Lecture 8:
Image Filtering II

Administrative

HW1 is due Tomorrow, Wednesday 2/5 11:59pm

HW2 should be released tomorrow,
due Wednesday 2/19 11:59pm

Last Time: Image Filtering

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

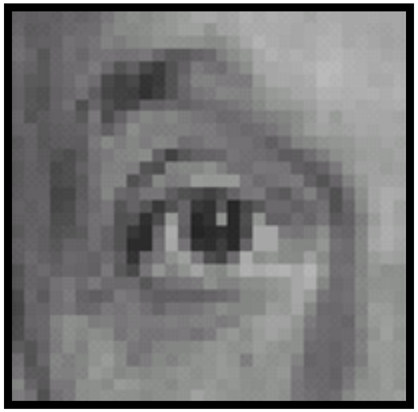
Filter

F11	F12	F13
F21	F22	F23
F31	F32	F33

Output

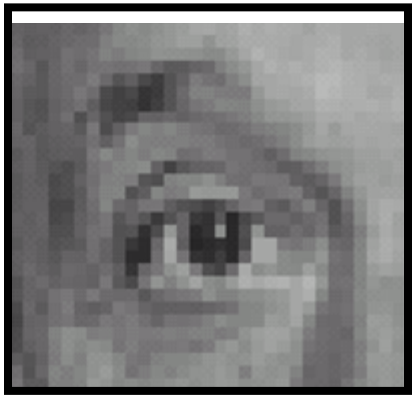
O11	O12	O13	O14
O21	O22	O23	O24
O31	O32	O33	O34

Last Time: Image Filtering



Original

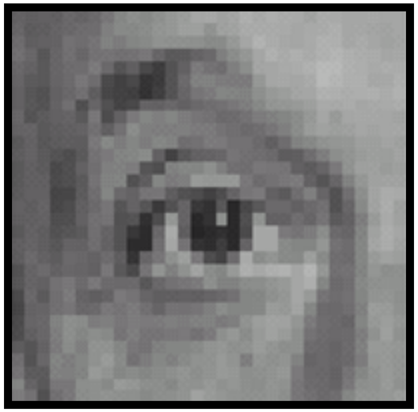
0	1	0
0	0	0
0	0	0



Shifted
DOWN
1 pixel

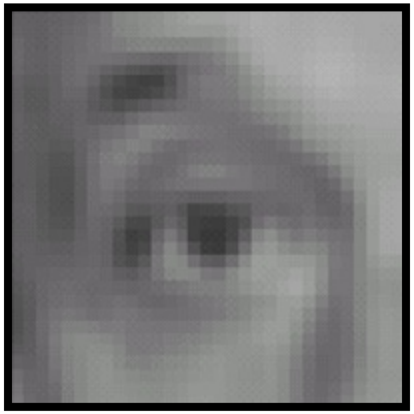
Slide Credit: D. Lowe

Last Time: Image Filtering



Original

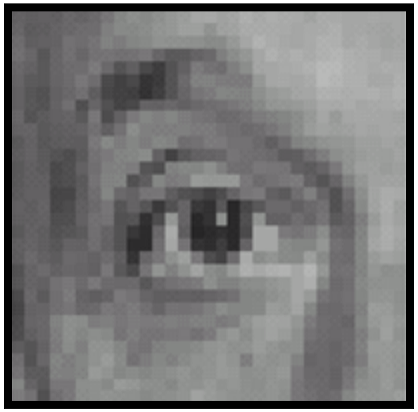
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9



Blur
(Box Filter)

Slide Credit: D. Lowe

Last Time: Image Filtering



Original

0	0	0
0	2	0
0	0	0

-

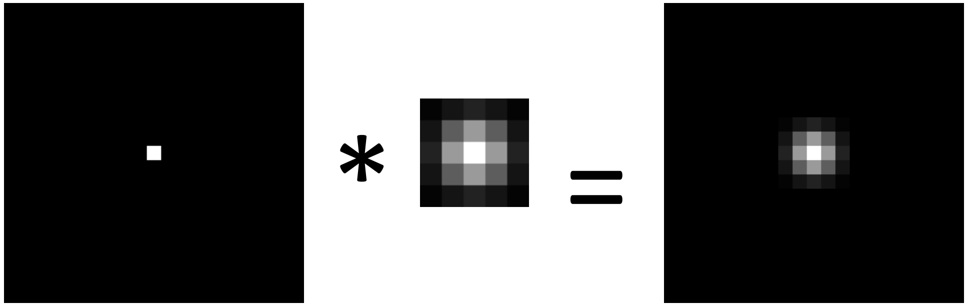
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Sharpened
(Acccentuates difference from local average)

Slide Credit: D. Lowe

Last Time: Convolution

- Linear: $f(aI + bI') = af(I) + bf(I')$
- Shift-Invariant: $\text{shift}(f(I)) = f(\text{shift}(I))$
- Any shift-invariant, linear operation is a convolution ($\ast$)
- Commutative: $f \ast g = g \ast f$
- Associative: $(f \ast g) \ast h = f \ast (g \ast h)$
- Distributes over +: $f \ast (g + h) = f \ast g + f \ast h$
- Scalars factor out: $kf \ast g = f \ast kg = k(f \ast g)$
- Identity (a single one with all zeros):



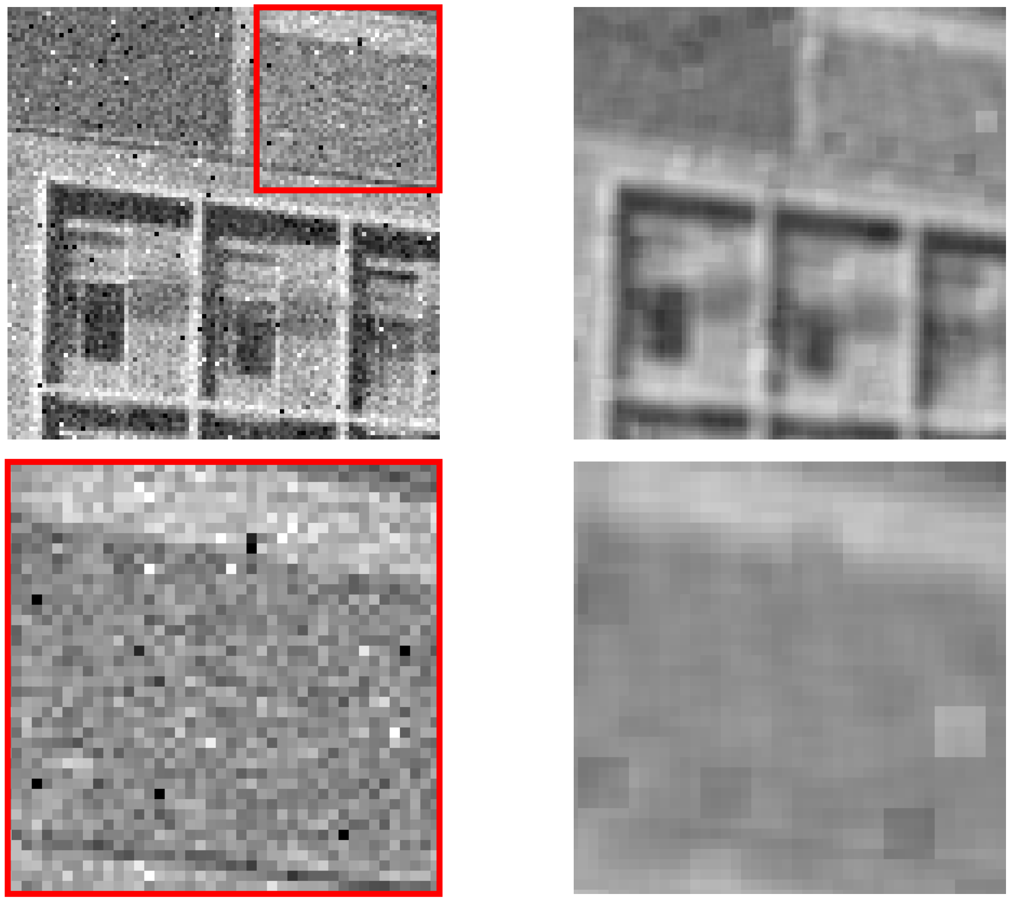
Property List: K. Grauman

Today:
Applications of
Linear Filters

Box Smoothing

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

Input　　　Box Filter



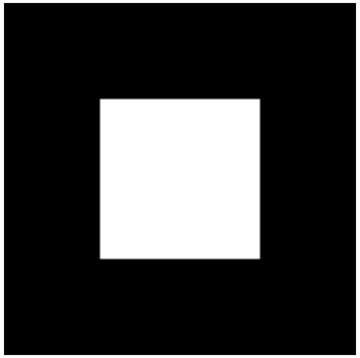
Problem: "Boxy" artifacts

Solution: Per-Pixel Weights

Intuition: weight contributions according to closeness to center.

Box Smoothing:

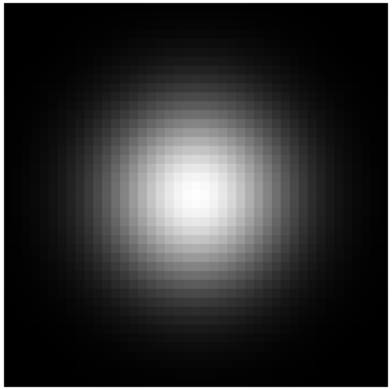
$$Filter_{ij} \propto 1$$



What's this?

Better Approach:

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

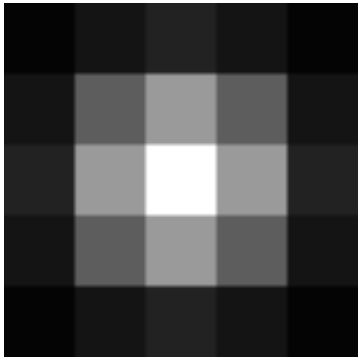


Recognize the Filter?

It's a Gaussian!

