Lecture 3: Image pyramids

Announcements

- Reminder: PS1 due next Weds.
- Section this week:
  - Complex numbers and frequencies
  - Fourier transform
- Suggested reading: Szeliski chapter or Torralba, Isola, Freeman chapter

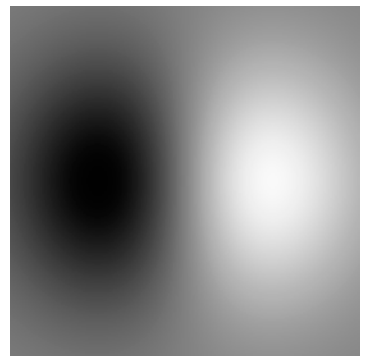
Today

- Image pyramids
- Image statistics
- Texture synthesis

Last class: linear filtering

Blurring

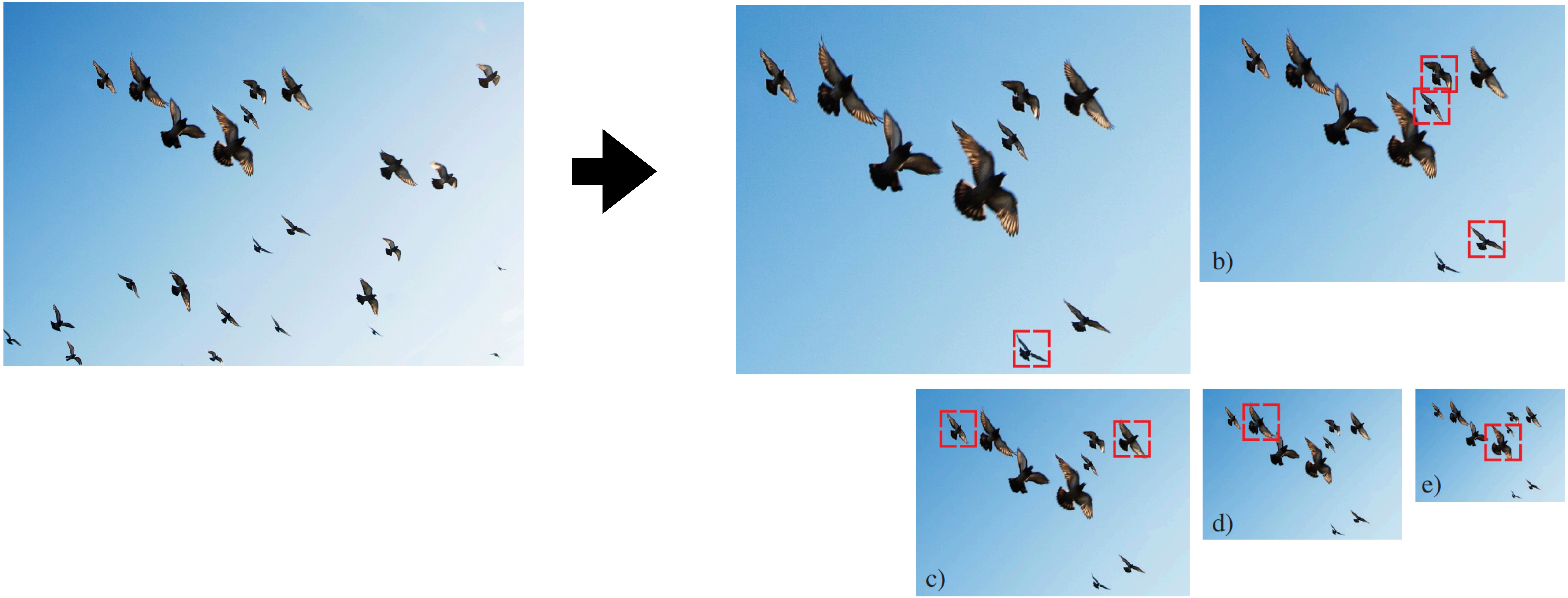
 

Derivative filters

We want scale invariance!

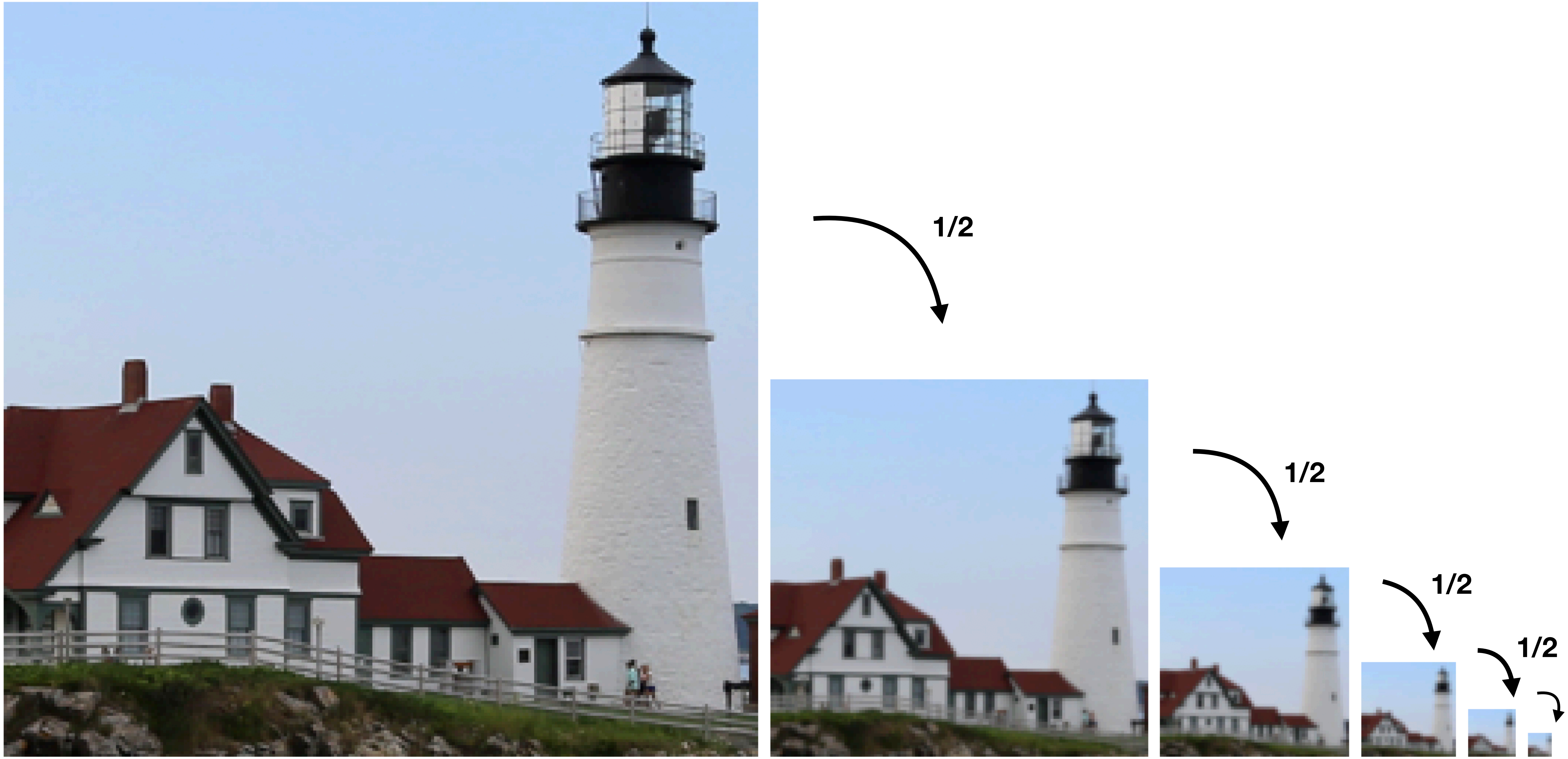
Source: Torralba, Freeman, Isola

Image pyramids



Source: Torralba, Freeman, Isola

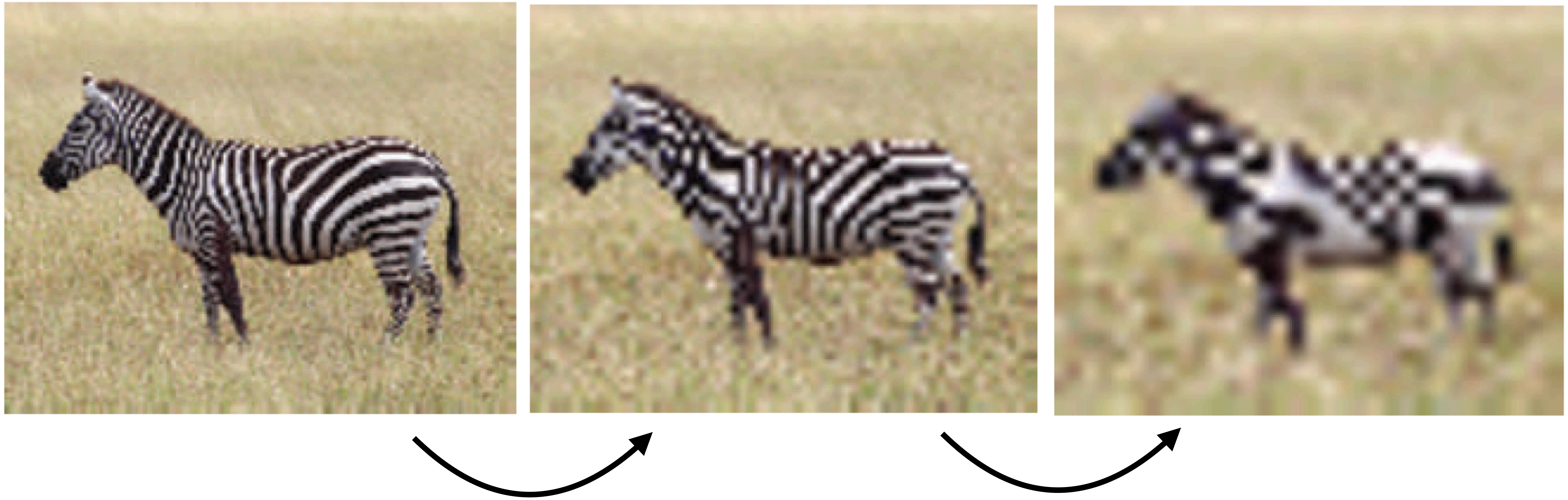
Image pyramid



Source: Torralba, Freeman, Isola

Subsampling and aliasing

103×128 52×64 26×32



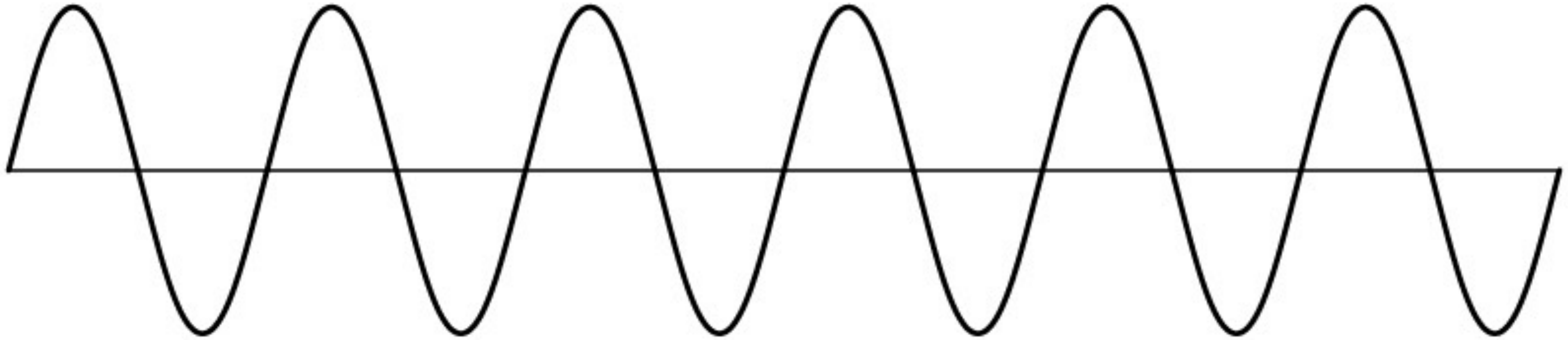
1/2 1/2

Idea #1: Throw away every other pixel.

Source: Torralba, Freeman, Isola

What's happening?

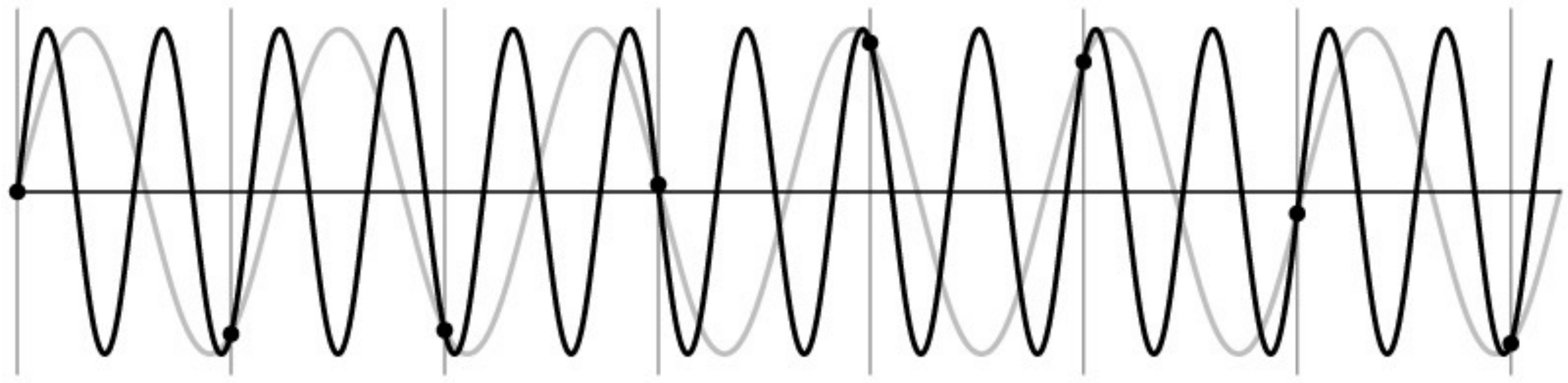
Consider a sinusoid:



Source: S. Marschner

Undersampling

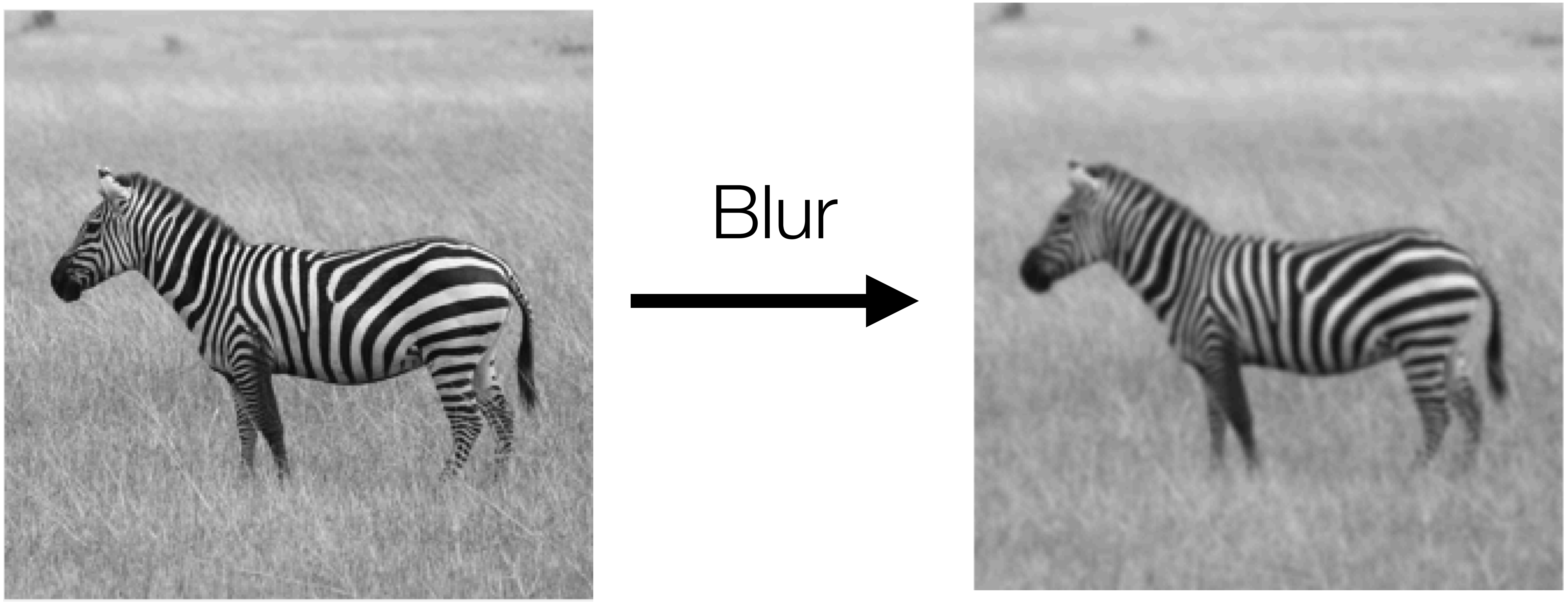
- What if we "missed" things between the samples?
- As expected, information is lost
- Unexpectedly: indistinguishable from low-frequency sinusoid!
- Also indistinguishable from higher frequencies
- **Aliasing:** signals "traveling in disguise" as other frequencies



Source: S. Marschner

Removing aliasing

- Remove the high frequencies first!
- Blur the image before downsampling
  - Next class we'll see *why* blurring does this

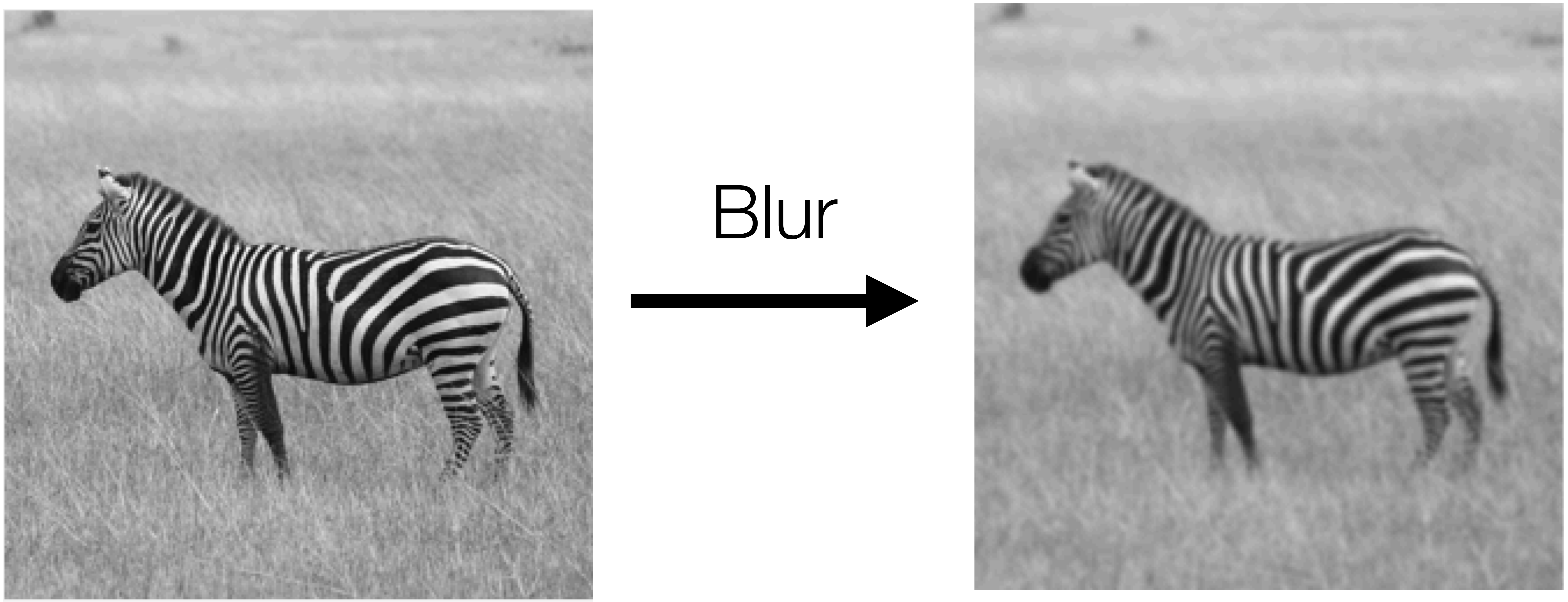


Source: Torralba, Freeman, Isola

Gaussian pyramid

For each level:

1. Blur input image with a Gaussian (or binomial) filter

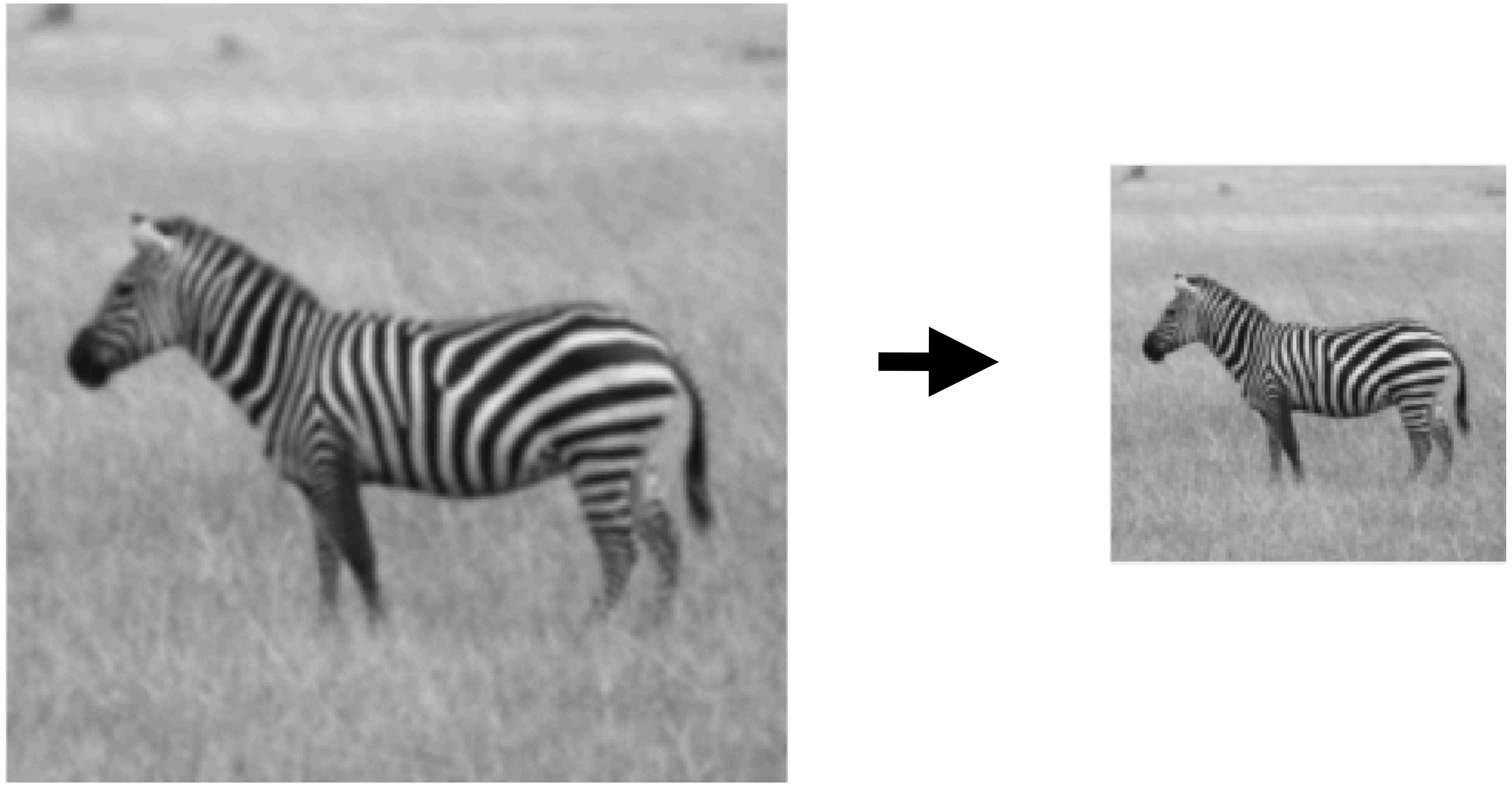


Source: Torralba, Freeman, Isola

Gaussian pyramid

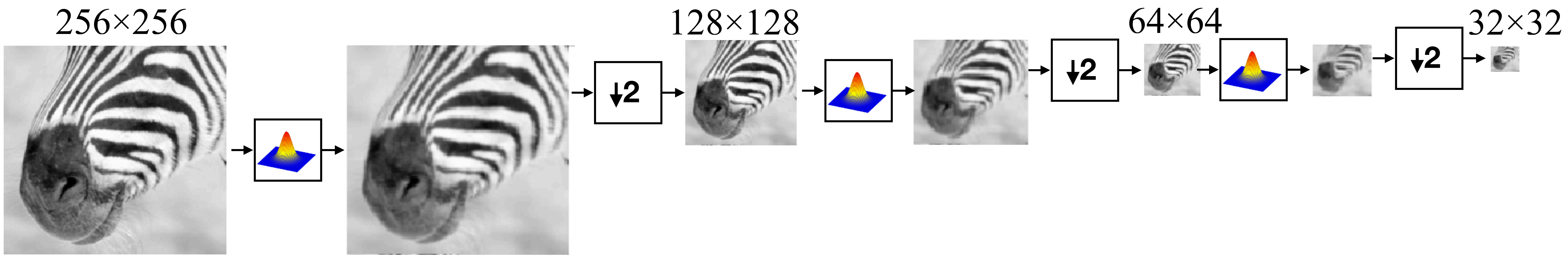
For each level:

