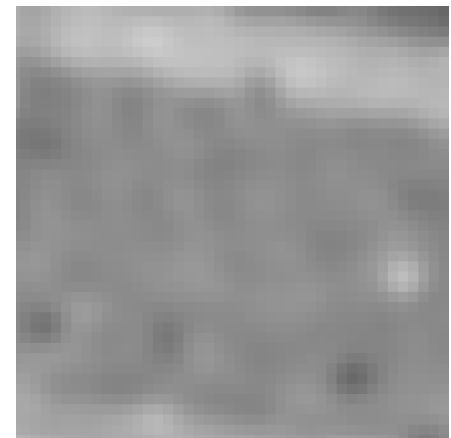
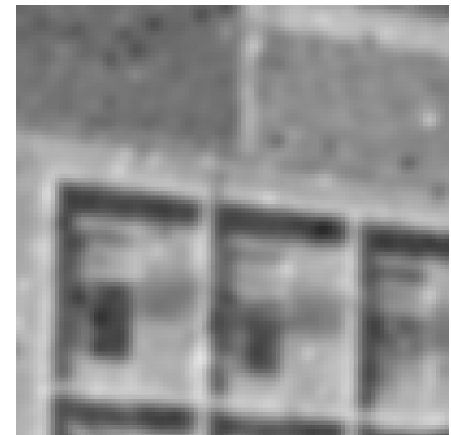
Input



