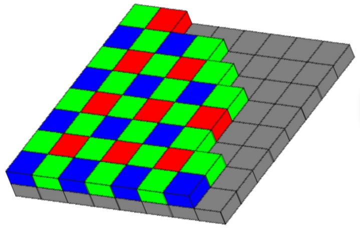
Lecture 5 & 6: Image Processing

COMP 590/776: Computer Vision

Instructor: Soumyadip (Roni) Sengupta

TA: Mykhailo (Misha) Shvets

Recap: A typical color imaging pipeline



Sensor with color filter array (CFA)
(CCD/CMOS)

ISO gain and raw-image processing

RGB Demoasicing

Bayer Pattern:
RG
GB

raw-RGB: linear w.r.t luminance.

Noise Reduction

White-Balance & Color Space Transform (CIE XYZ)

White-balance on raw-RGB.
raw-RGB -> CIE XYZ

Color Manipulation (Photo-finishing)

Mapping to sRGB output

s-RGB

JPEG Compression

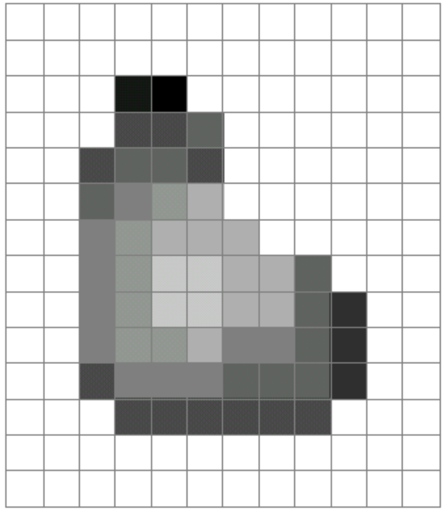
Save to file

Active research topics:
- Noise reduction, especially low-light imaging.
- Auto White Balancing (AWB)

So far: How cameras capture images of the world?
Next: How to we extract useful information and edit images? a.k.a Image Processing (often a whole course of its own)

What is an image?

- A grid (matrix) of intensity values



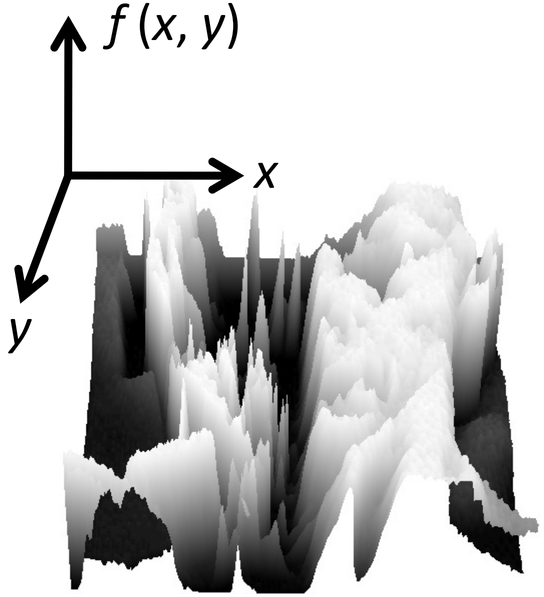
=

255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	20	0	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255
255	255	127	145	175	175	175	255	255	255	255	255
255	255	127	145	200	200	175	175	95	255	255	255
255	255	127	145	200	200	175	175	95	47	255	255
255	255	127	145	145	175	127	127	95	47	255	255
255	255	74	127	127	127	95	95	95	47	255	255
255	255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

(common to use one byte per value: 0 = black, 255 = white)

What is an image?

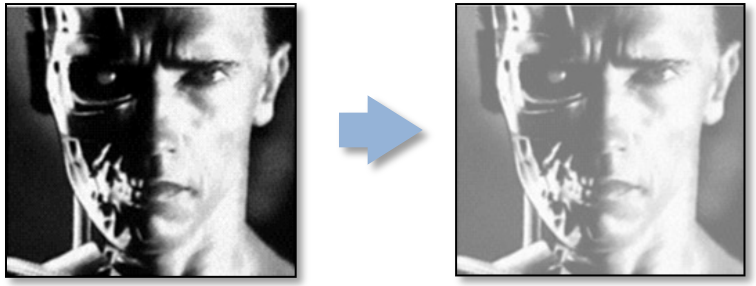
- Can think of a (grayscale) image as a **function** *f* from R^2 to R:
    - *f* (*x,y*) gives the **intensity** at position (*x,y*)



- A **digital** image is a discrete (**sampled**, **quantized**) version of this function

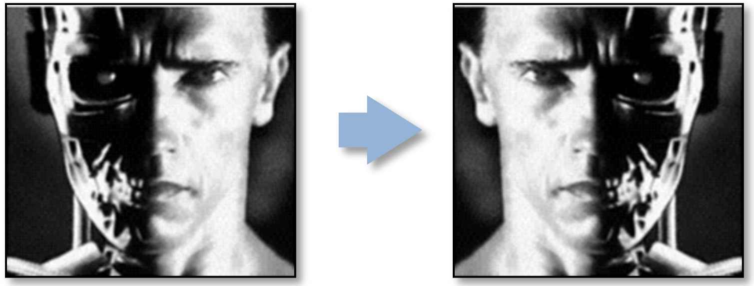
Image transformations

- As with any function, we can apply operators to an image



$g(x,y) = f(x,y) + 20$

brightness



$g(x,y) = f(-x,y)$

Horizontal flip

This lecture: Image Transformations & Filtering

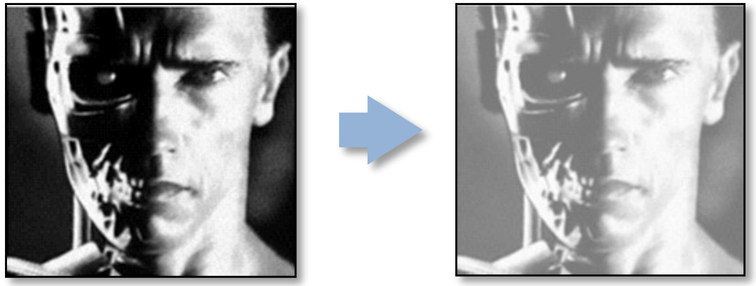
- Point Processing
- Linear Filtering
- Sampling & Aliasing
- Image Derivatives
- Edge Detection

This lecture: Image Transformations & Filtering

- Point Processing
- Linear Filtering
- Sampling & Aliasing
- Image Derivatives
- Edge Detection

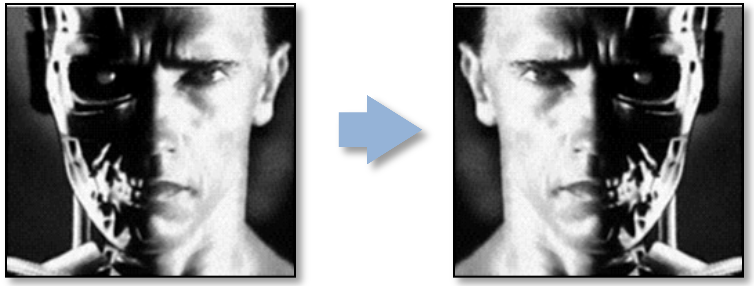
Examples of Point Processing

- As with any function, we can apply operators to an image



$g\ (x,y) = f\ (x,y) + 20$

brightness



$g\ (x,y) = f\ (-x,y)$

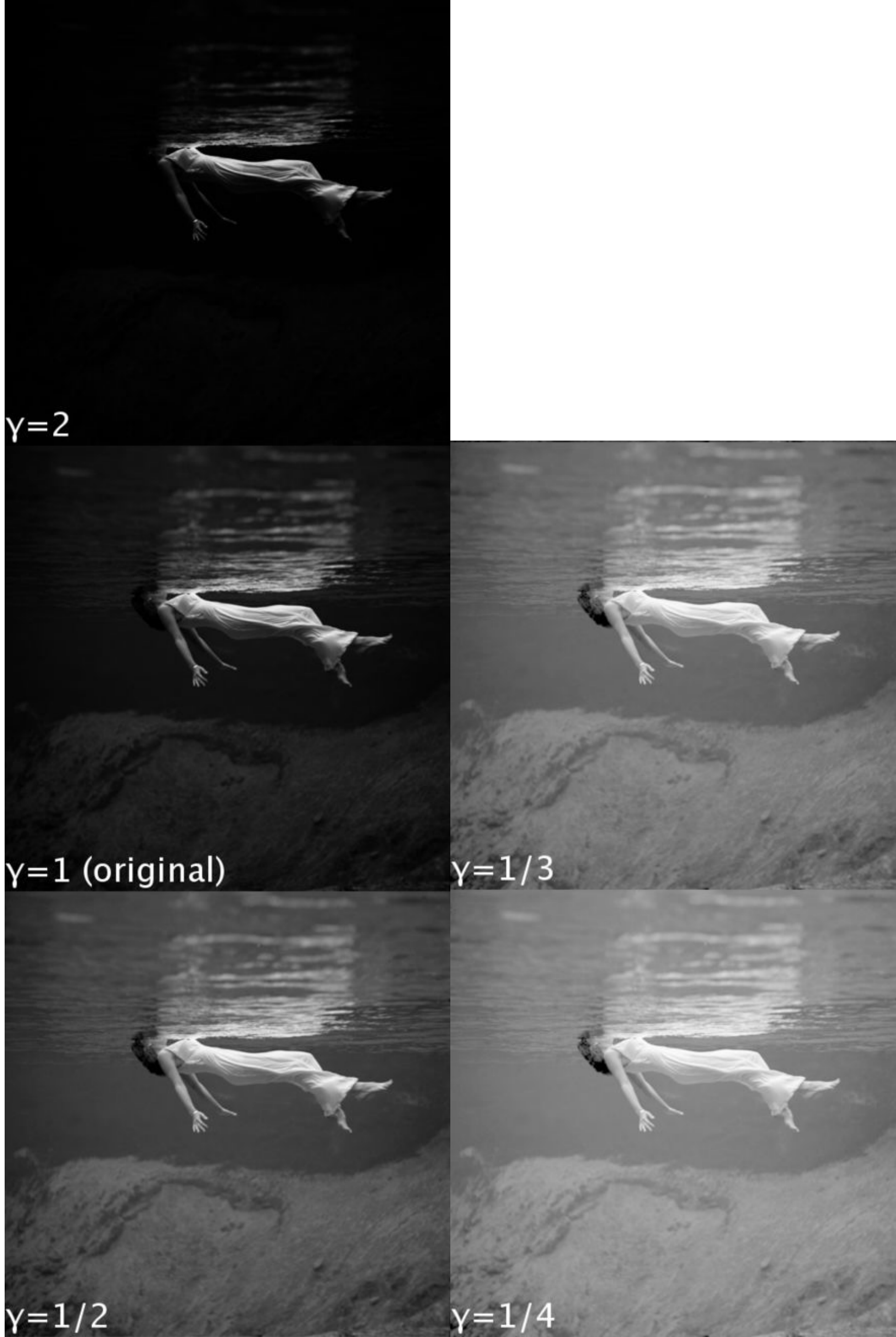
Horizontal flip

Gamma Correction $V_{\text{out}} = AV_{\text{in}}^{\gamma},$

Seen in last class

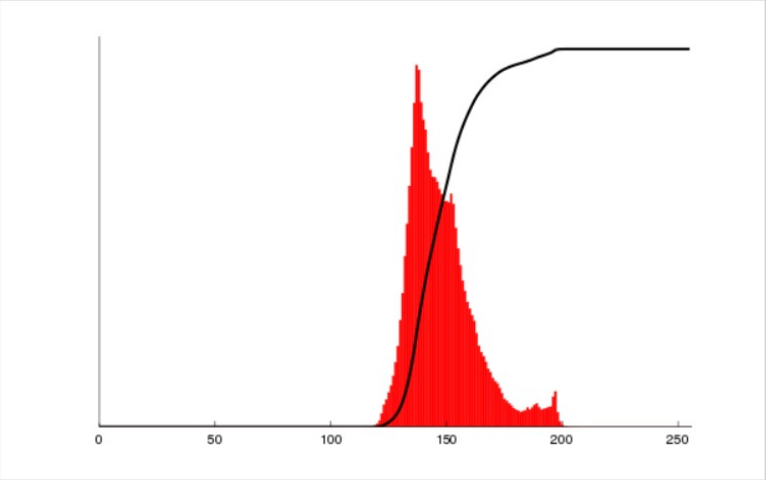
- Most images we work with are in sRGB space, i.e., already tone-mapped.
- Many Vision algorithms expects the image to be in linear space.
- sRGB -> Linear space conversion requires explicit knowledge of camera processing pipeline. Requires knowledge of tone-production curve.
- In absence of it it is very common to use $\gamma = 2.2$.
- Note: This is not accurate, just a cheap approximation that works for most Vision tasks.

credits: wiki

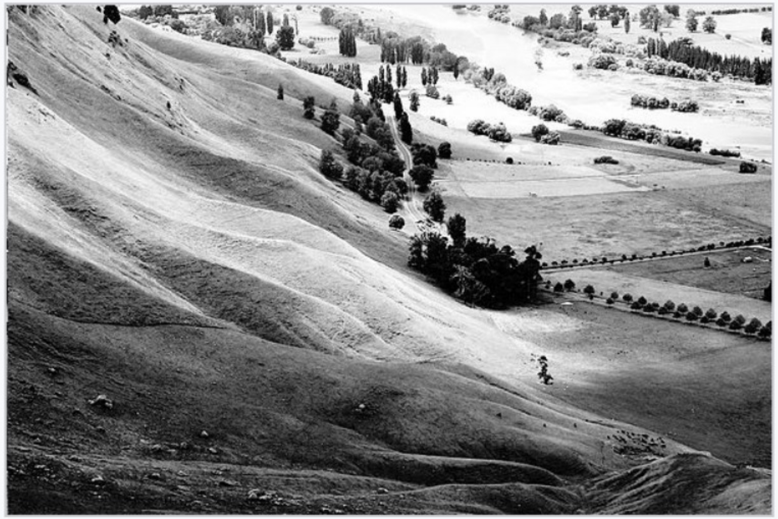


Histogram Equalization

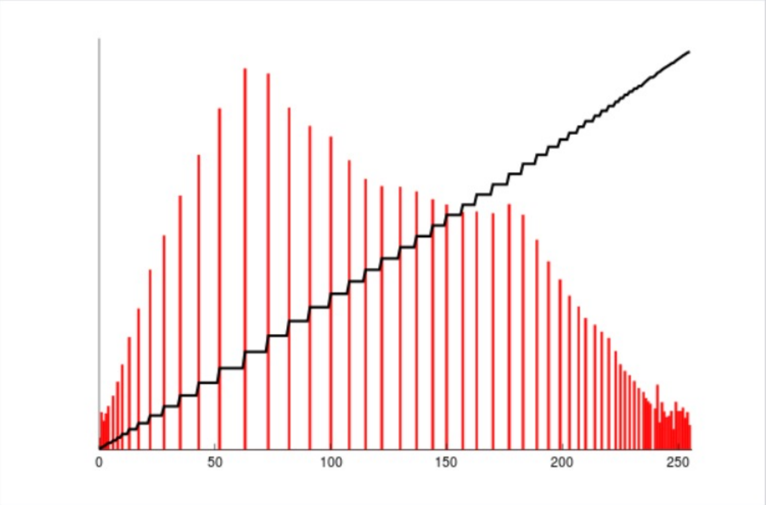
Before Histogram Equalization



Corresponding histogram (red) and cumulative histogram (black)



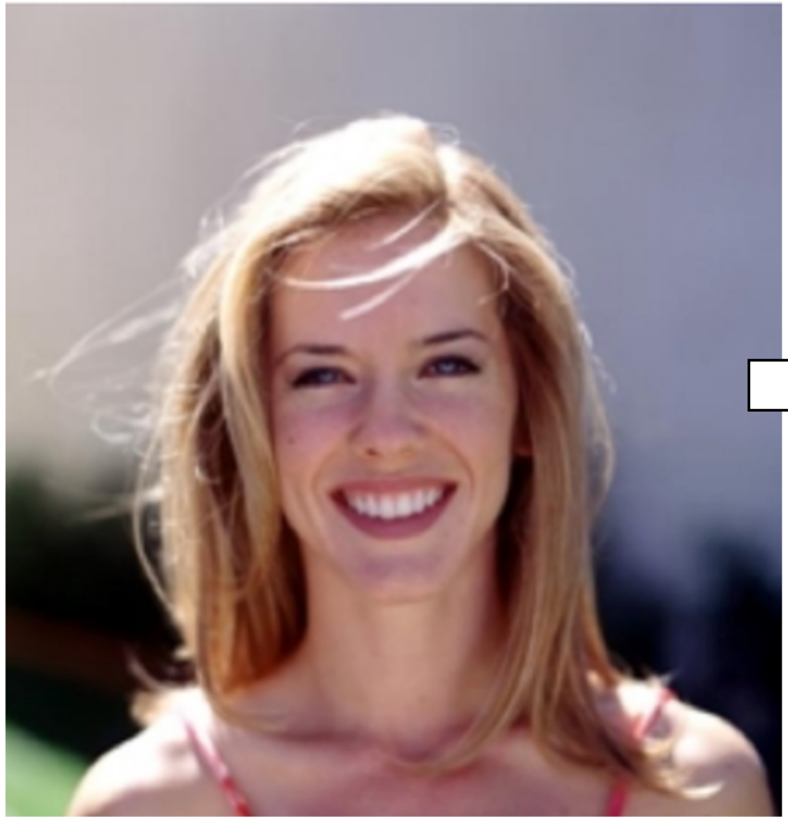
After Histogram Equalization



Corresponding histogram (red) and cumulative histogram (black)

Alpha Matting

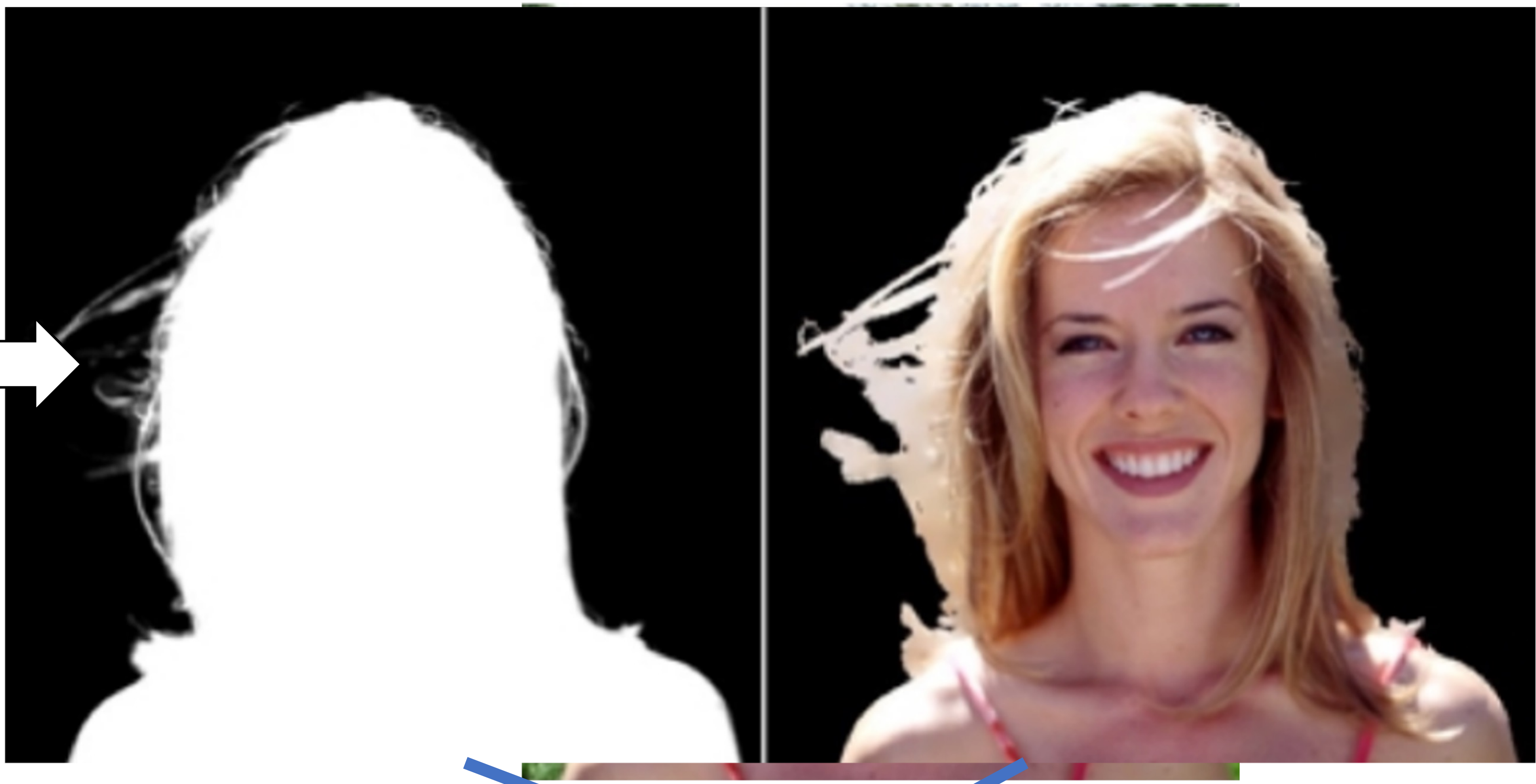
Image



Alpha matte

Composed Image

Foreground layer



$$I = \alpha * F + (1 - \alpha) * B$$

This lecture: Image Transformations & Filtering

- Point Processing
- Linear Filtering
- Sampling & Aliasing
- Image Derivatives
- Edge Detection

Filters

- Filtering
  - Form a new image whose pixel values are a combination of the original pixel values.

- Why?
  - To get useful information from images
    - E.g., extract edges or contours (to understand shape)
  - To enhance the image
    - E.g., to remove noise
    - E.g., to sharpen and “enhance image” a la CSI
  - A key operator in Convolutional Neural Networks

Canonical Image Processing problems

- Image Restoration
  - denoising
  - deblurring
- Image Compression
  - JPEG, HEIF, MPEG, …
- Computing Field Properties
  - optical flow
  - disparity
- Locating Structural Features
  - corners
  - edges

Question: Noise reduction

- Given a camera and a still scene, how can you reduce noise?

Take lots of images and average them!

What's the next best thing?

Source: S. Seitz

Image filtering

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

10	5	3
4	5	1
1	1	7

Local image data

Some function →

	7	

Modified image data

Source: L. Zhang

Linear filtering

- One simple version of filtering: linear filtering (cross-correlation, convolution)
  - Replace each pixel by a linear combination (a weighted sum) of its neighbors
- The prescription for the linear combination is called the “kernel” (or “mask”, “filter”)

10	5	3
4	6	1
1	1	8

Local image data

0	0	0
0	0.5	0
0	1	0.5

kernel

	8	

Modified image data

Source: L. Zhang

Cross-correlation

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

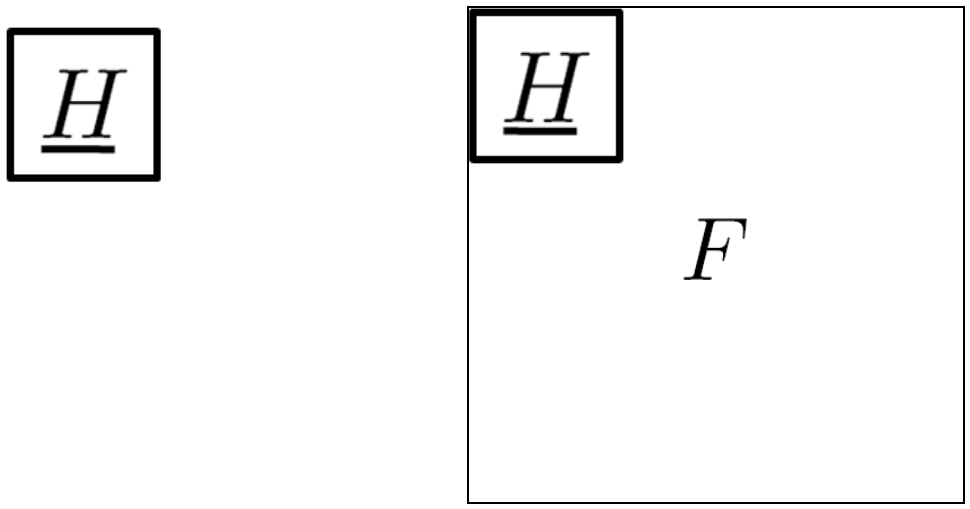
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$

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

$$G = H \otimes F$$

- Can think of as a "dot product" between local neighborhood and kernel for each pixel

Cross-Correlation



Adapted from F. Durand

Origin f

$$\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array} \qquad \begin{array}{c} w \\ \begin{array}{ccc} \mathbf{1} & \mathbf{2} & \mathbf{3} \\ \mathbf{4} & \mathbf{5} & \mathbf{6} \\ \mathbf{7} & \mathbf{8} & \mathbf{9} \end{array} \end{array}$$

(a)

Padded f

$$\begin{array}{ccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

(b)

Initial position for w

$$\begin{array}{ccccccc} \mathbf{1} & \mathbf{2} & \mathbf{3} & 0 & 0 & 0 & 0 \\ \mathbf{4} & \mathbf{5} & \mathbf{6} & 0 & 0 & 0 & 0 \\ \mathbf{7} & \mathbf{8} & \mathbf{9} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

(c)

Correlation result

$$\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{9} & \mathbf{8} & \mathbf{7} & 0 \\ 0 & \mathbf{6} & \mathbf{5} & \mathbf{4} & 0 \\ 0 & \mathbf{3} & \mathbf{2} & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}$$

(d)