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

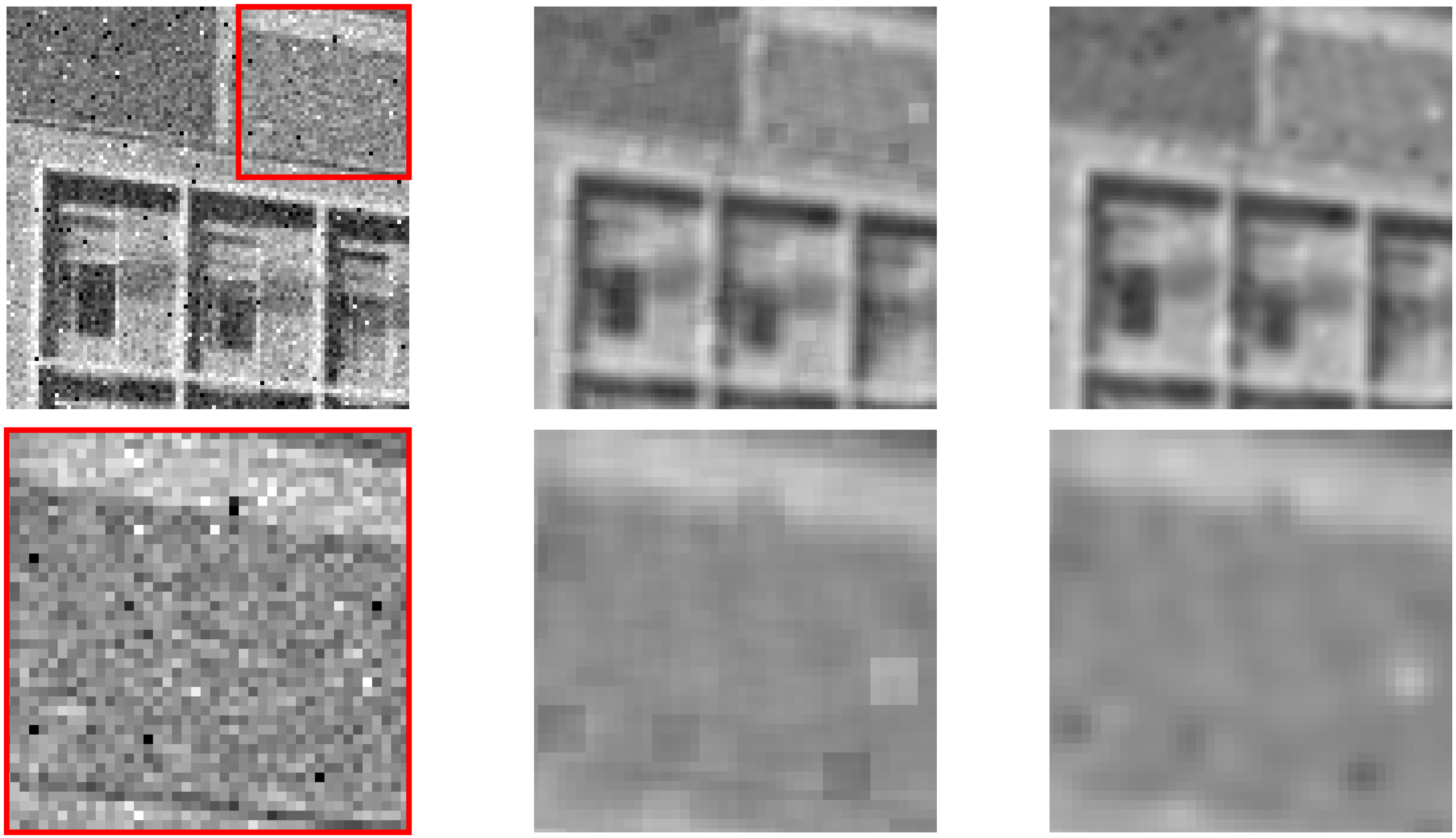
$$\begin{bmatrix} 0.003 & 0.013 & 0.022 & 0.013 & 0.003 \\ 0.013 & 0.060 & 0.098 & 0.060 & 0.013 \\ 0.022 & 0.098 & 0.162 & 0.098 & 0.022 \\ 0.013 & 0.060 & 0.098 & 0.060 & 0.013 \\ 0.003 & 0.013 & 0.022 & 0.013 & 0.003 \end{bmatrix}$$



Box Blur vs Gaussian Blur

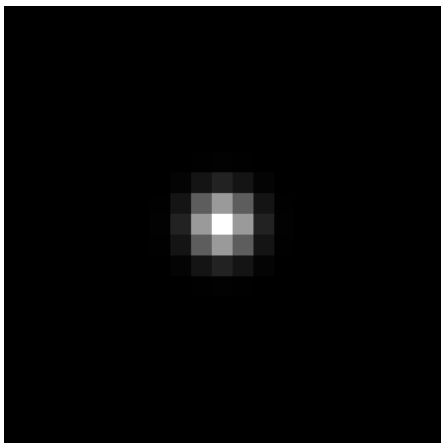
Still have some speckles, but it's not a big box

Input Box Filter Gauss. Filter

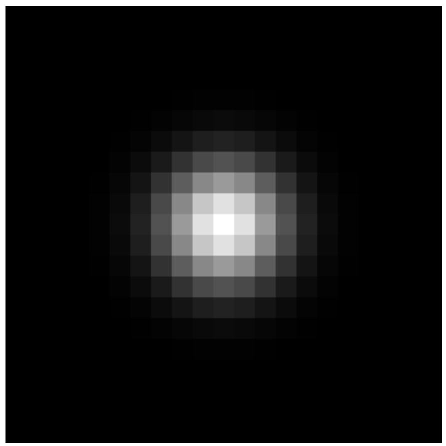


Gaussian Filters

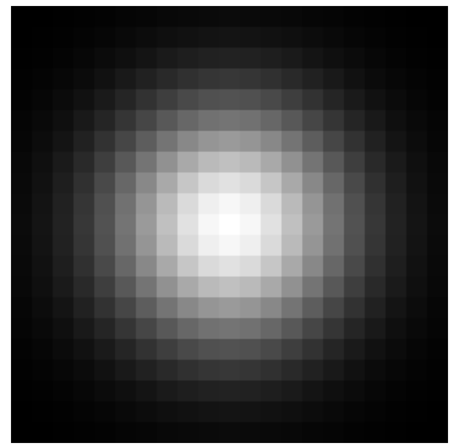
σ = 1
filter = 21x21



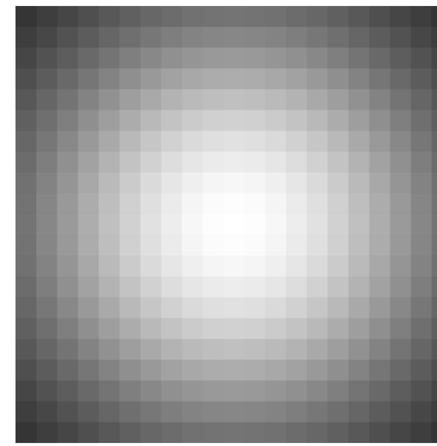
σ = 2
filter = 21x21



σ = 4
filter = 21x21

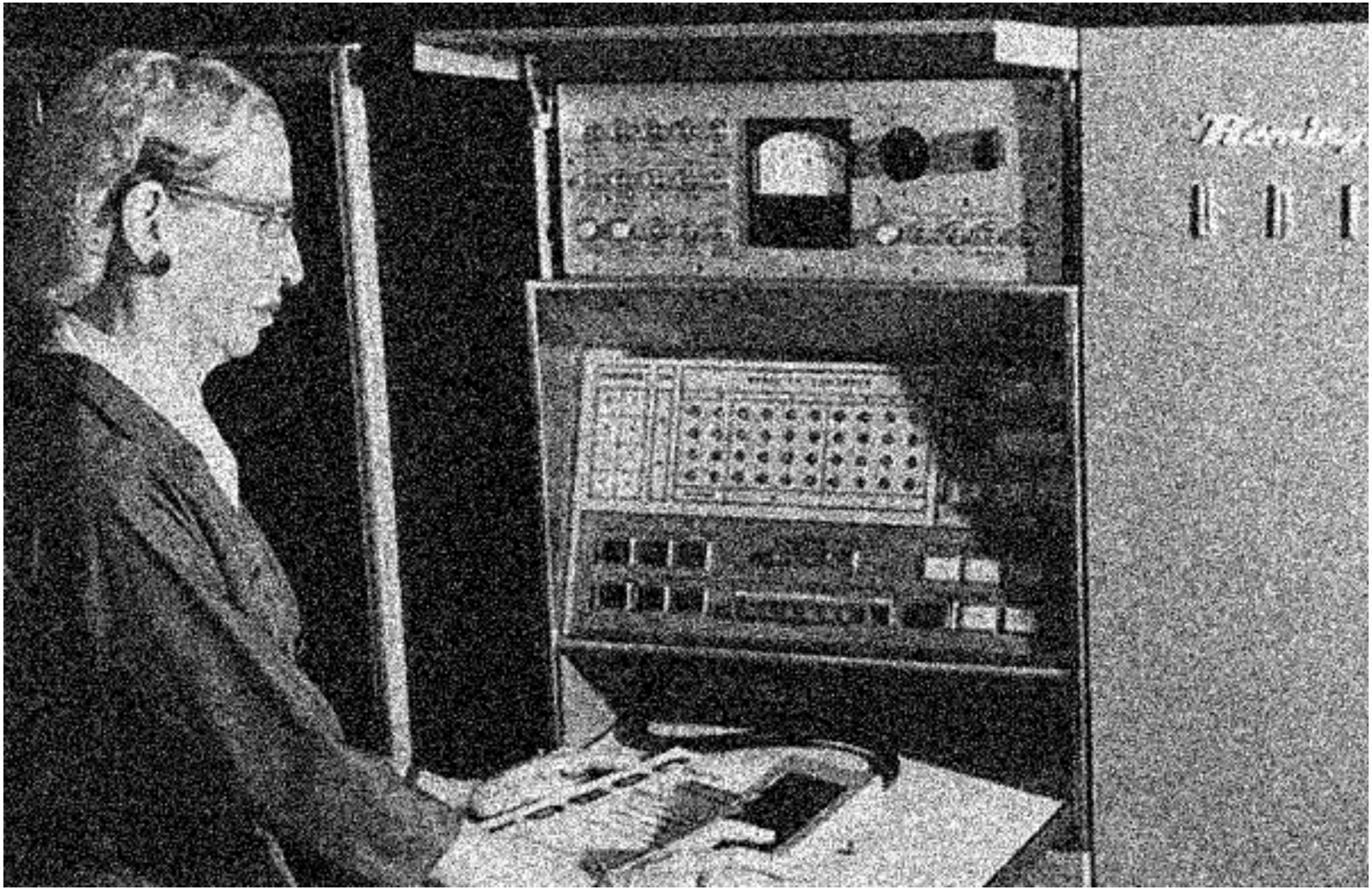


σ = 8
filter = 21x21



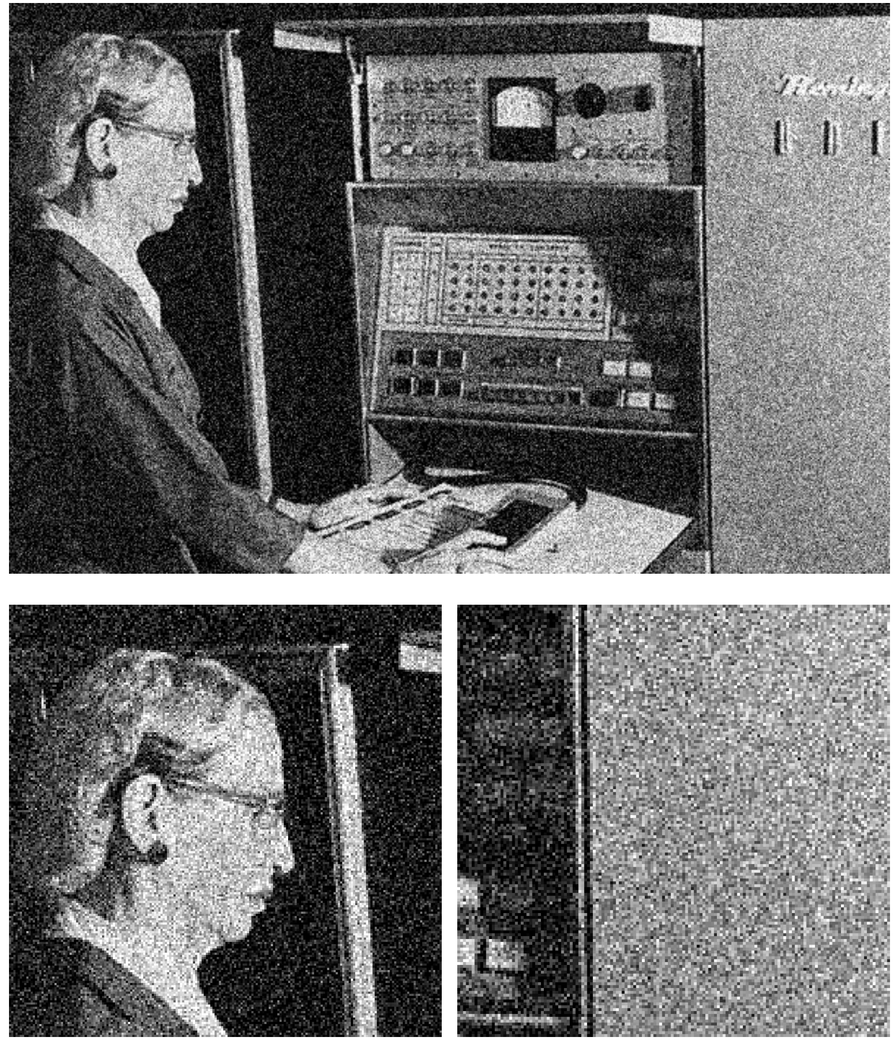
Note: filter visualizations are independently normalized throughout the slides so you can see them better