1. Blur input image with a Gaussian (or binomial) filter
2. Downsample (throw away every other pixel)



Source: Torralba, Freeman, Isola

Gaussian pyramid



Source: Torralba, Freeman, Isola

Gaussian pyramid

512×512

(original image)

256×256

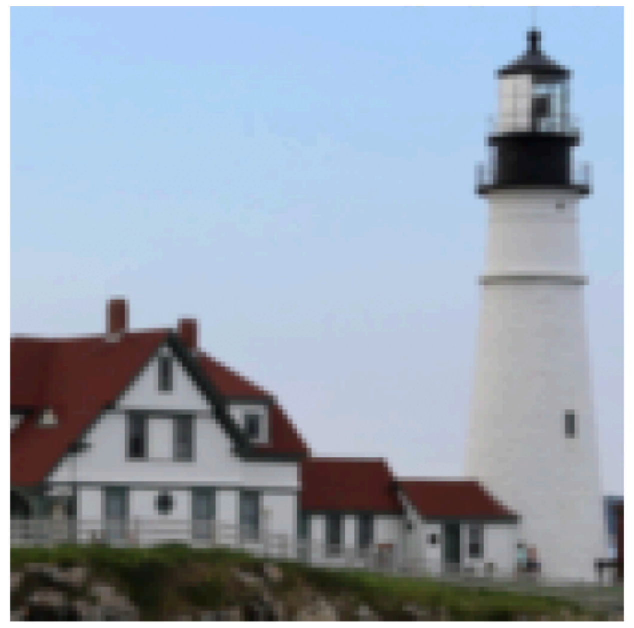
128×128

64×64

32×32

Source: Torralba, Freeman, Isola. Image from Forsyth & Ponce

Gaussian pyramid



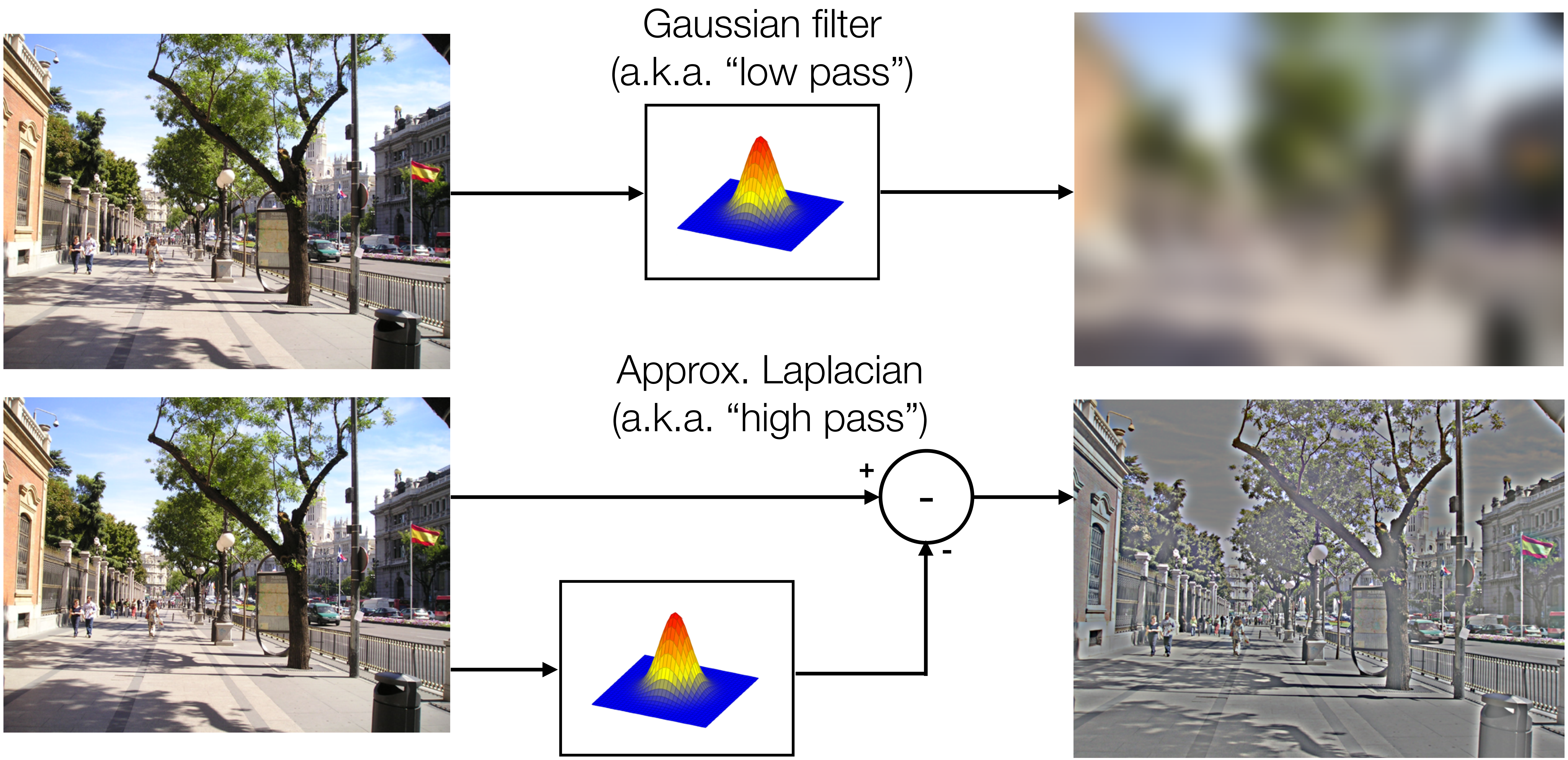
$$g_k \longrightarrow \boxed{G_k} \longrightarrow g_{k+1}$$

For each level

1. Blur input image with a Gaussian filter
2. Downsample image

Source: Torralba, Freeman, Isola

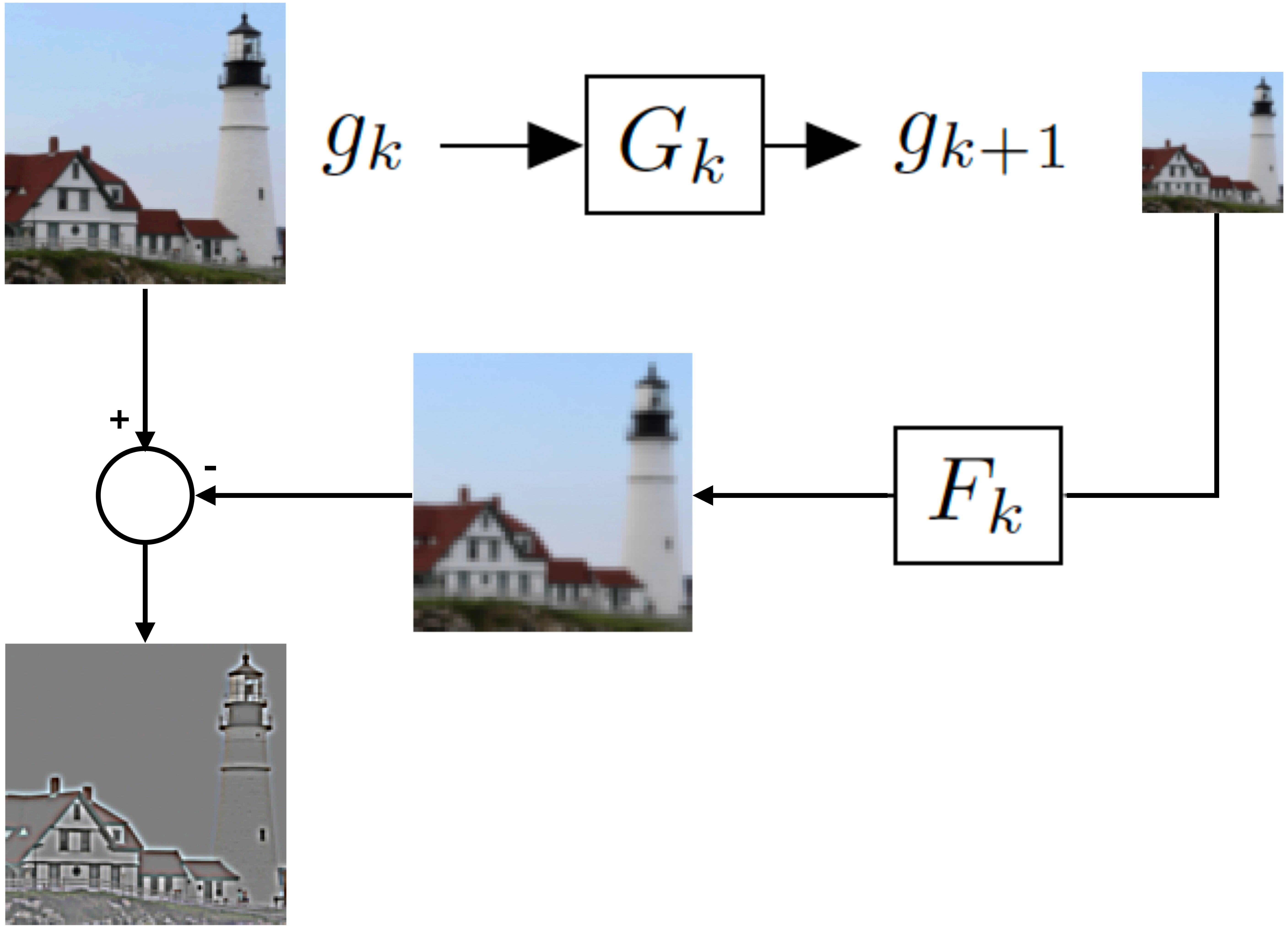
Recall: Laplacian



Source: Torralba, Freeman, Isola

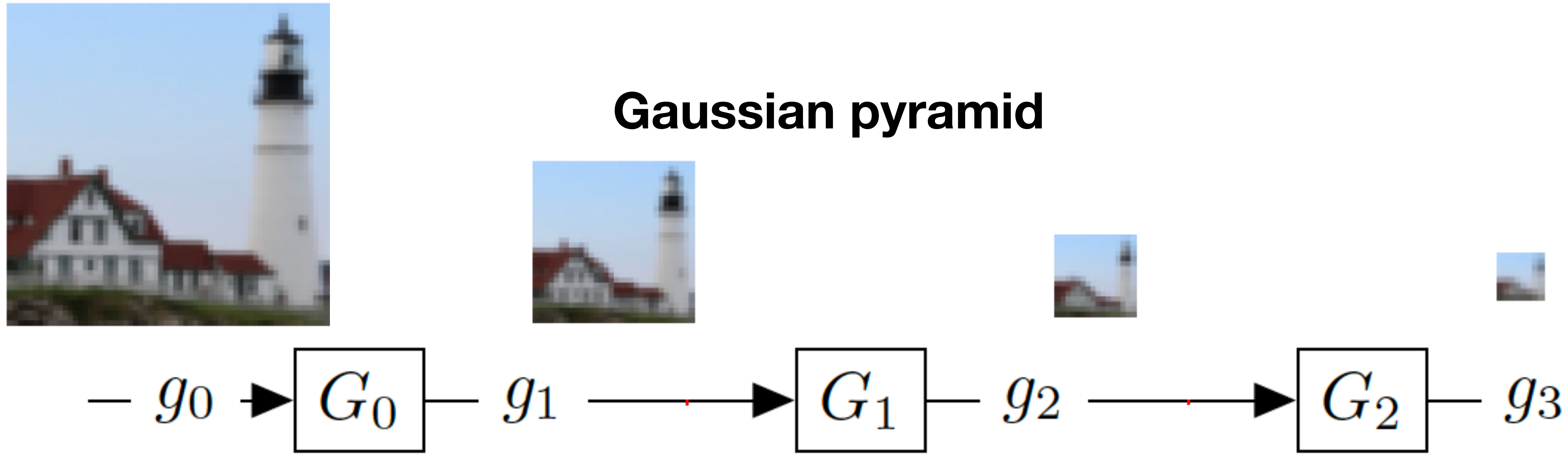
Laplacian Pyramid

Compute the difference between upsampled Gaussian pyramid level k+1 and Gaussian pyramid level k. Recall that this approximates the blurred Laplacian.



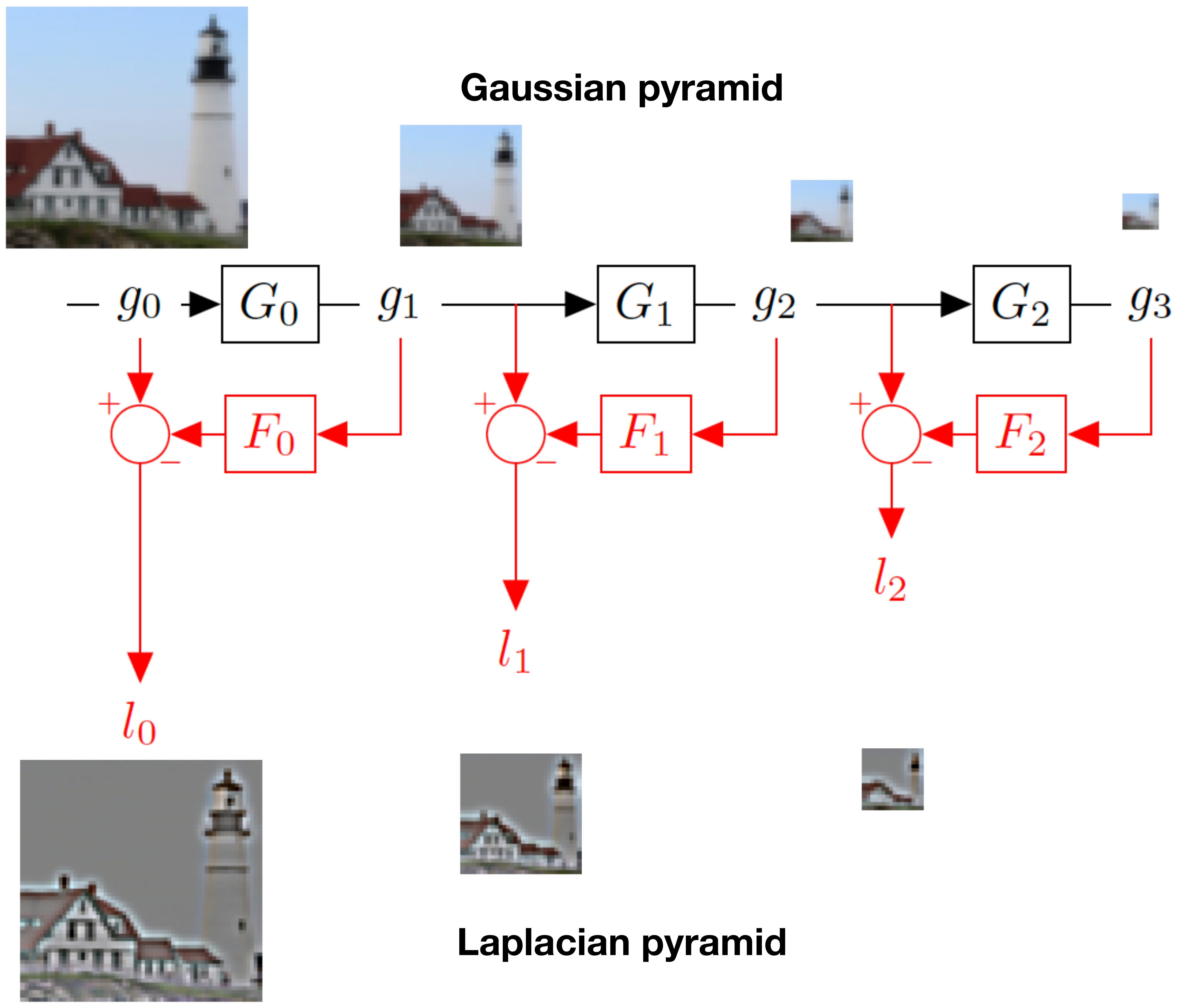
Source: Torralba, Freeman, Isola

Laplacian Pyramid



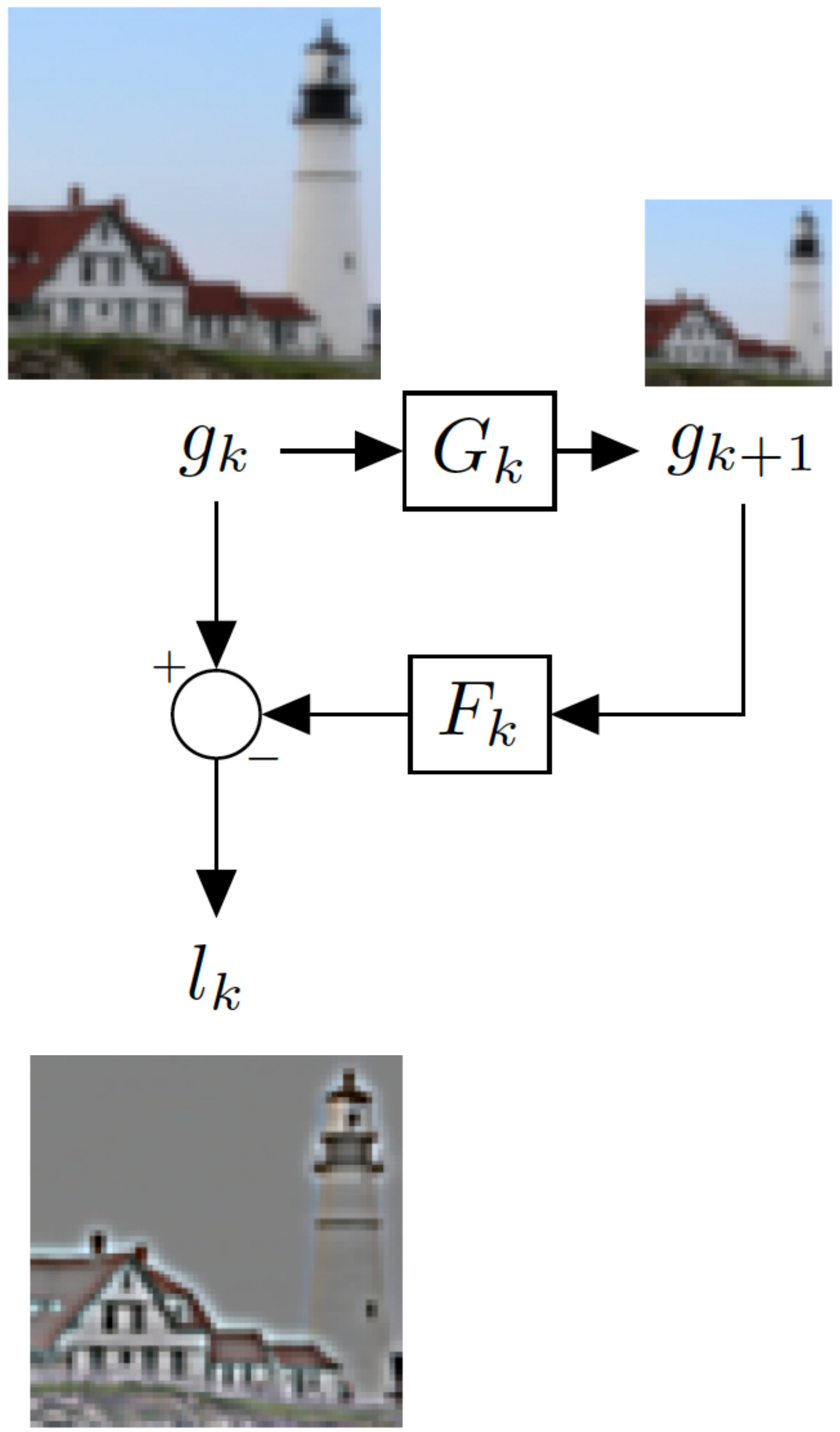
Source: Torralba, Freeman, Isola

Laplacian Pyramid



Source: Torralba, Freeman, Isola

Laplacian Pyramid



Blurring and downsampling:

$$g_{k+1} = G_k(g_k) = \text{downsample}(\text{blur}(g_k))$$

Upsampling, blurring, and subtraction:

