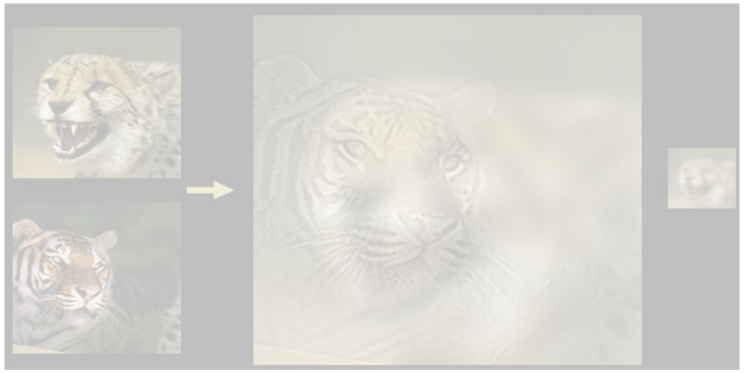
CS 4476 Computer Vision:

3. Image Filtering

Humphrey Shi

8/27/2024



The follow slides are mostly based on the lecture slides of Judy Hoffman, James Hays, and Svetlana Lazebnik, with credits to Devi Parikh, Kristen Grauman, Fei-Fei Li etc.

Outline

- Logistics & Course Recap
- Image Filtering

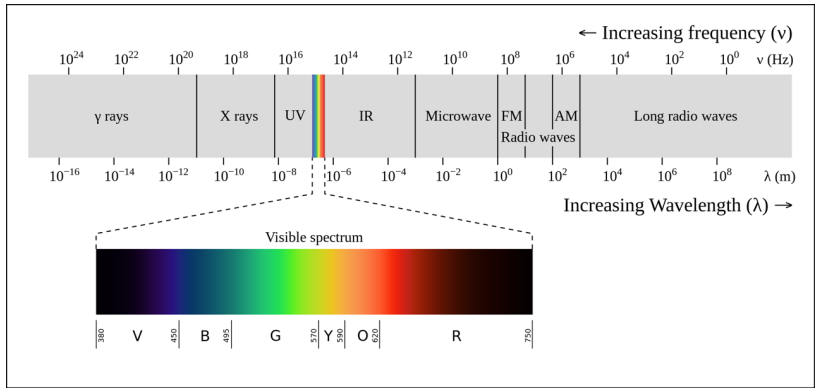
Logistics & Course Recap

Logistics

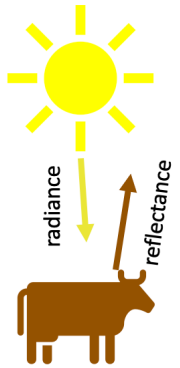
- Assignment 0 & 1 out on Canvas/GradeScope.
- TA office hours will be out this week.
- Claim your extra participation credit on Ed Discussion.

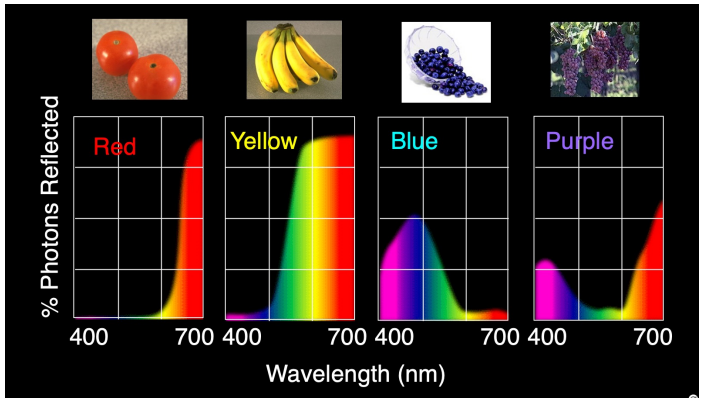
Recap: Light & Color

- **Color** is the result of interaction between physical **Light** in the environment and our visual system.

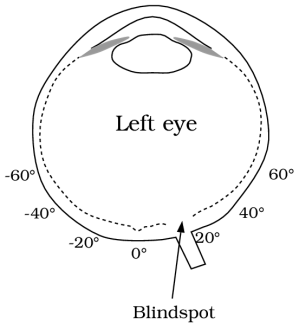


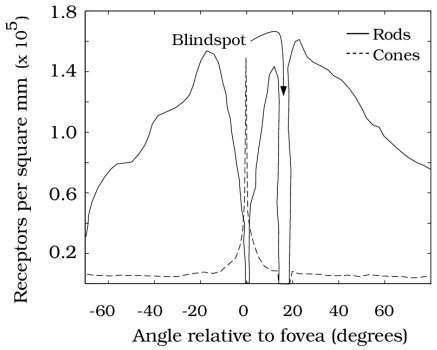
Visible light (400~700nm)

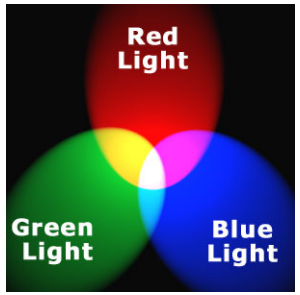


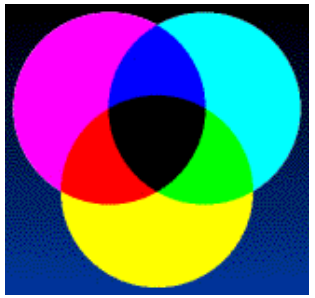


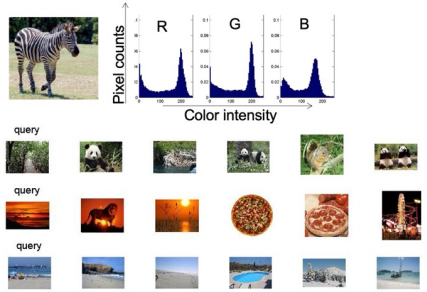
Light interacts with environments











Color perception is decided by light, environment and our visual system

Color mixing models

Color can be used in machine vision systems

Image Filtering

Digital Image Formation

(Review) What is in an Image?

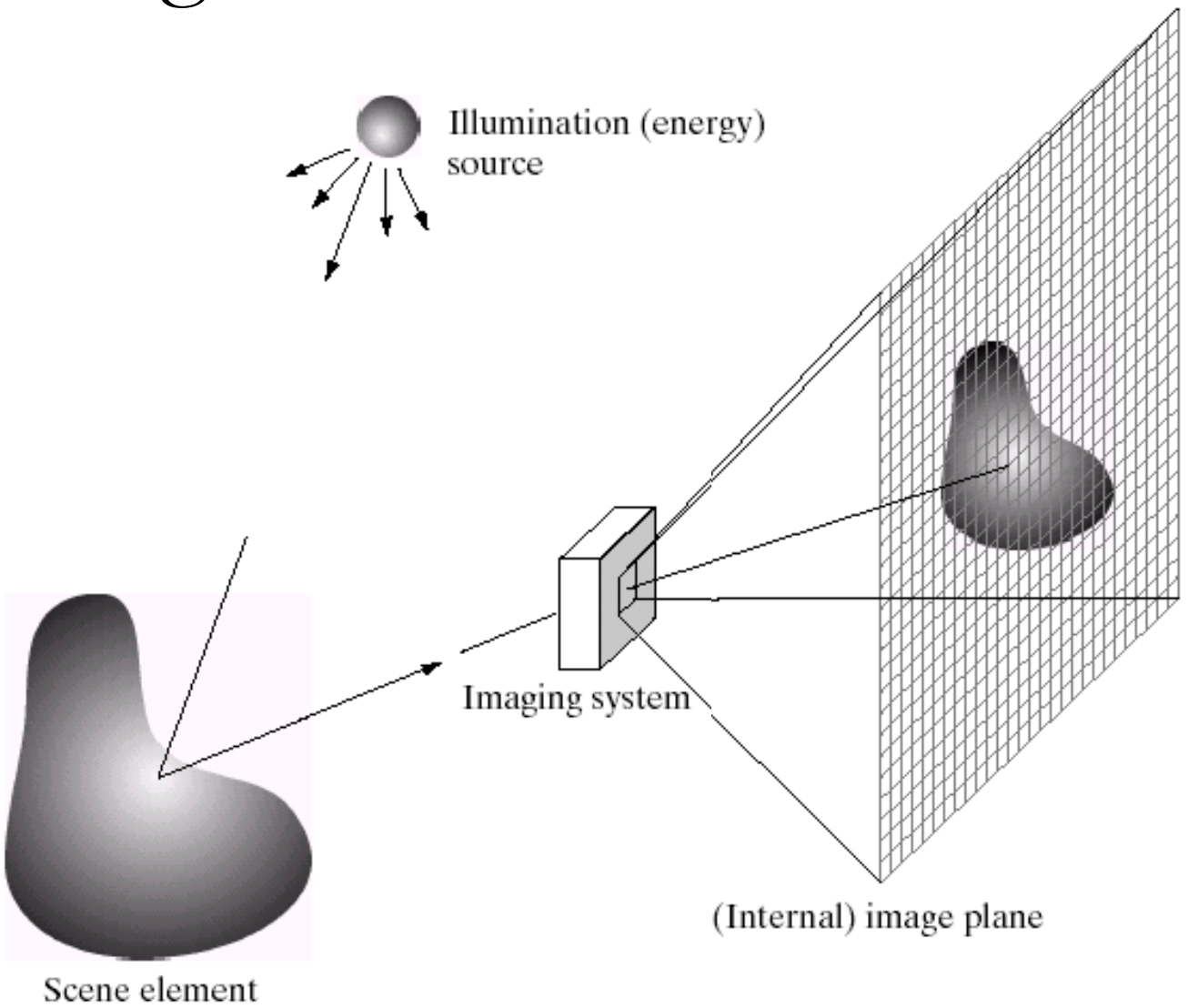

```python
from PIL import Image
import numpy as np

image = Image.open('./tech.jpg')
image_array = np.array(image)
print(image_array.shape)
print(image_array)
```

```
(1187, 949)
[[253 224 229 ... 216 216 216]
 [250 151 143 ... 112 112 112]
 [251 142 150 ... 112 112 112]
 ...
 [254  66  68 ...  80  80  80]
 [255 253 255 ... 254 254 254]
 [255 255 248 ... 255 255 255]]
```


```
(1187, 949, 3)
[[[249 254 255]
  [217 222 251]
  [224 226 255]
  ...
  [188 222 255]
  [188 222 255]
  [188 222 255]]

 [[247 251 255]
  [144 148 185]
  [135 138 193]
  ...
  [ 76 116 188]
  [ 74 116 188]
  [ 76 116 188]]

 [[248 251 255]
  [135 136 190]
  [142 142 214]
  ...
  [ 63 119 204]
  [ 62 120 204]
  [ 63 119 204]]

 ...

 [[254 254 254]
  [ 66  66  66]
  [ 68  68  68]
  ...
  [ 76  80  89]
  [ 74  81  87]
  [ 76  80  89]]

 [[255 255 255]
  [253 253 253]
  [255 255 255]
  ...
  [252 255 255]
  [252 255 255]
  [252 255 255]]

 [[255 255 255]
  [255 255 255]
  [248 248 248]
  ...
  [254 255 255]
  [254 255 255]
  [254 255 255]]]
```

- Images are made by arrays of numbers of brightness/colors.

How are Images Formed?



Slide credit: Derek Hoiem

Digital Camera

A digital camera replaces film with a sensor array

- Each cell in the array is light-sensitive diode that converts photons to electrons
- http://electronics.howstuffworks.com/digital-camera.htm

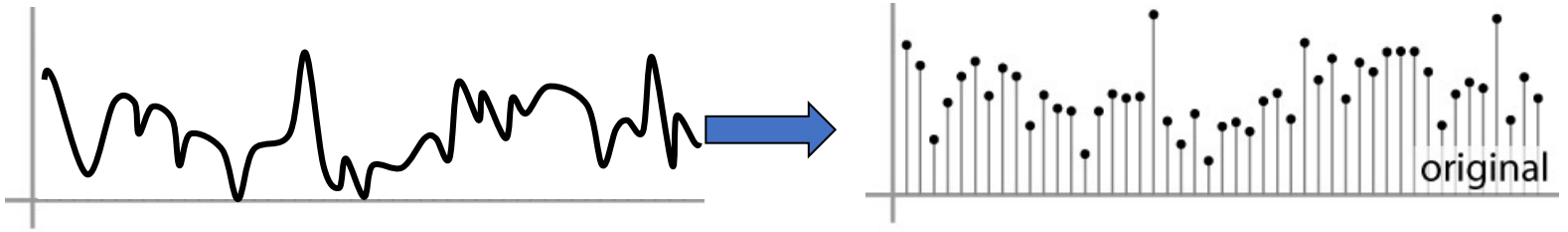
Image source: https://www.pinterest.se/pin/849984129648395414/

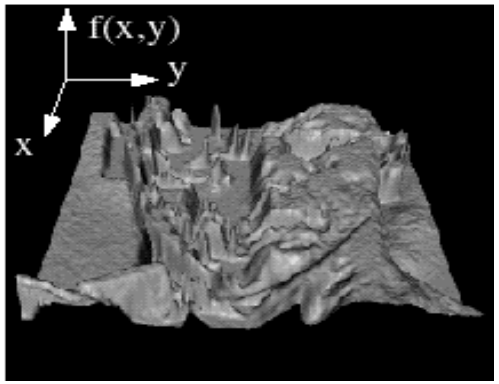
Slide credit: Steve Seitz

Digital Images

- **Sample** the 2D space on a regular grid
- **Quantize** each sample (round to nearest integer) Why?
- Image thus represented as a matrix of integer values.