Box Filter

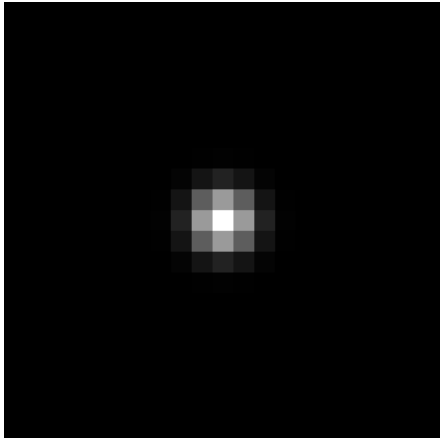


Gauss. Filter

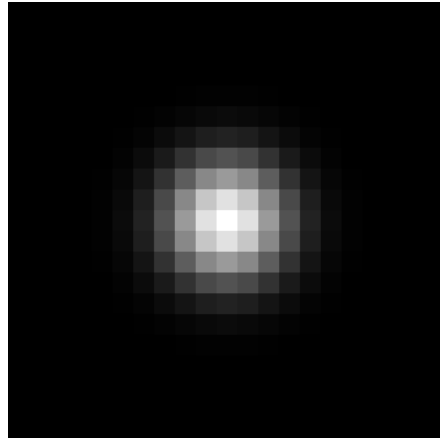


Gaussian Filters

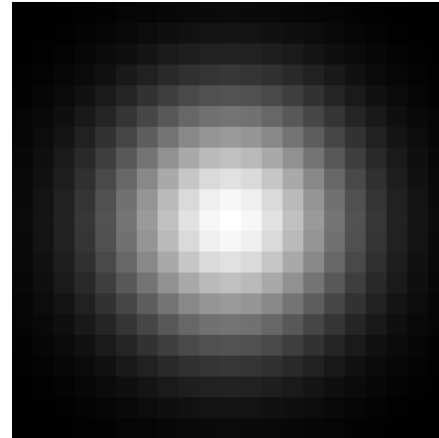
$\sigma = 1$
filter = 21x21



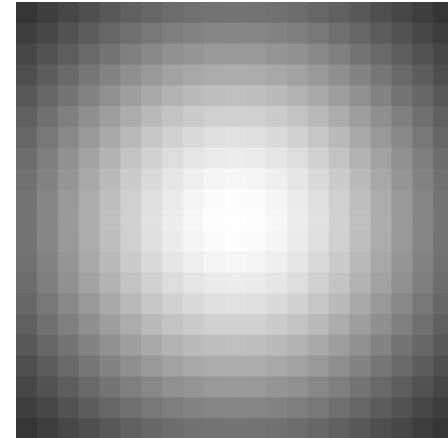
$\sigma = 2$
filter = 21x21



$\sigma = 4$
filter = 21x21

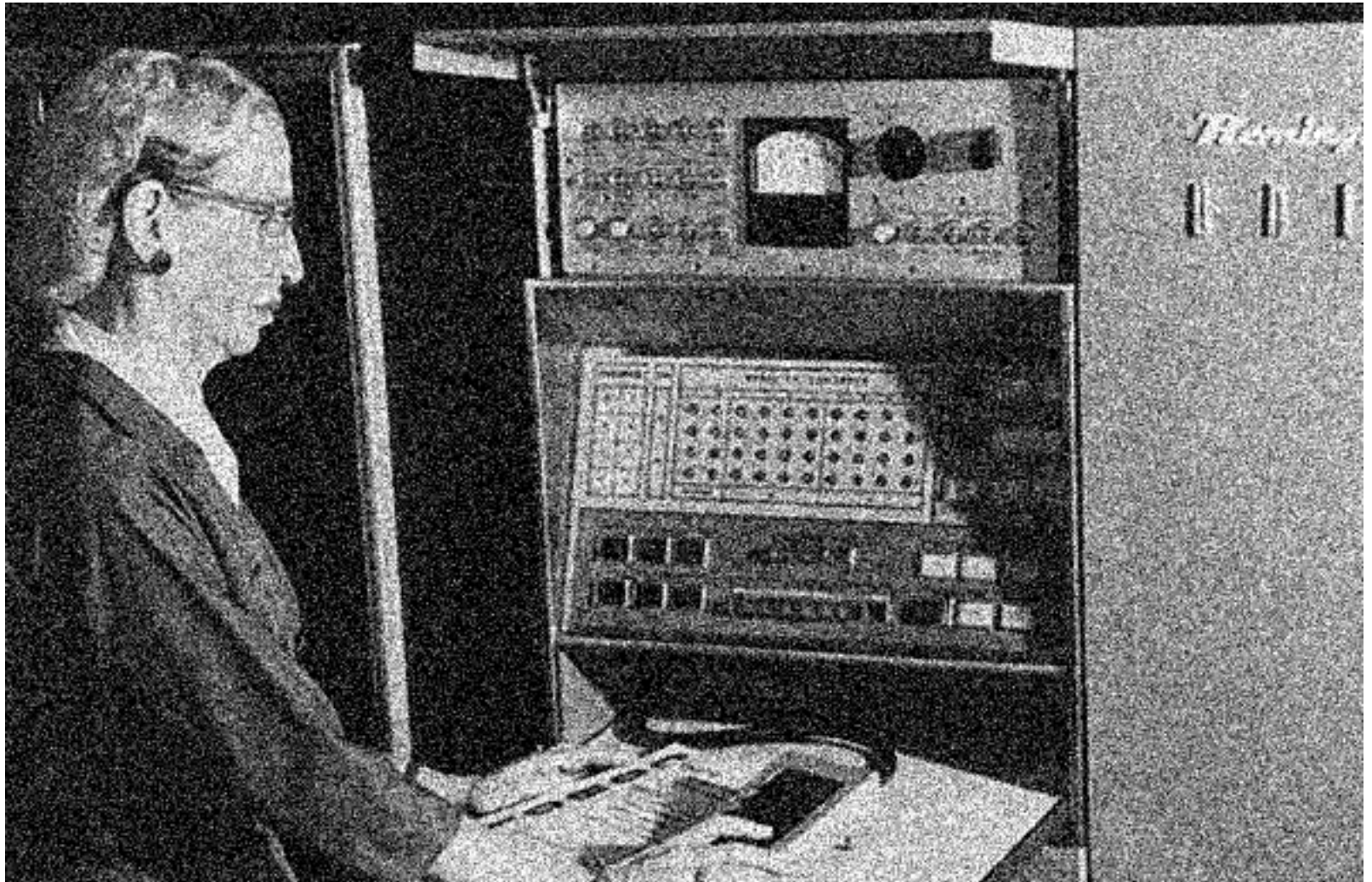


$\sigma = 8$
filter = 21x21



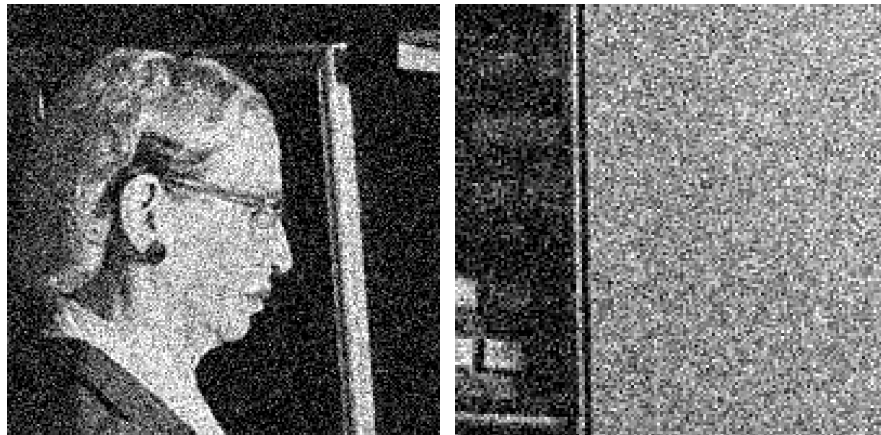
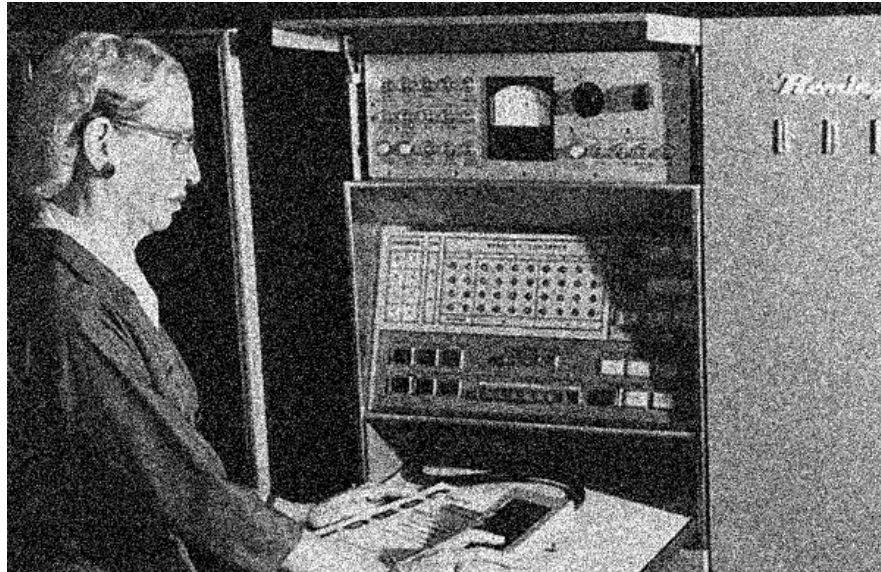
Note: filter visualizations are independently normalized throughout the slides so you can see them better

Applying Gaussian Filters



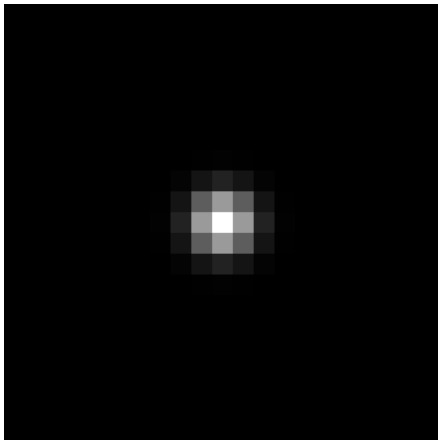
Applying Gaussian Filters

Input Image
(no filter)



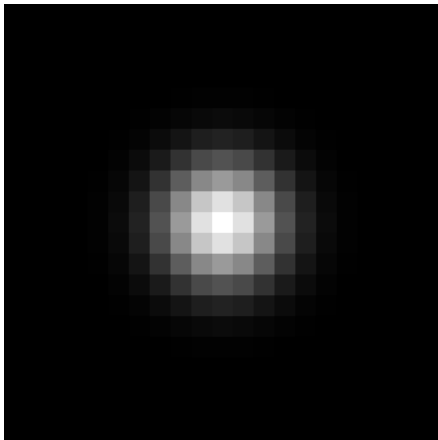
Applying Gaussian Filters

$$\sigma = 1$$



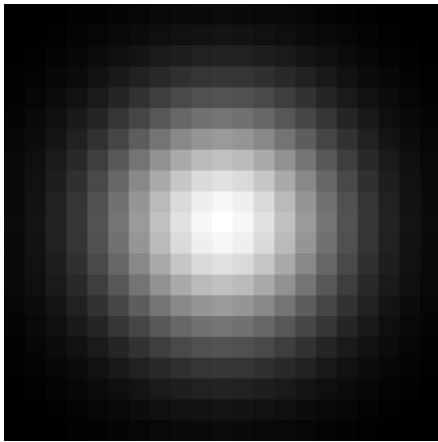
Applying Gaussian Filters

$$\sigma = 2$$



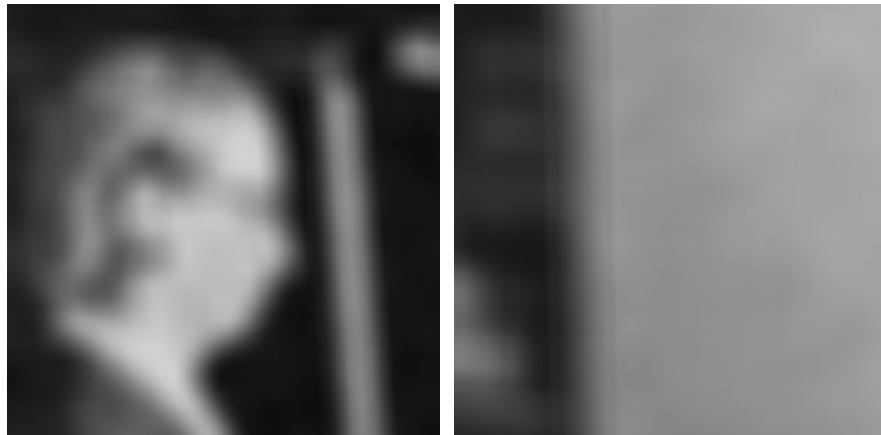
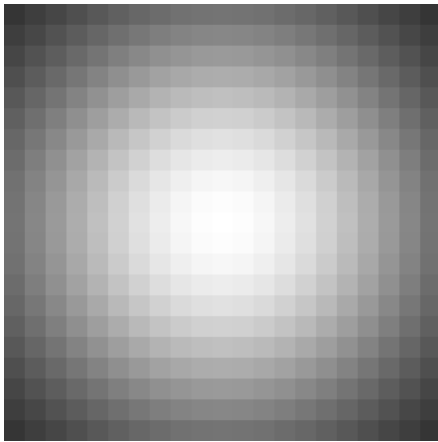
Applying Gaussian Filters

