# Point Processing and Image Filtering

Jun-Yan Zhu

16-726 Learning-based Image Synthesis, Spring 2023

What is an Image?

- We can think of an **image** as a function, f, from R^2 to R:
  - $f(x, y)$ gives the **intensity** at position (x, y)
  - Realistically, we expect the image only to be defined over a rectangle, with a finite range.

- A color image is just three functions pasted together. We can write this as a "vector-valued" function:

$$f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$$

What is an Image?

Single channel grayscale image *f*(*x, y*)

What is an Image?

RGB image © A.A. Efros

BGR version

$$f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$$

- Common bugs: a function call requires BGR (RGB), you feed RGB (BGR)
- The same applies to display/visualization/save function.
- The crazy thing about deep network: the function calls still work. (e.g., recognize car and building in the image).

4

Problem: Dynamic Range

1

1500

25,000
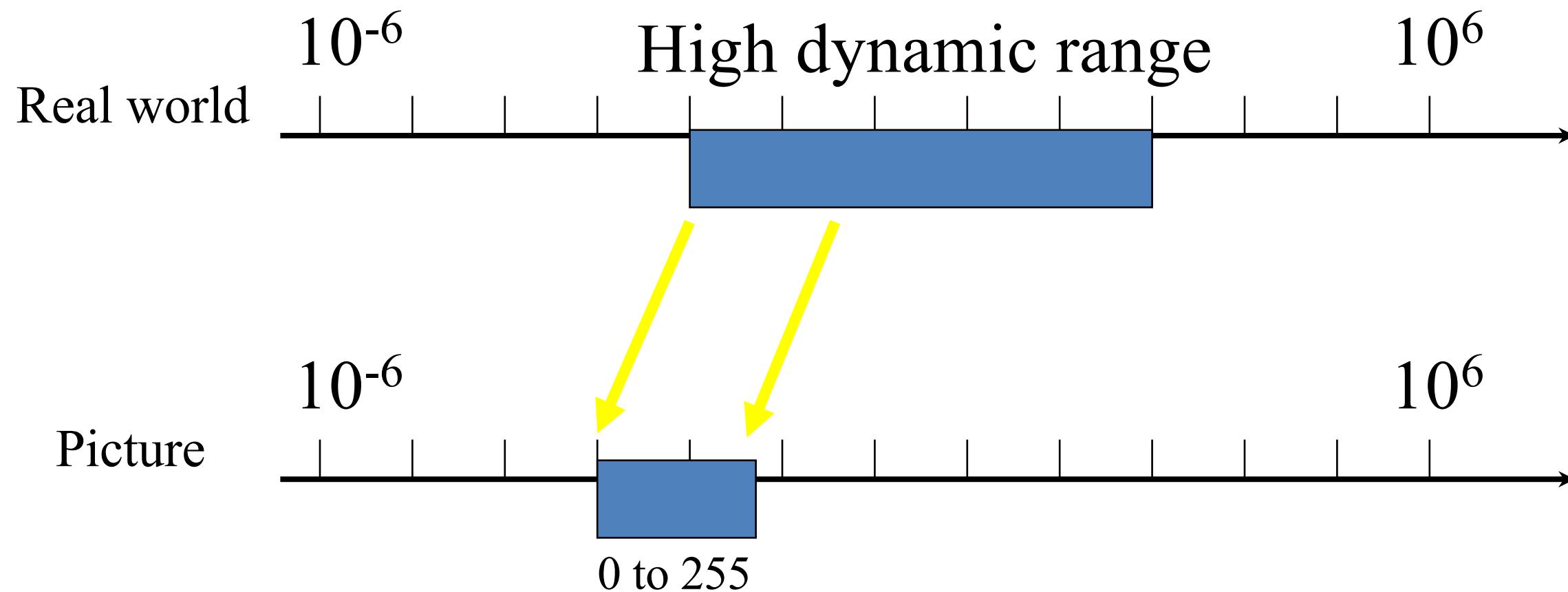
400,000
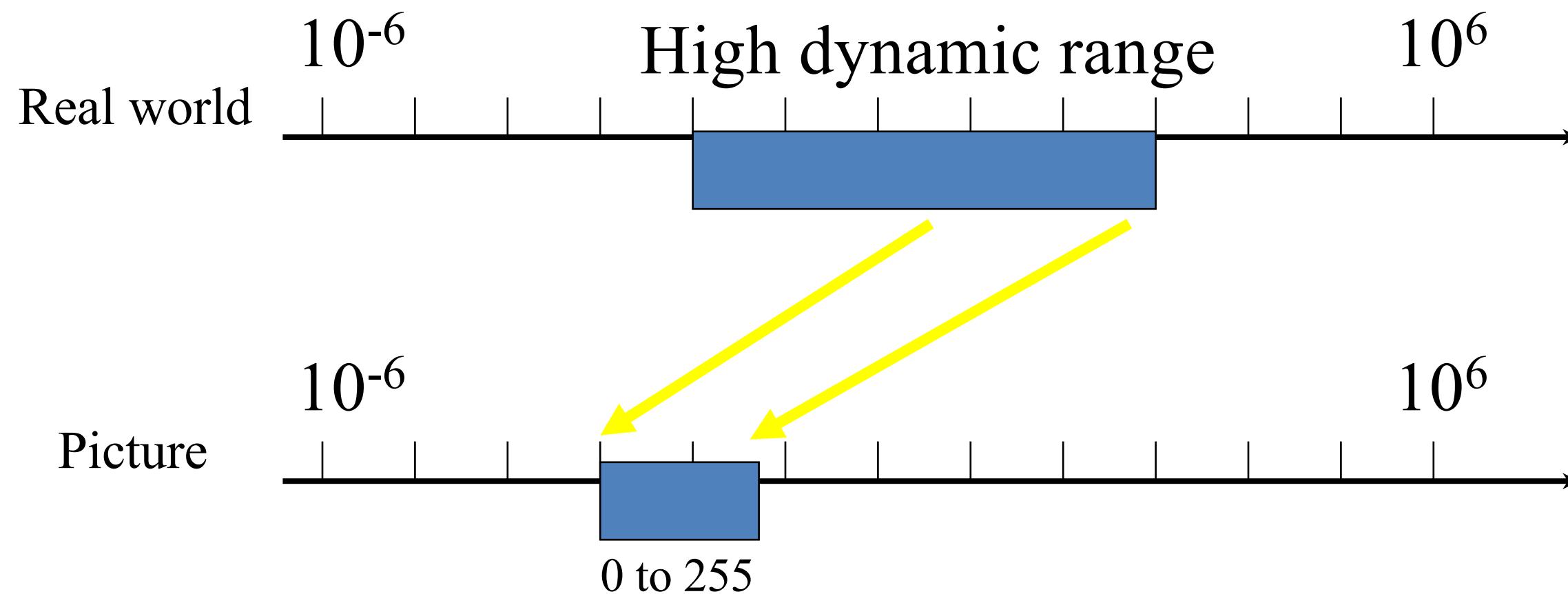
2,000,000,000

The real world is
High dynamic range

Long Exposure

10^{-6} High dynamic range 10^{6}

Real world

10^{-6} 10^{6}

Picture

0 to 255

Short Exposure

10^{-6} High dynamic range 10^{6}

Real world

10^{-6} 10^{6}

Picture

0 to 255
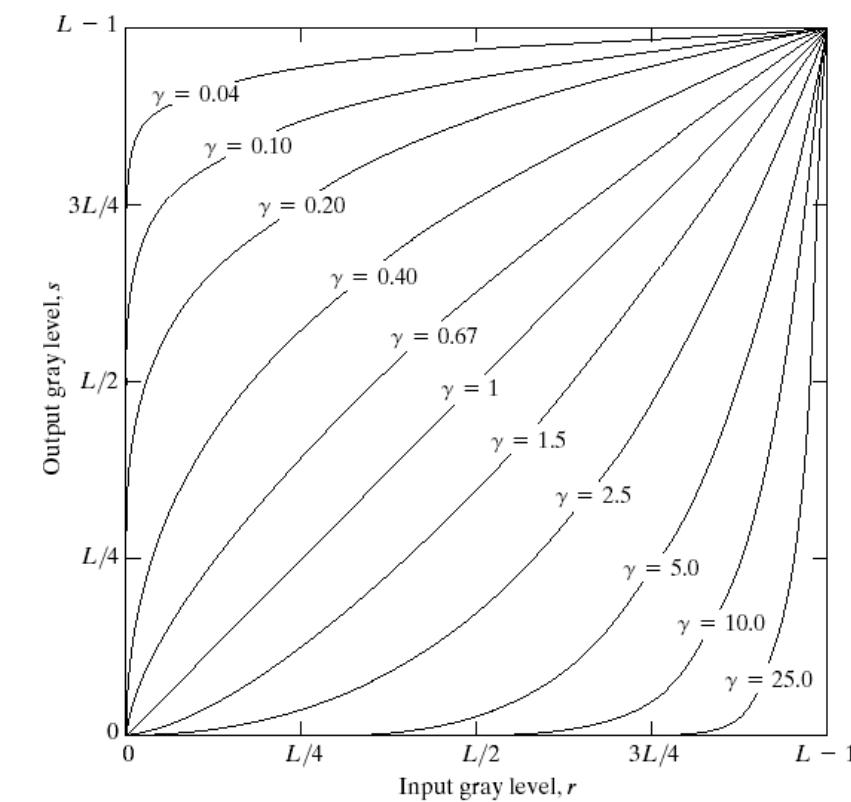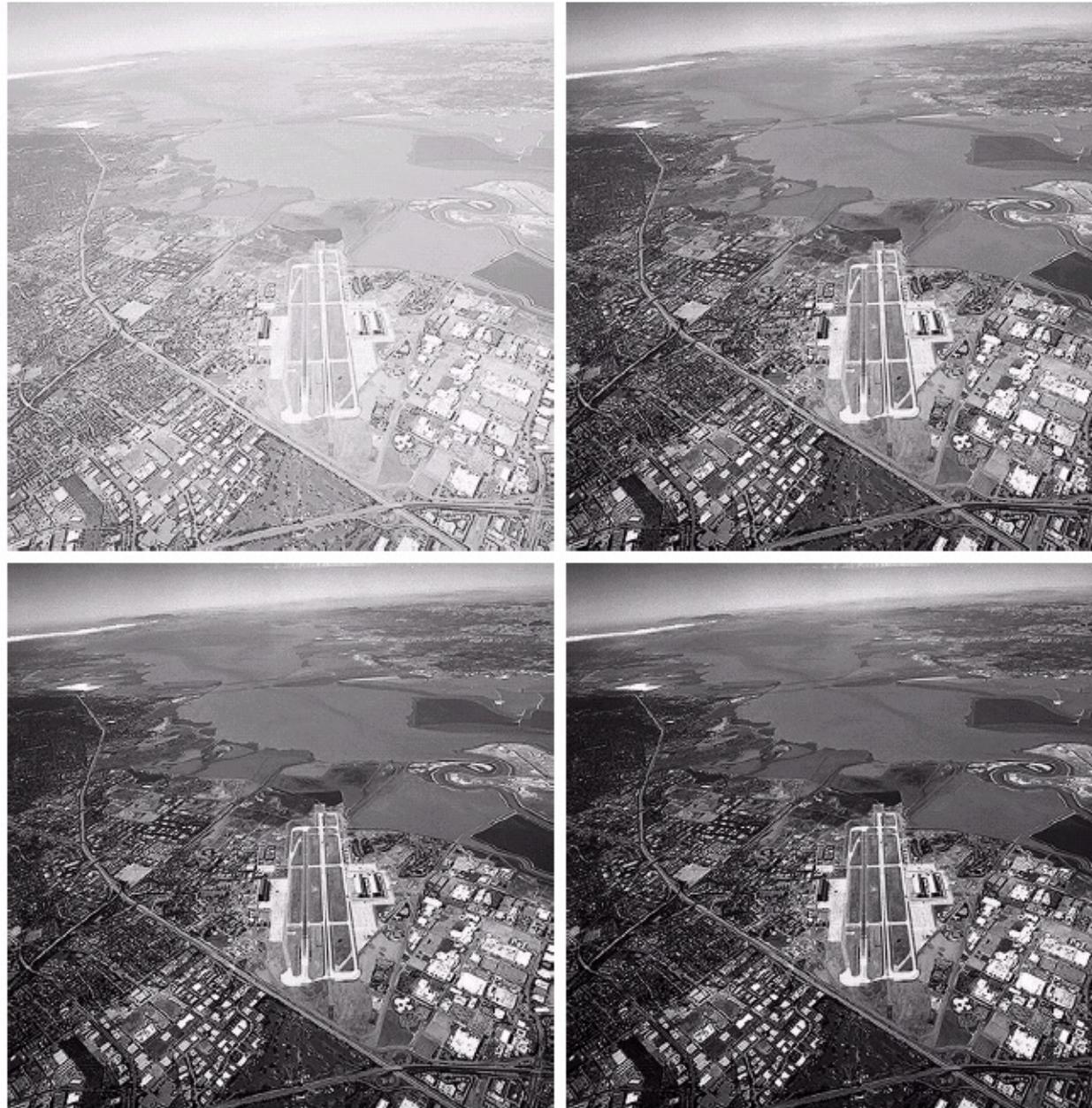
Basic Point Processing: Enhancement

a b
c d

FIGURE 3.9
(a) Aerial image.
(b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 3.0, 4.0,$ and 5.0, respectively. (Original image for this example courtesy of NASA.)

FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases).
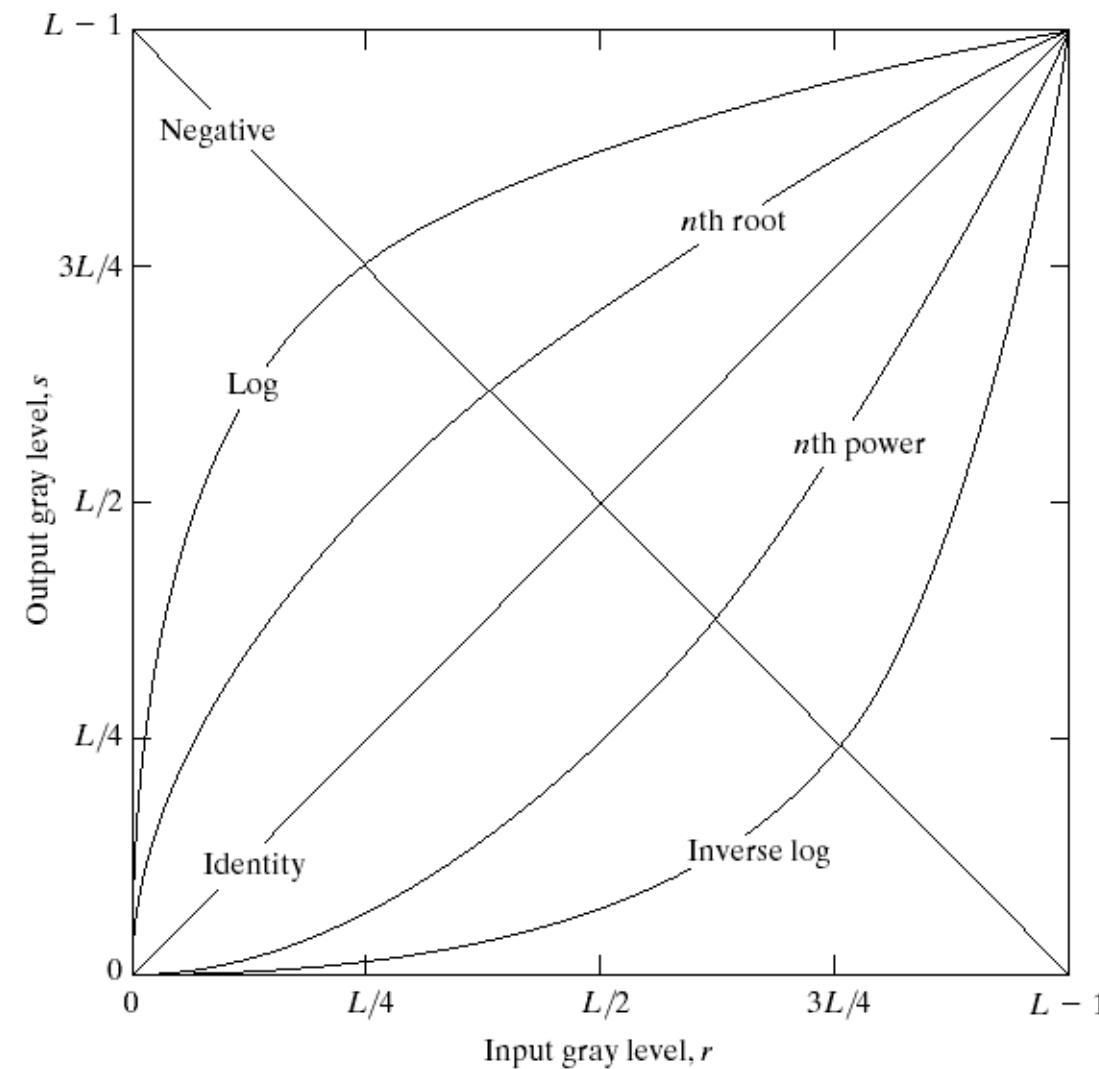
$$c \cdot f(x,y)^{\gamma}$$

Power-law transformations

Basic Point Processing
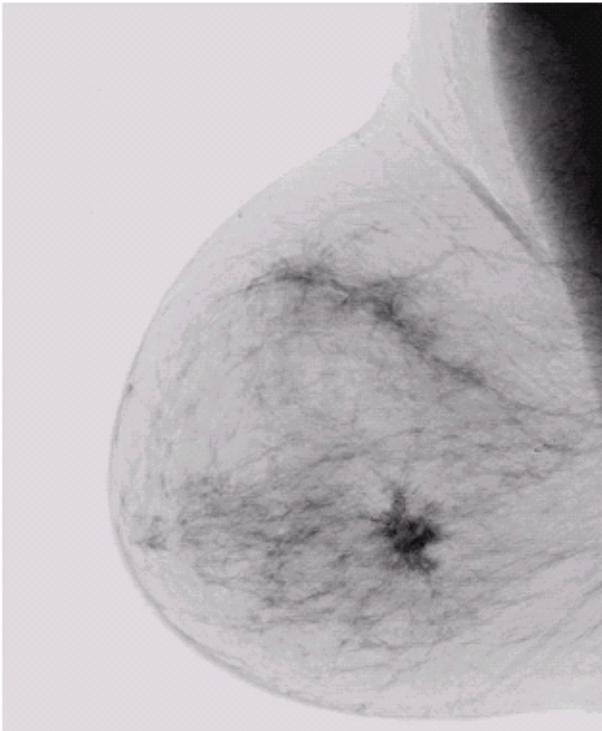
FIGURE 3.3 Some basic gray-level transformation functions used for image enhancement.

Basic Point Processing
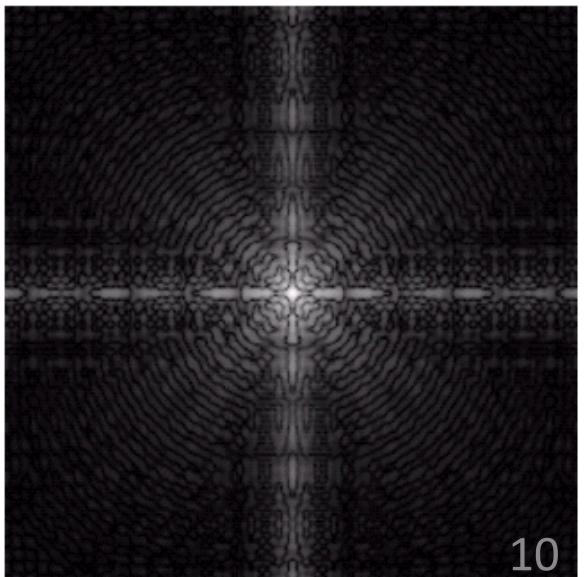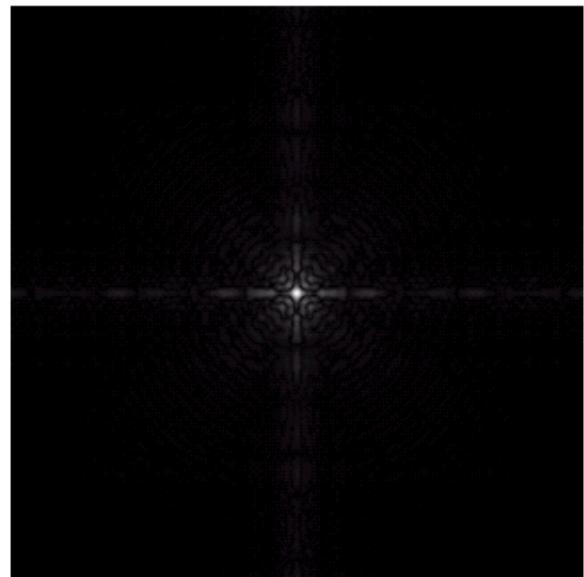
a b

FIGURE 3.4
(a) Original digital mammogram. (b) Negative image obtained using the negative transformation in Eq. (3.2-1). (Courtesy of G.E. Medical Systems.)

Negative
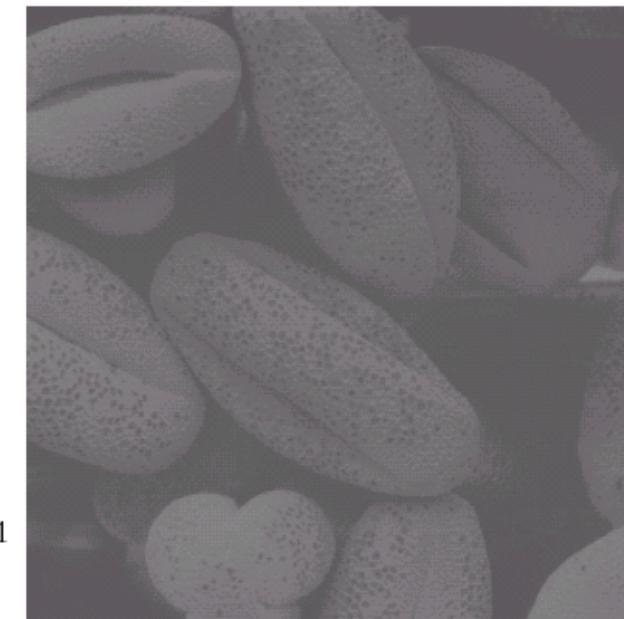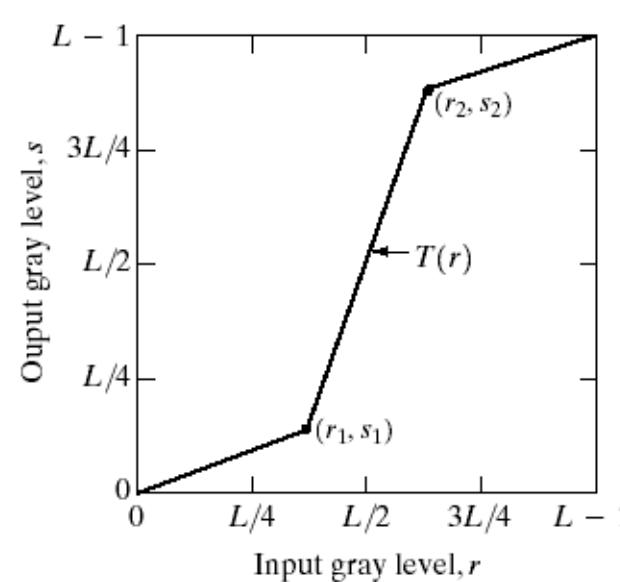
Log

a b

FIGURE 3.5
(a) Fourier spectrum.
(b) Result of applying the log transformation given in Eq. (3.2-2) with $c = 1$.