Sampling in 1D

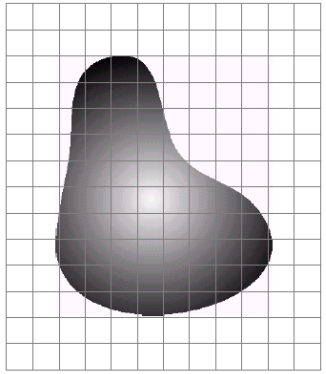




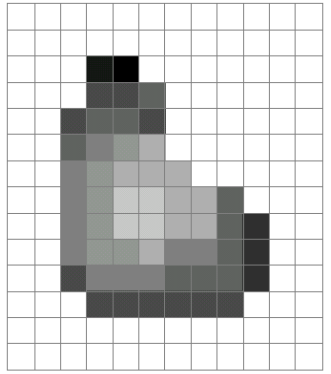
j

i

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30



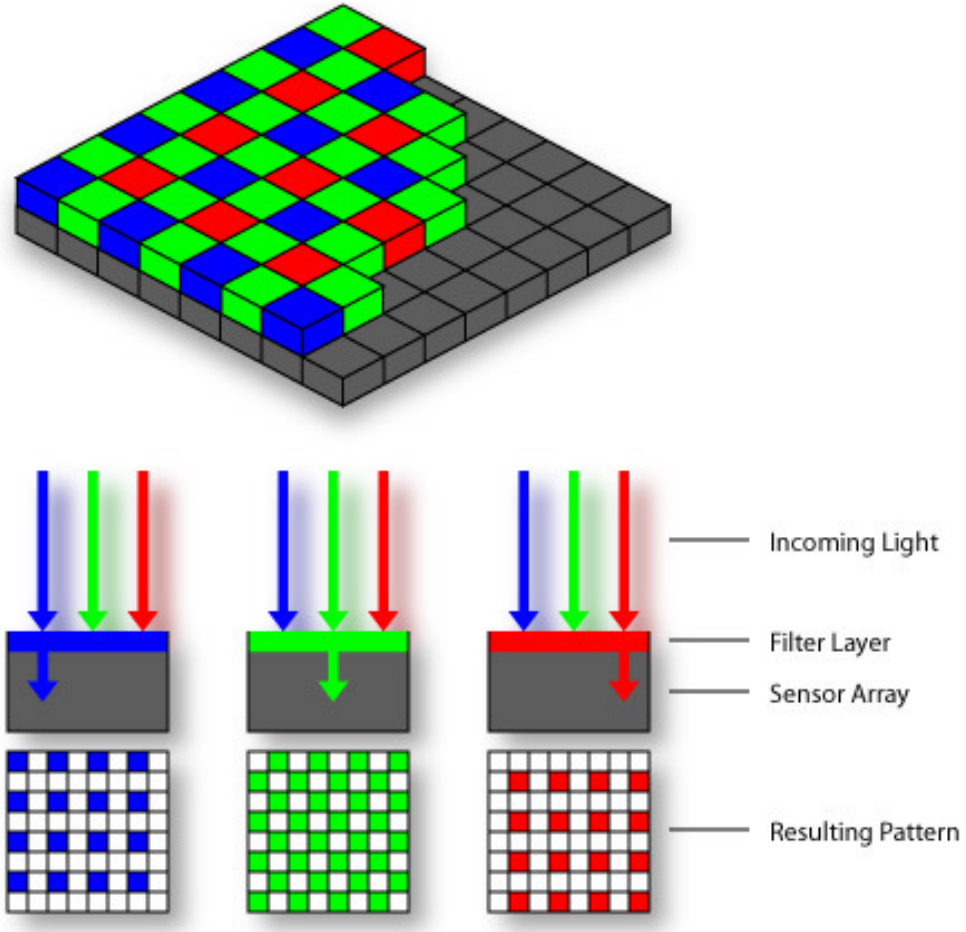
Continuous Image



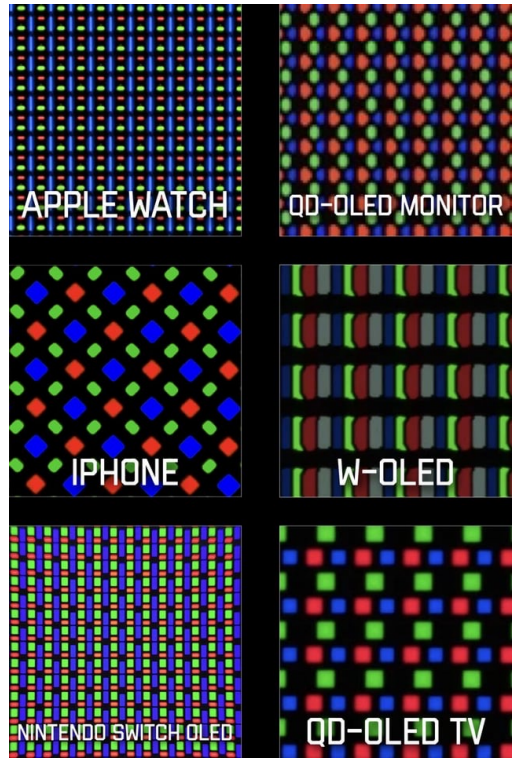
Quantized/Sampled Image

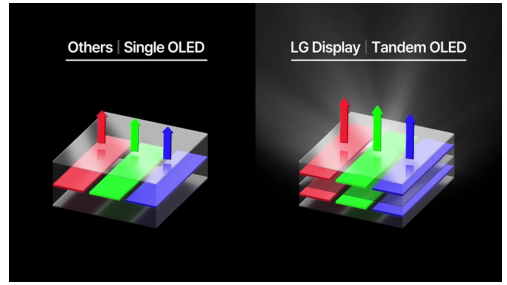
Slide credit: Kristen Grauman, Adapted from Steve Seitz

Digital Color Images



Estimate RGB
at ‘G’ cells from
neighboring values





Slide Credit: S. Seitz

Image credit, credit 2

Digital color images

Color images,
RGB color space

R

G

B

Slide credit: Kristen Grauman

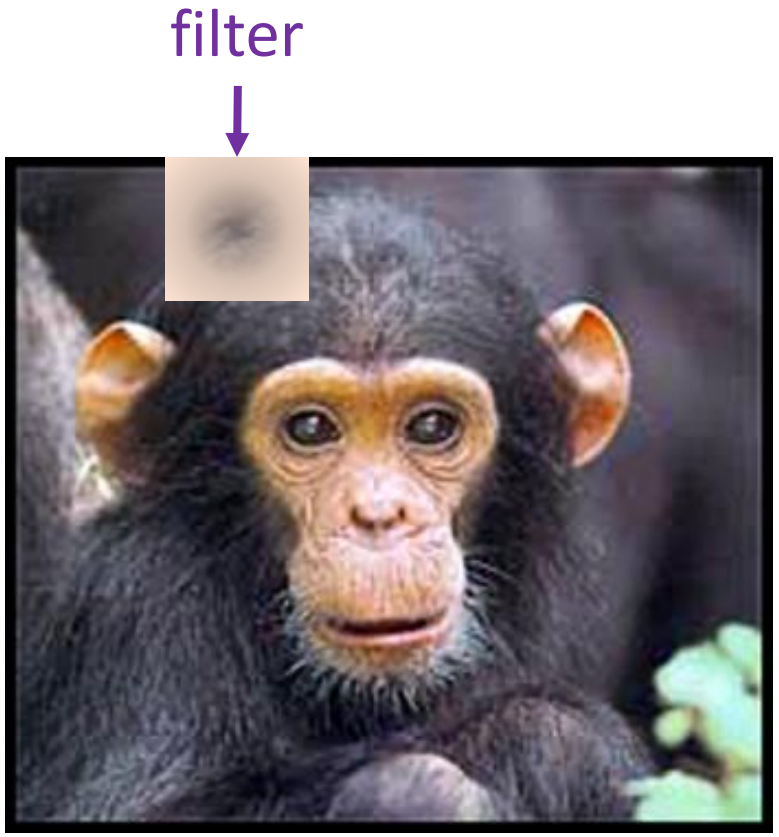
Image Filtering

Image Noise, Filtering and Convolution

Image Filtering

- **Idea**: Compute a function of the local neighborhood at each pixel in the image
  - Function specified by a "filter" or mask saying how to combine values from neighbors.

- Uses of filtering:
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (template matching)



Slide credit: Kristen Grauman, Adapted from Derek Hoiem

Motivation: Image Noise reduction

- A long-studied area in signal/image processing and computer vision.

1972

[j6] Grant B. Anderson, Thomas S. Huang:
Correction to "Piecewise Fourier Transformation for Picture Bandwidth Compression". IEEE Trans. Commun. 20(3): 488-492 (1972)

1971

[j5] Thomas S. Huang:
How the Fast Fourier Transform Got its Name. Computer 4(3): 15 (1971)

[j4] Grant B. Anderson, Thomas S. Huang:
Frequency-domain image errors. Pattern Recognit. 3(2): 185-192 (1971)

1970

[j3] Thomas S. Huang:
Introduction to the special issue on image enhancement. Pattern Recognit. 2(2): 67-68 (1970)

[–] **1960 – 1969**

1969

[c1] Grant B. Anderson, Thomas S. Huang:
Errors in frequency-domain processing of images. AFIPS Spring Joint Computing Conference 1969: 173-185

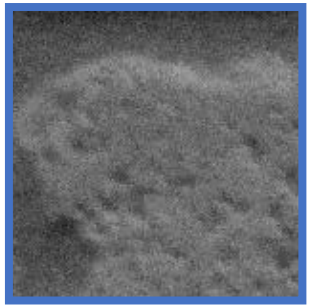
1967

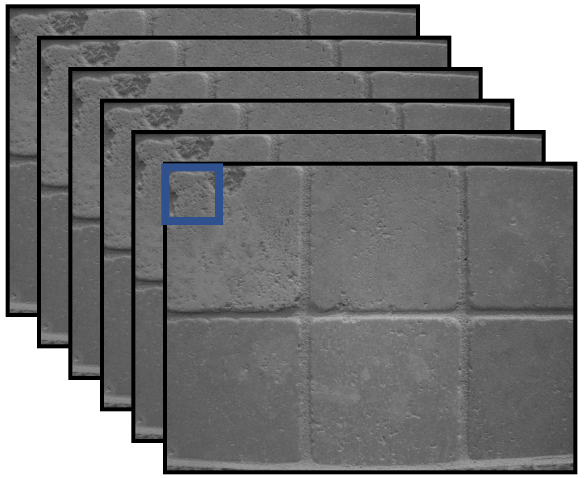
[j2] Thomas S. Huang, Mohamed Chikhaoui:
The effect of BSC on PCM picture quality. IEEE Trans. Inf. Theory 13(2): 270-273 (1967)

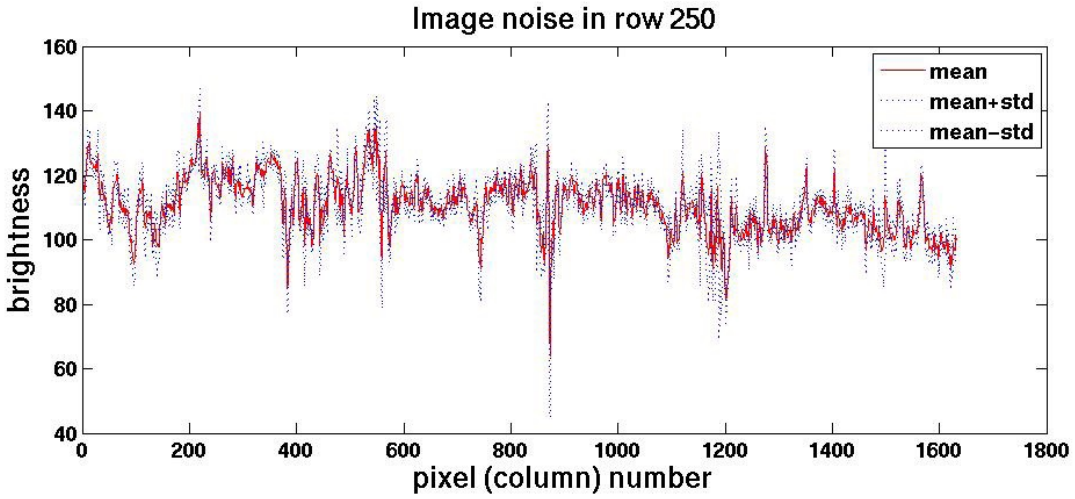
1965

[j1] Thomas S. Huang:
The subjective effect of two-dimensional pictorial noise. IEEE Trans. Inf. Theory 11(1): 43-53 (1965)

Motivation: Image Noise Reduction







- Even multiple images of the **same static scene** will not be identical.

Slide credit: Adapted from Kristen Grauman

Common Types of Noise

- **Salt and pepper noise:** random occurrences of black and white pixels
- **Impulse noise:** random occurrences of white pixels
- **Gaussian noise:** variations in intensity drawn from a Gaussian normal distribution

Original

Salt and pepper noise

Impulse noise

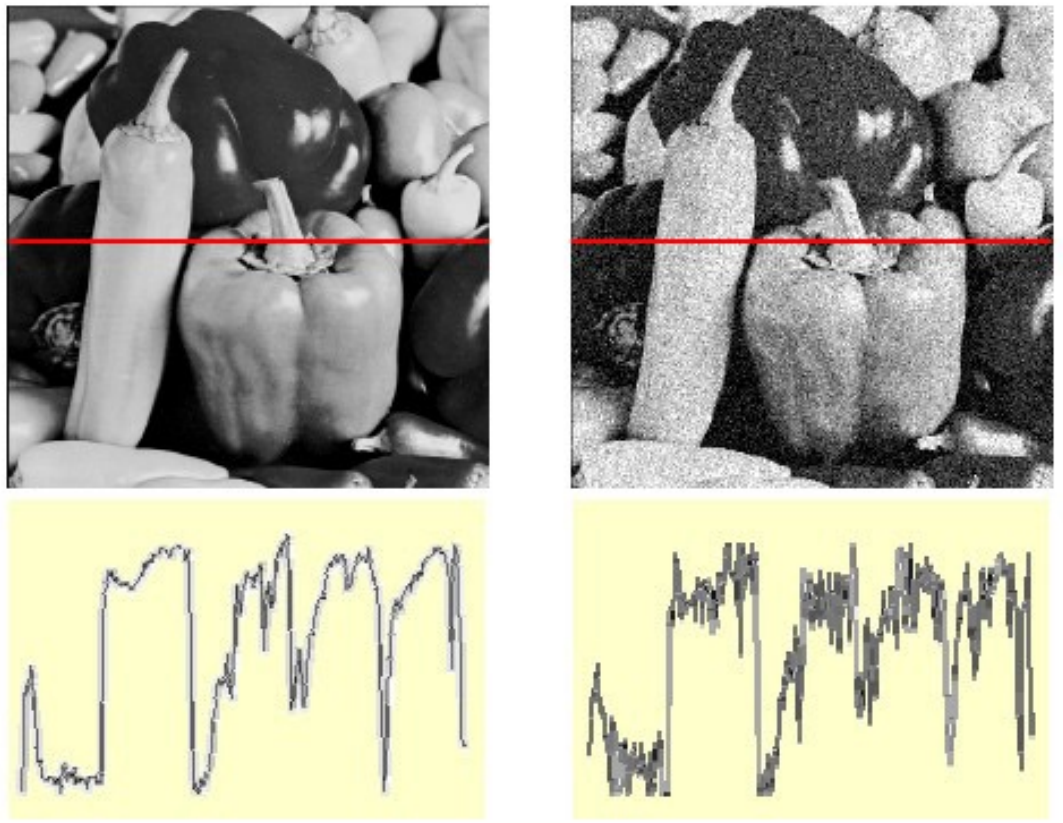
Gaussian noise

Slide credit: Steve Seitz

Gaussian noise

```
>> noise = randn(size(im)).*sigma;
>> output = im + noise;
```

Image
Noise



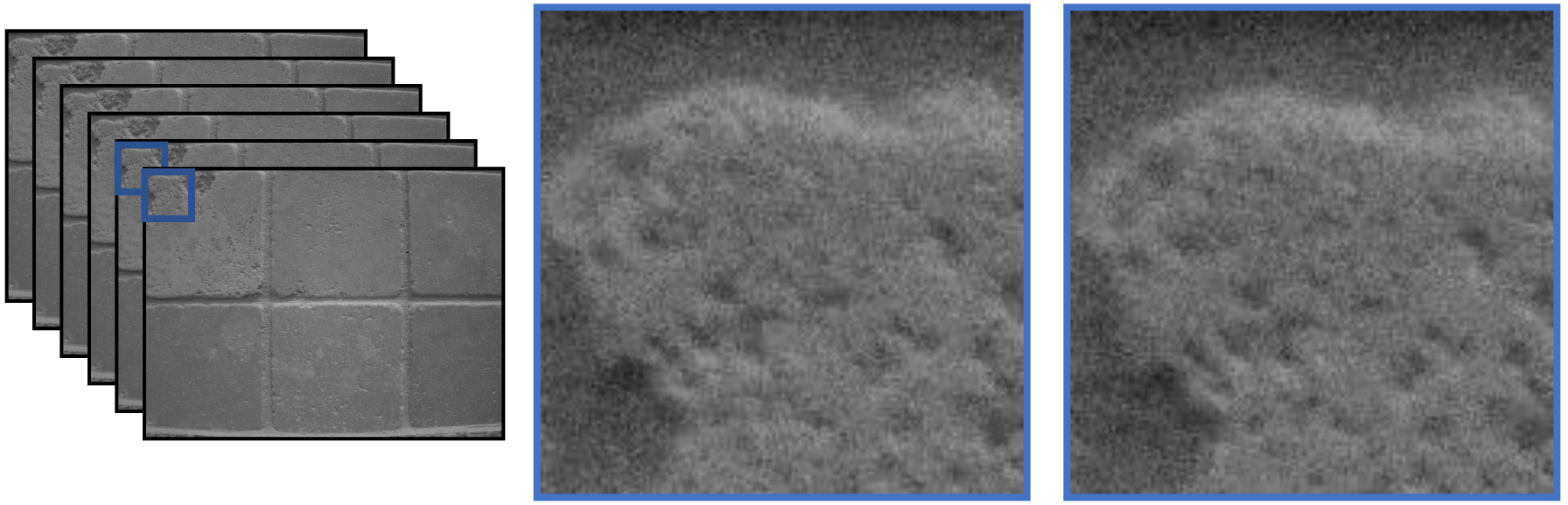
$$f(x,y) = \overbrace{\bar{f}(x,y)}^{\text{Ideal Image}} + \overbrace{\eta(x,y)}^{\text{Noise process}}$$

Gaussian i.i.d. ("white") noise:
$\eta(x,y) \sim \mathcal{N}(\mu,\sigma)$

What is impact of the sigma?