$$\sigma = 4$$



Applying Gaussian Filters

$\sigma = 8$



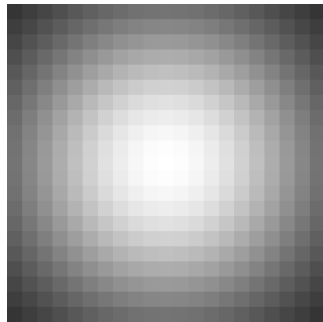
Picking a Filter Size

Too small filter \rightarrow bad approximation

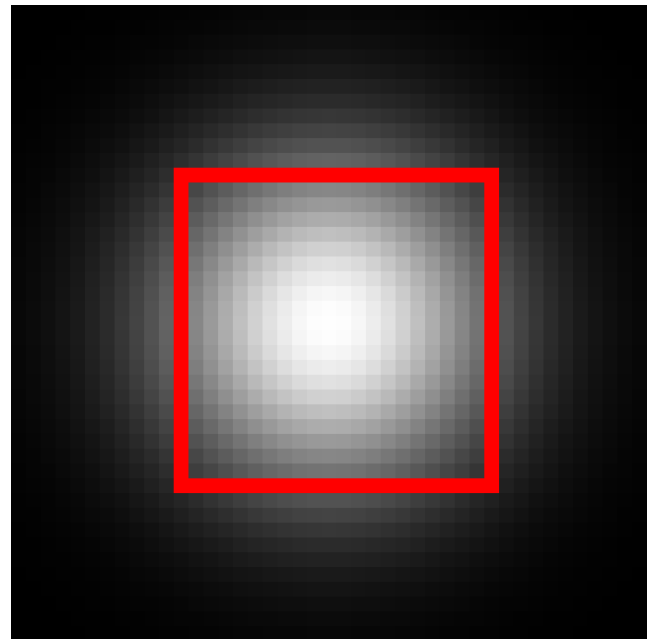
Want size $\approx 6\sigma$ (99.7% of energy)

Left far too small; right slightly too small!

$\sigma = 8$, size = 21



$\sigma = 8$, size = 43



Runtime Complexity

Image size = $N \times N = 6 \times 6$

Filter size = $M \times M = 3 \times 3$

I11	I12	I13	I14	I15	I16
I21	F11	F12	F13	I25	I26
I31	F21	F22	F23	I35	I36
I41	F31	F32	F33	I45	I46
I51	I52	I53	I54	I55	I56
I61	I62	I63	I64	I65	I66

for ImageY in range(N):

 for ImageX in range(N):

 for FilterY in range(M):

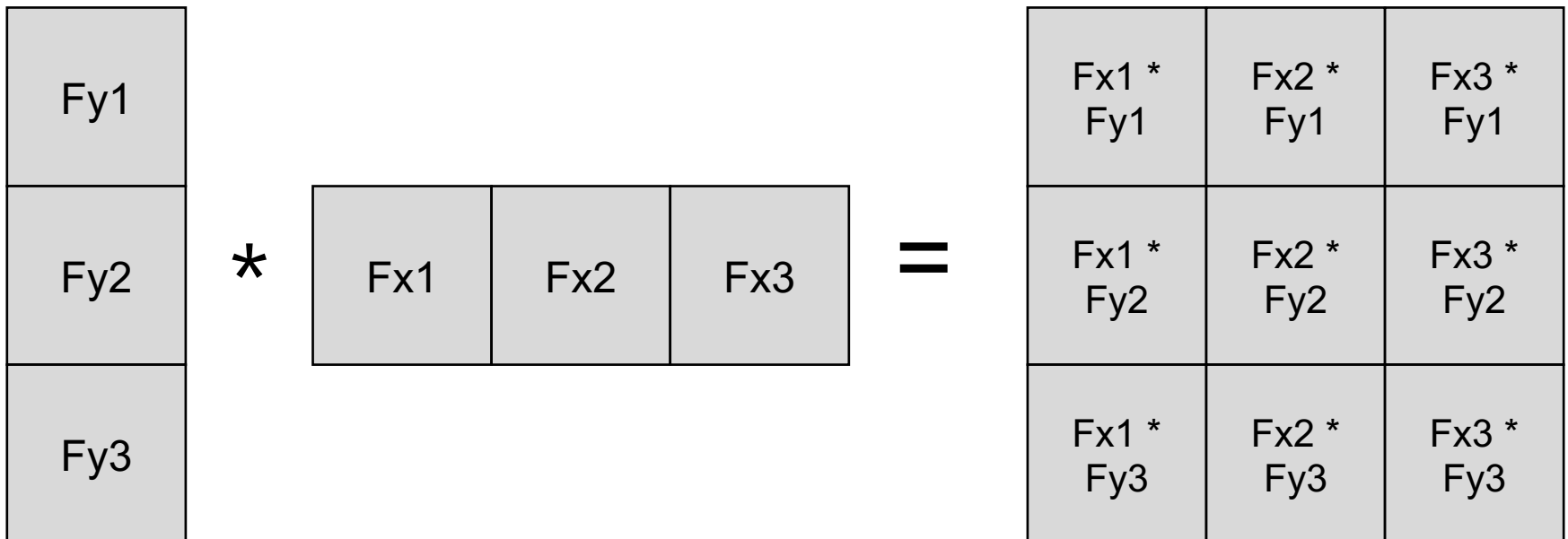
 for FilterX in range(M):

 ...