$$l_k = F_k(g_k, g_{k+1}) = ????$$

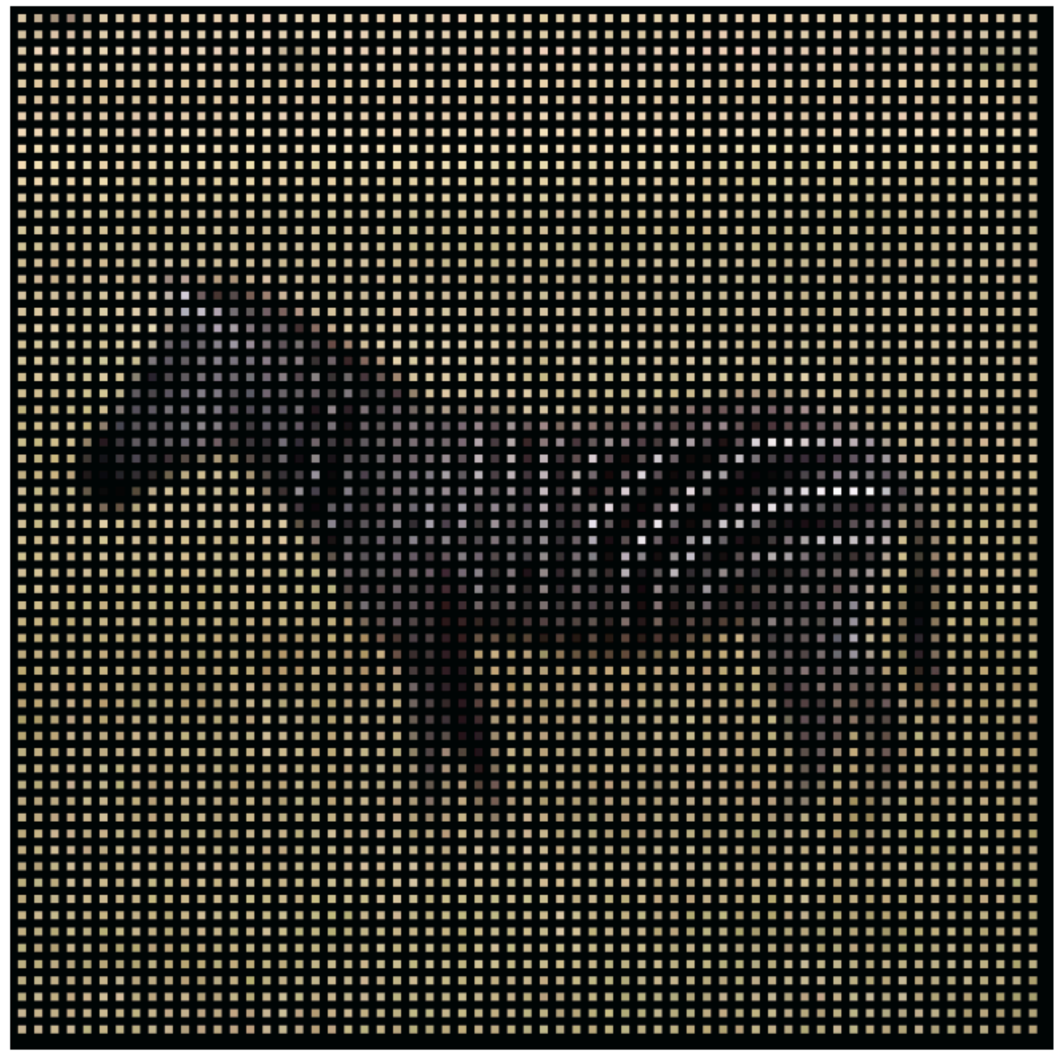
Source: Torralba, Freeman, Isola

Upsampling

64x64

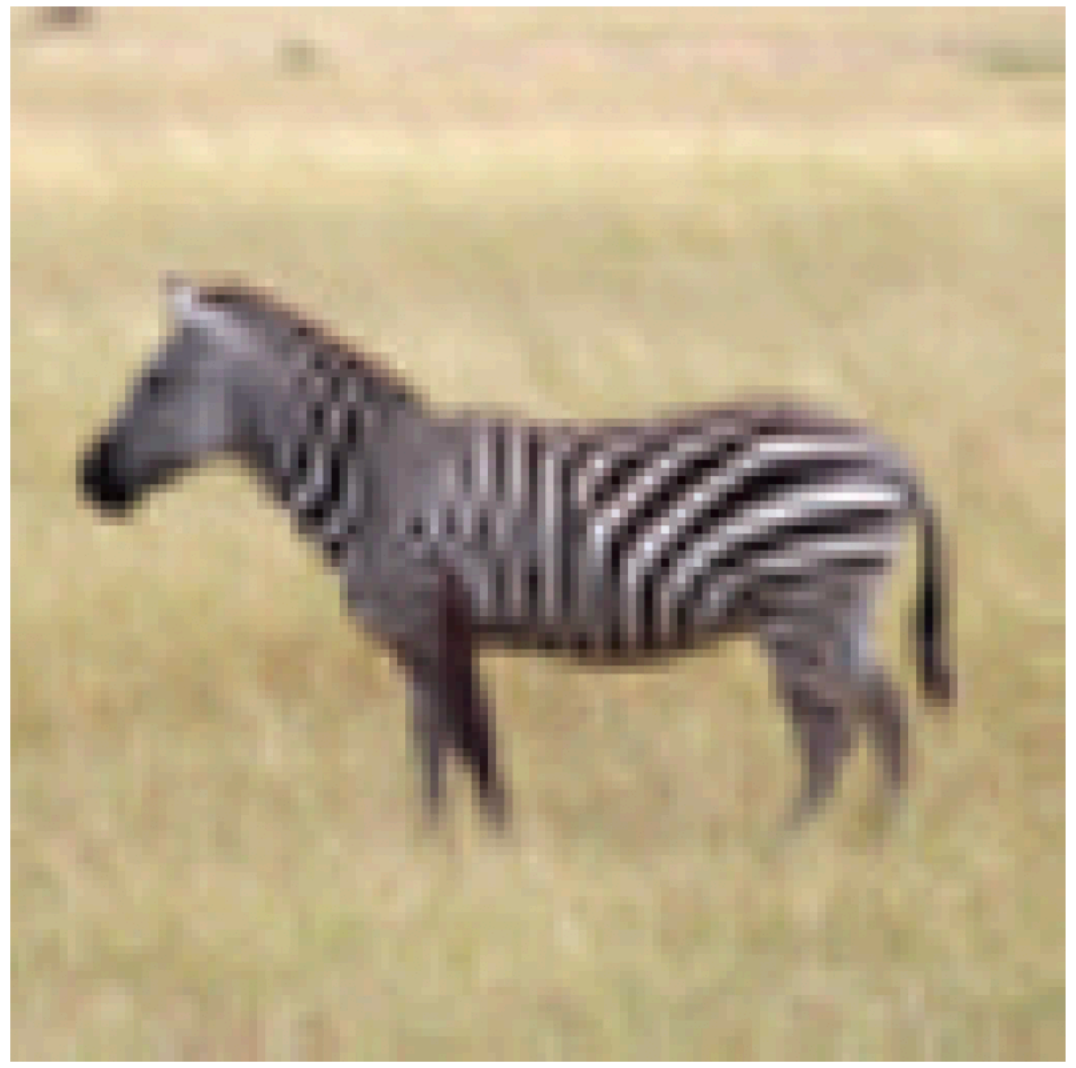
Insert zeros

128x128



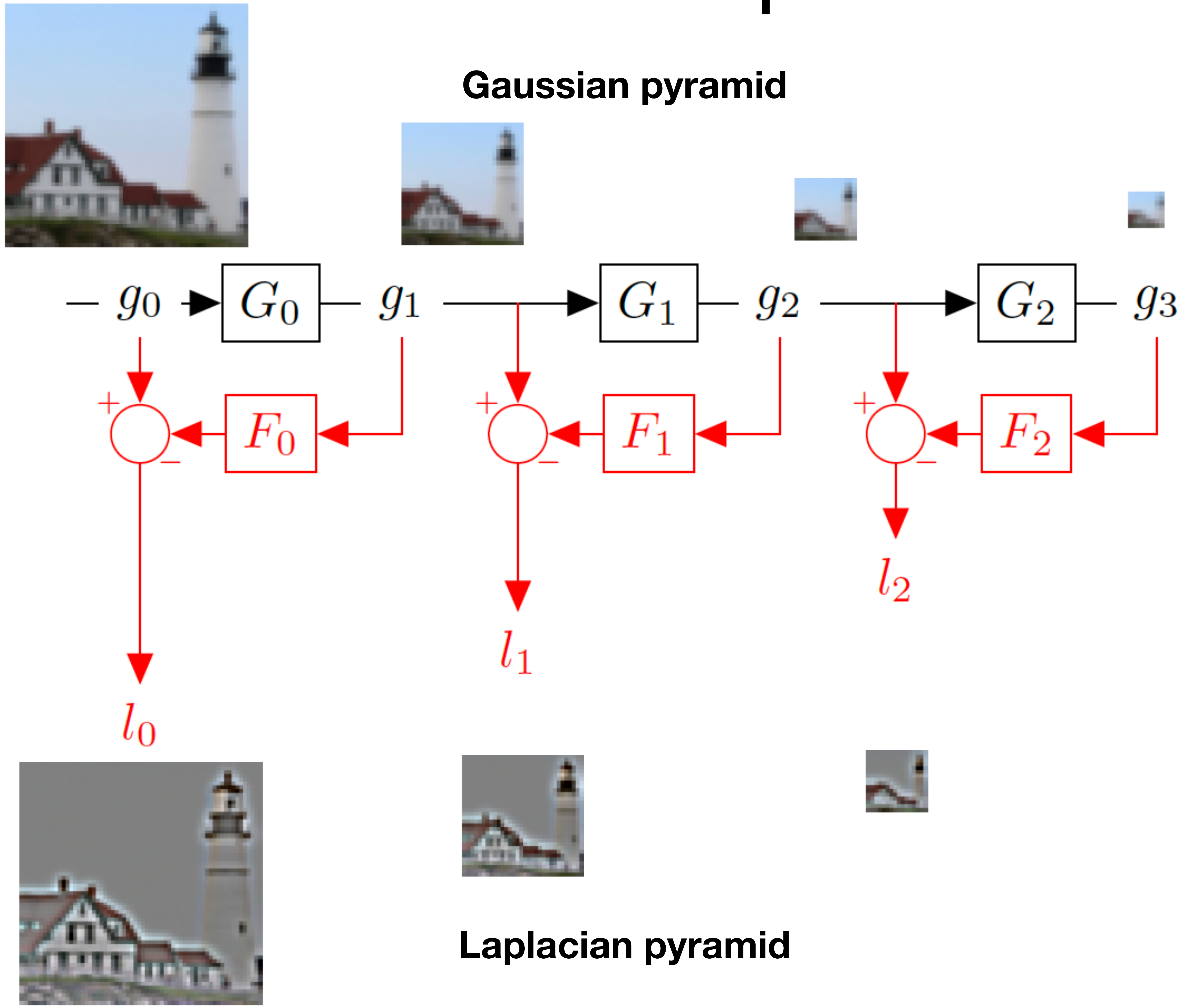
$$\circ \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0.5 & 1 & 0.5 \\ 0.25 & 0.5 & 0.25 \end{bmatrix} =$$

128x128



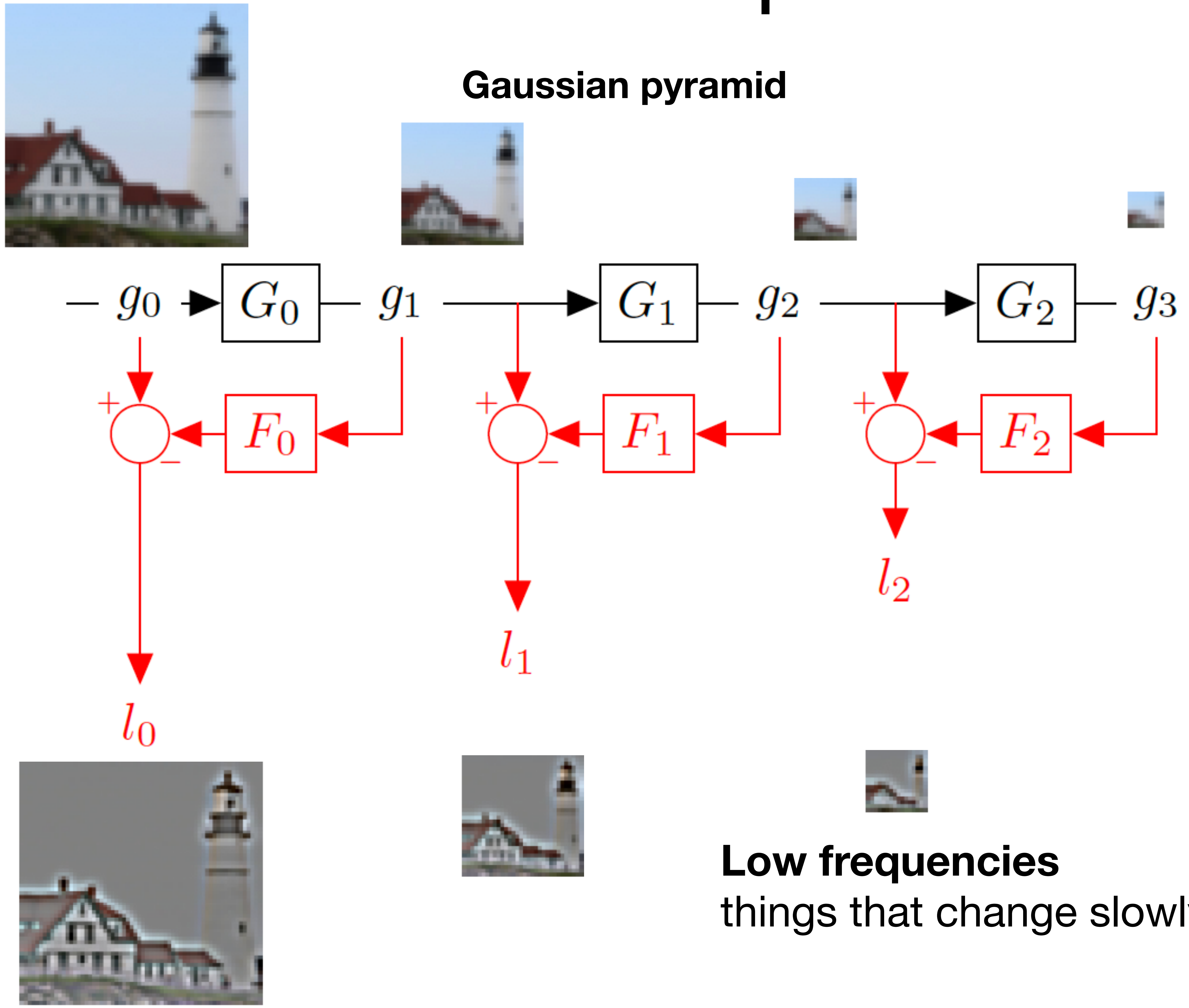
Source: Torralba, Freeman, Isola

Laplacian Pyramid



Source: Torralba, Freeman, Isola

Laplacian Pyramid



Low frequencies
things that change slowly

High frequencies
things that change fast

Laplacian Pyramid



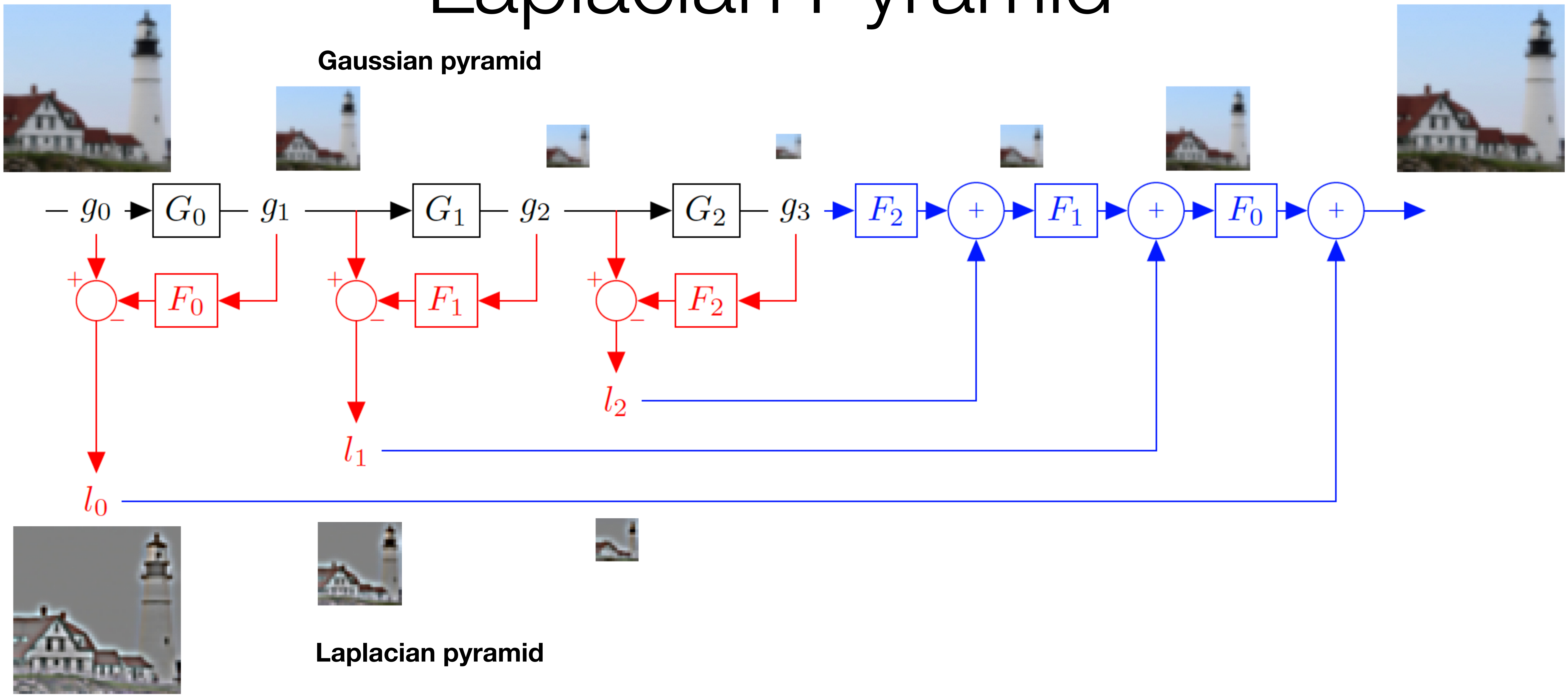

Gaussian residual

Laplacian pyramid

Can we invert the Laplacian Pyramid?

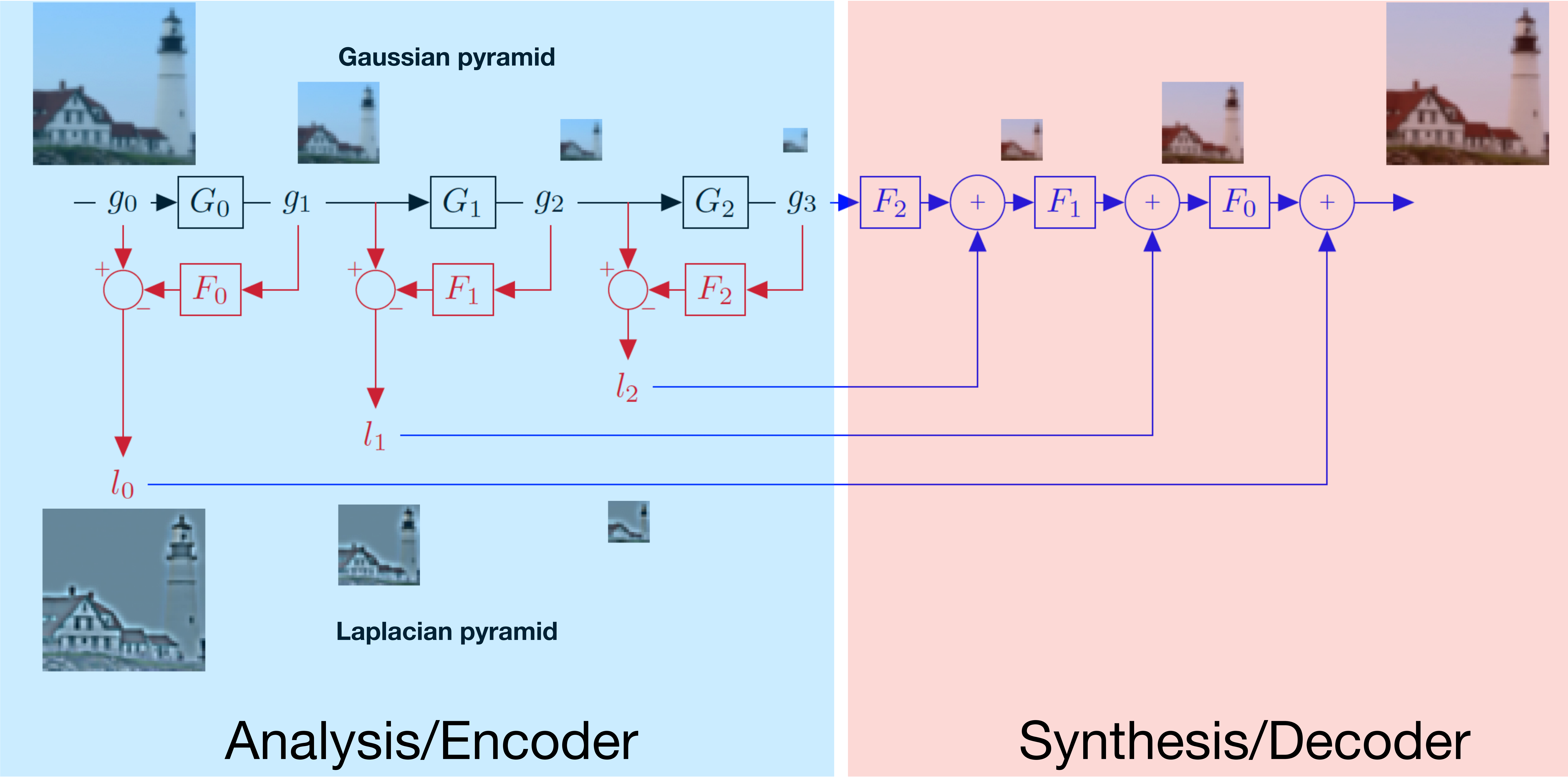
Source: Torralba, Freeman, Isola

Laplacian Pyramid



Source: Torralba, Freeman, Isola

Laplacian Pyramid



Source: Torralba, Freeman, Isola

Laplacian pyramid applications

- Texture synthesis
- Image compression
- Noise removal
- Computing image “keypoints”

Source: Torralba, Freeman, Isola

Image Blending

Source: Torralba, Freeman, Isola

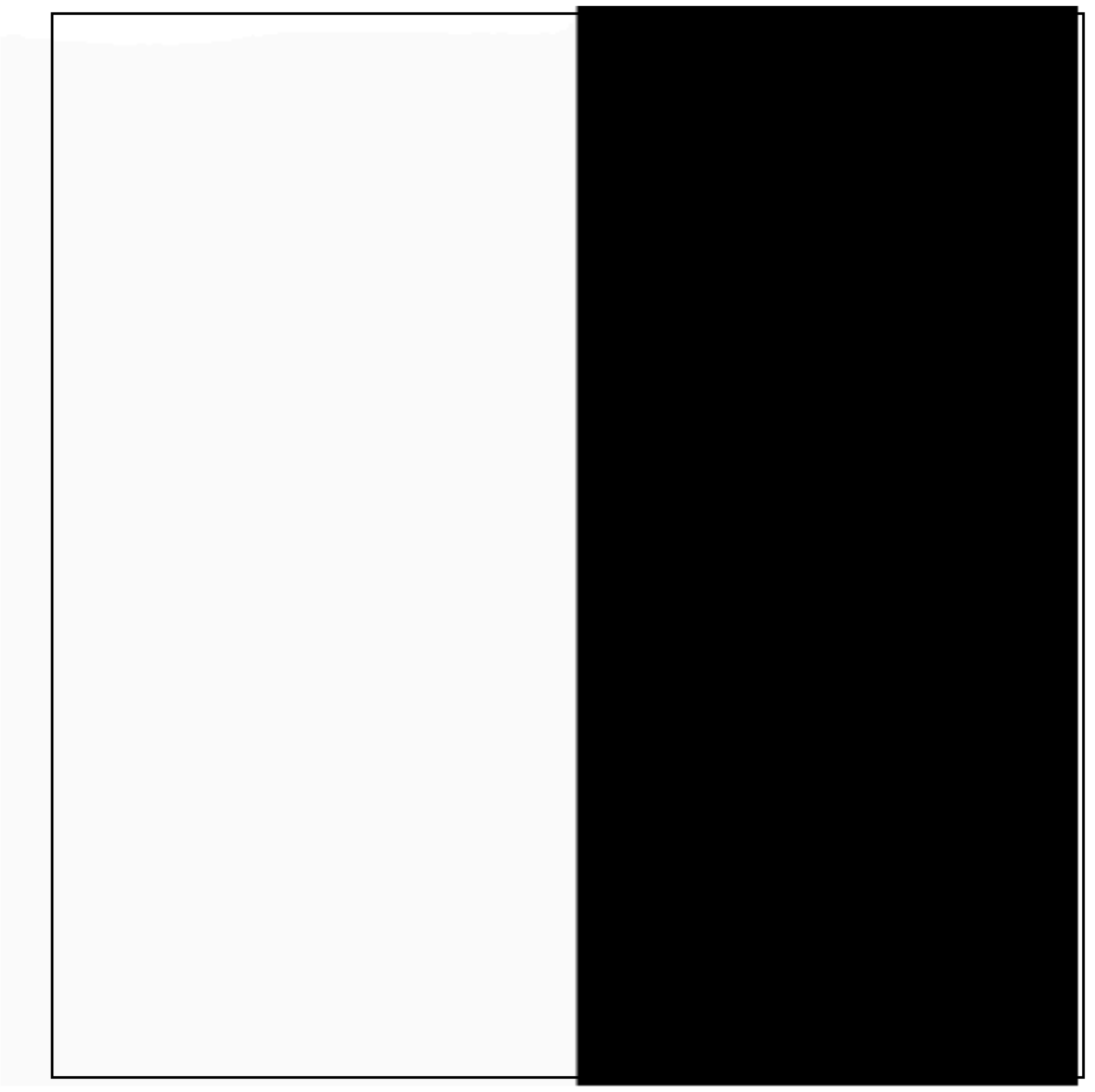
Image Blending

Source: Torralba, Freeman, Isola

Image Blending


I^A


I^B


m


I

$$I = m * I^A + (1 - m) * I^B$$

Source: Torralba, Freeman, Isola

Image Blending with the Laplacian Pyramid

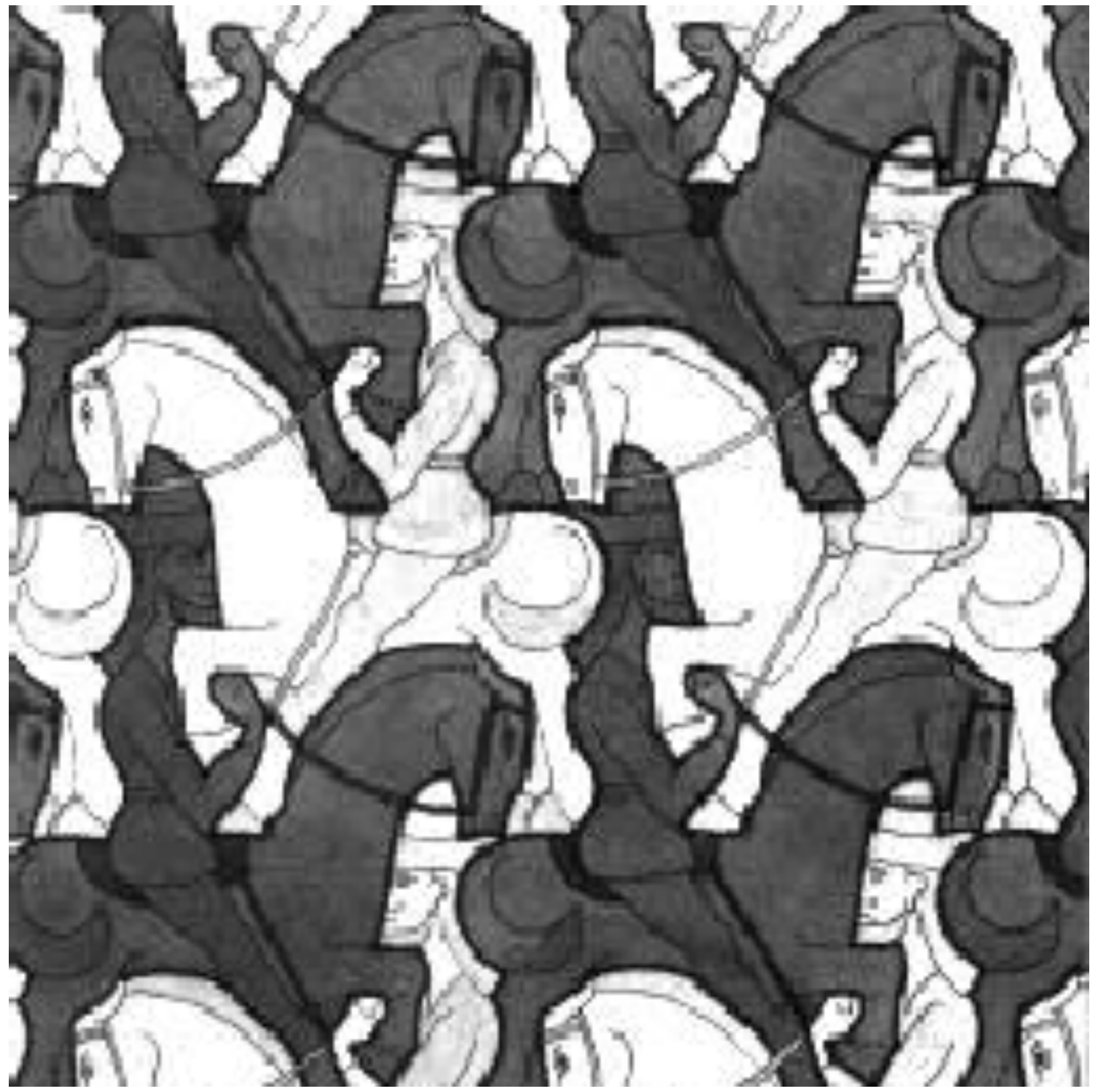
$$l_k = l_k^A * m_k + l_k^B * (1 - m_k)$$