Applying Gaussian Filters



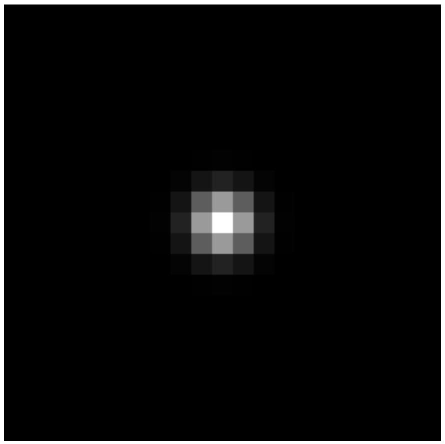
Applying Gaussian Filters

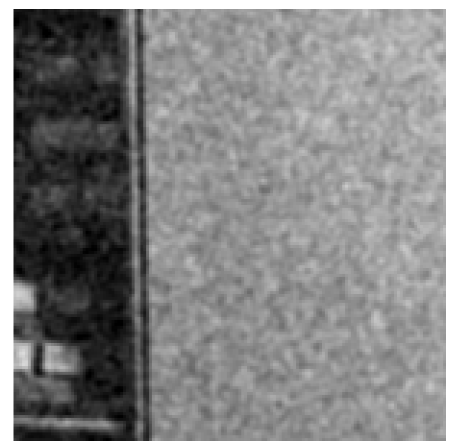
Input Image
(no filter)



Applying Gaussian Filters

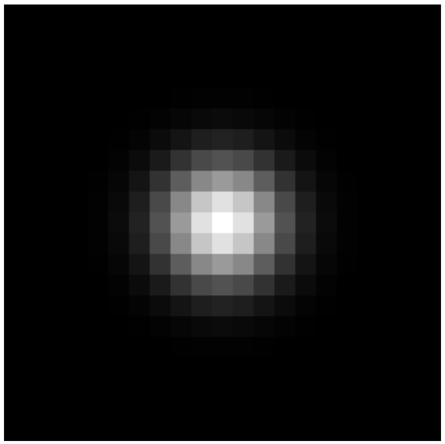
$\sigma = 1$

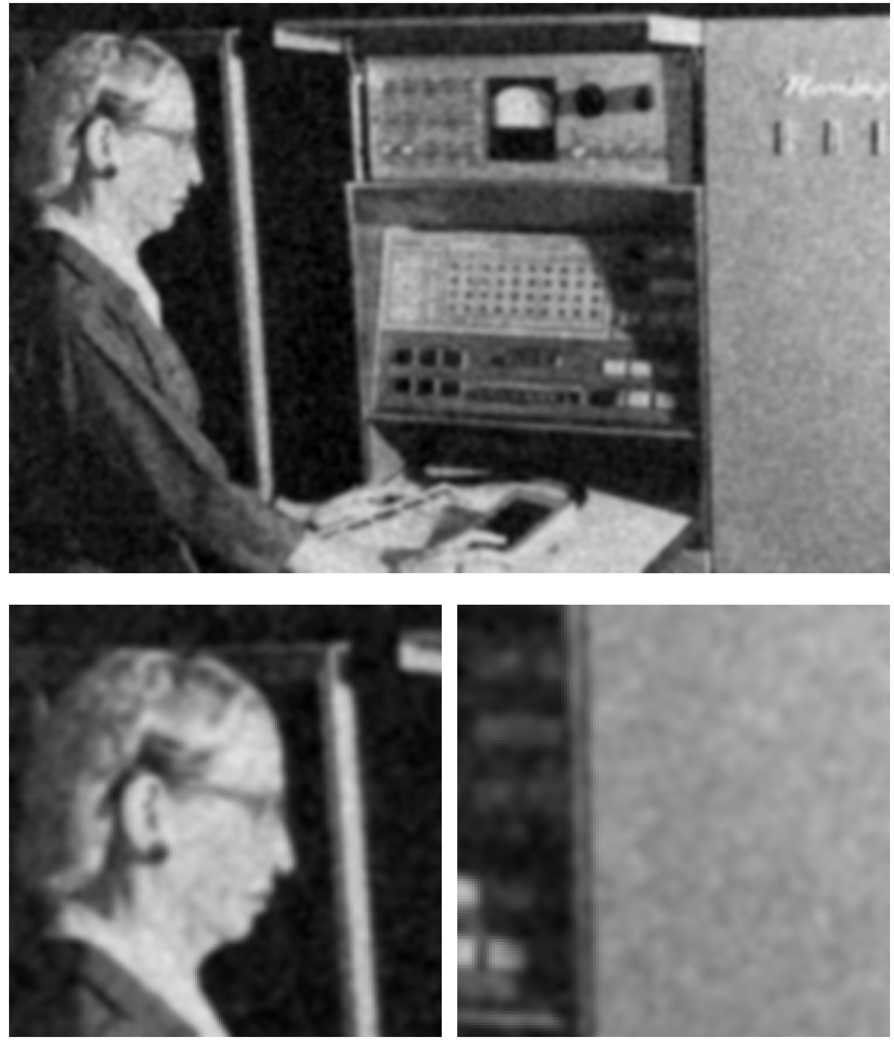




Applying Gaussian Filters

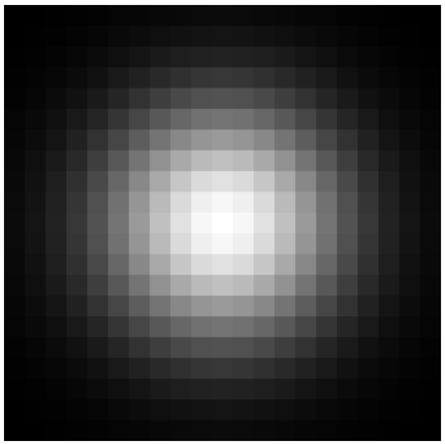
σ = 2





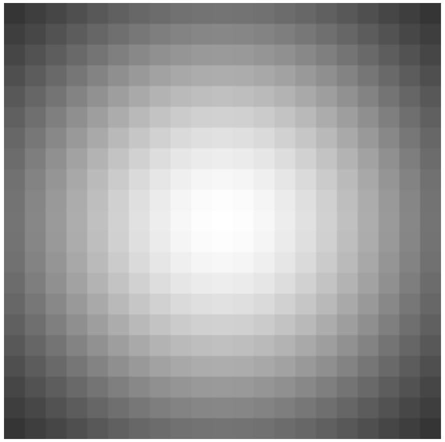
Applying Gaussian Filters

σ = 4



Applying Gaussian Filters

$\sigma = 8$



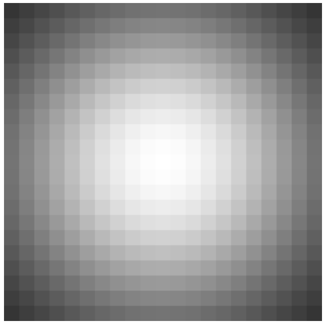
Gaussian Blur: Filter Size

Too small filter → bad approximation

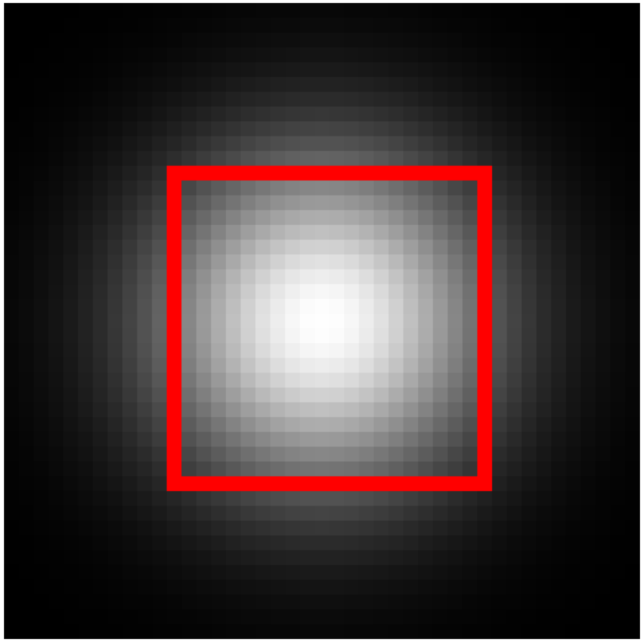
Want size ≈ 6σ (99.7% of energy)

Left far too small; right slightly too small!

$\sigma = 8$, size = 21



$\sigma = 8$, size = 43



Runtime Complexity

Image size = NxN = 6x6

Filter size = MxM = 3x3


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

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

Separable Filters

Conv(vector, transposed vector) → outer product

$$\begin{bmatrix} Fy1 \\ Fy2 \\ Fy3 \end{bmatrix} * \begin{bmatrix} Fx1 & Fx2 & Fx3 \end{bmatrix} = \begin{bmatrix} Fx1 * Fy1 & Fx2 * Fy1 & Fx3 * Fy1 \\ Fx1 * Fy2 & Fx2 * Fy2 & Fx3 * Fy2 \\ Fx1 * Fy3 & Fx2 * Fy3 & Fx3 * Fy3 \end{bmatrix}$$

(Using “full” convolution with zero padding)
(Also ignoring filter flips)

Separable Filters: Gaussian

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

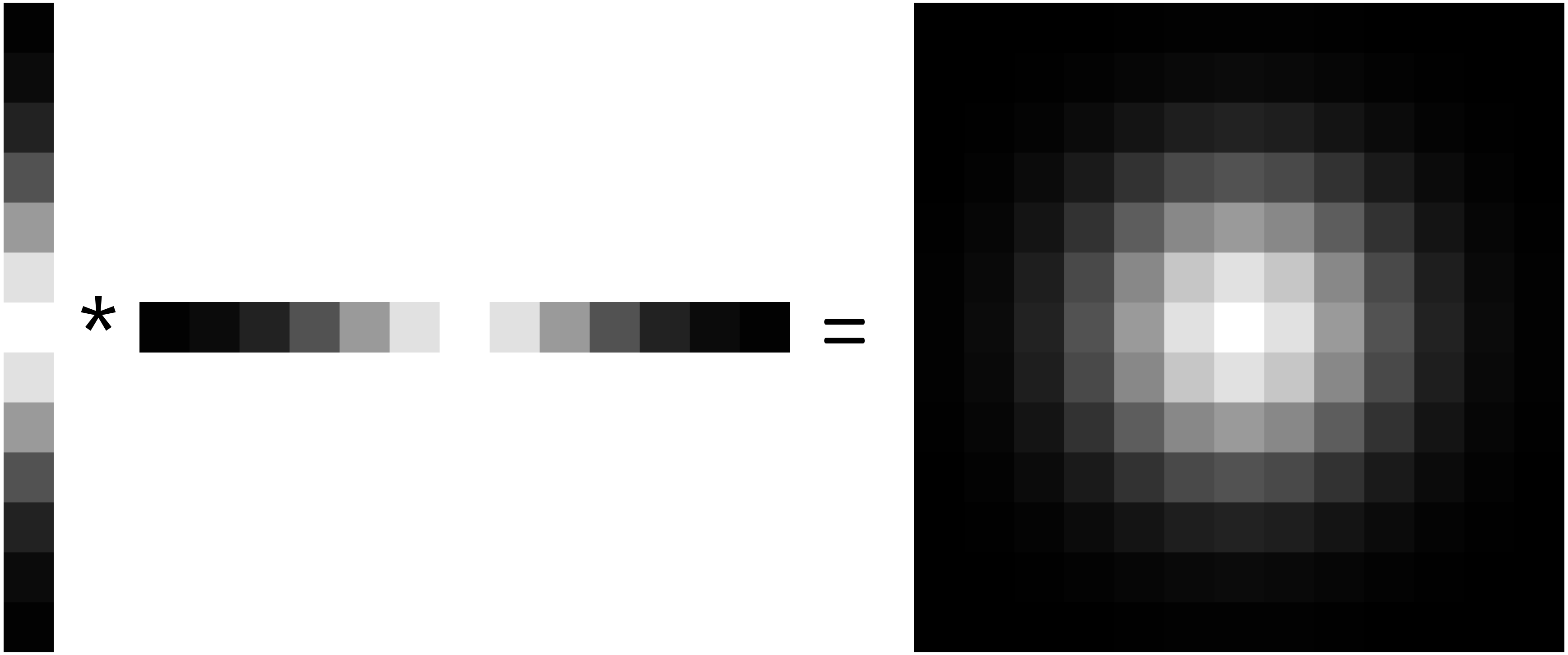
$$\rightarrow$$

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

Separable Filters: Gaussian

1D Gaussian $*$ 1D Gaussian = 2D Gaussian

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



Separable Filters: Runtime Complexity

Image size = NxN = 6x6

Filter size = Mx1 = 3x1

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?