Convolution

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

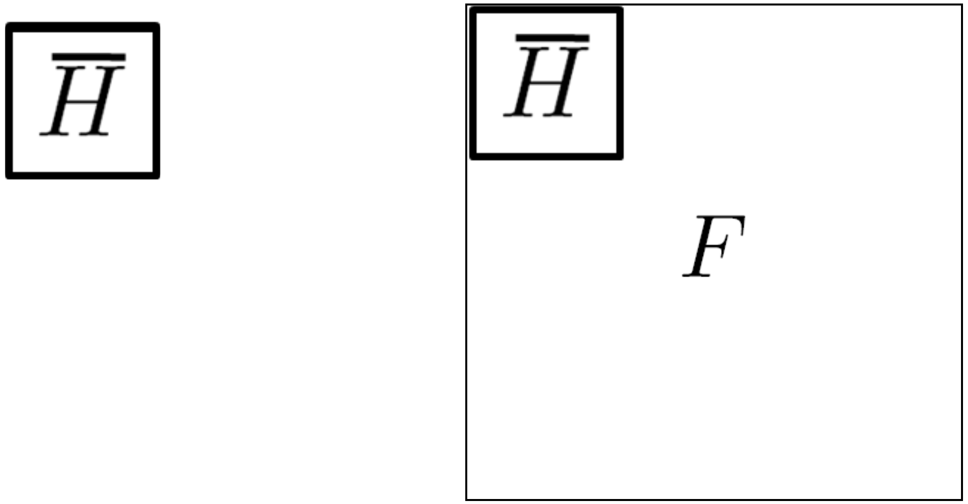
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

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

Cross-correlation

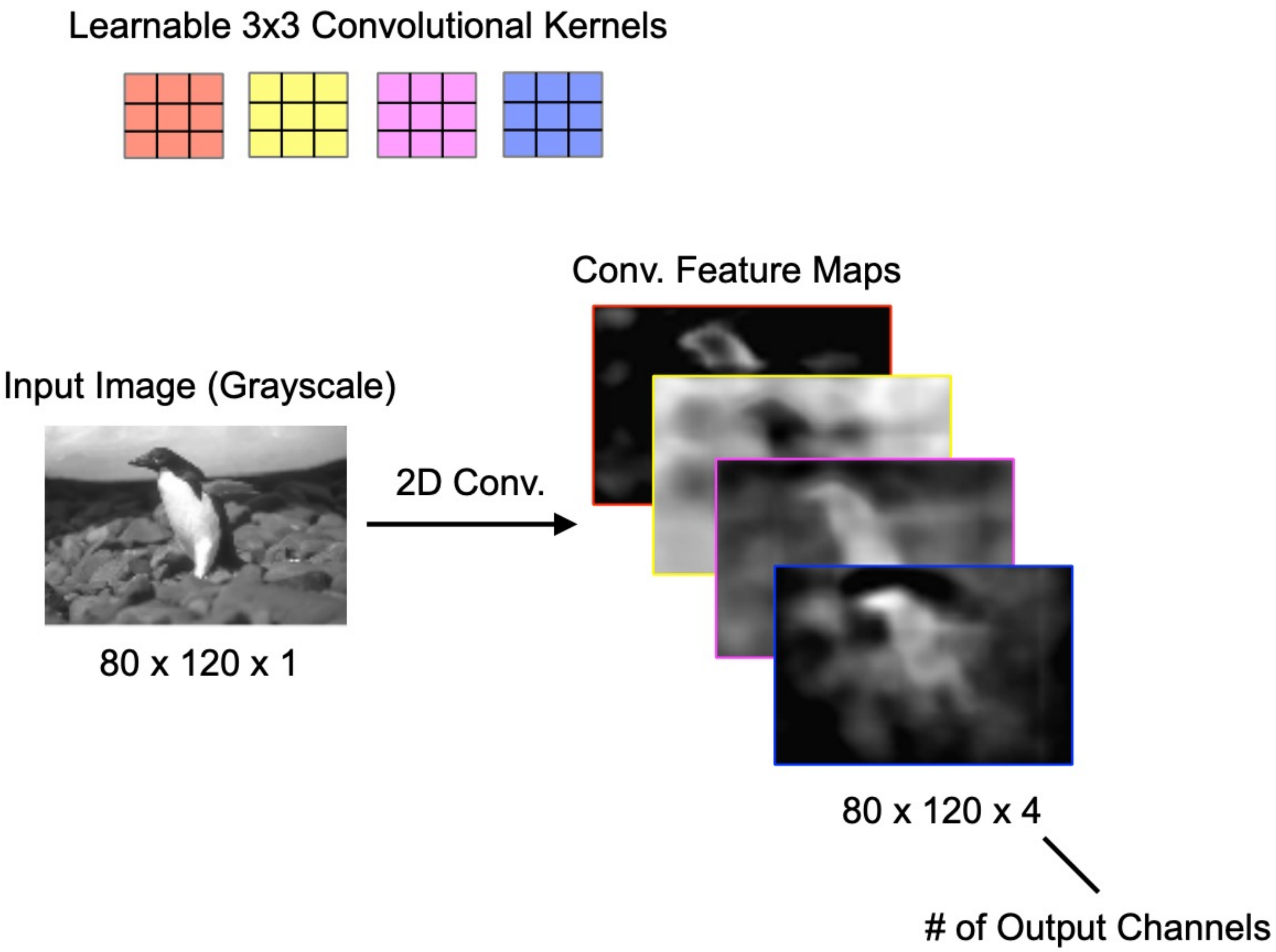
- Convolution is **commutative** and **associative**

Convolution



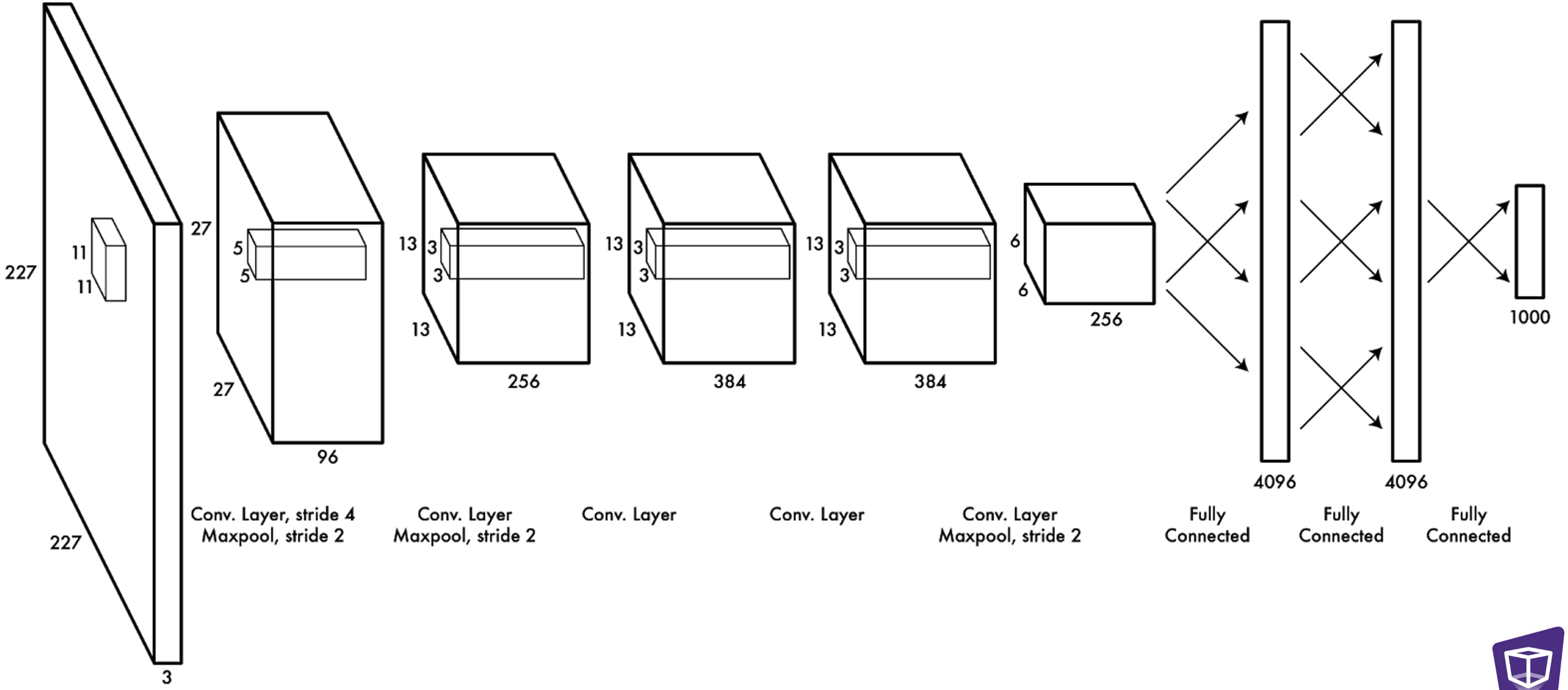
Adapted from F. Durand

Convolutional Networks



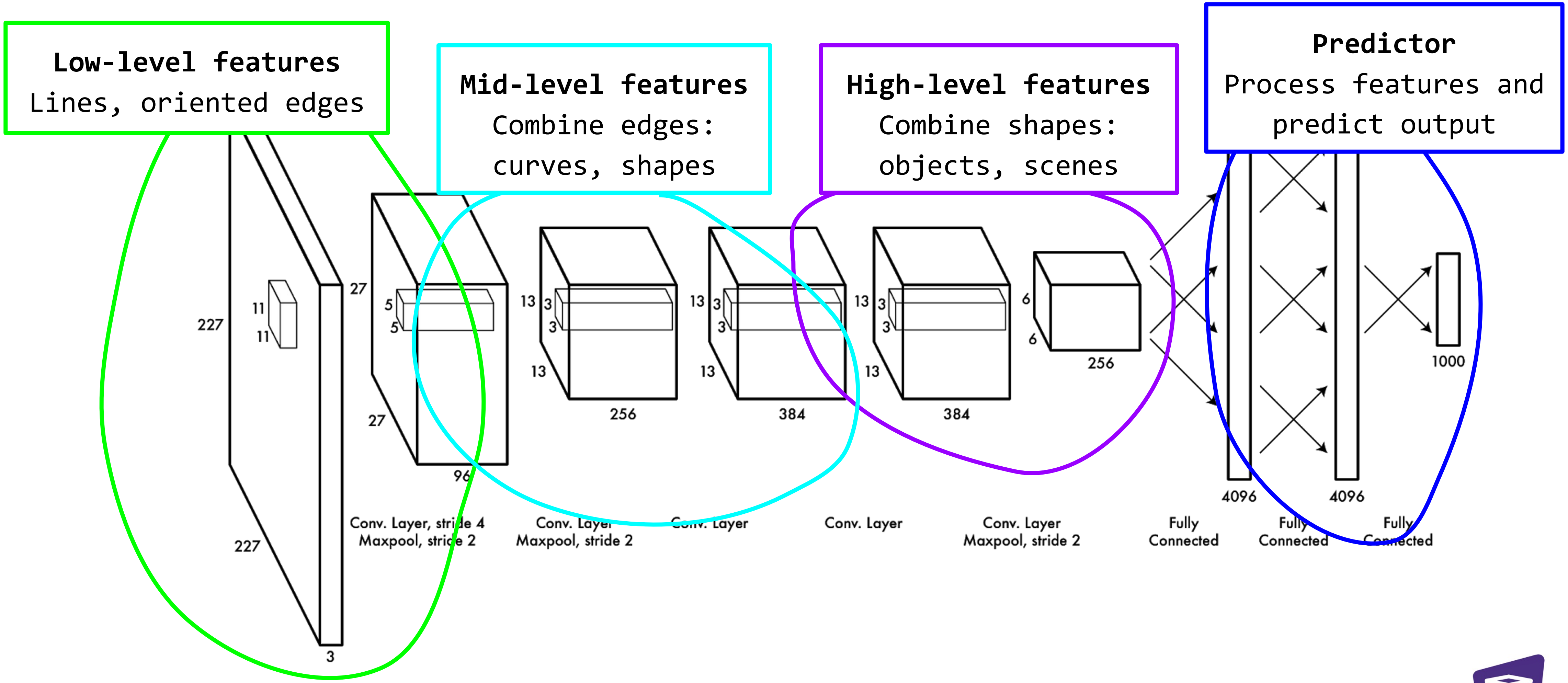
Slide Credits: Gedas Bertasius

AlexNet: An Early Example

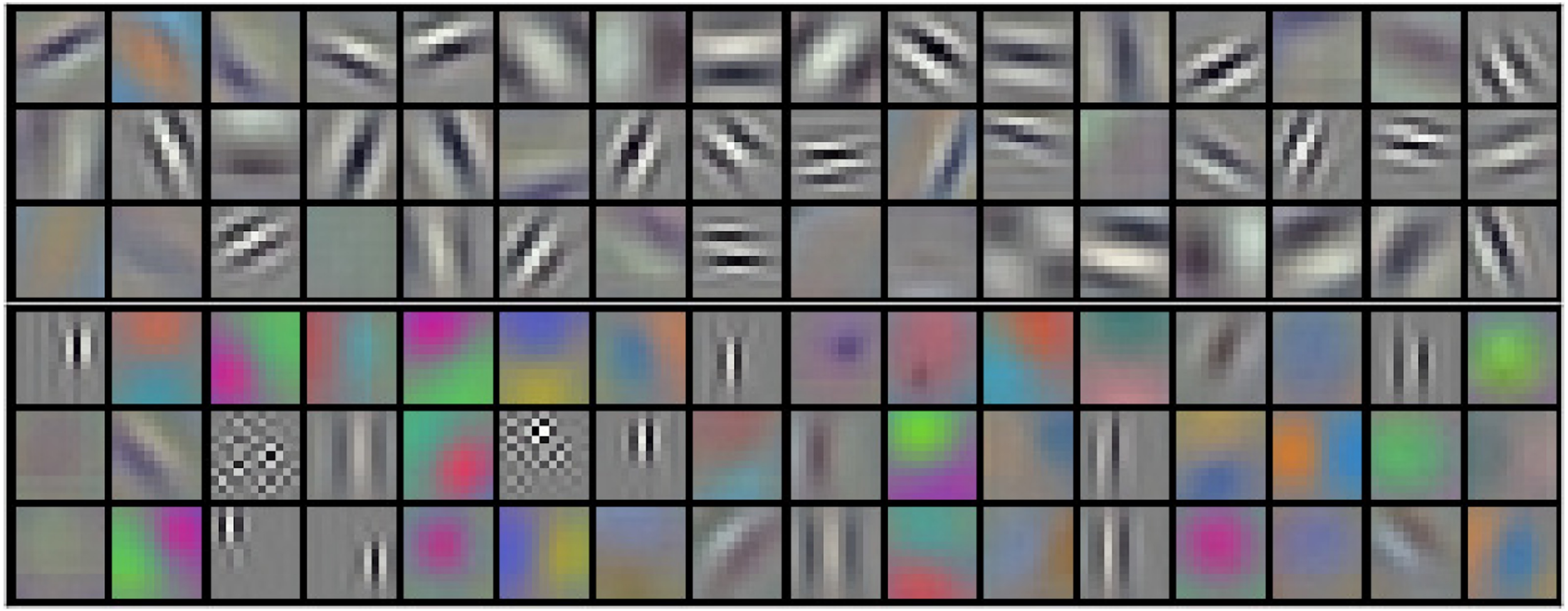


https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

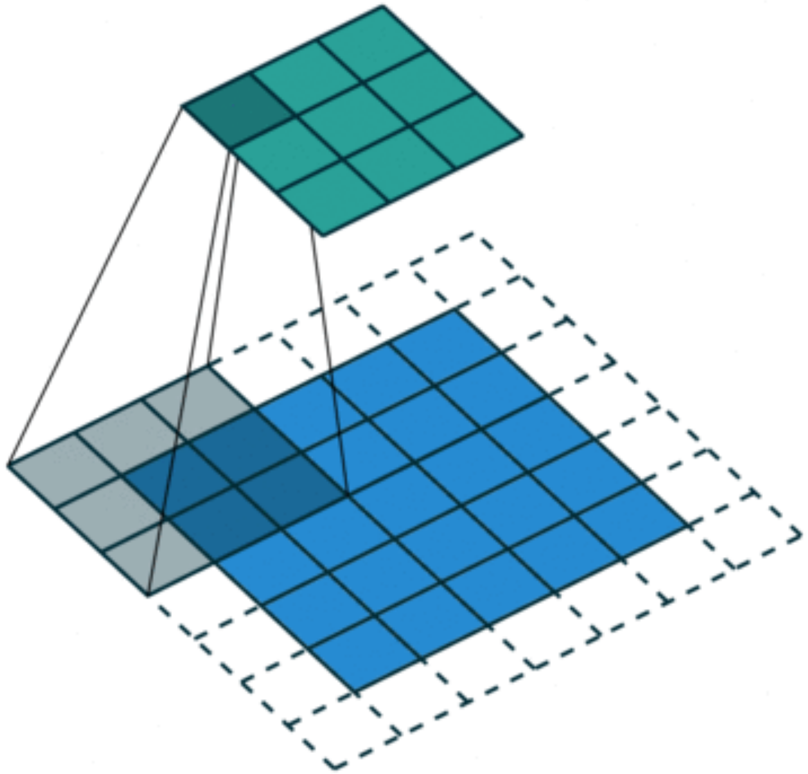
Where Models Learn Features of an Image



What are Models Actually Looking For?



Padding & Stride in CNN



padding=1, stride=2

CLASS torch.nn.Conv2d(*in_channels*, *out_channels*, *kernel_size*, *stride=1*, *padding=0*, *dilation=1*, *groups=1*, *bias=True*, *padding_mode='zeros'*, *device=None*, *dtype=None*) [SOURCE]

Applies a 2D convolution over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size (N, C_{in}, H, W) and output $(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$ can be precisely described as:

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k)$$

where $\star$ is the valid 2D cross-correlation operator, N is a batch size, C denotes a number of channels, H is a height of input planes in pixels, and W is width in pixels.

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

Convolution is nice!

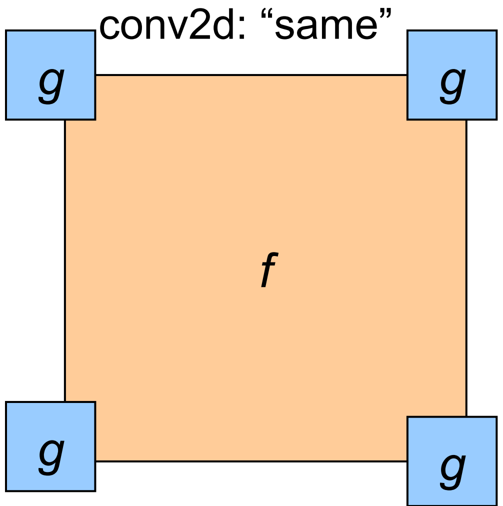
- Notation: $b = c \star a$

- Convolution is a multiplication-like operation
  - commutative $a \star b = b \star a$
  - associative $a \star (b \star c) = (a \star b) \star c$
  - distributes over addition $a \star (b + c) = a \star b + a \star c$
  - scalars factor out $\alpha a \star b = a \star \alpha b = \alpha(a \star b)$
  - identity: unit impulse *e* = [..., 0, 0, 1, 0, 0, ...] $a \star e = a$

- Usefulness of associativity
  - often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - this is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$

Conv/Filtering: Practical matters

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



Source: S. Marschner

Mean filtering

H

$*$

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

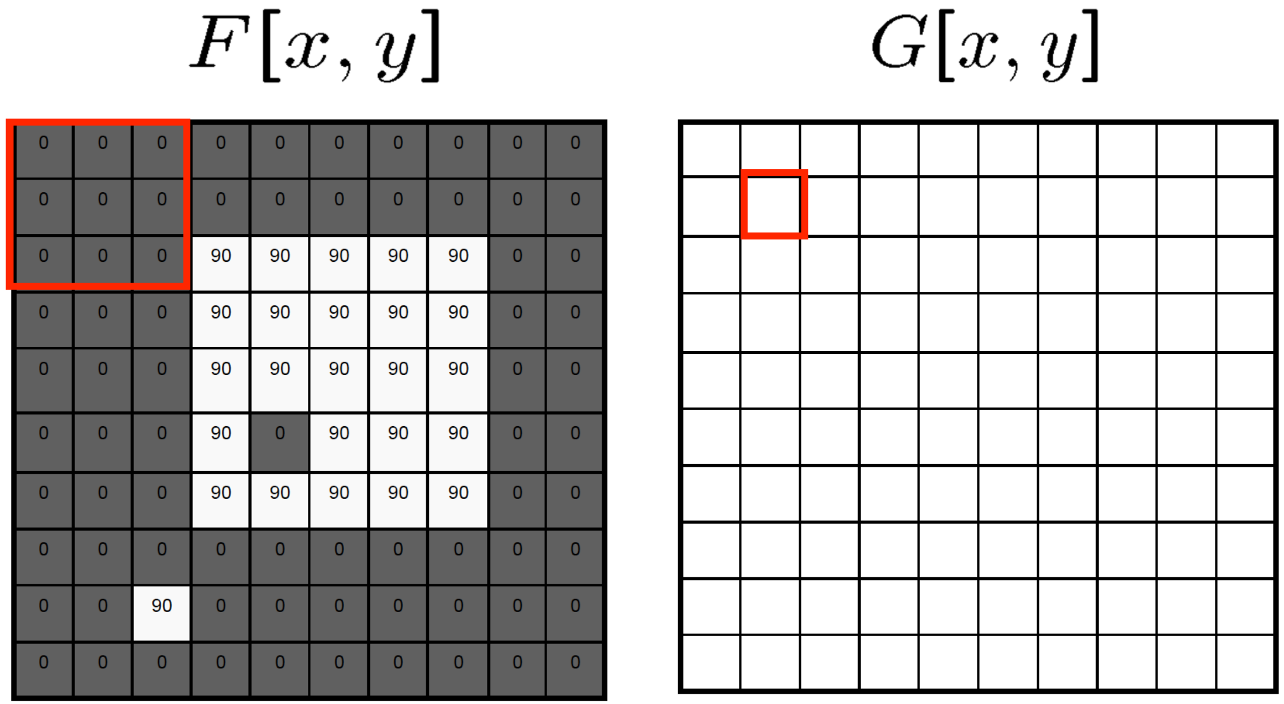
F

$=$

	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	

G

Mean filtering/Moving average



Mean filtering/Moving average

$F[x, y]$

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							

Mean filtering/Moving average

$F[x, y]$

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						

Mean filtering/Moving average

$F[x, y]$

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					

Mean filtering/Moving average

$F[x, y]$

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				

Mean filtering/Moving average

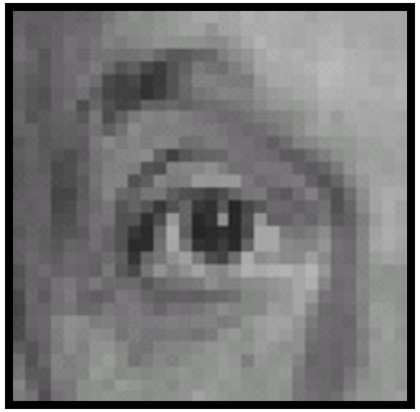
$F[x, y]$

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	

Linear filters: examples



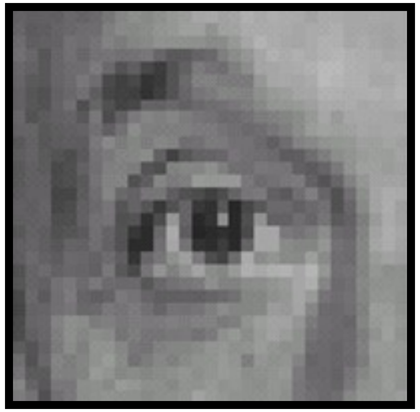
Original

$*$

0	0	0
0	1	0
0	0	0

Source: D. Lowe

Linear filters: examples



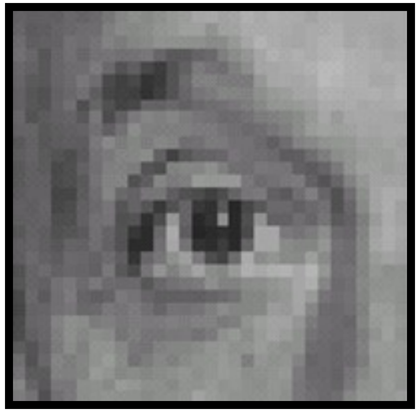
Original

*

0	0	0
1	0	0
0	0	0

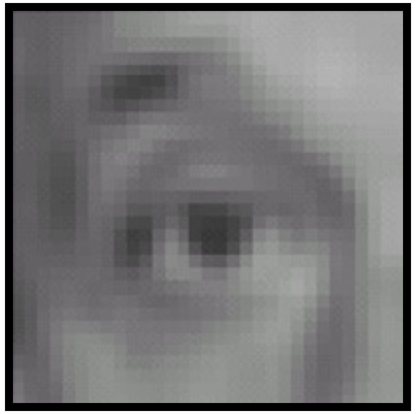
Source: D. Lowe

Linear filters: examples



Original

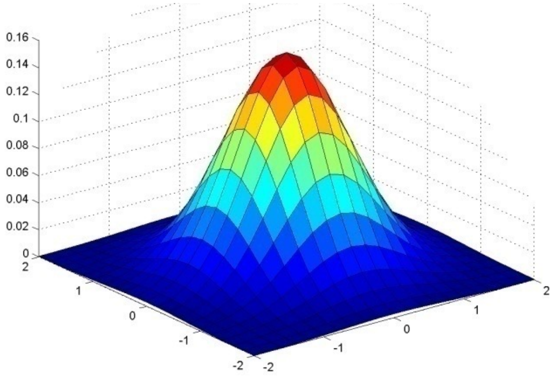
$$* \ \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$

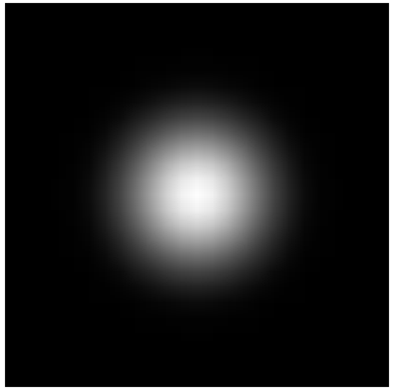


Blur (with a mean filter)