Source: Torralba, Freeman, Isola

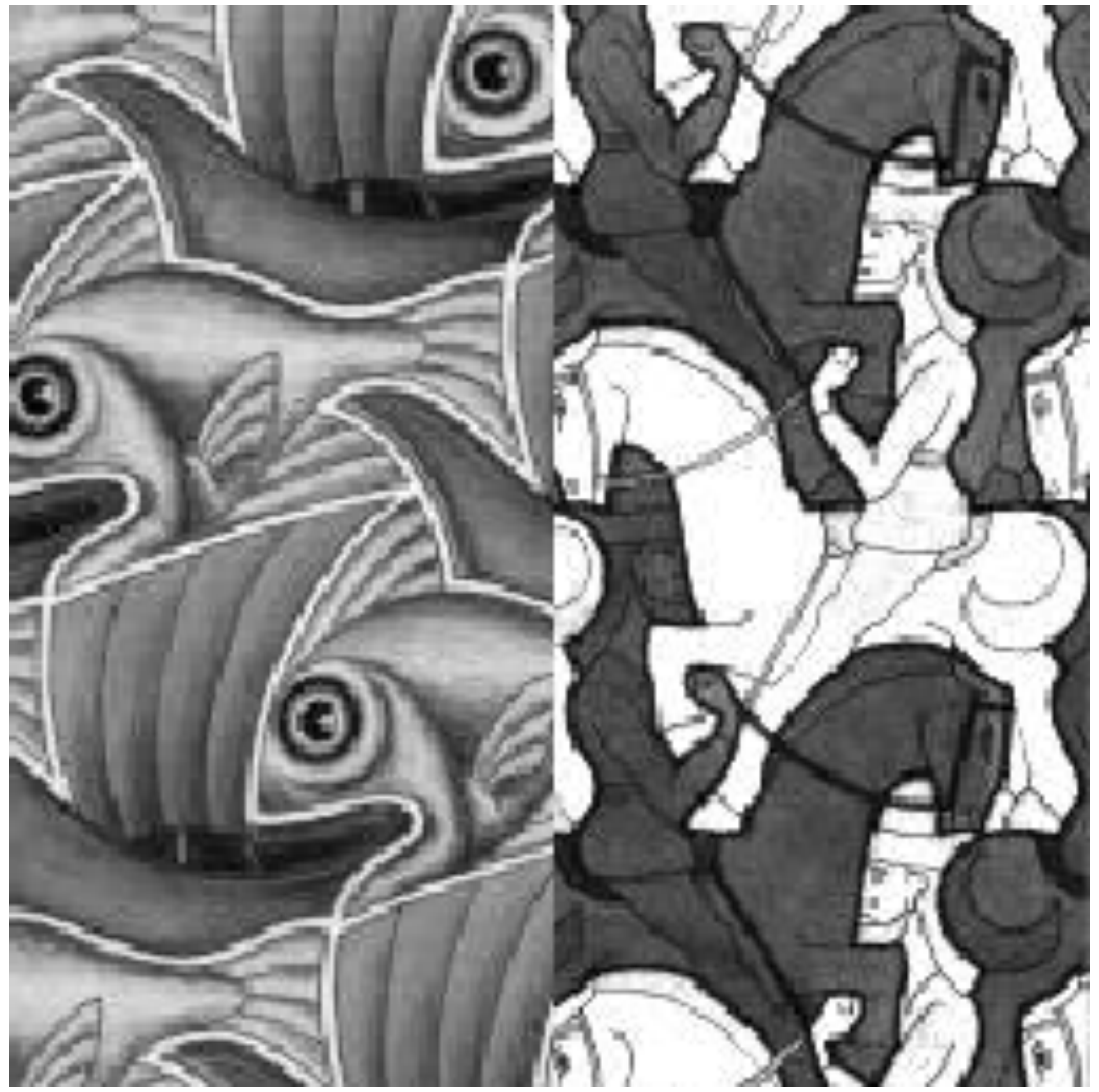
Image Blending with the Laplacian Pyramid

Source: Torralba, Freeman, Isola

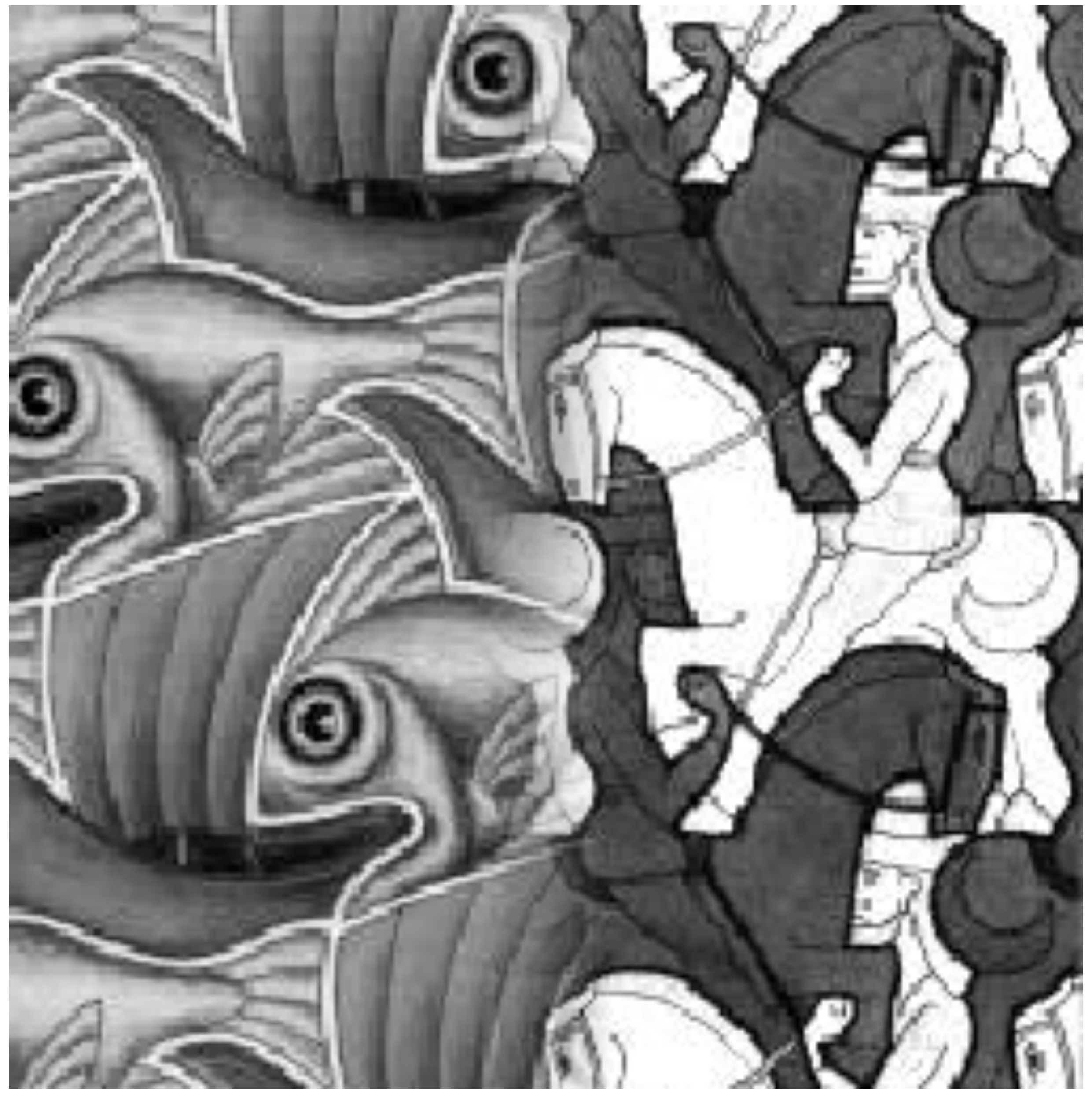
+



=



Simple blend



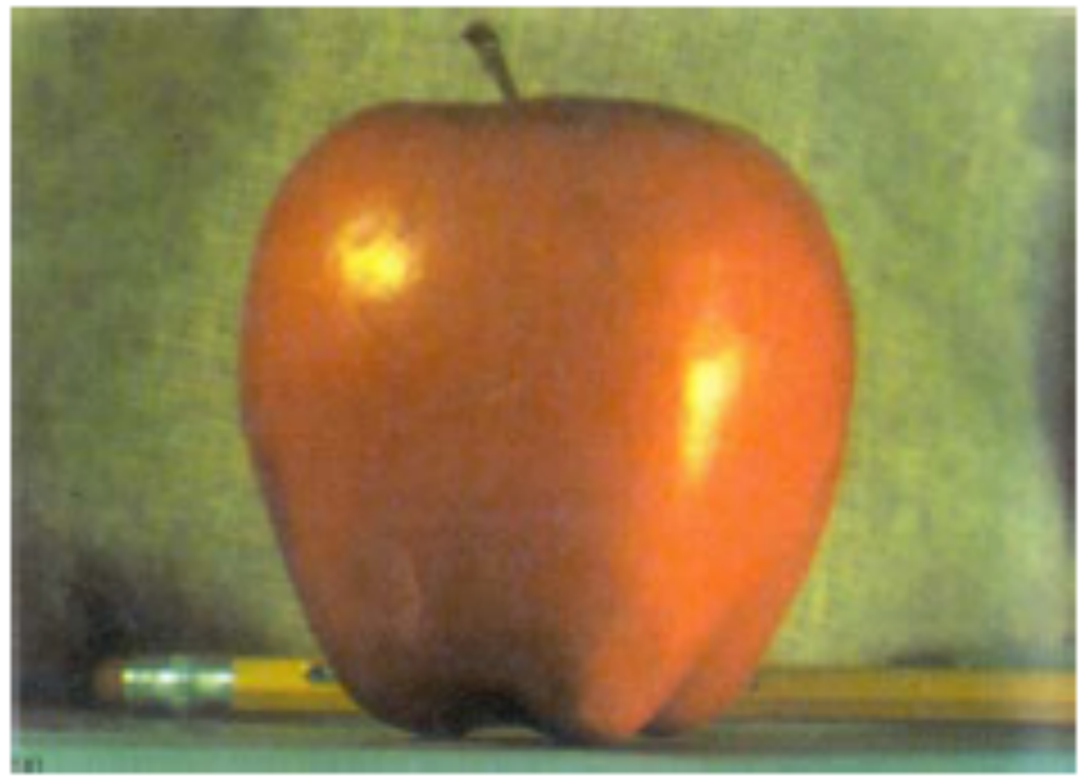
With Laplacian pyr.

Source: A. Efros

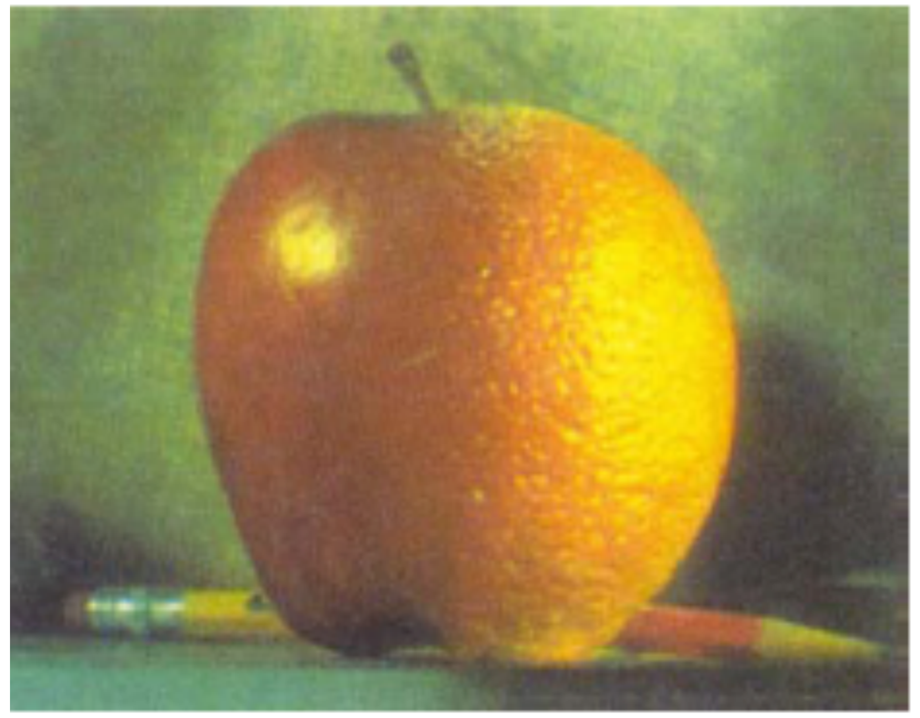
Photo credit: Chris Cameron

Source: A. Efros

Image Blending (PS2 problem)

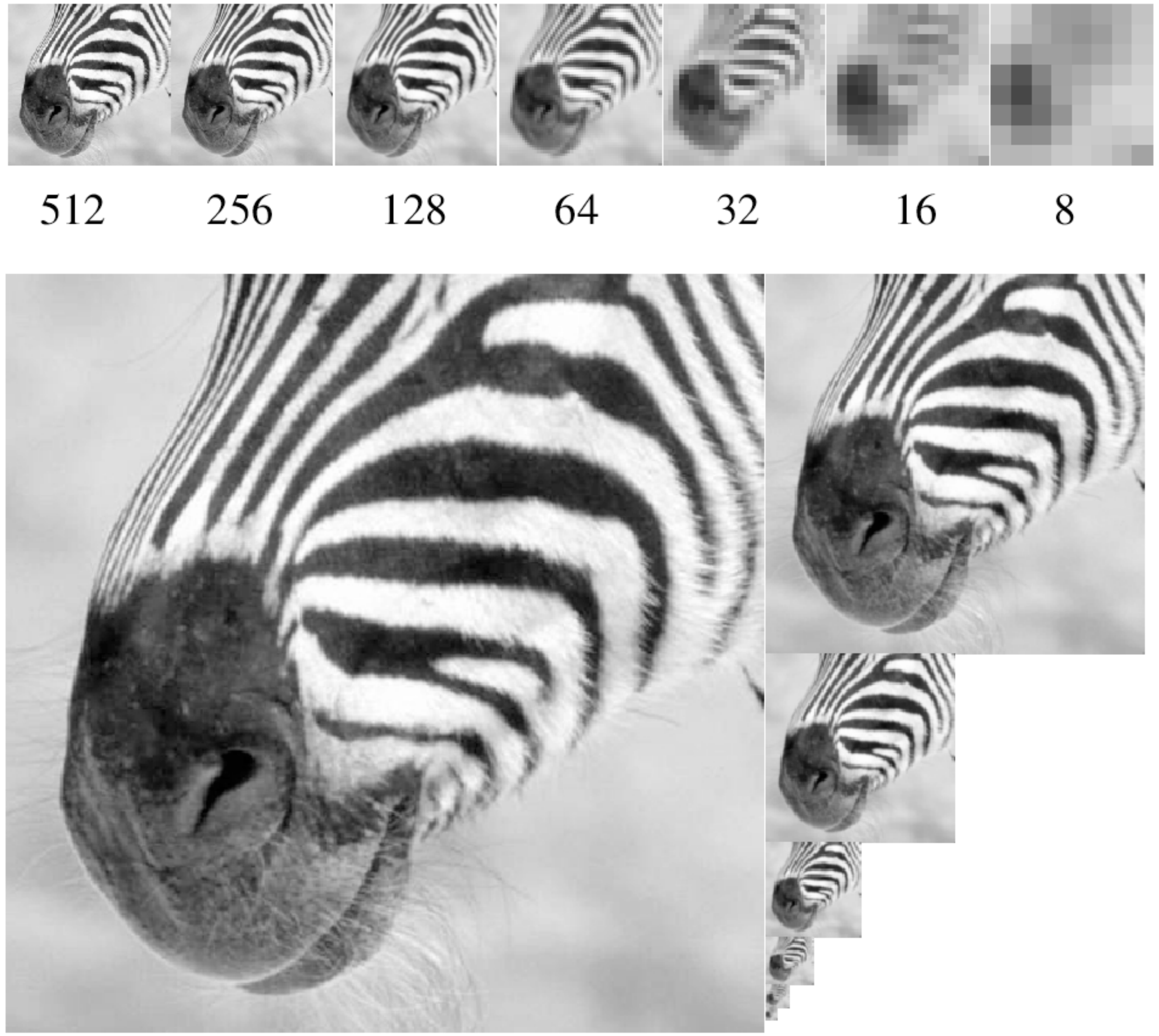


- Build Laplacian pyramid for both images: L_A, L_B
- Build Gaussian pyramid for mask: G
- Build a combined Laplacian pyramid
- Collapse L to obtain the blended image

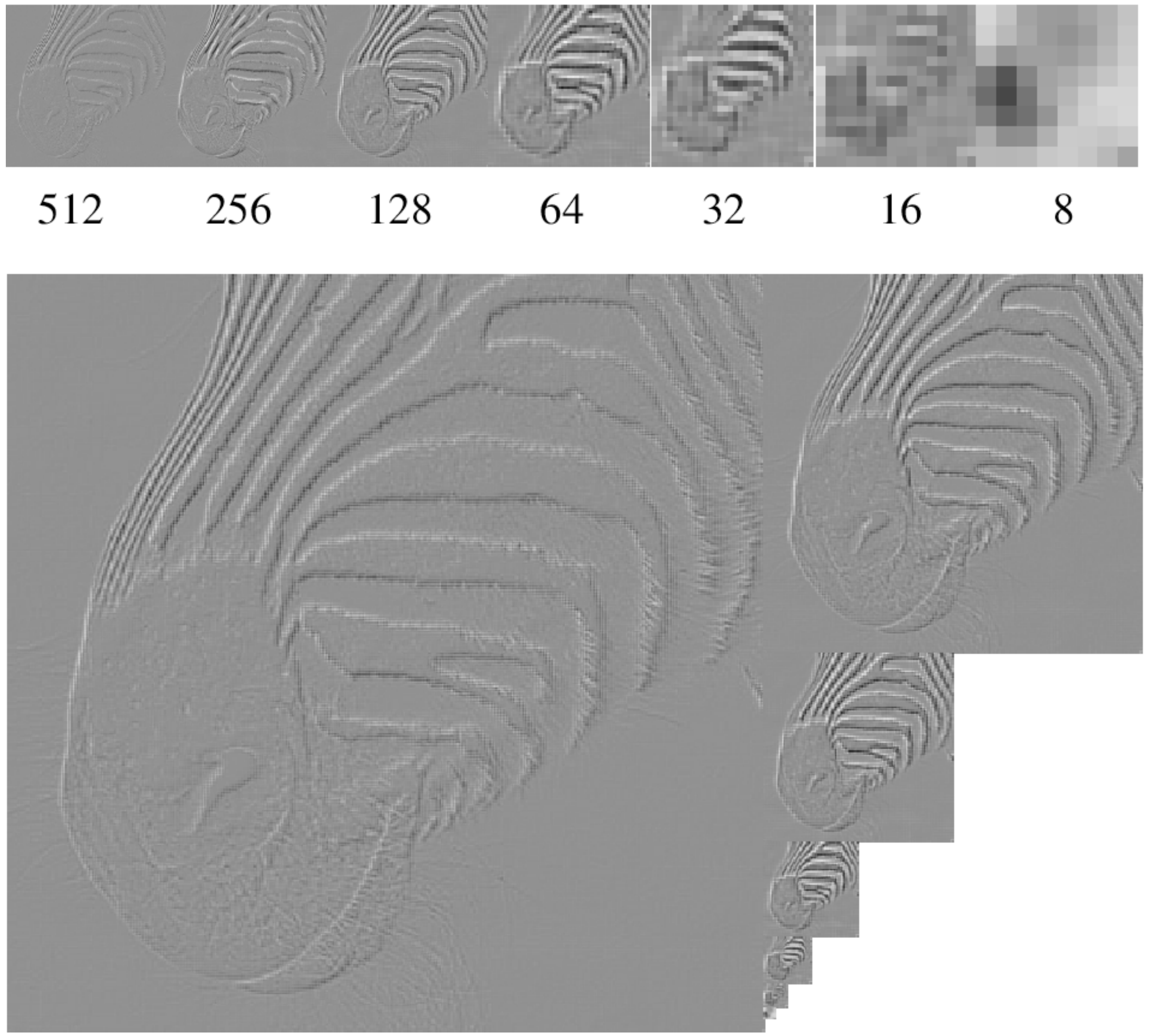


Source: Torralba, Freeman, Isola

Image pyramids



Gaussian Pyramid

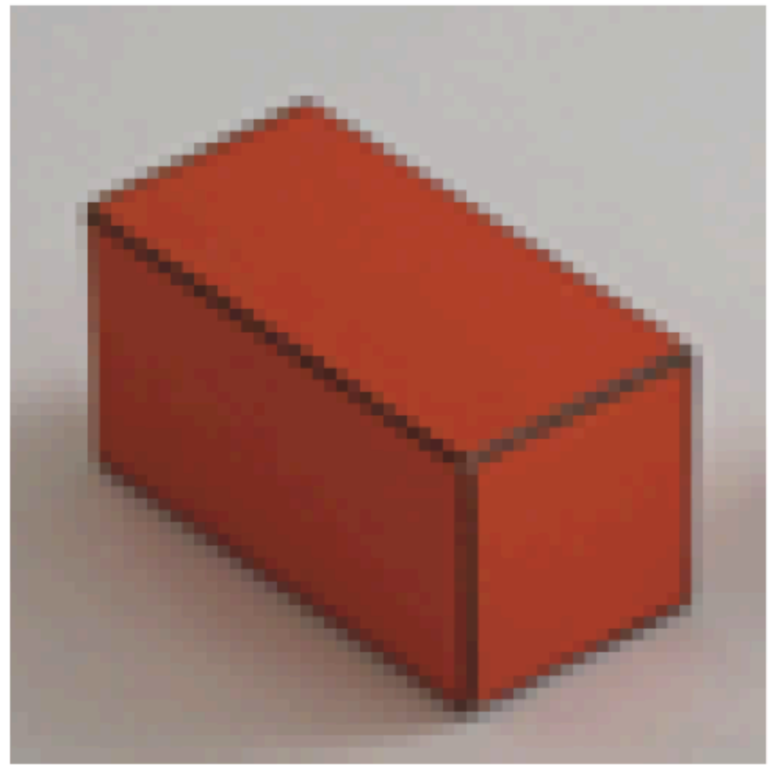


Laplacian Pyramid

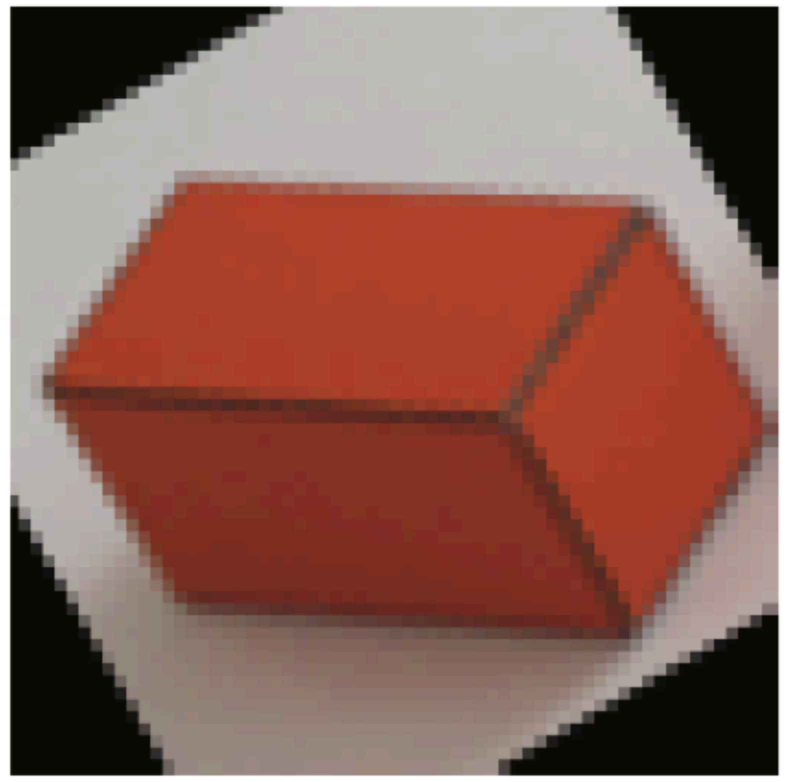
And many more: steerable filters, wavelets, … (and later) convolutional networks!