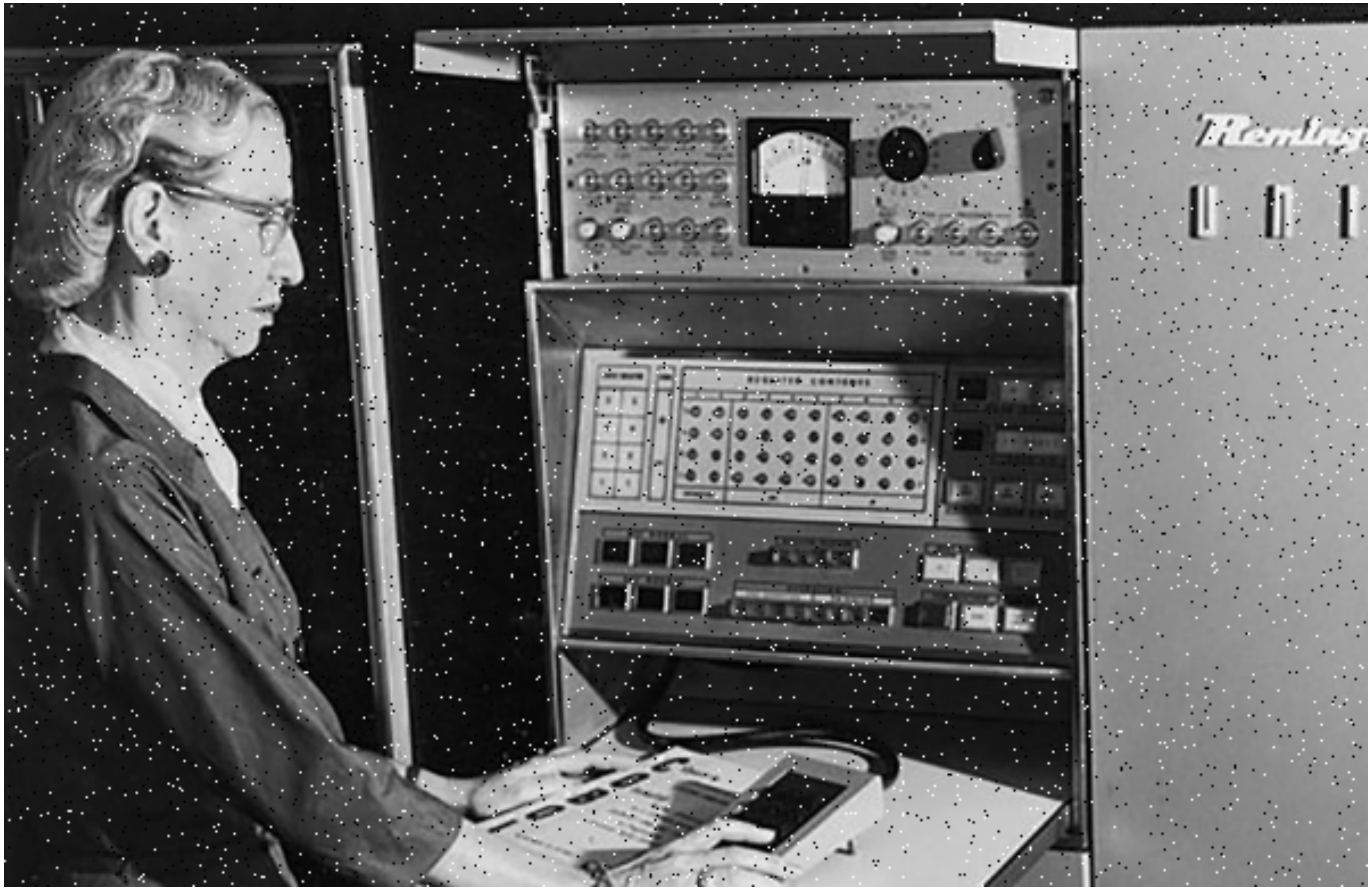
```
for ImageY in range(N):
    for ImageX in range(N):
        for FilterY in range(M):
            ...
for ImageY in range(N):
    for ImageX in range(N):
        for FilterX in range(M):
            ...
```

Time: $O(N^2M)$

Why Gaussian?

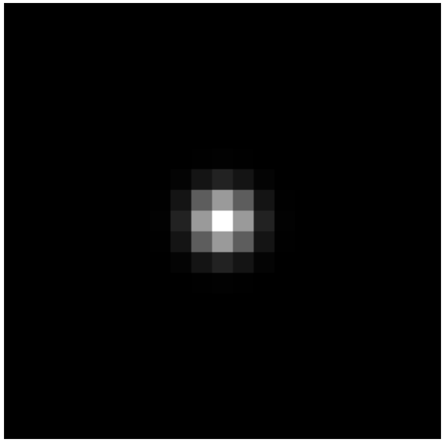
Gaussian filtering removes parts of the signal above a certain frequency. Often noise is high frequency and signal is low frequency.

Where Gaussian Fails



Where Gaussian Fails

σ = 1



Where Gaussian Fails

Means can be arbitrarily distorted by outliers

Signal

10	12	9	8	1000	11	10	12

Filter

0.1	0.8	0.1

Output

11.5	9.2	107.3	801.9	109.8	10.3

What else is an "average" other than a mean?

Median Filter

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]

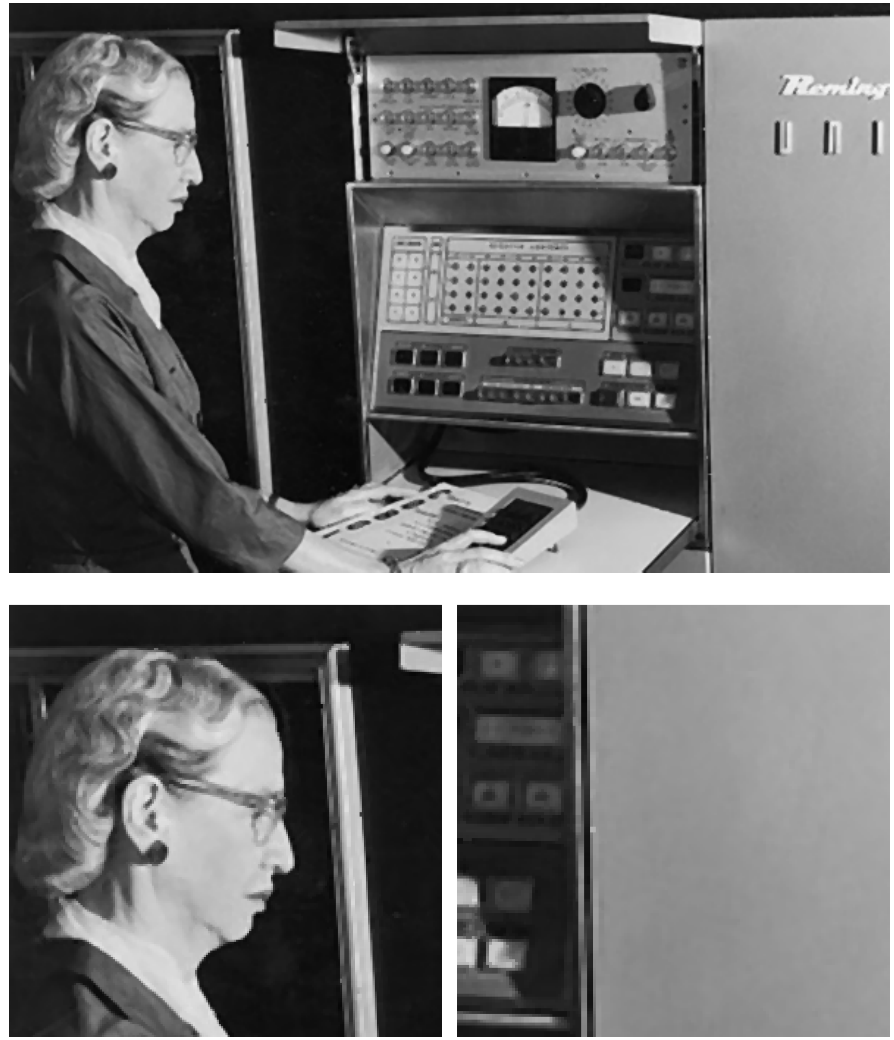
Sort

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

95

Median Filter

Median Filter (size=3)



Median Filter

Median
Filter
(size = 7)

Is Median Filter Linear?