Time: $O(N^2M^2)$

Separability

Conv(vector, transposed vector) \rightarrow outer product



Separability

$$Filter_{ij} \propto \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

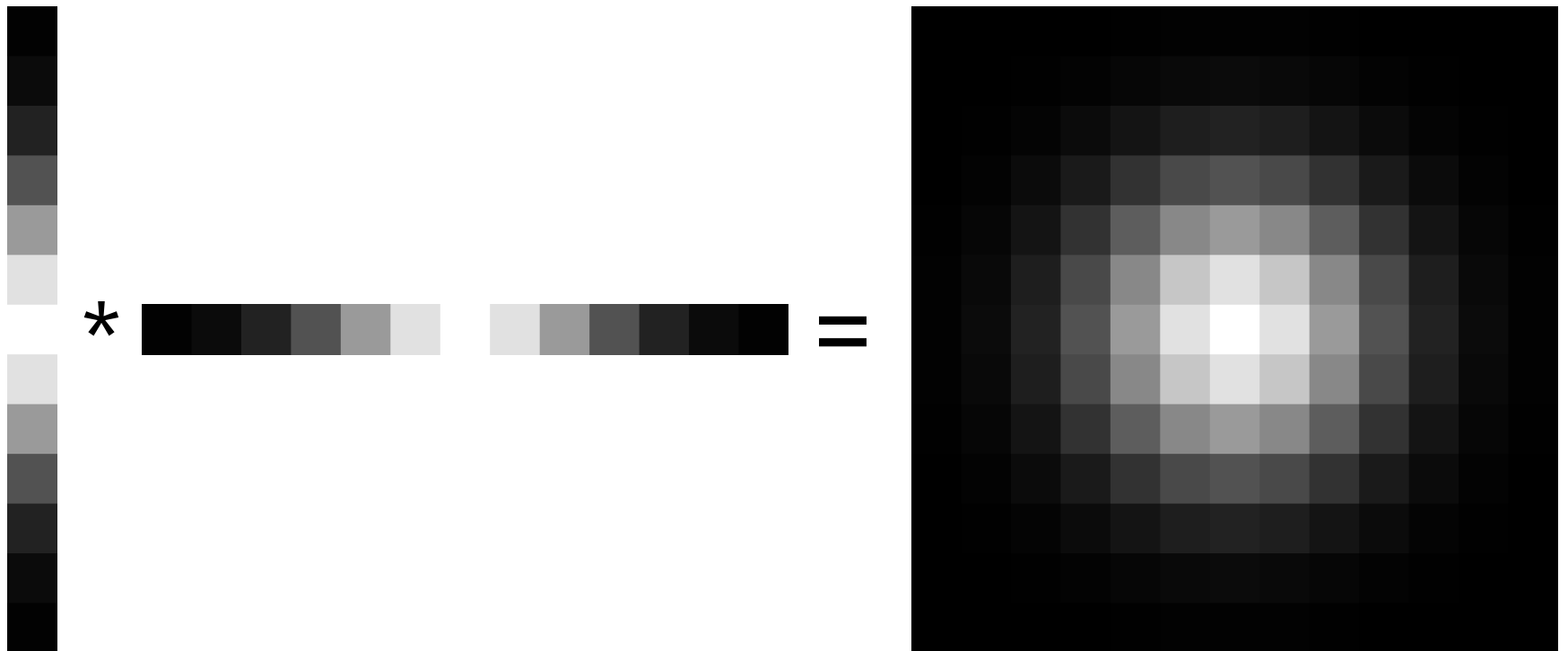
→

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

Separability

1D Gaussian * 1D Gaussian = 2D Gaussian

Image * 2D Gauss = Image * (1D Gauss * 1D Gauss)
= (Image * 1D Gauss) * 1D Gauss



Runtime Complexity

Image size = $N \times N = 6 \times 6$

Filter size = $M \times 1 = 3 \times 1$

I11	I12	I13	I14	I15	I16
I21	F1	I23	I24	I25	I26
I31	F2	I33	I34	I35	I36
I41	F3	I43	I44	I45	I46
I51	I52	I53	I54	I55	I56
I61	I62	I63	I64	I65	I66

What are my compute savings for a 13x13 filter?

```
for ImageY in range(N):
    for ImageX in range(N):
        for FilterY in range(M):
            ...
for ImageY in range(N):
    for ImageX in range(N):
        for FilterX in range(M):
            ...
```

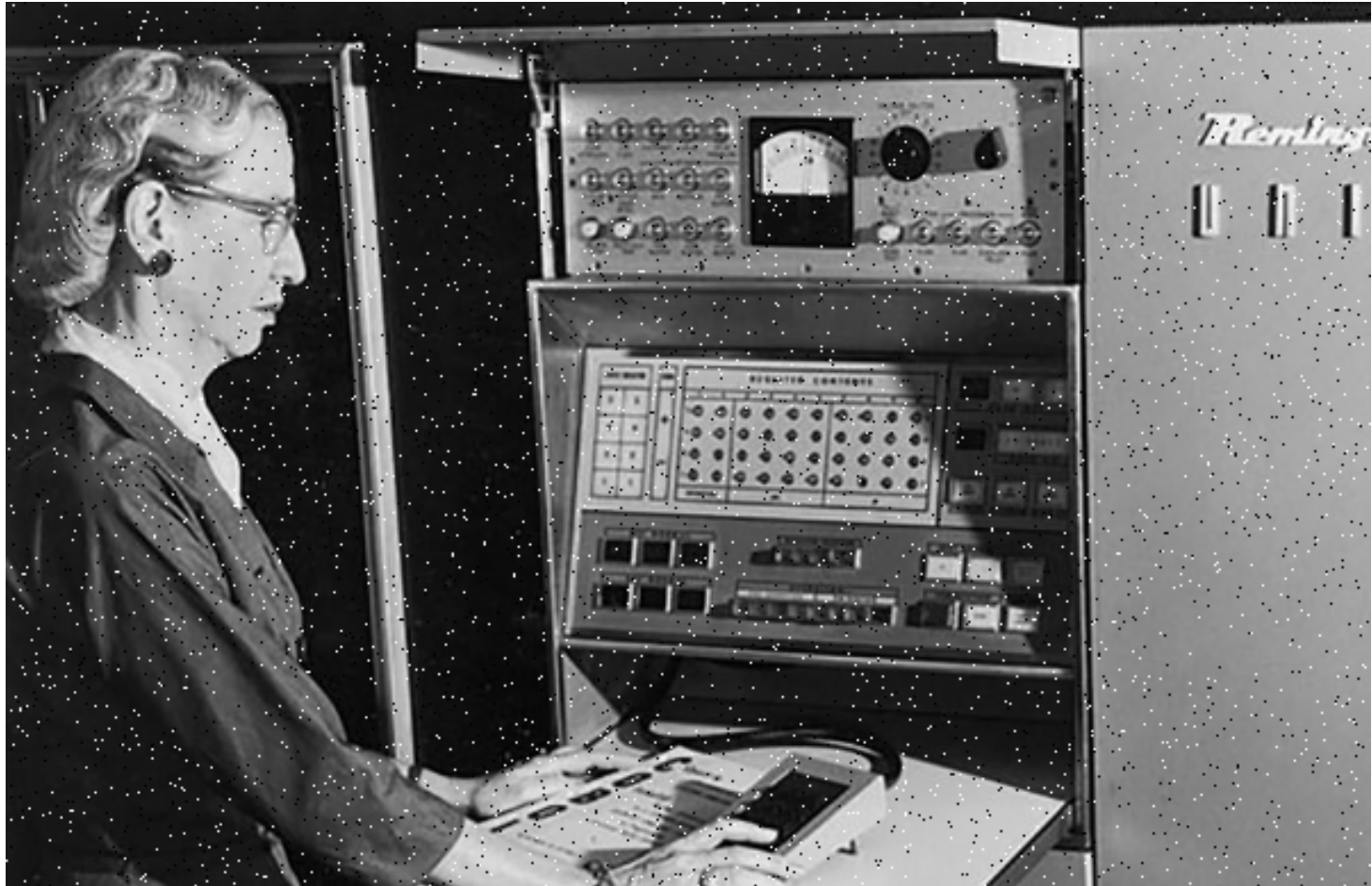
Time: $O(N^2M)$

Why Gaussian?

Gaussian filtering removes parts of the signal above a certain frequency. Often noise is high frequency and signal is low frequency.

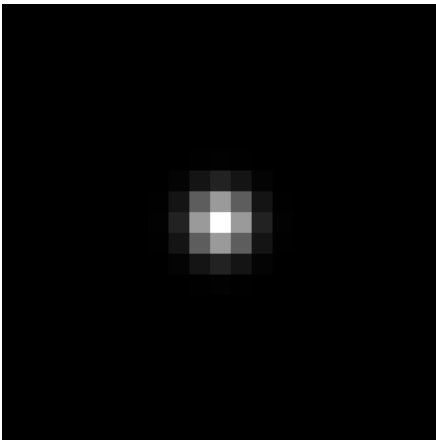


Where Gaussian Fails



Applying Gaussian Filters

$$\sigma = 1$$



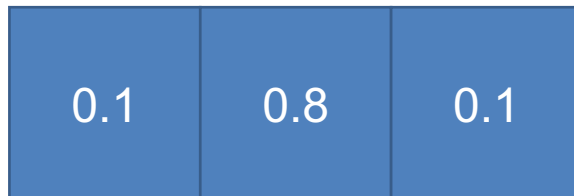
Why Does This Fail?

Means can be arbitrarily distorted by outliers

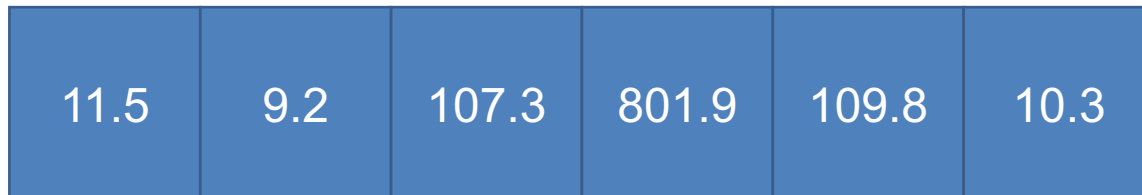
Signal



Filter



Output



What else is an “average” other than a mean?