Slide credit: Kristen Grauman
Figure from Martial Hebert

Motivation: Image Noise Reduction



- Even multiple images of the same static scene will not be identical.
- How could we reduce the noise?
  - i.e., give an estimate of the true intensities
- **What if there's only one image?**

Slide credit: Kristen Grauman

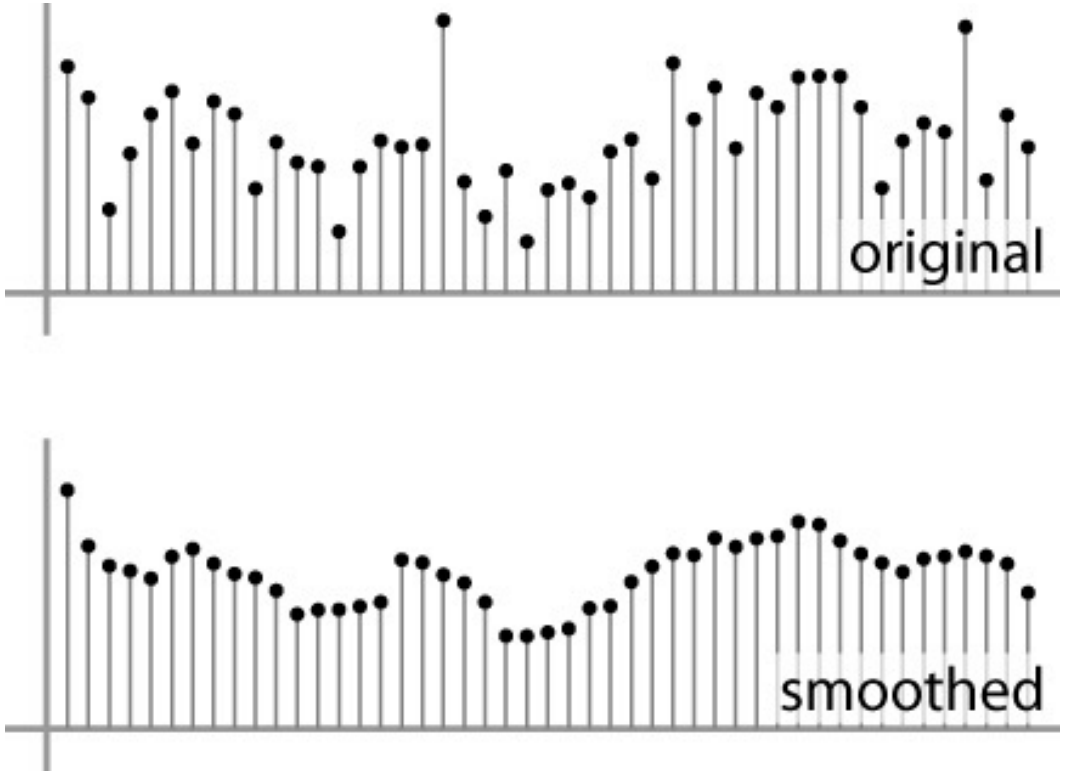
First Attempt at a Solution

- Let's replace each pixel with an average of all the values in its neighborhood
- Assumptions:
  - Expect pixels to be like their neighbors
  - Expect noise processes to be independent from pixel to pixel

Slide credit: Kristen Grauman

First Attempt at a Solution

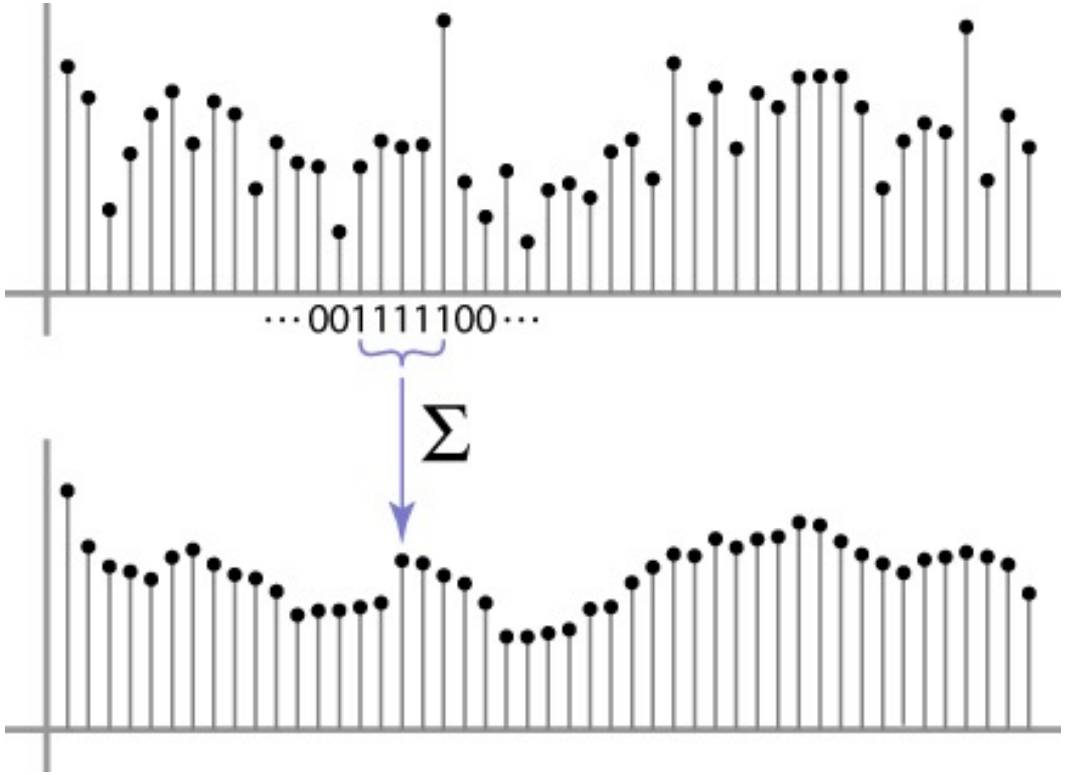
- Let's replace each pixel with an average of all the values in its neighborhood
- Moving average in 1D:



Slide credit: S. Marschner

Weighted Moving Average

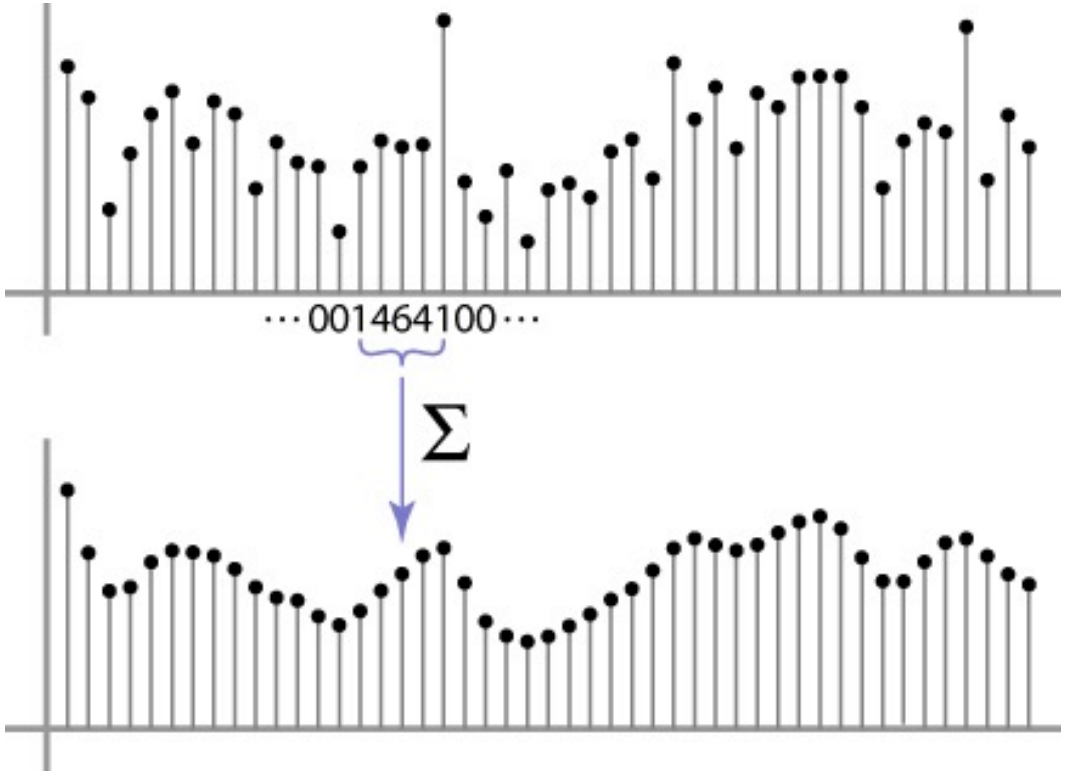
- Can add weights to our moving average
- *Weights* [1, 1, 1, 1, 1] / 5



Slide credit: S. Marschner

Weighted Moving Average

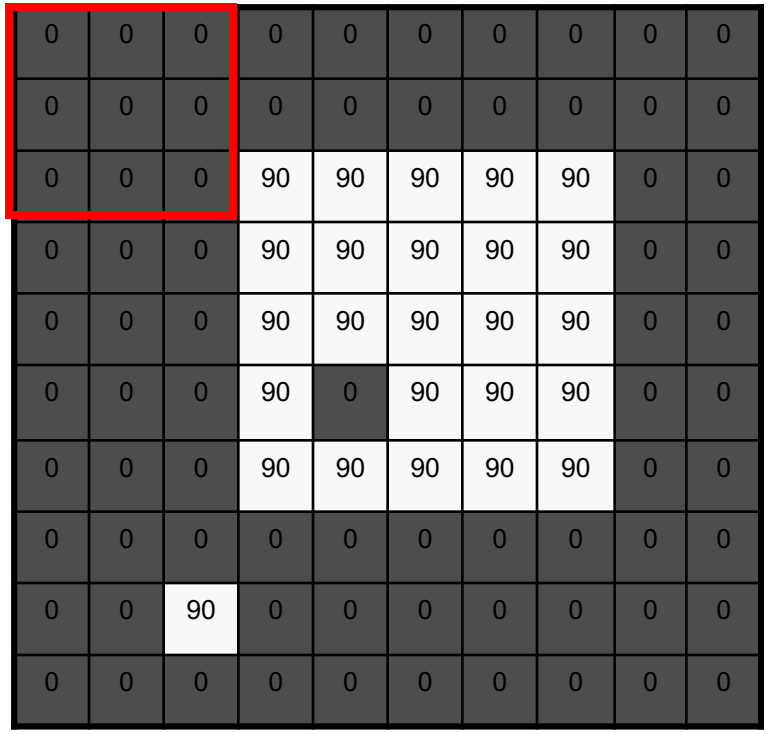
- Non-uniform weights [1, 4, 6, 4, 1] / 16