Source: Torralba, Freeman, Isola

Are Gaussian/Laplacian pyramids linear filters?



g[n, m] → H → f[n, m]



Recall: linear filter

$$f[n,m] = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} h[n,m,k,l] g[k,l]$$

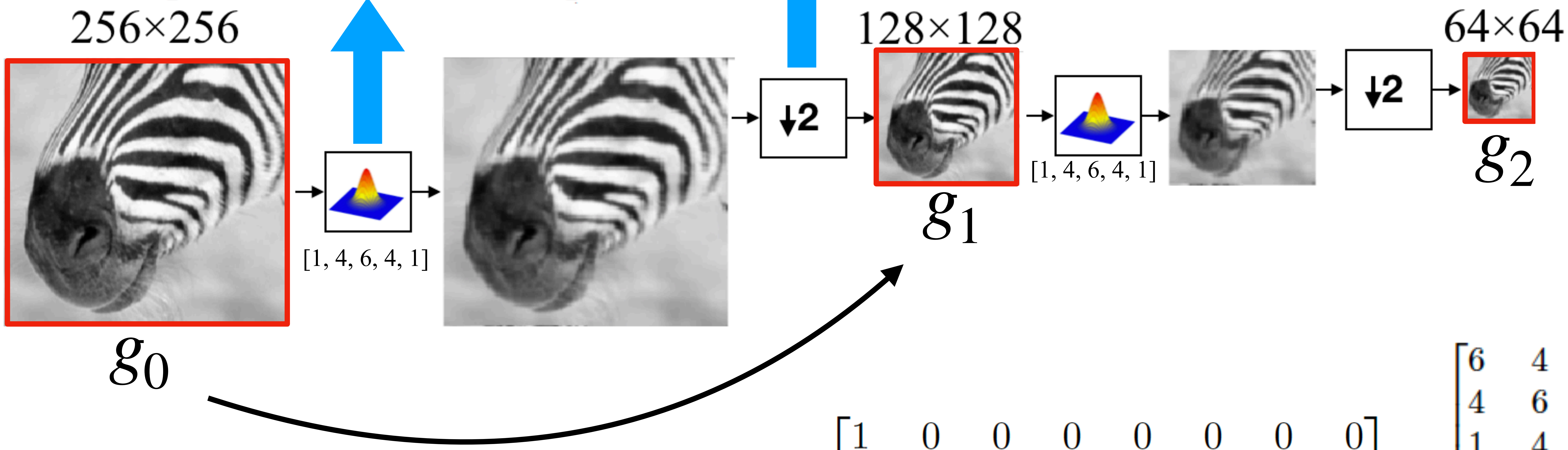
Equivalent to matrix multiplication with some matrix H:

$$f = Hg$$

1D Gaussian pyramid as a linear transformation

$$\frac{1}{16}\begin{bmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

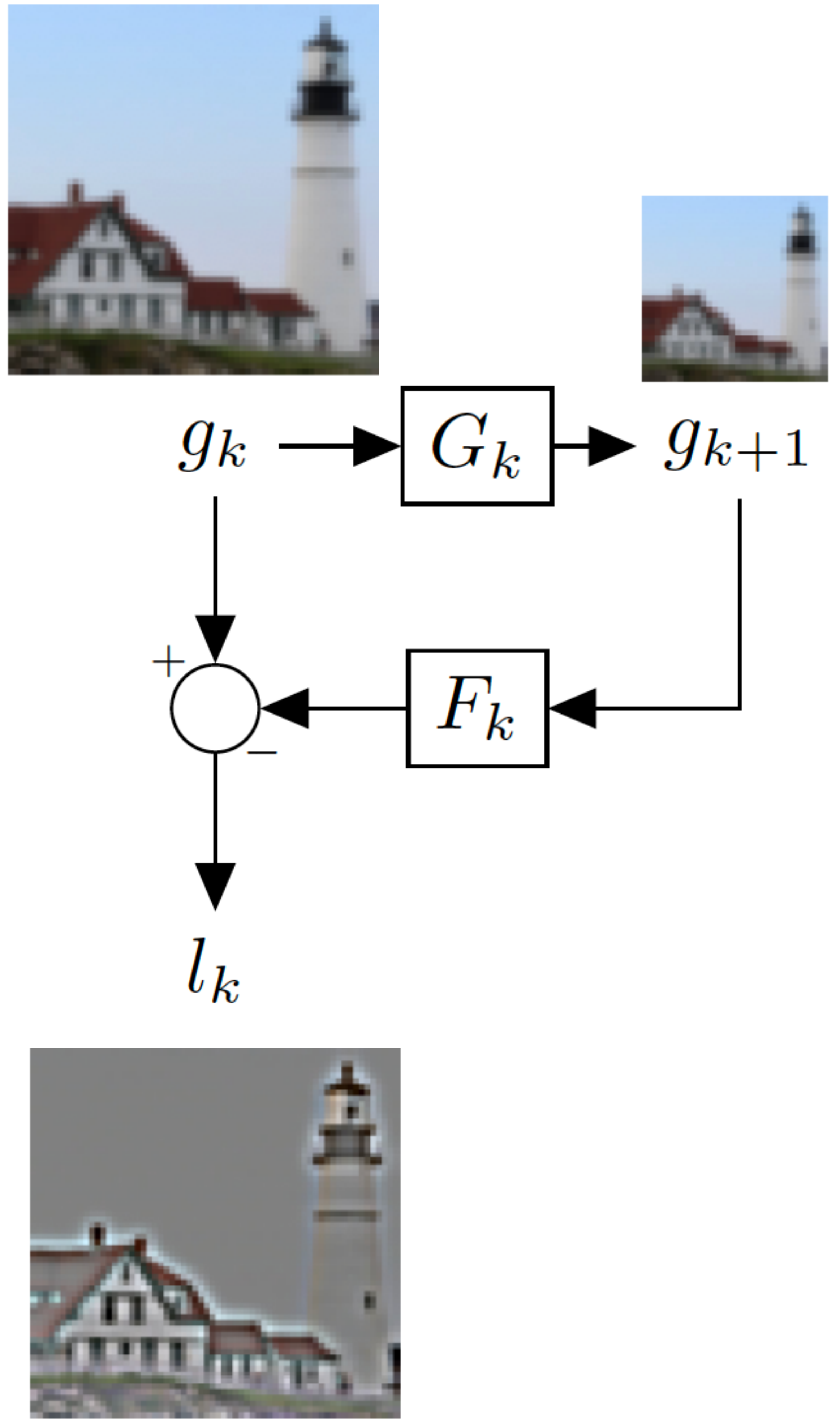


$$g_1 = G_0 g_0$$

$$G_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \frac{1}{16}\begin{bmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 \end{bmatrix}$$

Source: Torralba, Freeman, Isola

Laplacian Pyramid



Blurring and downsampling:

$$G_0 = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{(Downsampling by 2)}} \underbrace{\frac{1}{16} \begin{bmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 \end{bmatrix}}_{\text{(blur)}}$$

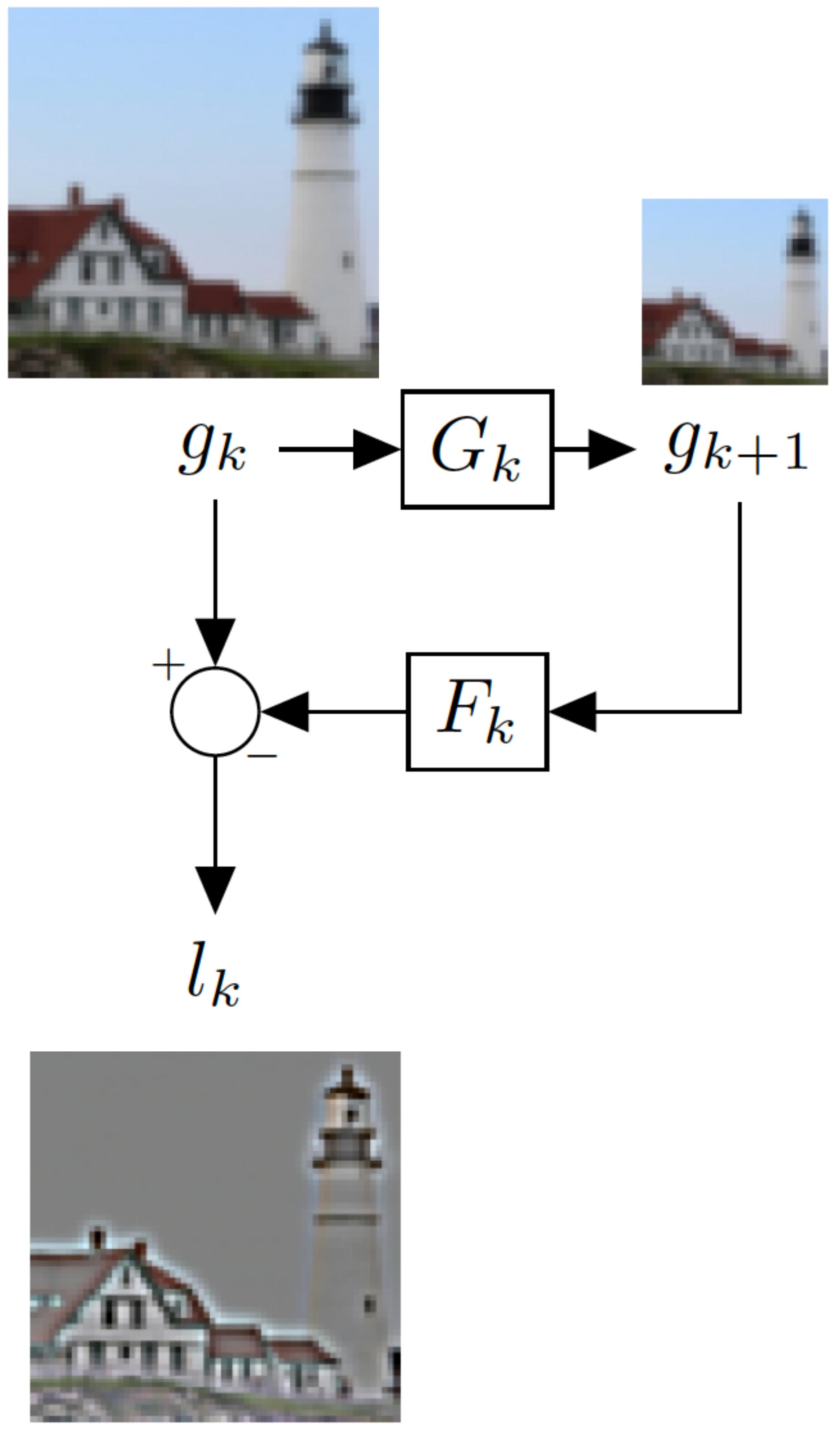
Upsampling and blurring:

$$F_0 = \underbrace{\frac{1}{8} \begin{bmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 \end{bmatrix}}_{\text{(blur)}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\text{(Upsampling by 2)}}$$

$$\boxed{l_0 = (I_0 - F_0 G_0) g_0}$$

Source: Torralba, Freeman, Isola

The Laplacian Pyramid

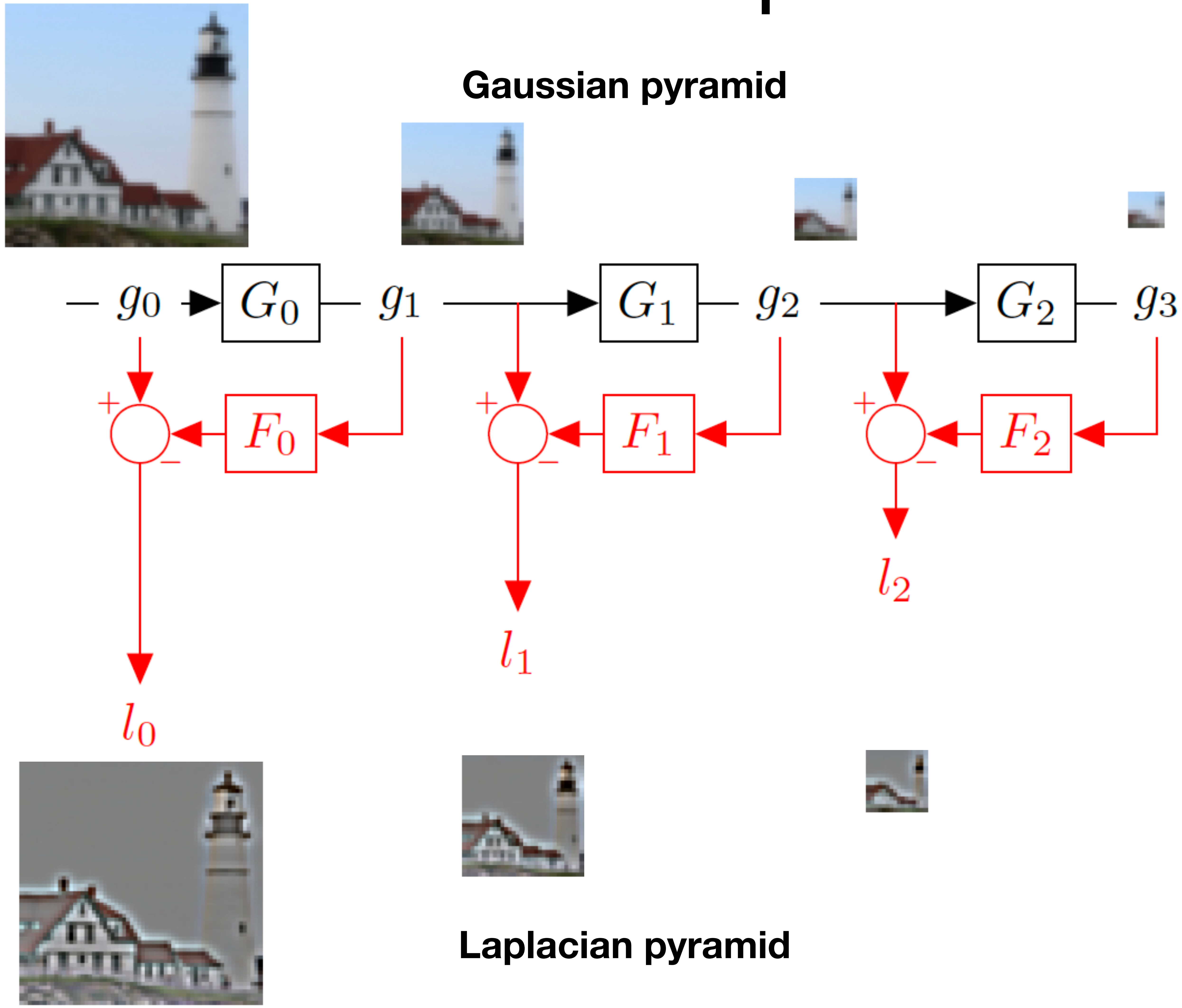


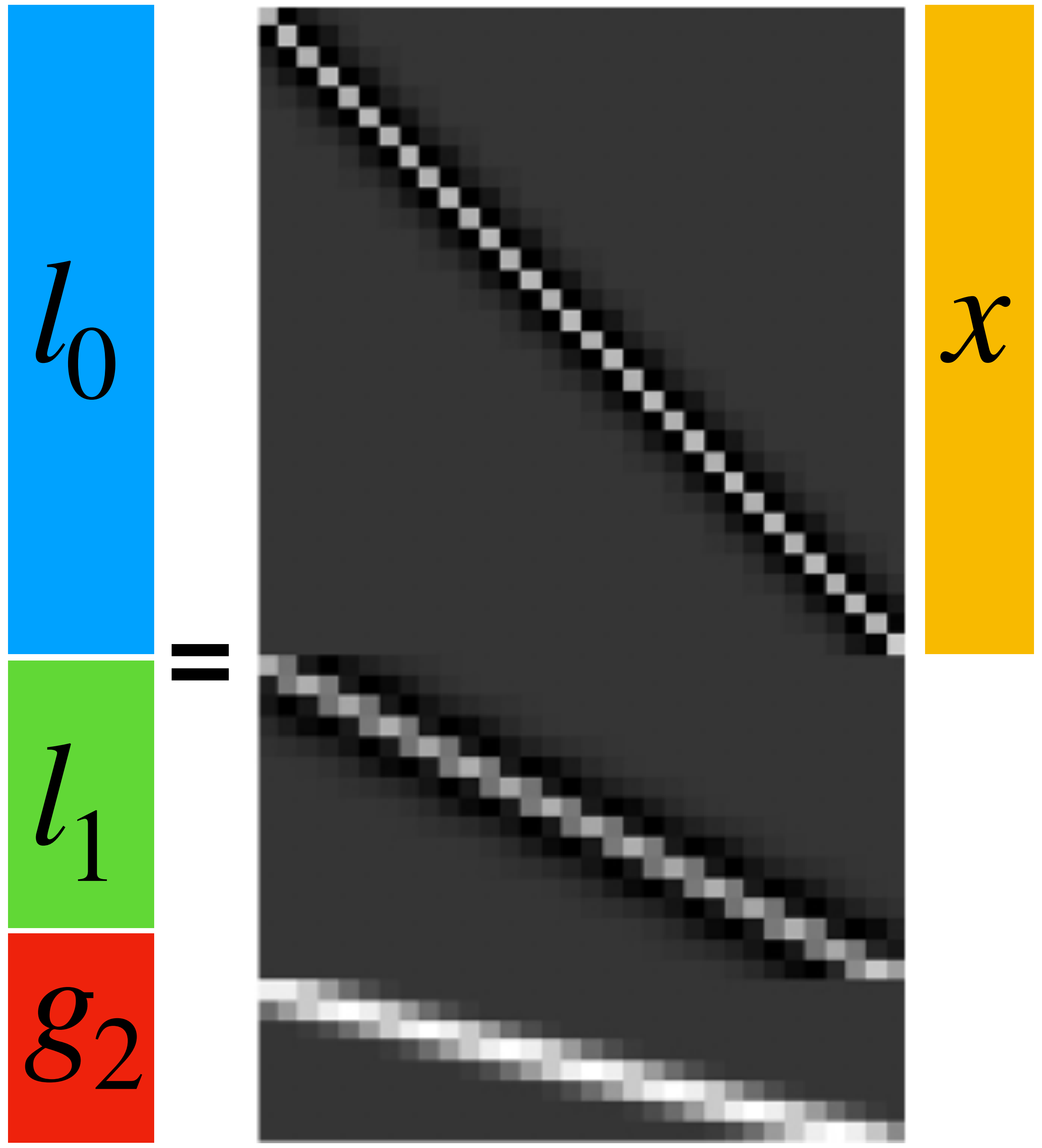
$$\boxed{l_0 = (I_0 - F_0 G_0) g_0}$$

$$= \frac{1}{256} \begin{bmatrix} 182 & -56 & -24 & -8 & -2 & 0 & 0 & 0 \\ -56 & 192 & -56 & -32 & -8 & 0 & 0 & 0 \\ -24 & -56 & 180 & -56 & -24 & -8 & -2 & 0 \\ -8 & -32 & -56 & 192 & -56 & -32 & -8 & 0 \\ -2 & -8 & -24 & -56 & 180 & -56 & -24 & -8 \\ 0 & 0 & -8 & -32 & -56 & 192 & -56 & -32 \\ 0 & 0 & -2 & -8 & -24 & -56 & 182 & -48 \\ 0 & 0 & 0 & 0 & -8 & -32 & -48 & 224 \end{bmatrix} x$$