Non-linear Filters (2D)

40	81	13	22
125	830	76	80
144	92	108	95
132	102	106	87

[040, 081, 013, 125, 830, 076, 144, 092, 108]

↓ Sort ↓

[013, 040, 076, 081, 092, 108, 125, 144, 830]

↓
92

[830, 076, 080, 092, 108, 095, 102, 106, 087]

↓ Sort ↓

[076, 080, 087, 092, 095, 102, 106, 108, 830]

↓
95

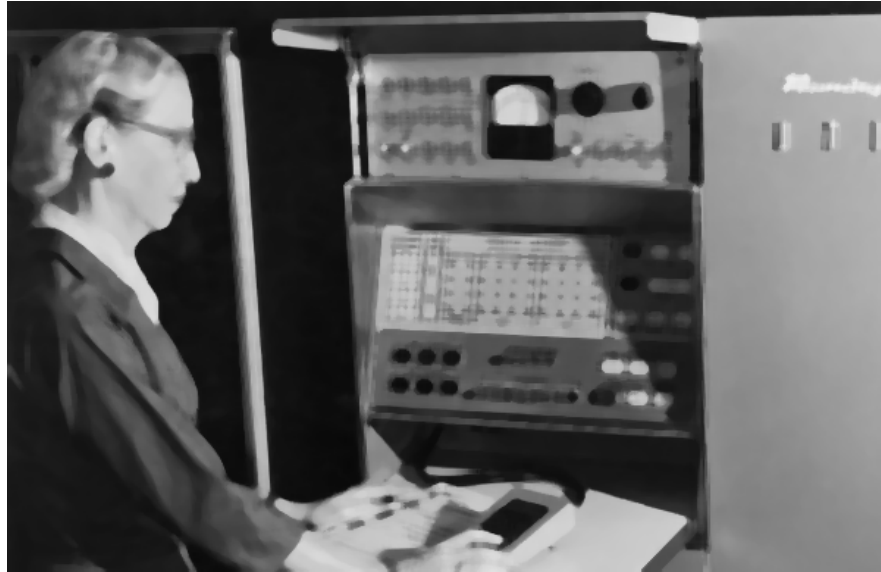
Applying Median Filter

Median
Filter
(size=3)



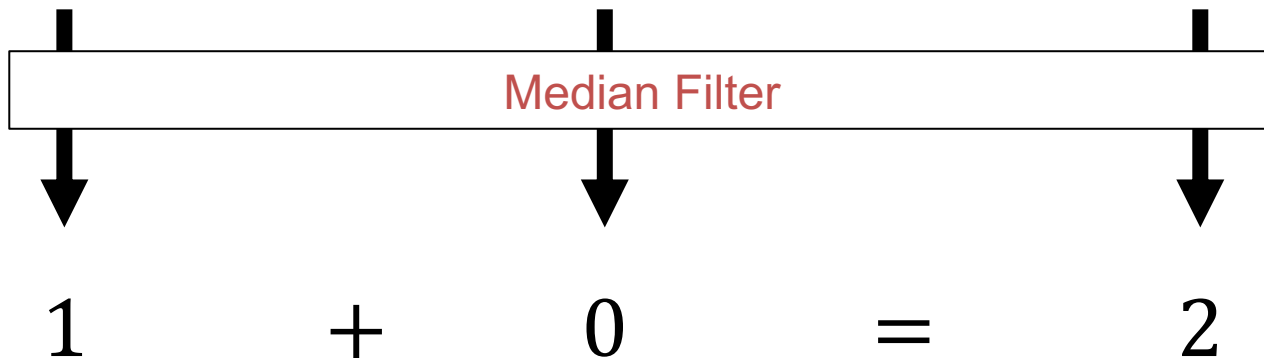
Applying Median Filter

Median
Filter
(size = 7)



Is Median Filtering Linear?

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 2 & 2 & 2 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}$$