Source: D. Lowe

Gaussian kernel

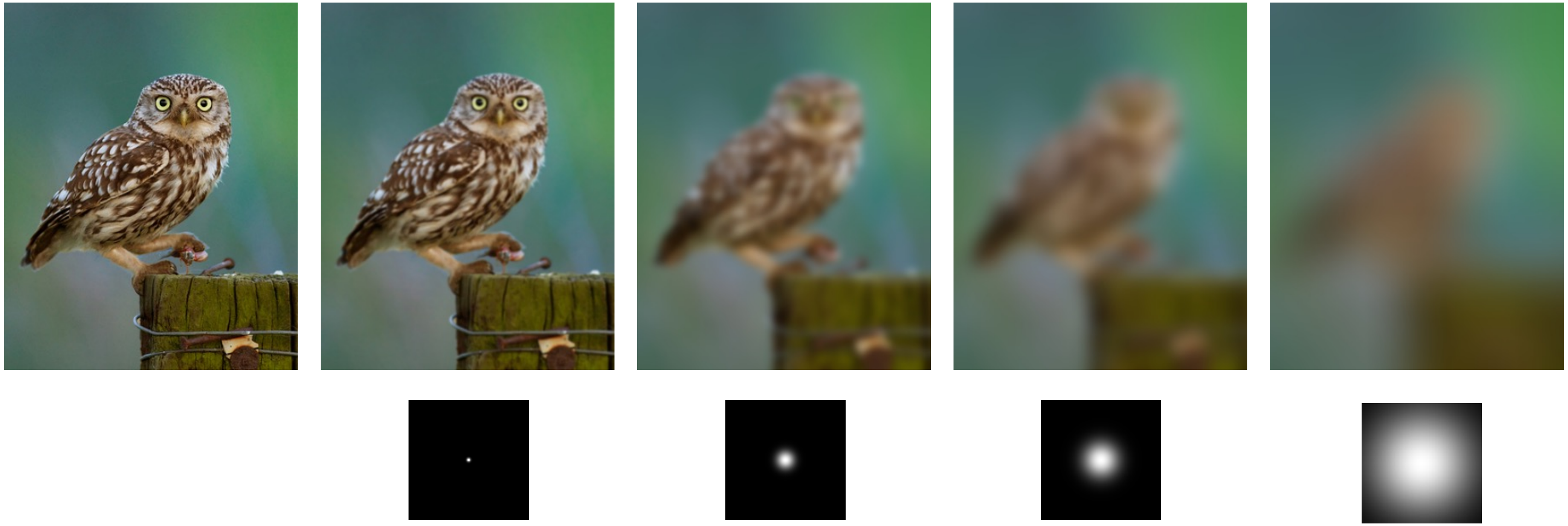




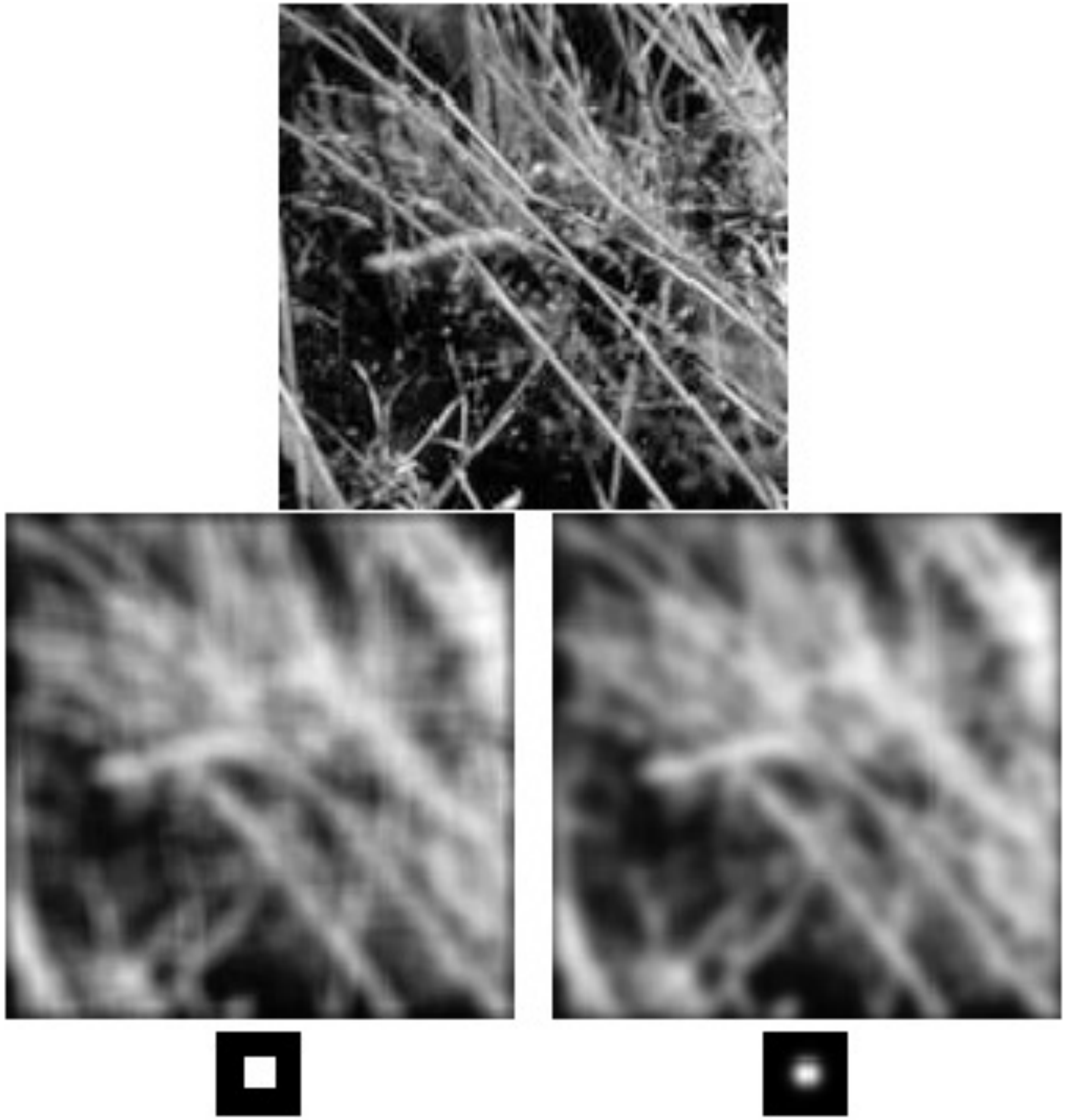
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Source: C. Rasmussen

Gaussian filters

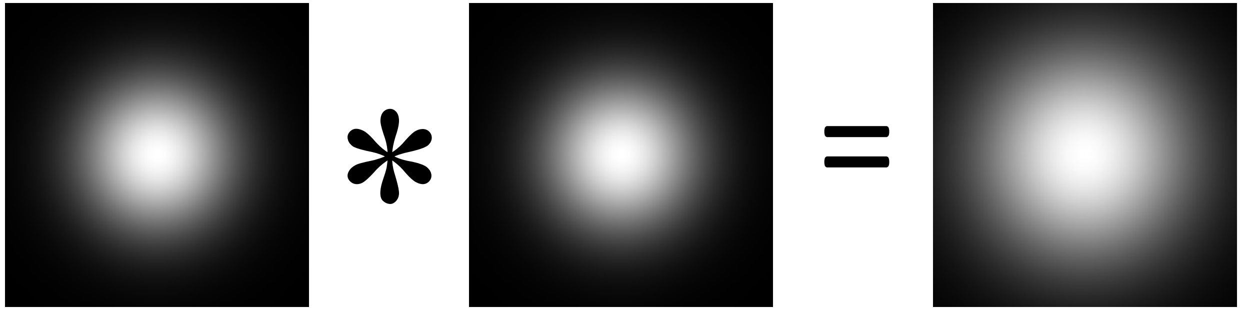


Mean vs. Gaussian filtering



Gaussian filter

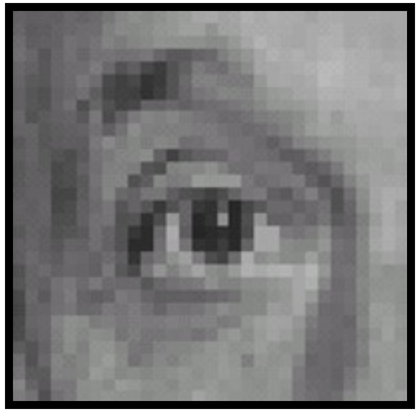
- Removes “high-frequency” components from the image (low-pass filter)
- Convolution with self is another Gaussian



- Convolving twice with Gaussian kernel of width σ =
  Convolving once with kernel of width $\sigma\sqrt{2}$

Source: K. Grauman

Linear filters: examples



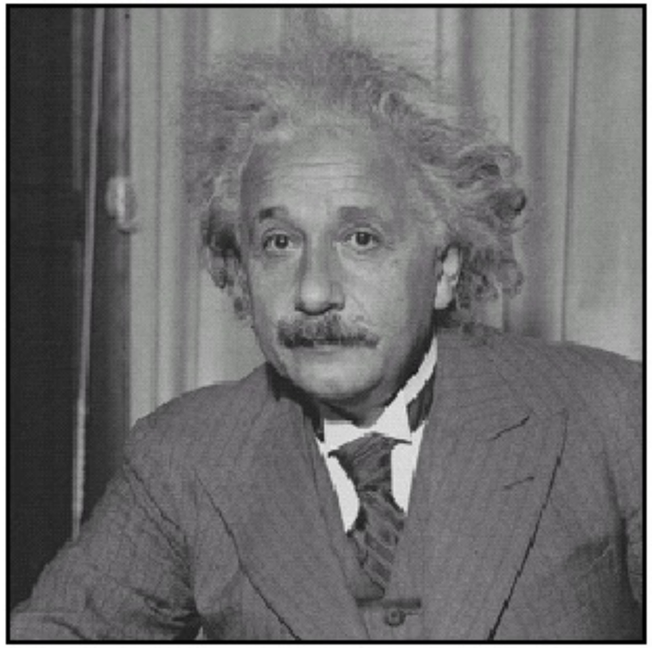
Original

$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) =$$

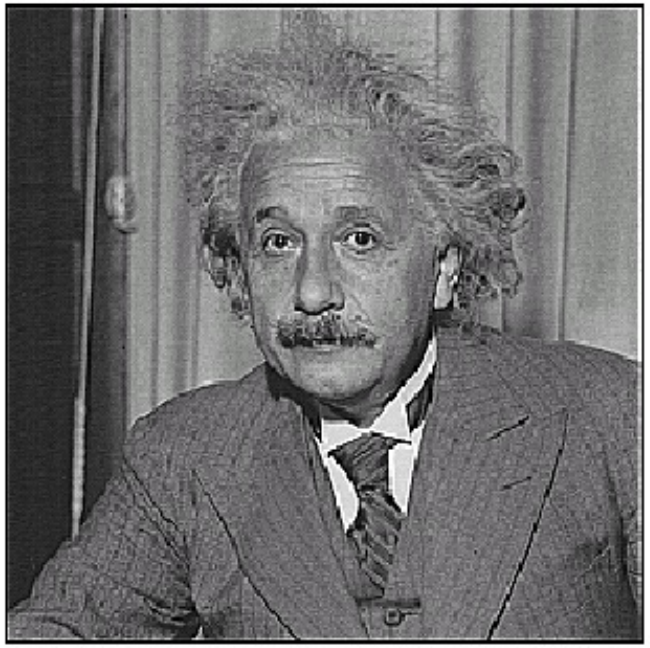
Sharpening filter
(accentuates edges)

Source: D. Lowe

Sharpening



before



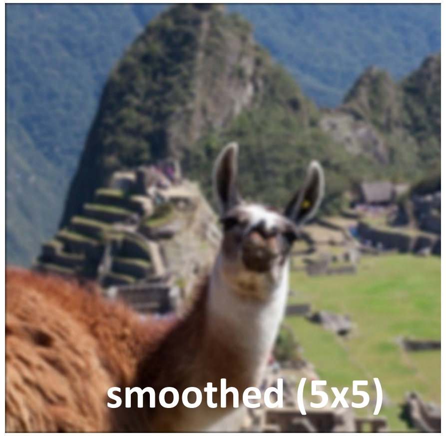
after

Source: D. Lowe

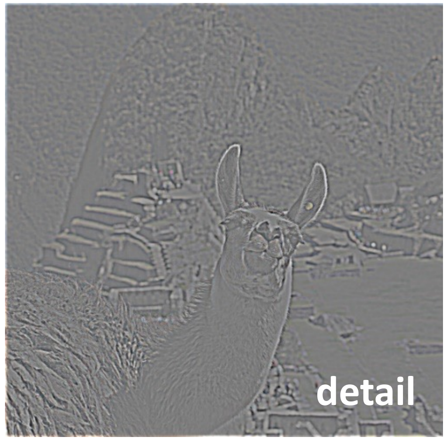
Sharpening

- What does blurring take away?

–



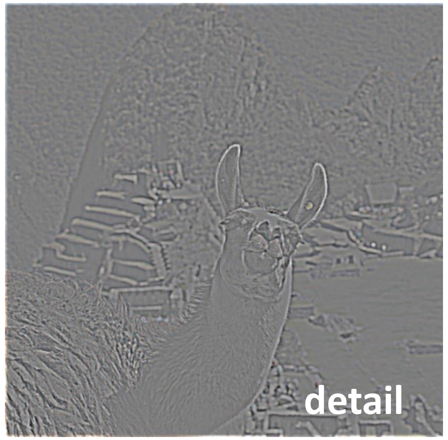
=



(This "detail extraction" operation is also called a ***high-pass filter***)

Let's add it back:

+ α



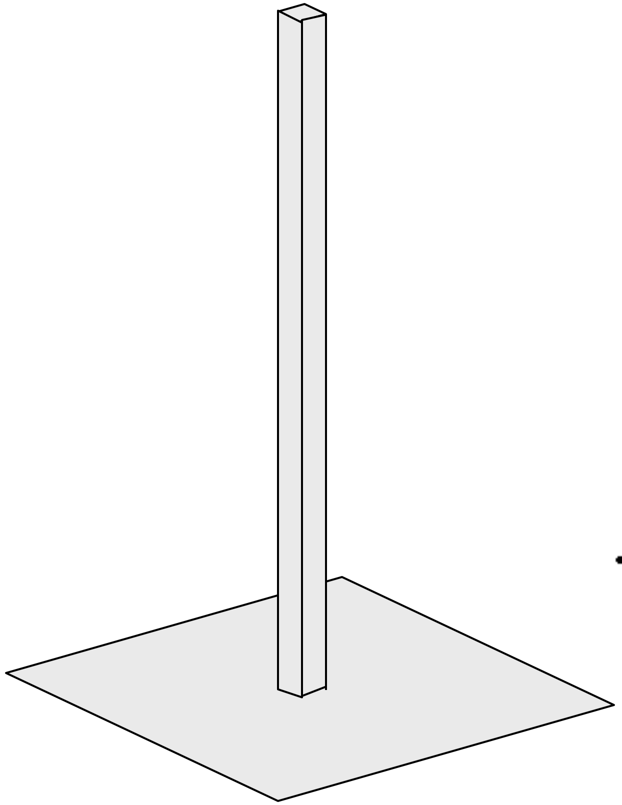
=

Photo credit: https://www.flickr.com/photos/geezaweezer/16089096376/

Sharpen filter

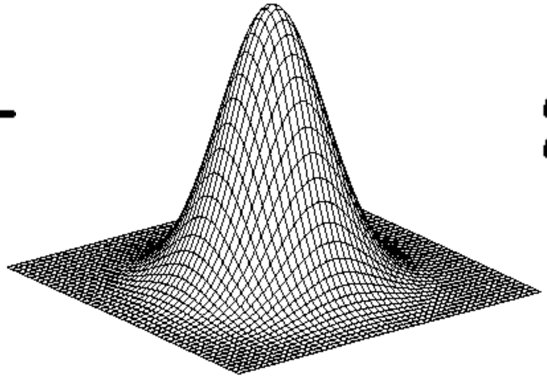
$$\underset{\text{image}}{F} + \alpha\,(F - \underbrace{F * H}_{\text{blurred image}}) =$$

unit impulse (identity kernel with single 1 in center, zeros elsewhere)



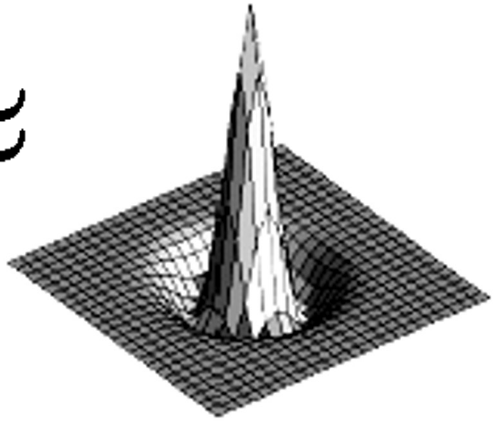
scaled impulse

$-$



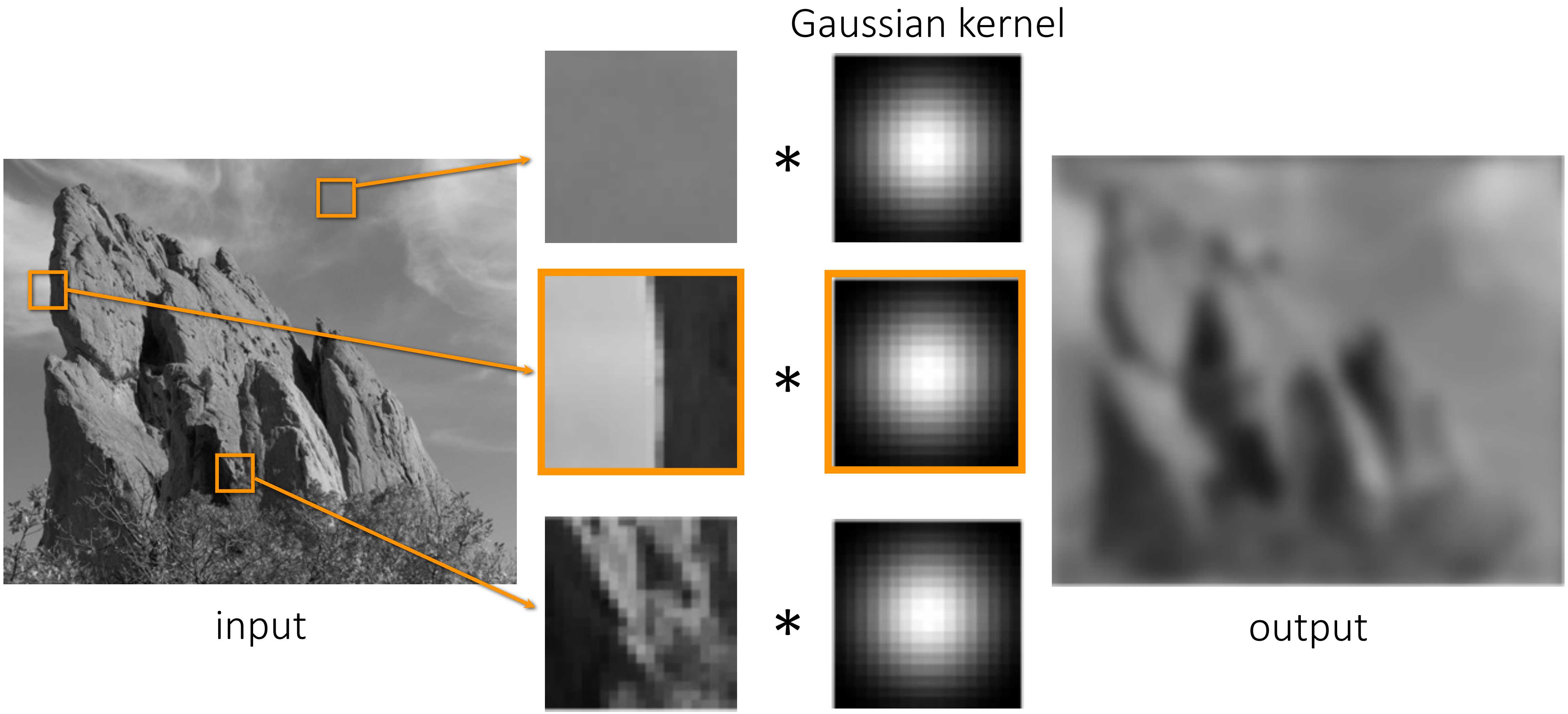
Gaussian

$\approx$



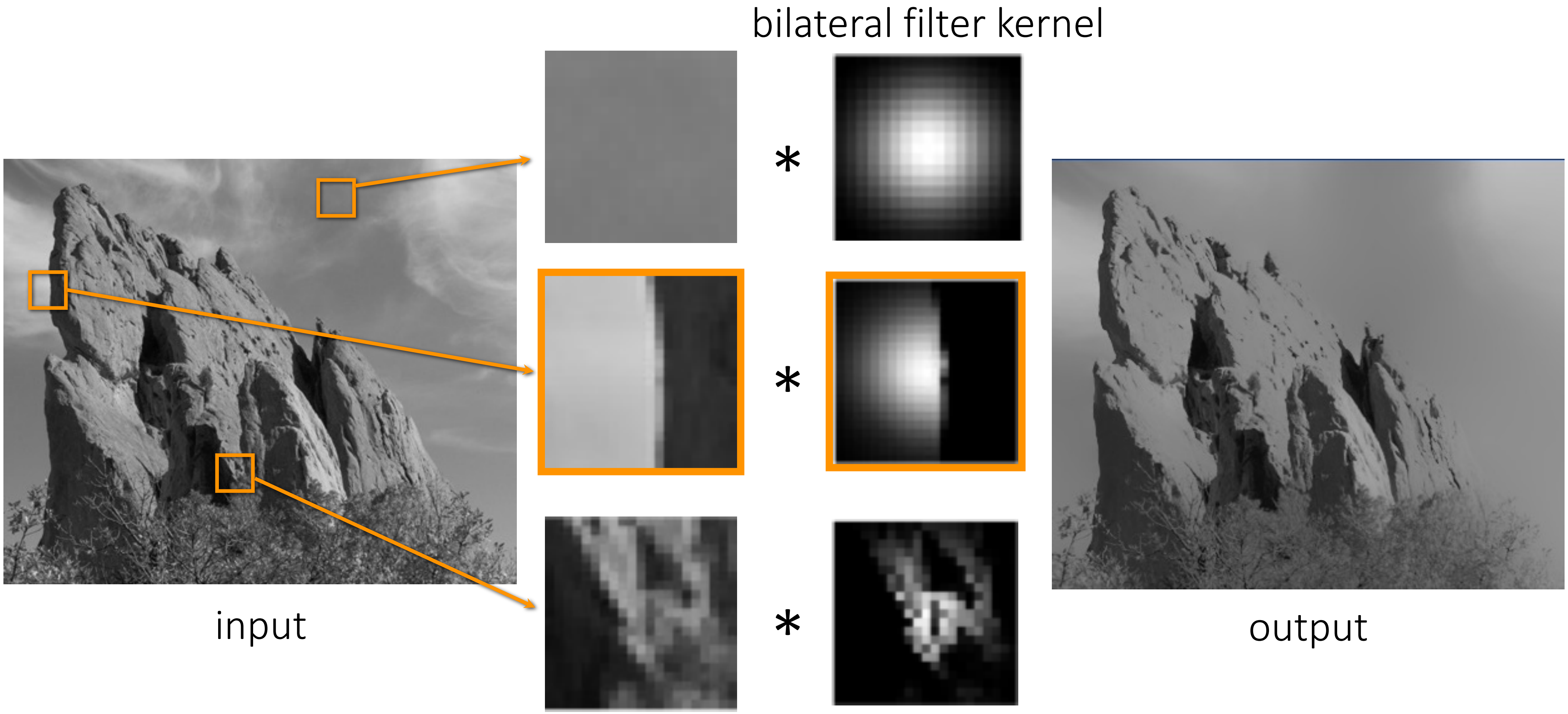
Sharpen filter

The problem with Gaussian filtering



Blur kernel averages across edges

The bilateral filtering solution



Do not blur if there is an edge! How does it do that?