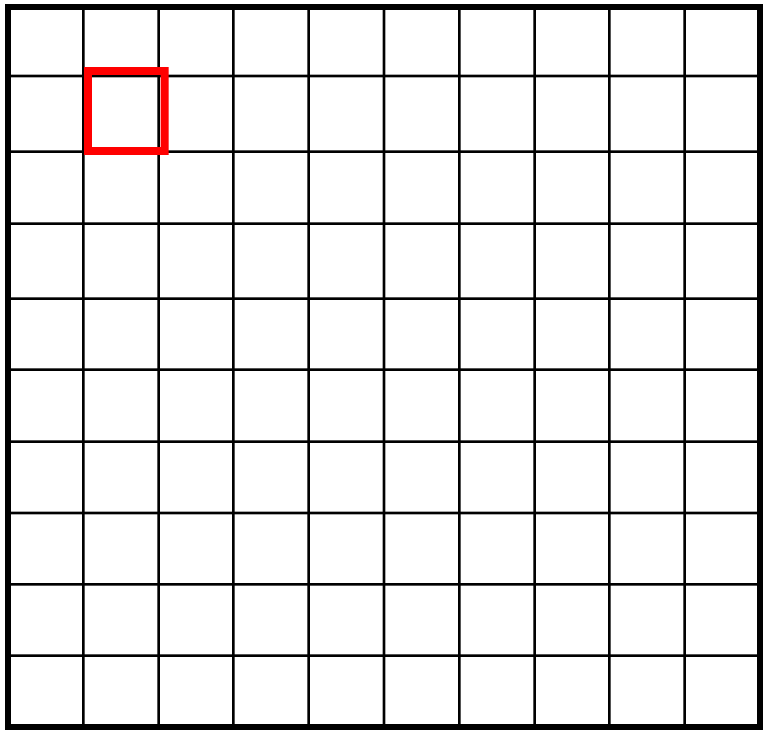
Slide credit: S. Marschner

Moving Average In 2D

$F[x, y]$



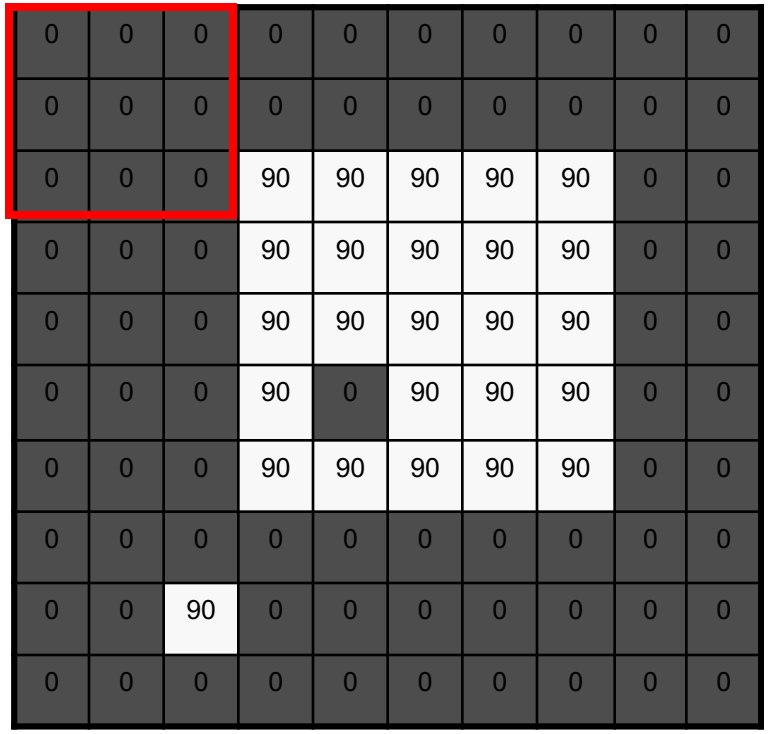
$G[x, y]$



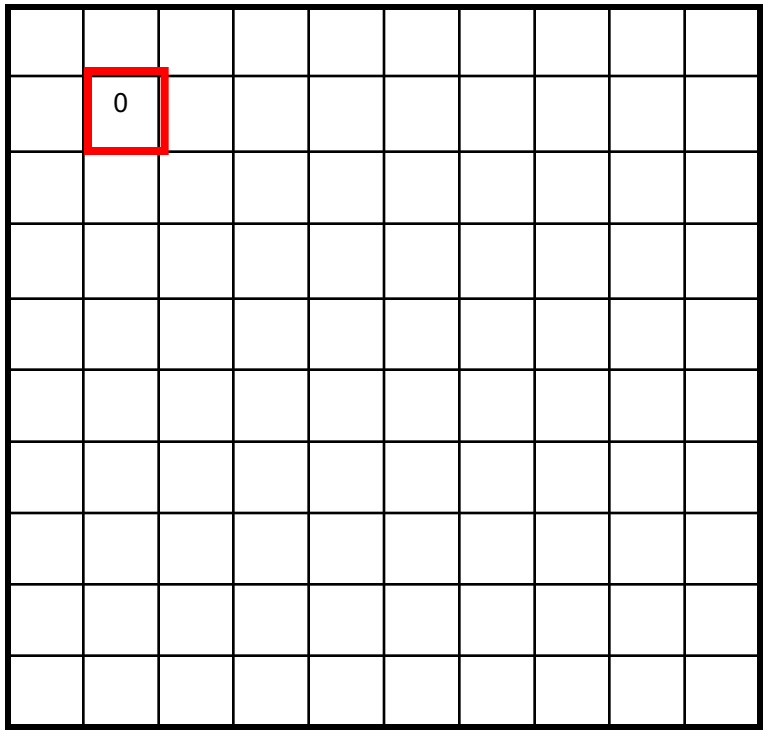
Slide credit: Steve Seitz

Moving Average In 2D

$F[x, y]$



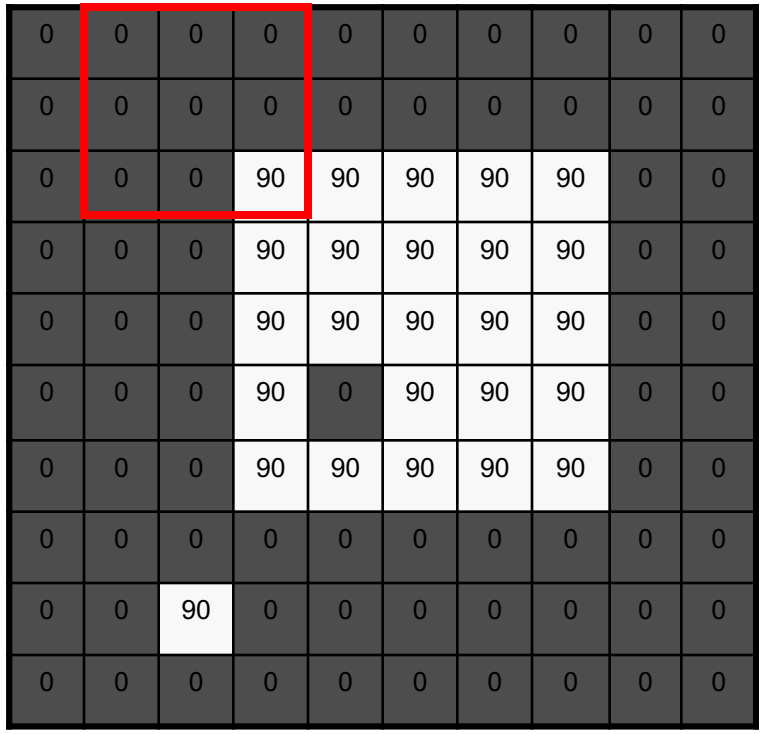
$G[x, y]$



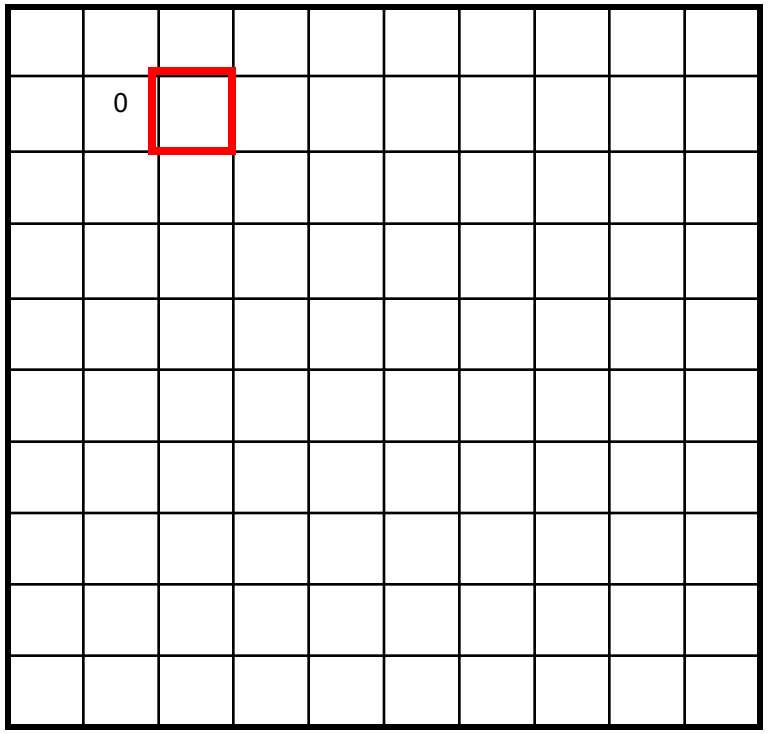
Slide credit: Steve Seitz

Moving Average In 2D

$F[x, y]$



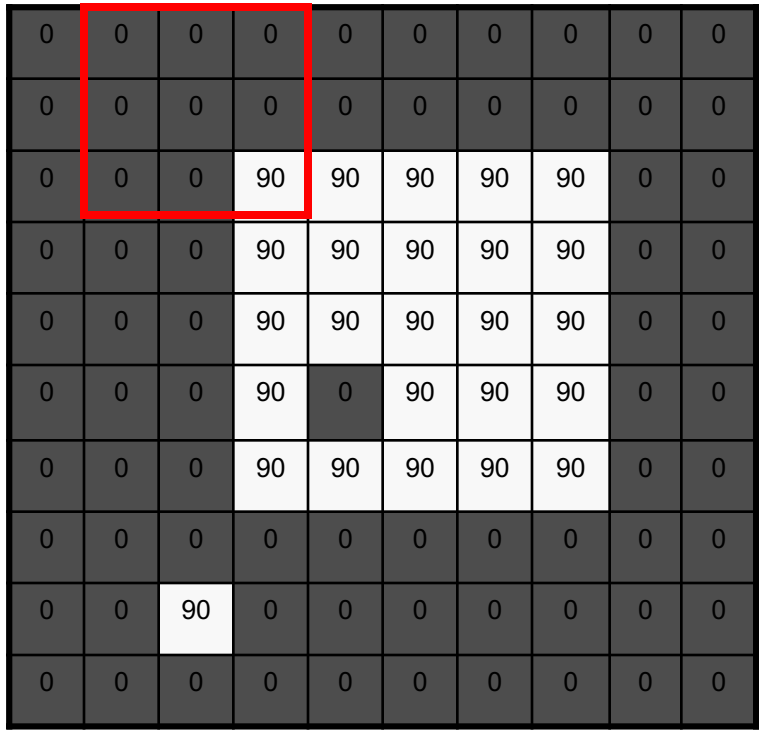
$G[x, y]$



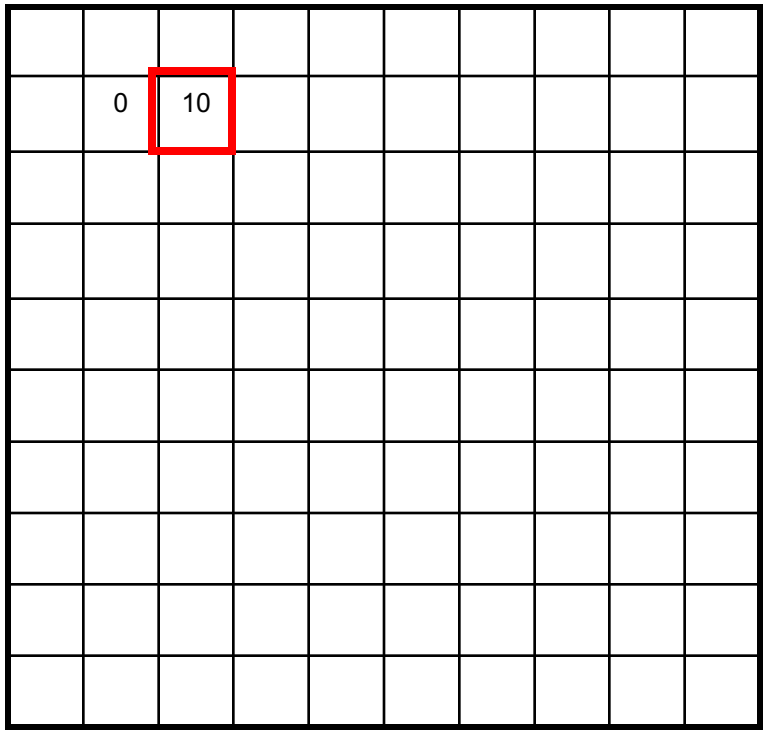
Slide credit: Steve Seitz

Moving Average In 2D

$F[x, y]$



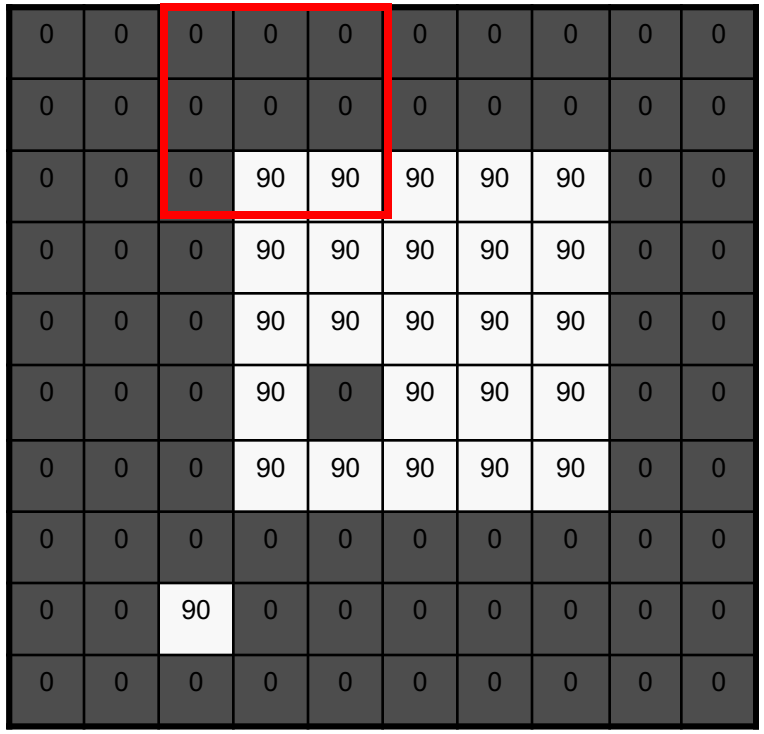
$G[x, y]$



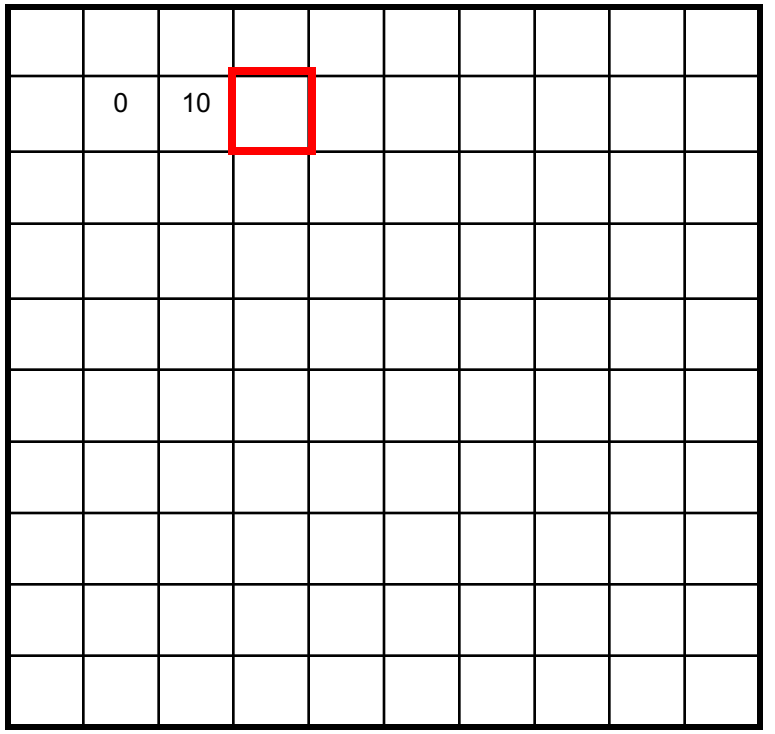
Slide credit: Steve Seitz

Moving Average In 2D

$F[x, y]$



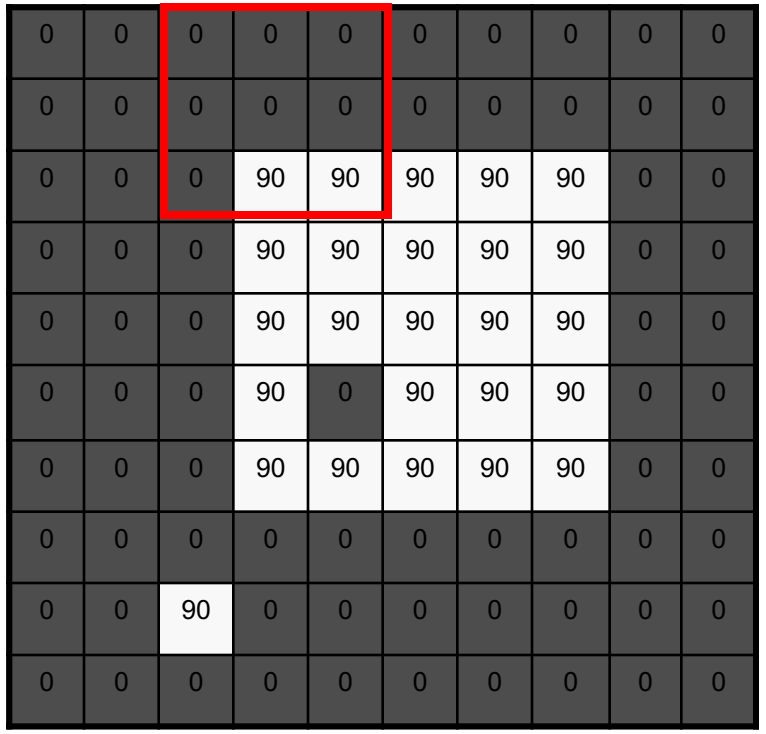
$G[x, y]$



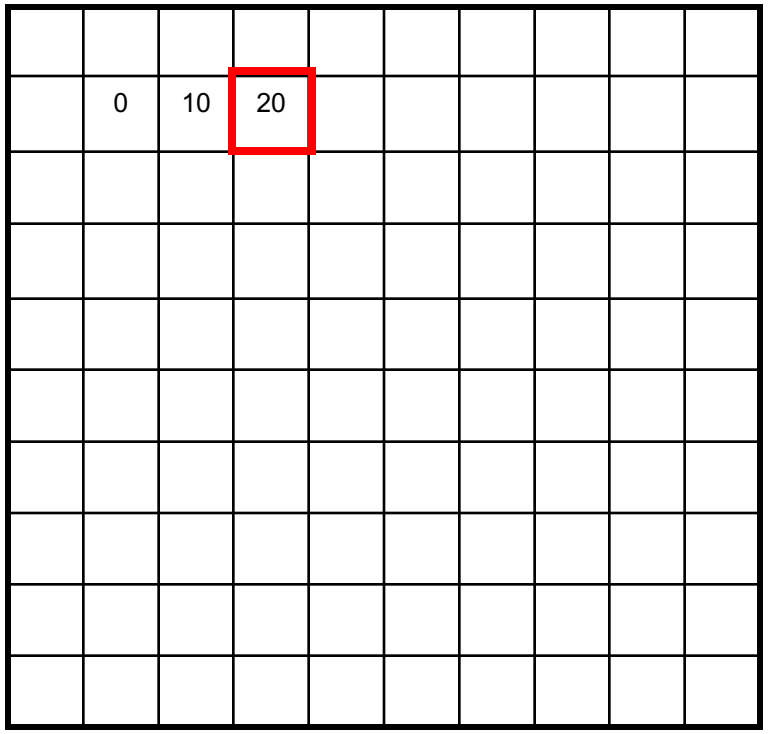
Slide credit: Steve Seitz

Moving Average In 2D

$F[x, y]$



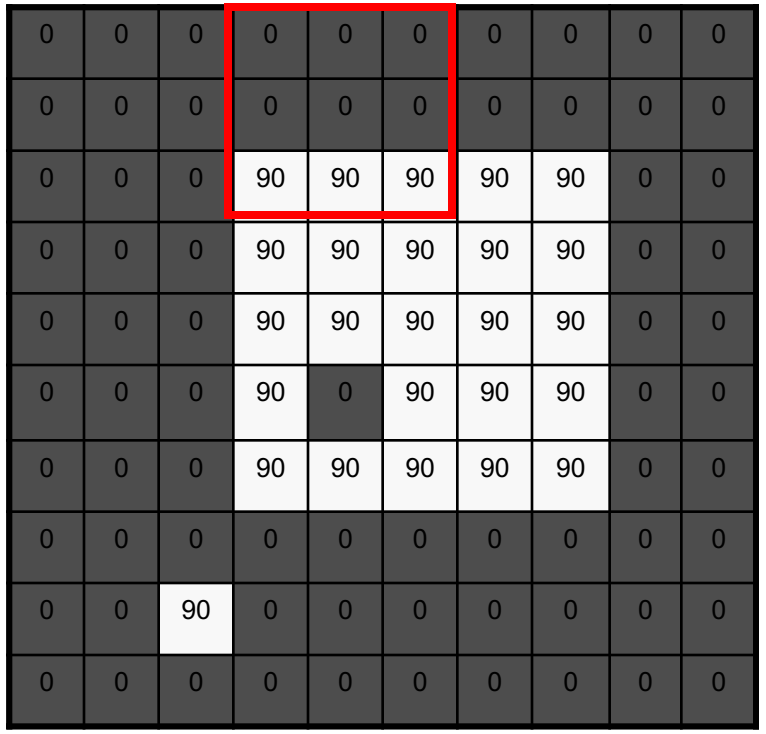
$G[x, y]$



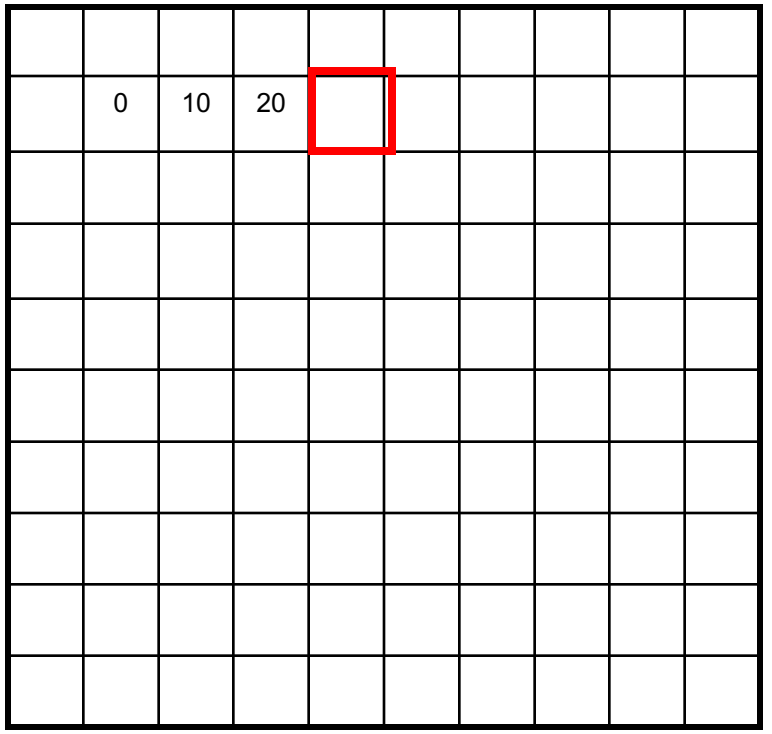
Slide credit: Steve Seitz

Moving Average In 2D

$F[x, y]$



$G[x, y]$



Slide credit: Steve Seitz

Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30					

Slide credit: Steve Seitz

Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30					

Slide credit: Steve Seitz

Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30	30				

Slide credit: Steve Seitz

Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Slide credit: Steve Seitz

Correlation Filtering

Say the averaging window size is 2k+1 x 2k+1:

$$G[i,j] = \underbrace{\frac{1}{(2k+1)^2}}_{\text{Attribute uniform weight to each pixel}} \underbrace{\sum_{u=-k}^{k}\sum_{v=-k}^{k} F[i+u, j+v]}_{\text{Loop over all pixels in neighborhood around image pixel F[i,j]}}$$