Some Examples of Filtering

Filtering – Sharpening

Image



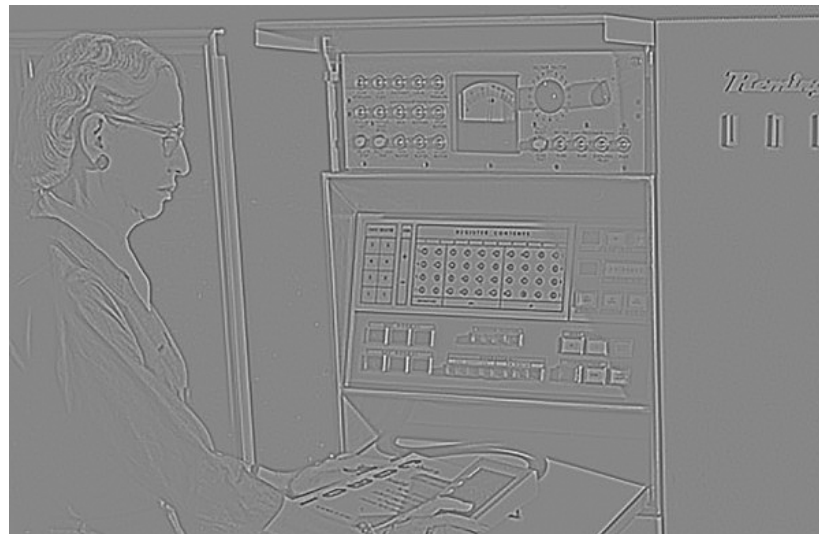
Smoothed



-

Details

=



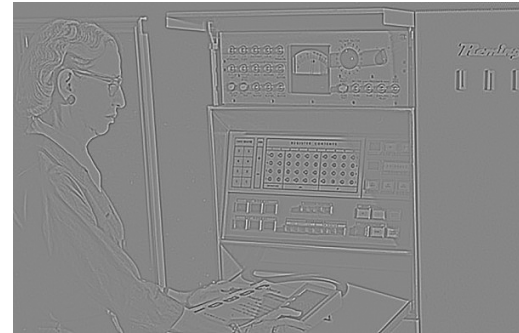
Filtering – Sharpening

Image



$+ \alpha$

Details



“Sharpened” $\alpha=1$

=



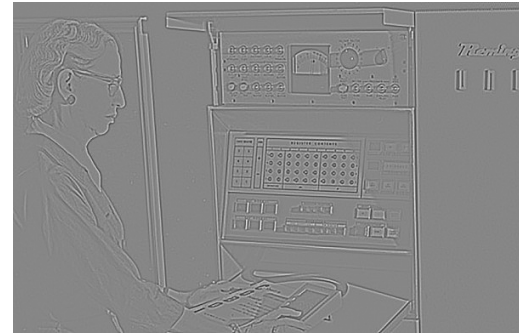
Filtering – Sharpening

Image



+ α

Details



“Sharpened” $\alpha=0$

=



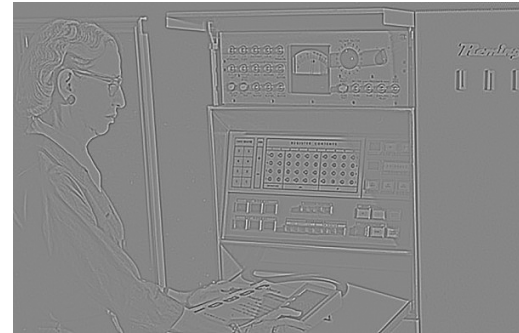
Filtering – Sharpening

Image



+ α

Details



“Sharpened” $\alpha=2$

=



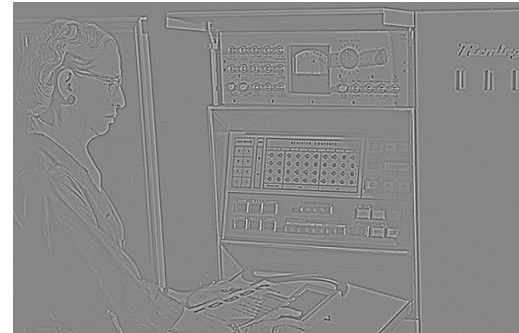
Filtering – Sharpening

Image



+ α

Details



“Sharpened” $\alpha=0$

=



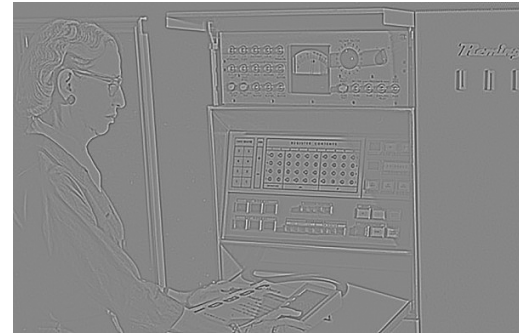
Filtering – Extreme Sharpening

Image



+ α

Details



“Sharpened” $\alpha=10$

=



Filtering

What's this Filter?

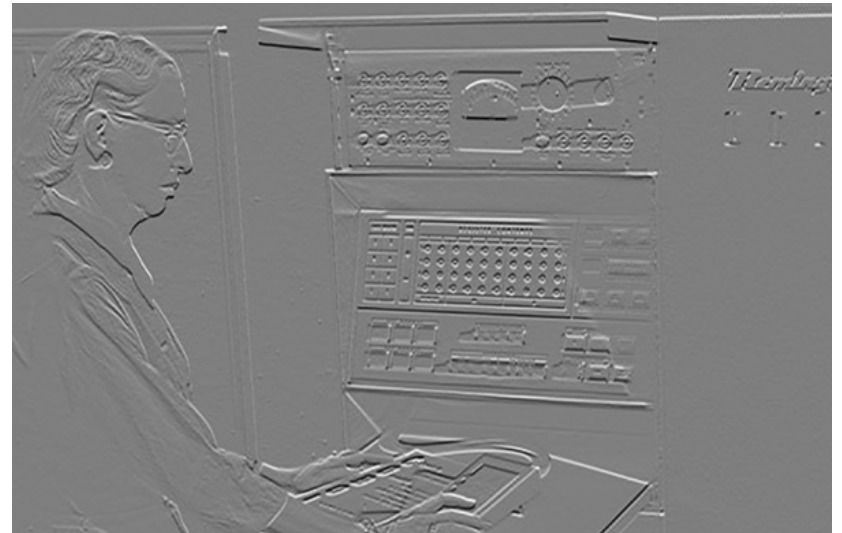
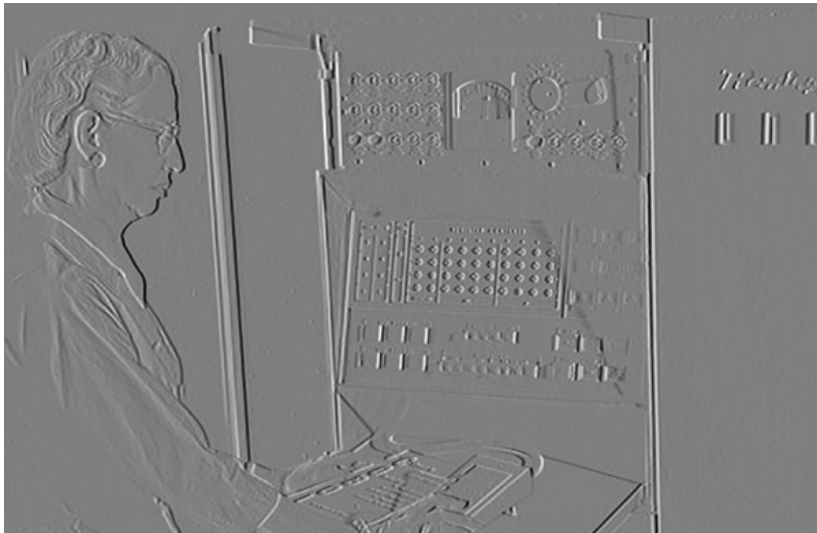
-1	0	1
----	---	---

Dx

-1	0	1
----	---	---

^T

Dy



Filtering – Derivatives

$$(Dx^2 + Dy^2)^{1/2}$$



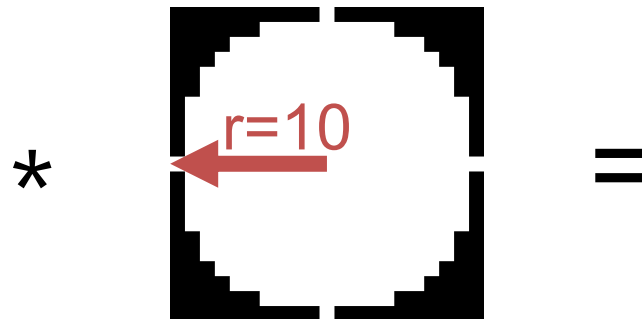
Filtering – Counting

How many “on” pixels have
10+ neighbors within 10 pixels?

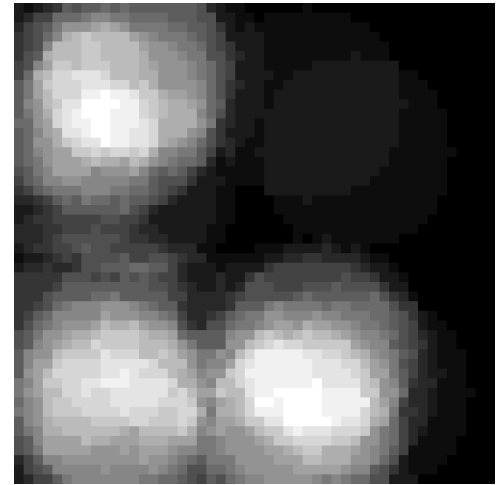
Pixels



Disk



???



Filtering – Counting

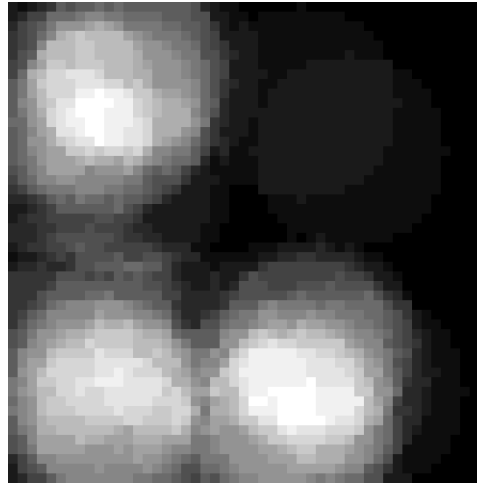
How many “on” pixels have
10+ neighbors within 10 pixels?