If F is a linear filter then it must satisfy:

$$F(x + y) = F(x) + F(y)$$

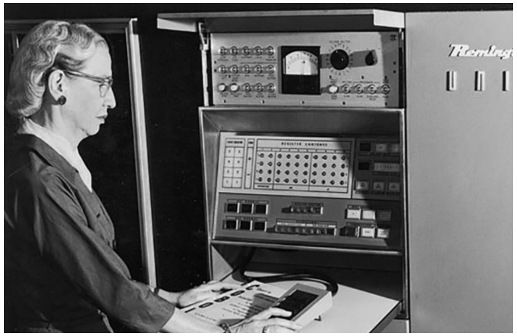
$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 2 & 2 & 2 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}$$

Median Filter

$$1 + 0 = 2$$

Sharpening Filter

Image

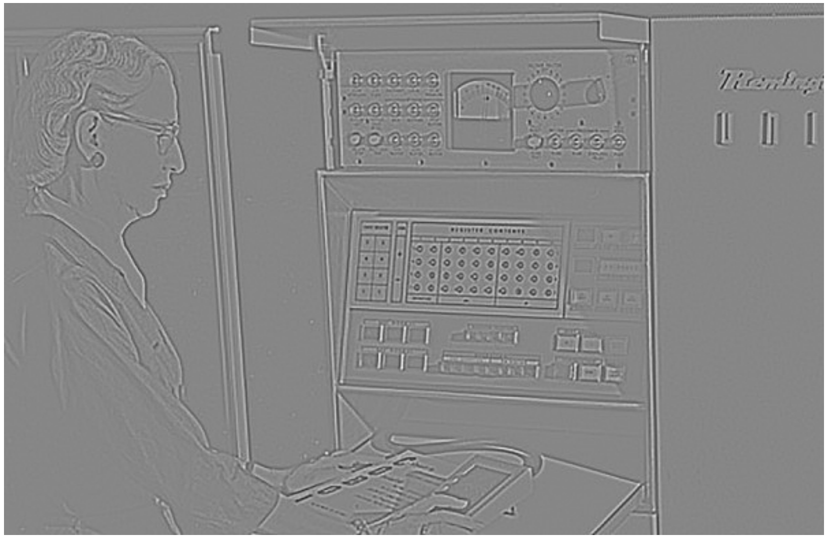


-

Smoothed

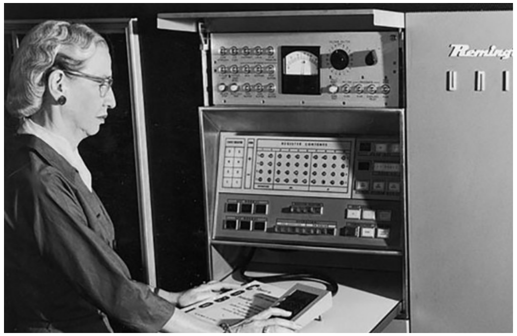
Details

=



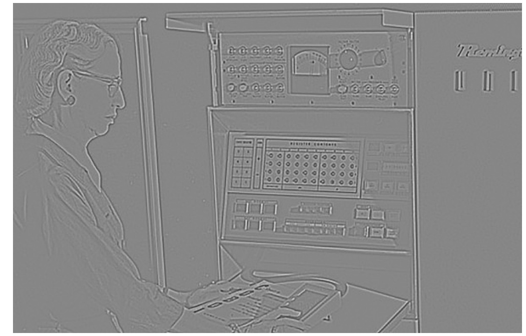
Sharpening Filter

Image



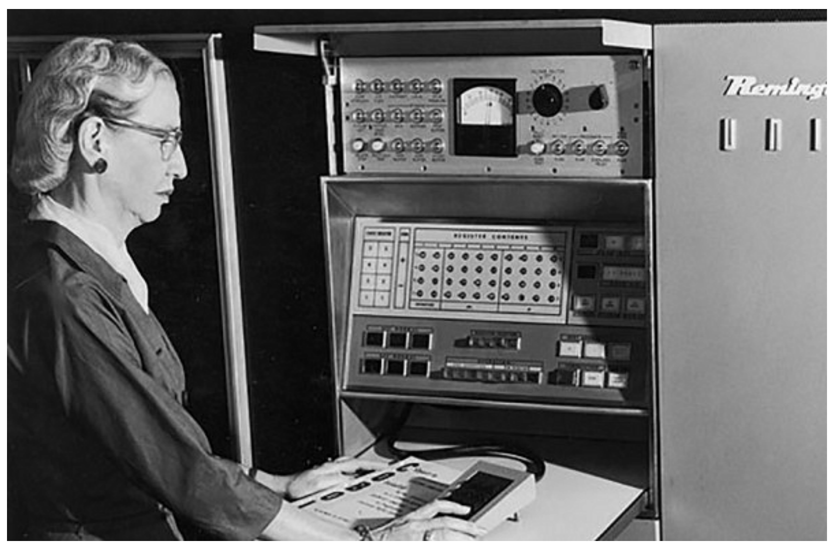
$+\alpha$

Details



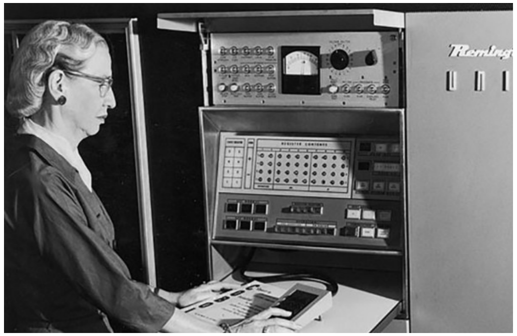
"Sharpened" $\alpha=1$

=



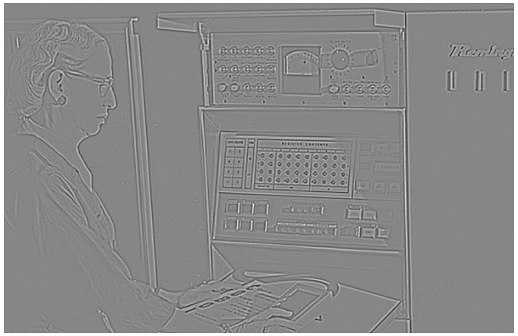
Sharpening Filter

Image



+α

Details

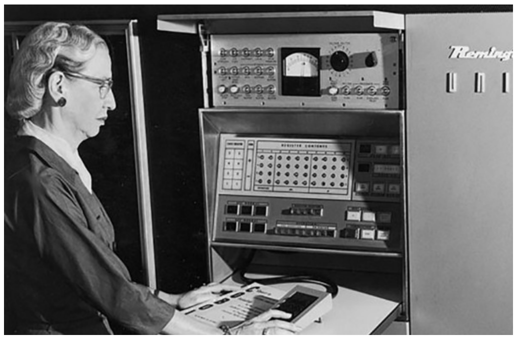


"Sharpened" α=0

=

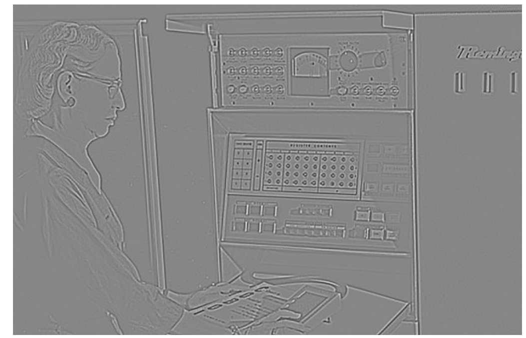
Sharpening Filter

Image



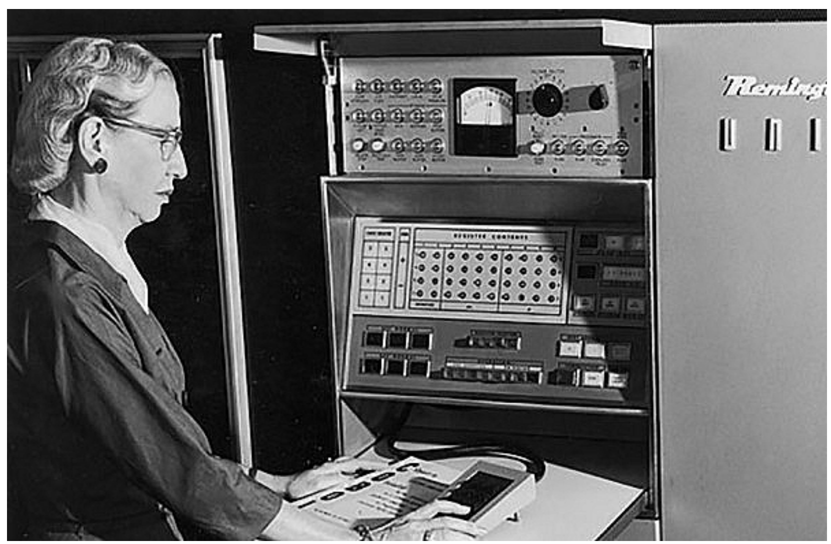
$+\alpha$

Details



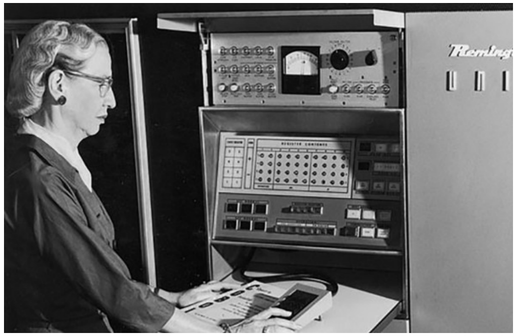
"Sharpened" $\alpha=2$

$=$



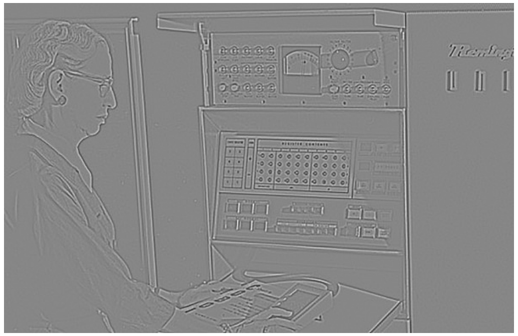
Sharpening Filter

Image



$+\alpha$

Details

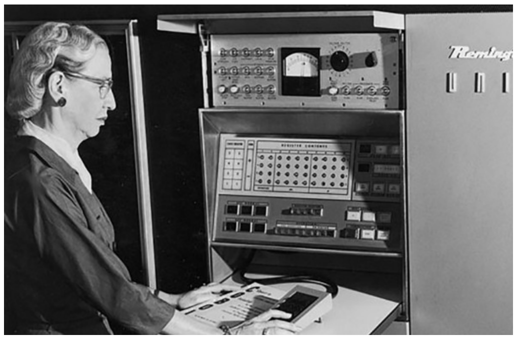


"Sharpened" $\alpha=0$

$=$

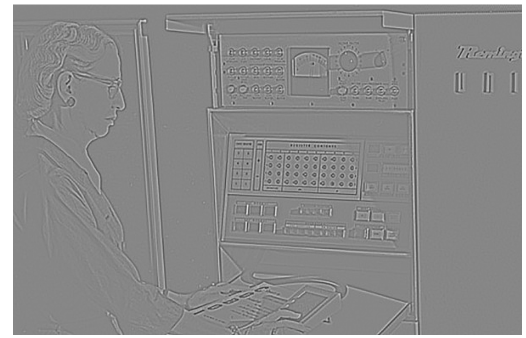
Sharpening Filter

Image



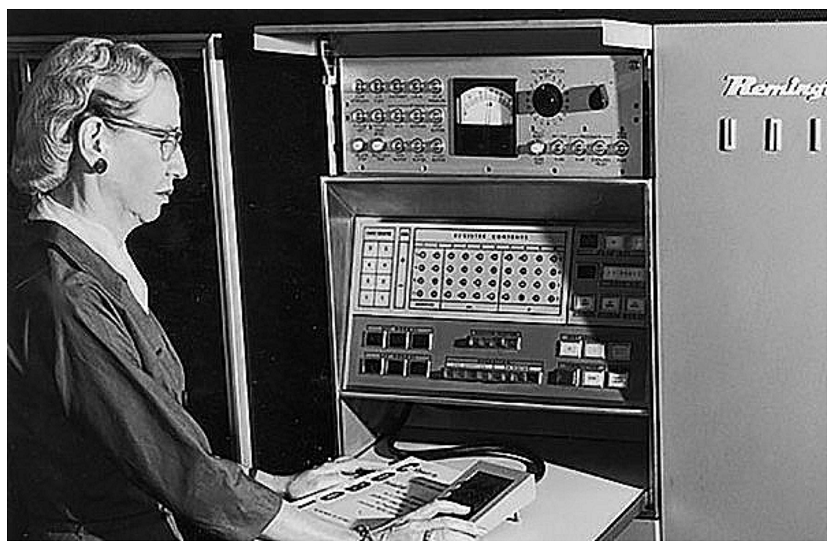
$+\alpha$

Details



“Sharpened” $\alpha=10$

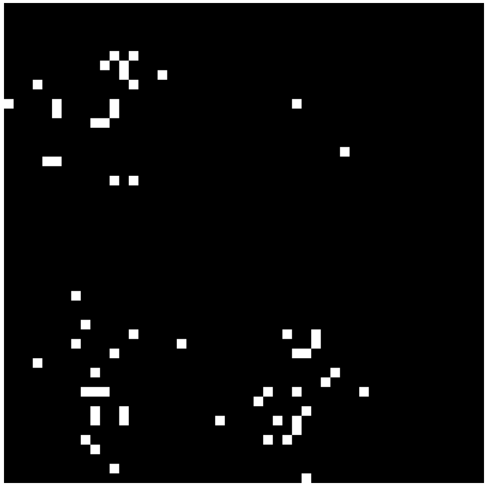
$=$



Filtering: Counting

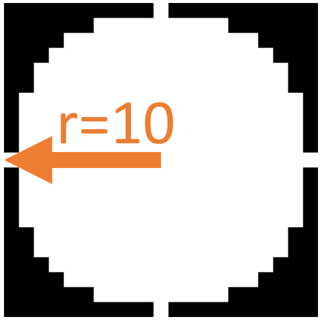
How many “on” pixels have
10+ neighbors within 10 pixels?