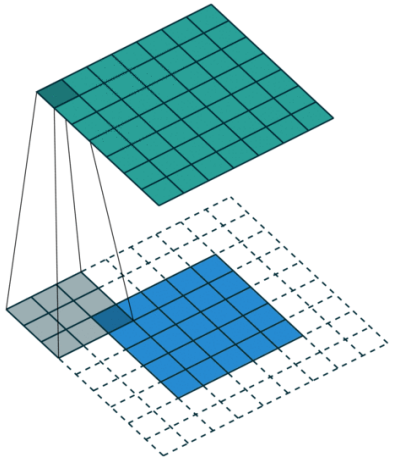
Now generalize to allow different weights depending on neighboring pixel's relative position:

$$G[i,j] = \sum_{u=-k}^{k}\sum_{v=-k}^{k} \underbrace{H[u,v]}_{\text{Non-uniform weights}} F[i+u, j+v]$$

Slide credit: Kristen Grauman

Correlation Filtering

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



This is called cross-correlation, denoted $G = H \otimes F$

Filtering an image: replace each pixel with a linear combination of its neighbors.

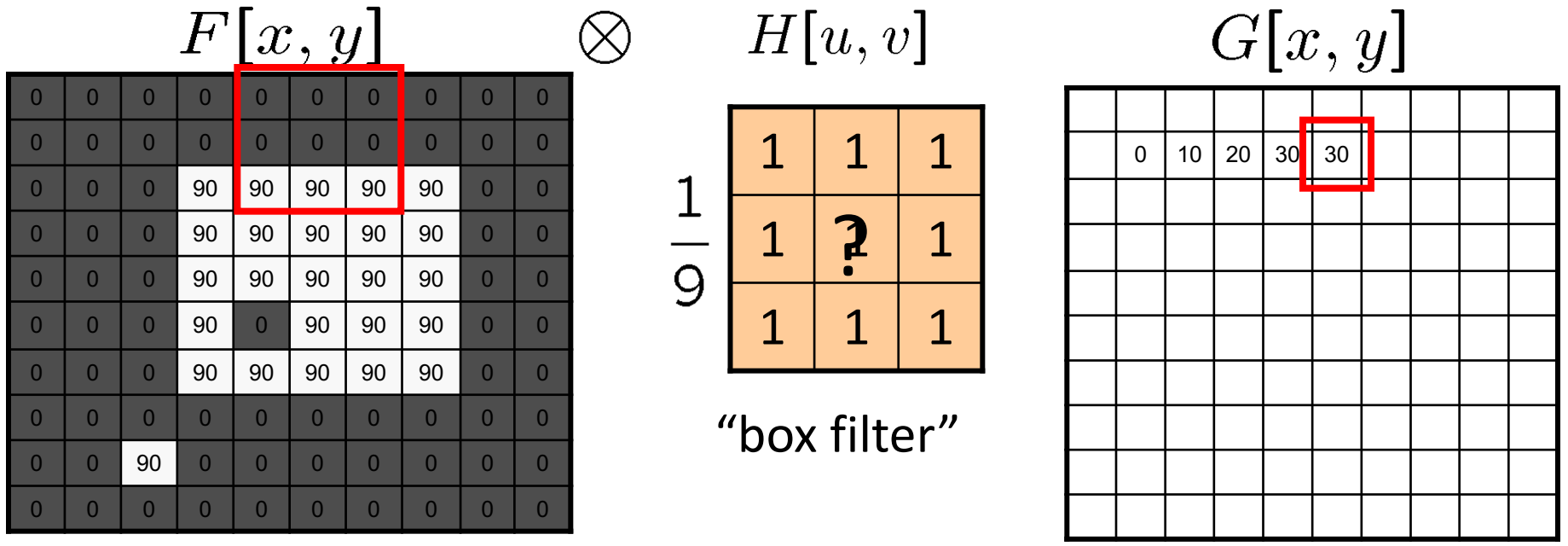
The filter "kernel" or "mask" *H*[*u,v*] is the prescription for the weights in the linear combination.

Slide credit: Judy Hoffman adapted from Kristen Grauman

https://towardsdatascience.com/convolution-vs-correlation-af868b6b4fb5

Averaging filter

- What values belong in the kernel H for the moving average example?



Talk to your neighbor

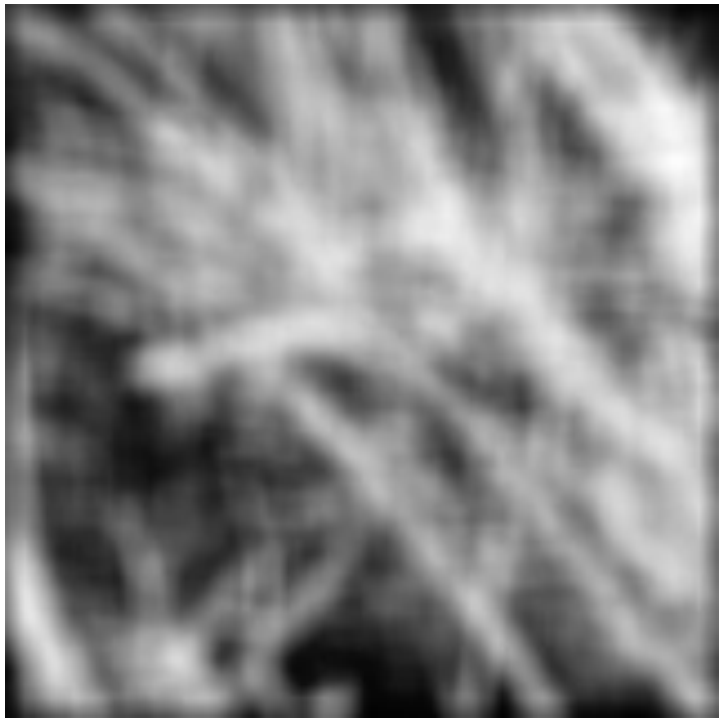
$$G = H \otimes F$$

Slide credit: Kristen Grauman

Smoothing by Averaging

depicts box filter:
white = high value, black = low value

original



filtered

What if the filter size was 5 x 5 instead of 3 x 3?

Slide credit: Kristen Grauman

Gaussian Filter

- What if we want nearest neighboring pixels to have the most influence on the output?

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x, y]$

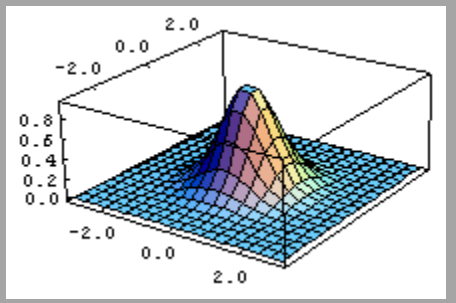
$$\frac{1}{16}$$

1	2	1
2	4	2
1	2	1

$H[u, v]$

This kernel is an approximation of a 2d Gaussian function:

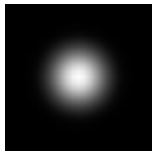
$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$

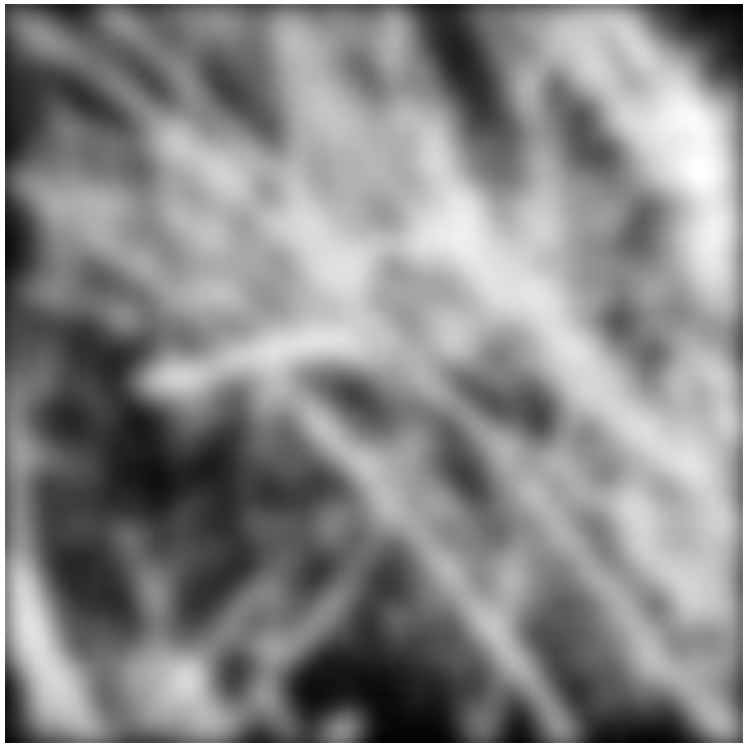


- Removes high-frequency components from the image ("low-pass filter").