Bilateral filtering vs Gaussian filtering

Gaussian filtering

$$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$$

Bilateral filtering

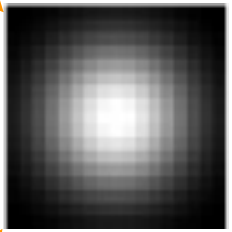
$$h[m,n] = \frac{1}{W_{mn}} \sum_{k,l} g[k,l] r_{mn}[k,l] f[m+k, n+l]$$

Bilateral filtering vs Gaussian filtering

Gaussian filtering

$$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$$

σ_s



Spatial weighting: favor *nearby* pixels

Bilateral filtering

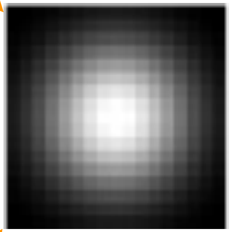
$$h[m,n] = \frac{1}{W_{mn}} \sum_{k,l} g[k,l] r_{mn}[k,l] f[m+k, n+l]$$

Bilateral filtering vs Gaussian filtering

Gaussian filtering

$$h[m,n] = \sum_{k,l} \boxed{g[k,l]} f[m+k, n+l]$$

σ_s

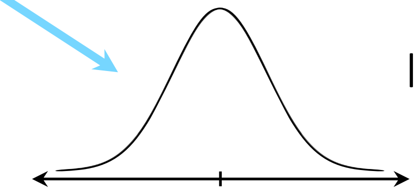


Spatial weighting:
favor *nearby* pixels

Bilateral filtering

$$h[m,n] = \frac{1}{W_{mn}} \sum_{k,l} \boxed{g[k,l]} \boxed{r_{mn}[k,l]} f[m+k, n+l]$$

σ_r



$x = f[m,n] - f[m+k, n+l]$

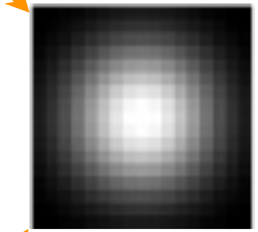
Intensity range weighting:
favor *similar* pixels

Bilateral filtering vs Gaussian filtering

Gaussian filtering

$$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$$

σ_s



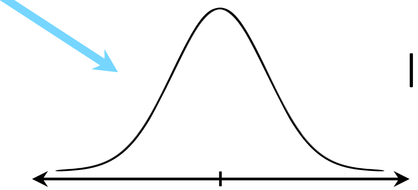
Spatial weighting: favor *nearby* pixels

Bilateral filtering

$$h[m,n] = \frac{1}{W_{mn}} \sum_{k,l} g[k,l] r_{mn}[k,l] f[m+k, n+l]$$

Normalization factor

σ_r



$x = f[m,n] - f[m+k, n+l]$

Intensity range weighting: favor *similar* pixels

Bilateral filtering vs Gaussian filtering

Gaussian filtering

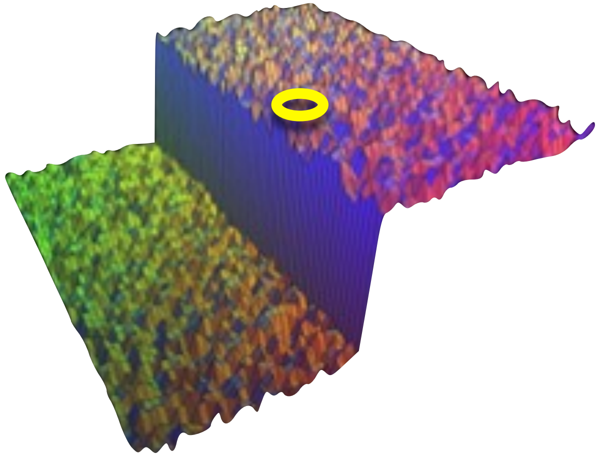
Smooths everything nearby (even edges)
Only depends on *spatial* distance

Bilateral filtering

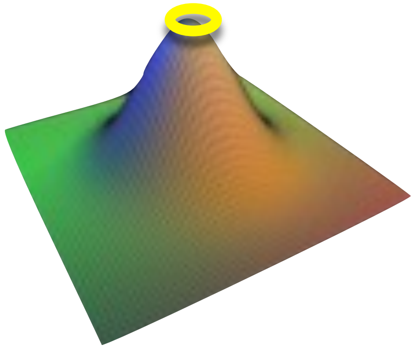
Smooths ‘close’ pixels in space and intensity
Depends on *spatial* and *intensity* distance

Gaussian filtering visualization

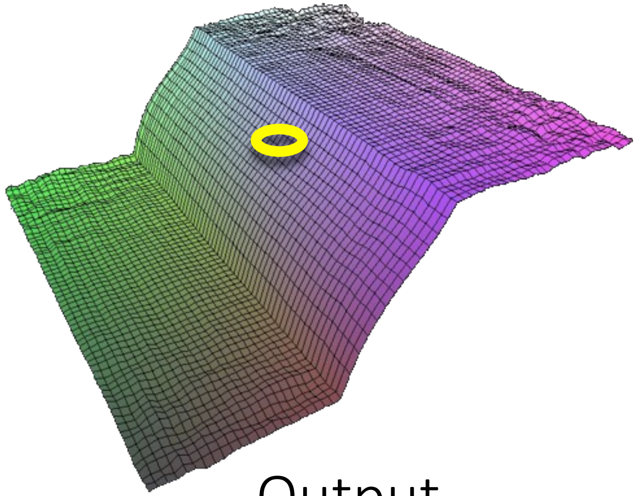
$$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$$



Input



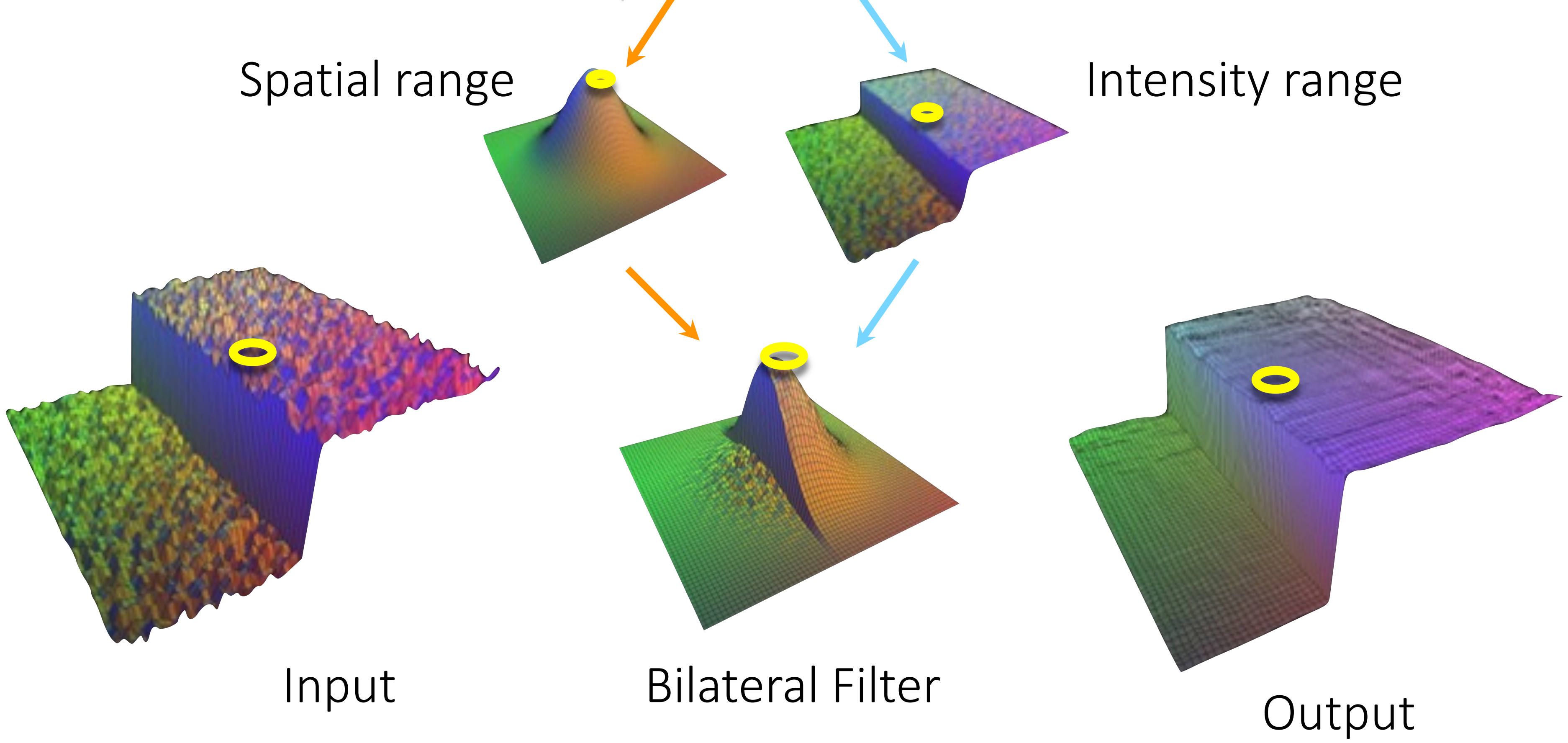
Gaussian Filter



Output

Bilateral filtering visualization

$$h[m,n] = \frac{1}{W_{mn}} \sum_{k,l} g[k,l] r_{mn}[k,l] f[m+k, n+l]$$



Denoising

noisy input

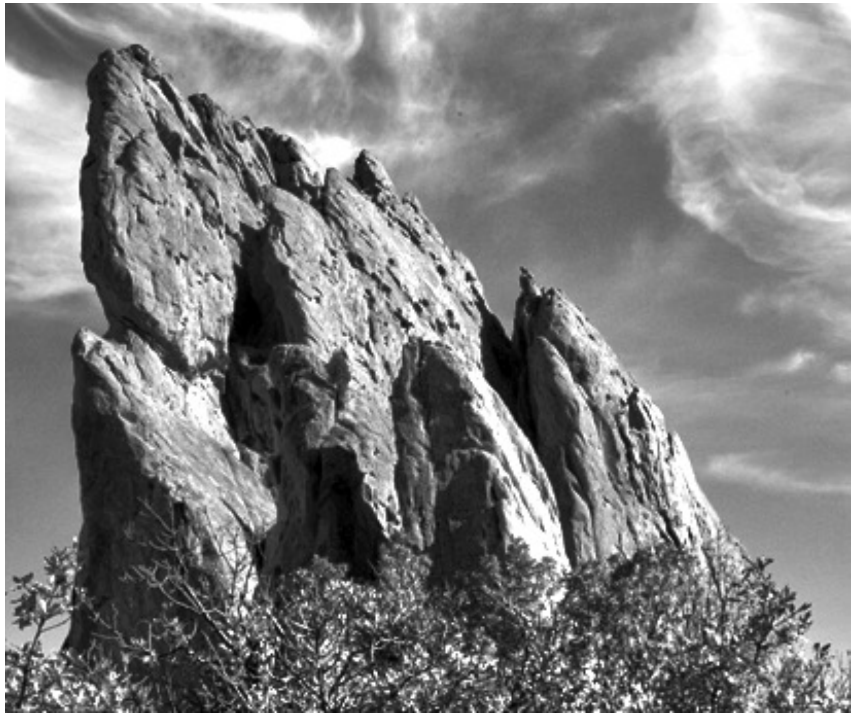
bilateral filtering

median filtering

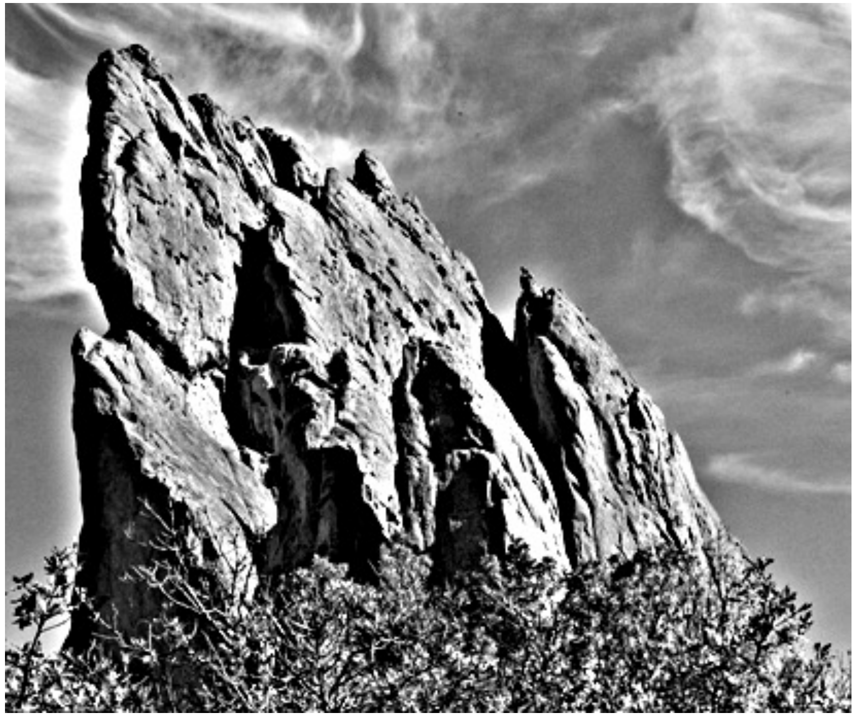
Contrast enhancement

How would you use Gaussian or bilateral filtering for sharpening?

input



sharpening based on bilateral filtering



sharpening based on Gaussian filtering

Photo retouching

Photo retouching

original

digital pore removal (aka bilateral filtering)

Before

After

Close-up comparison

original

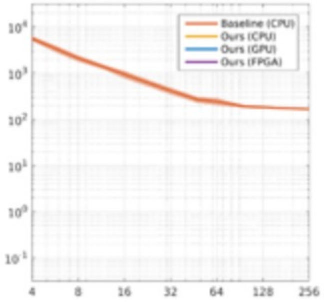
digital pore removal (aka bilateral filtering)

Cartoonization

input

cartoon rendition

Actively used in various research problems

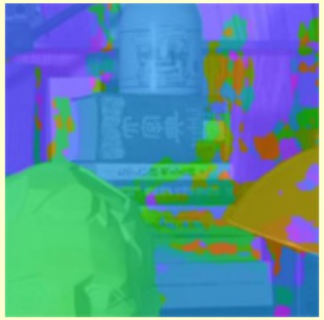


A Hardware-Friendly Bilateral Solver for Real-Time Virtual Reality Video
Amrita Mazumdar, Armin Alaghi, **Jonathan T. Barron**, David Gallup, Luis Ceze, Mark Oskin, Steven M. Seitz
High-Performance Graphics (HPG), 2017
project page

A reformulation of the bilateral solver can be implemented efficiently on GPUs and FPGAs.

Deep Bilateral Learning for Real-Time Image Enhancement
Michaël Gharbi, Jiawen Chen, **Jonathan T. Barron**, Samuel W. Hasinoff, Frédo Durand
SIGGRAPH, 2017
project page / video / bibtex / press

By training a deep network in bilateral space we can learn a model for high-resolution and real-time image enhancement.



The Fast Bilateral Solver
Jonathan T. Barron, Ben Poole
ECCV, 2016 **(Oral Presentation, Best Paper Honorable Mention)**
arXiv / bibtex / video (they messed up my slides, use →) / keynote (or PDF) / code / depth super-res results / reviews

Our solver smooths things better than other filters and faster than other optimization algorithms, and you can backprop through it.

This lecture: Image Transformations & Filtering

- Point Processing
- Linear Filtering
- Sampling & Aliasing
- Image Derivatives
- Edge Detection

Image half-sizing

This image is too big to fit on the screen. How can we reduce it?

How to generate a half-sized version?

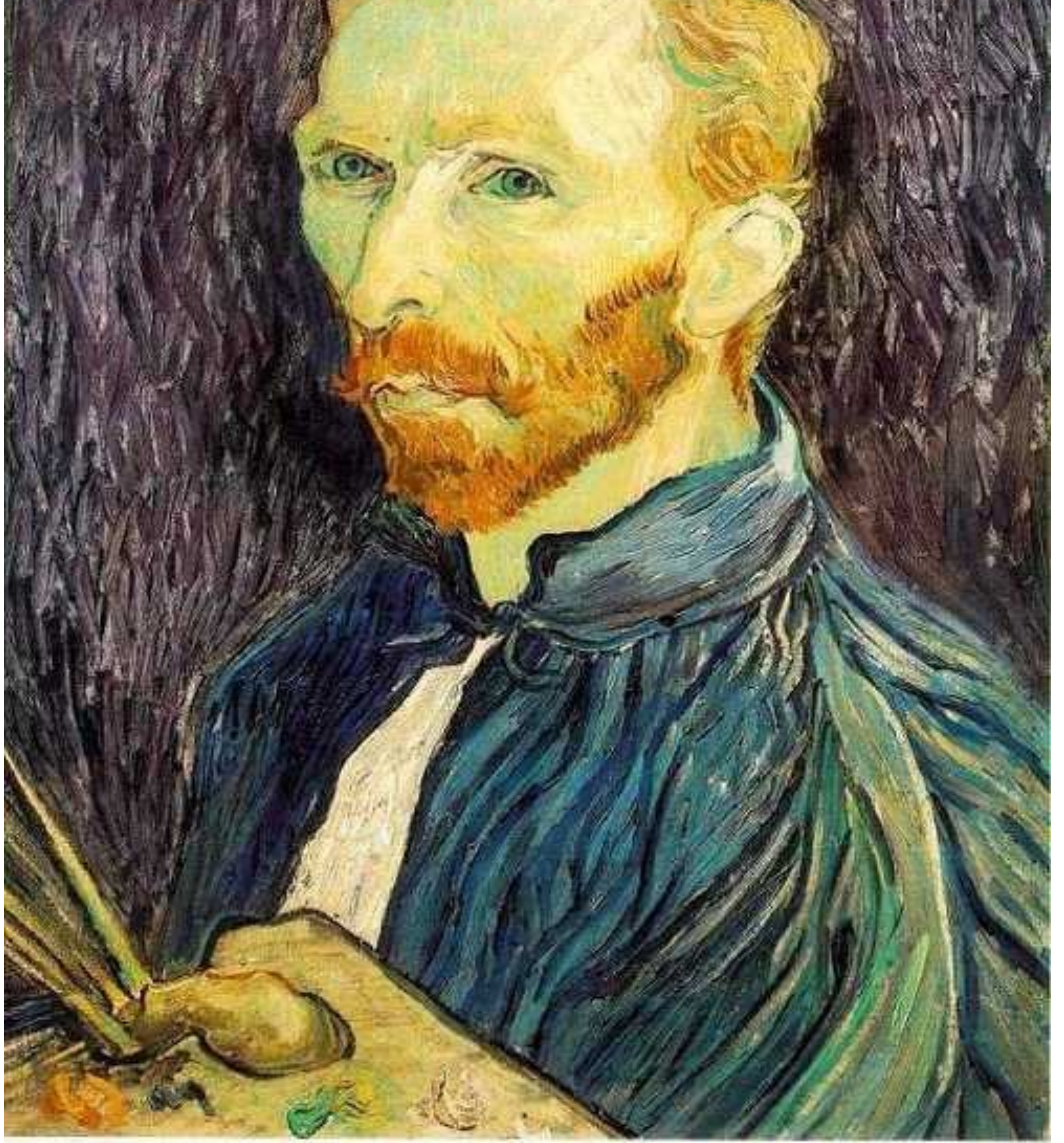
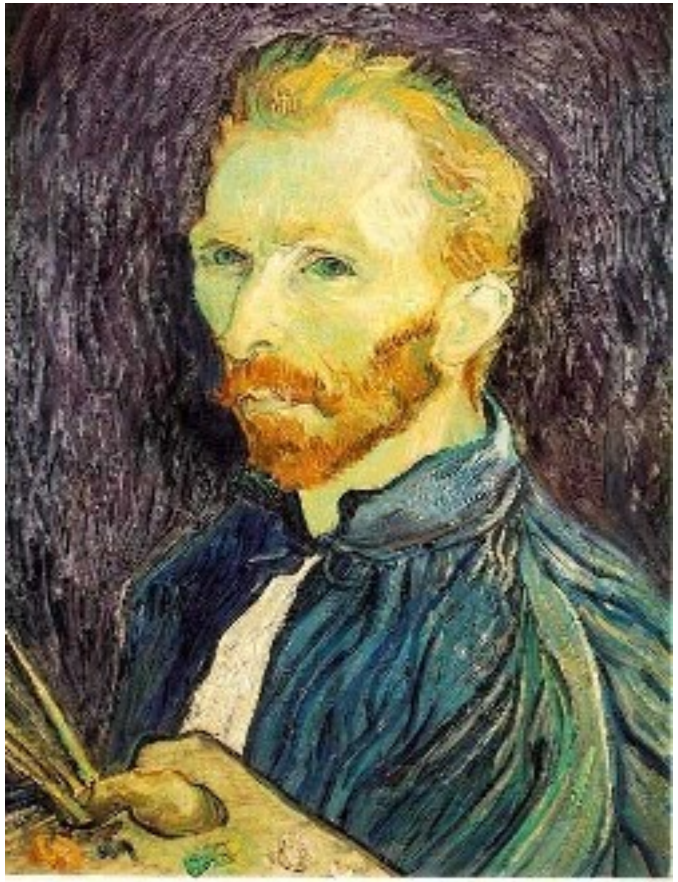


Image sub-sampling



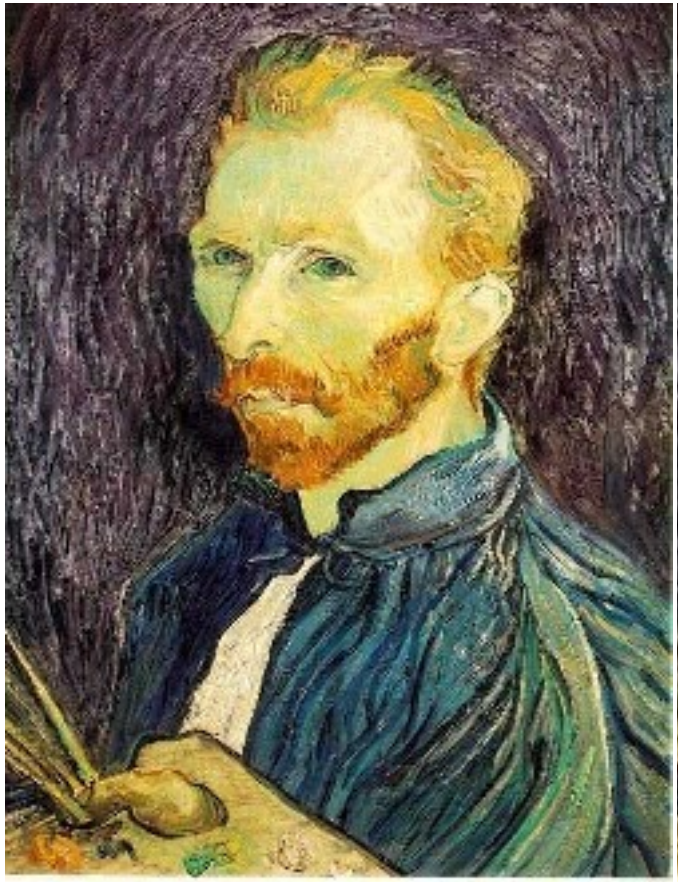
Throw away every other row and column to create a 1/2 size image
- called *image sub-sampling*

1/4

1/8

Slide by Steve Seitz

Image sub-sampling

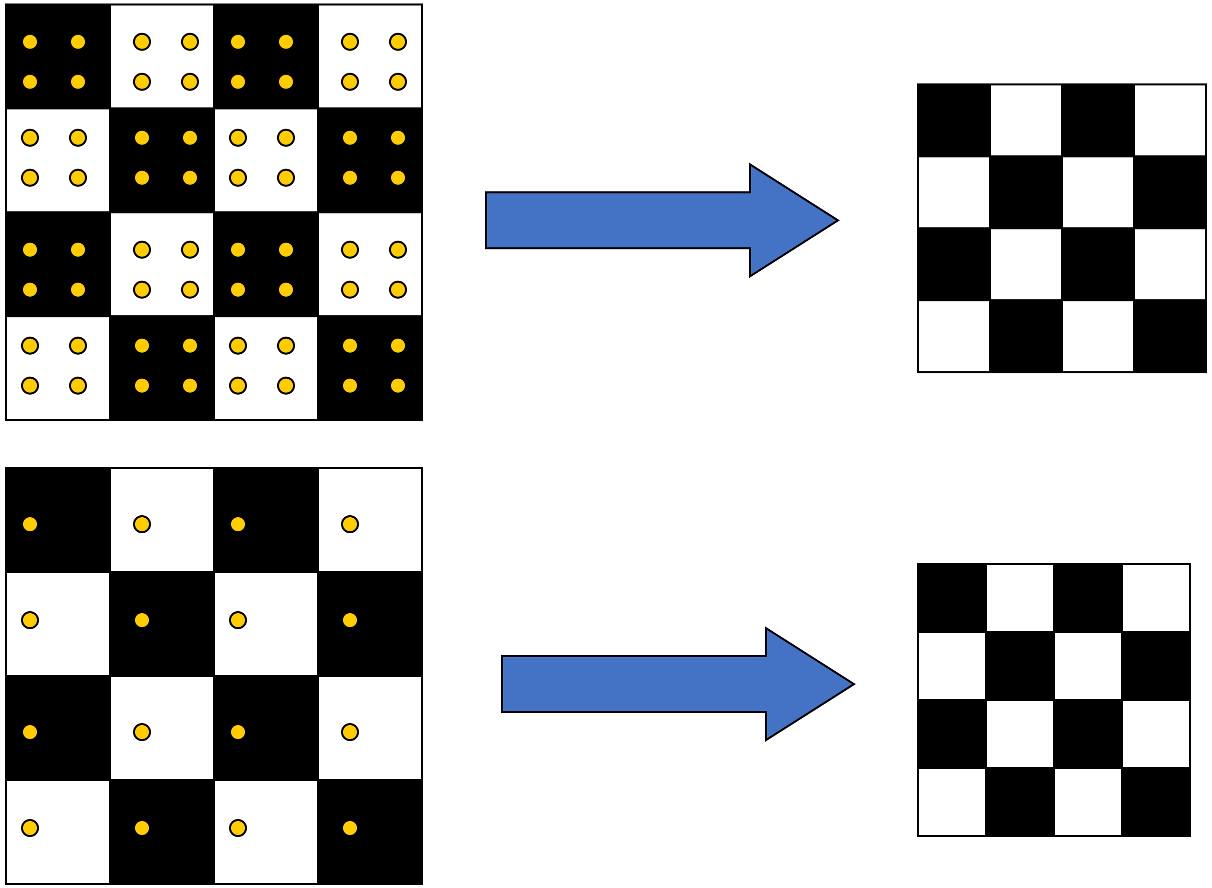


1/2 1/4 (2x zoom) 1/8 (4x zoom)

Aliasing! What do we do?