Contrast Stretching
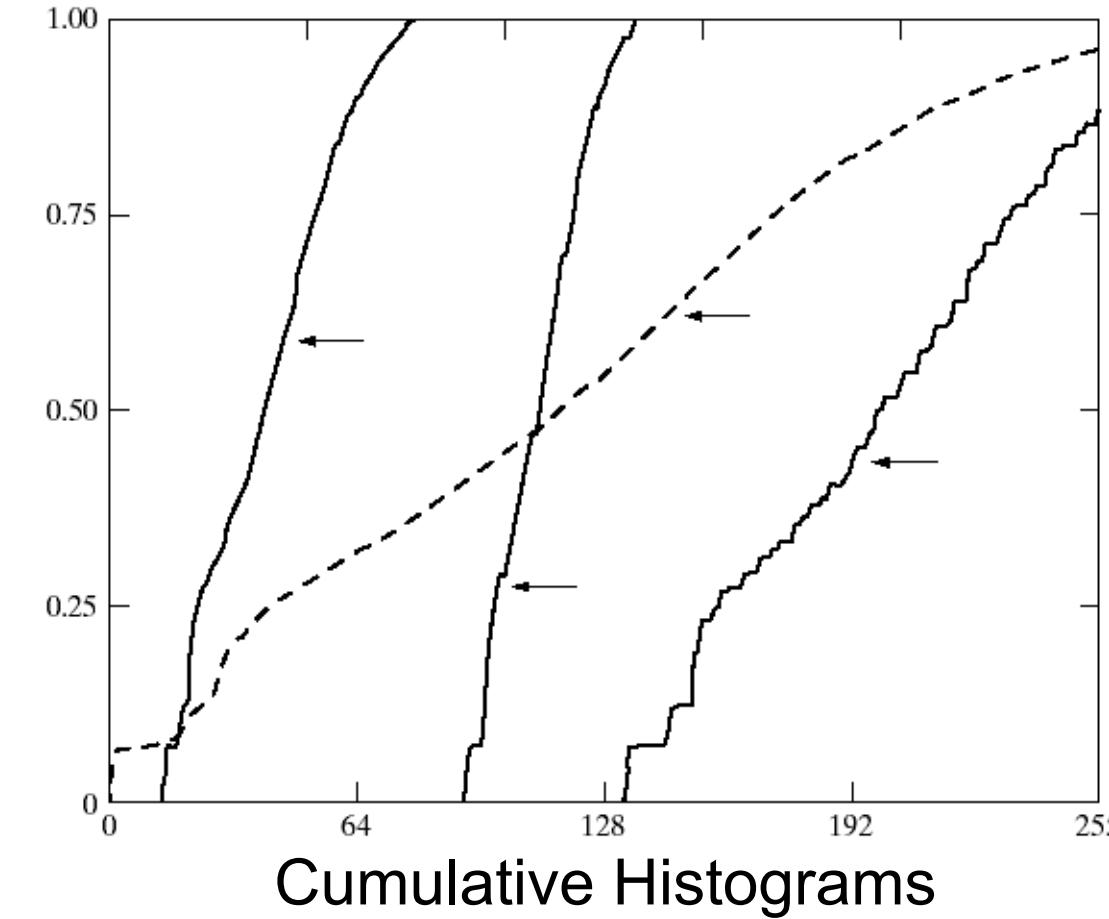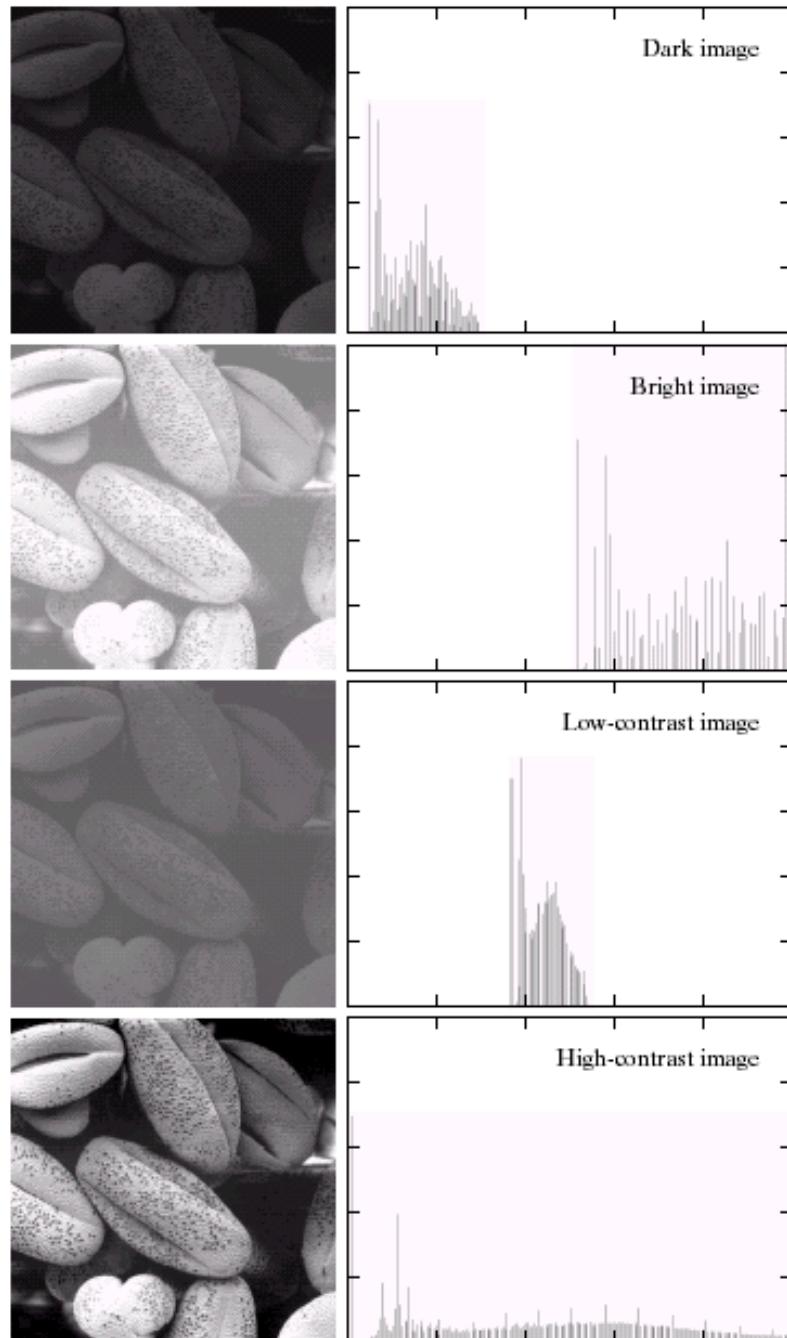
a b
c d

FIGURE 3.10
Contrast stretching.
(a) Form of transformation function. (b) A low-contrast image. (c) Result of contrast stretching. (d) Result of thresholding. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

Learning Mapping functions

Image Histograms

Dark image

Bright image

Low-contrast image

High-contrast image

Cumulative Histograms
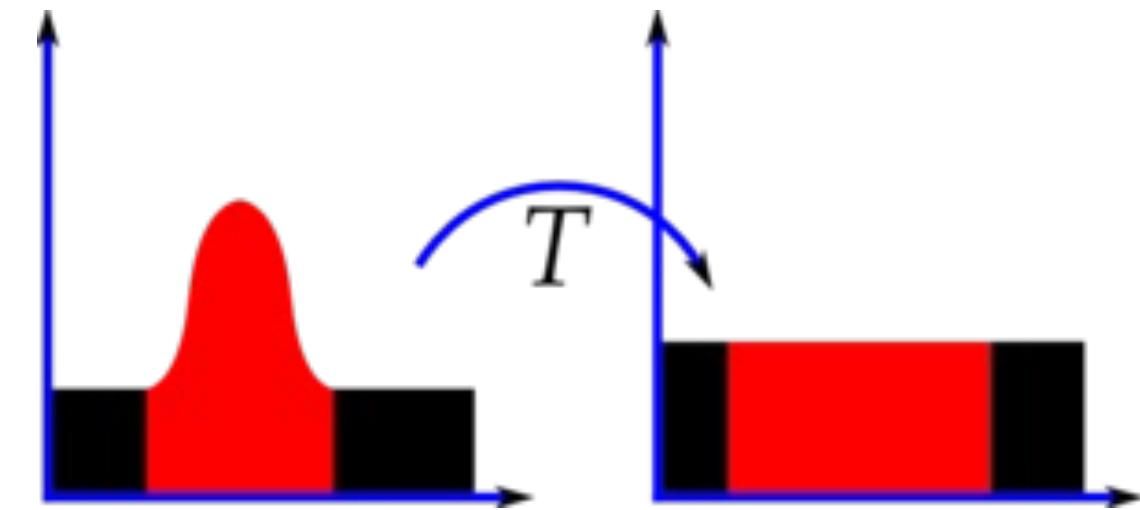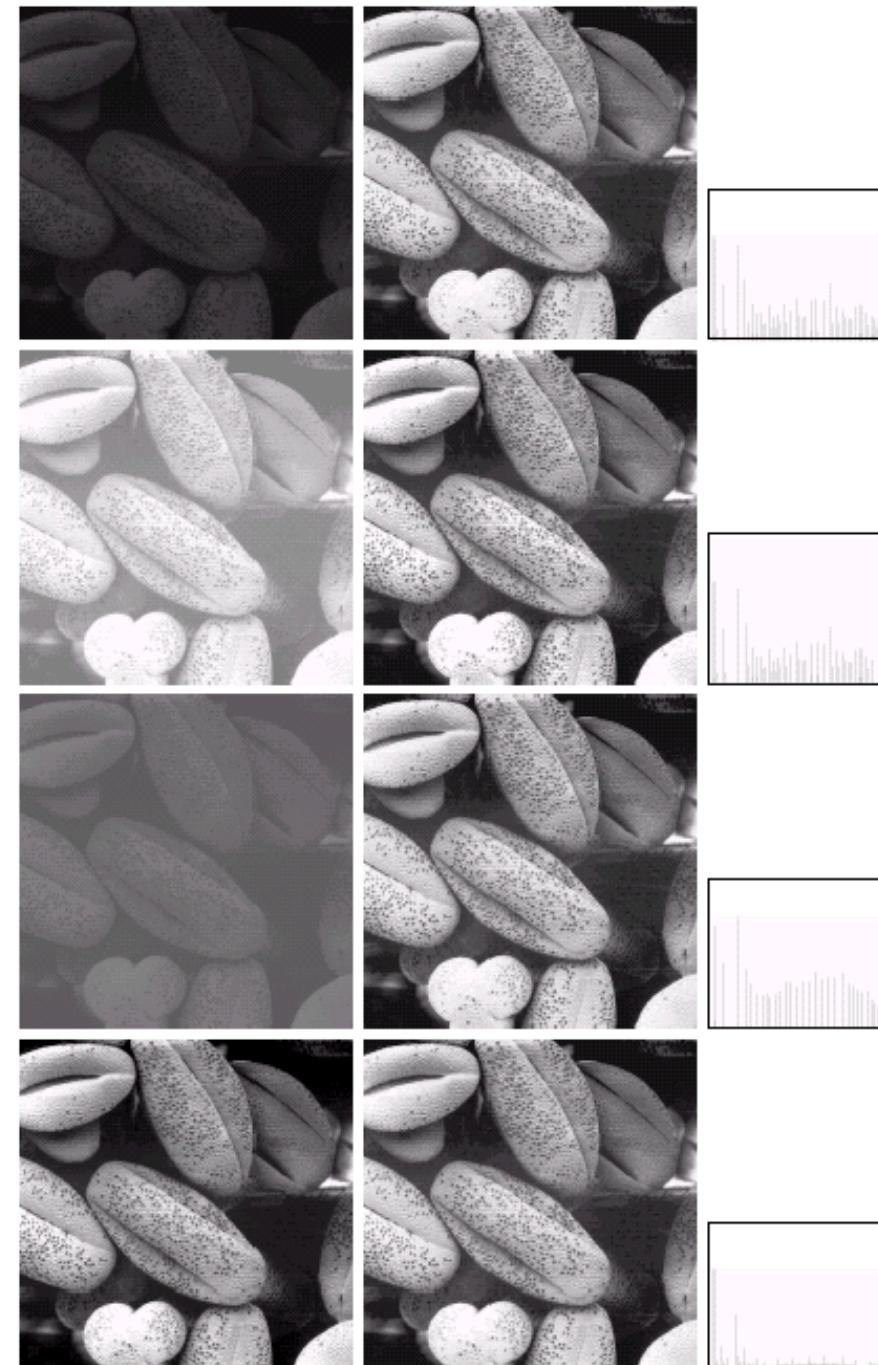
$$T(f(x, y))$$

a b

FIGURE 3.15 Four basic image types: dark, light, low contrast, high contrast, and their corresponding histograms. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

13

Histogram Equalization
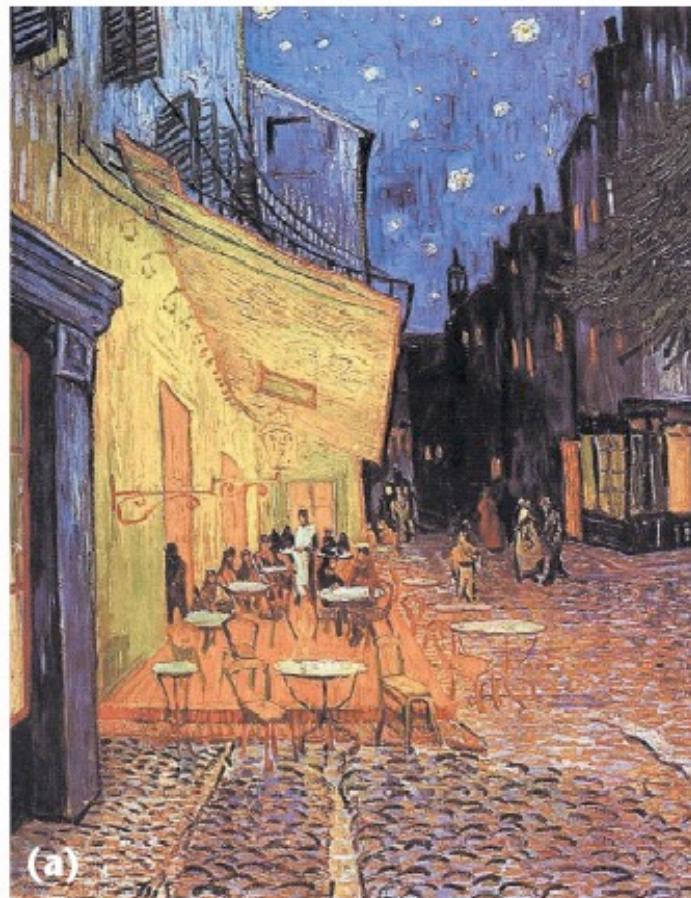
credit @ Zefram

<u>Goal</u>: match input image's CDF to a uniform distribution's CDF.

<u>CDF</u>: cumulative distribution function

<u>Parameters</u>: a lookup table of 256 values

a b c

FIGURE 3.17 (a) Images from Fig. 3.15. (b) Results of histogram equalization. (c) Corresponding histograms.

Color Transfer [Reinhard, et al, 2001]
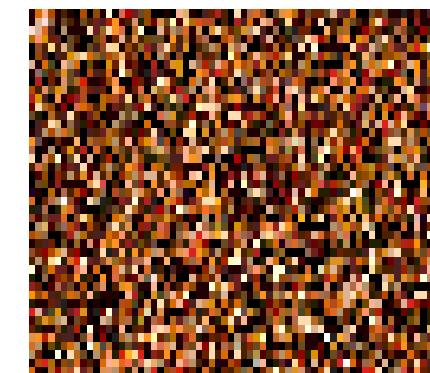
<u>Method</u>: Adjust the mean and standard derivation for each channel
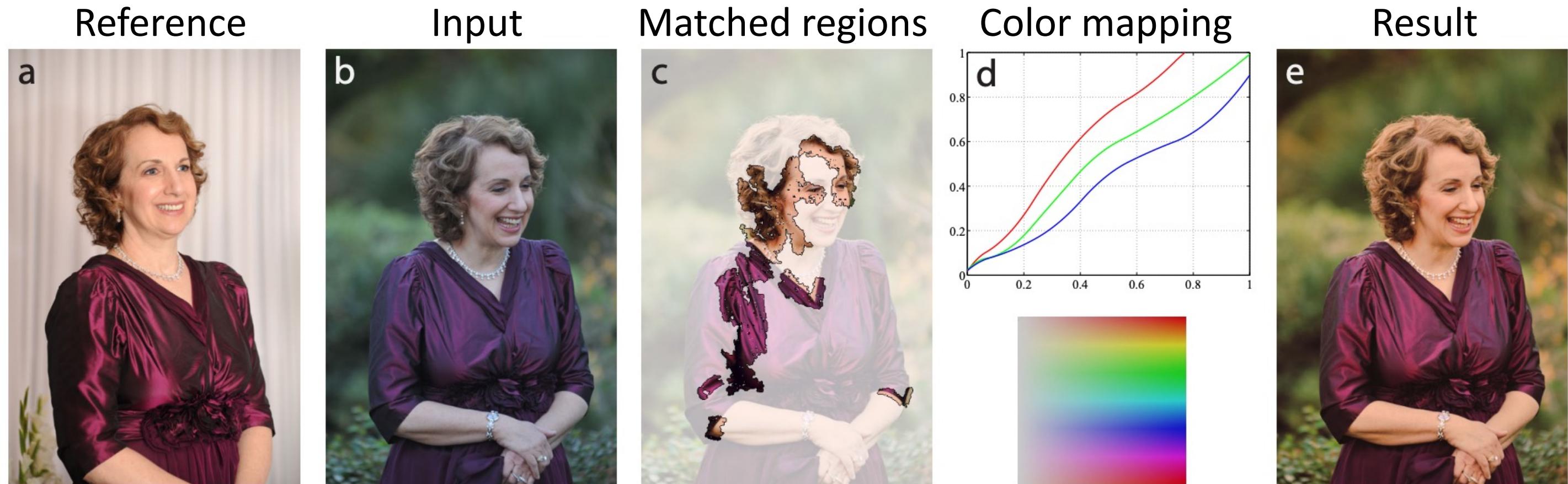
Limitations of Point Processing

Q: What happens if I reshuffle all pixels within the image?

A: It's histogram won't change. No histogram statistics will be affected…

Solution: only transfer across shared regions

Reference   Input   Matched regions   Color mapping   Result

NRDC: Non-Rigid Dense Correspondence with Applications for Image Enhancement
HaCohen et al., SIGGRAPH 2011.
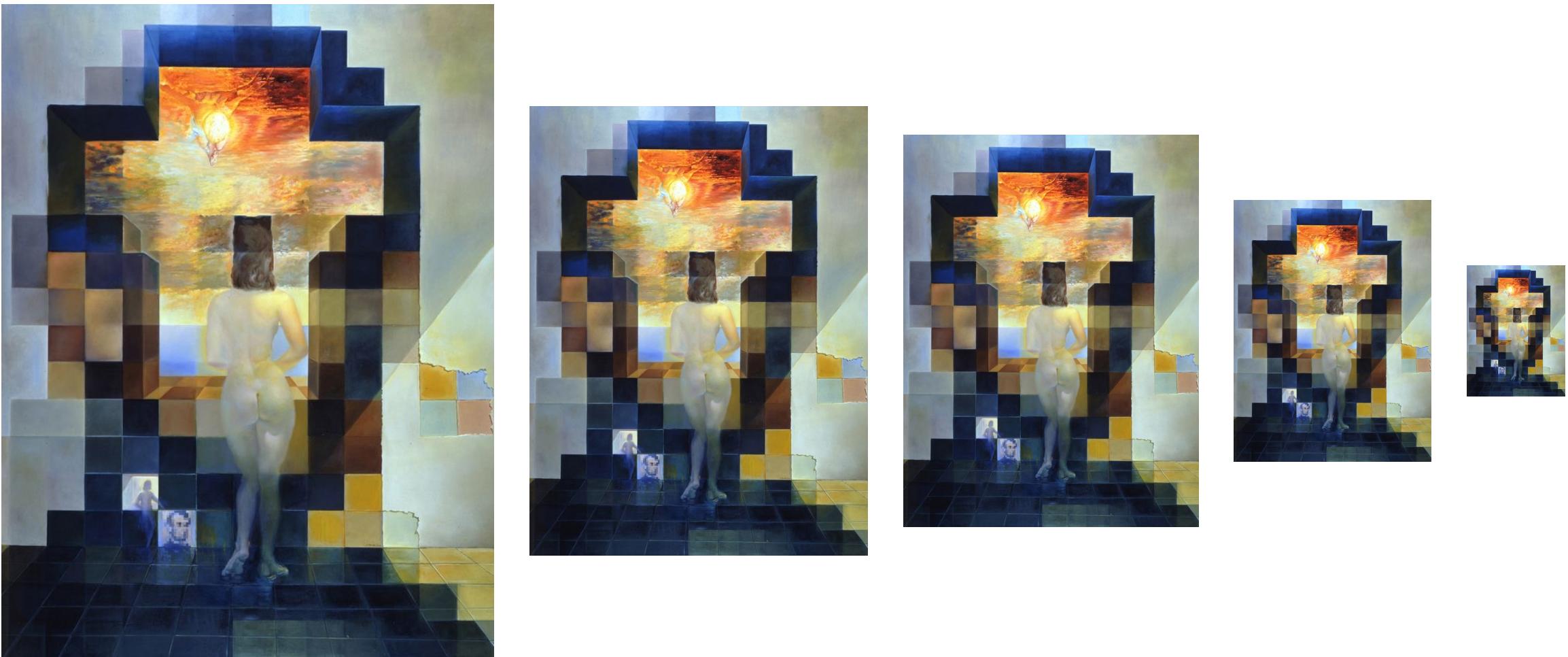
A more general idea: transfer color from sky to sky, mountain to mountain, people to people

Limitations of Point Processing

- Q: What if I want to replace the sky in my image

- A: point processing cannot create new structure/texture/object. But Why?
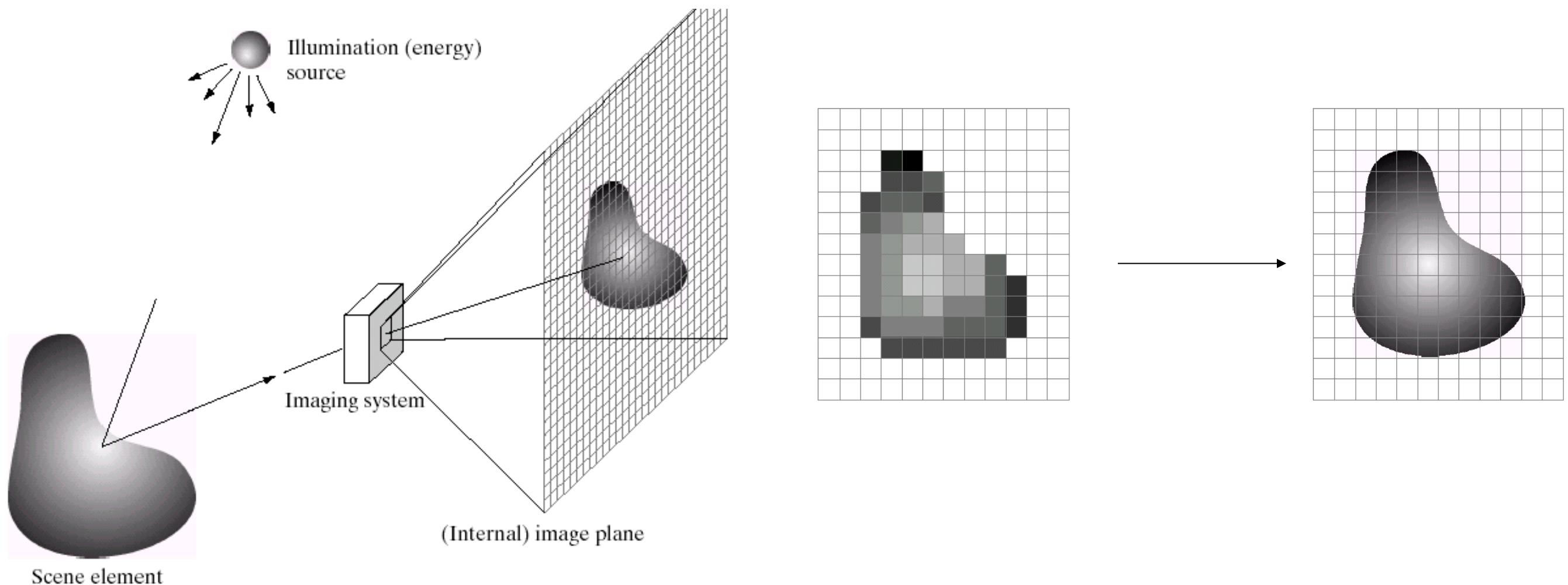
Point Processing and <u>Image Filtering</u>