Pixels



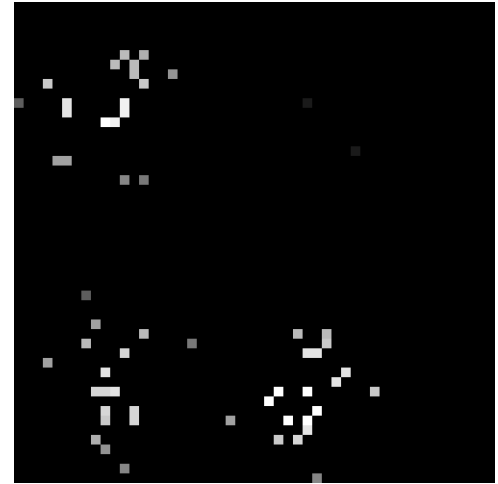
X

Density



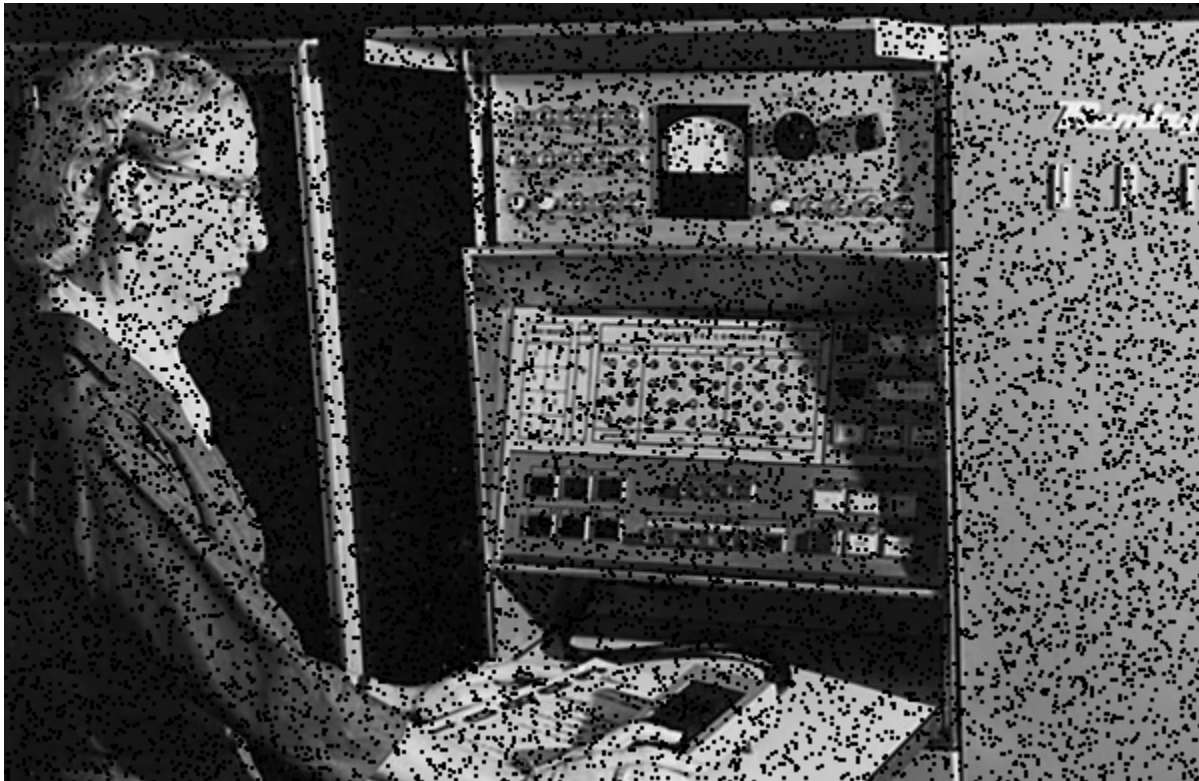
=

Answer



Filtering – Missing Data

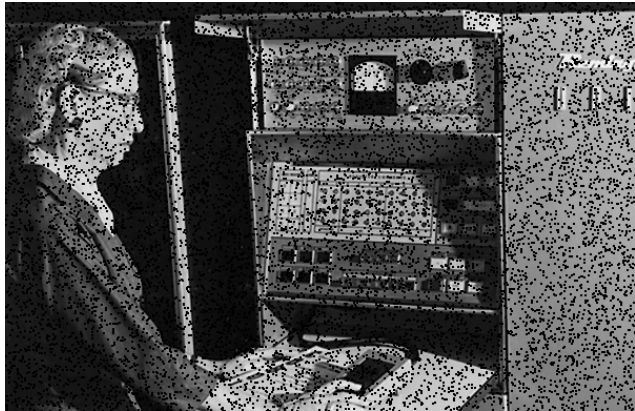
Oh no! Missing data!
(and we know where)



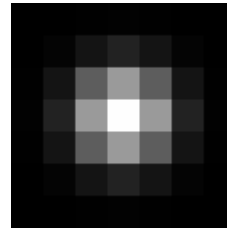
Common with many non-normal cameras (e.g., depth cameras)

Filtering – Missing Data

Image

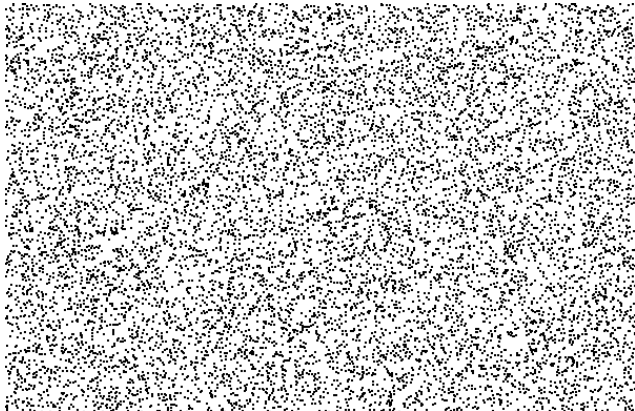


*

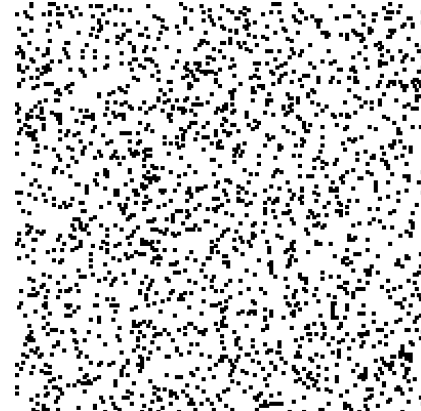
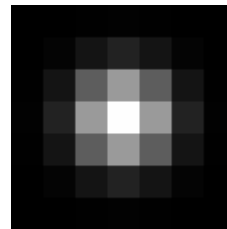


Per-element /

Binary
Mask

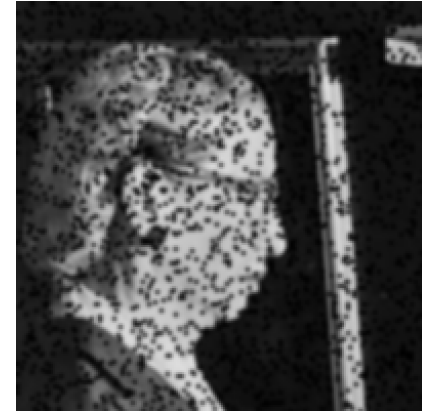
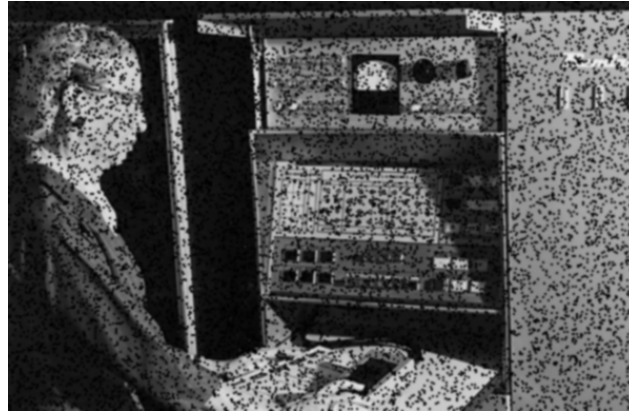