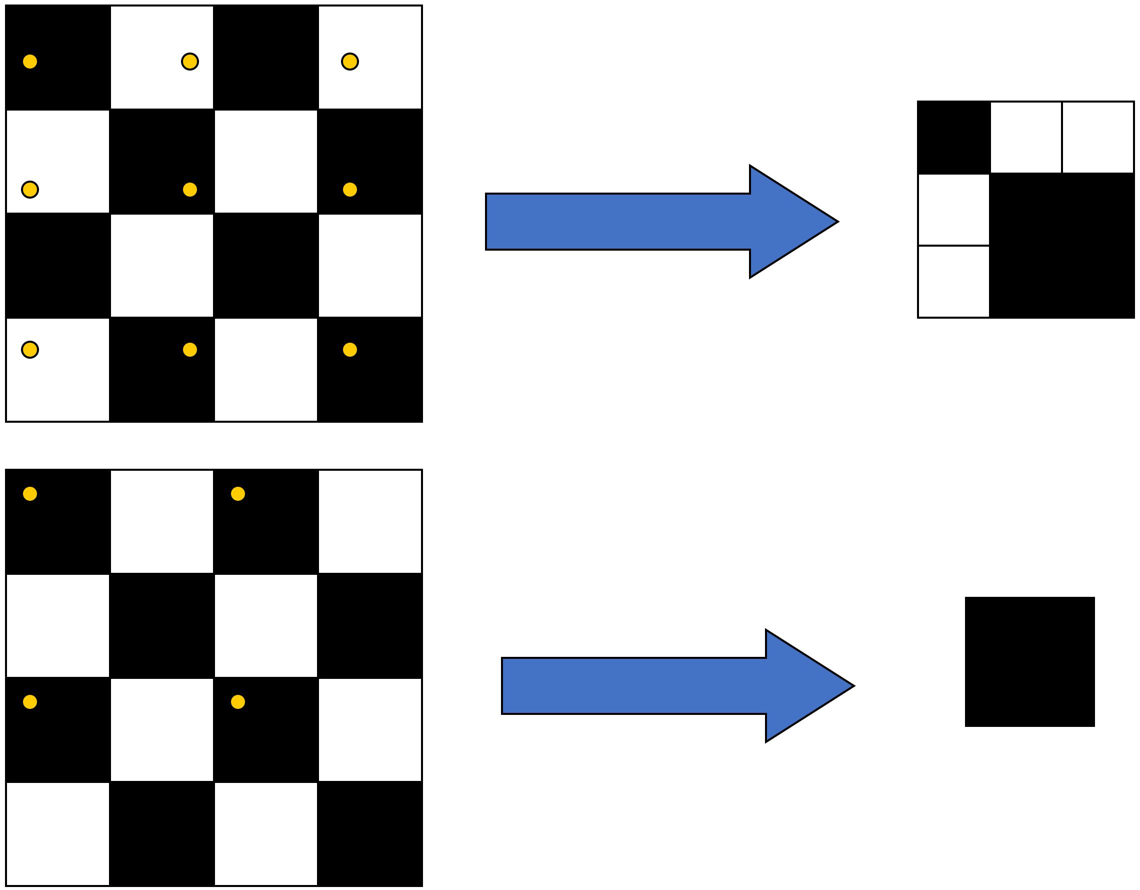
Slide by Steve Seitz

Sampling an image



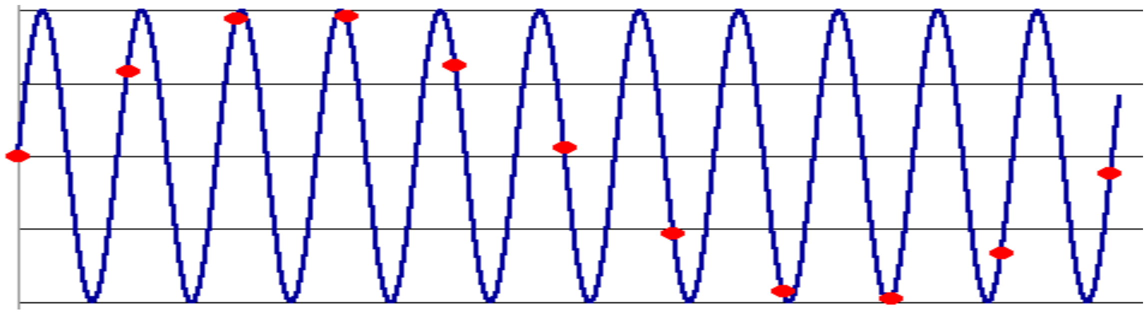
Examples of GOOD sampling

Undersampling



Examples of BAD sampling -> Aliasing

Aliasing

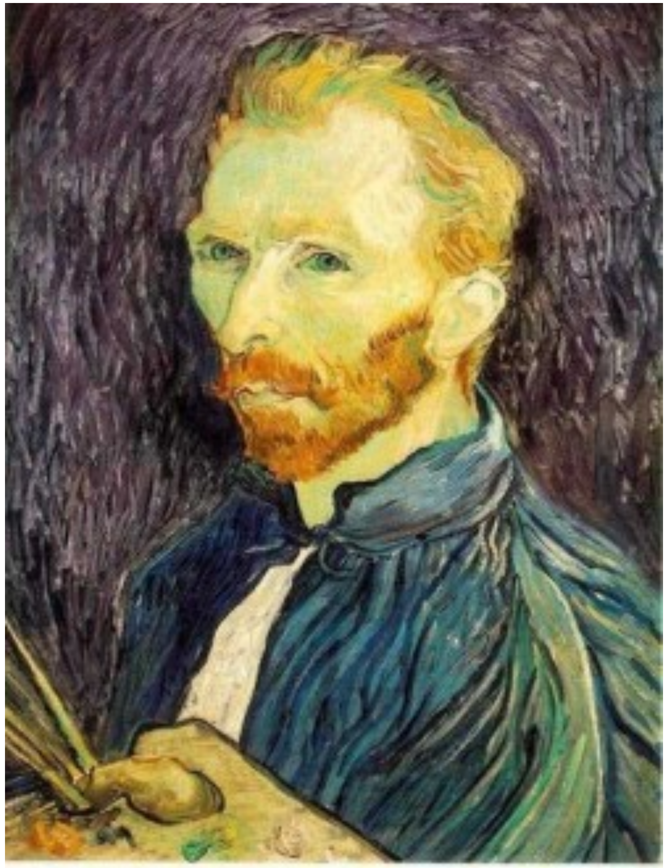


- Occurs when your sampling rate is not high enough to capture the amount of detail in your image
- Can give you the wrong signal/image—an *alias*

- To do sampling right, need to understand the structure of your signal/image
- Enter Monsieur Fourier…
  - "But what is the Fourier Transform? A visual introduction." (We will learn it next class!) https://www.youtube.com/watch?v=spUNpyF58BY
- To avoid aliasing:
  - sampling rate ≥ 2 * max frequency in the image
    - said another way: ≥ two samples per cycle
  - This minimum sampling rate is called the **Nyquist rate**

Source: L. Zhang

Gaussian (lowpass) pre-filtering



Gaussian 1/2

G 1/4

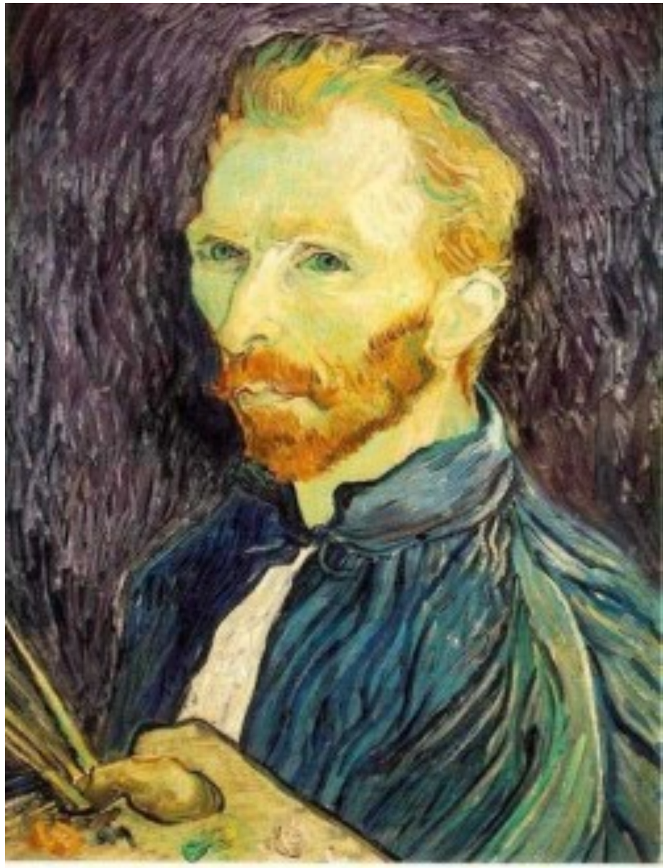
G 1/8

Solution: filter the image, *then* subsample

- Filter size should double for each ½ size reduction.

Slide by Steve Seitz

Subsampling with Gaussian pre-filtering



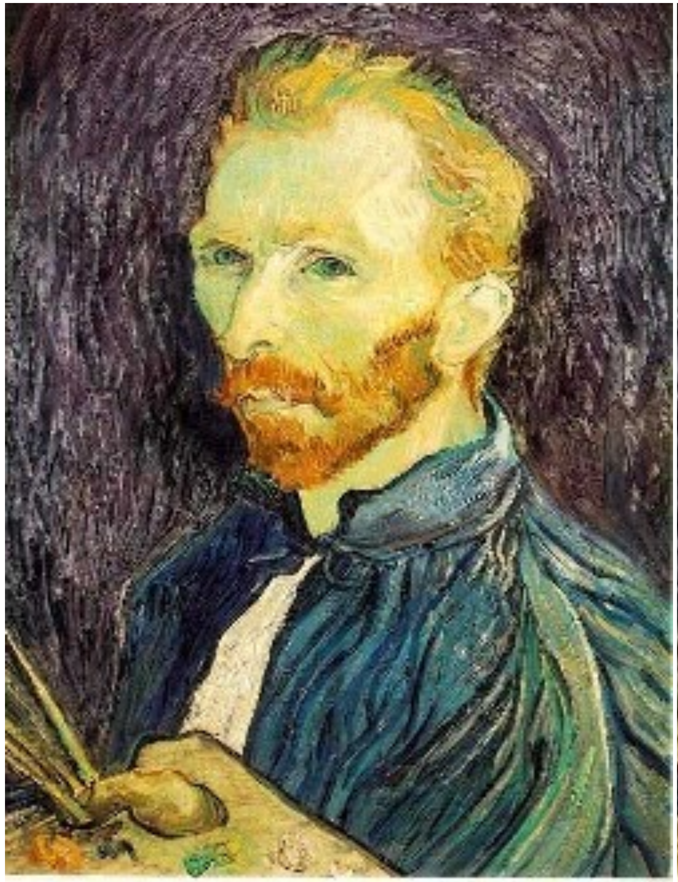
Gaussian 1/2

G 1/4

G 1/8

Slide by Steve Seitz

Compare with...



1/2

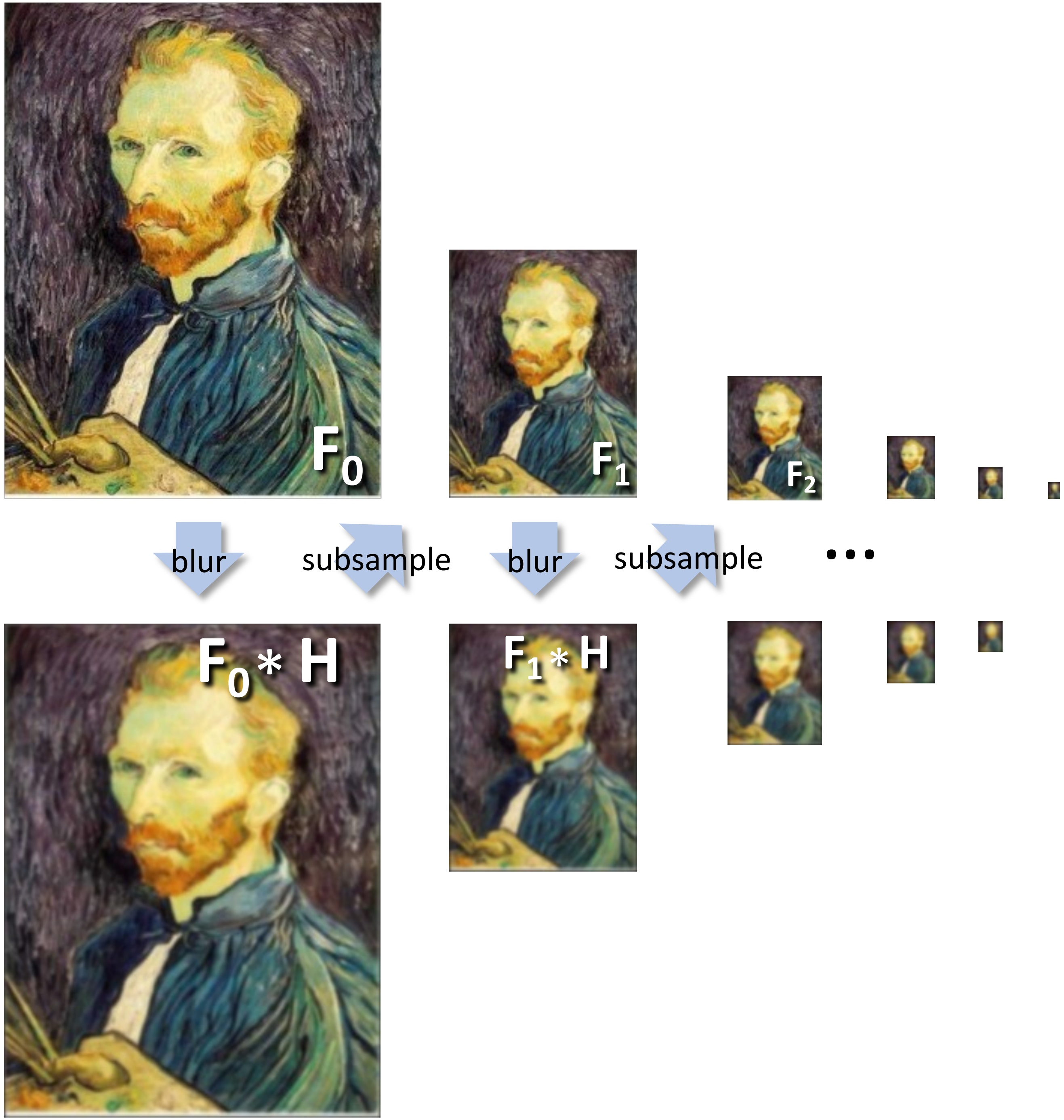
1/4 (2x zoom)

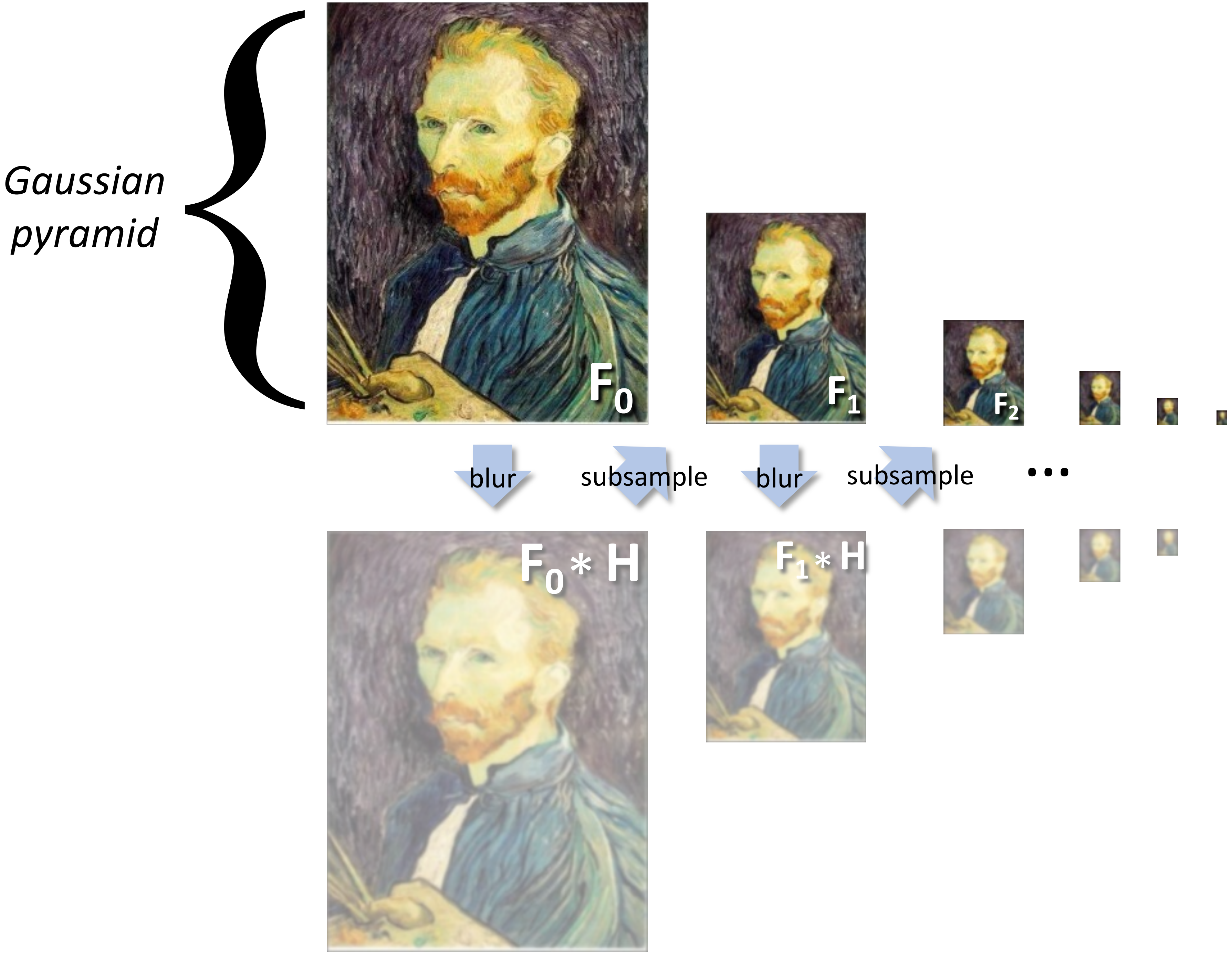
1/8 (4x zoom)

Slide by Steve Seitz

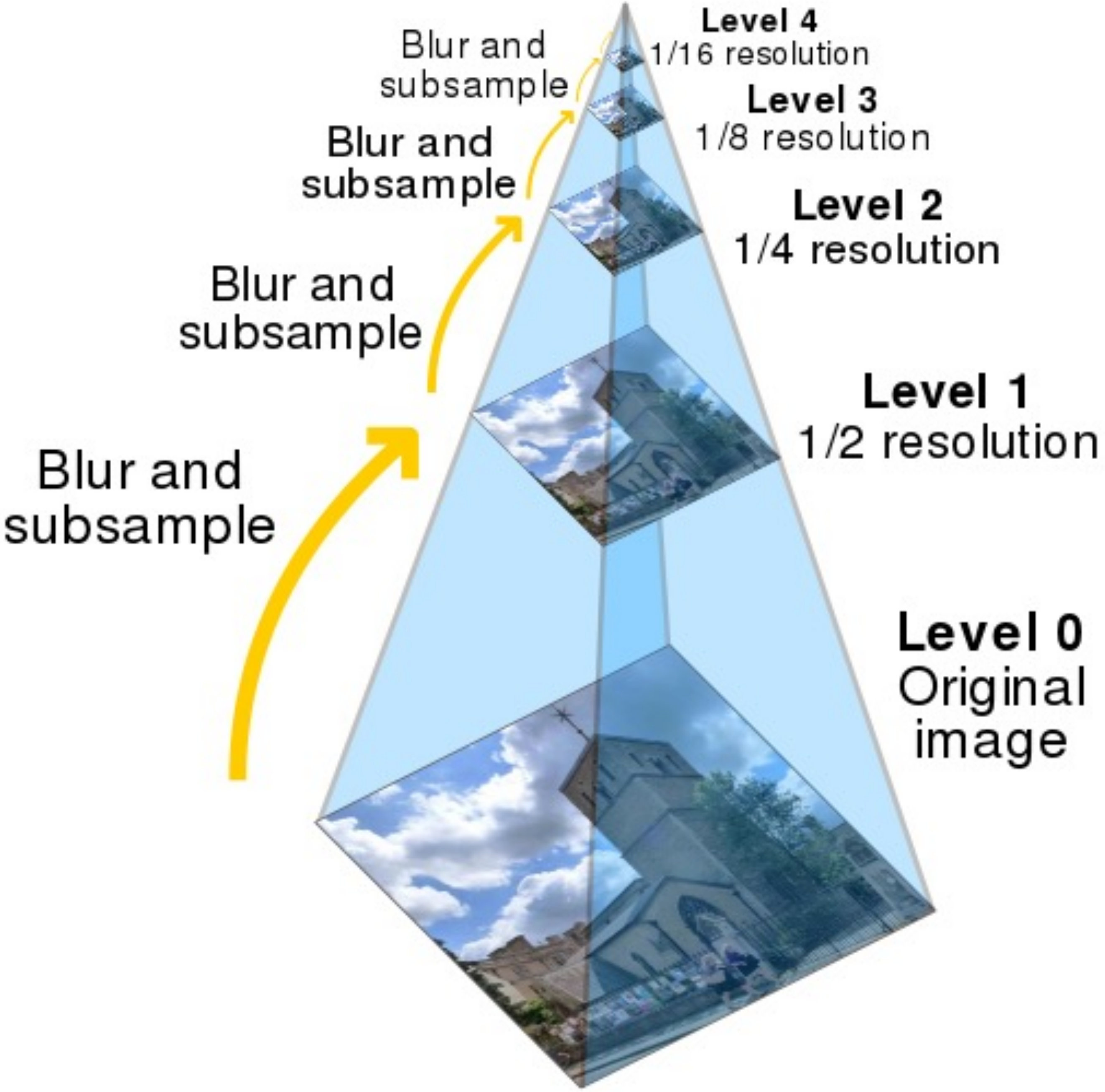
Gaussian pre-filtering

- Solution: filter the image, *then* subsample





Gaussian pyramid
F0
F1
F2
blur
subsample
blur
subsample
...
F0 * H
F1 * H



Similar to Gaussian Pyramids, there are Laplacian Pyramids.

- What do Laplace filters do?

We will learn about Laplacian pyramids and how it is used for blending and compositing images in Lecture 10.

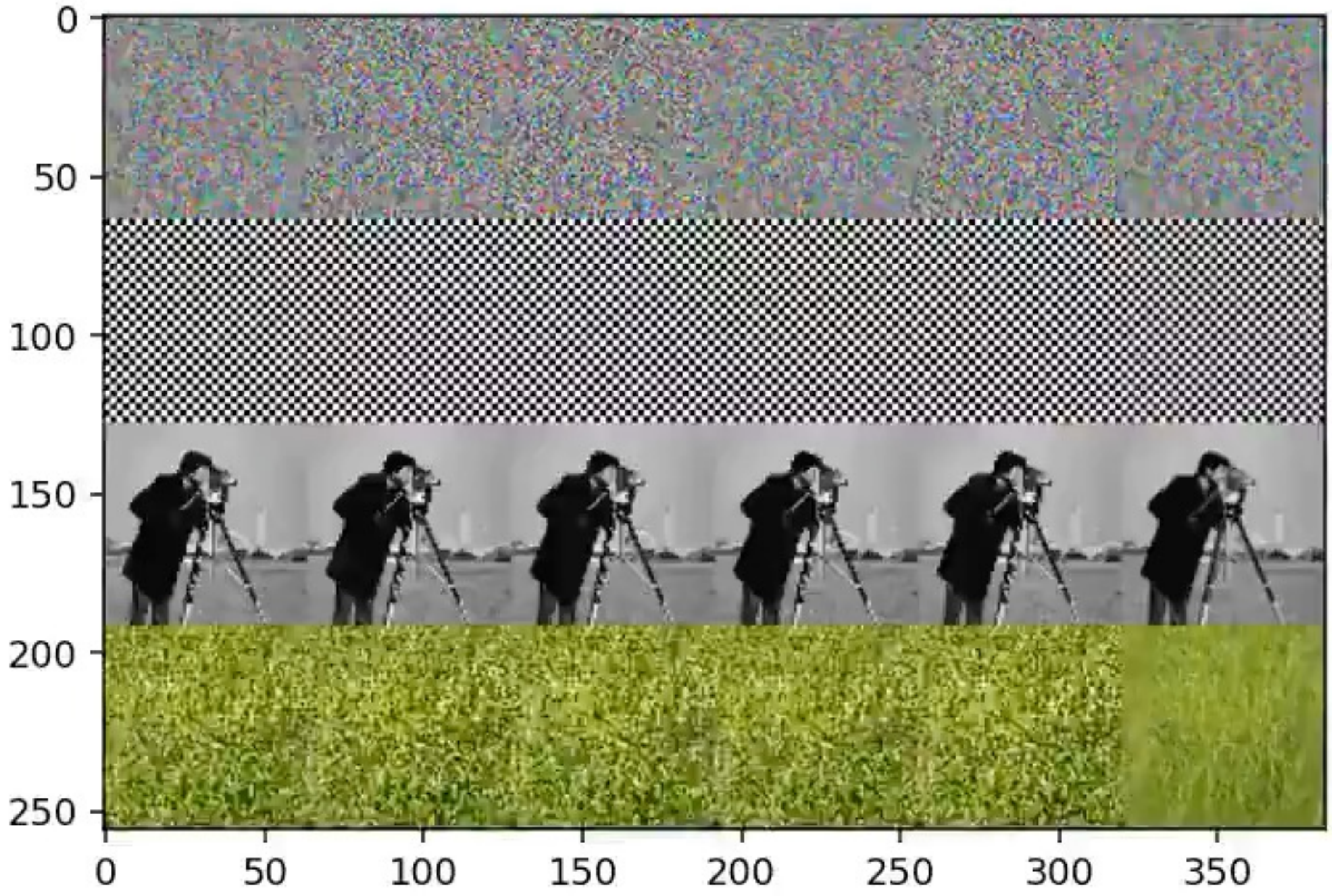
We will use something similar to stitching and blending images to form a panorama in our HWs!

What are they good for?

- Improve Search
  - Search over translations
    - Classic coarse-to-fine strategy
  - Search over scale
    - Template matching
    - E.g. find a face at different scales

A real problem!