Slide credit: Steve Seitz

Smoothing with a Gaussian

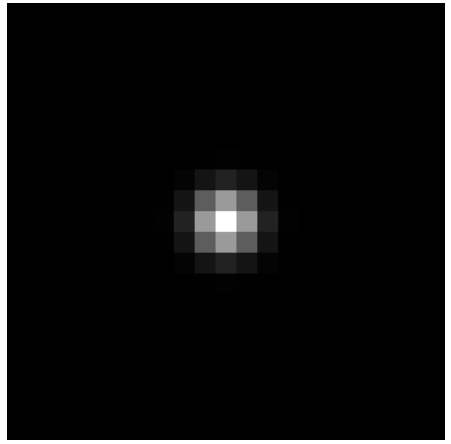




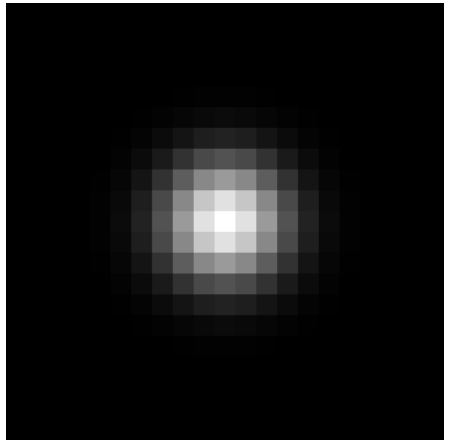
Slide credit: Kristen Grauman

Gaussian Filters

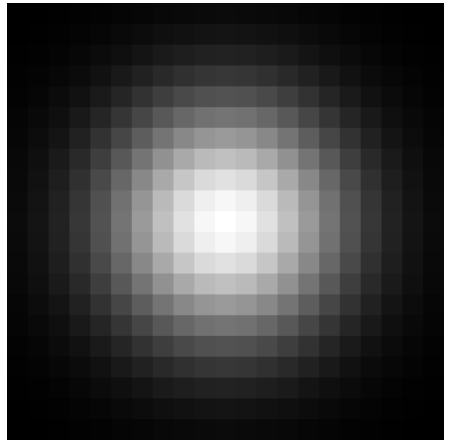
σ = 1
filter = 21x21



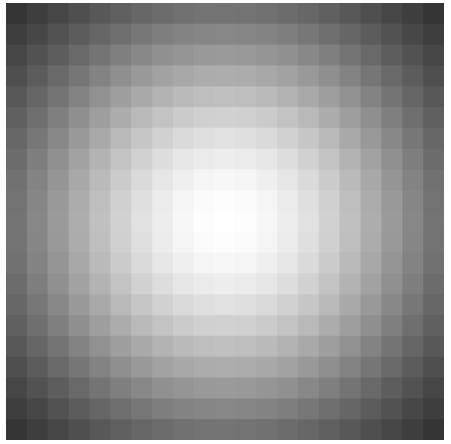
σ = 2
filter = 21x21



σ = 4
filter = 21x21



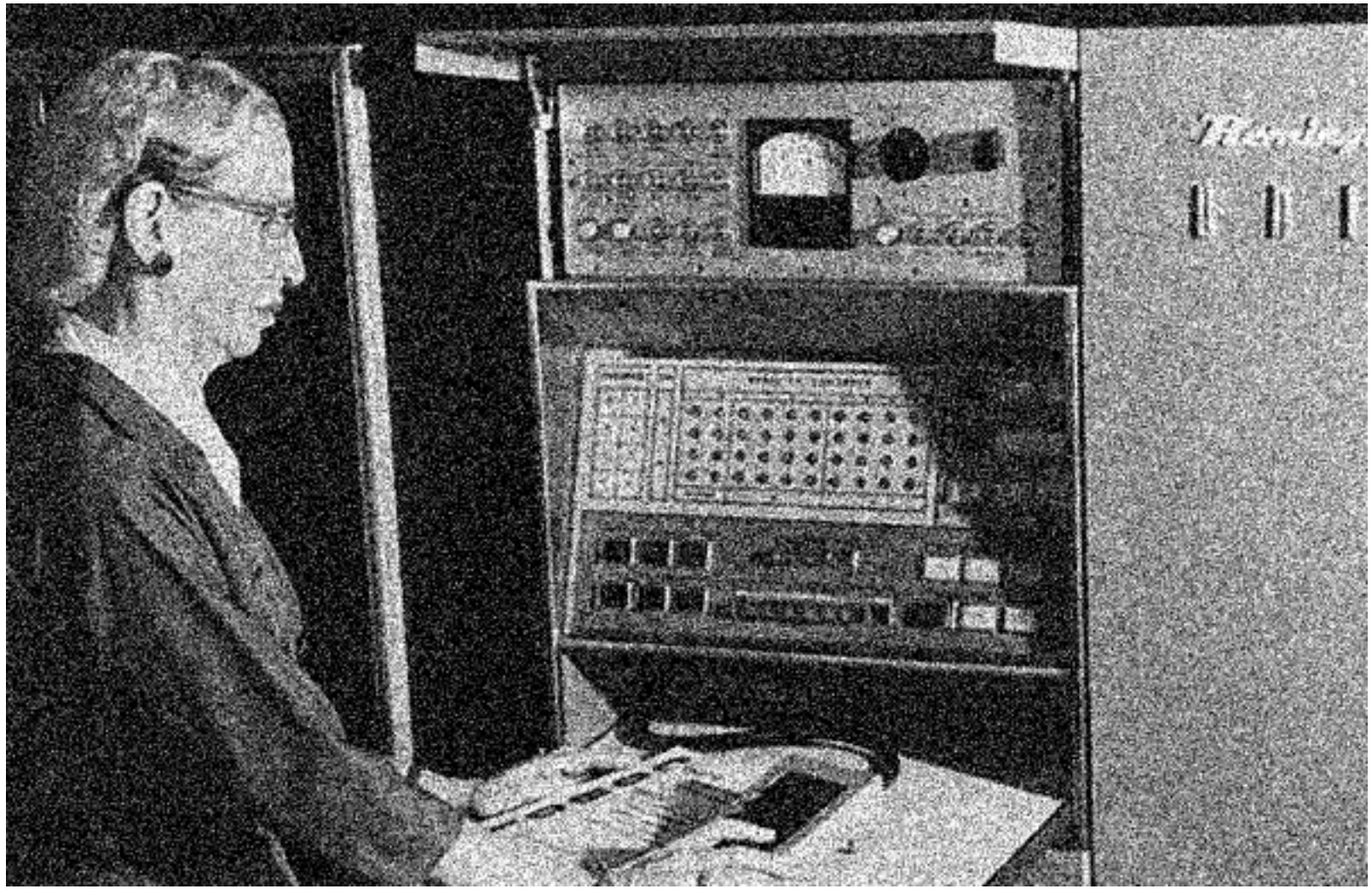
σ = 8
filter = 21x21



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

Slide credit: David Fouhey

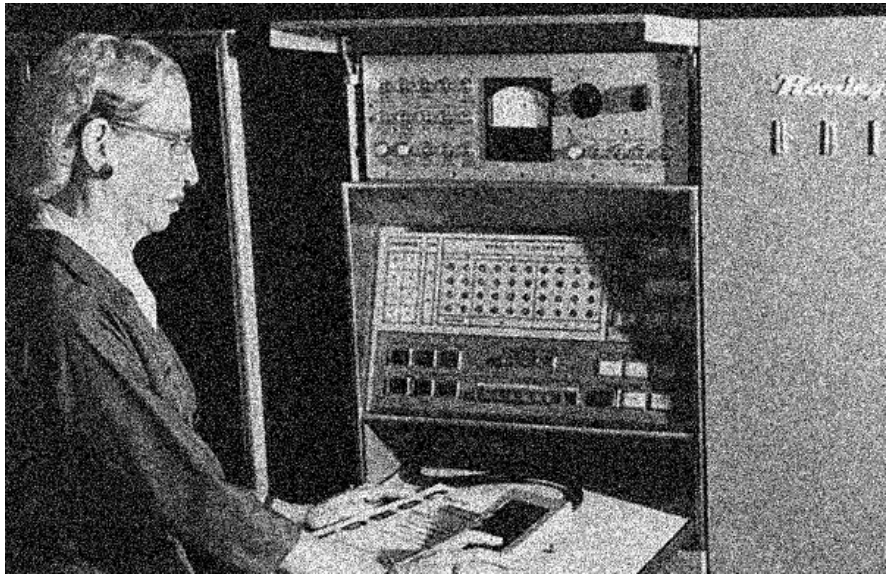
Applying Gaussian Filters

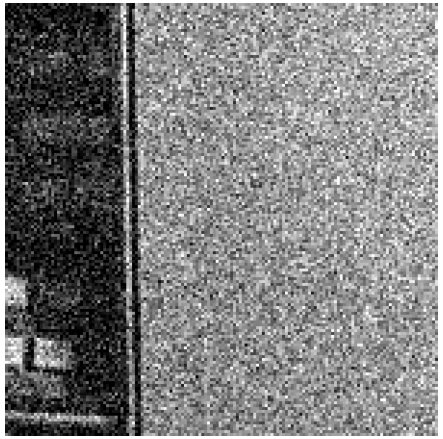


Slide credit: David Fouhey

Applying Gaussian Filters

Input Image
(no filter)

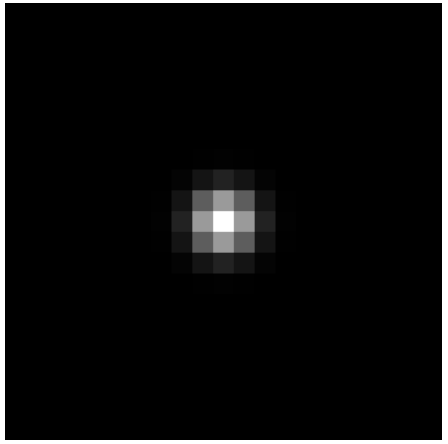


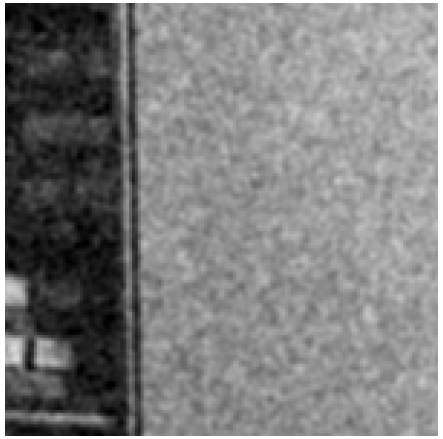


Slide credit: David Fouhey

Applying Gaussian Filters

$\sigma = 1$

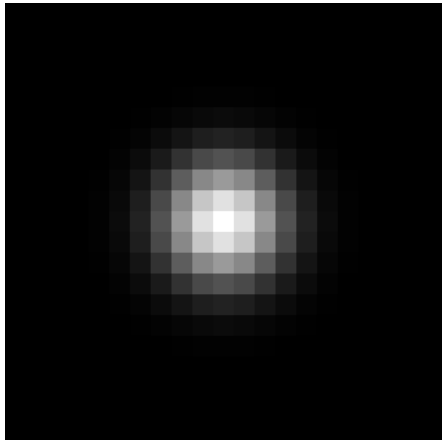


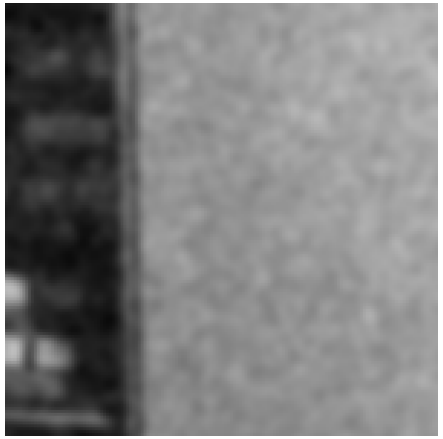


Slide credit: David Fouhey

Applying Gaussian Filters

σ = 2

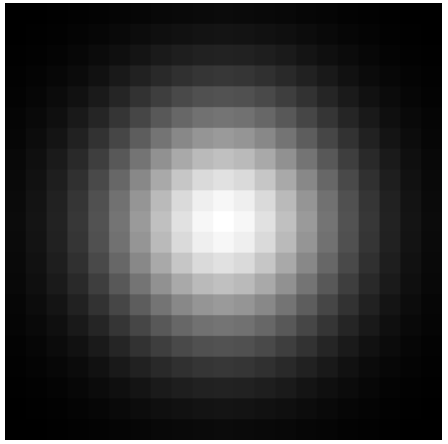




Slide credit: David Fouhey

Applying Gaussian Filters

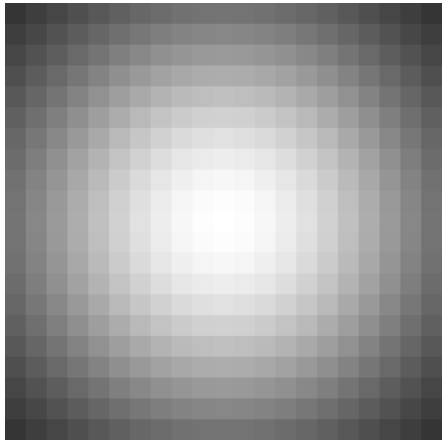
σ = 4



Slide credit: David Fouhey

Applying Gaussian Filters

σ = 8



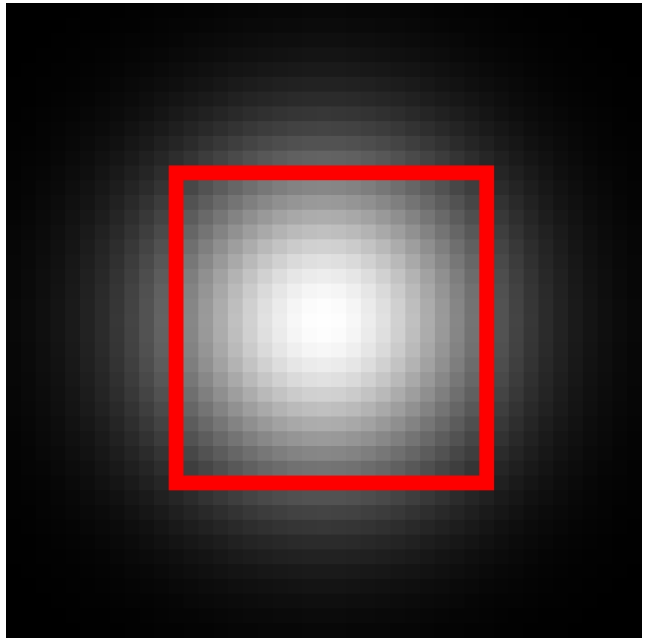
Slide credit: David Fouhey

Picking a Filter Size

Too small filter → bad approximation

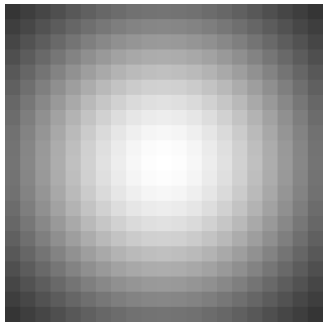
Approx Rule: Want filter size ≈ 6σ (99.7% of energy)