Pixels



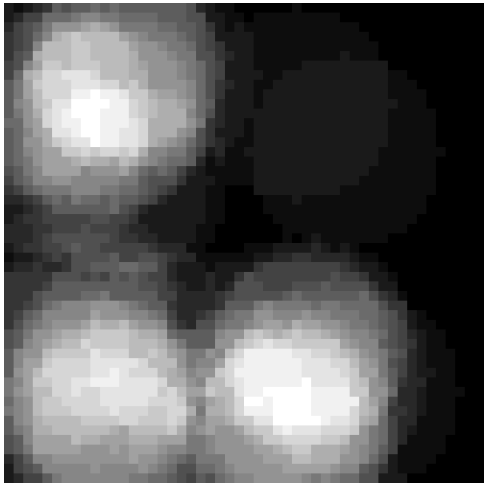
Disk

*



=

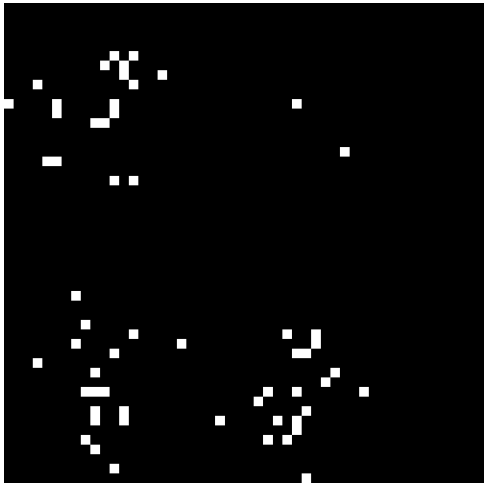
???



Filtering: Counting

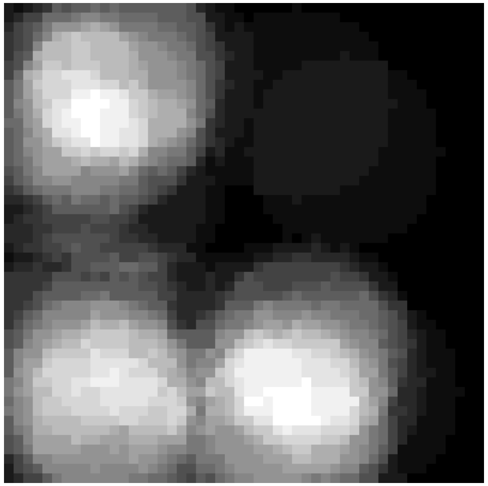
How many “on” pixels have
10+ neighbors within 10 pixels?

Pixels



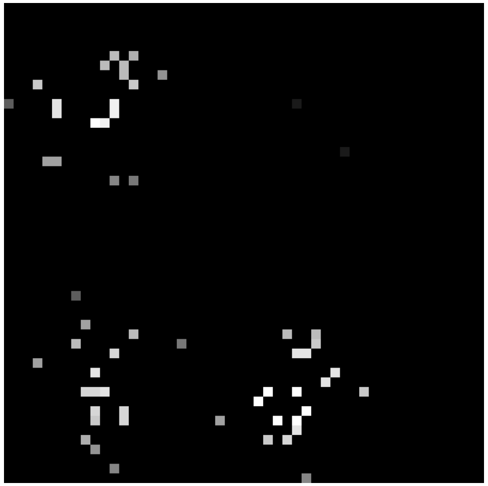
X

Density



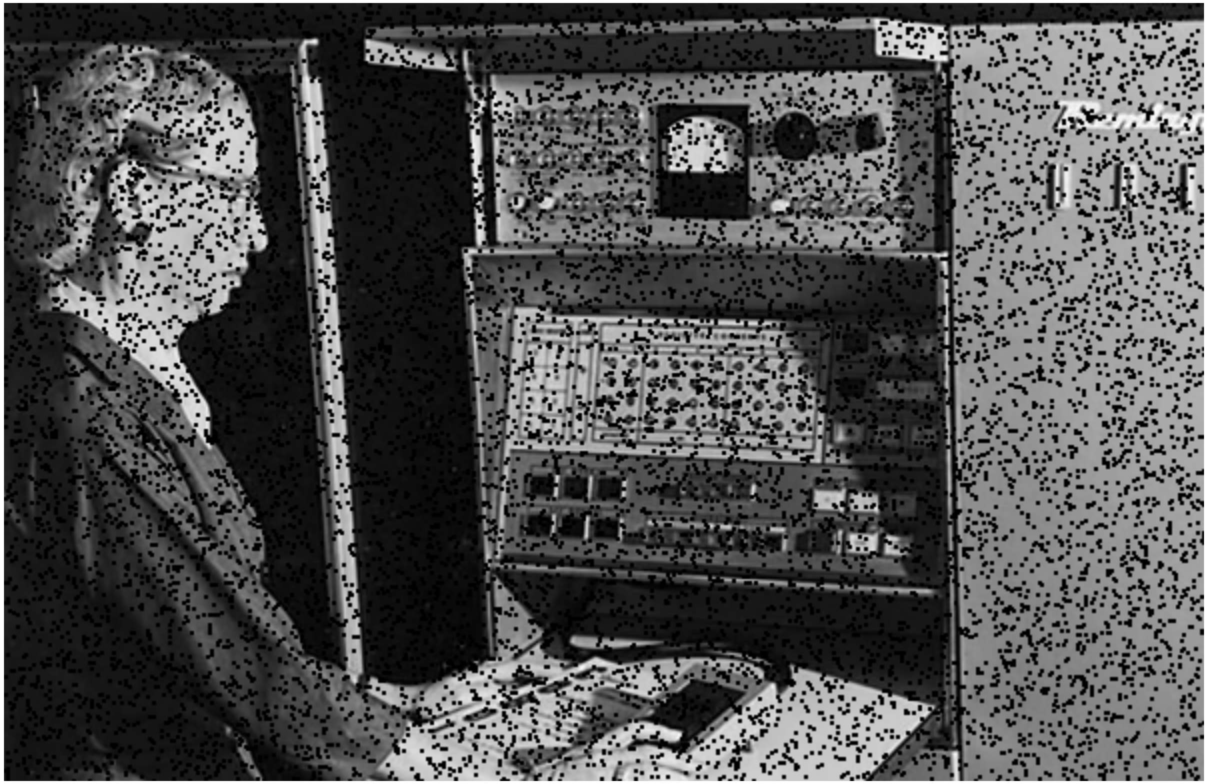
=

Answer



Filtering: Missing Data

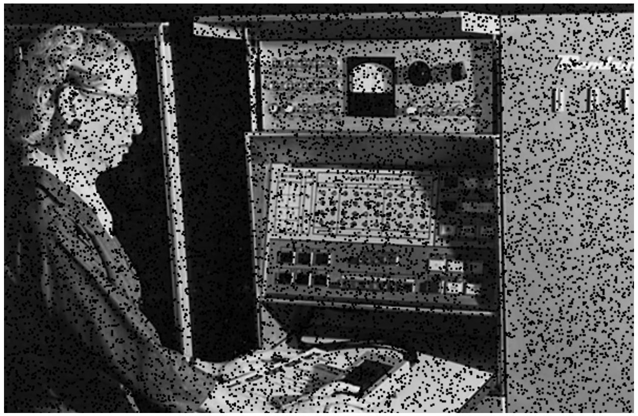
Oh no! Missing data!
(and we know where)



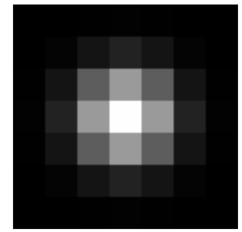
Common with many non-normal cameras (e.g., depth cameras)