*



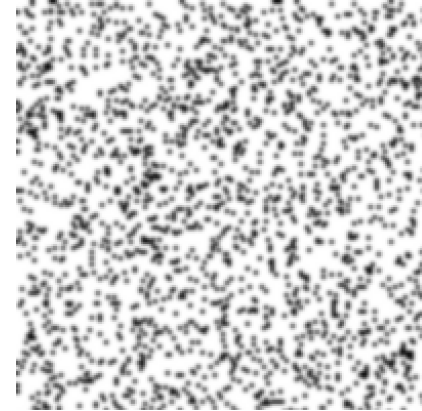
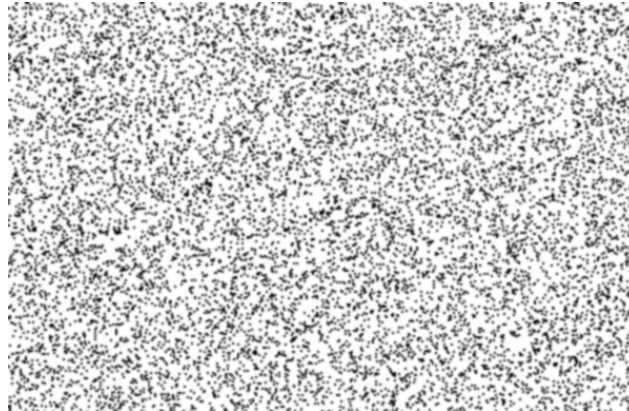
Filtering – Missing Data

Image



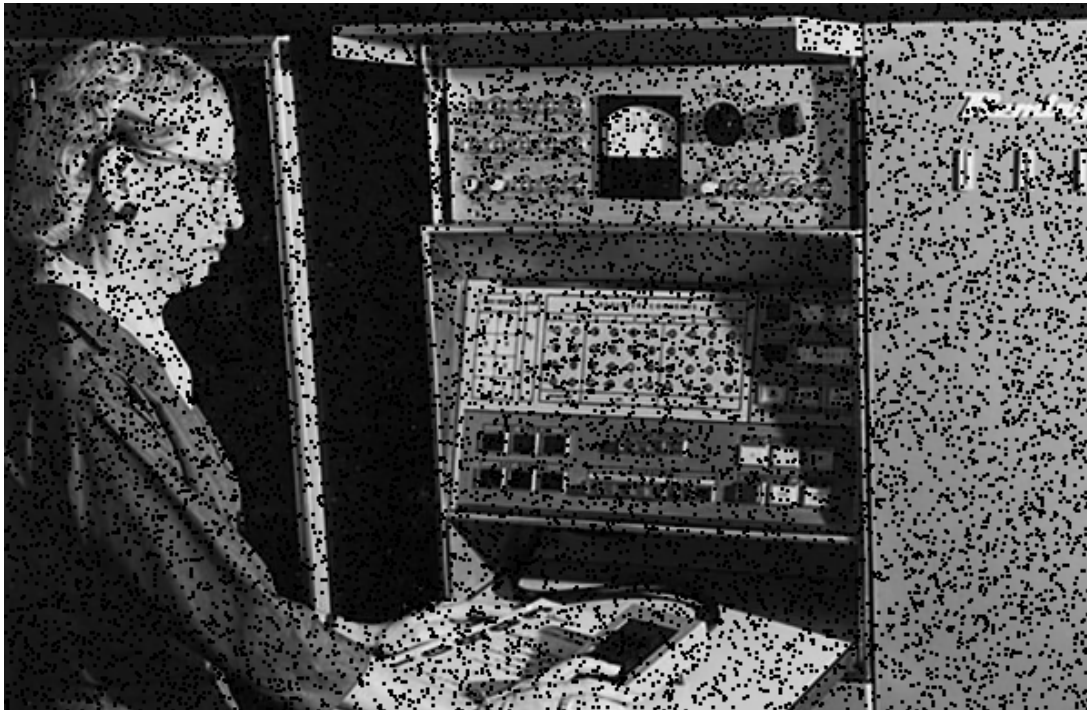
Per-element /

Binary
Mask



Filtering – Missing Data

Before



Filtering – Missing Data

After



Filtering – Missing Data

After (without missing data)



Application: Hybrid Images



A. Oliva, A. Torralba, P.G. Schyns,
[Hybrid Images](#), SIGGRAPH 2006

Changing expression

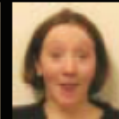
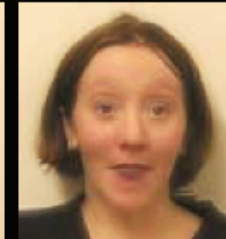


SIGGRAPH2006

Sad

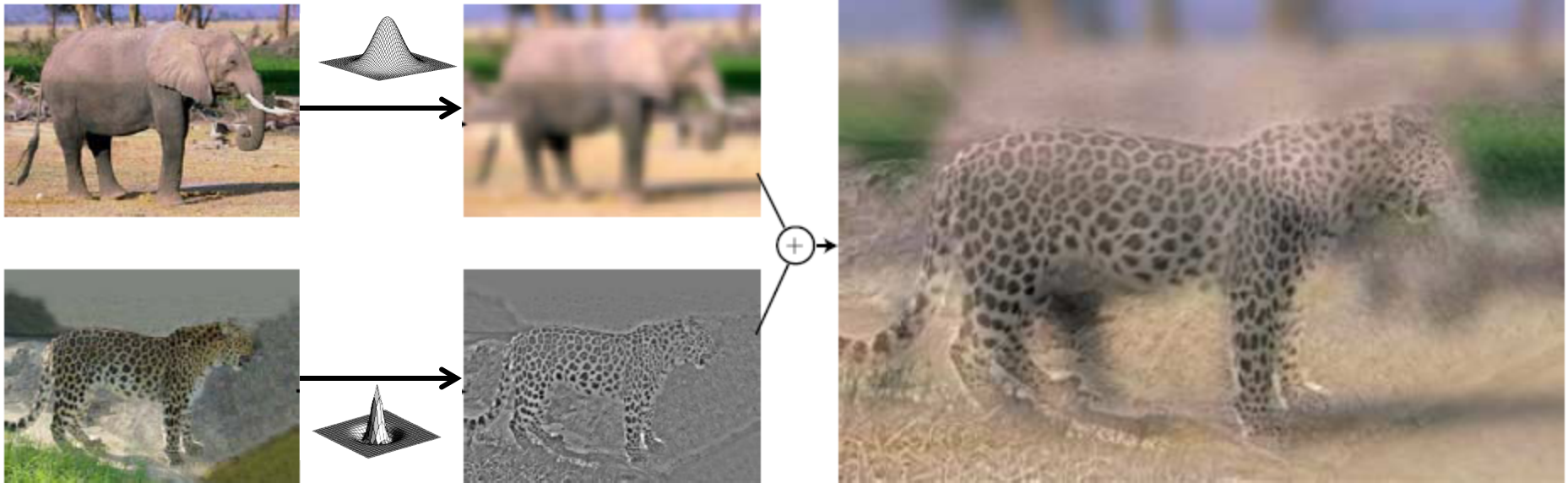


Surprised



Application: Hybrid Images

Gaussian Filter



Laplacian Filter

- A. Oliva, A. Torralba, P.G. Schyns, [Hybrid Images](#), SIGGRAPH 2006