Jun-Yan Zhu
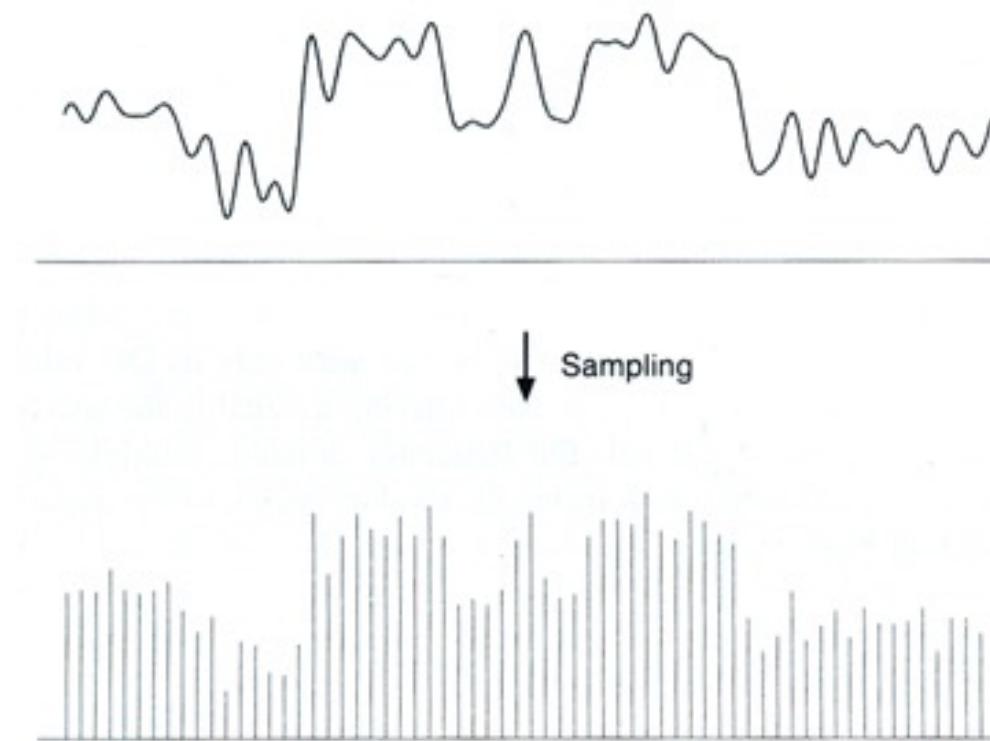
16-726 Learning-based Image Synthesis, Spring 2023

Sampling and Reconstruction

Illumination (energy) source

Scene element

Imaging system

(Internal) image plane

Sampled Representations
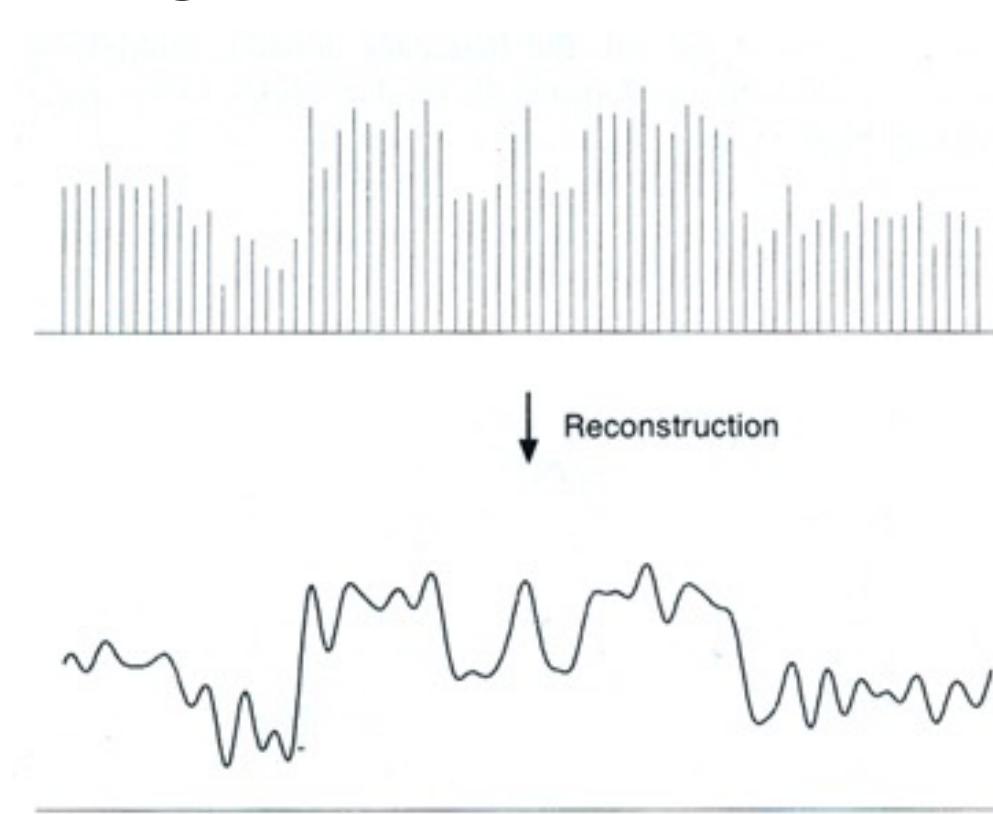
- How to store and compute with continuous functions?
- Common scheme for representation: samples
  - write down the function's values at many points

Sampling

[FvDFH fig.14.14b / Wolberg]
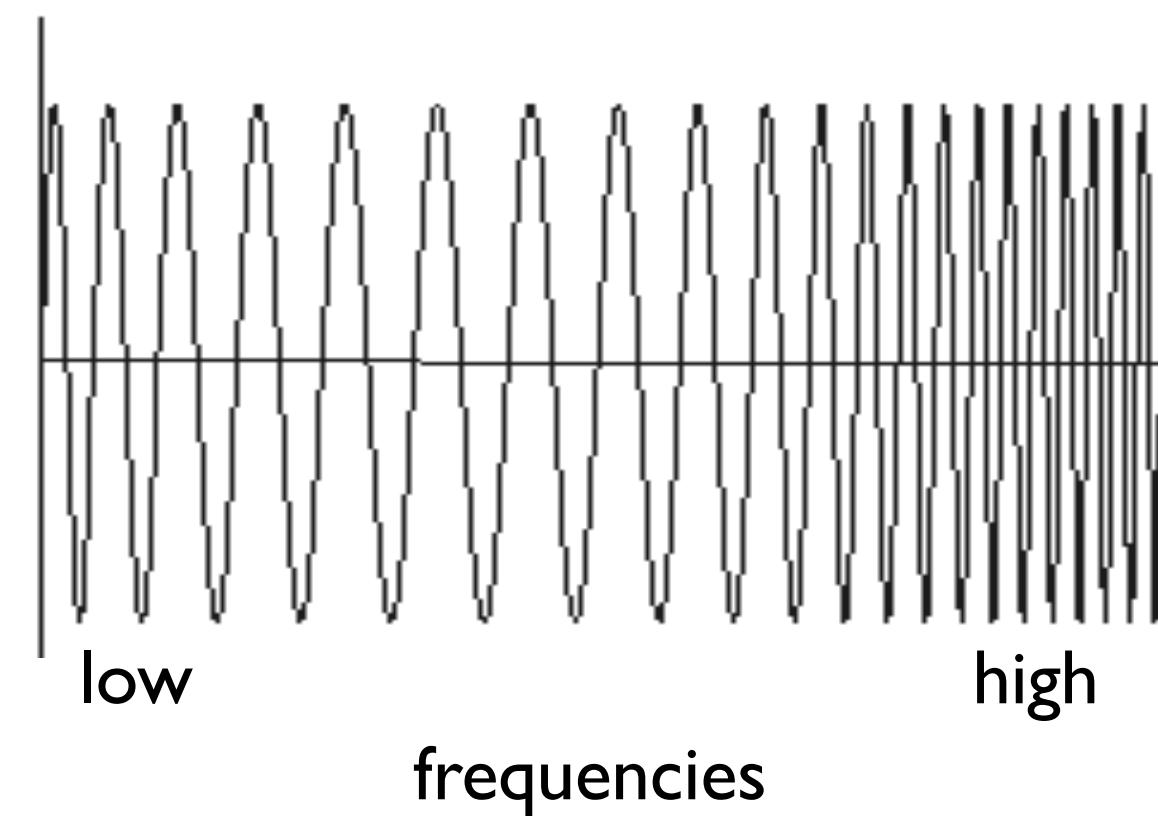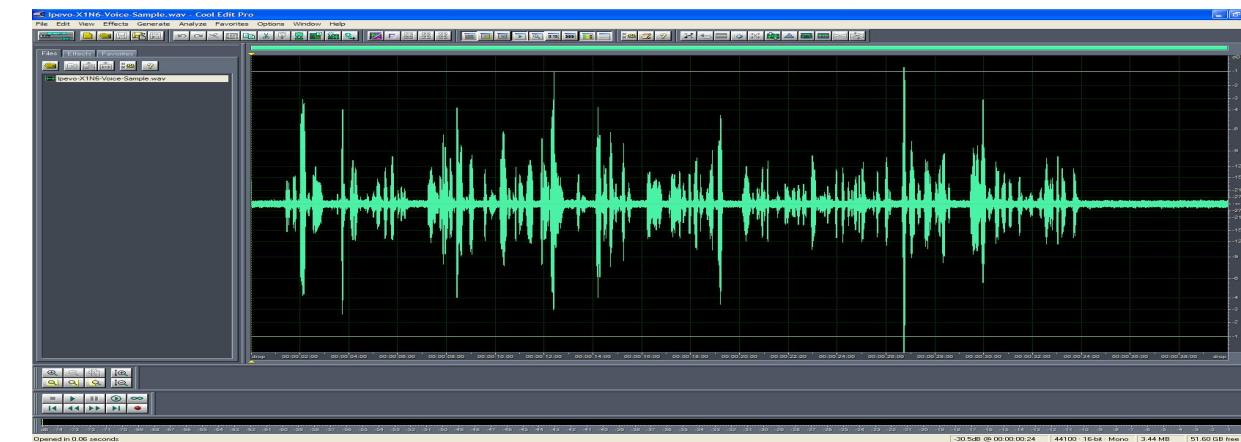
Reconstruction

- Making samples back into a continuous function
  - for output (need realizable method)
  - for analysis or processing (need mathematical method)
  - amounts to "guessing" what the function did in between
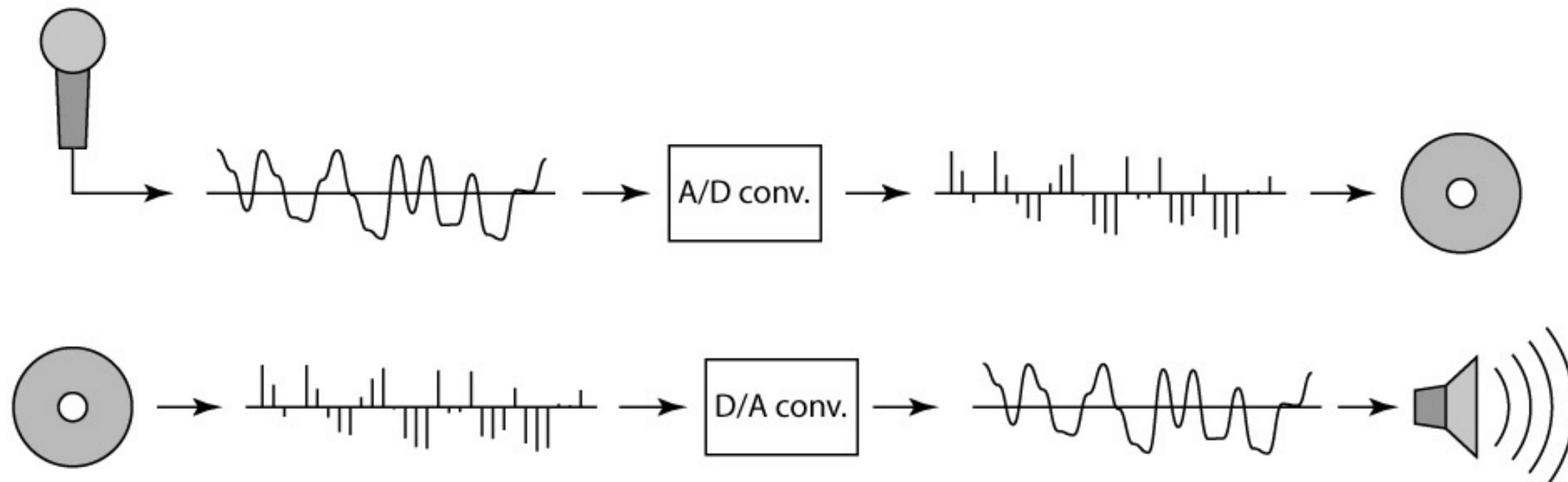
Reconstruction

[FvDFH fig.14.14b / Wolberg]

1D Example: Audio

low   high

frequencies
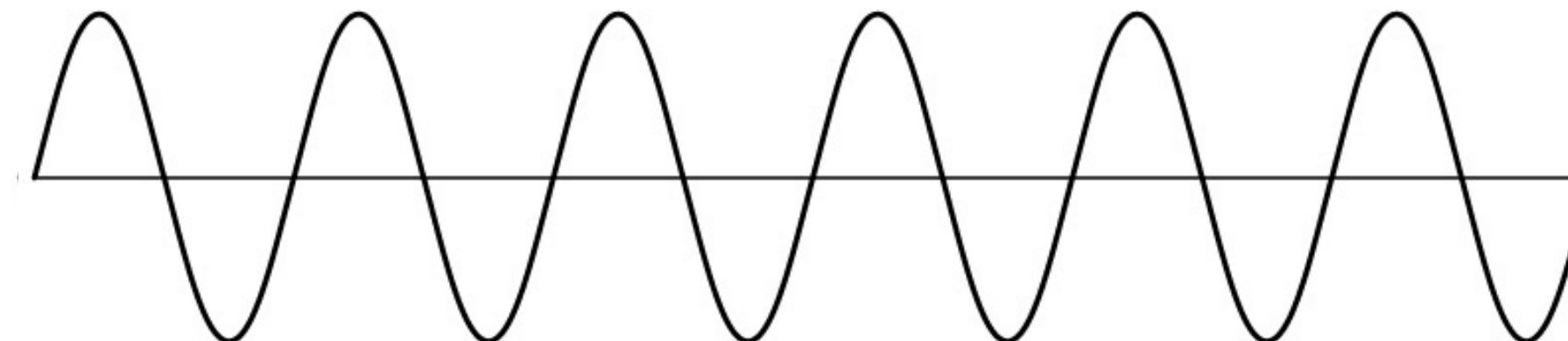
Sampling in Digital Audio

- Recording: sound to analog to samples to disc
- Playback: disc to samples to analog to sound again
  - how can we be sure we are filling in the gaps correctly?
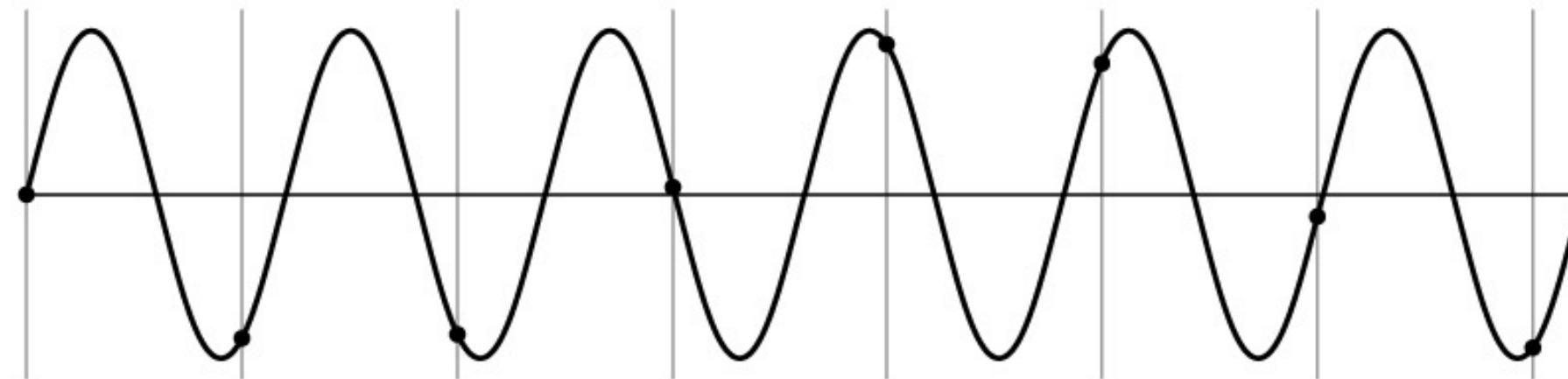
Sampling and Reconstruction

- Simple example: a sign wave
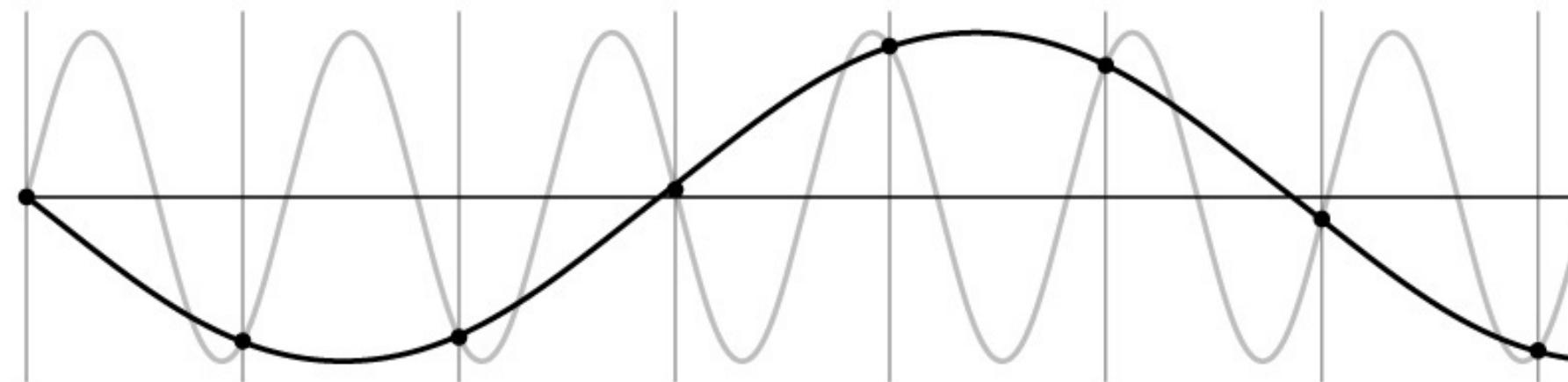
Undersampling

- What if we "missed" things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
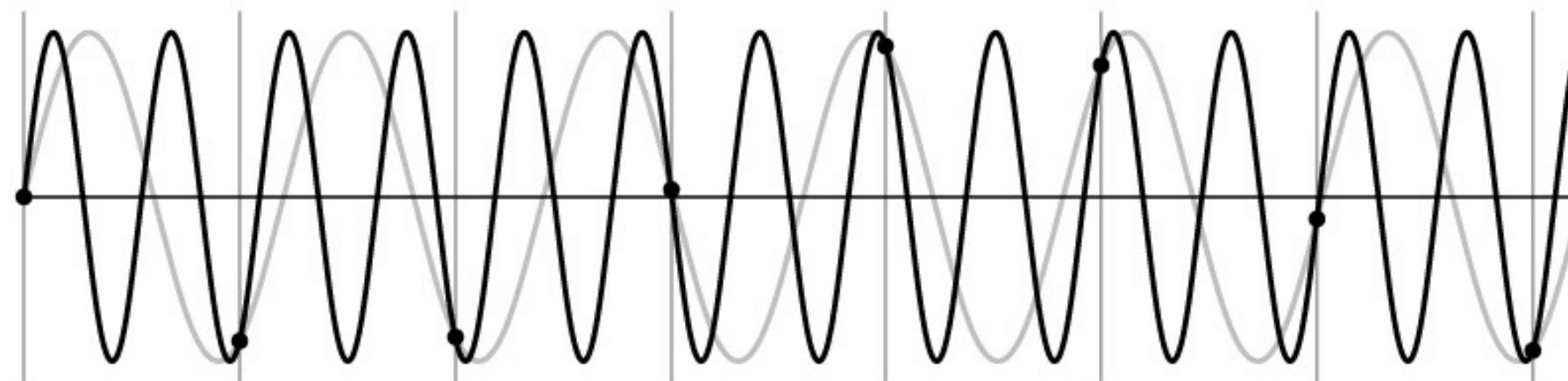
Undersampling

- What if we "missed" things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
  - surprising result: indistinguishable from lower frequency

Undersampling

- What if we "missed" things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
  - surprising result: indistinguishable from lower frequency
  - also, was always indistinguishable from higher frequencies
  - *aliasing*: signals "traveling in disguise" as other frequencies
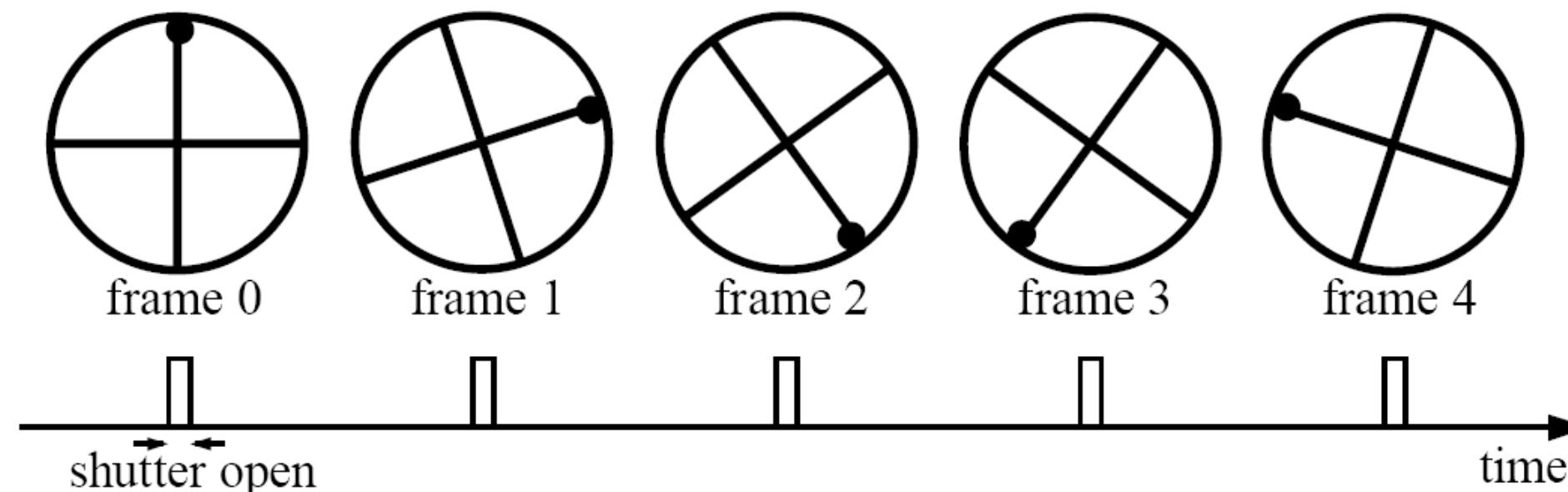
Aliasing in Videos

Imagine a spoked wheel moving to the right (rotating clockwise). Mark wheel with dot so we can see what's happening.

If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):

Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)

Aliasing in Images

Disintegrating textures
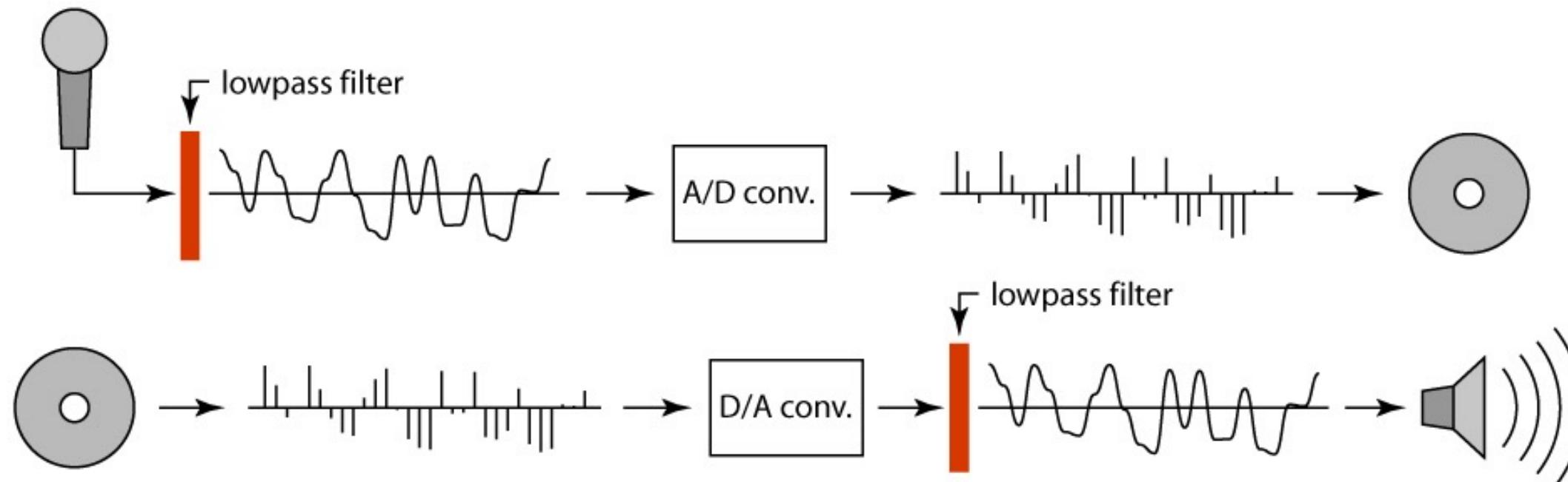
Antialiasing

- What can we do about aliasing?