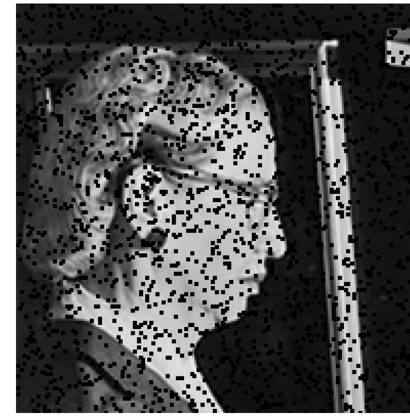
Filtering: Missing Data

Image



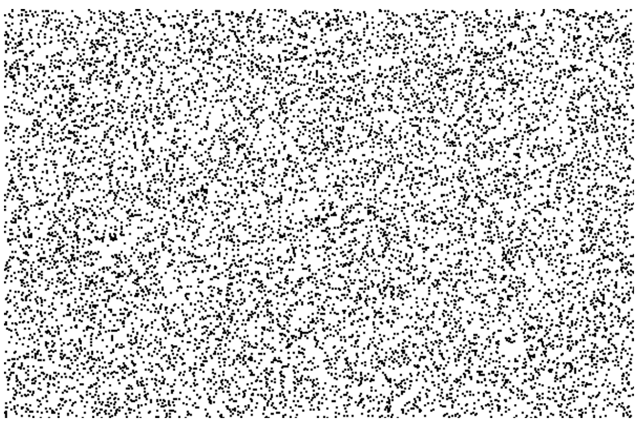
*



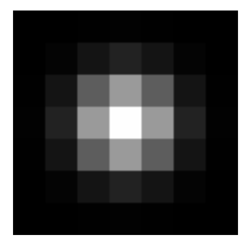


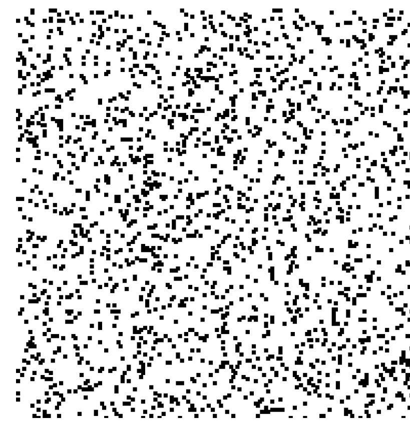
Per-element Division

Binary Mask



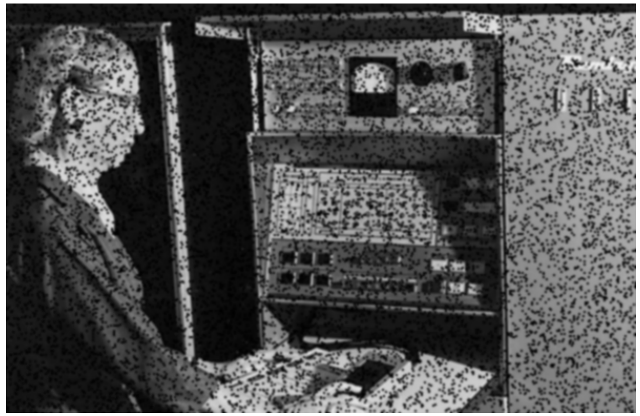
*

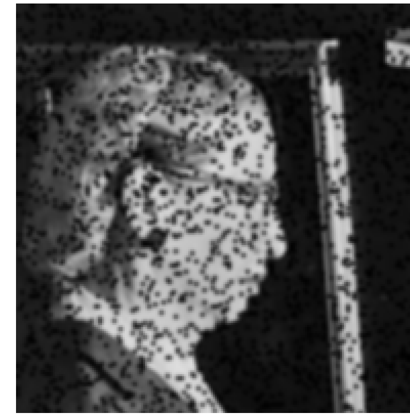




Filtering: Missing Data

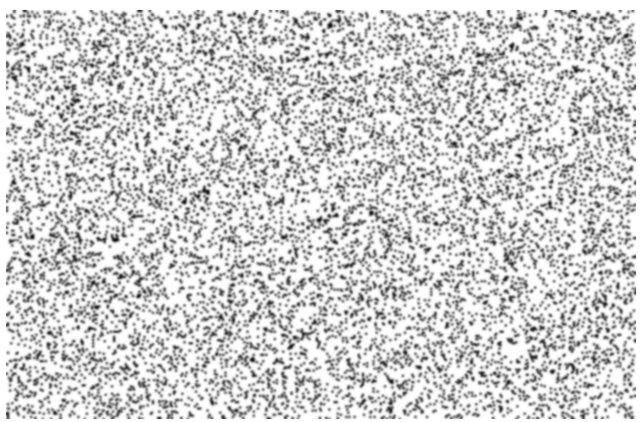
Image

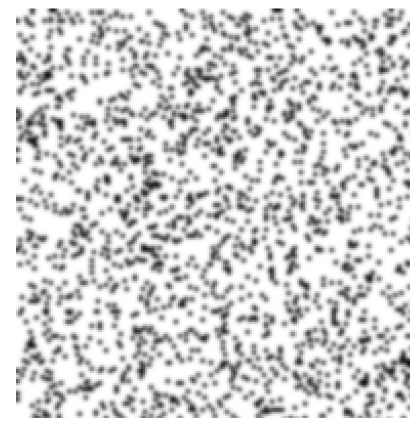




Per-element Division

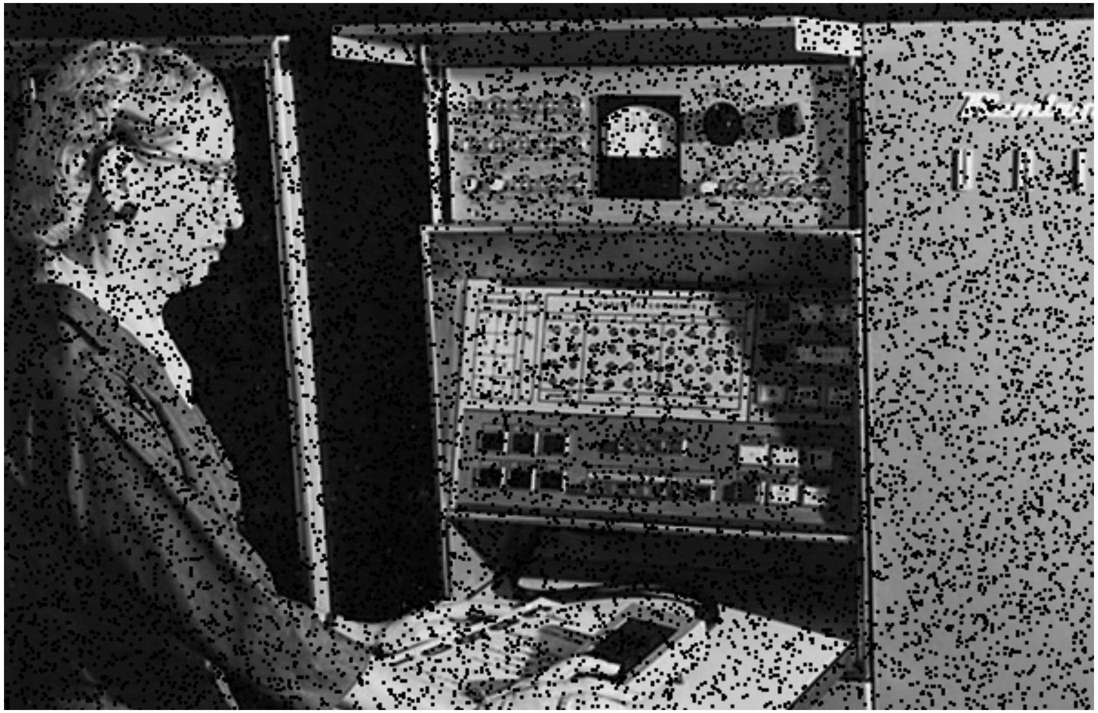
Binary Mask

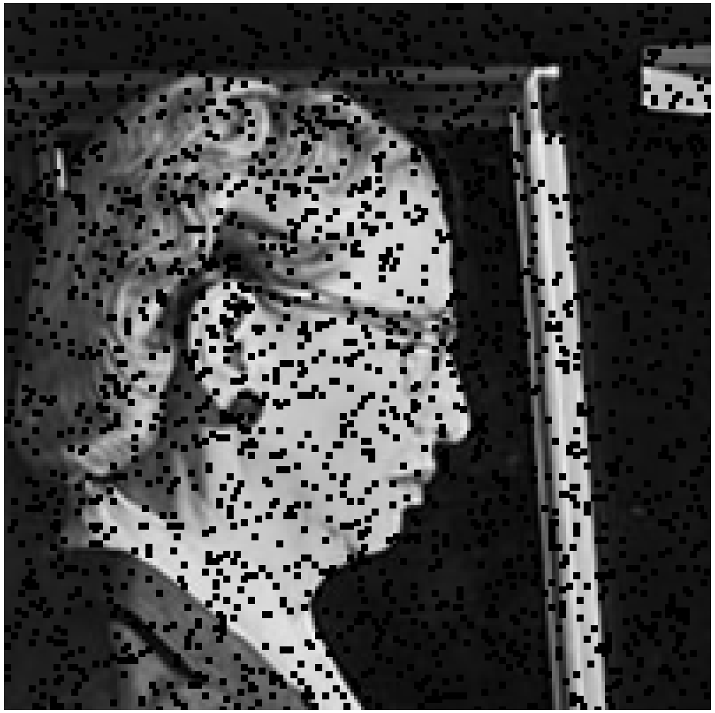




Filtering: Missing Data

Before





Filtering: Missing Data

After

Filtering: Missing Data

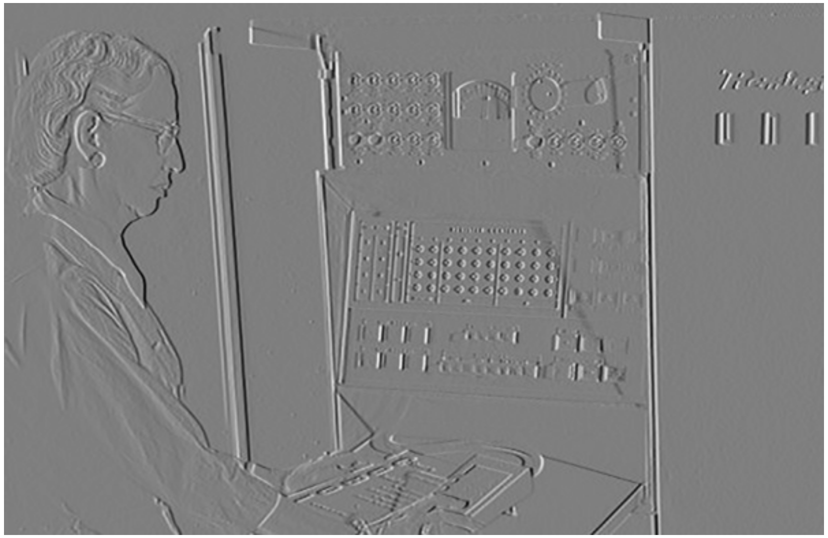
Original Image (No missing data)

Filtering

What's this Filter?

-1	0	1

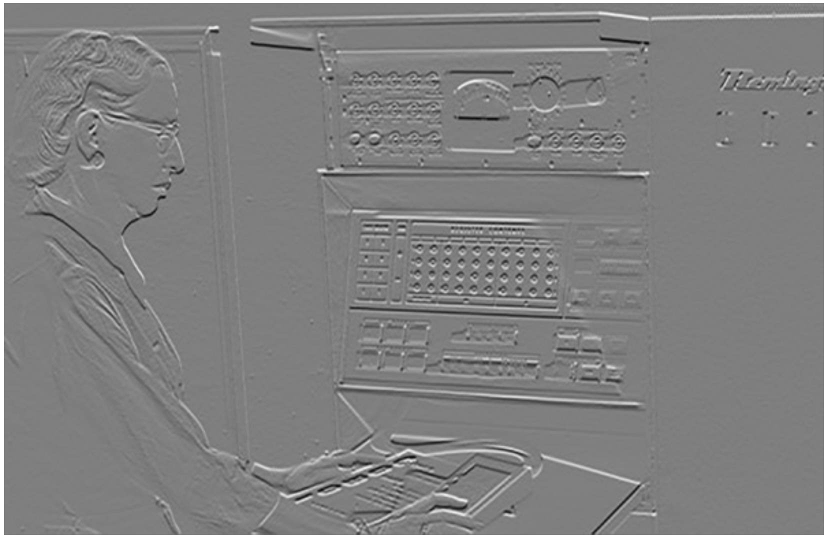
Derivative Dx



-1	0	1

T

Derivative Dy



Images as Functions

Image is function f(x,y)

Remember:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon, y) - f(x,y)}{\epsilon}$$

Approximate:

-1	1

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

Another one:

-1	0	1

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x-1,y)}{2}$$

Other Differentiation Operators

	Horizontal	Vertical
Prewitt	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$
Sobel	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Why might people use these compared to [-1,0,1]?