128 x 128 → 64x64



Open-CV:
default, bicubic, Lanczos4

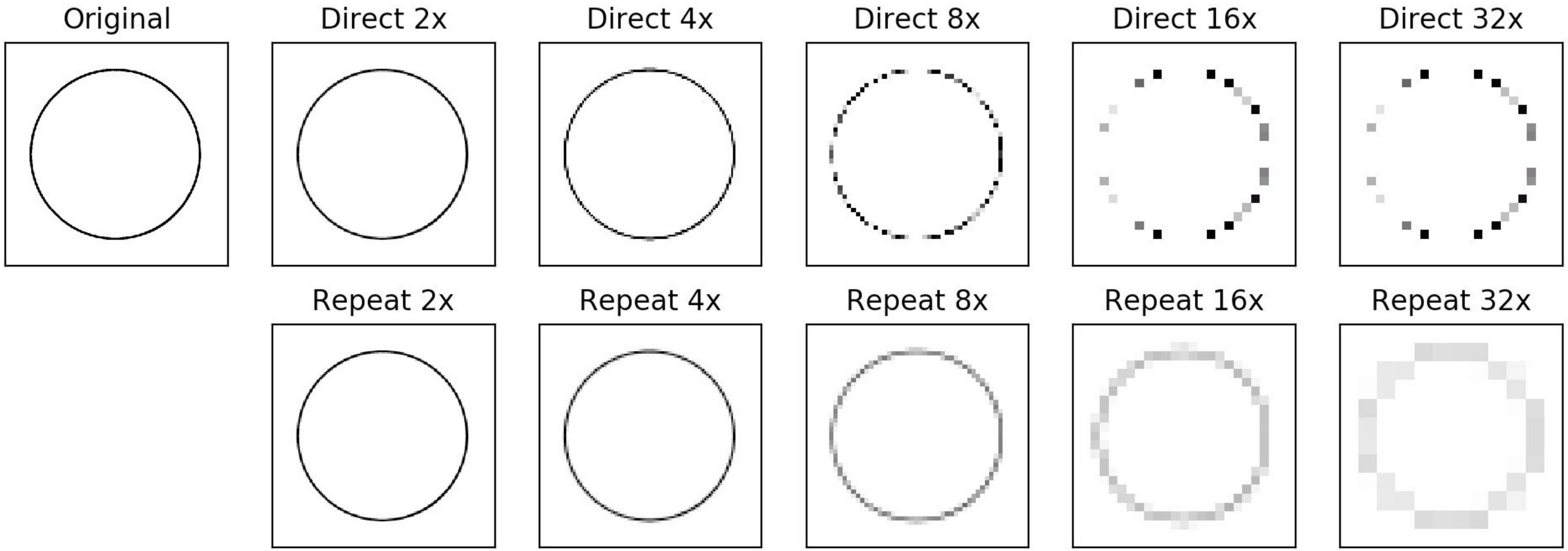
Pytorch:
bilinear, bicubic

PIL: Lanczos

Credit: @jaakkolehtinen

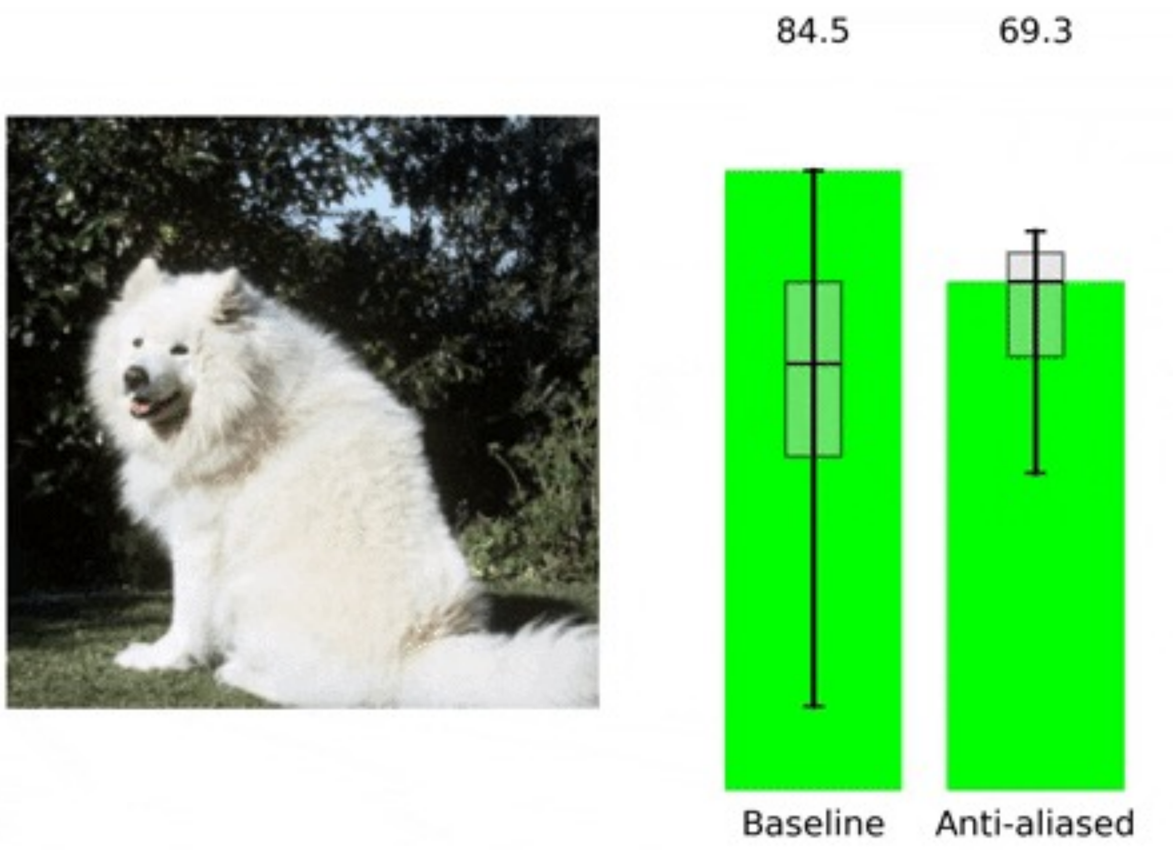
A real problem!

Bilinear downsampling with F.interpolate() in PyTorch



Credits: Richard Zhang

problems in NN too



Anti-aliasing in CNNs

```
pip install antialiased-cnns
```

Making Convolutional Networks Shift-Invariant Again,
Richard Zhang ICML 2019

Upsampling

- This image is too small for this screen: 
- How can we make it 10 times as big?
- Simplest approach:

  repeat each row

  and column 10 times
- ("Nearest neighbor

  interpolation")

Image interpolation

Original image:  x 10


Nearest-neighbor interpolation


Bilinear interpolation


Bicubic interpolation

Modern methods

(a) Bicubic (b) SRCNN (c) A+ (d) RAISR

(e) Bicubic (f) SRCNN (g) A+ (h) RAISR

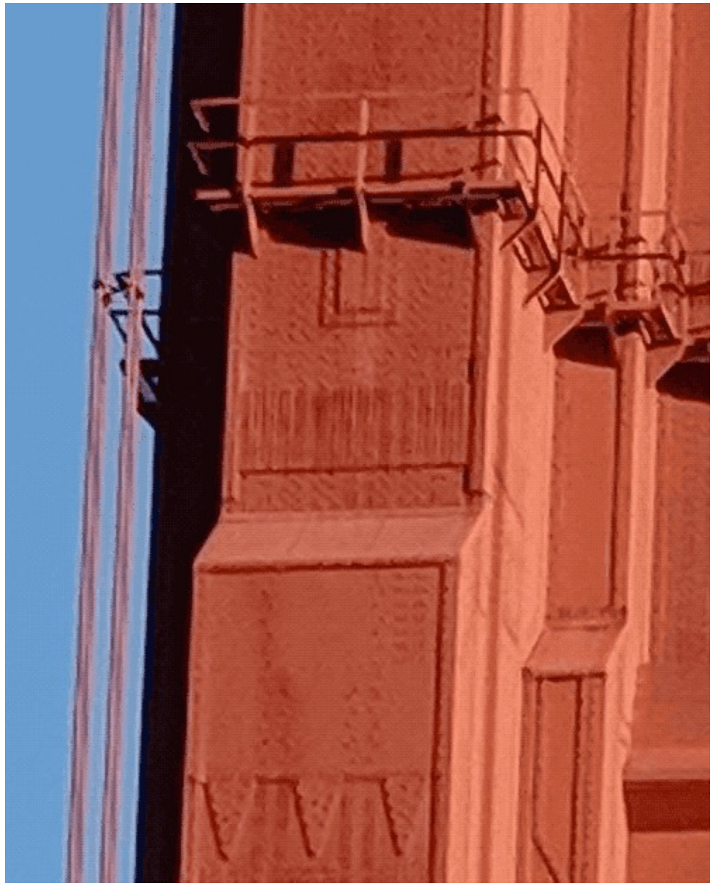
From Romano, et al: RAISR: Rapid and Accurate Image Super Resolution, https://arxiv.org/abs/1606.01299

Super-resolution with multiple images

- Can do better upsampling if you have multiple images of the scene taken with small (subpixel) shifts
- Some cellphone cameras (like the Google Pixel line) capture a **burst** of photos
- Can we use that burst for upsampling?

Google Pixel 3 Super Res Zoom

Effect of hand tremor as seen in a cropped burst of photos, after global alignment



Example photo with and without super res zoom (smart burst align and merge)

https://ai.googleblog.com/2018/10/see-better-and-further-with-super-res.html

This lecture: Image Transformations & Filtering

- Point Processing
- Linear Filtering
- Sampling & Aliasing
- Image Derivatives
- Edge Detection

Partial derivatives with convolution

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:

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

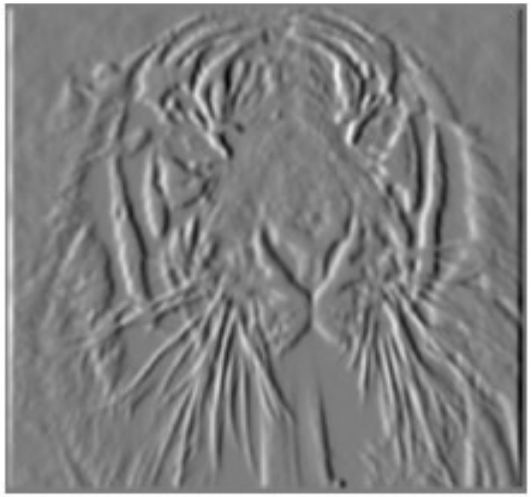
-1	1

Another one:

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

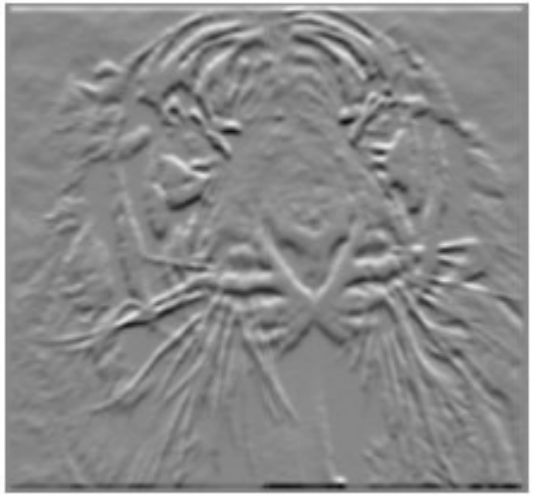
-1	0	1

Image Gradient



$$\frac{\partial f(x,y)}{\partial x}$$

-1	1



$$\frac{\partial f(x,y)}{\partial y}$$

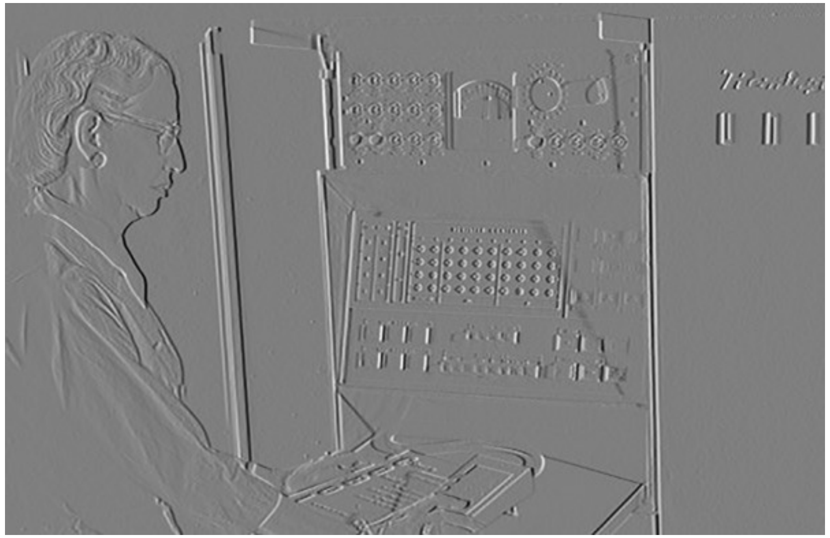
-1
1

or

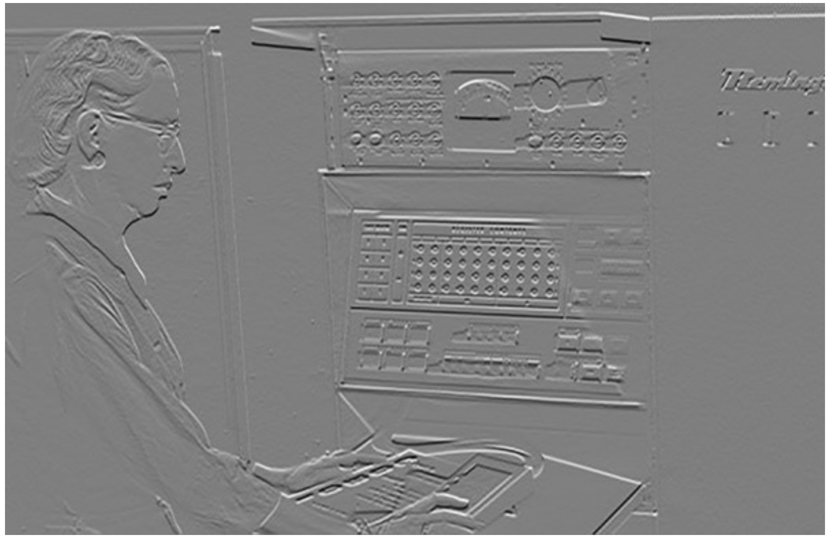
1
-1

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Partial Derivatives



$$\frac{\partial f(x,y)}{\partial x}$$



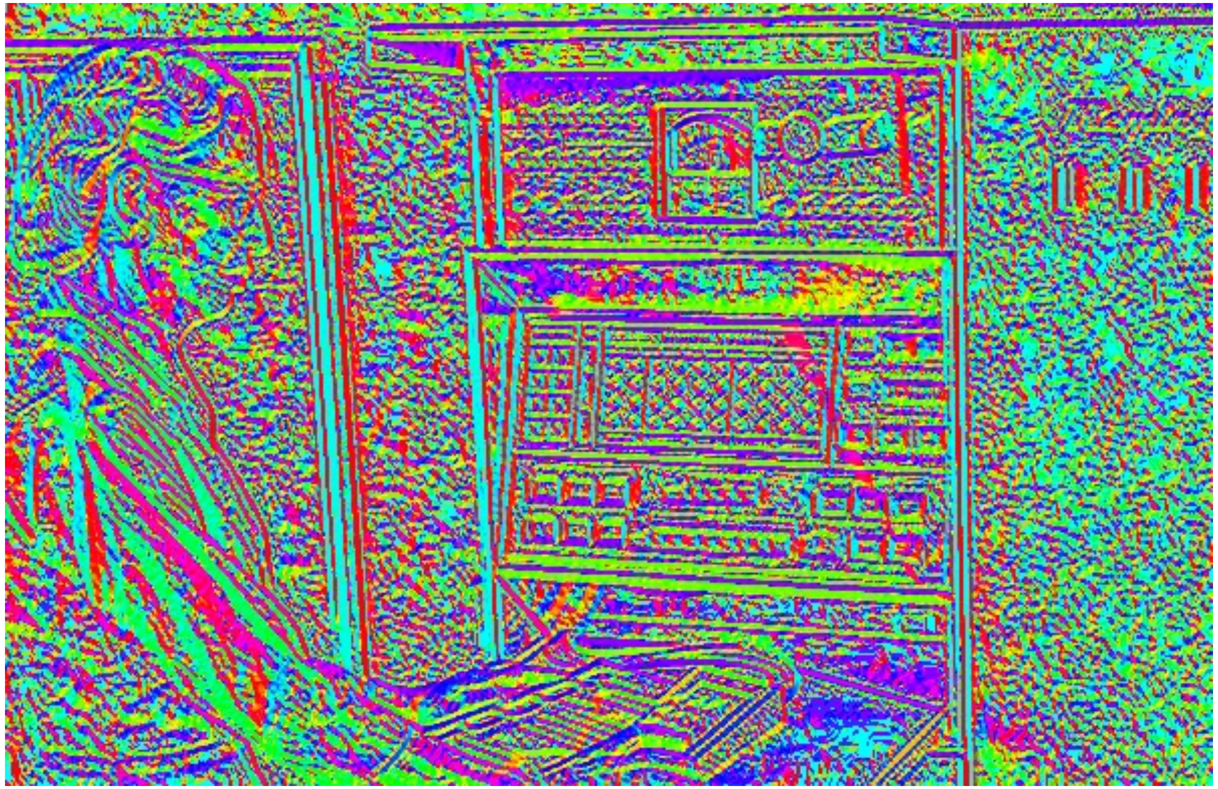
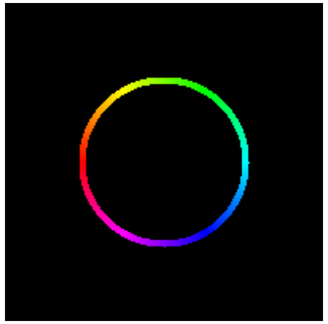
$$\frac{\partial f(x,y)}{\partial y}$$

Gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Gradient Orientation

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

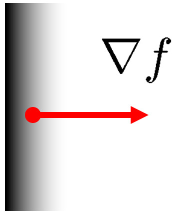
all the gradients

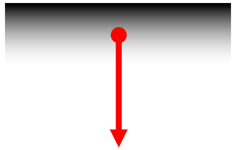
Source: D. Fouhey

Image gradient

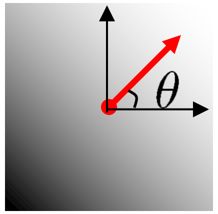
- The gradient of an image:

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

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



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



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

The gradient points in the direction of most rapid increase in intensity

- How does this direction relate to the direction of the edge?

The *edge strength* is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

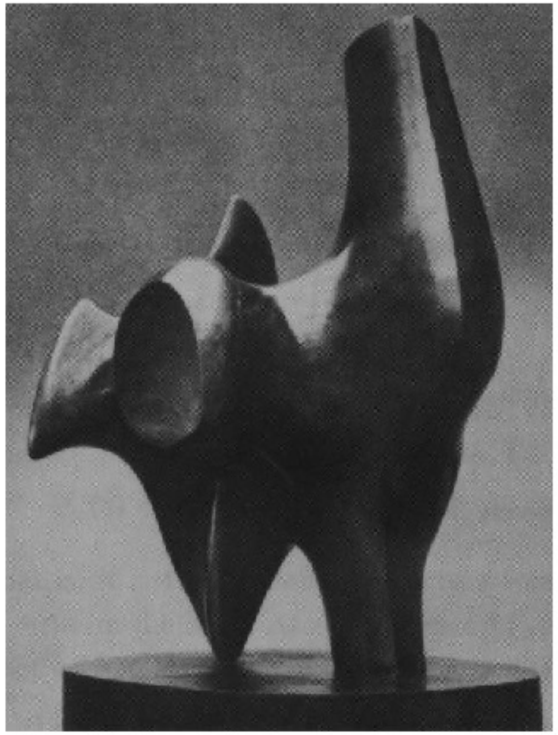
The gradient direction is given by $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

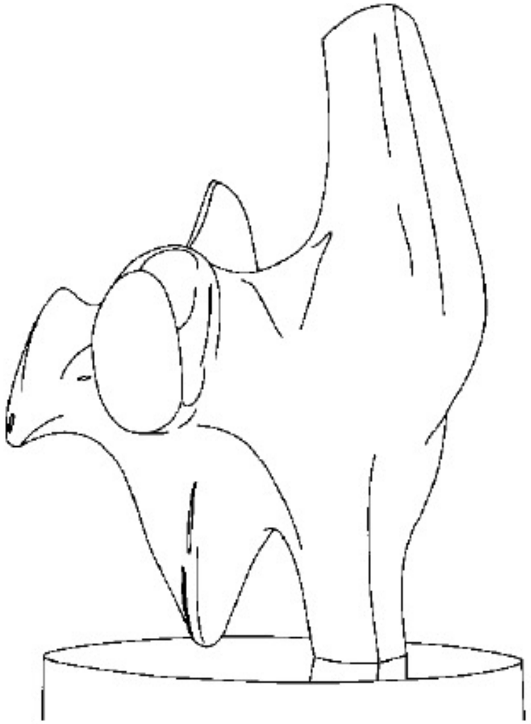
Source: Steve Seitz

This lecture: Image Transformations & Filtering

- Point Processing
- Linear Filtering
- Sampling & Aliasing
- Image Derivatives
- Edge Detection

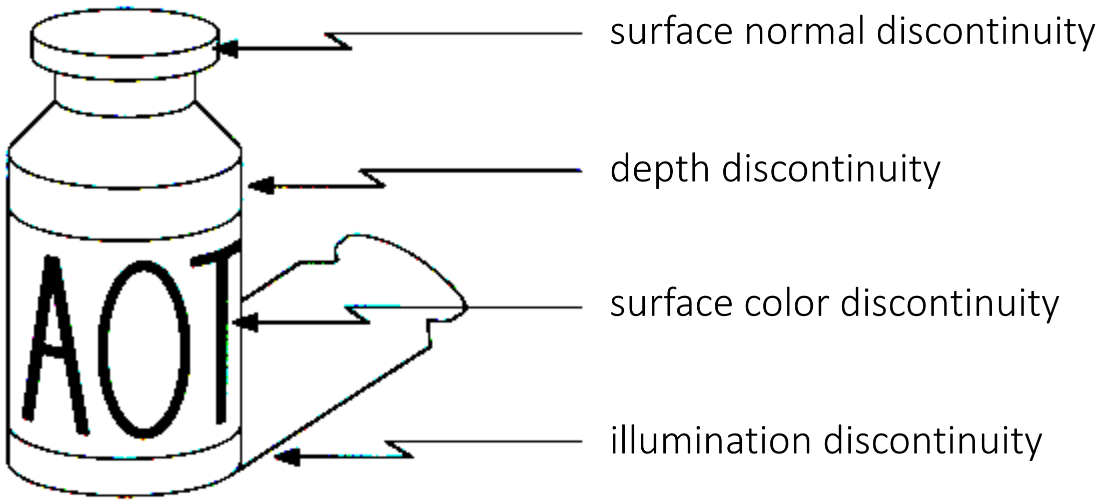
Edge detection





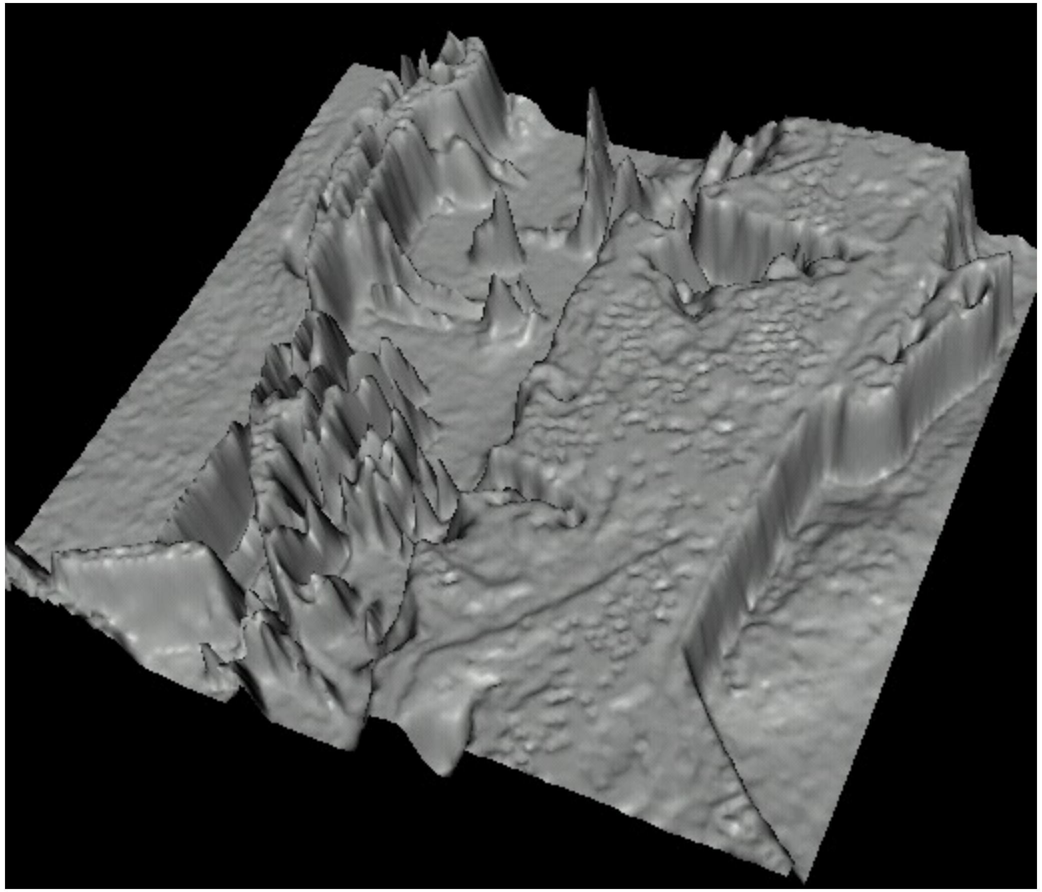
- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - More compact than pixels

Origin of edges



- Edges are caused by a variety of factors

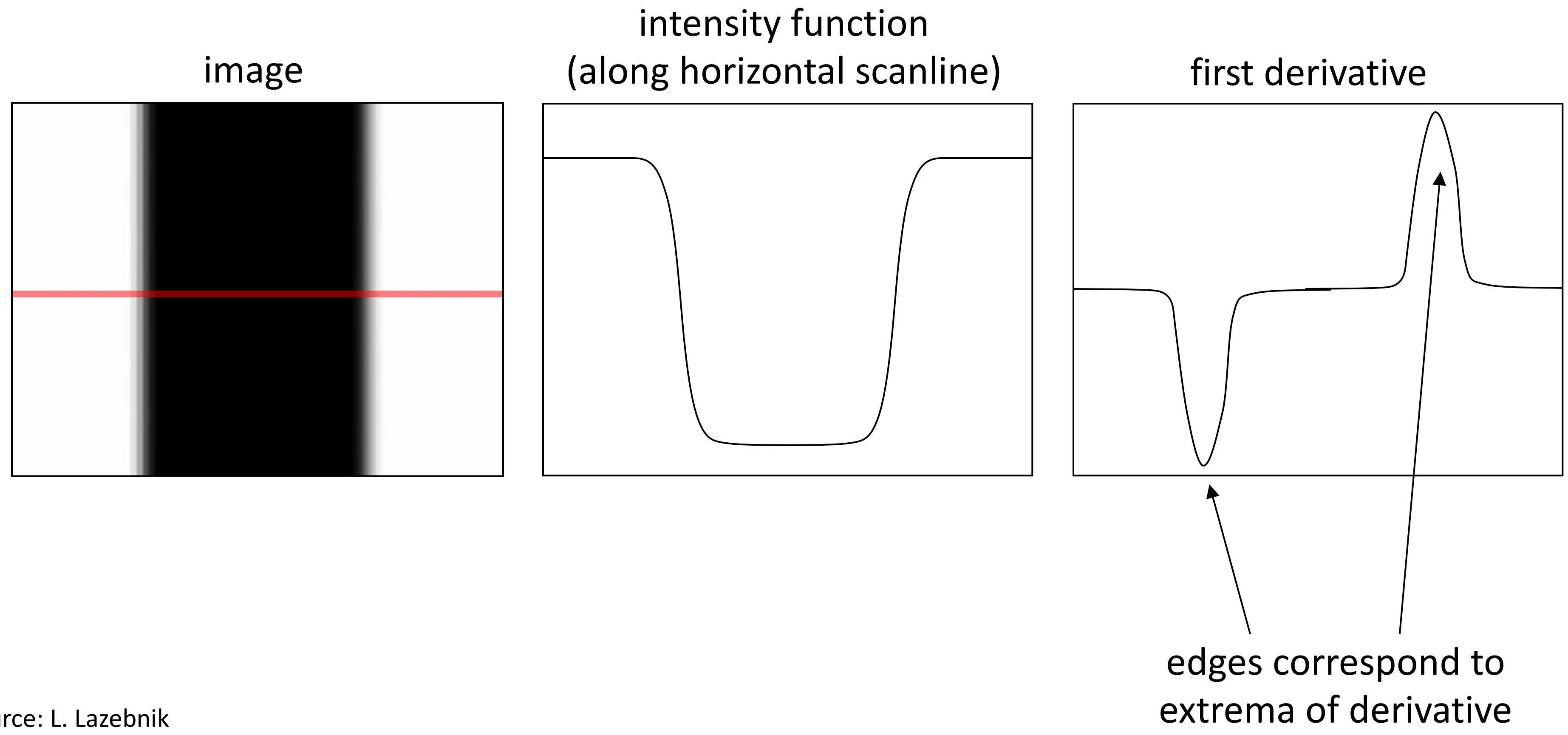
Images as functions…



- Edges look like steep cliffs

Characterizing edges

- An edge is a place of *rapid change* in the image intensity function



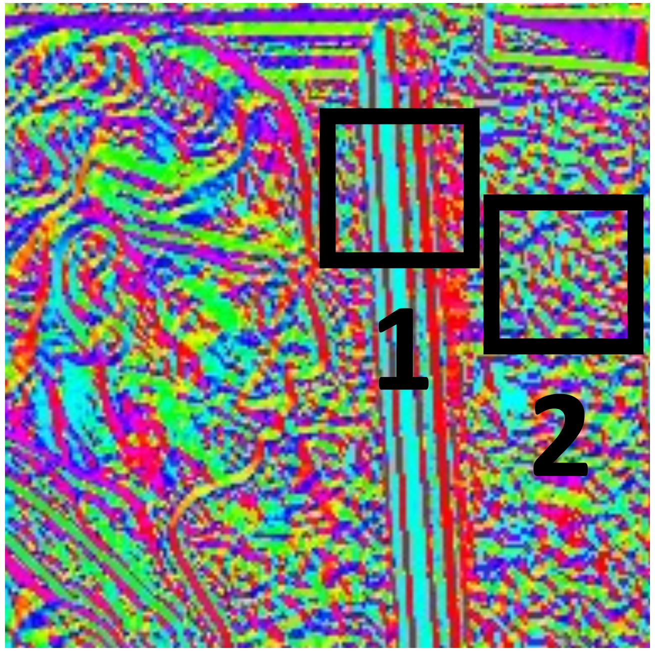
Source: L. Lazebnik

Image Gradient & Edges

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

Direction of image gradients

Why is there structure at 1 and not at 2?