- Sample more often
  - Join the Mega-Pixel craze of the photo industry
  - But this can't go on forever

- Make the signal less "wiggly"
  - Get rid of some high frequencies
  - Will loose information
  - But it's better than aliasing
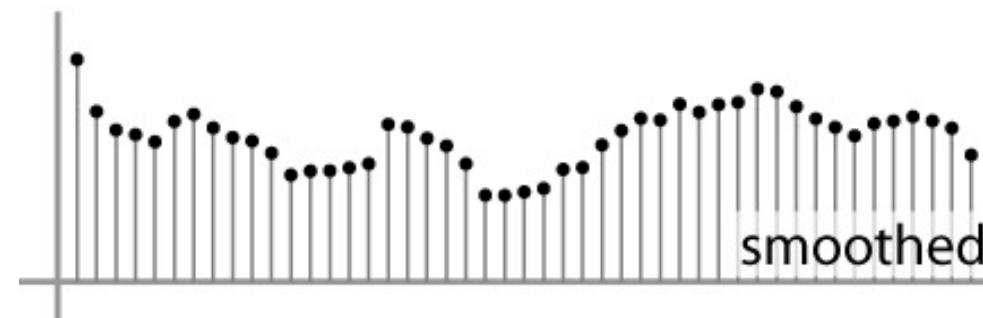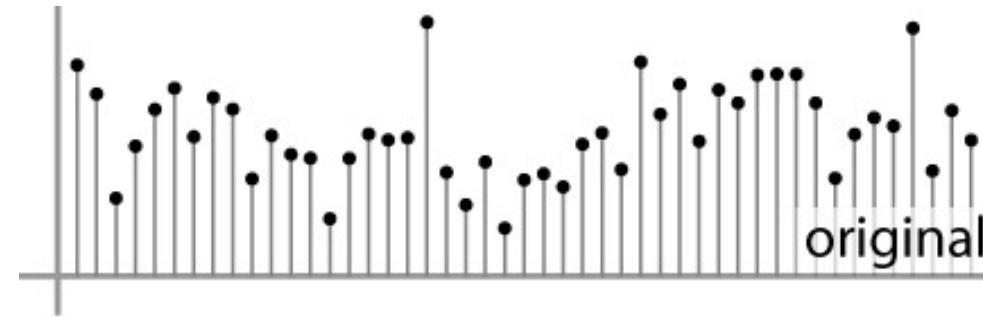
Preventing Aliasing

- Introduce lowpass filters:
  - remove high frequencies leaving only safe, low frequencies
  - choose lowest frequency in reconstruction (disambiguate)
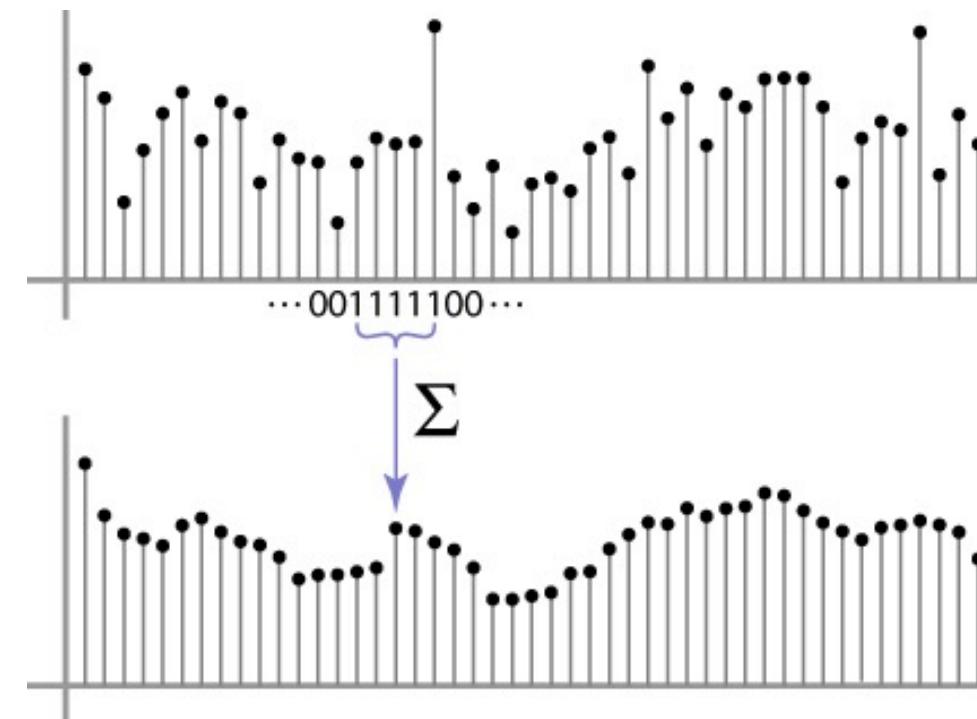
Moving Average

- basic idea: define a new function by averaging over a sliding window

- a simple example to start off: smoothing

original

smoothed

Moving Average

- Can add weights to our moving average
- *Weights* [..., 0, 1, 1, 1, 1, 1, 0, ...] / 5

$$\cdots 001111100 \cdots$$

$$\Sigma$$

Cross-correlation

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

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

- A "dot product" between local neighborhood and kernel for each pixel.

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

$$G = H \otimes F$$

2D Box Filter

$$h[\cdot,\cdot]$$

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

1	1	1
1	1	1
1	1	1

Image filtering

$$h[\cdot,\cdot] \quad \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$$f[\cdot,\cdot]$$

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

$$g[\cdot,\cdot]$$

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

Credit: S. Seitz

Image filtering

$$h[\cdot,\cdot] \quad \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

$$f[\cdot,\cdot] \qquad\qquad g[\cdot,\cdot]$$

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

0	10								

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

Credit: S. Seitz

Image filtering

$h[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

$g[.,.]$

	0	10	20						

Credit: S. Seitz

Image filtering

$$h[\cdot,\cdot] \quad \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

$f[\cdot,\cdot]$

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

$g[\cdot,\cdot]$

	0	10	20	30					

Credit: S. Seitz

Image filtering

$$h[\cdot,\cdot] \quad \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$$f[\cdot,\cdot]$$

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

$$g[\cdot,\cdot]$$

	0	10	20	30	30				

41

Credit: S. Seitz

Image filtering

$$h[\cdot,\cdot] \quad \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$$f[\cdot,\cdot] \qquad\qquad g[\cdot,\cdot]$$

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

	0	10	20	30	30				
			?						

Image filtering

$$h[\cdot,\cdot] \quad \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$$f[\cdot,\cdot]$$

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

$$g[\cdot,\cdot]$$

	0	10	20	30	30				
					?				
			50						

Credit: S. Seitz

Image filtering

$$h[\cdot,\cdot] \quad \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$$f[\cdot,\cdot]$$

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

$$g[\cdot,\cdot]$$

	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	

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

44

Credit: S. Seitz
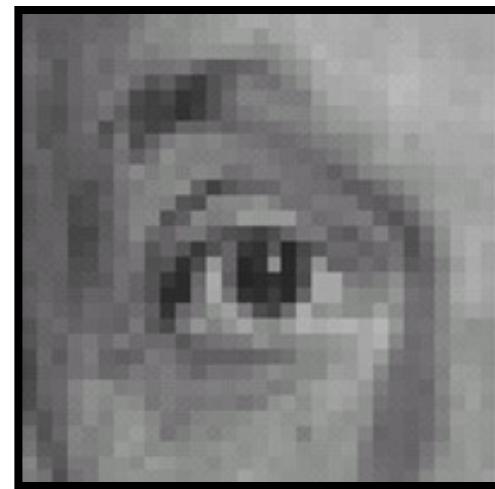
2D Box Filter

What does it do?

- Replaces each pixel with an average of its neighborhood

- Achieve smoothing effect (remove sharp features)

$$h[\cdot,\cdot]$$

$$\frac{1}{9}
\begin{array}{|c|c|c|}
\hline
1 & 1 & 1 \\
\hline
1 & 1 & 1 \\
\hline
1 & 1 & 1 \\
\hline
\end{array}$$
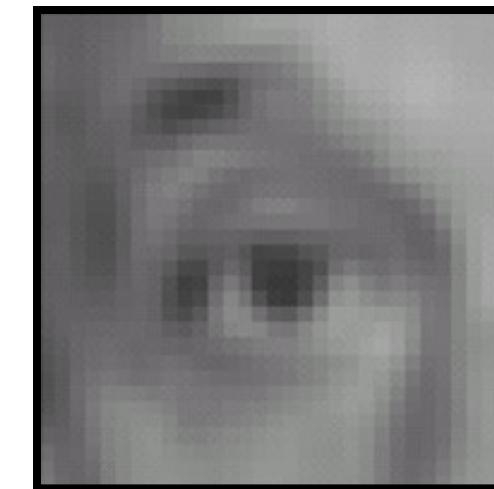
Linear filters: Examples
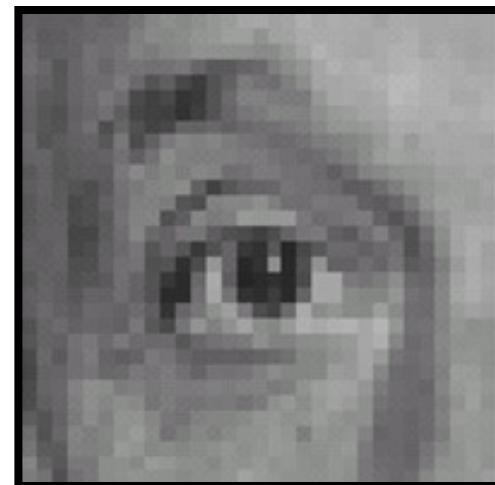
Original

$$\frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} =$$

Blur (with a mean filter)

Practice with Linear Filters
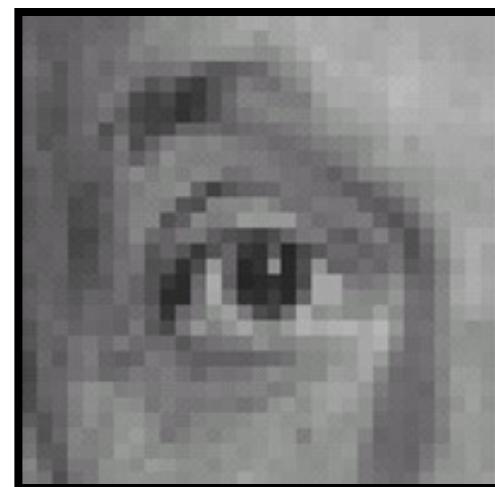
Original

0	0	0
0	1	0
0	0	0

?
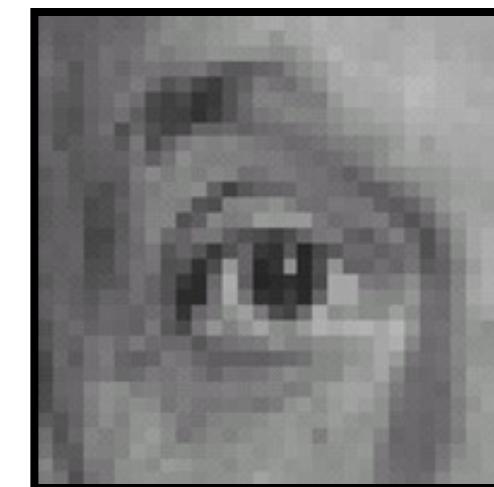
Practice with Linear Filters
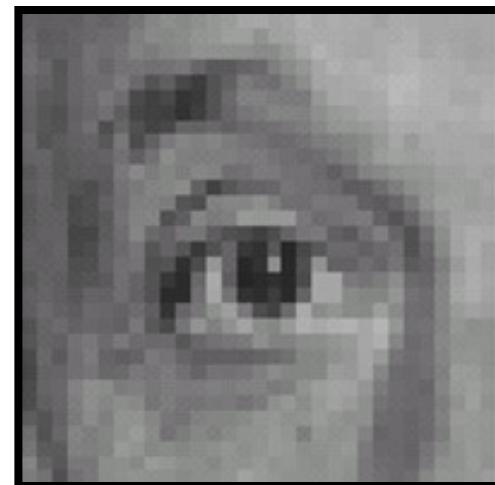
Original

0	0	0
0	1	0
0	0	0

Filtered
(no change)

Practice with Linear Filters

0	0	0
0	0	1
0	0	0

?
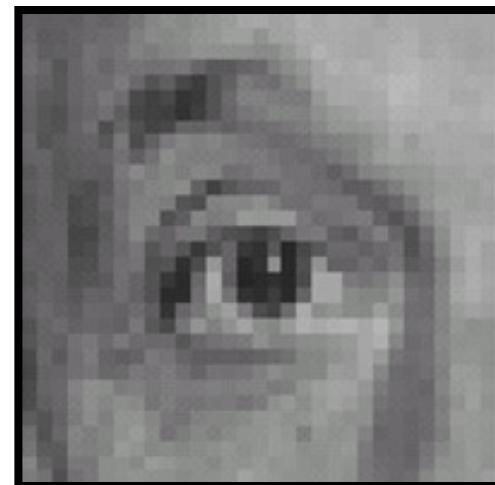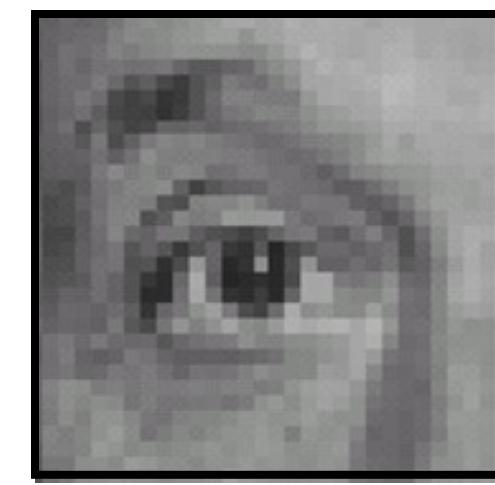
Original

Practice with Linear Filters
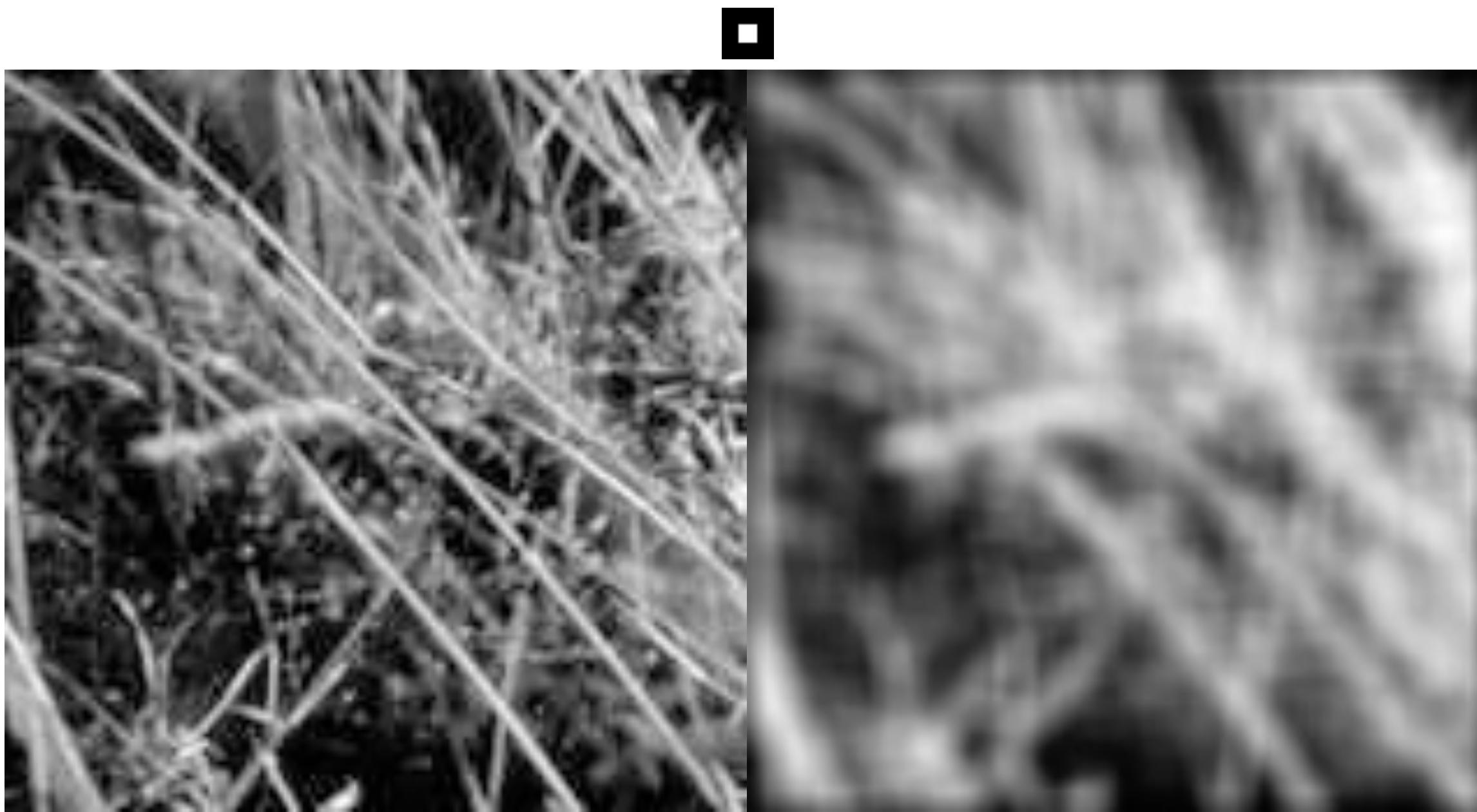
Original

0	0	0
0	0	1
0	0	0
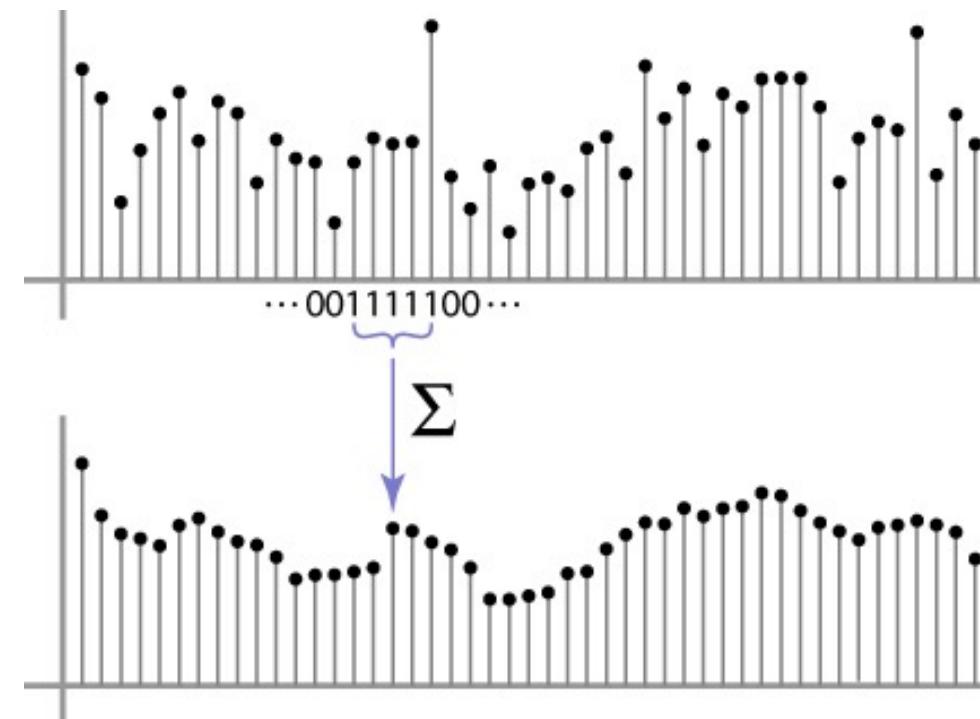
Shifted left
By 1 pixel

Source: D. Lowe

Back to 2D Box Filter

Moving Average

- Can add weights to our moving average
- *Weights* [..., 0, 1, 1, 1, 1, 1, 0, ...] / 5