Source: Torralba, Freeman, Isola

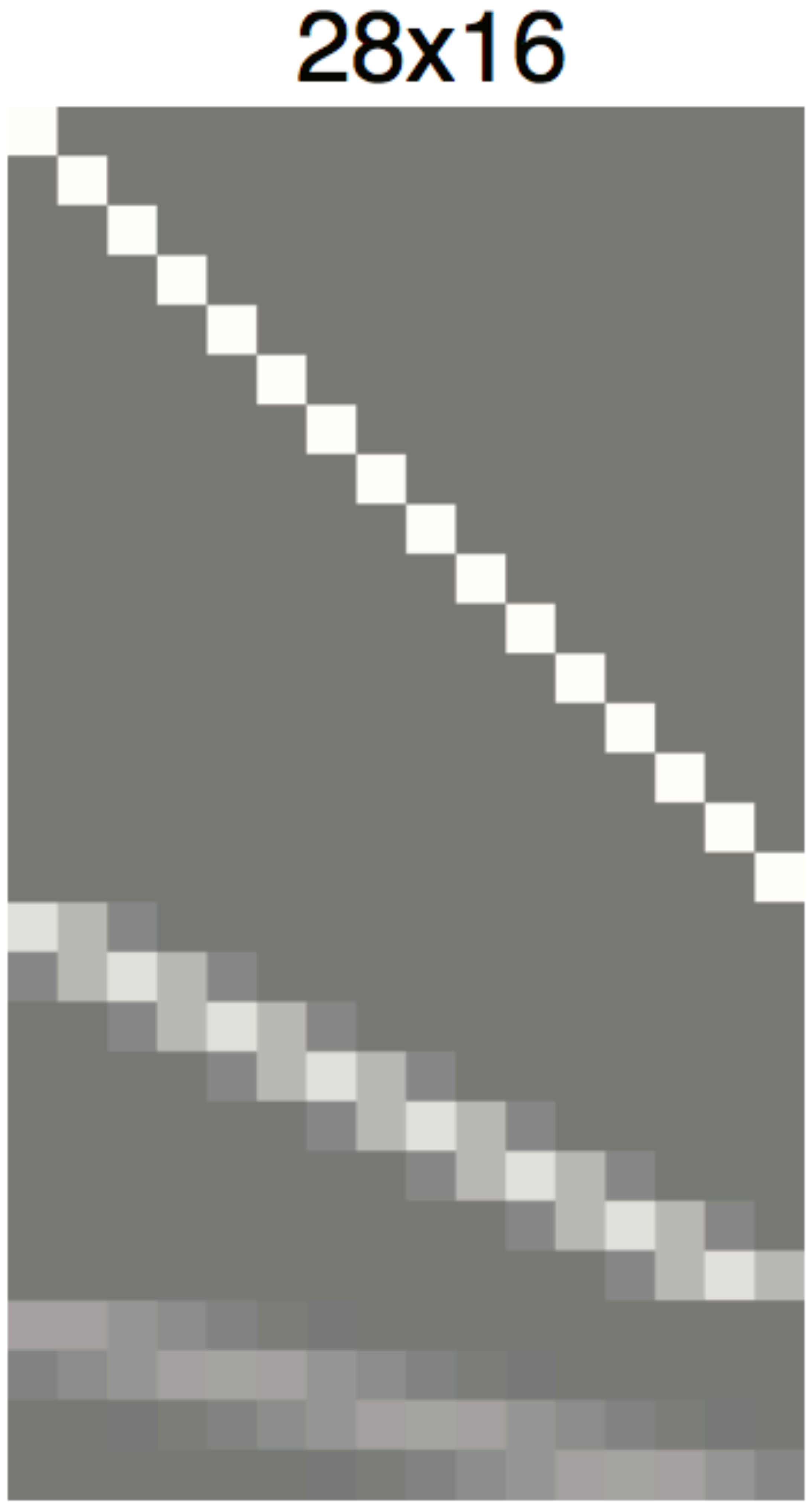
Laplacian Pyramid



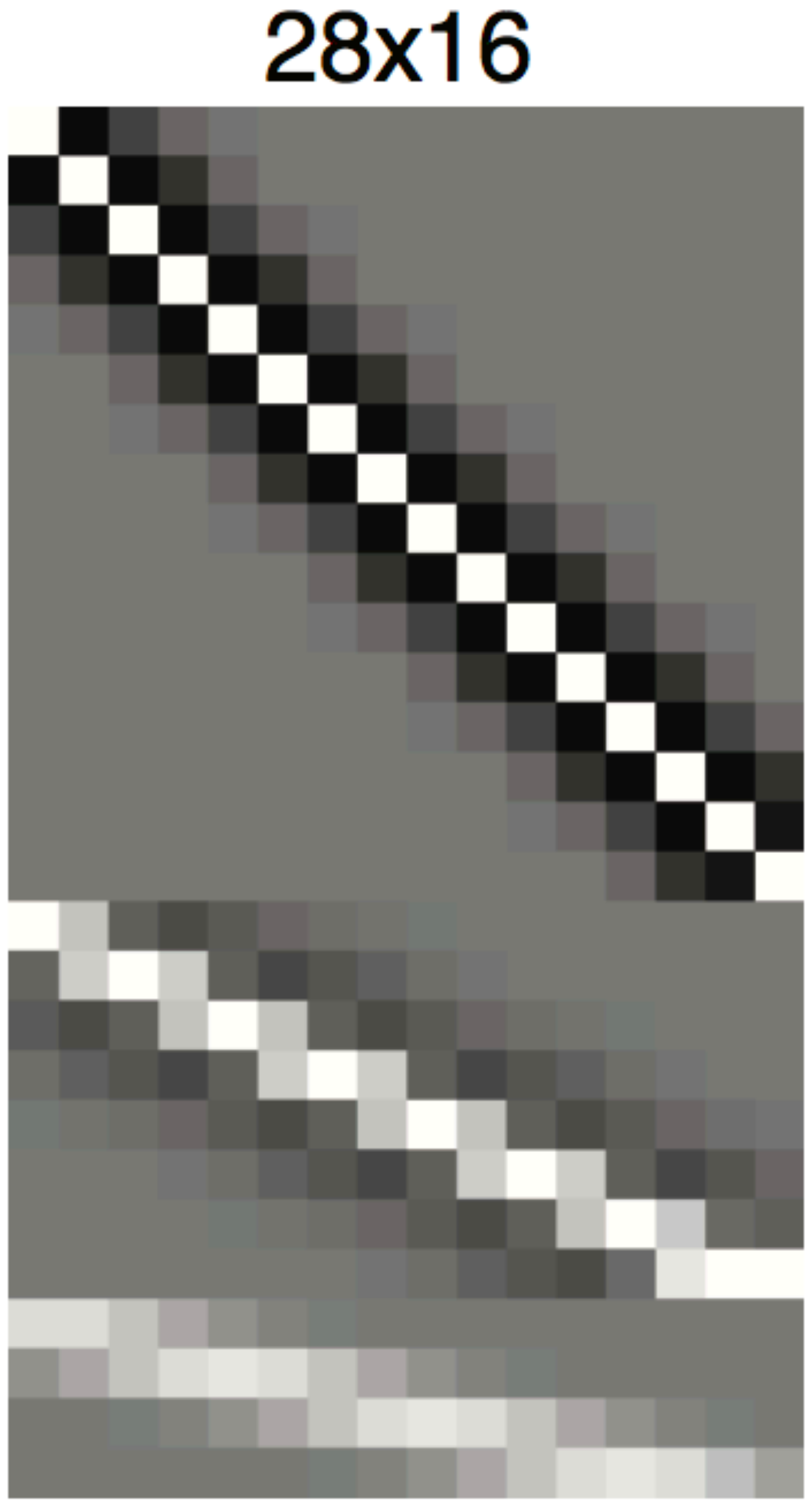


Source: Torralba, Freeman, Isola

Linear Image Transforms



Gaussian pyramid



Laplacian pyramid

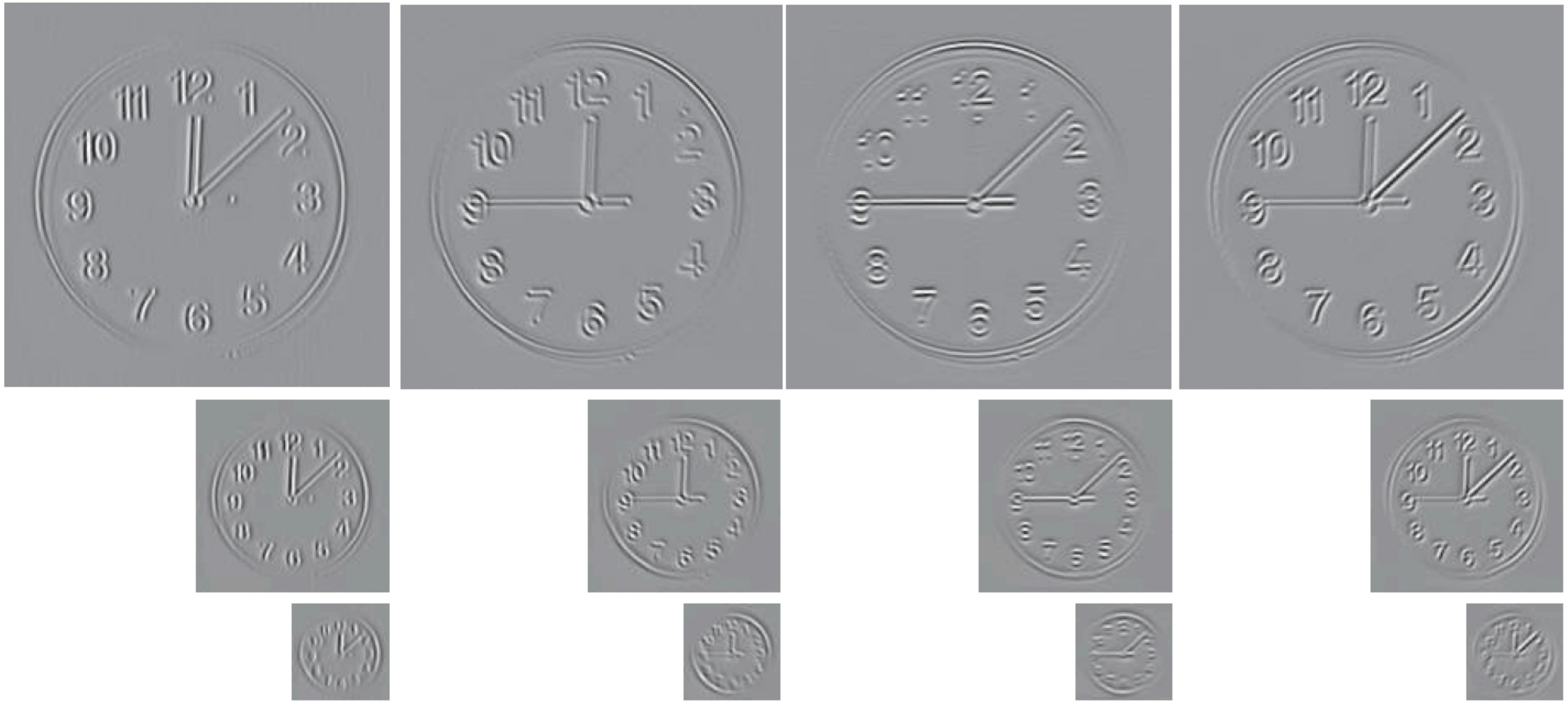
Source: Torralba, Freeman, Isola

One other pyramid: the Steerable Pyramid

Source: Torralba, Freeman, Isola

Steerable Pyramid

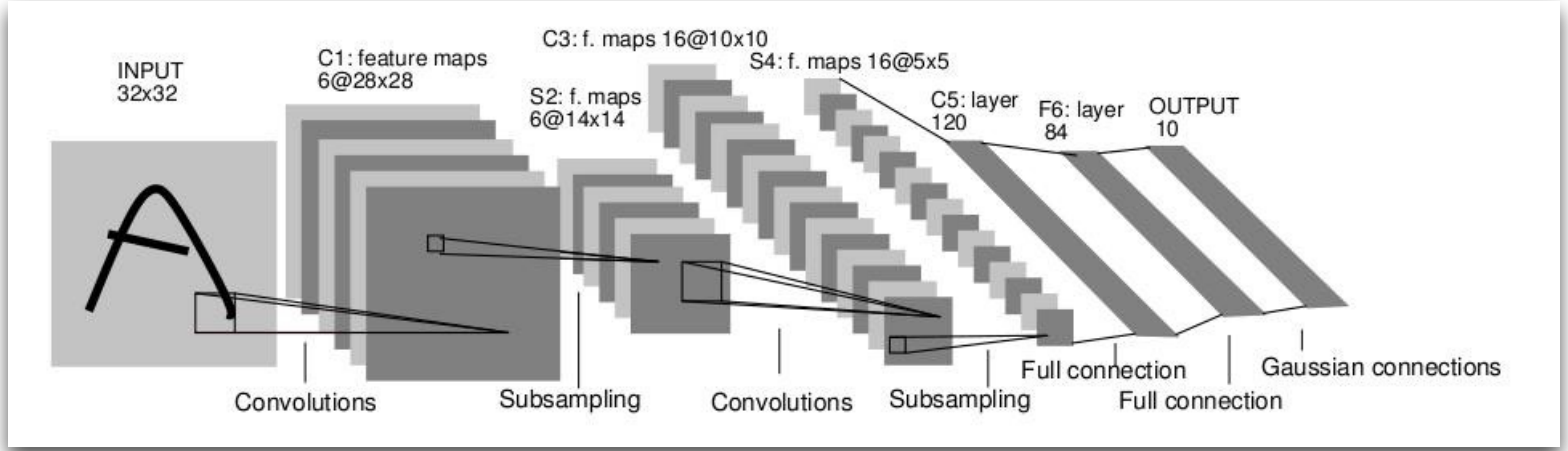
Oriented gradient



Source: Torralba, Freeman, Isola

For more powerful nonlinear models…

- Later, we'll study a powerful nonlinear model, convolutional neural networks
- Their main building blocks are linear filters.



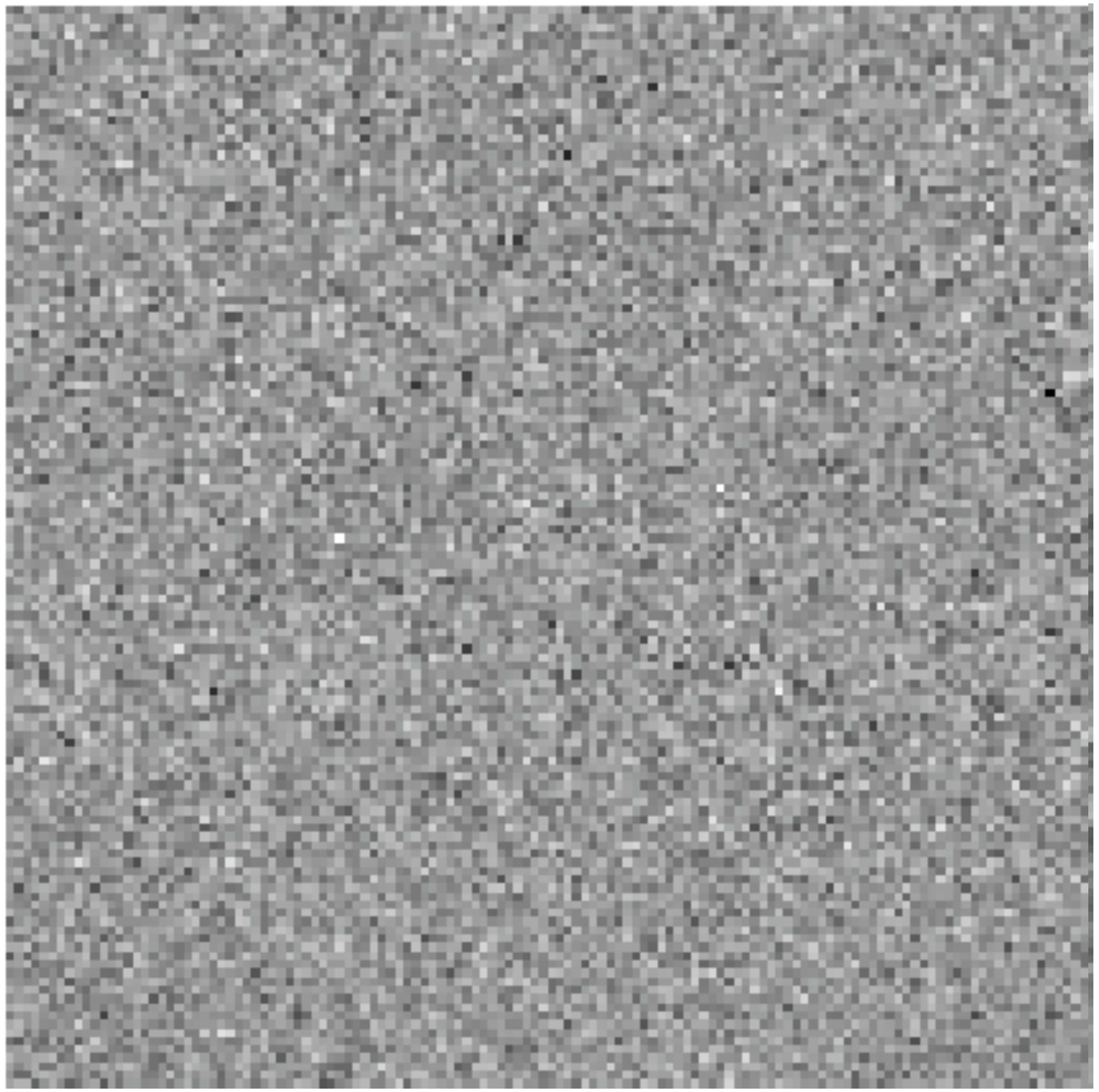
[LeCun et al. 1989]

Today

- Image pyramids
- **Image statistics**
- Texture synthesis

What makes an image "natural"?

Natural image

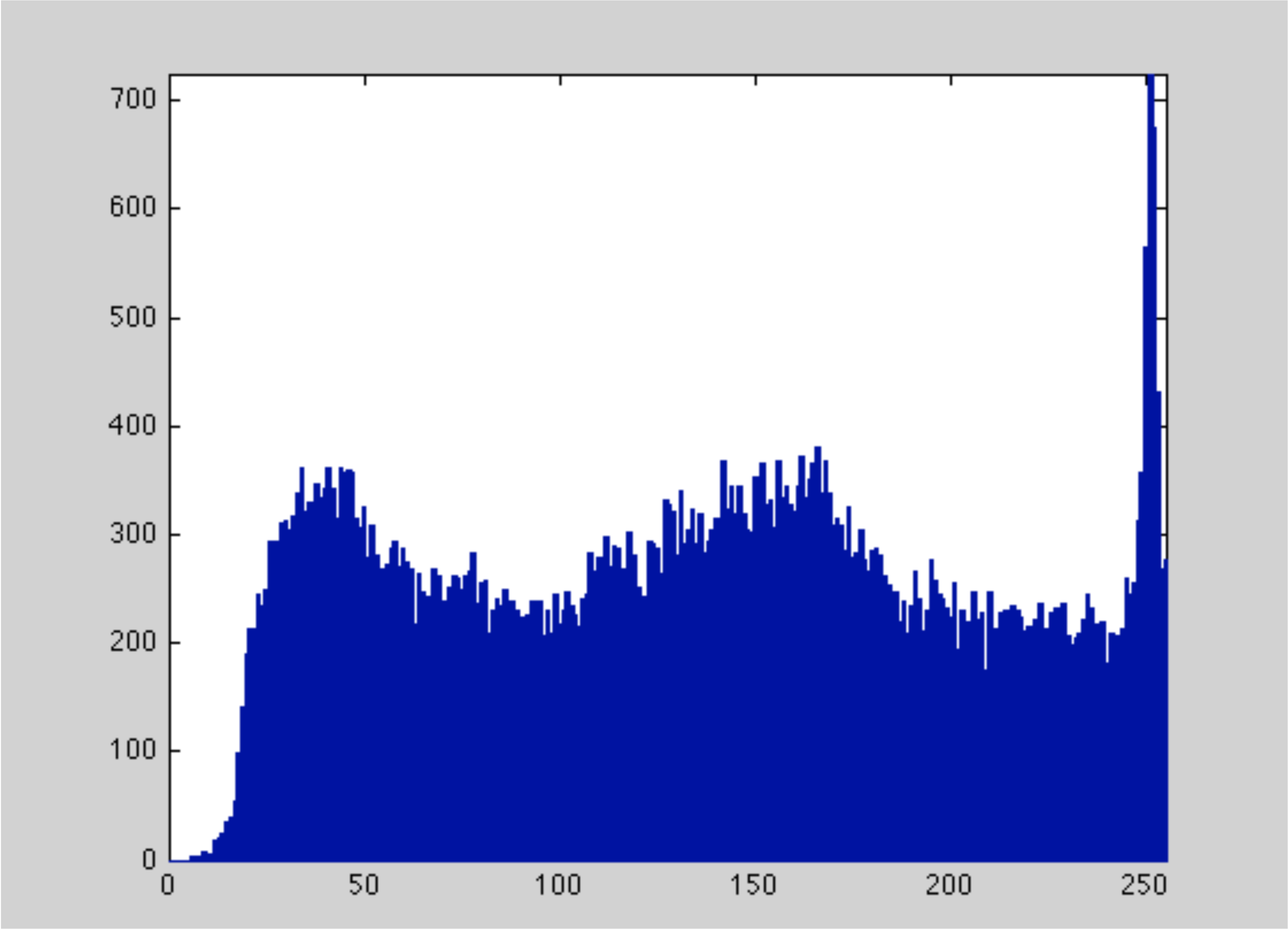


"Fake" image

Source: Torralba, Freeman, Isola

Is it the distribution of pixel intensities?

No real structure here…



Intensity histogram

What about gradients?

$\otimes$ [-1, 1] =

h[m,n]

g[m,n]

f[m,n]

Source: Torralba, Freeman, Isola

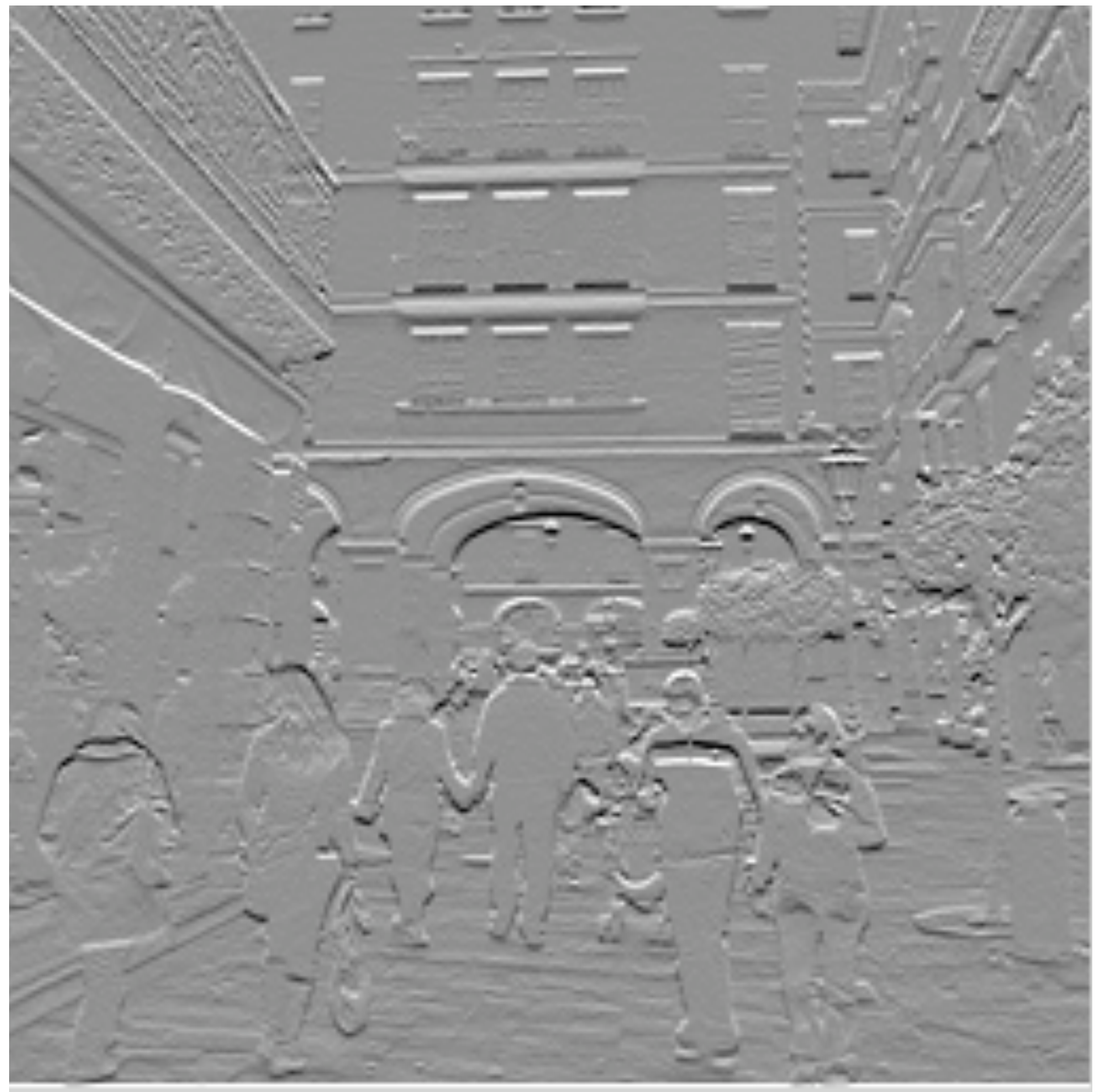
What about gradients?