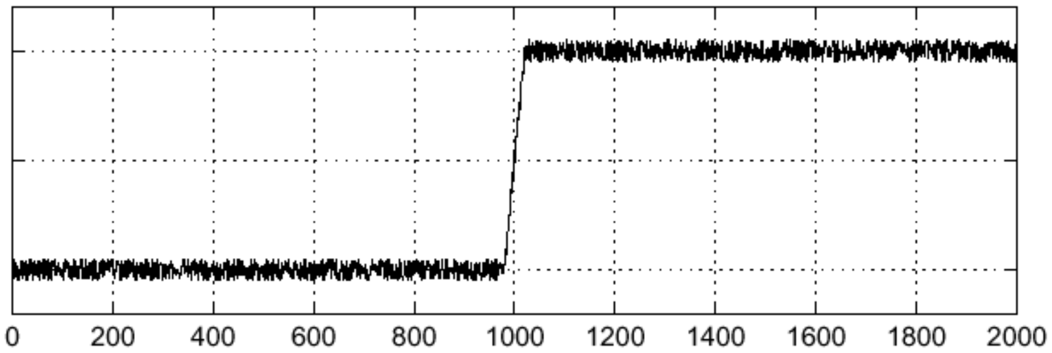


Source: D. Fouhey

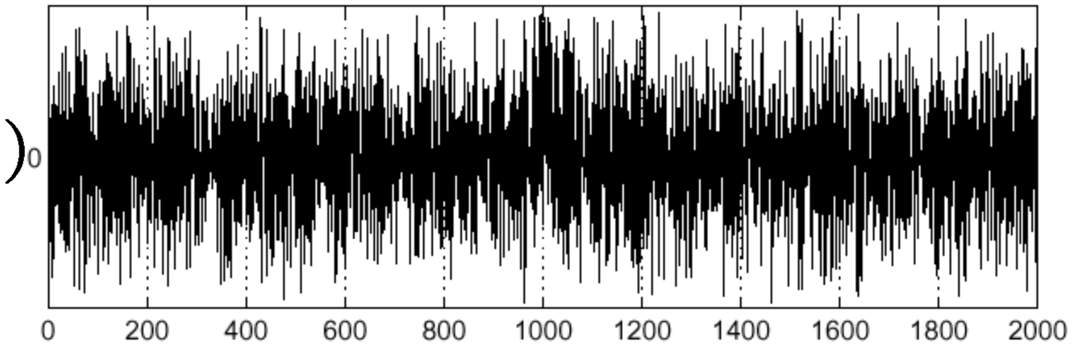
Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal

$f(x)$



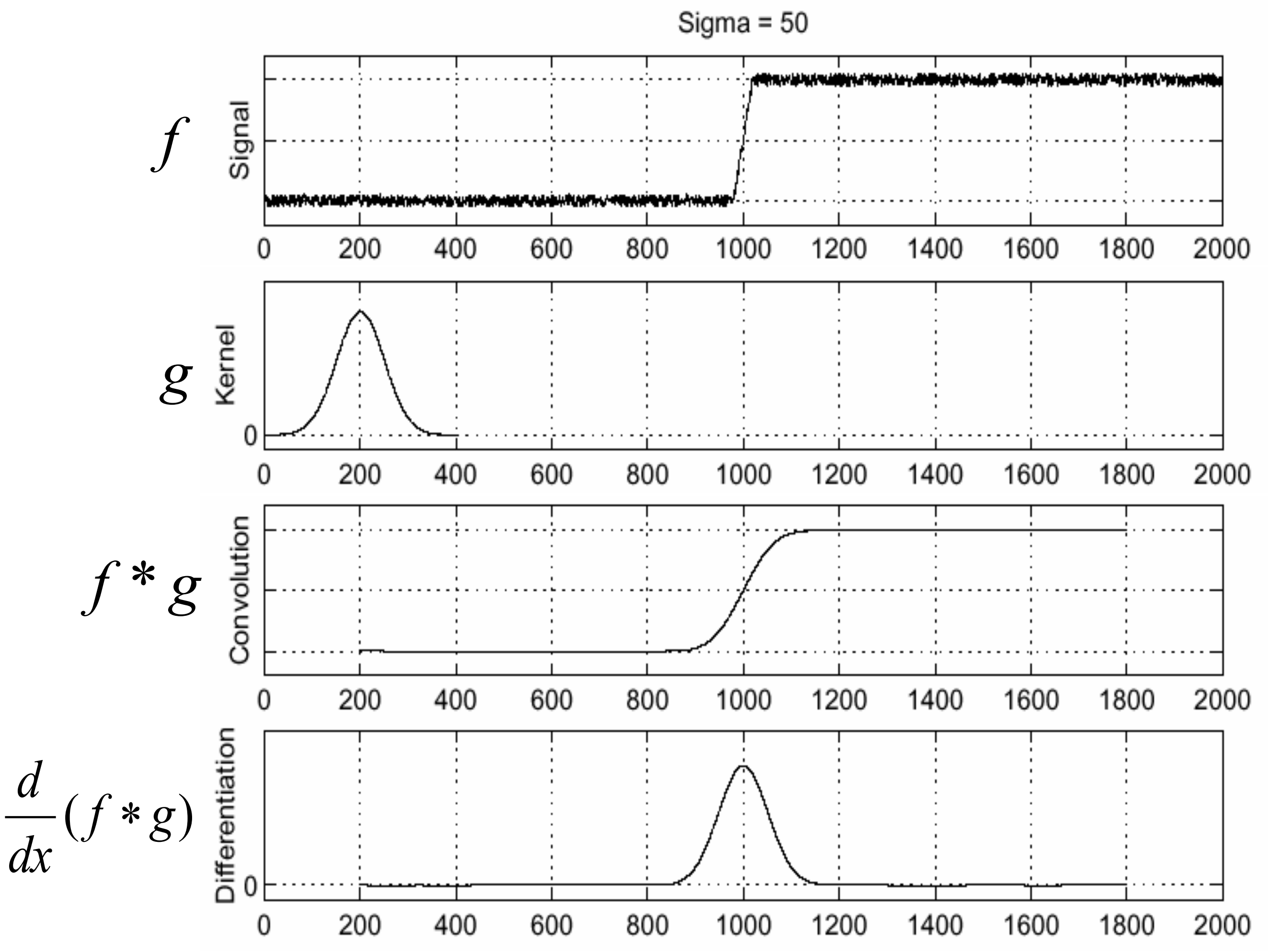
$\frac{d}{dx}f(x)$



Where is the edge?

Source: S. Seitz

Solution: smooth first

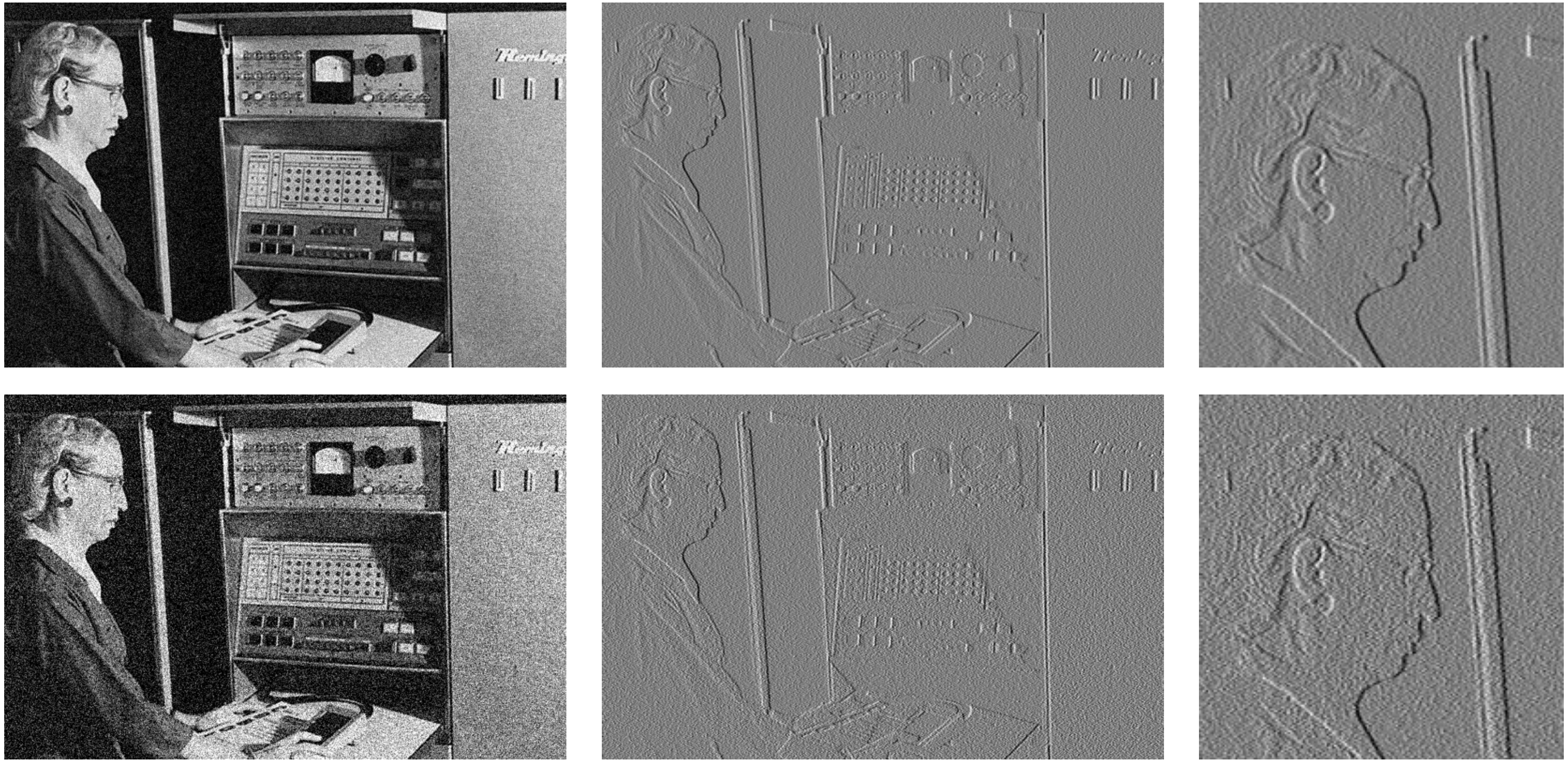


- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Source: S. Seitz

Noise in 2D

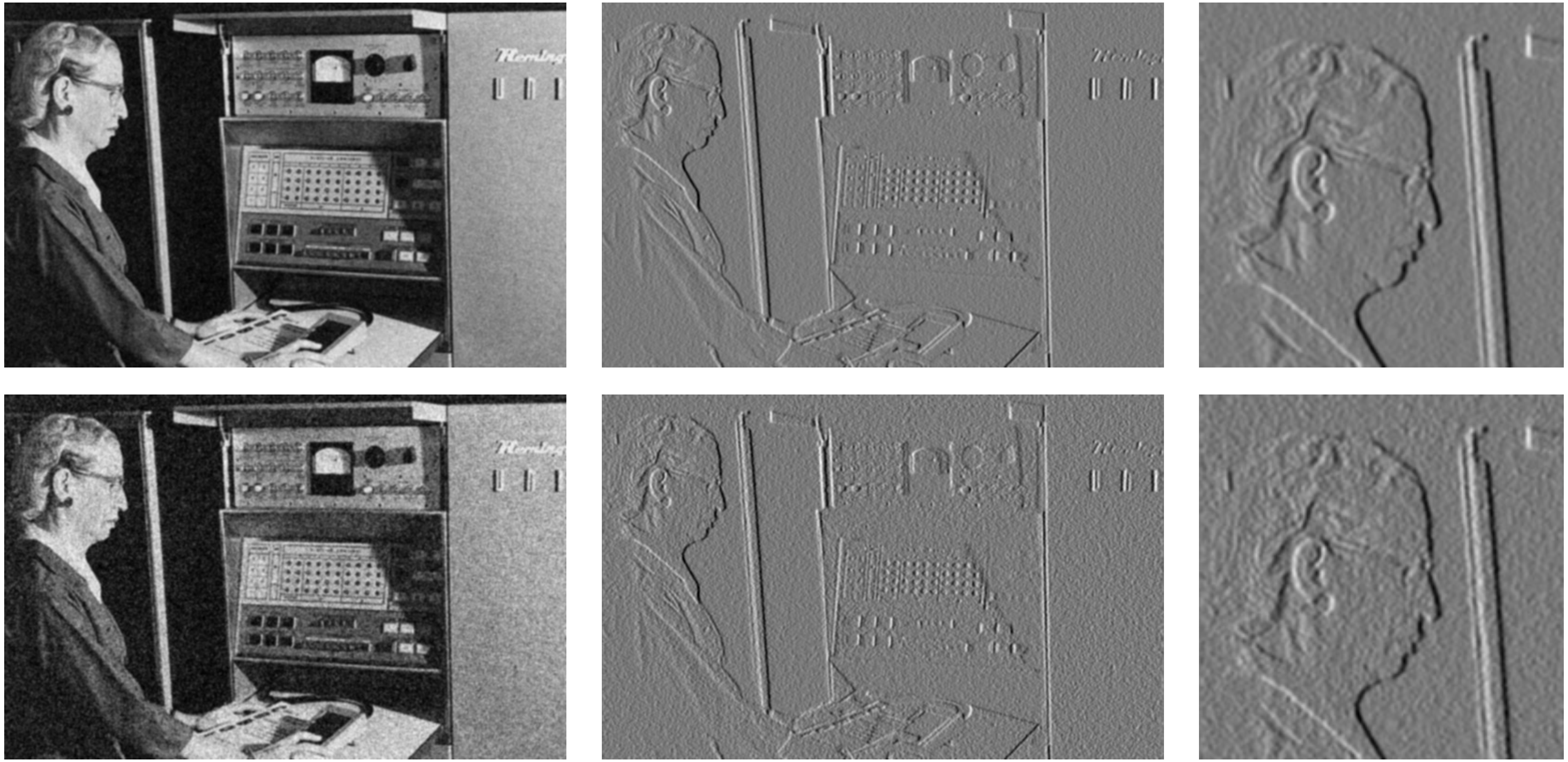
Noisy Input | Ix via [-1,0,1] | Zoom



Source: D. Fouhey

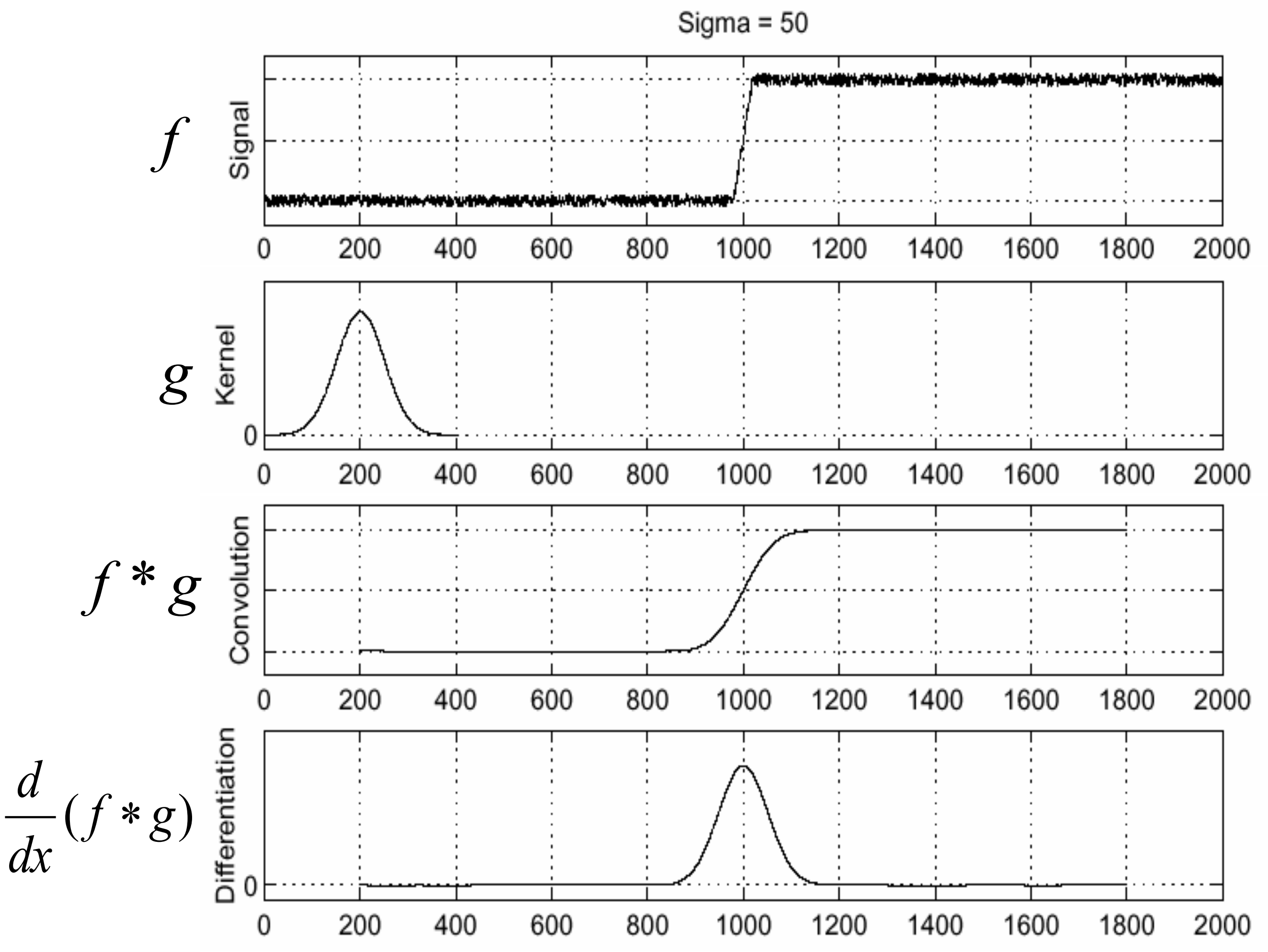
Noise + Smoothing

Smoothed Input | Ix via [-1,01] | Zoom



Source: D. Fouhey

How many convolutions here?

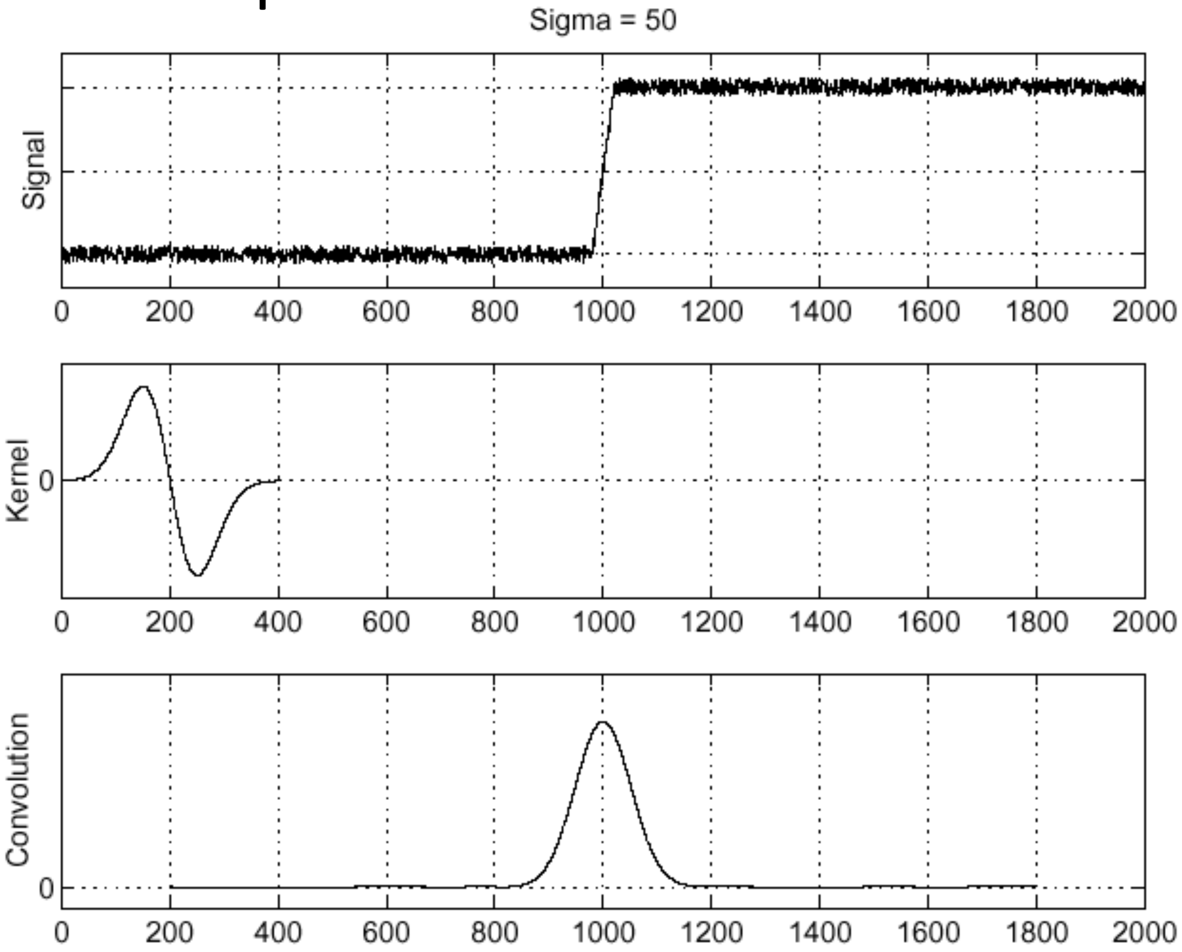


can we reduce this?