··· 001111100 ···

Σ
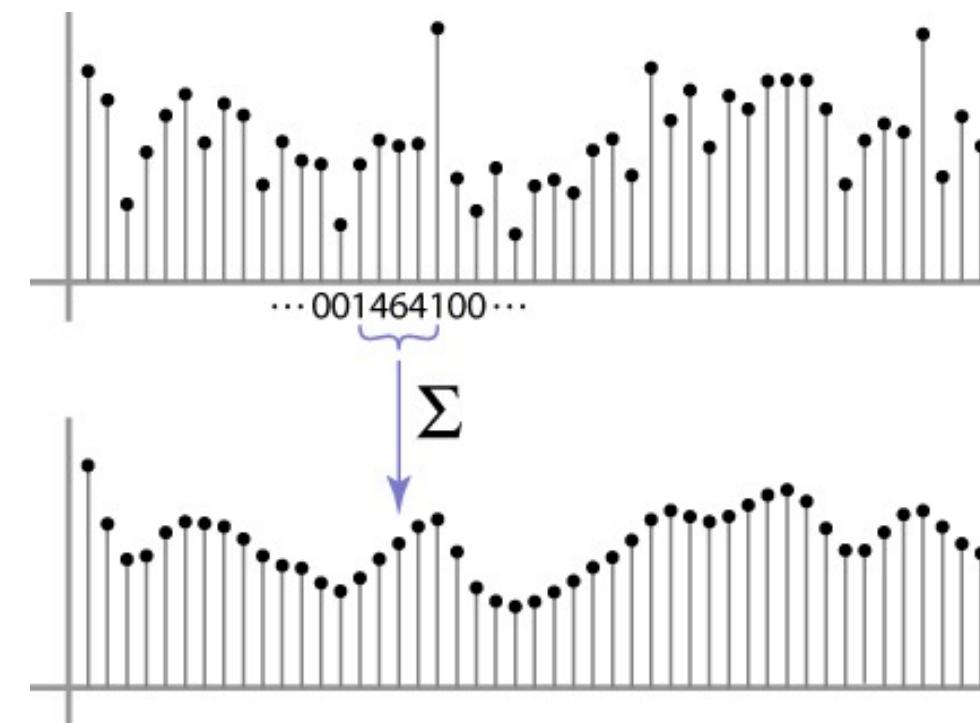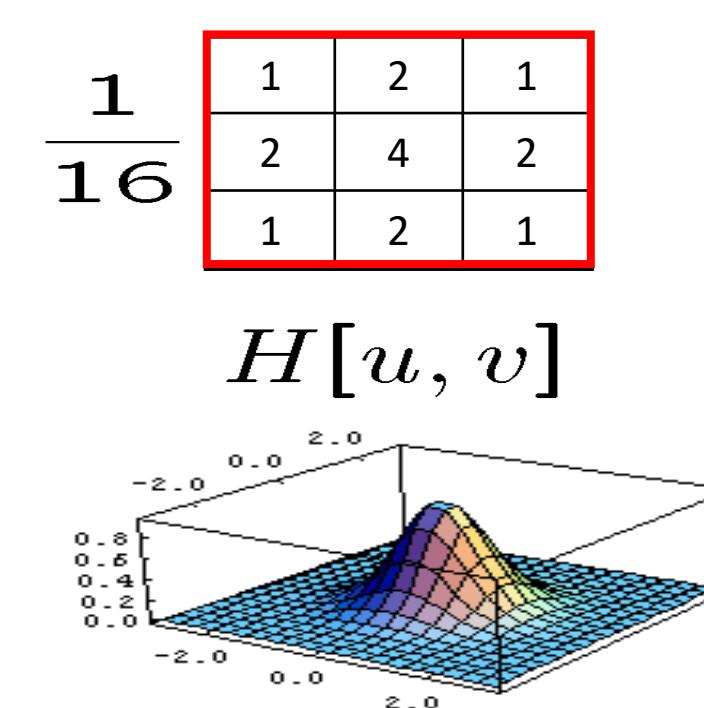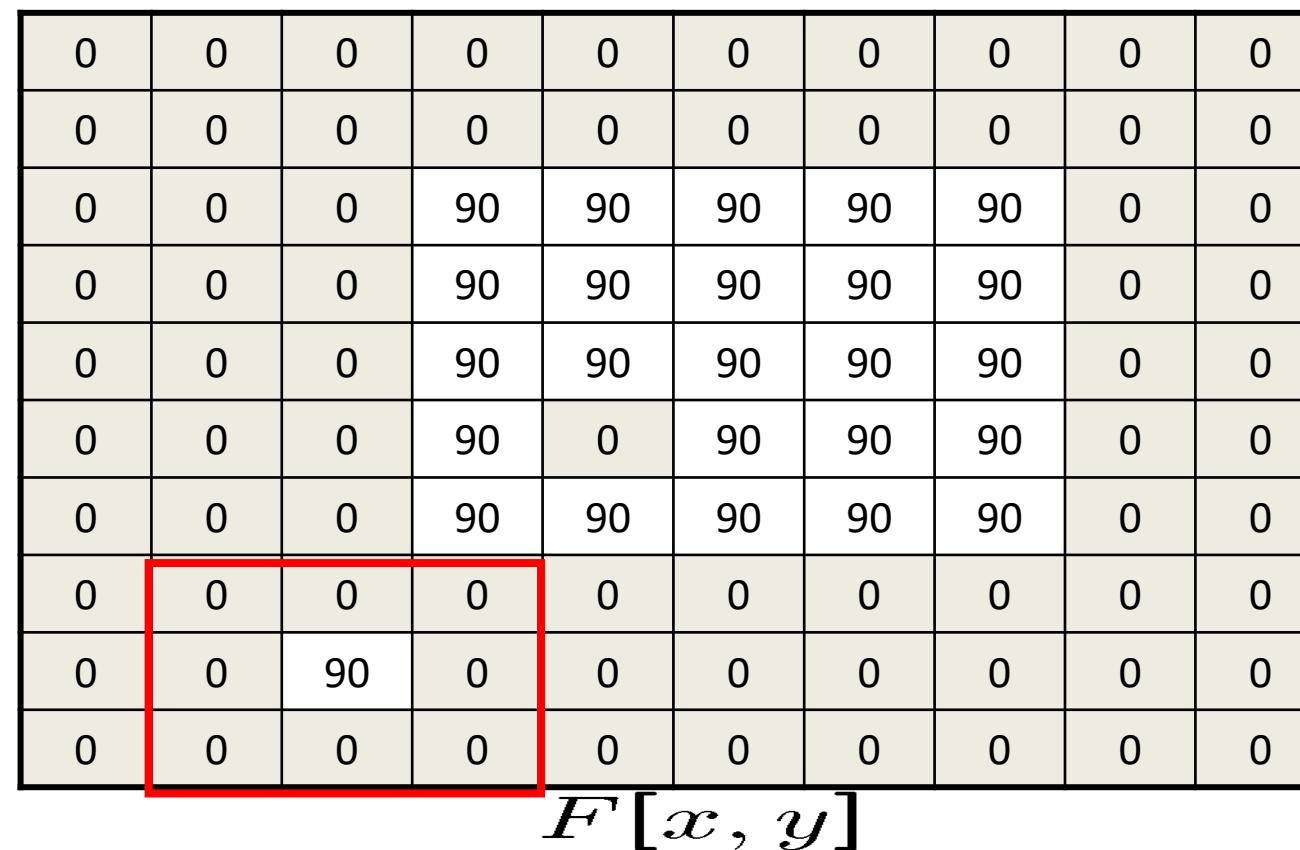
Weighted Moving Average

- bell curve (gaussian-like) weights [..., 1, 4, 6, 4, 1, ...]

Gaussian filtering
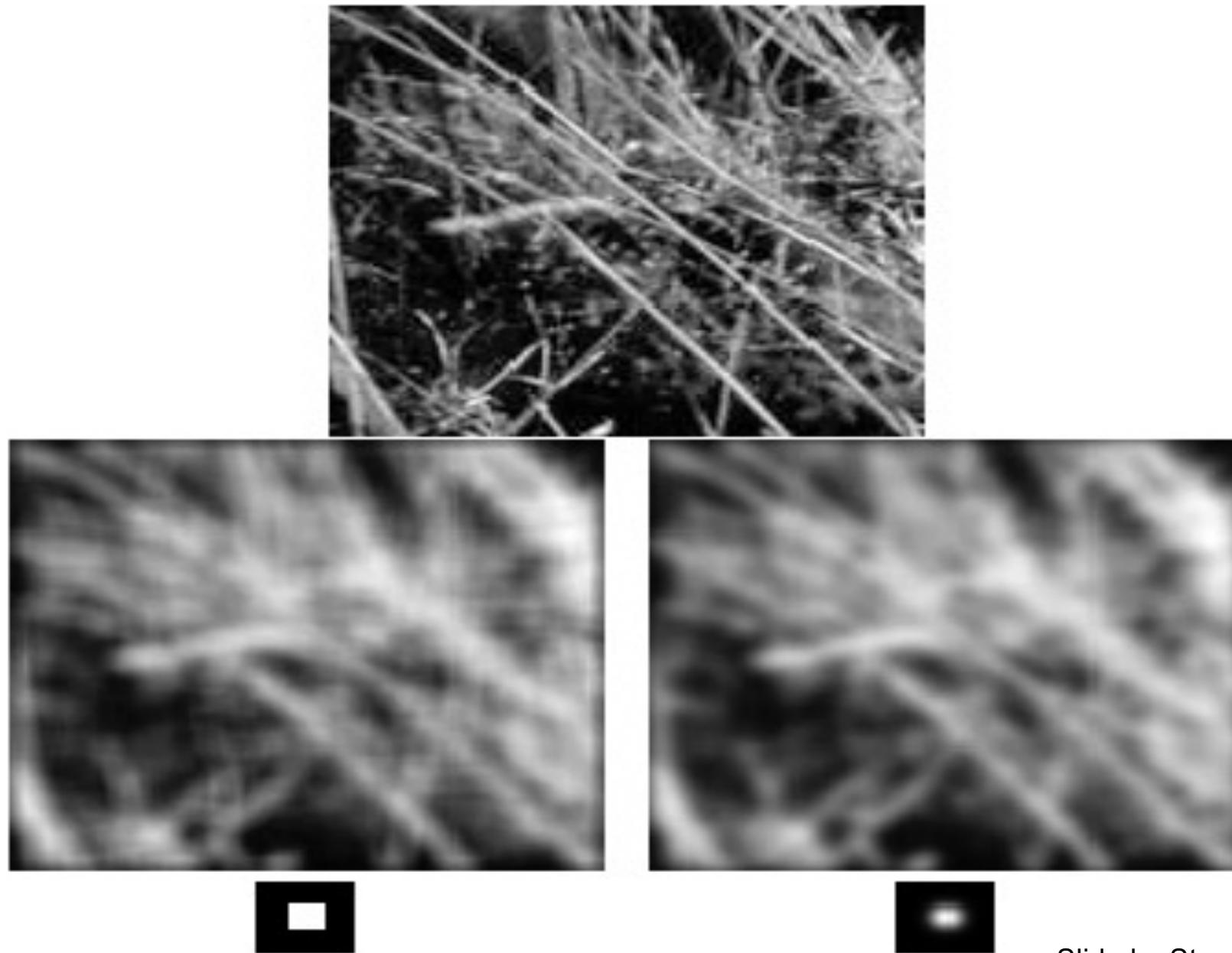
- A Gaussian kernel gives less weight to pixels further from the center of the window

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

$$F[x, y]$$

$$\frac{1}{16} \quad \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

$$H[u, v]$$

- This kernel is an approximation of a Gaussian function:

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

54
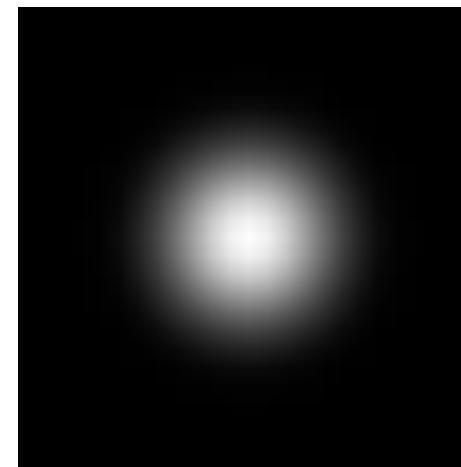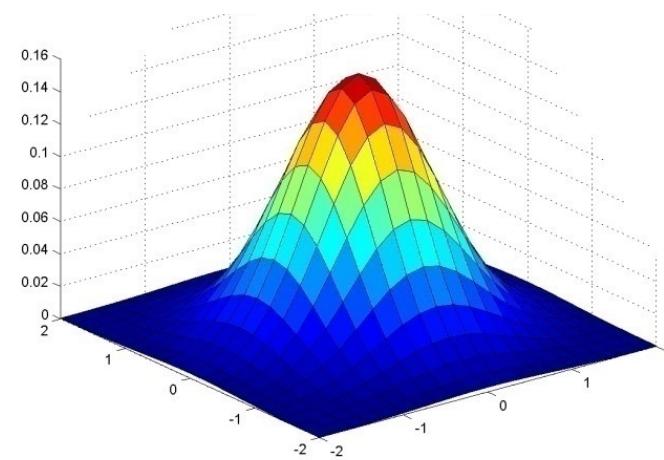
Mean vs. Gaussian Filtering

55

Gaussian filtering

- Weight contributions of neighboring pixels by nearness
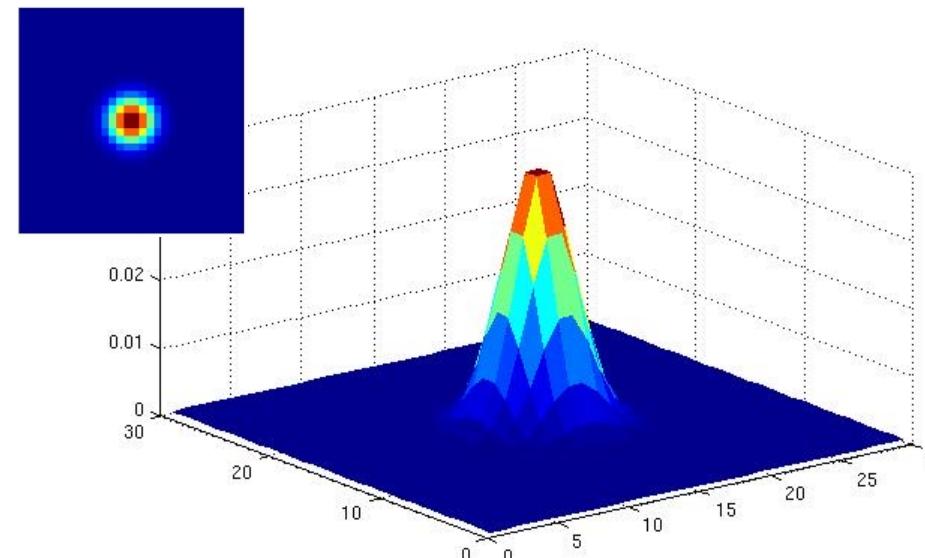
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$
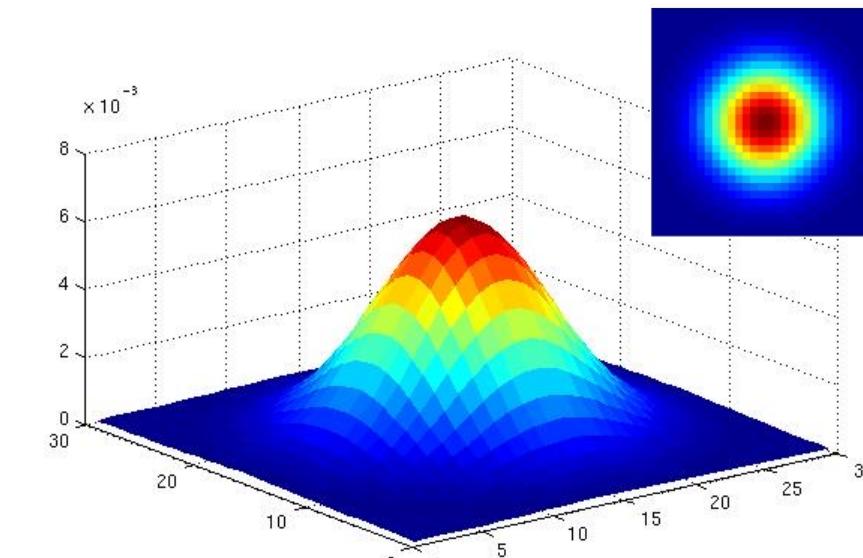
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Gaussian Kernel

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$
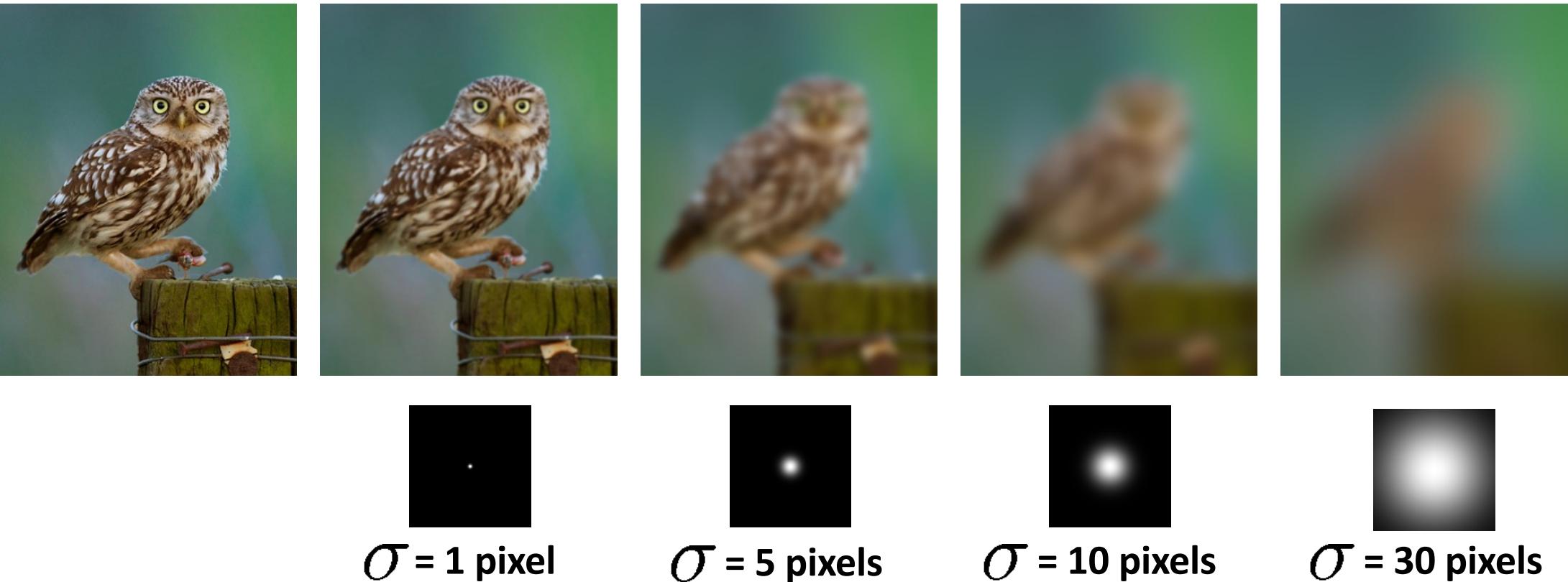
σ = 2 with 30 x 30 kernel

σ = 5 with 30 x 30 kernel
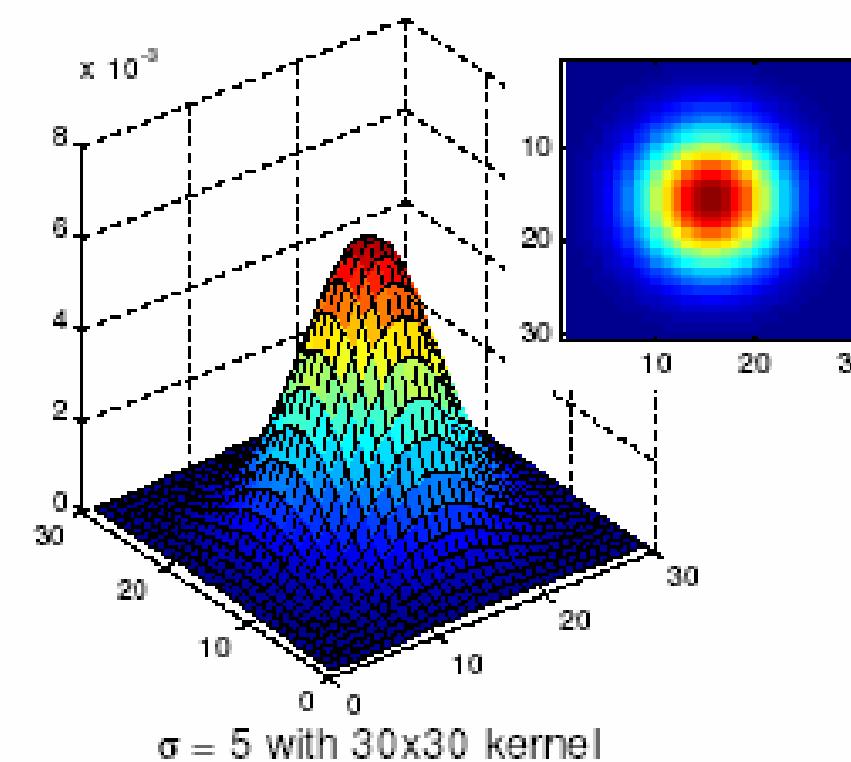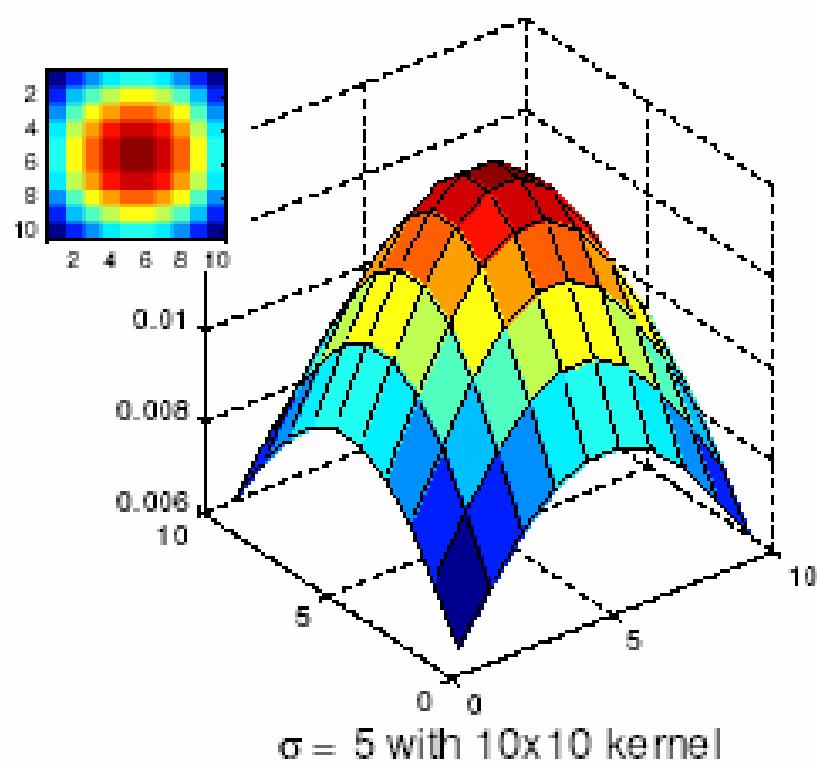
- Standard deviation σ: determines extent of smoothing

57

Gaussian filters

σ = 1 pixel σ = 5 pixels σ = 10 pixels σ = 30 pixels

Choosing kernel width

- Gaussian function has infinite support, but discrete filters use finite kernels

$\sigma = 5$ with 10x10 kernel

$\sigma = 5$ with 30x30 kernel

Source: K. Grauman

Cross-correlation vs. Convolution

- **cross-correlation**: $G[i, j] = \sum\limits_{u=-k}^{k} \sum\limits_{v=-k}^{k} H[u, v] F[i + u, j + v]$

$$G = H \otimes F$$

- A **convolution** operation is a cross-correlation where the filter is flipped both horizontally and vertically before being applied to the image:
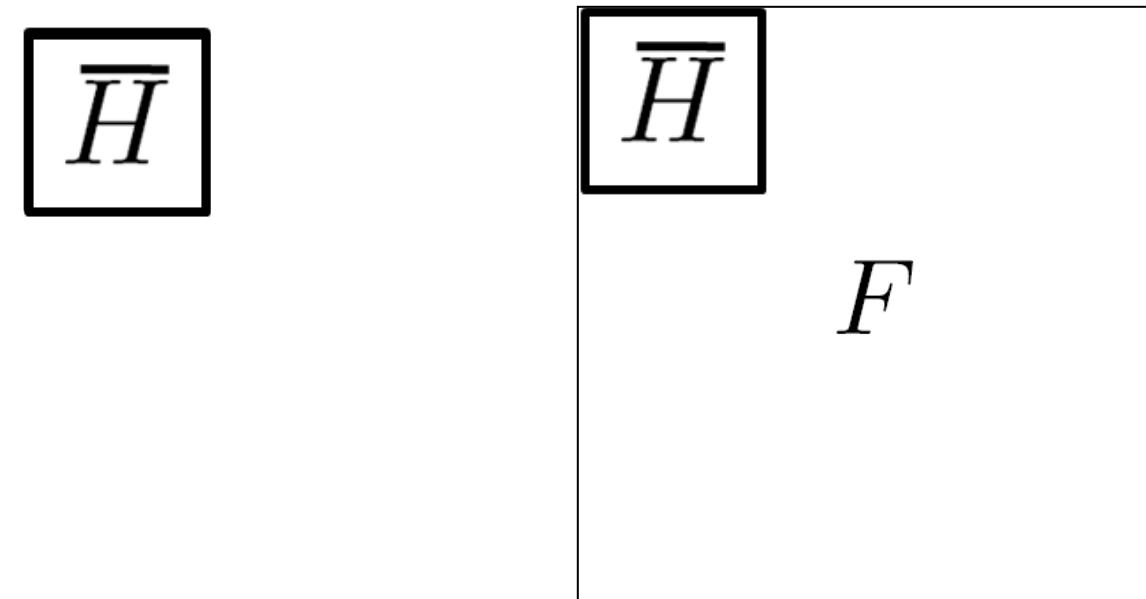
$$G[i, j] = \sum\limits_{u=-k}^{k} \sum\limits_{v=-k}^{k} H[u, v] F[i - u, j - v]$$

- It is written:

$$G = H \star F$$

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

Convolution

$$\overline{H}$$

$$\overline{H}$$

$$F$$

Different DL libraries handle it differently. Some flips; some not.
Be careful when you port Conv weights from one library to another.
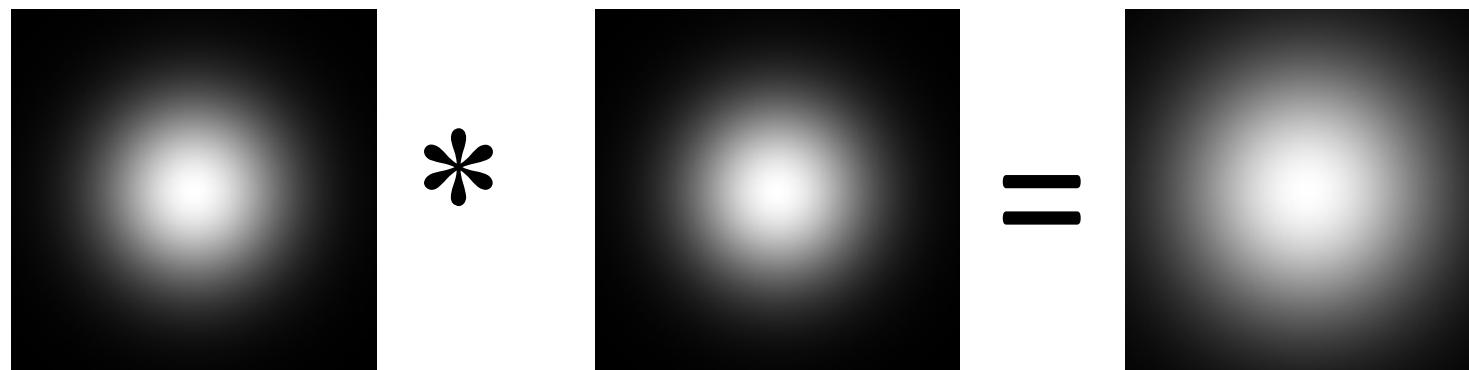
Computer vision 101: most students fail to flip the kernel.