$\otimes$

$[-1, 1]^T$

h[m,n]

=

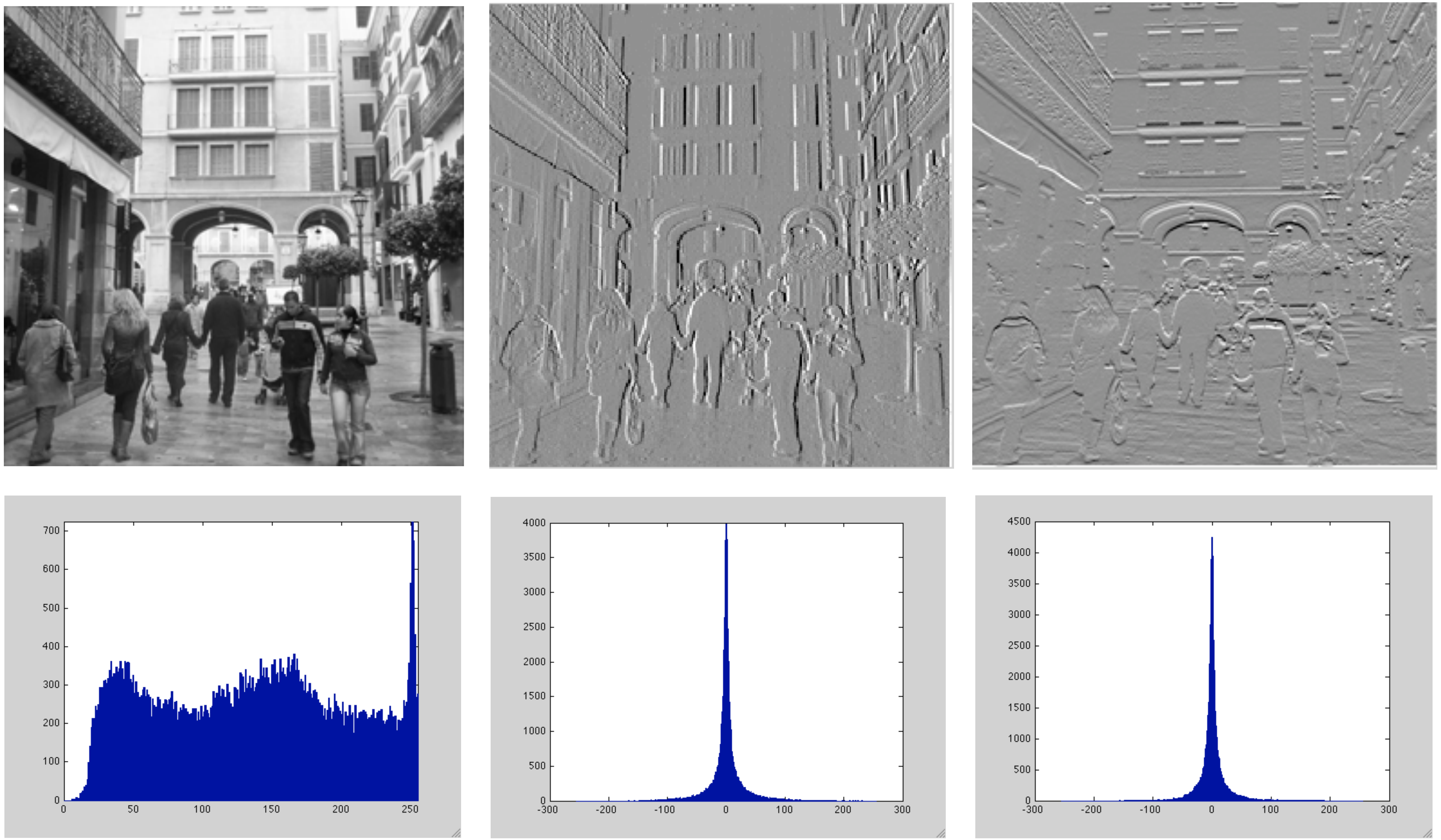


g[m,n]

f[m,n]

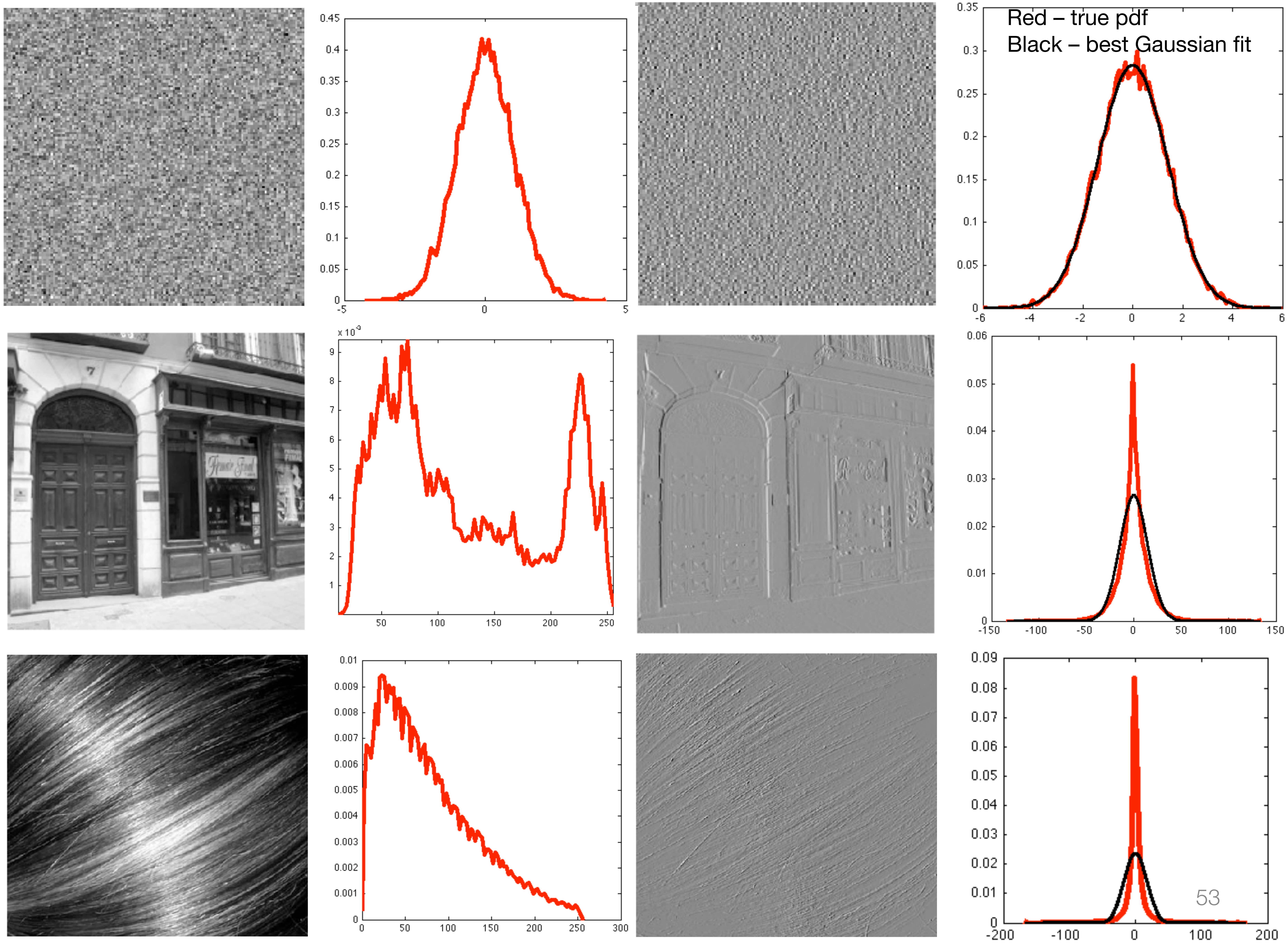
Source: Torralba, Freeman, Isola

Filter response distribution is pretty consistent!



Source: Torralba, Freeman, Isola

Image | Intensity histogram | [1 -1] filter output | [1 -1] output histogram

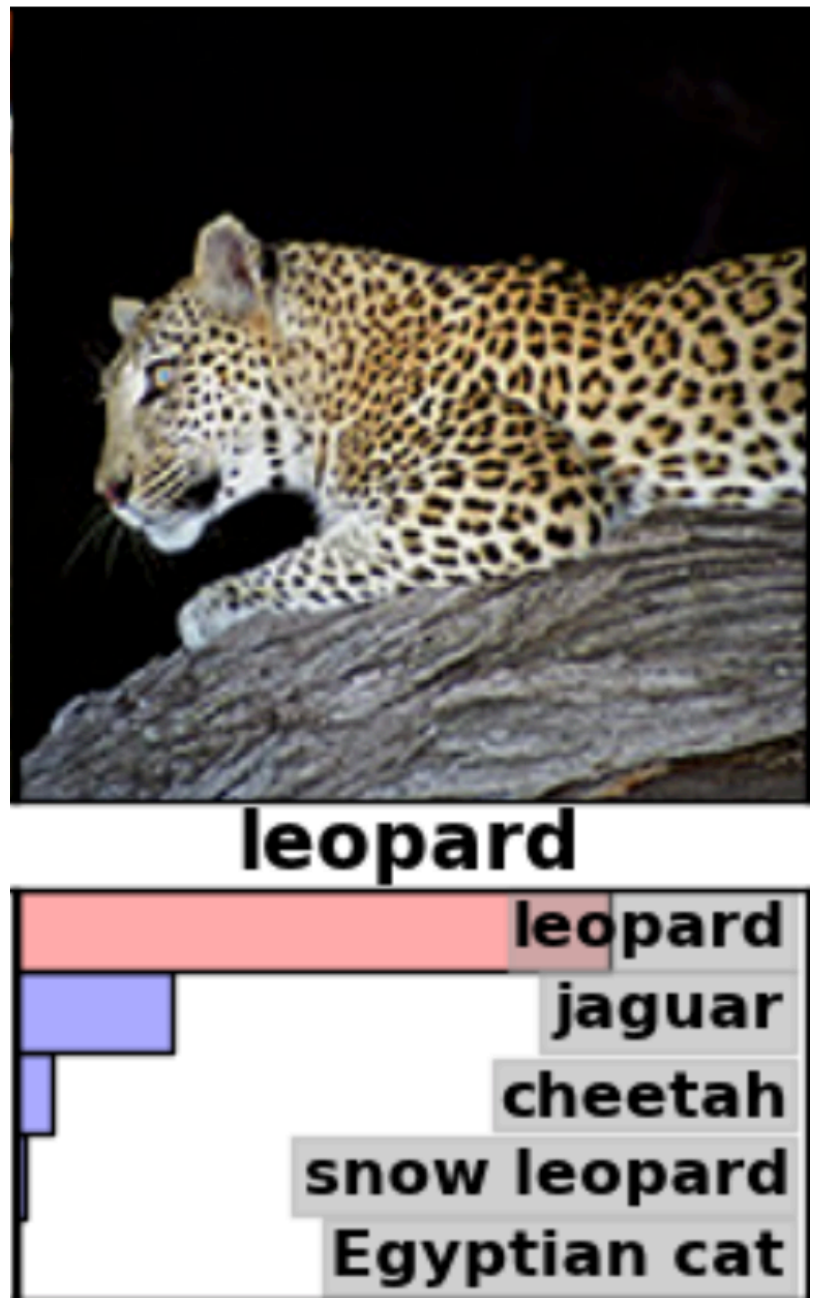


Source: Torralba, Freeman, Isola

Applications of image statistics

Compression

Image restoration



Learning
(later in course)

Taking a picture…

What the camera give us…

How do we correct this?

Slides R. Fergus

Deblurring

Slides R. Fergus

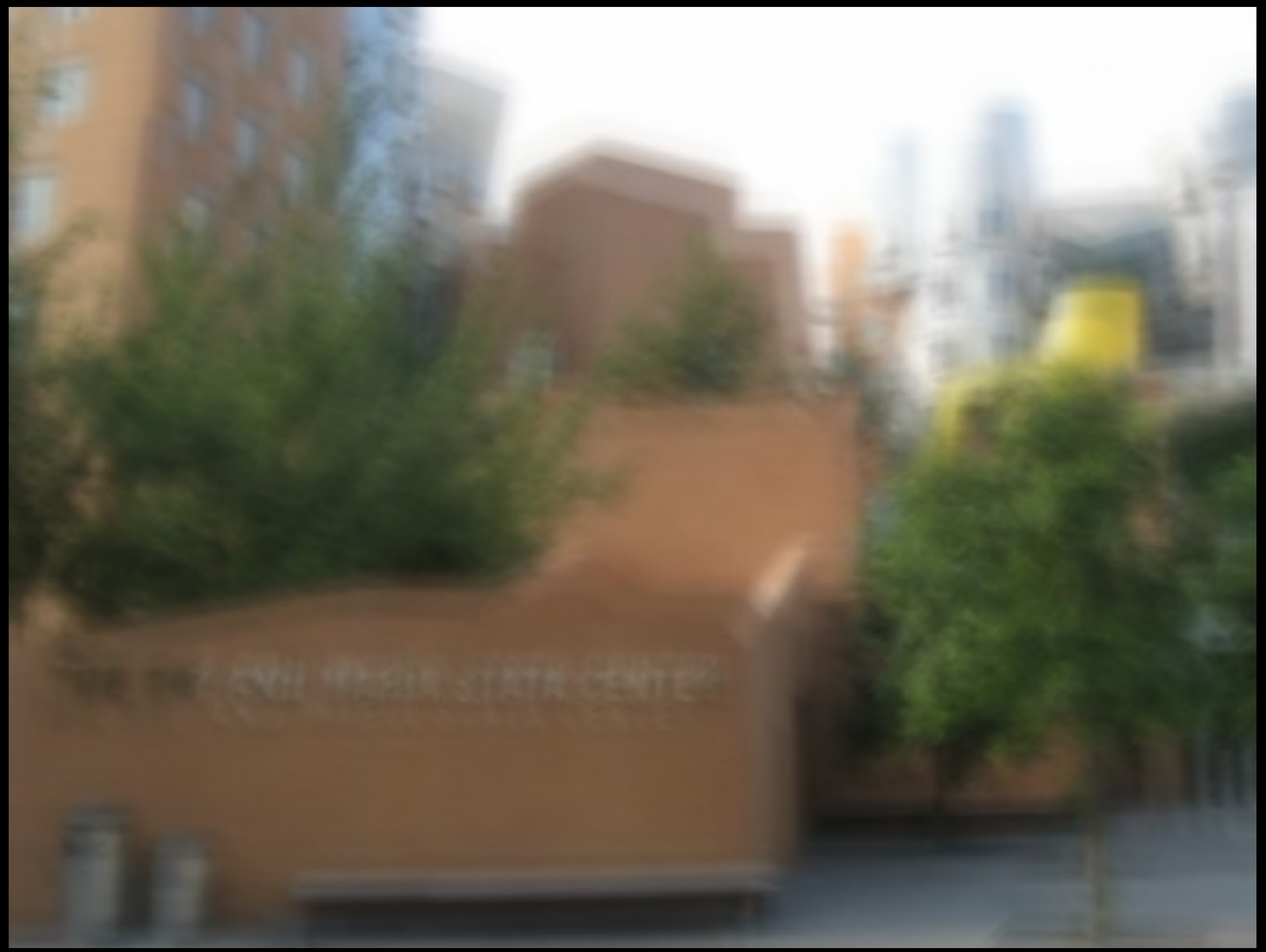
Deblurring

Slides R. Fergus

Deblurring

Slides R. Fergus

Image formation process



Blurry image

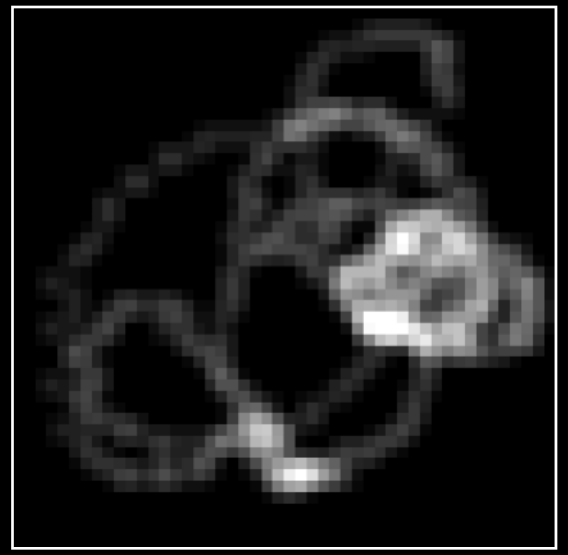
Input to algorithm

=

Sharp image

Desired output

$\otimes$

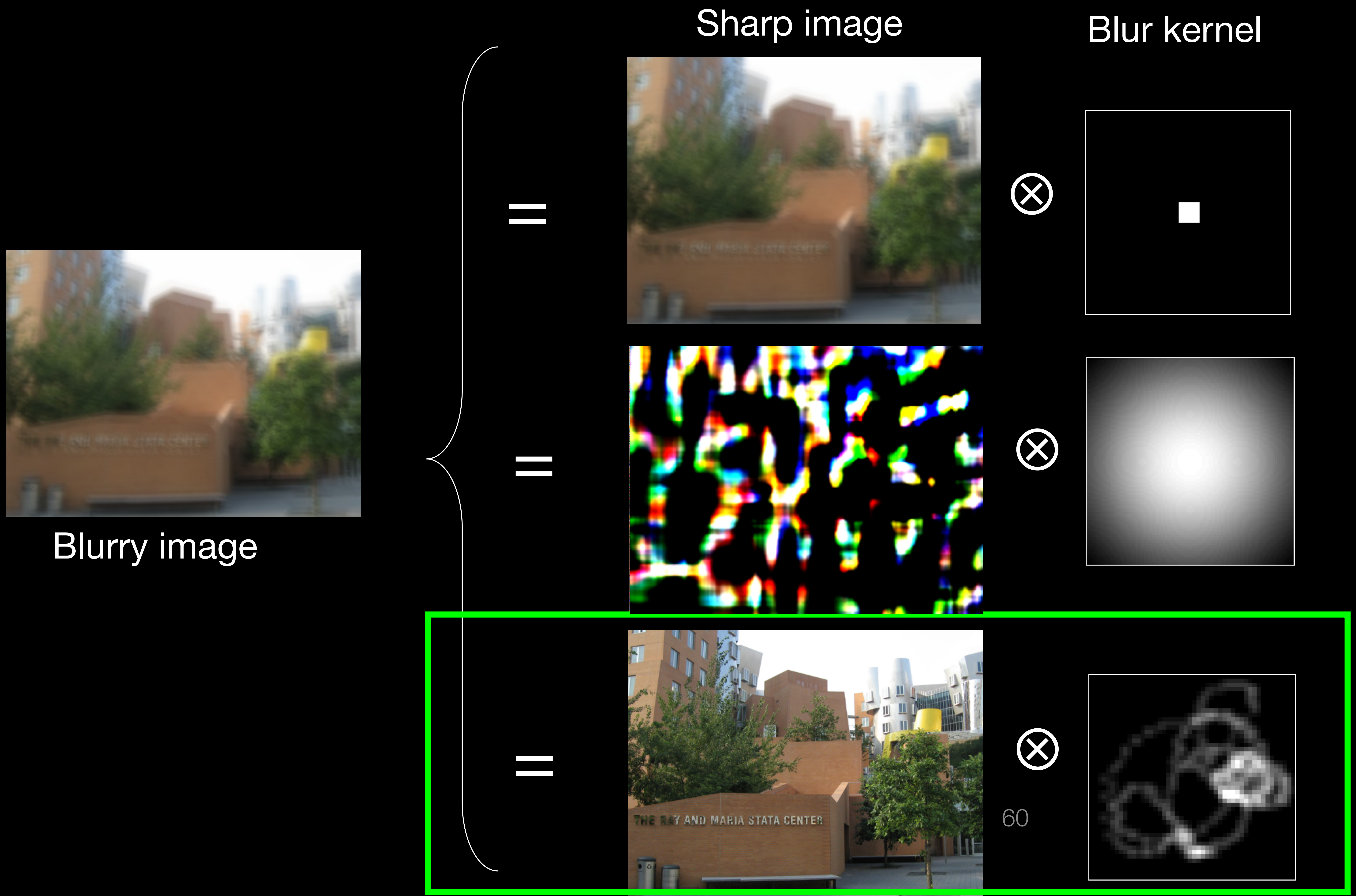


Blur kernel

Convolution operator

Source: R. Fergus

Multiple possible solutions

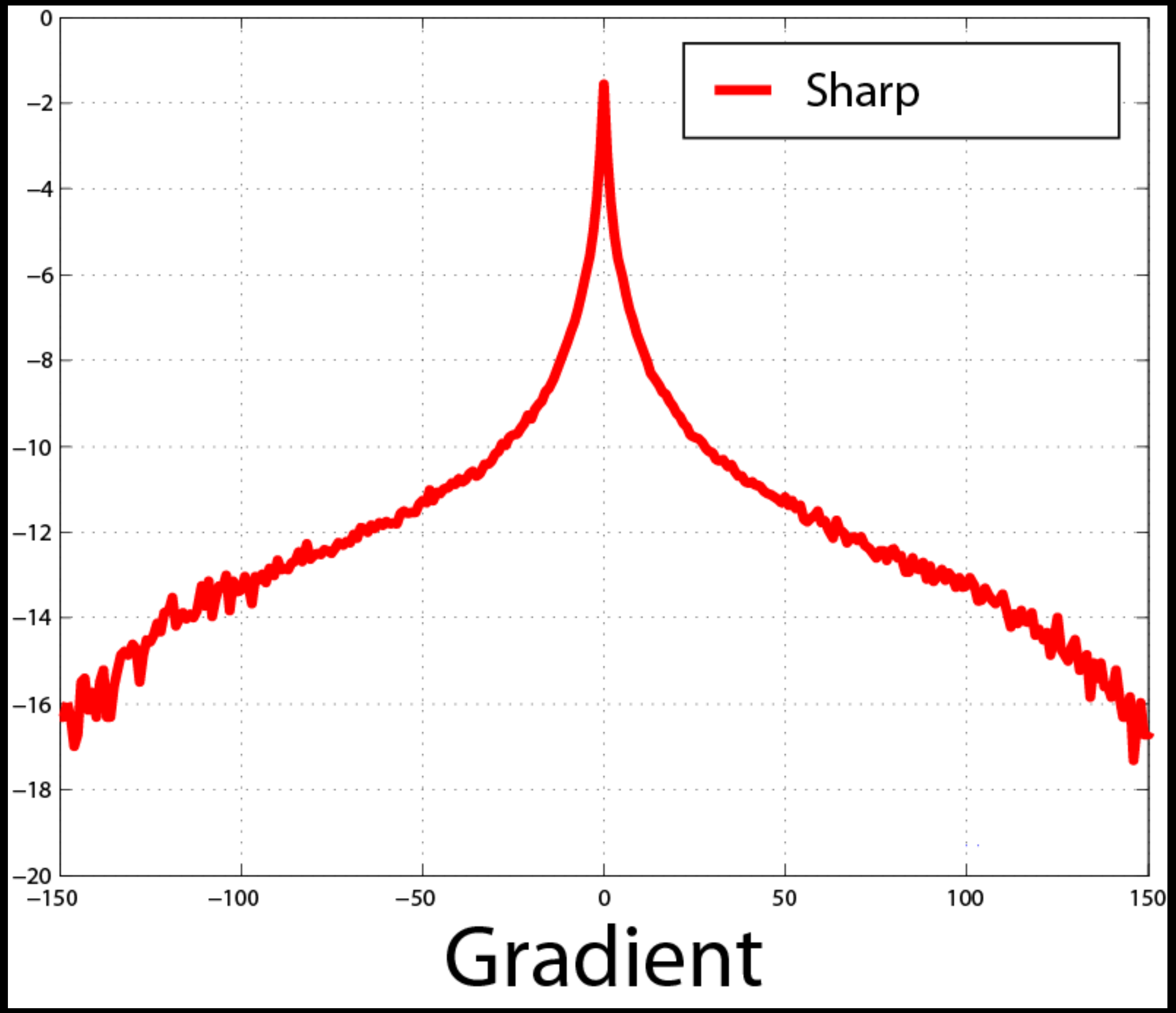


Source: R. Fergus

Natural image statistics

Characteristic distribution with heavy tails

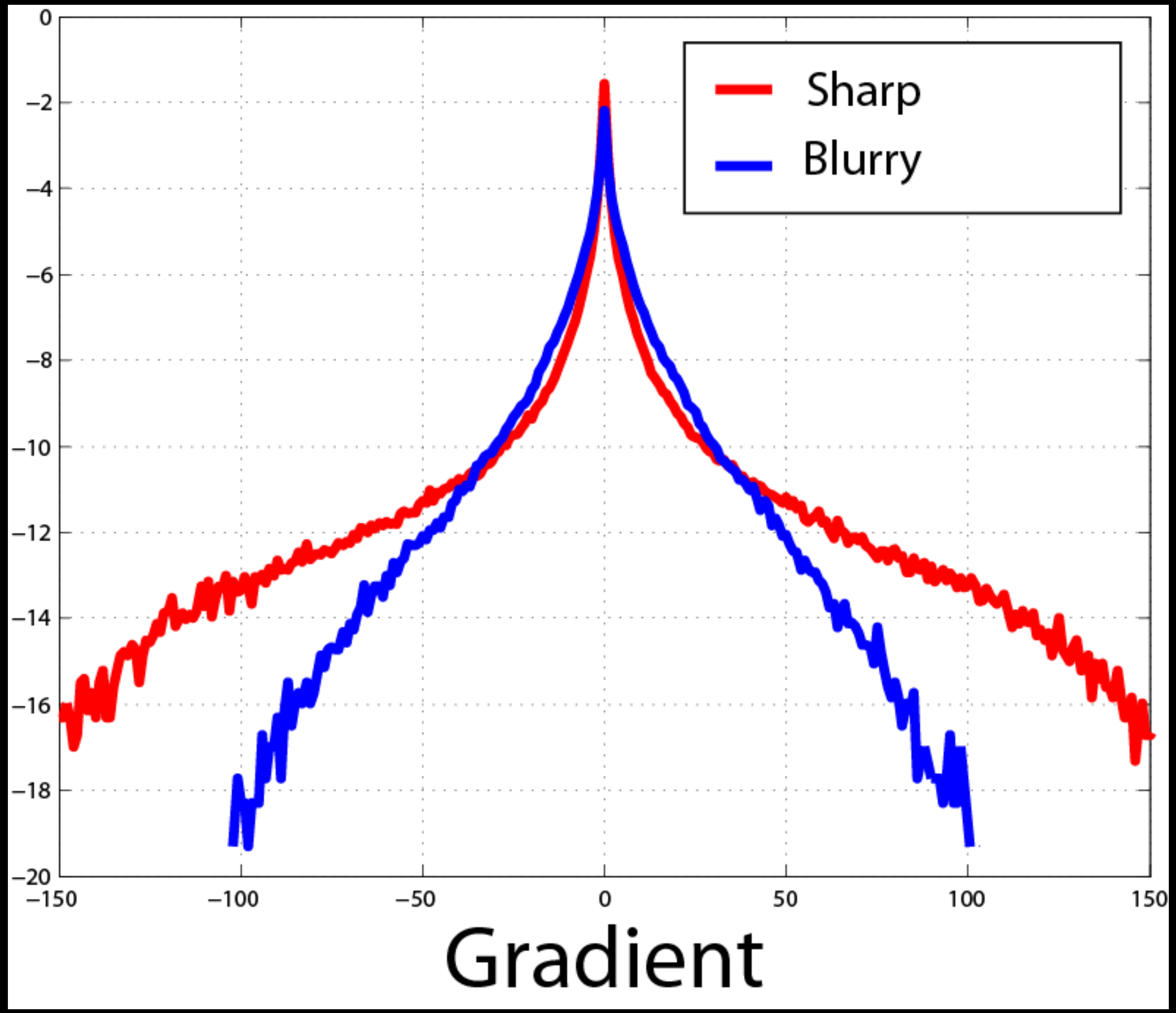
Histogram of image gradients



Source: R. Fergus

Blurry images have different statistics

Histogram of image gradients



Source: R. Fergus

Removing motion blur

Original photograph

After matching filter distribution

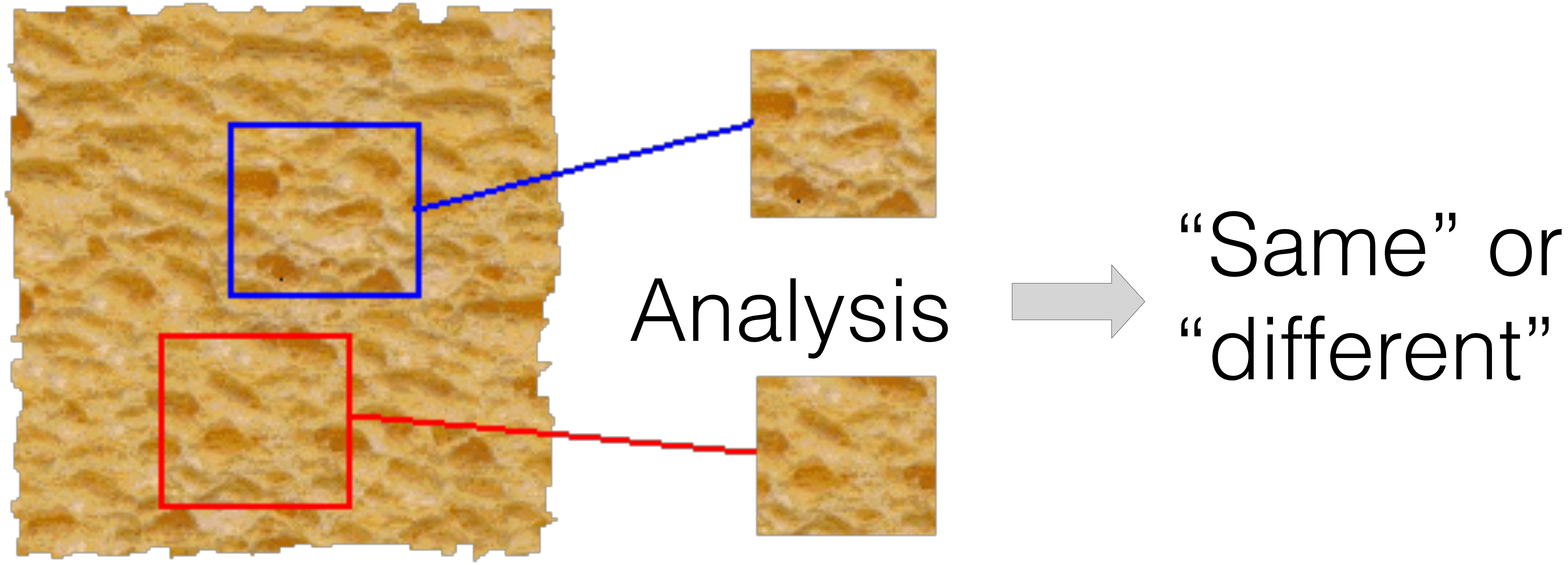
Solve for an image with a distribution of edge gradients that "matches" a normal image.