Slightly too small



σ = 8, size = 43

Far too small



σ = 8, size = 21

Slide credit: David Fouhey

Runtime Complexity

Image size = NxN = 6x6

Filter size = MxM = 3x3

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

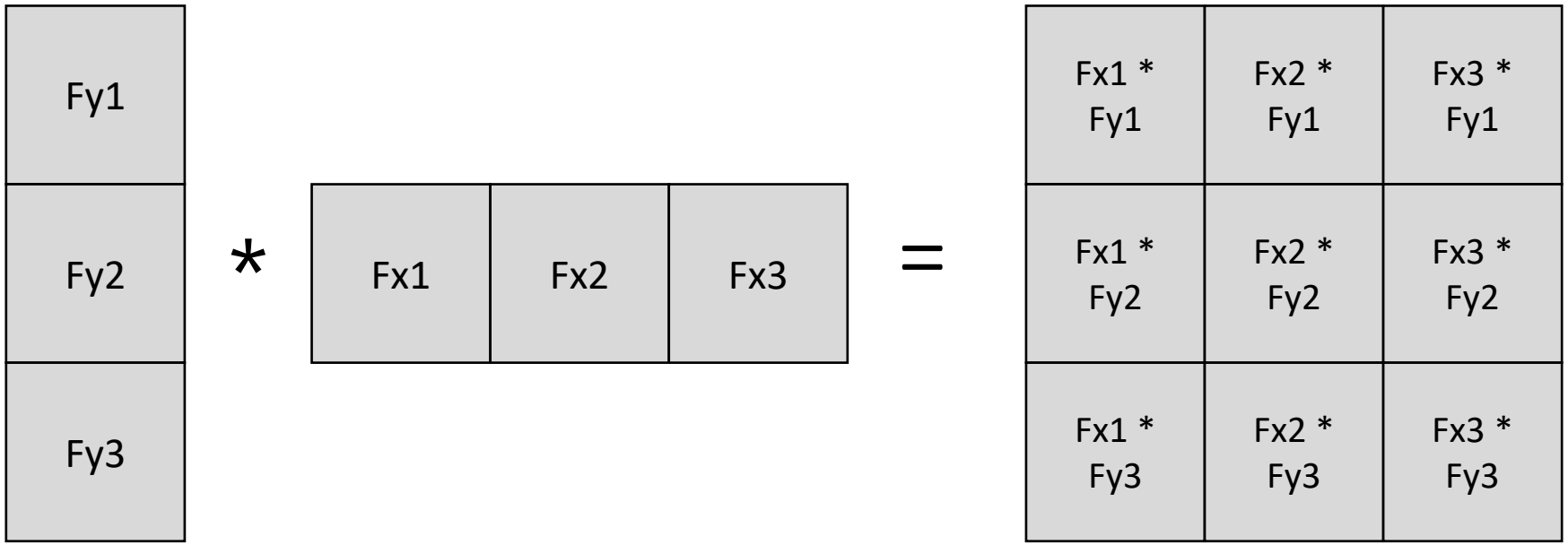
```
for ImageY in range(N):
    for ImageX in range(N):
        for FilterY in range(M):
            for FilterX in range(M):
                …
```

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

Slide credit: David Fouhey

Separability

Conv(vector, transposed vector) → outer product



Slide credit: David Fouhey

Separability

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

$$\rightarrow$$

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

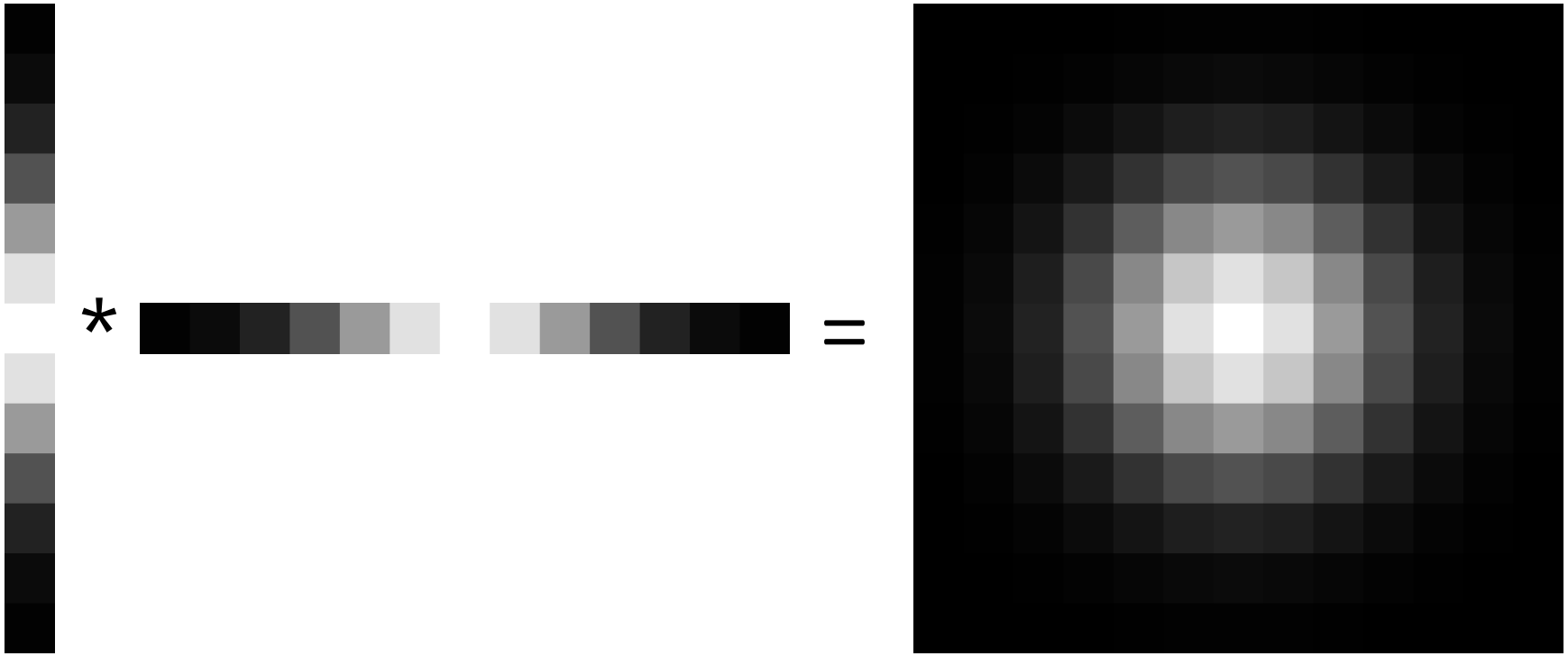
Slide credit: David Fouhey

Separability

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

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

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

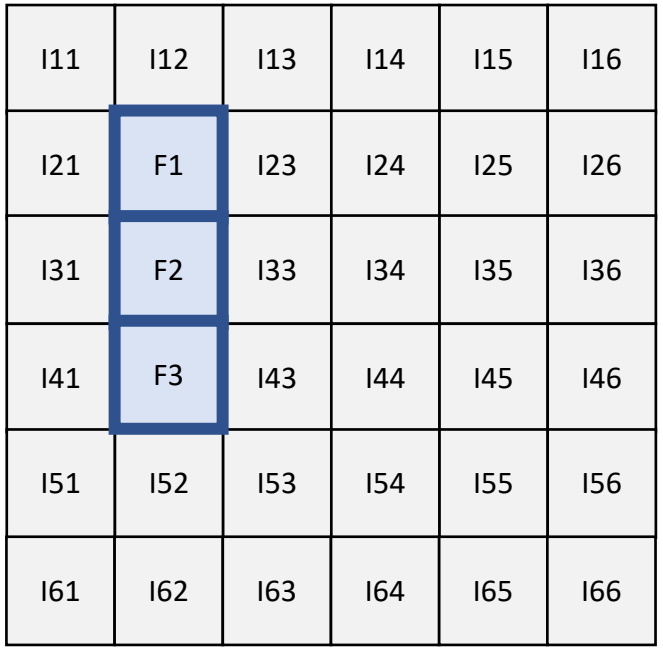


Slide credit: David Fouhey

Runtime Complexity

Image size = NxN = 6x6

Filter size = Mx1 = 3x1




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

Slide credit: David Fouhey

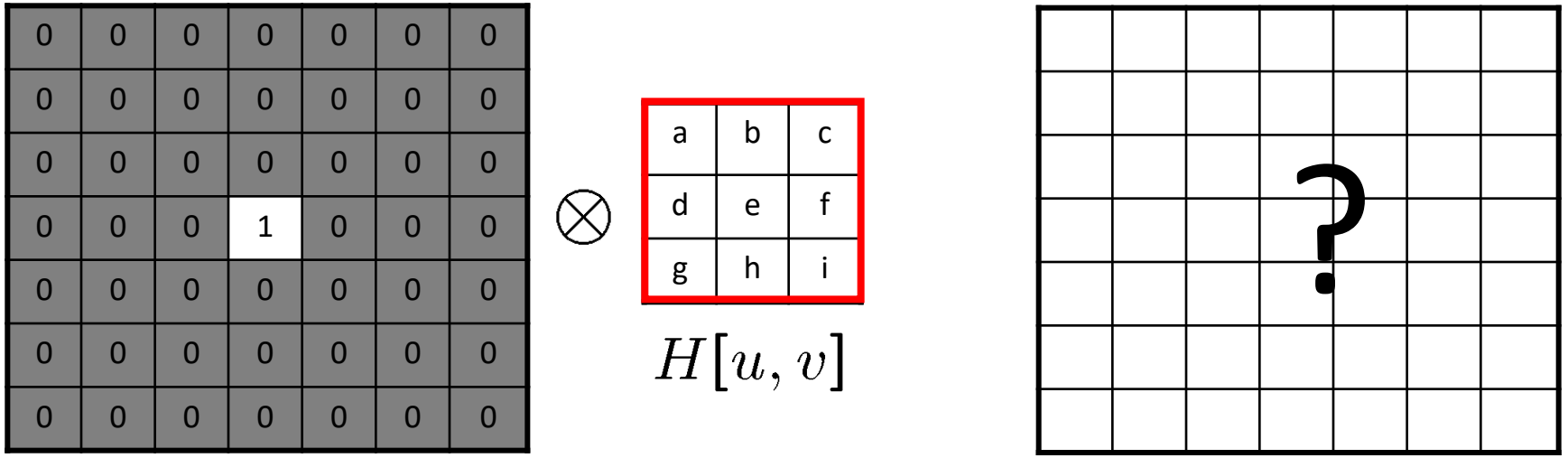
Properties of smoothing filters

- Smoothing
  - Values positive
  - Sum to 1 → constant regions same as input
  - Amount of smoothing proportional to mask size
  - Remove "high-frequency" components; "low-pass" filter

Slide credit: Kristen Grauman

Filtering an impulse signal

What is the result of filtering the impulse signal (image) *F* with the arbitrary kernel *H*?



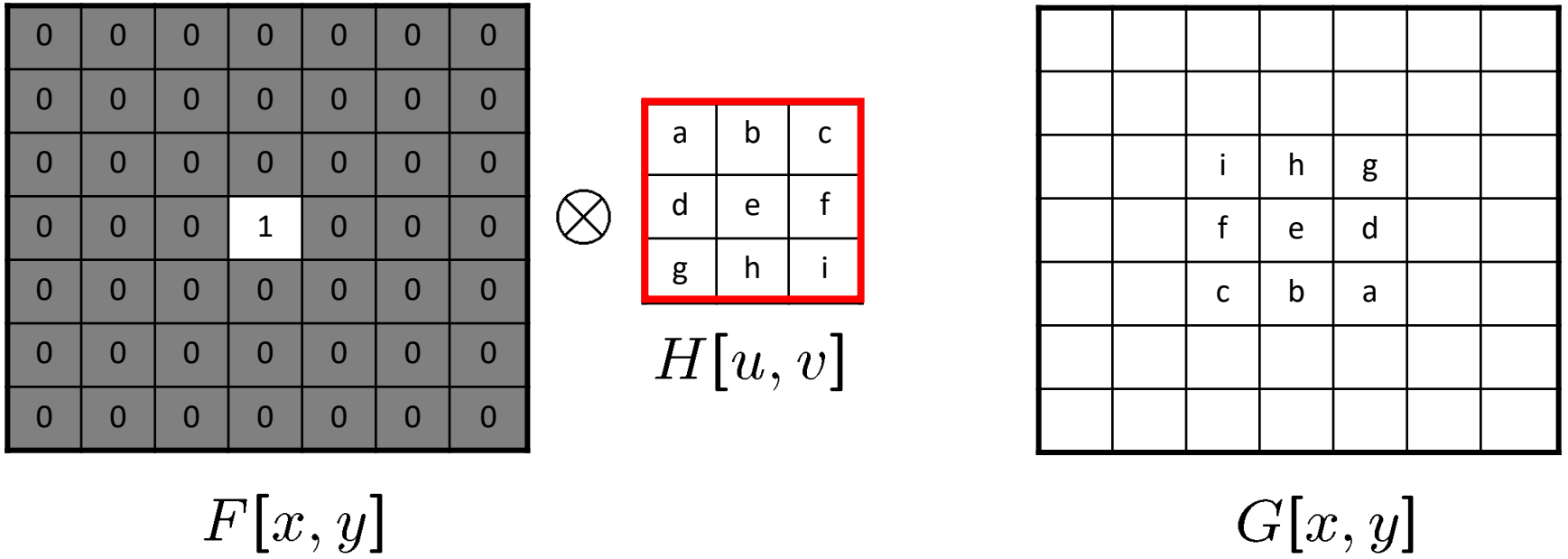
$F[x, y]$ $\otimes$ $H[u, v]$ $G[x, y]$

Talk with your neighbor

Slide credit: Kristen Grauman

Filtering an impulse signal

What is the result of filtering the impulse signal (image) *F* with the arbitrary kernel *H*?