Convolution is nice!

- Notation: $b = c * a$
- Convolution is a multiplication-like operation
  - Commutative: $a * b = b * a$
  - Associative: $a * (b * c) = (a * b) * c$
  - distributes over addition: $a * (b + c) = a * b + a * c$
  - scalars factor out: $\alpha a * b = a * \alpha b = \alpha(a * b)$
  - identity: unit impulse e = [..., 0, 0, 1, 0, 0, ...]: $a * e = a$
- Conceptually no distinction between filter and signal
- Usefulness of associativity
  - often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - this is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$

Gaussian and Convolution

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

- Convolution with self is another Gaussian

 * **=**

 — Convolving twice with Gaussian kernel of width

= convolving once with kernel of width σ

$$\sigma\sqrt{2}$$

Source: K. Grauman
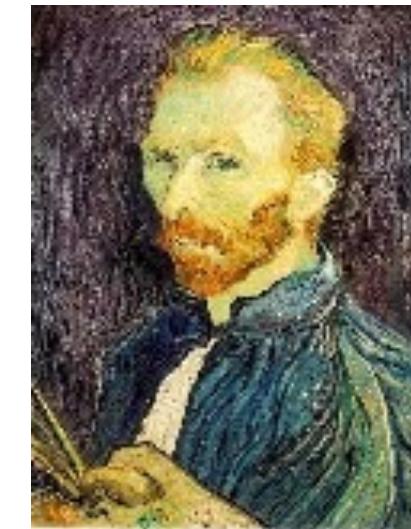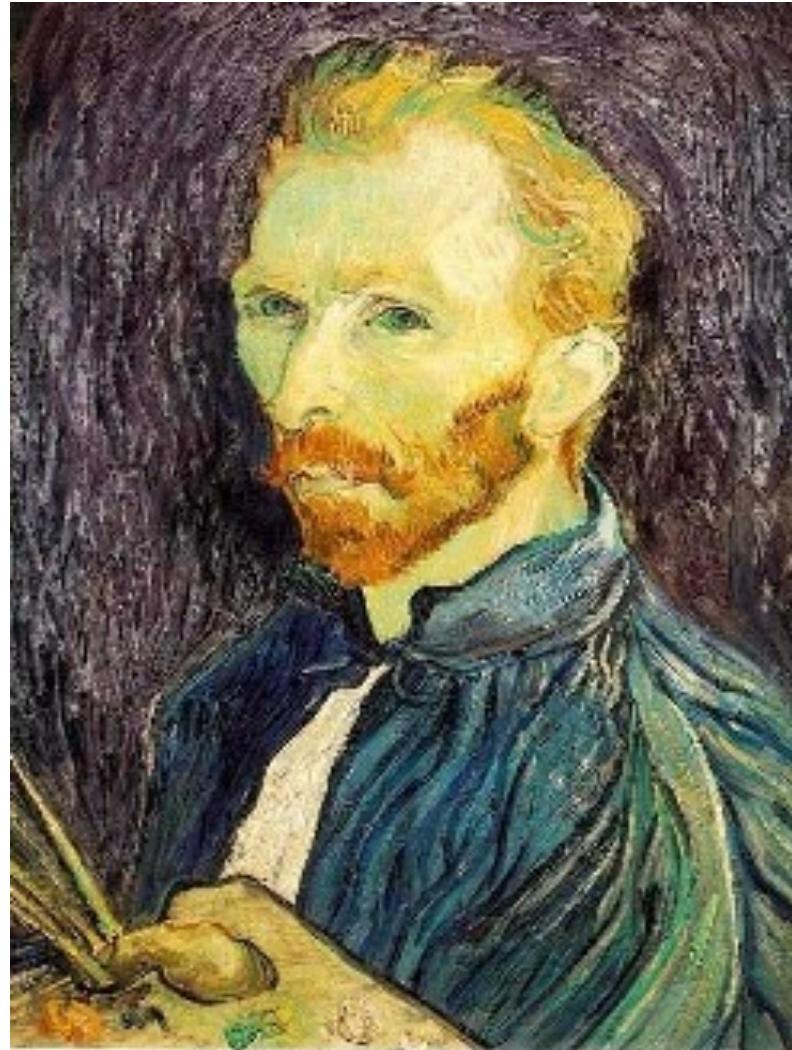
Image h

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
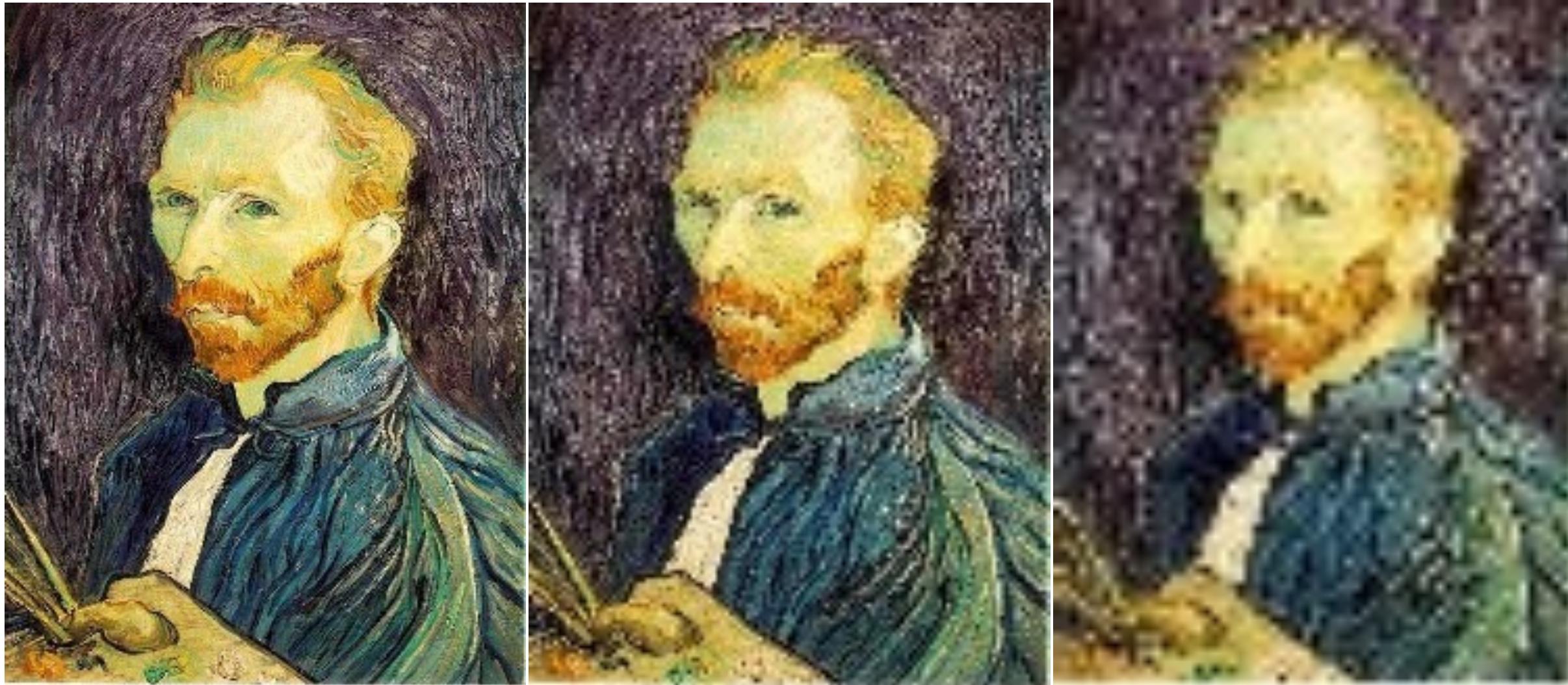
1/4

1/8

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

Image sub-sampling
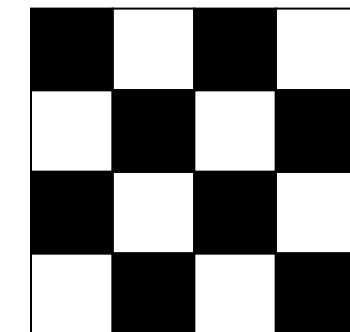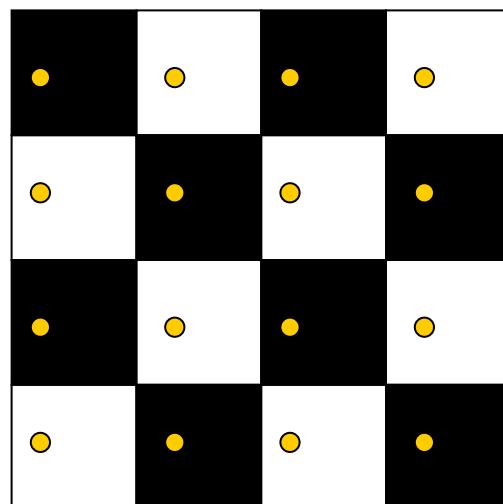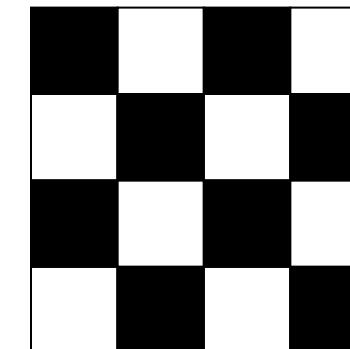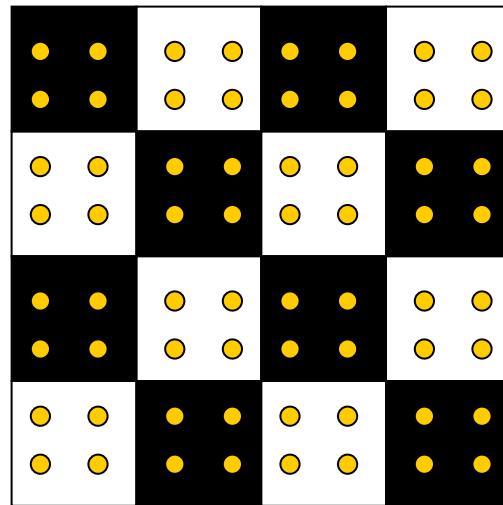
1/2   1/4 (2x zoom)   1/8 (4x zoom)
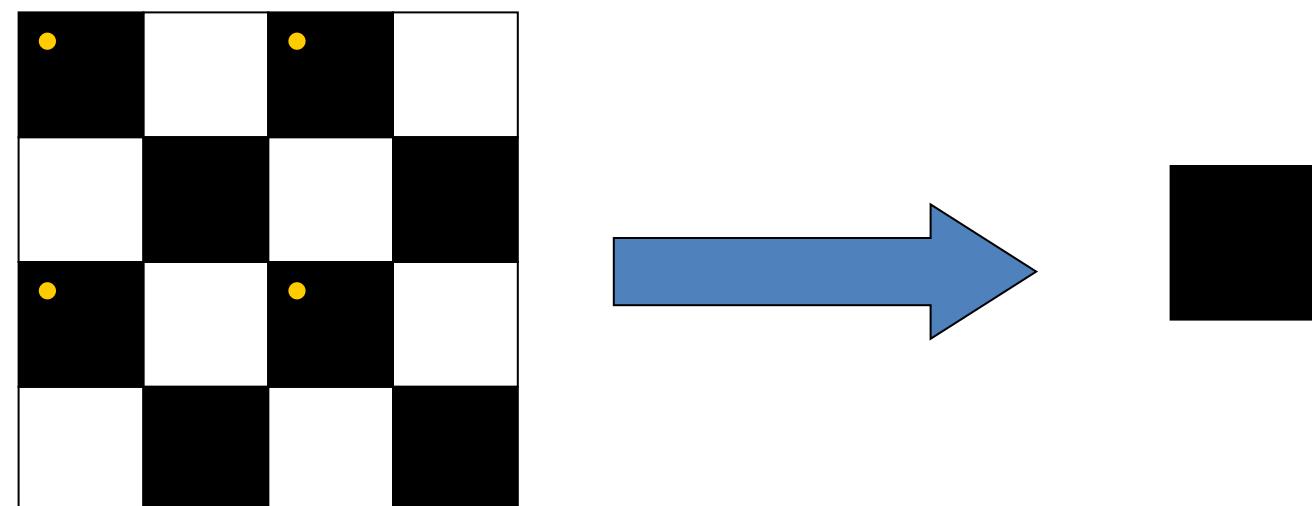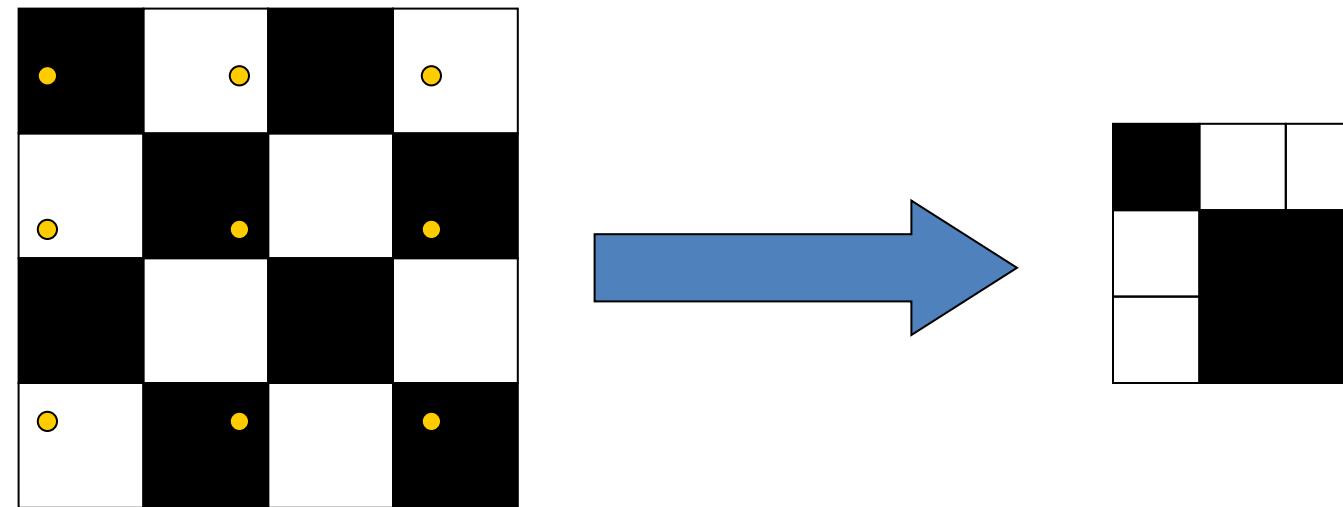
Aliasing! What do we do?

Sampling an image
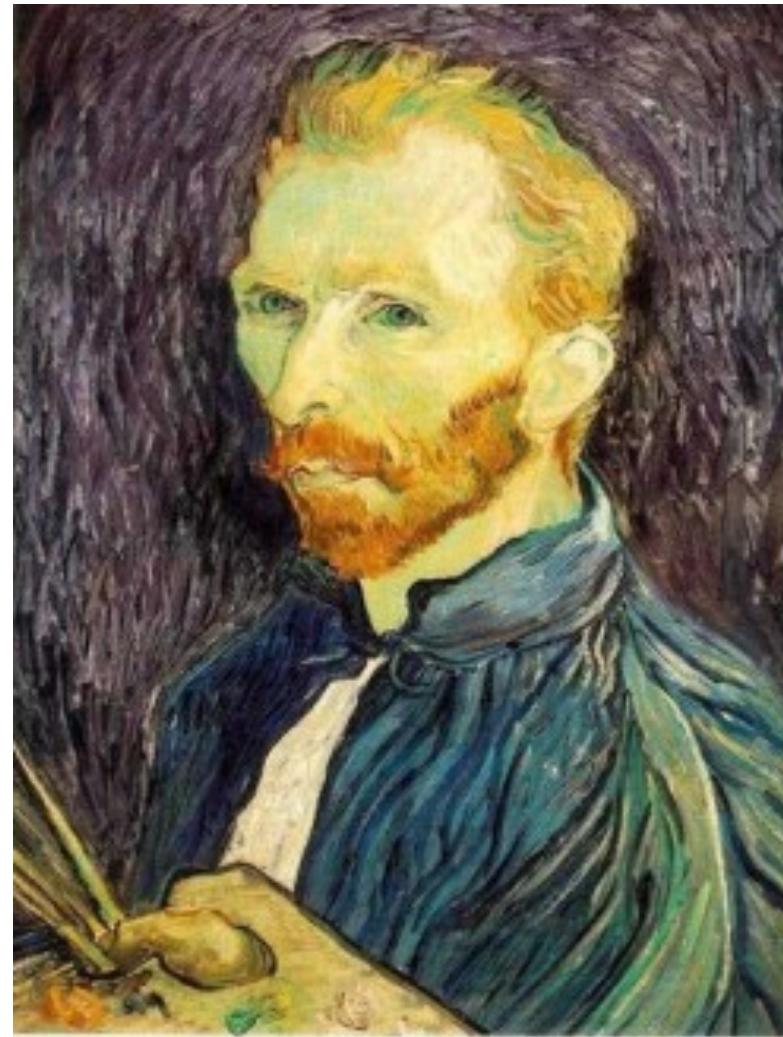
Examples of GOOD sampling

Undersampling
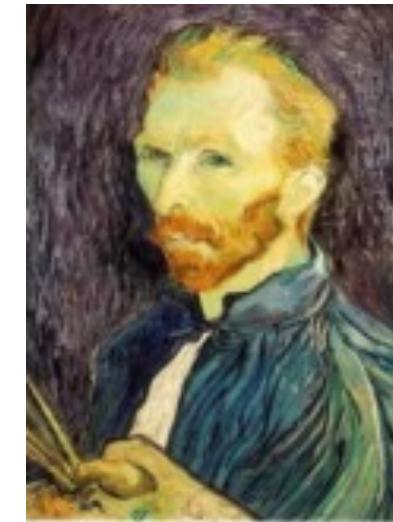
Examples of BAD sampling -> Aliasing

Gaussian (lowpass) pre-filtering

Gaussian 1/2

G 1/4
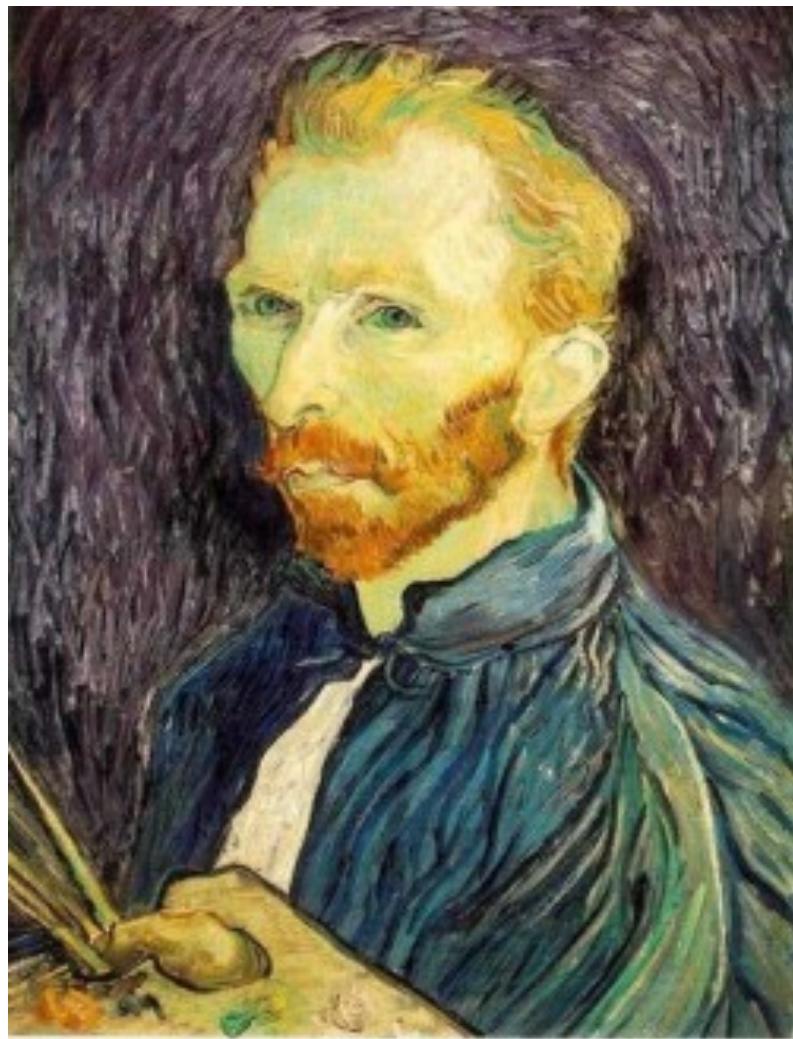
G 1/8
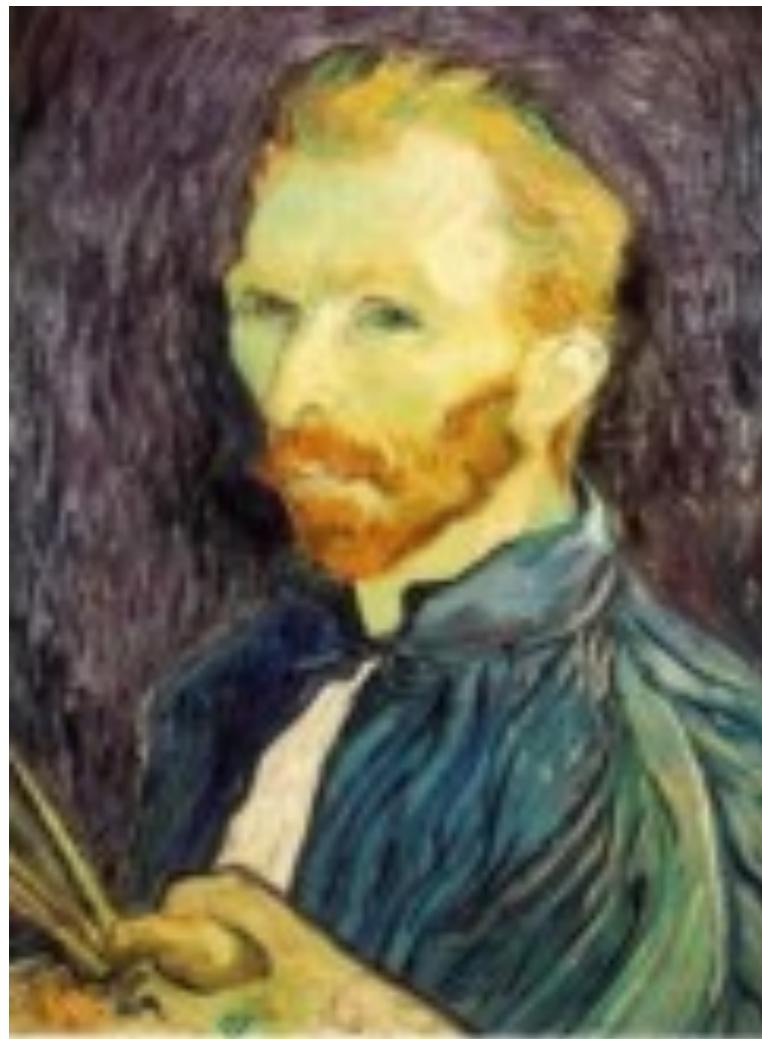
Solution: filter the image, *then* subsample

- Filter size should double for each ½ size reduction. Why?