Derivative theorem of convolution

$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

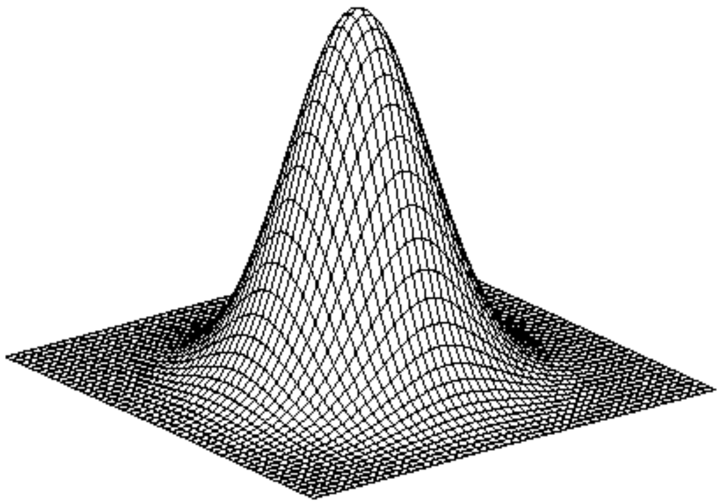
- This saves us one operation:



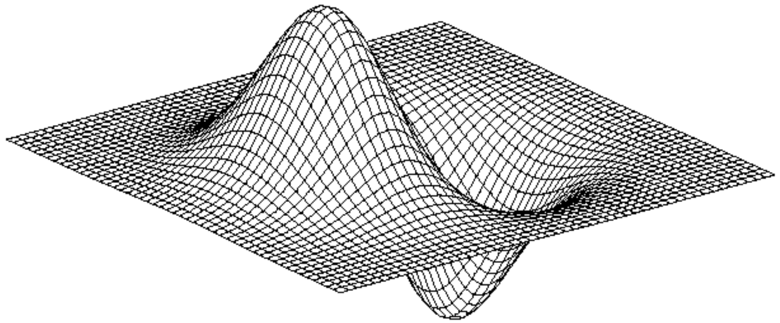
$$\frac{\partial}{\partial x}h$$

$$(\frac{\partial}{\partial x}h) \star f$$

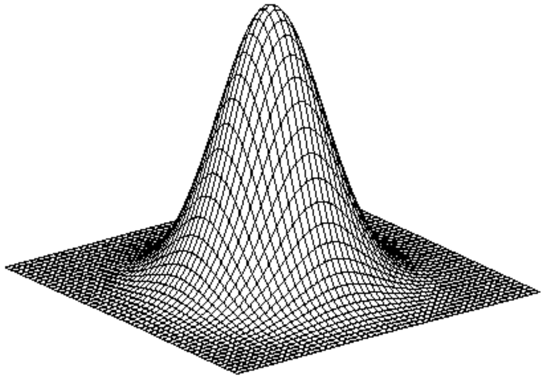
Derivative of Gaussian filter



* [1 -1] =

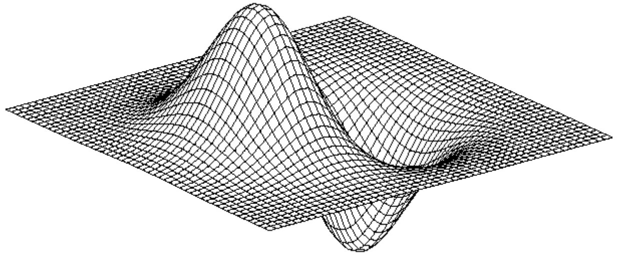


2D edge detection filters



Gaussian

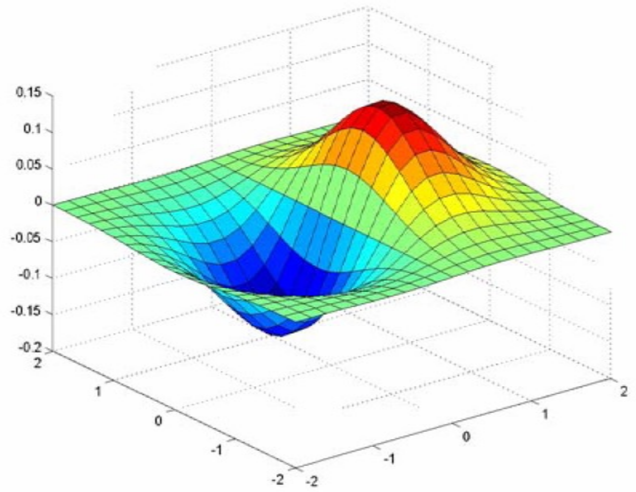
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



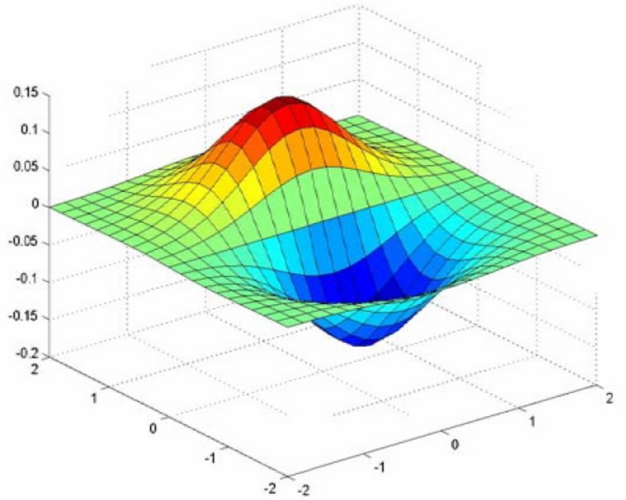
derivative of Gaussian (x)

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

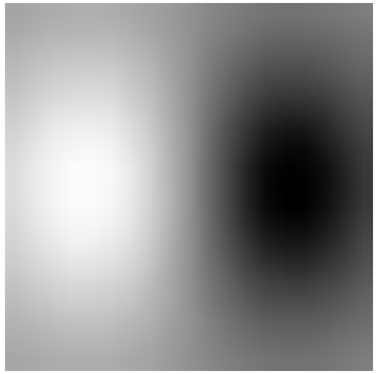
Derivative of Gaussian filter

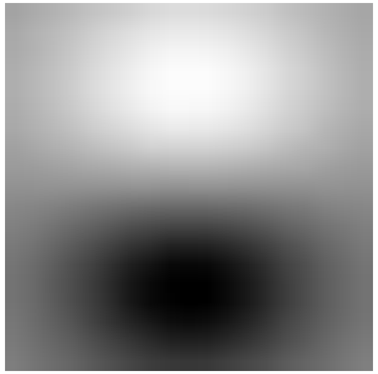


x-direction



y-direction





The Sobel operator

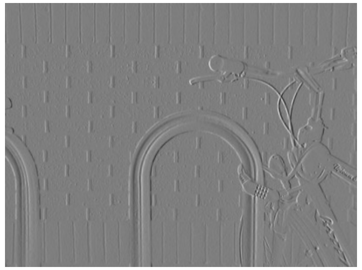
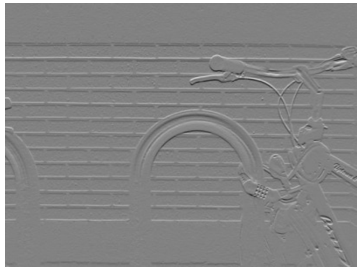
- Common approximation of derivative of Gaussian

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \frac{1}{8}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

$$s_x \qquad\qquad s_y$$

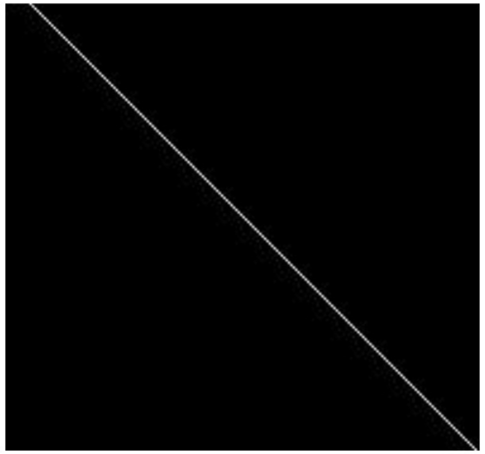
- The standard definition of the Sobel operator omits the 1/8 term
  - doesn't make a difference for edge detection
  - the 1/8 term **is** needed to get the right gradient magnitude

Sobel operator: example

Source: Wikipedia

Image with Edge



Edge Location

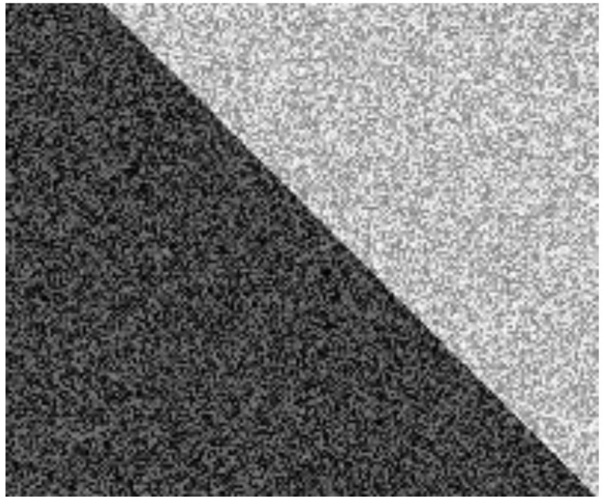
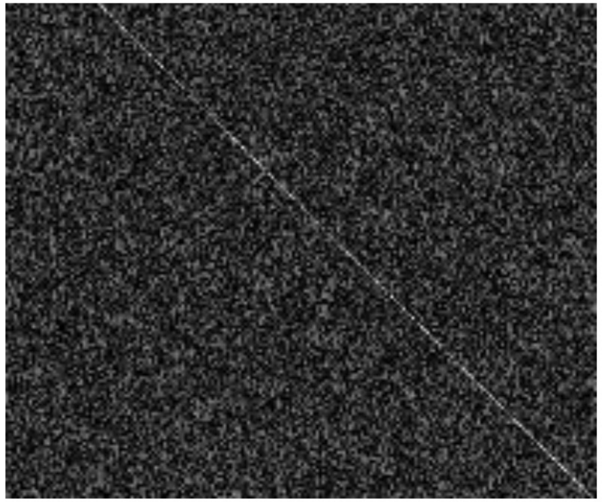
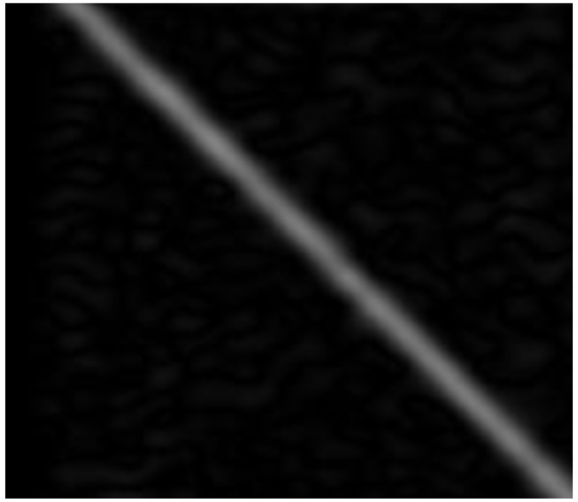


Image + Noise



Derivatives detect
edge *and* noise



Smoothed derivative removes
noise, but blurs edge

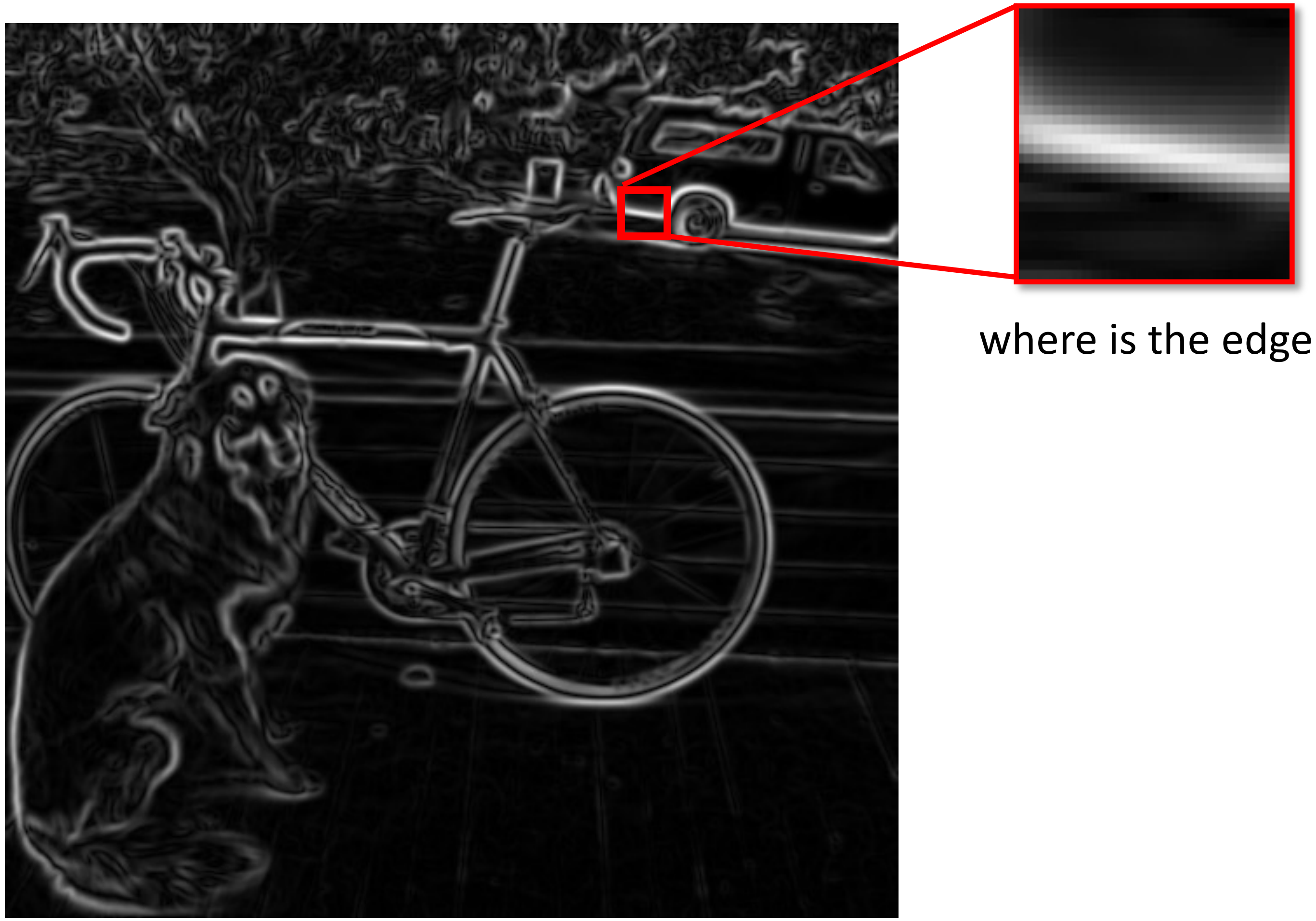
Example

original image

Demo: http://bigwww.epfl.ch/demo/ip/demos/edgeDetector/

Image credit: Joseph Redmon

Finding edges

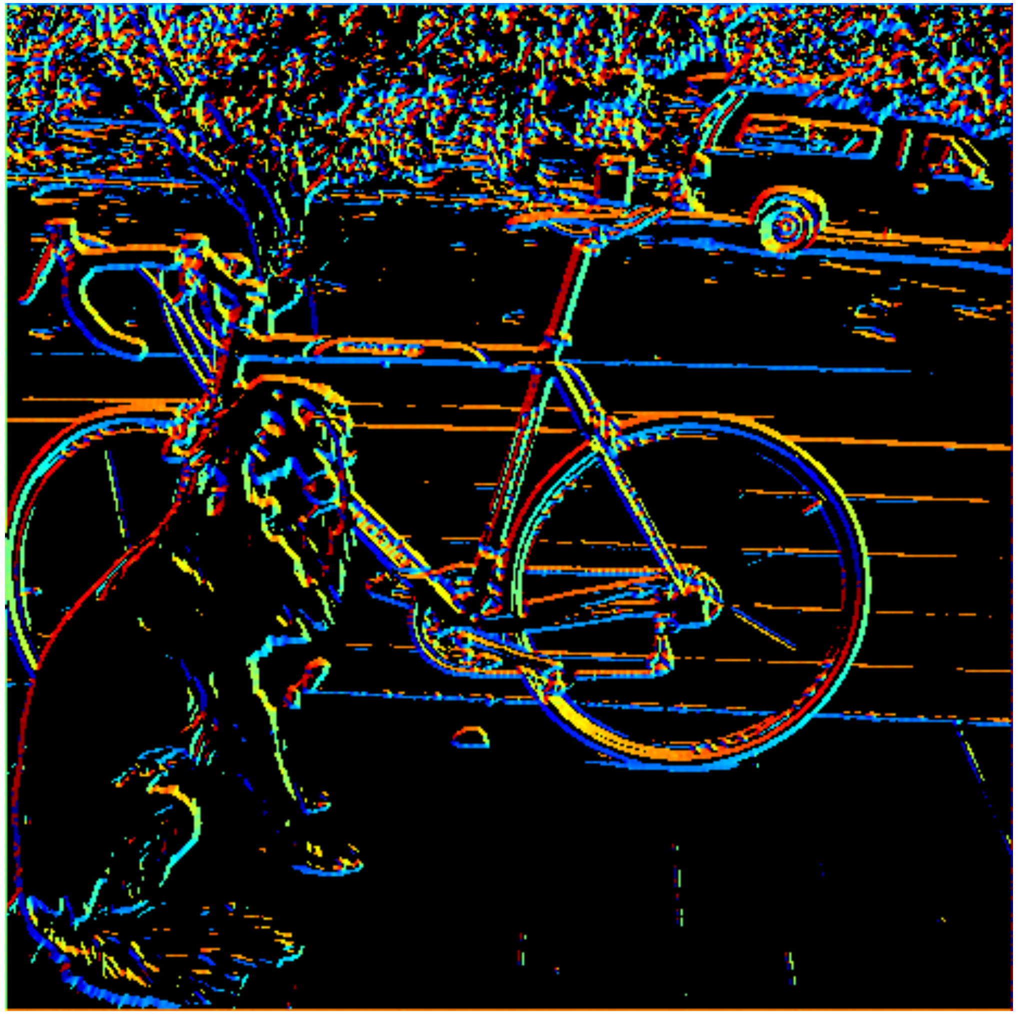


where is the edge?

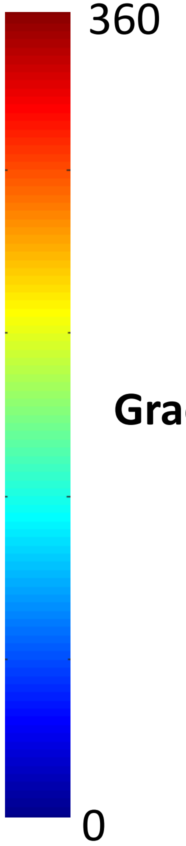
smoothed gradient magnitude

Get Orientation at Each Pixel

- Get orientation (below, threshold at minimum gradient magnitude)

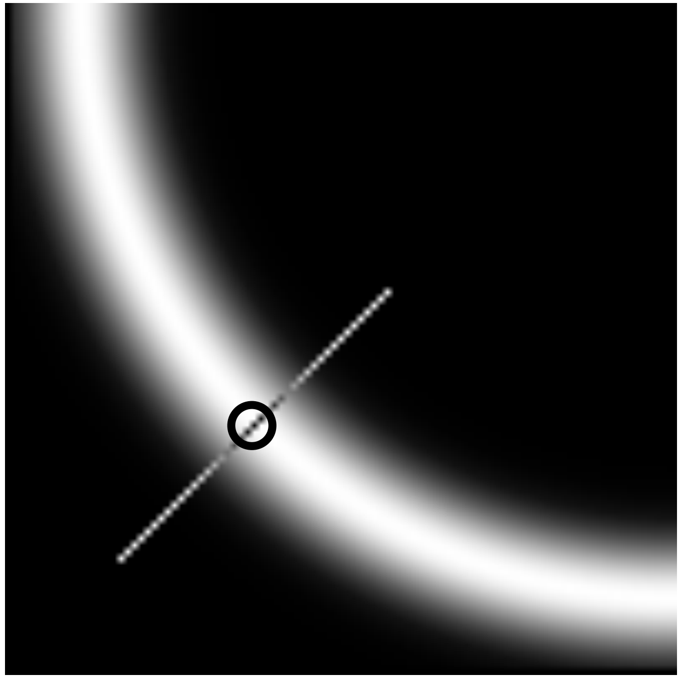


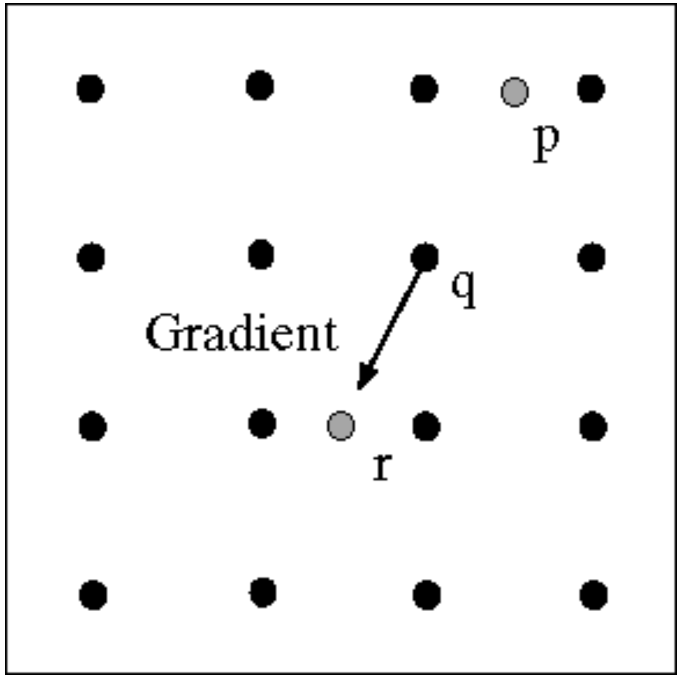
theta = atan2(gy, gx)

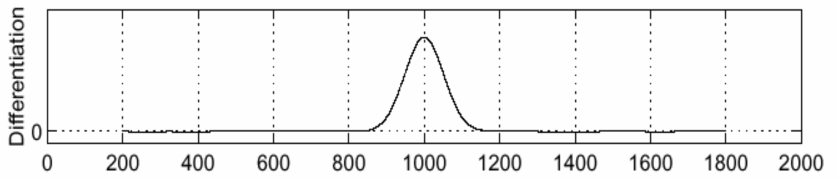


Gradient orientation angle

Non-maximum supression





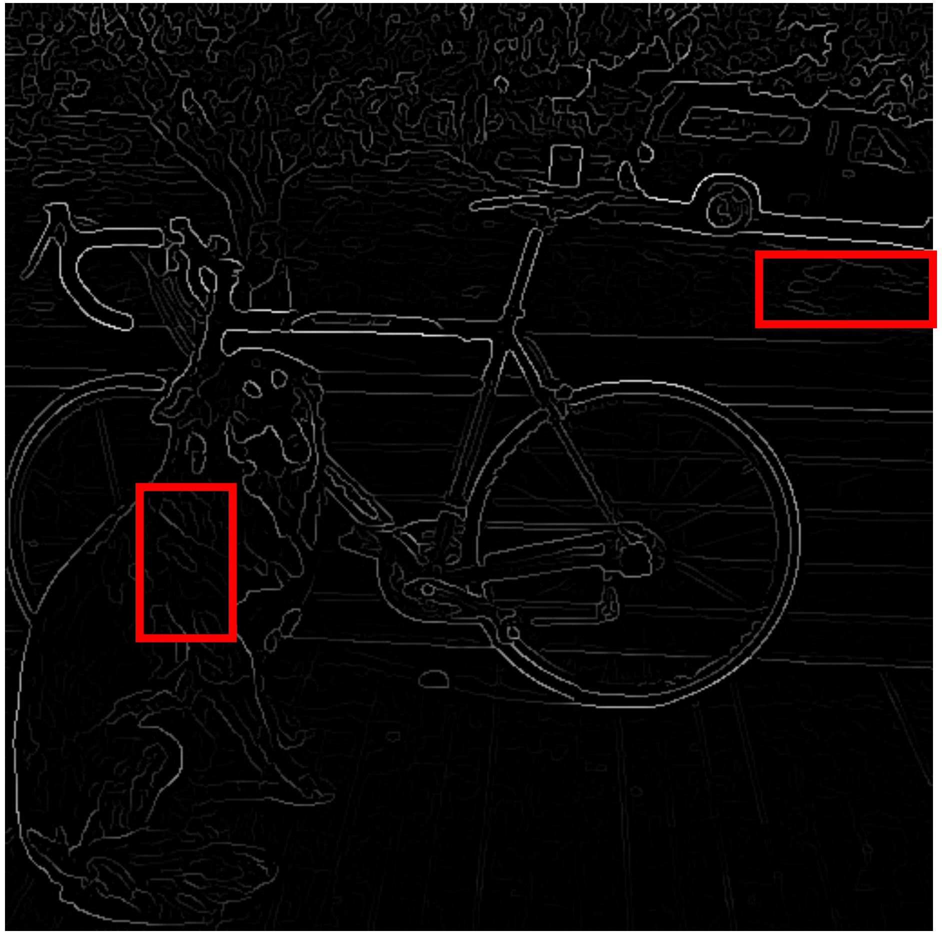


- Check if pixel is local maximum along gradient direction
  - requires *interpolating* pixels p and r

Before Non-max Suppression

After Non-max Suppression

Still noise exists!



Thresholding edges

- Still some noise
- Only want strong edges
- 2 thresholds, 3 cases
  - $R > T$: strong edge
  - $R < T$ but $R > t$: weak edge
  - $R < t$: no edge

- Strong edges are edges!
- Weak edges are edges
  iff they connect to strong
  - Look in some neighborhood
    (usually 8 closest)

Canny edge detector

1. Filter image with derivative of Gaussian

2. Find magnitude and orientation of gradient

3. Non-maximum suppression

4. Linking and thresholding (hysteresis):
    - Define two thresholds: low and high
    - Use the high threshold to start edge curves and the low threshold to continue them

Source: D. Lowe, L. Fei-Fei, J. Redmon

Canny edge detector

- Our first computer vision pipeline!
- Still a widely used edge detector in computer vision

J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

- Depends on several parameters:

high threshold
low threshold
σ : width of the Gaussian blur

Canny edge detector

original

Canny with $\sigma = 1$

Canny with $\sigma = 2$

- The choice of σ depends on desired behavior
  - large σ detects "large-scale" edges
  - small σ detects fine edges

Source: S. Seitz

Slide Credits

- CS5670, Introduction to Computer Vision, **Cornell Tech, by Noah Snavely.**
- CS 194-26/294-26: Intro to Computer Vision and Computational Photography, **UC Berkeley, by Alyosha Efros.**
- **CS 15-463, 663, 862, CMU, by Computational Photography, Ioannis Gkioulekas.**

Suggested Reading

- Szeliski, Chapter 3.1, 3.2, 3.3, 3.5 (3.4 will be covered in next lecture)
- Forsyth & Ponce, Chapter 4, Chapter 5.1, 5.2, 5.3