Slide credit: Kristen Grauman

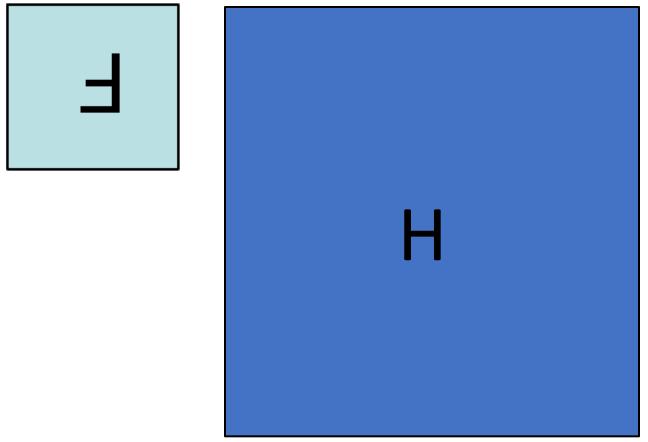
Convolution

- Convolution:
  - Flip the filter in both dimensions (bottom to top, right to left)
  - Then apply cross-correlation

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

$$G = H \star F$$

Notation for convolution operator



Slide credit: Kristen Grauman

Convolution vs. Correlation

Convolution

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

$$G = H \star F$$

Cross-correlation

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

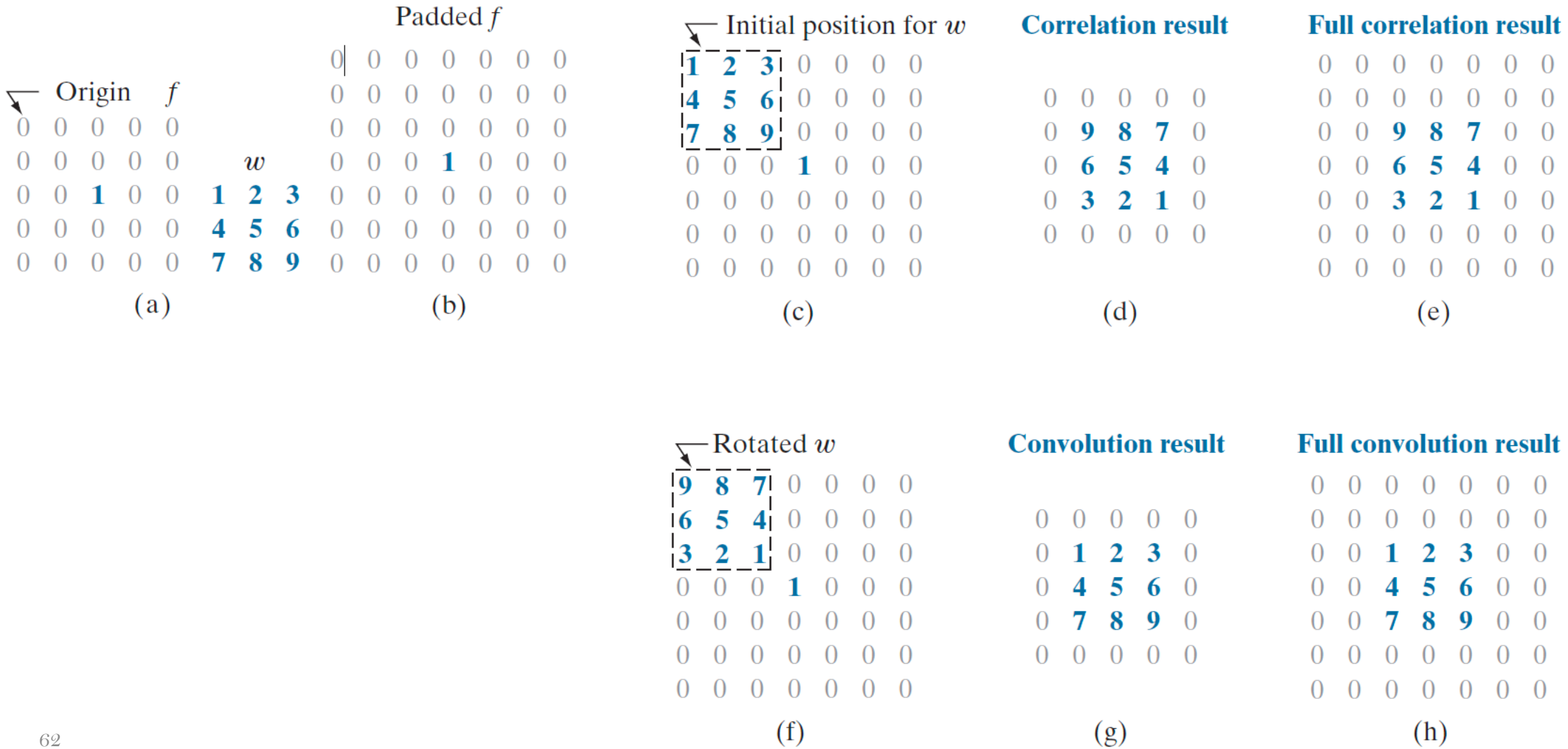
$$G = H \otimes F$$

For a Gaussian or box filter, how will the outputs differ?

If the input is an impulse signal, how will the outputs differ?

Slide credit: Kristen Grauman

Convolution vs. correlation



https://towardsdatascience.com/convolution-vs-correlation-af868b6b4fb5

Boundary Issues

Convolution doesn't keep the whole image.

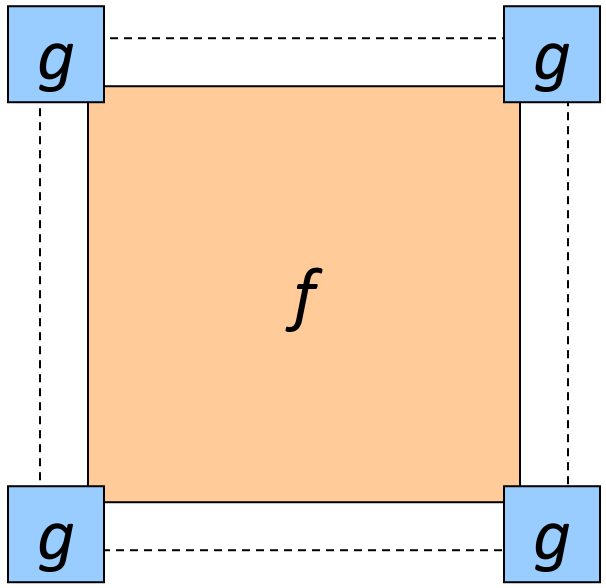
Suppose **f** is the image and **g** the filter.

Full – any part of g touches f.

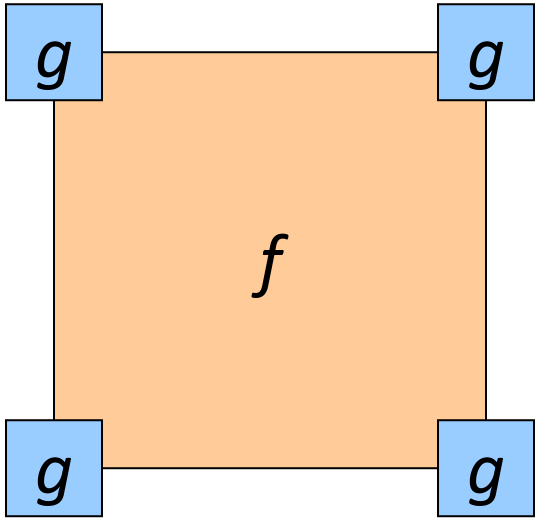
Same – same size as f;

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

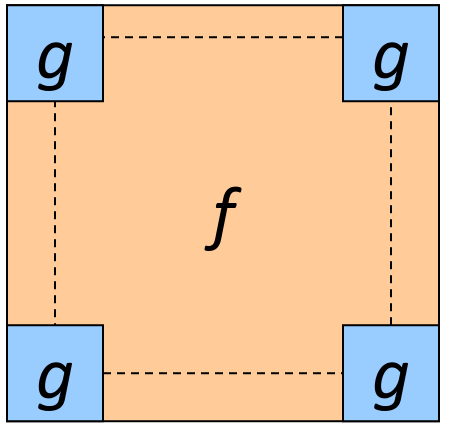
full



same



valid



f/g Diagram Credit: D. Lowe

Slide credit: David Fouhey

Boundary issues

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

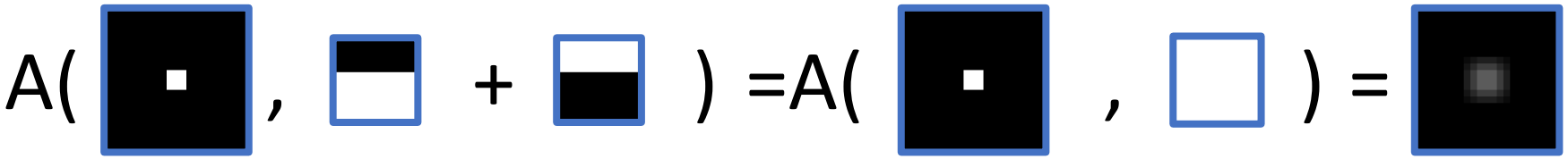
Slide credit: S. Marschner

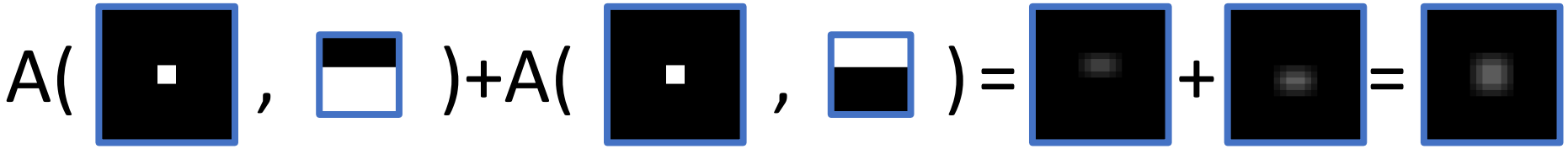
Properties of Convolution: Linear

Given: Image (I), Filters (f1, f2)

Linear: apply(I,f1+f2) = apply(I,f1) + apply(I,f2)

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





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

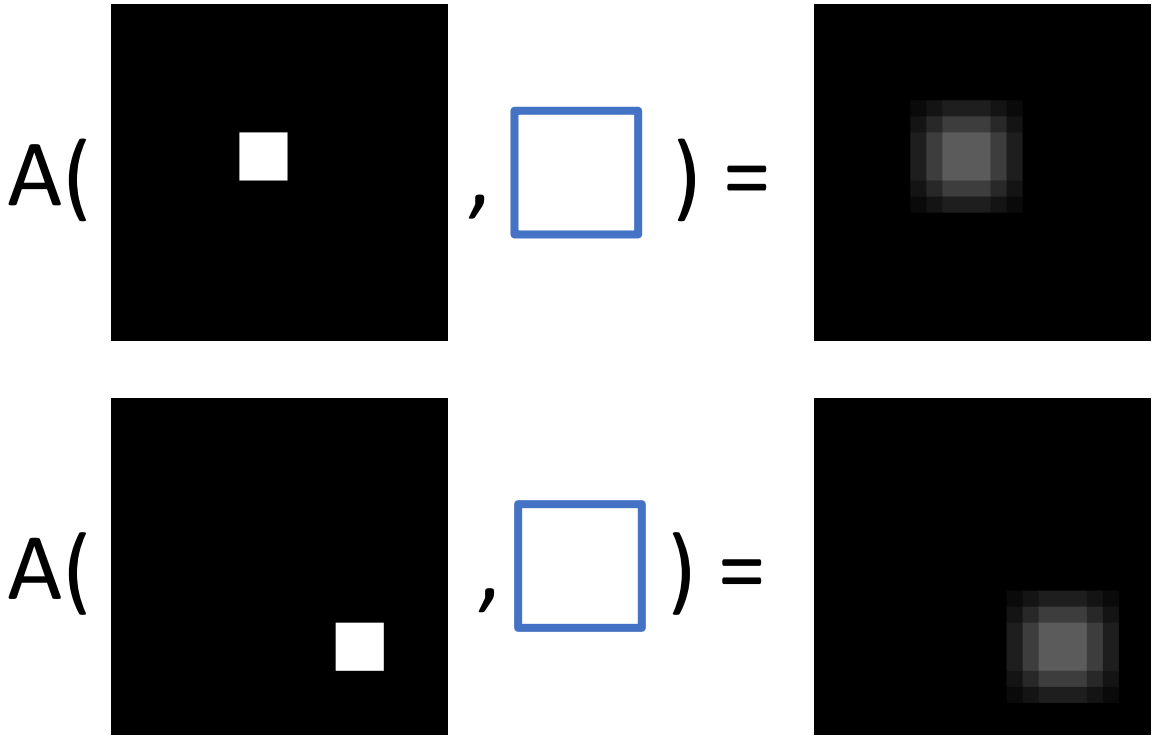
Slide credit: David Fouhey

Properties of Convolution: Shift-Invariant

Given: Image (I), Filter (f)

Shift-invariant: shift(apply(I,f)) = apply(shift(I,f))

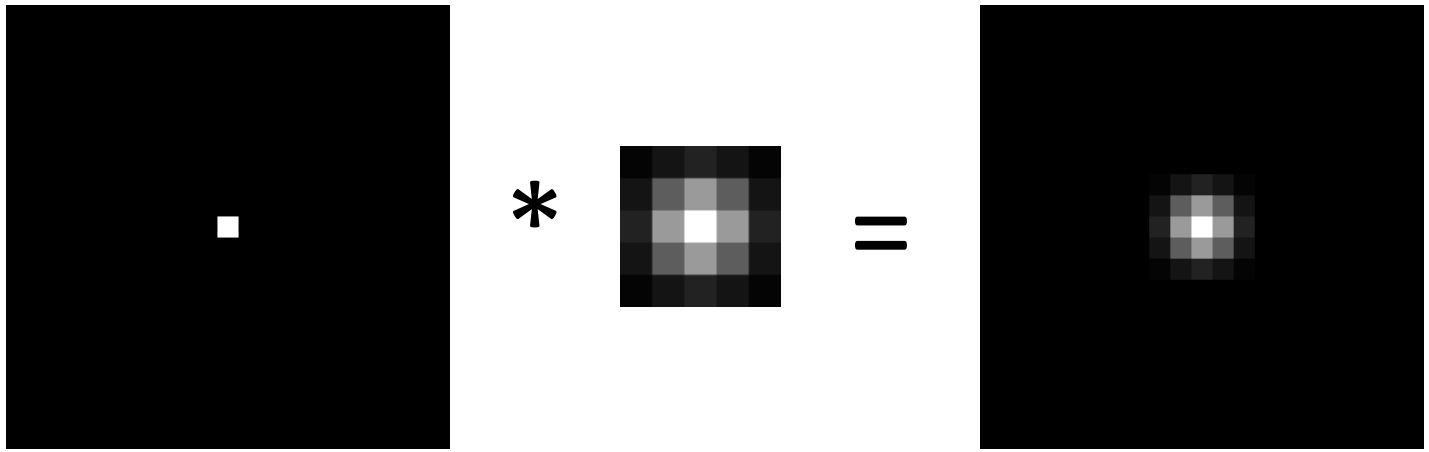
Intuitively: only depends on filter neighborhood



Slide credit: David Fouhey

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):



Slide: David Fouhey; Property List: K. Grauman