Subsampling with Gaussian pre-filtering



Gaussian 1/2   G 1/4   G 1/8

70
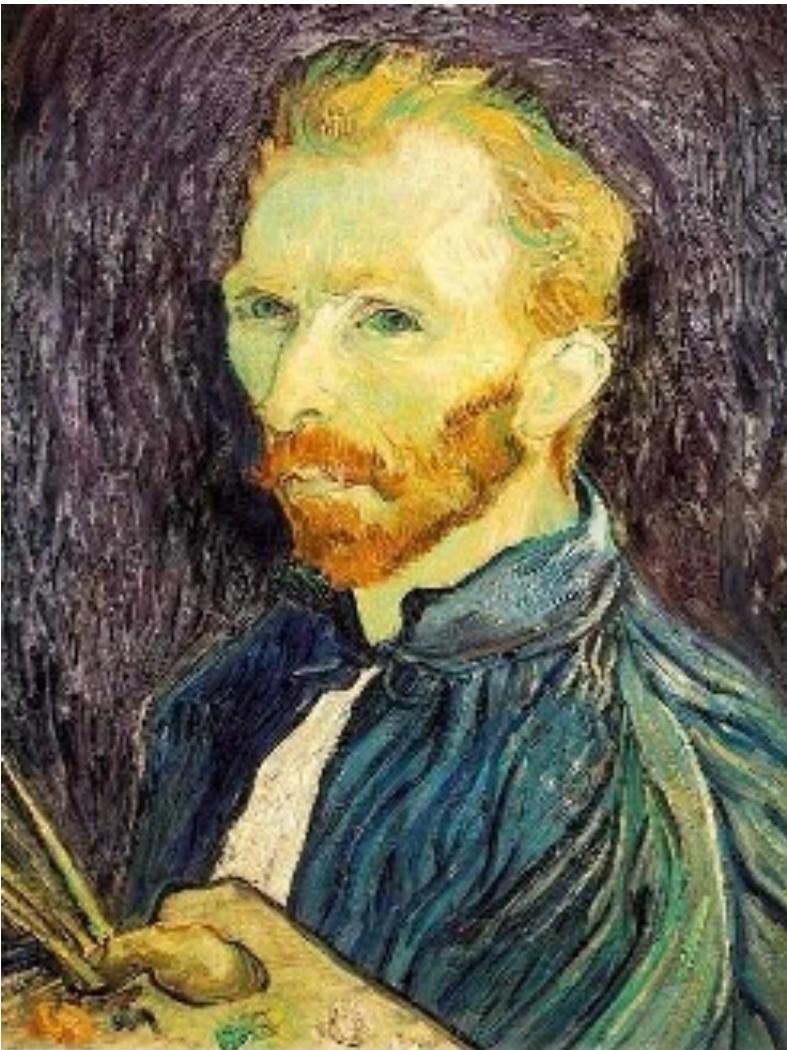
# Compare with…
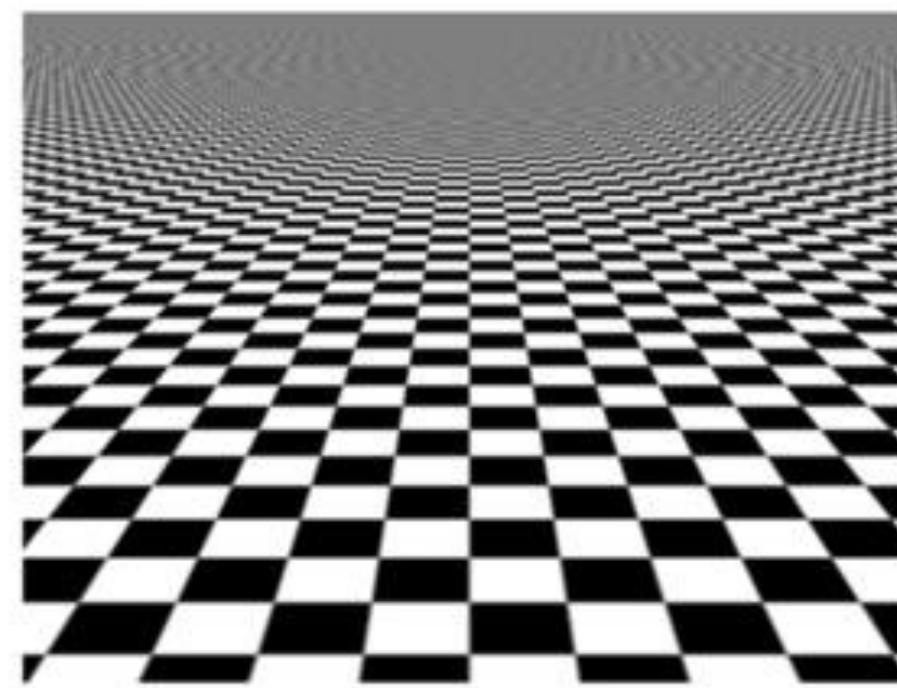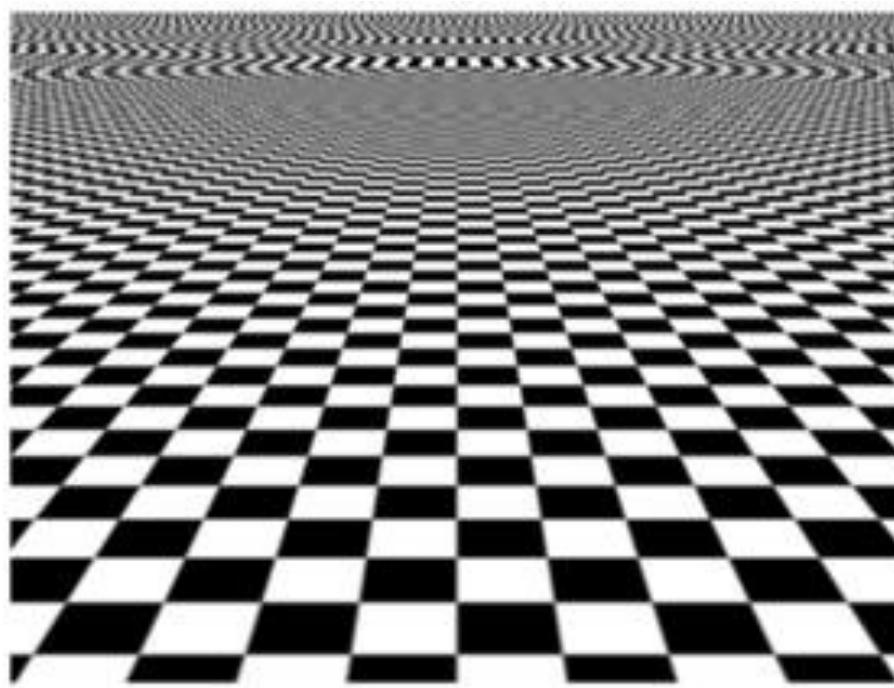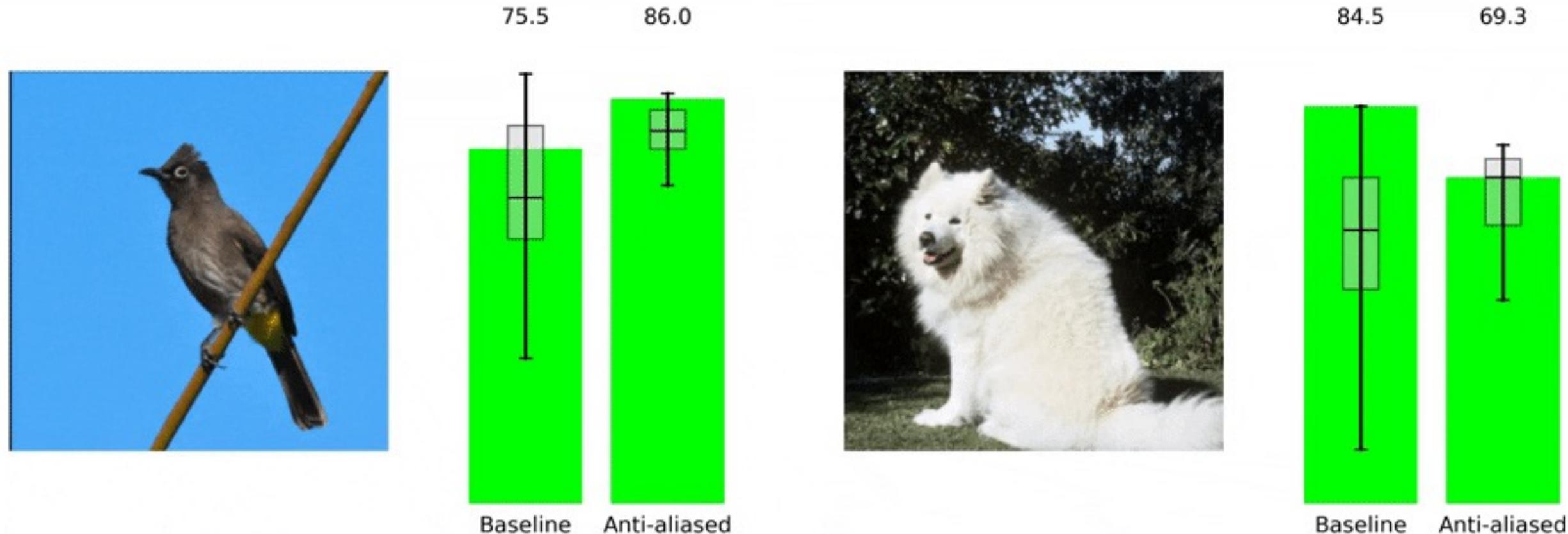
1/2   1/4 (2x zoom)   1/8 (4x zoom)

More Gaussian pre-filtering
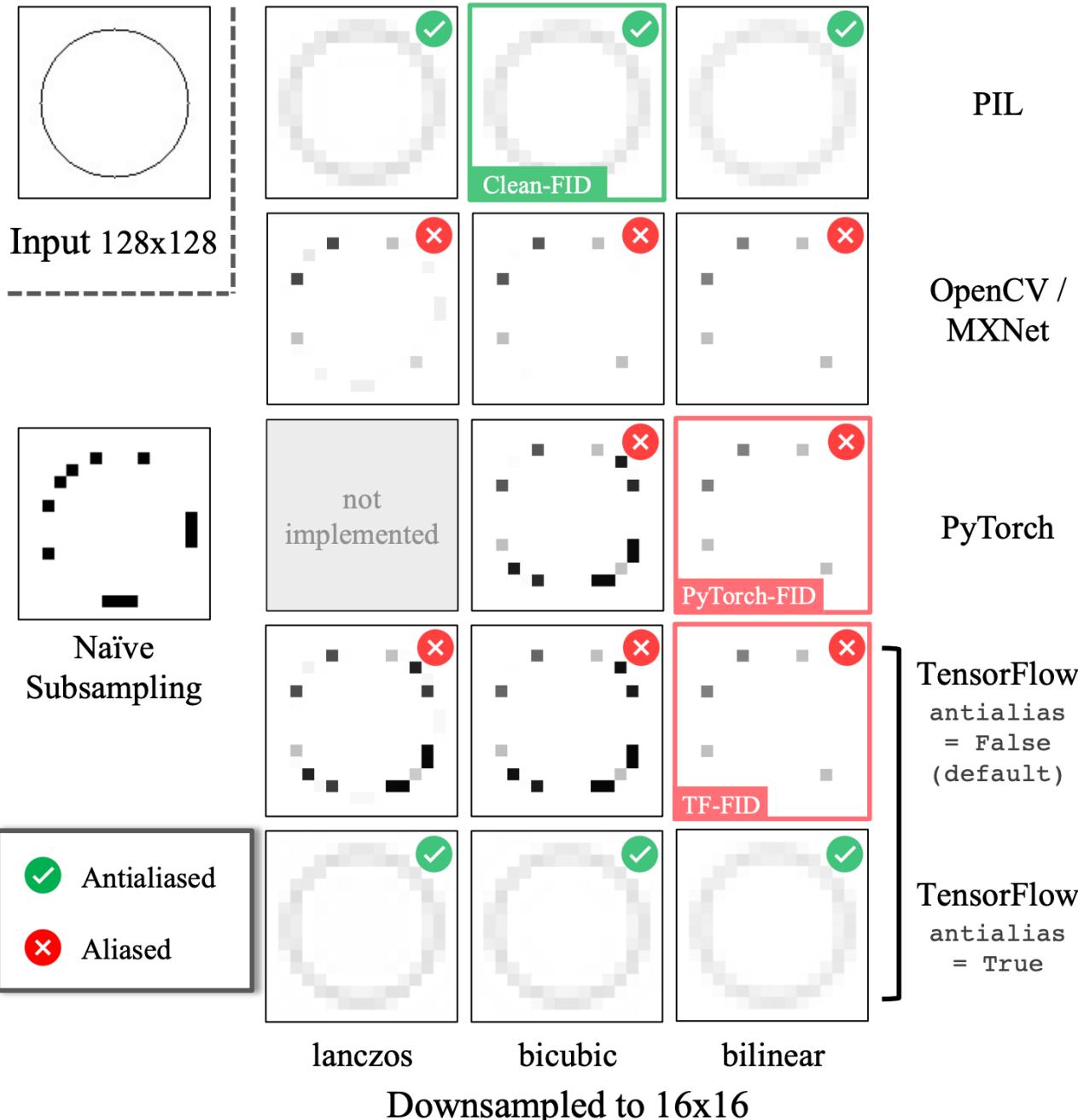
Aliasing in Deep Learning

More and more neural networks (e.g., StyleGAN2, StyleGAN3) use anti-aliasing layers
See Zhang. ICML 2019. "Making Convolutional Networks Shift-Invariant Again"

Aliasing in Deep Learning

Input 128x128

Naïve Subsampling

Antialiased

Aliased

PIL

OpenCV / MXNet

PyTorch

Clean-FID

PyTorch-FID

TF-FID

TensorFlow
antialias = False (default)

TensorFlow
antialias = True

lanczos   bicubic   bilinear

Downsampled to 16x16

- Many deep learning & computer vision libraries **fail** to pre-filter images correctly before downsampling. They use a fixed filter size for different resizing ratios.

- If you use them to resize your images, your results may contain aliasing artifacts.

- Recommendation:
  Use PIL. Use antialias=True.

74

Clean-FID [Parmer et al., 2021]

Animal Faces-HQ dataset (AFHQ)

We release a new dataset of animal faces, Animal Faces-HQ (AFHQ), consisting of 15,000 high-quality images at 512×512 resolution. The figure above shows example images of the AFHQ dataset. The dataset includes three domains of cat, dog, and wildlife, each providing about 5000 images. By having multiple (three) domains and diverse images of various breeds per each domain, AFHQ sets a challenging image-to-image translation problem. For each domain, we select 500 images as a test set and provide all remaining images as a training set. To download the dataset, run the following command:
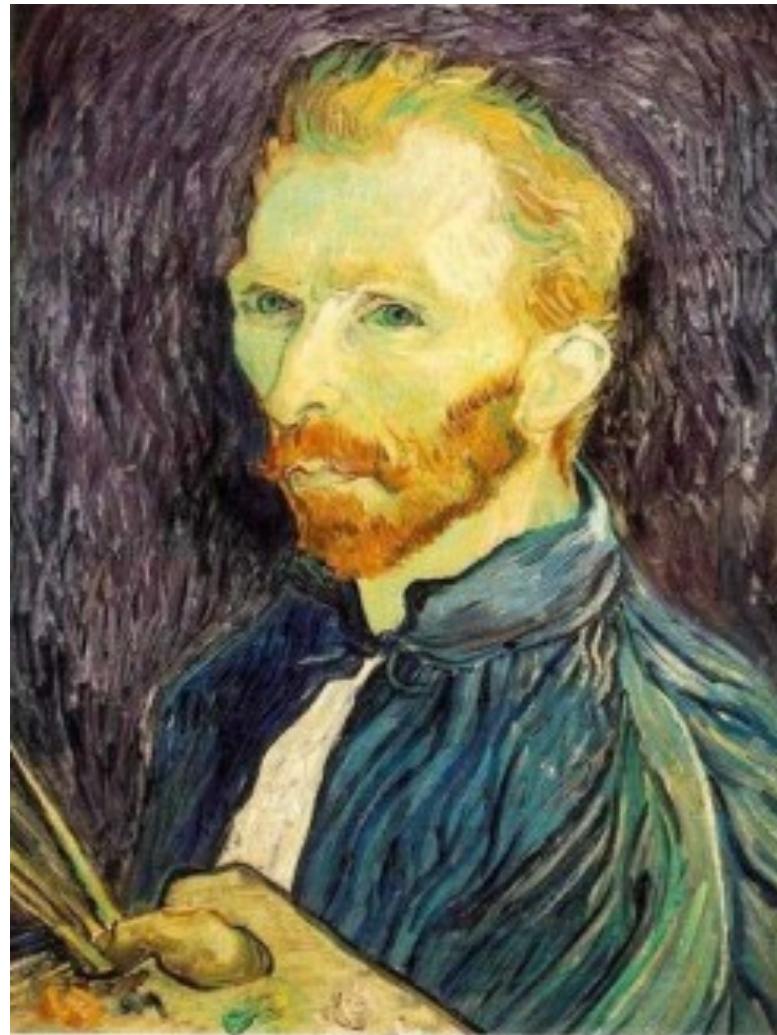
```
bash download.sh afhq-dataset
```

[Update: 2021.07.01] We rebuild the original AFHQ dataset by using high-quality resize filtering (i.e., Lanczos resampling). Please see the clean FID paper that brings attention to the unfortunate software library situation for downsampling. We thank to Alias-Free GAN authors for their suggestion and contribution to the updated AFHQ dataset. If you use the updated dataset, we recommend to cite not only our paper but also their paper.

The differences from the original dataset are as follows:

- We resize the images using Lanczos resampling instead of nearest neighbor downsampling.
- About 2% of the original images had been removed. So the set is now has 15803 images, whereas the original had 16130.
- Images are saved as PNG format to avoid compression artifacts. This makes the files bigger than the original, but it's worth it.

Iterative Gaussian (lowpass) pre-filtering

Gaussian 1/2

G 1/4

G 1/8

filter the image, *then* subsample

- Filter size should double for each ½ size reduction. Why?
- How can we speed this up?

Slide by Steve Seitz
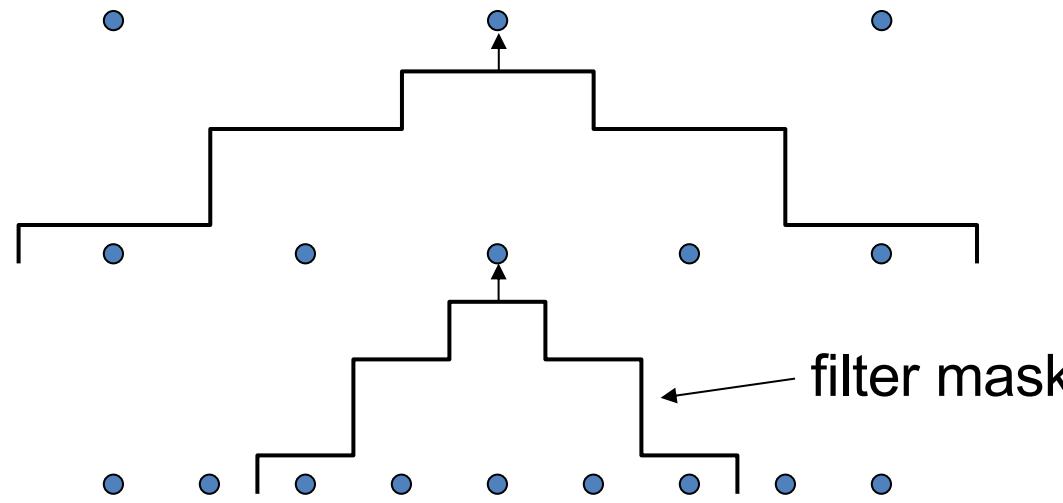
512   256   128   64   32   16   8

A bar in the big images is a hair on the zebra's nose; in smaller images, a stripe; in the smallest, the animal's nose

Gaussian pyramid construction

filter mask

Repeat
- Filter
- Subsample

Until minimum resolution reached
- can specify desired number of levels (e.g., 3-level pyramid)

The whole pyramid is only 4/3 the size of the original image!

What are they good for?

- Improve Search
    - Search over translations
        - Classic coarse-to-fine strategy

        speedup + avoid local minimum (lower resolution -> fewer local minimum)
    - Search over scale
        - E.g., find a face at different scales

Partial derivatives with convolution

- For 2D function f(x,y), the partial derivative is:

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

- For discrete data, we can approximate using finite differences:
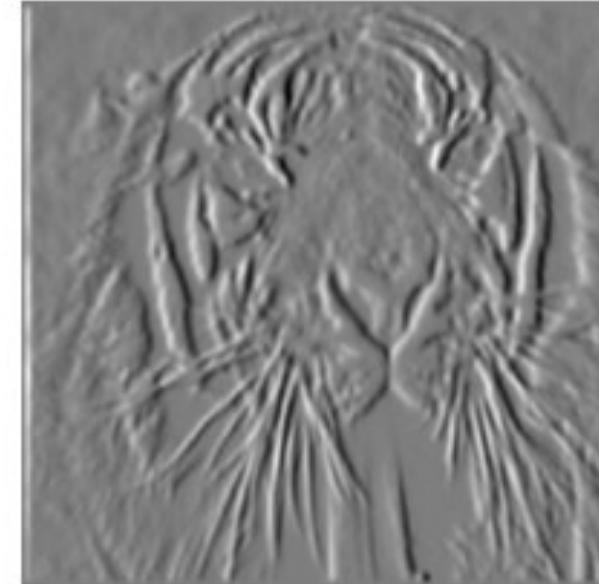
$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

- To implement above as convolution, what would be the associated filter?

Partial derivatives of an image
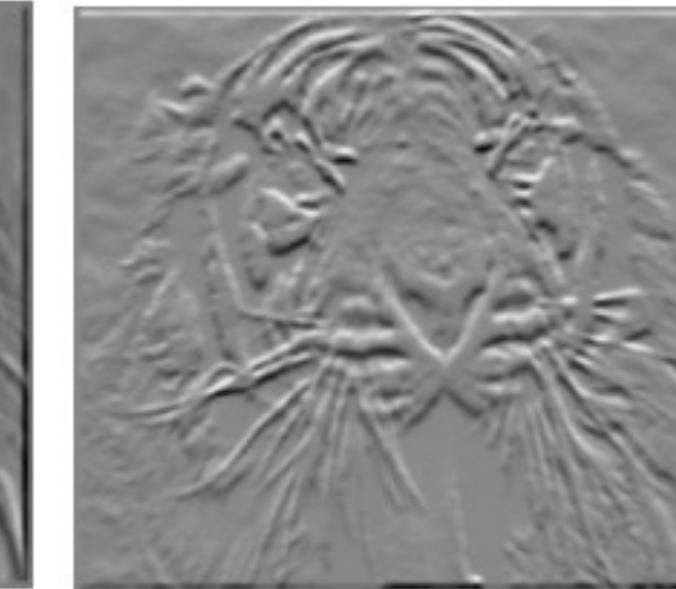
$$\frac{\partial f(x, y)}{\partial x}$$

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

| -1 | 1 |

| -1 | | 1 |
| 1 | or | -1 |

Which shows changes with respect to x?