Images as Functions or Points

Key idea: can treat image as a point in $R^{(HxW)}$ or as a function of x,y.

$$\nabla I(x,y) = \begin{bmatrix} \dfrac{\partial I}{\partial x}(x,y) \\ \\ \dfrac{\partial I}{\partial y}(x,y) \end{bmatrix}$$

How much the intensity of the image changes as you go horizontally at (x,y)
(Often called Ix)

Image Gradient

Compute derivatives Ix and Iy with filters

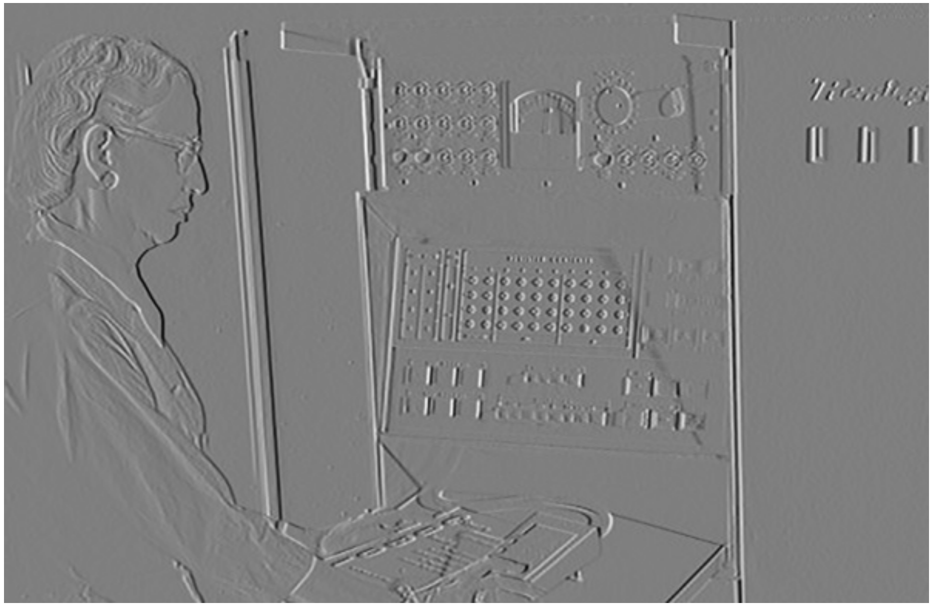
Ix

Iy

Image Gradient

Compute derivatives Ix and Iy with filters

Ix



Iy

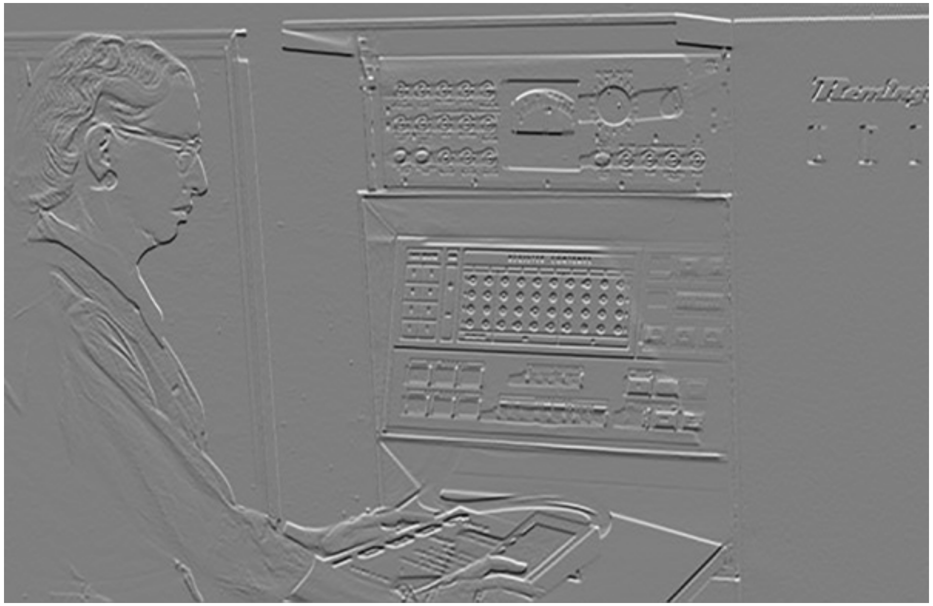


Image Gradient Magnitude

Gradient Magnitude $(Ix^2 + Iy^2)^{1/2}$

Gives rate of change at each pixel

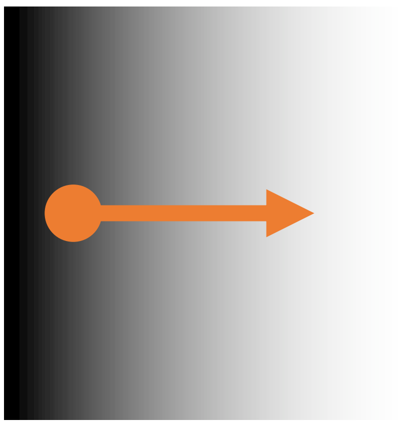
Image Gradient Magnitude

Gradient Magnitude $(Ix^2 + Iy^2)^{1/2}$

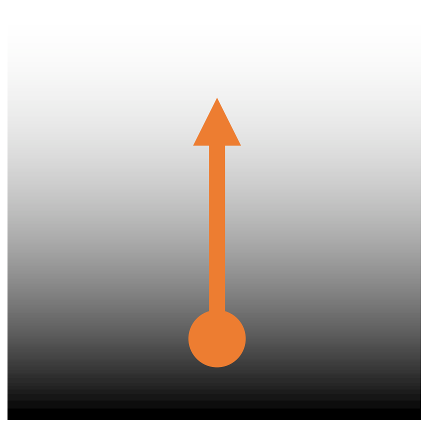
Gives rate of change at each pixel

Image Gradient Direction

Gradient Direction atan2(Ix, Iy)
Gives direction of change at each pixel



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

Figure Credit: S. Seitz

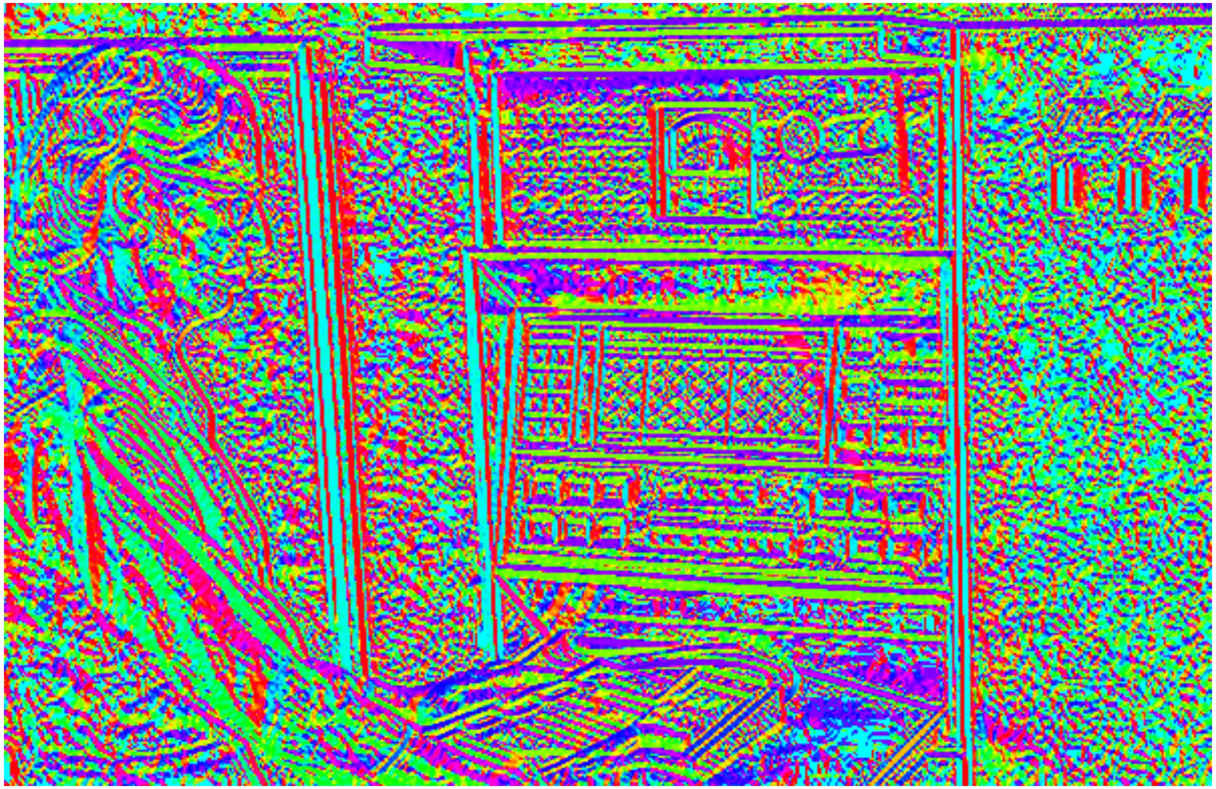
Image Gradient Direction

Gradient Direction atan2(Ix, Iy)

Gives direction of change at each pixel

Image Gradient Direction

Gradient Direction atan2(Ix, Iy)
Gives direction of change at each pixel



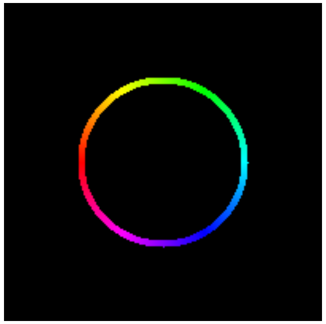
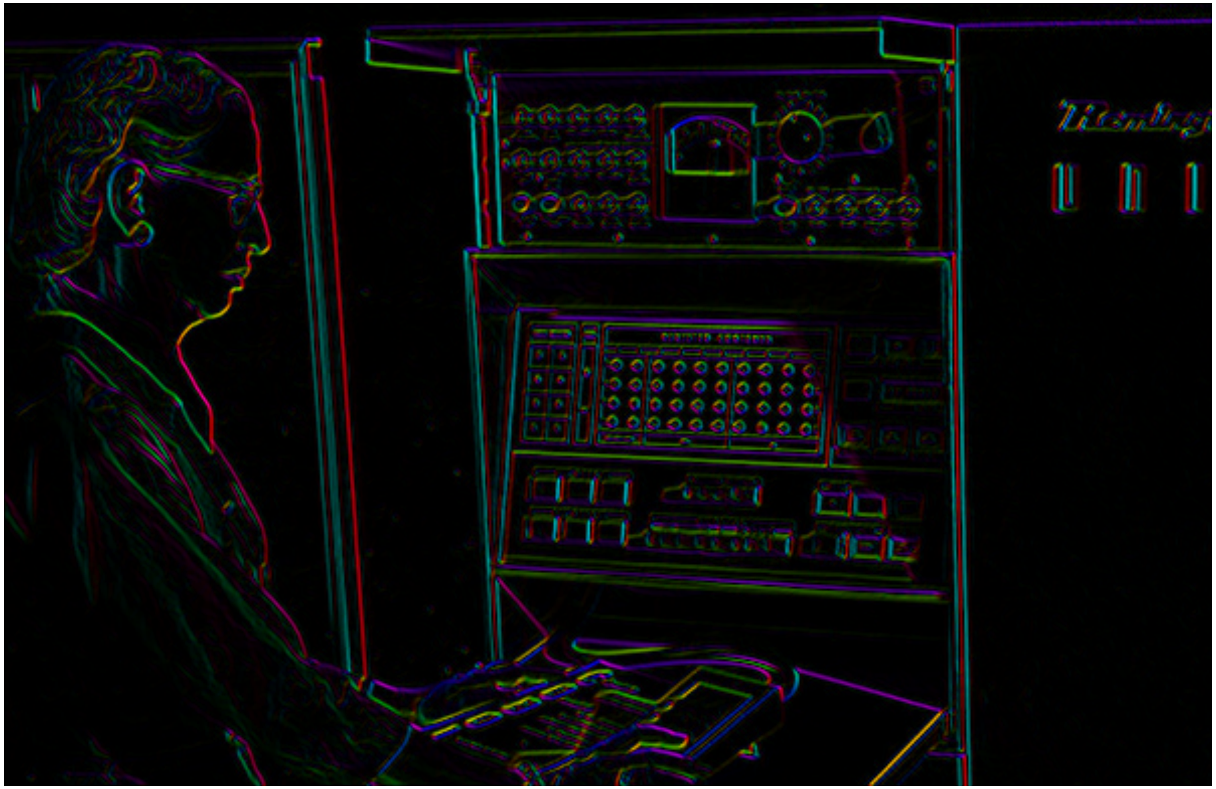
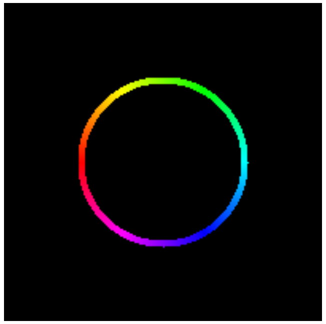


Image Gradient Direction

Gradient Direction atan2(Ix, Iy)
Gives direction of change at each pixel





I'm making the lightness equal to gradient magnitude

Next Time:
Edge + Corner Detection