Finite difference filters

- Other approximations of derivative filters exist:
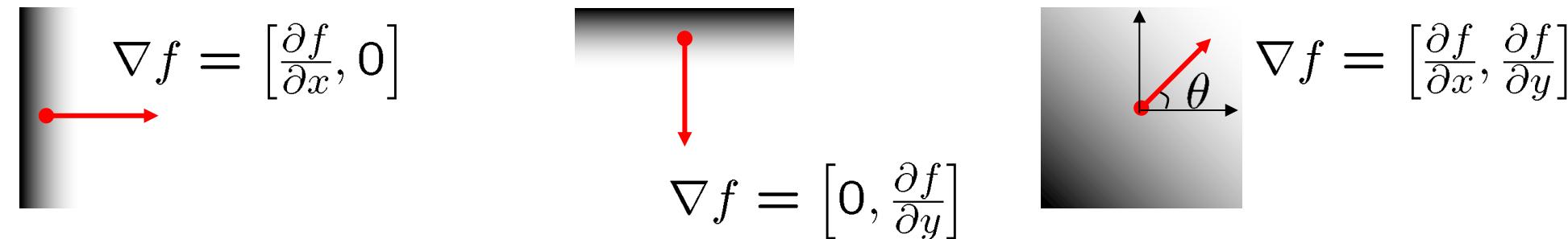
Prewitt: $M_x = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$; $M_y = \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$

Sobel: $M_x = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$; $M_y = \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$

Roberts: $M_x = \begin{array}{|c|c|} \hline 0 & 1 \\ \hline -1 & 0 \\ \hline \end{array}$; $M_y = \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & -1 \\ \hline \end{array}$

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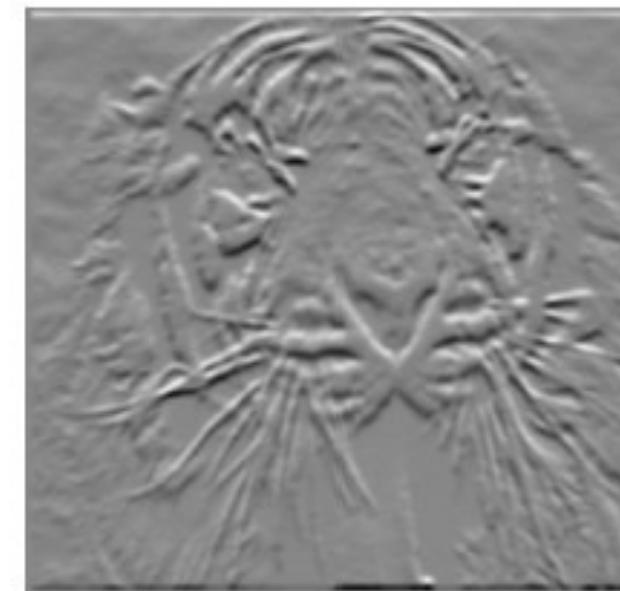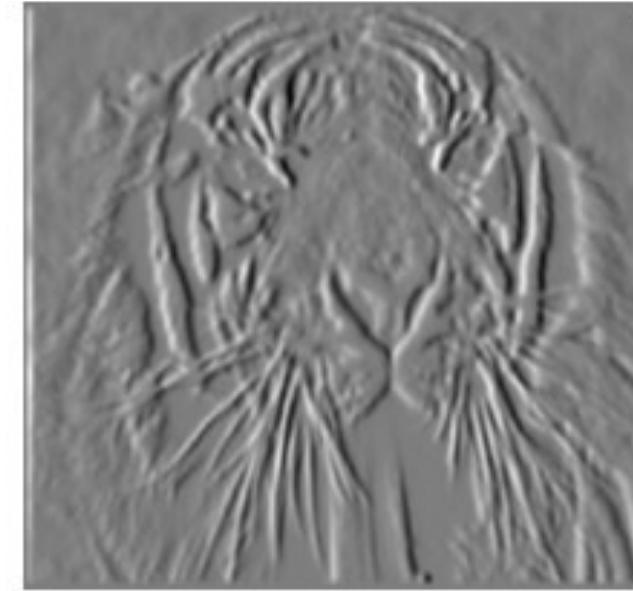
- How does this direction relate to the direction of the edge?

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

The *edge strength* is given by the gradient magnitude

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

Source: Steve Seitz

83

Image Gradient

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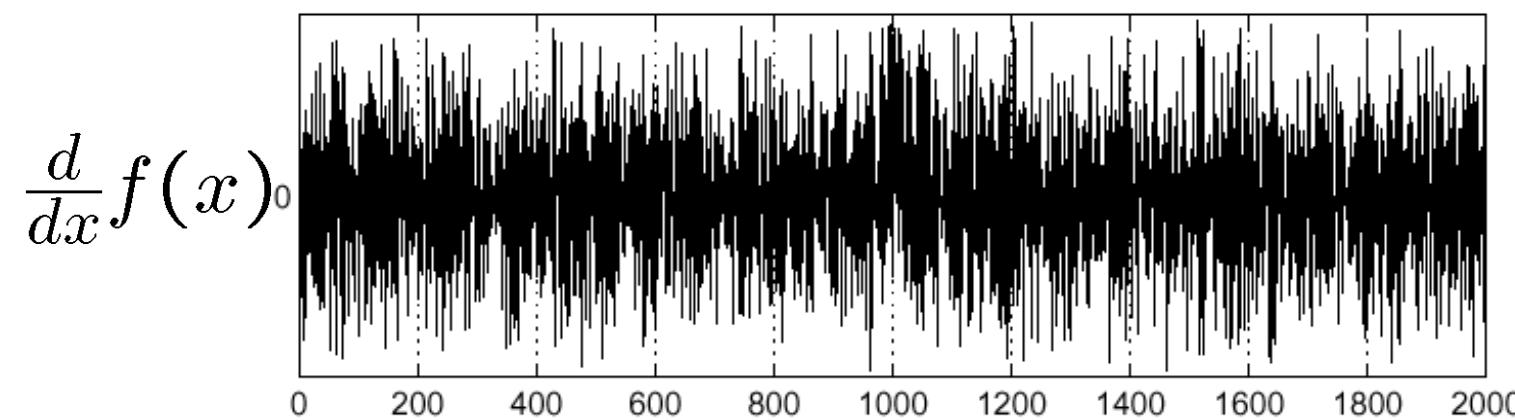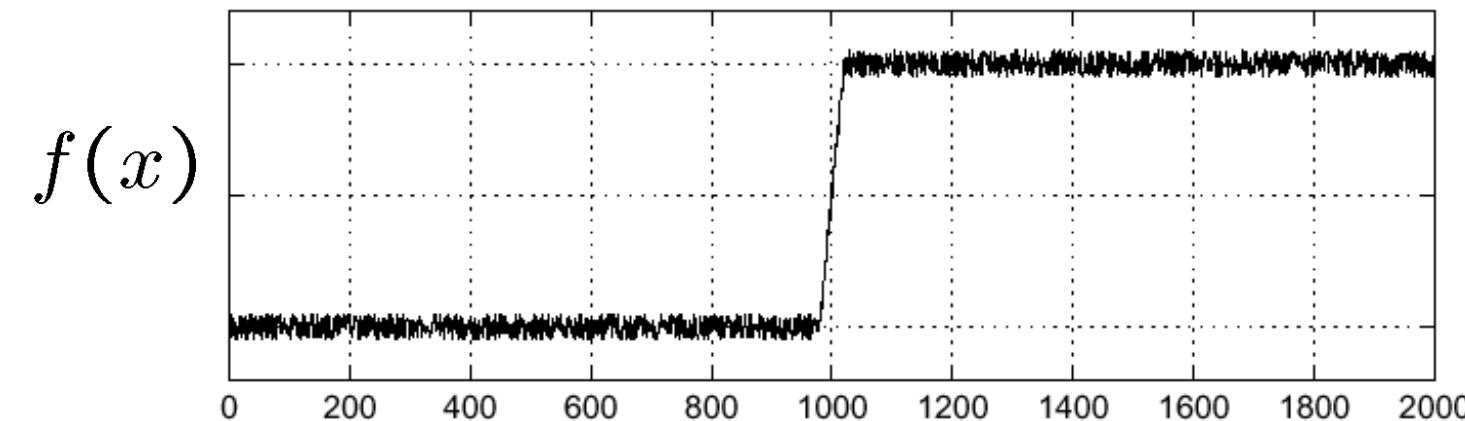
$$\frac{\partial f(x, y)}{\partial y}$$

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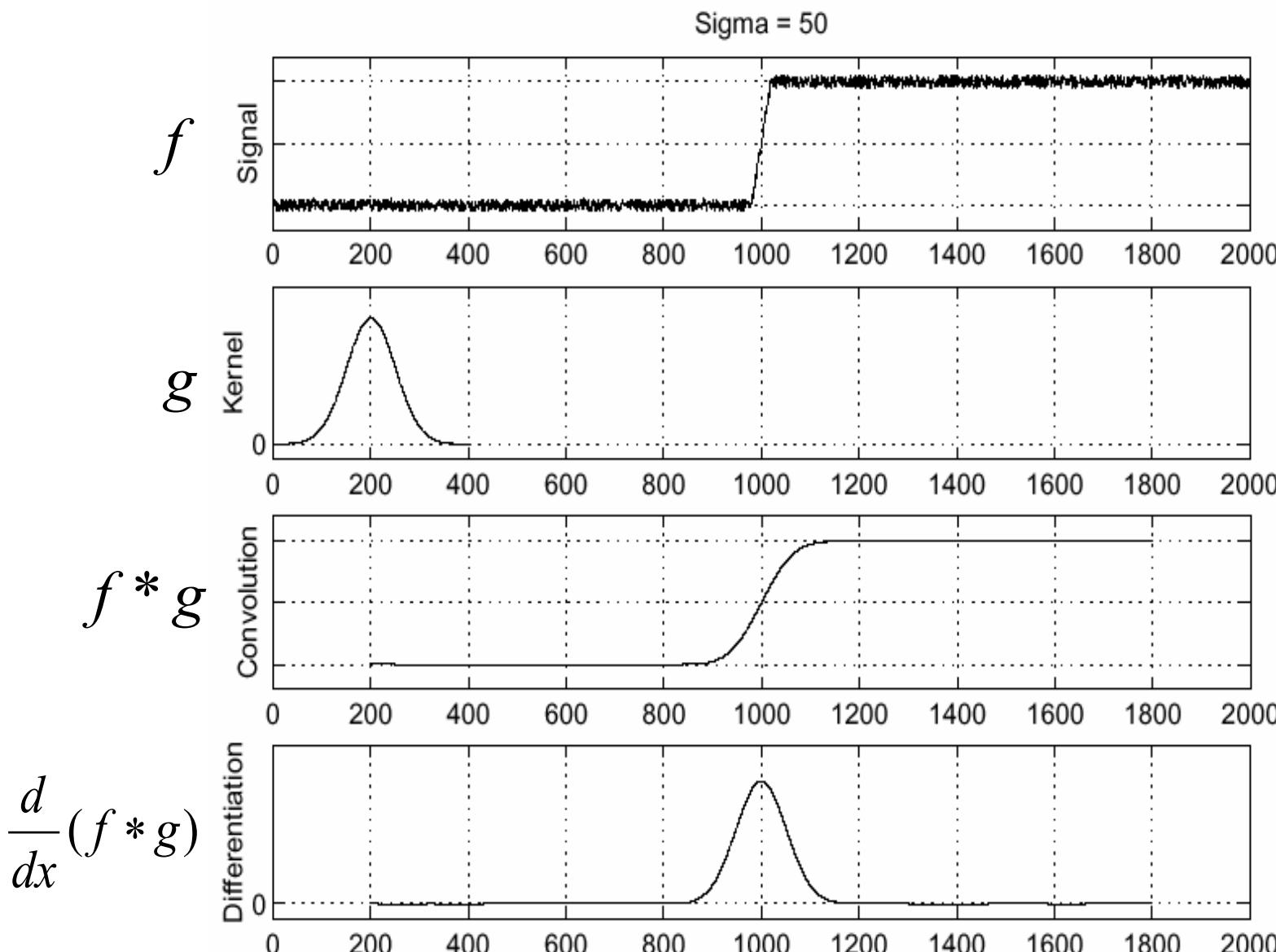
Effects of noise

- 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?
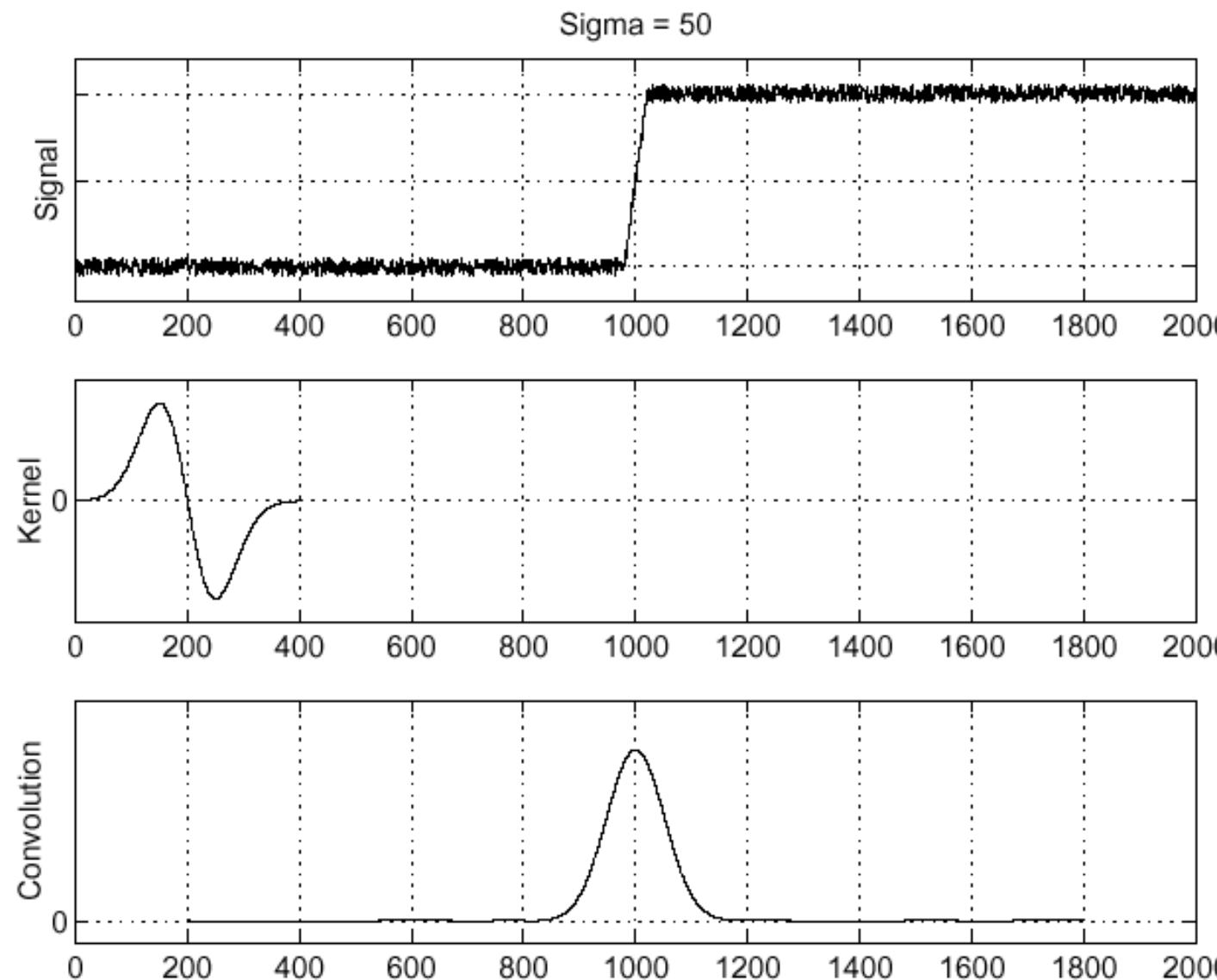
Solution: smooth first

Sigma = 50

f

g

$f * g$
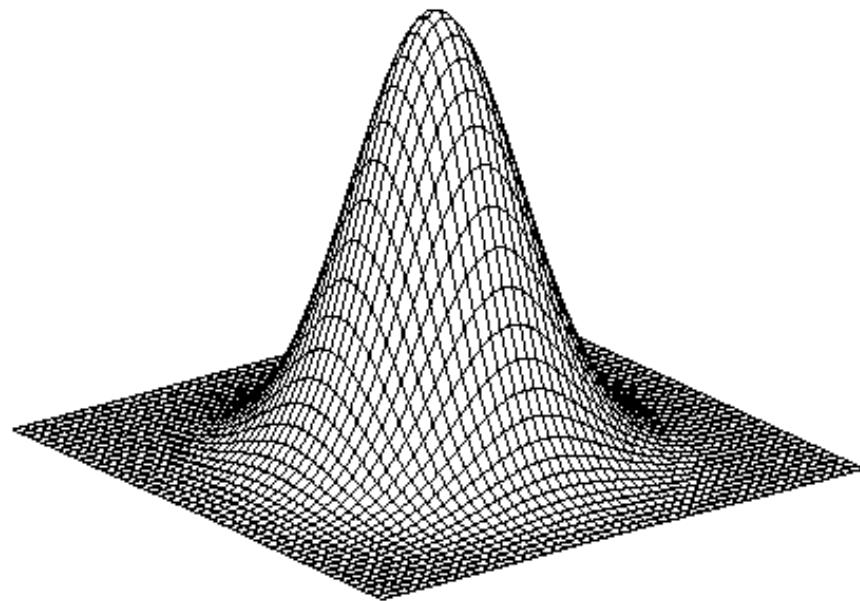
$\dfrac{d}{dx}(f * g)$

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

86

Derivative theorem of convolution
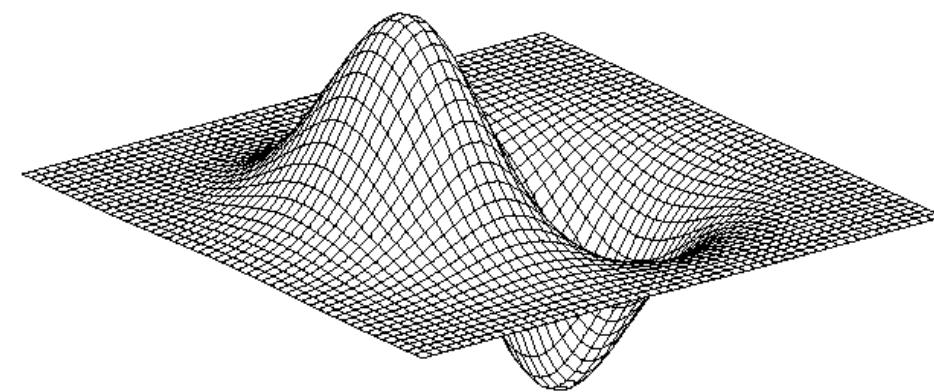
- This saves us one operation: $\quad \frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$

Sigma = 50

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

$(\frac{\partial}{\partial x}h) \star f$
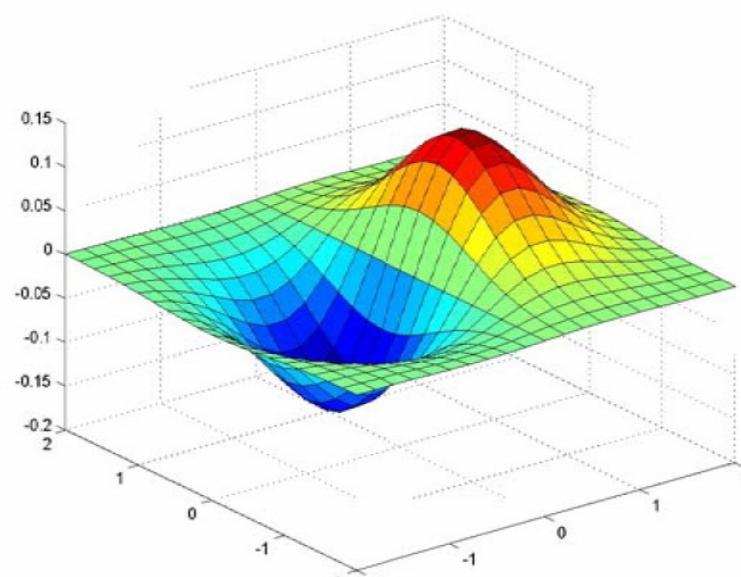
Derivative of Gaussian filter
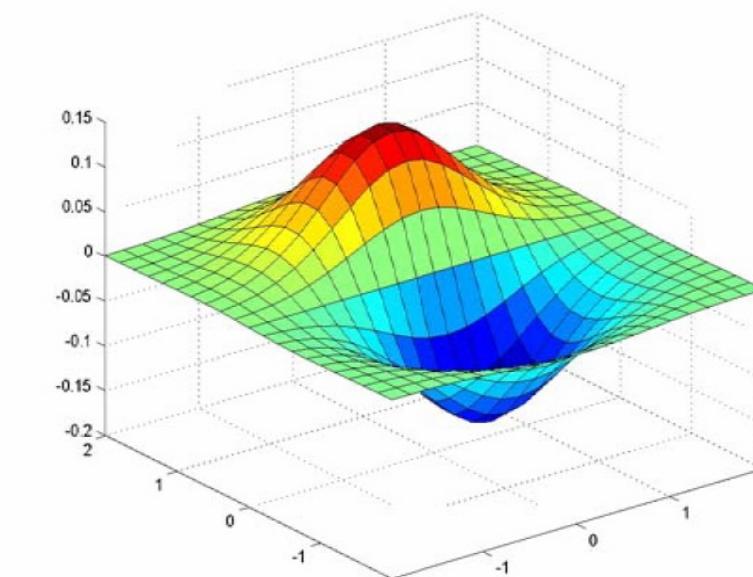

* [1 -1]   =
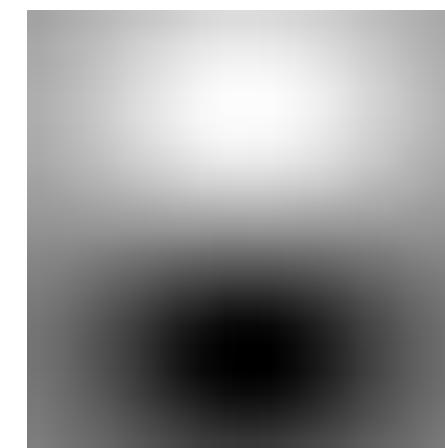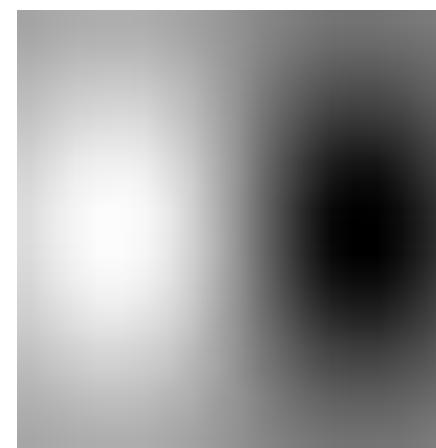
Derivative of Gaussian filter


x-direction
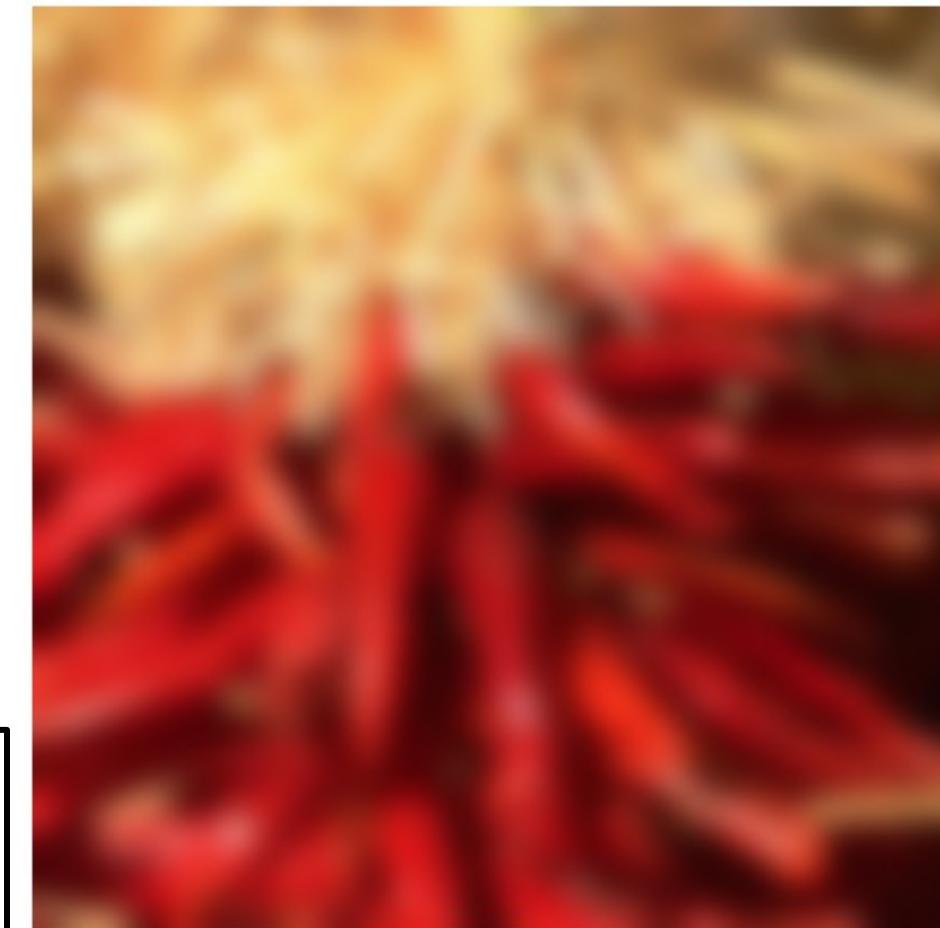
y-direction

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
    - copy edge
    - reflect across edge

Boundary artifacts in deep image synthesis
Try different padding choices.
Try reflected padding first.

Source: S. Marschner

Thank You!

16-726, Spring 2023
https://learning-image-synthesis.github.io/sp23/