Source: R. Fergus

Today

- Image pyramids
- Image statistics
- **Texture synthesis**

Texture analysis

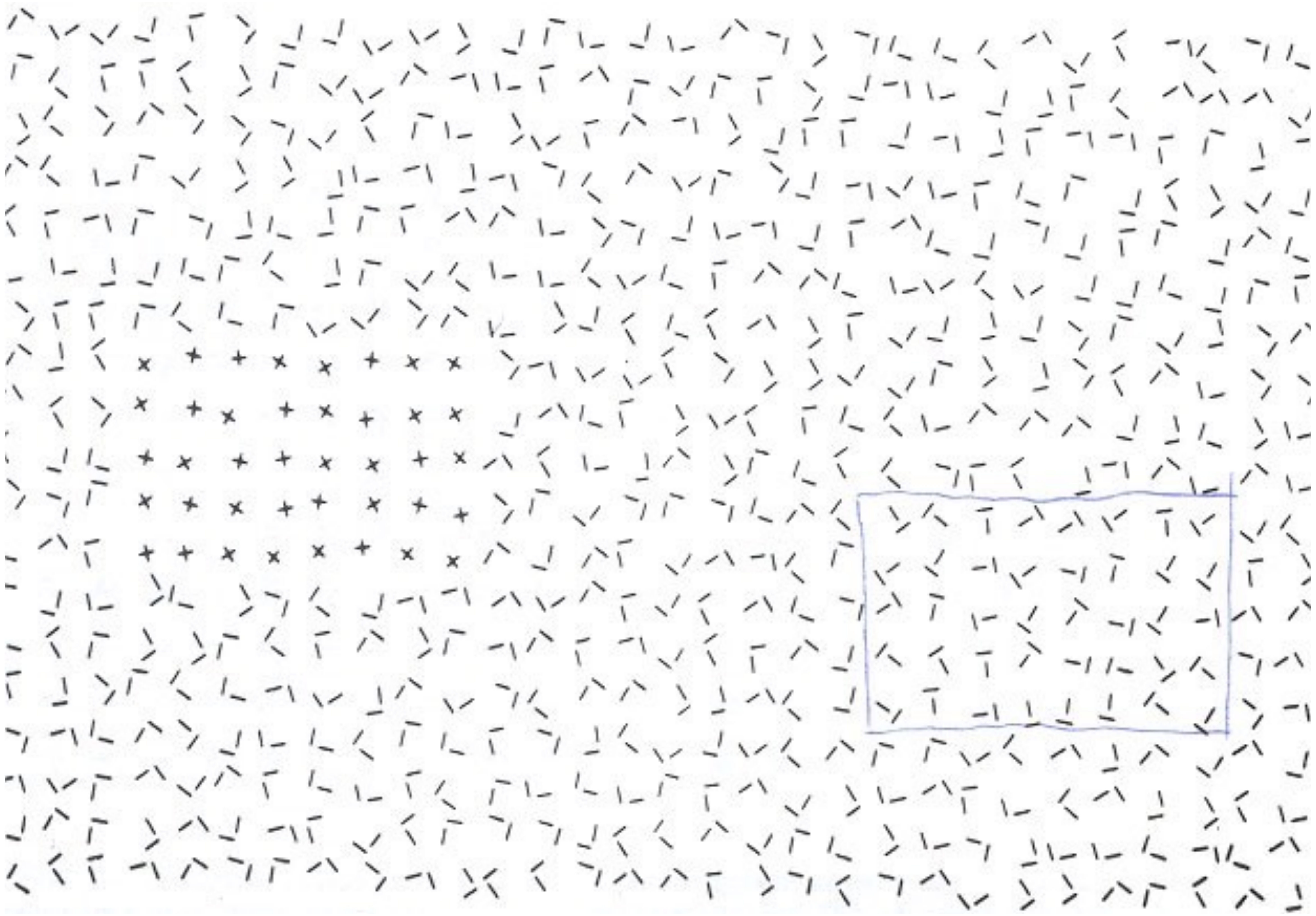


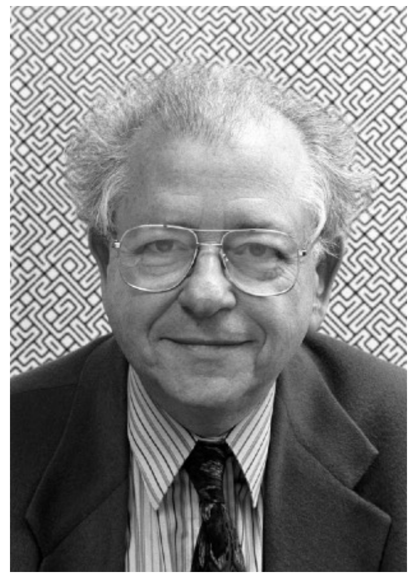
True (infinite) texture

What we'd like: are they made of the same "stuff". Are these textures similar?

Source: A. Efros

How do humans analyze texture?





Béla Julesz

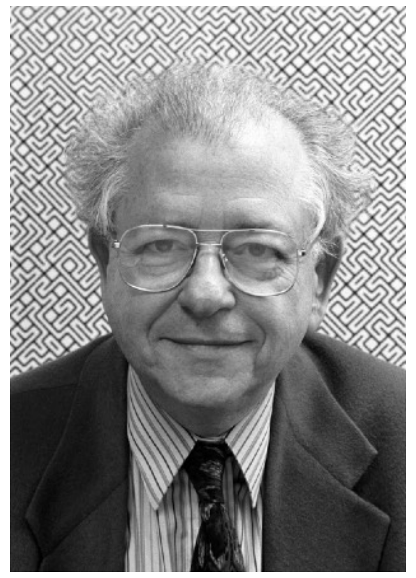
Human vision is sensitive to the difference of some types of elements and appears to be “numb” on other types of differences.

Source: A. Efros

Julesz Conjecture

Textures cannot be spontaneously discriminated if they have the same first-order and second-order statistics and differ only in their third-order or higher-order statistics.

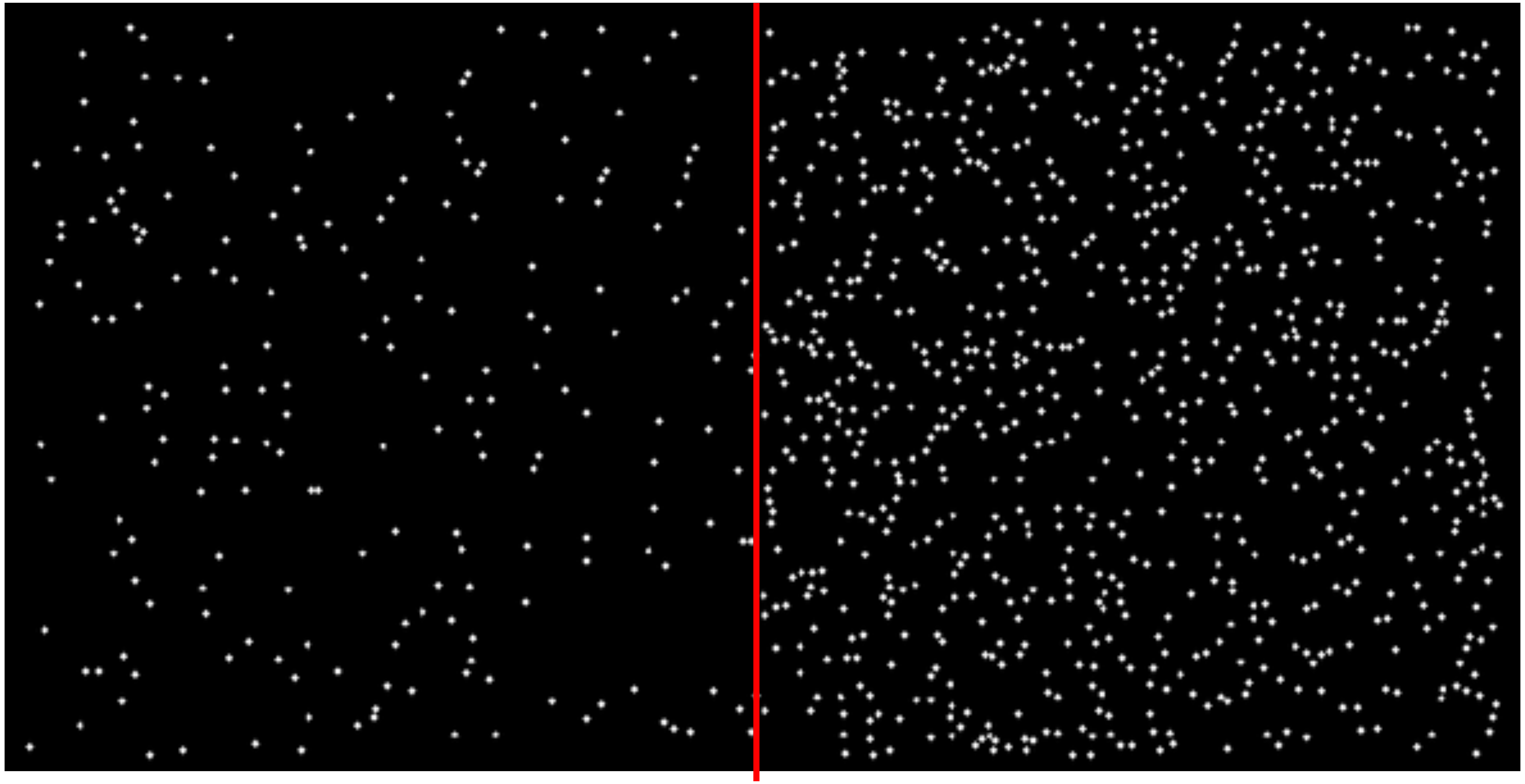
Somewhat imprecise (and later proved wrong)



Béla Julesz

Source: A. Efros

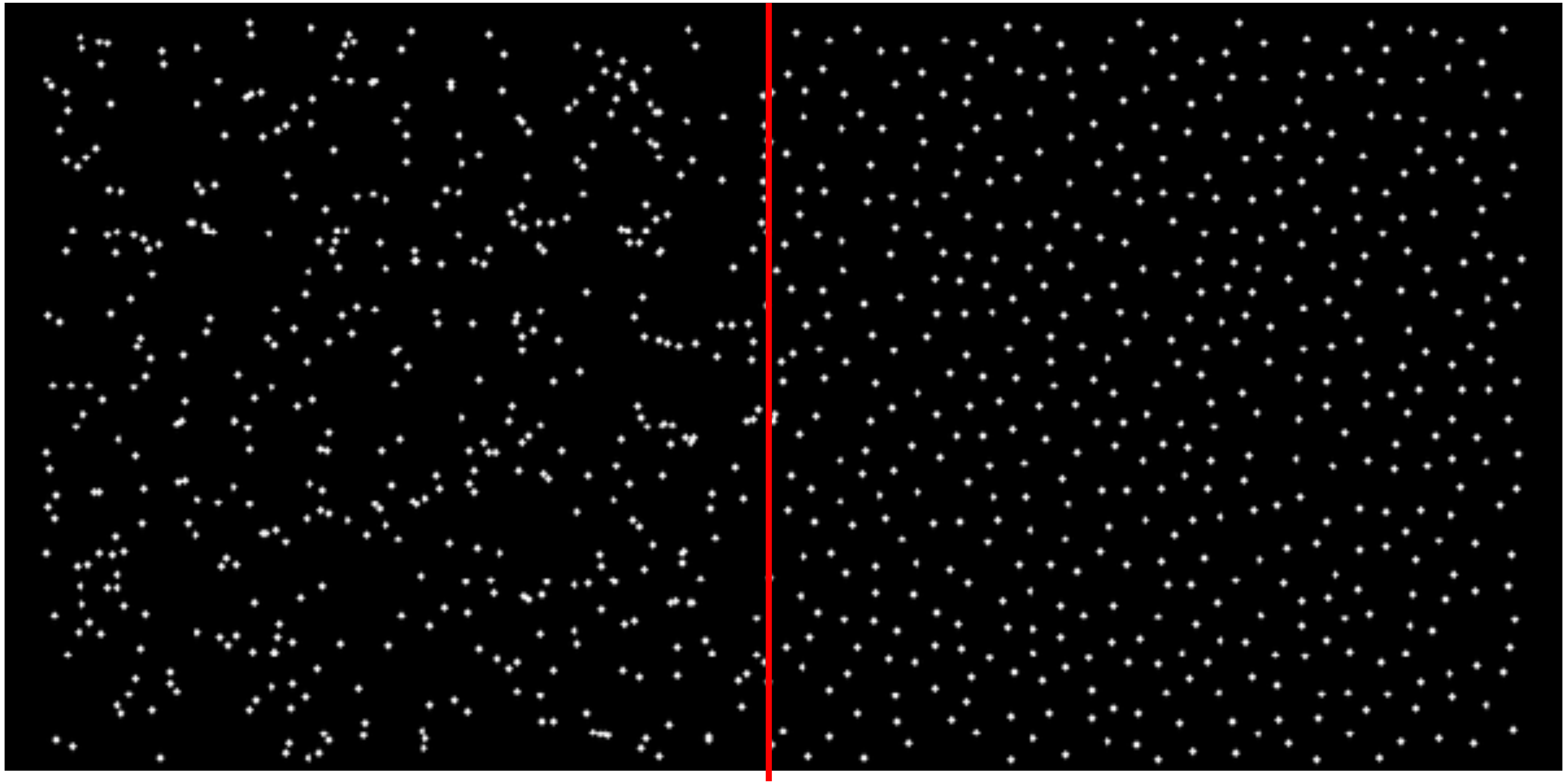
1^{st} order statistics differ



5% white 20% white

Source: A. Efros

2^{nd} order statistics differ

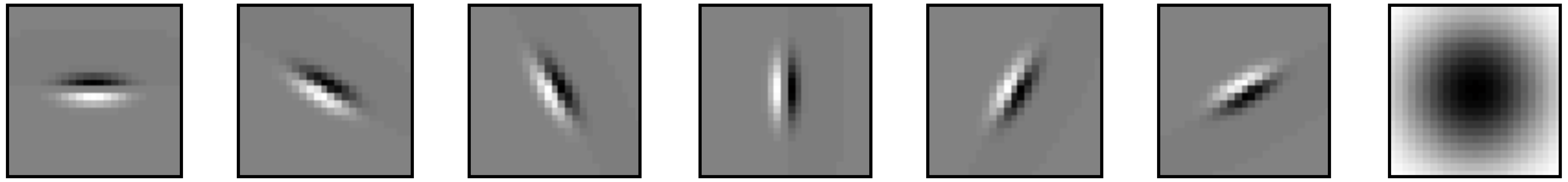


10% white

Source: A. Efros

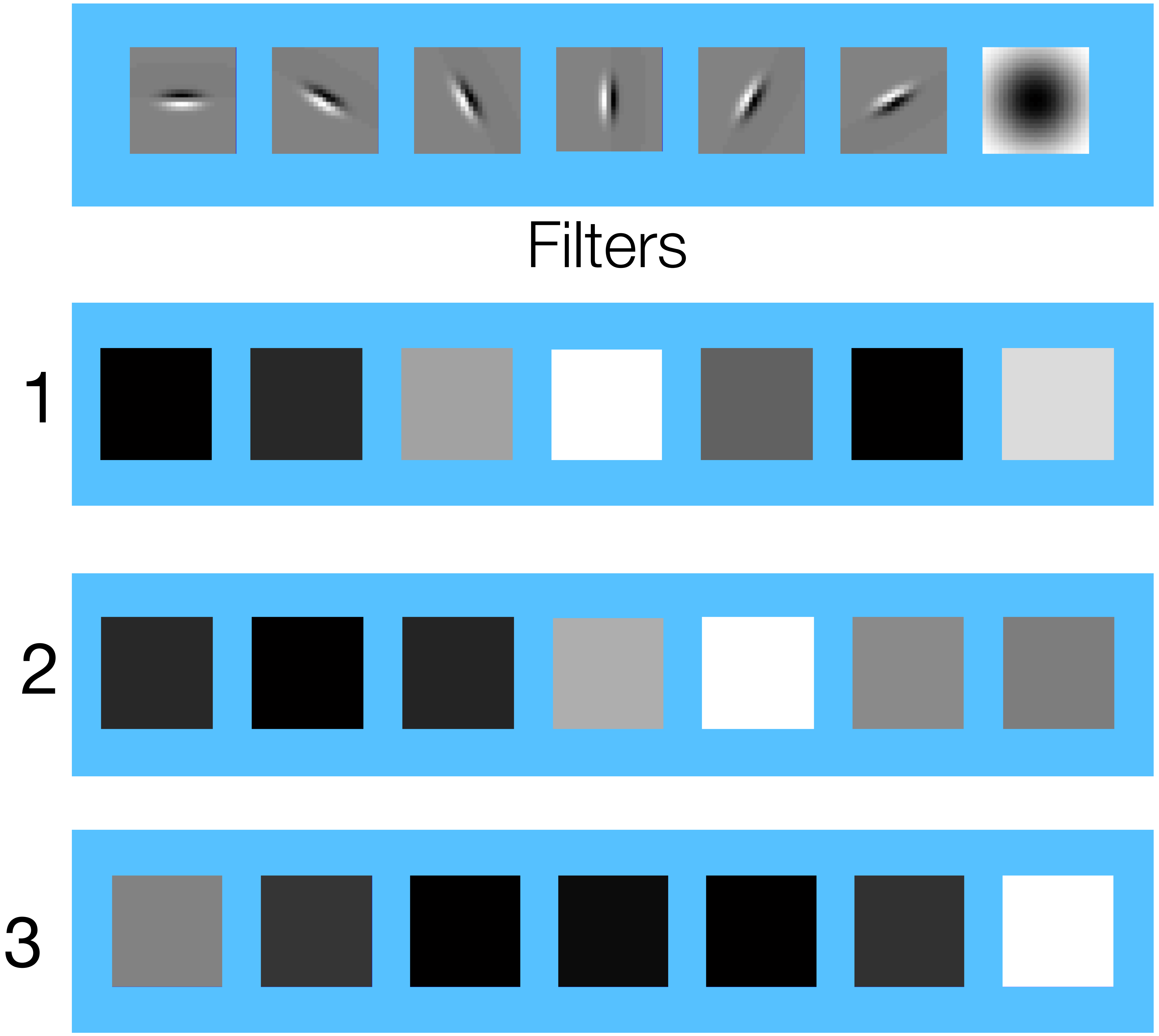
How can we represent texture in natural images?

Idea #1: Record simple statistics (e.g., mean, standard deviation) of absolute filter responses

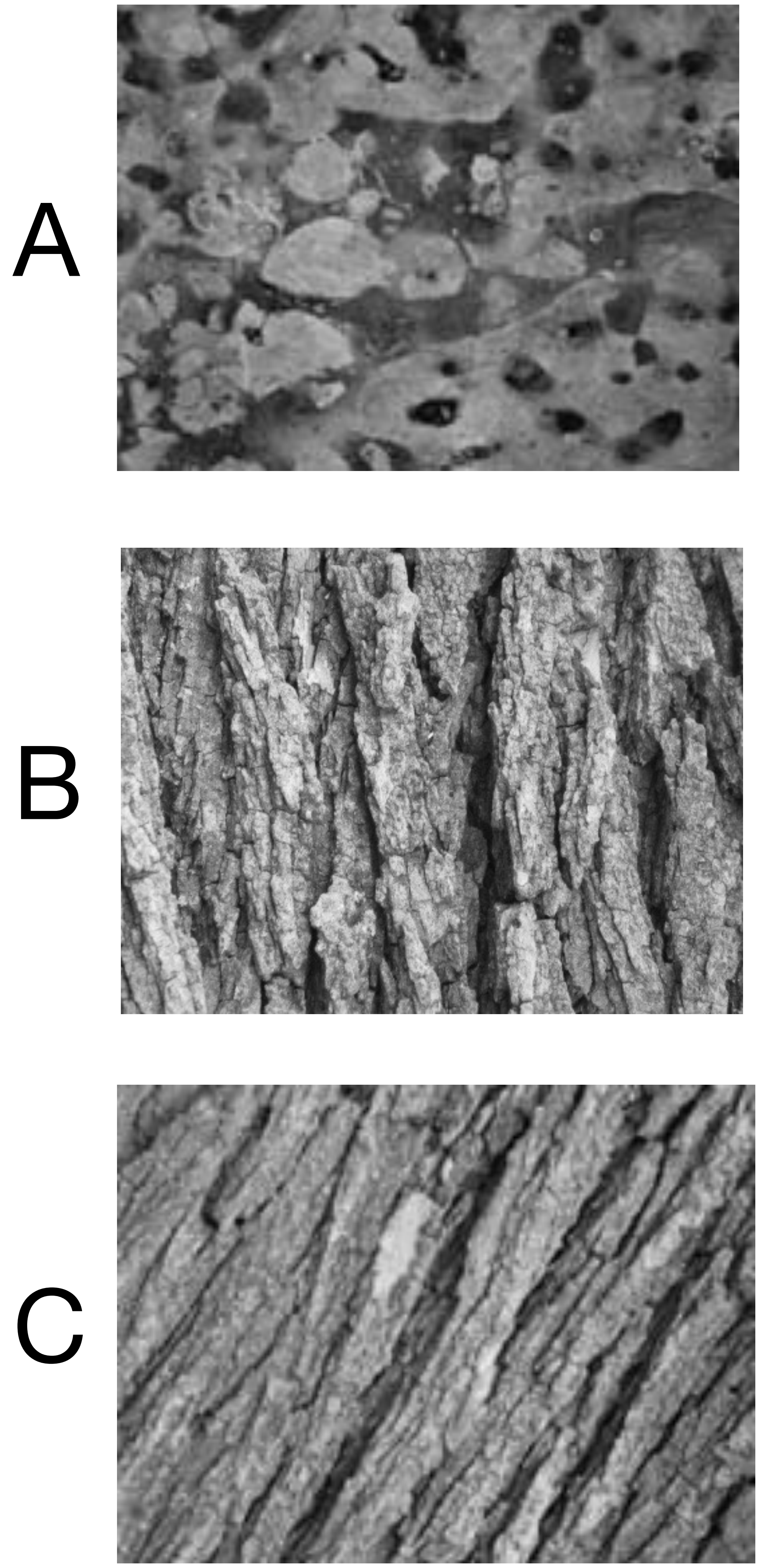


Source: A. Efros

Can you match the texture to the response?



Mean abs. responses



Source: A. Efros

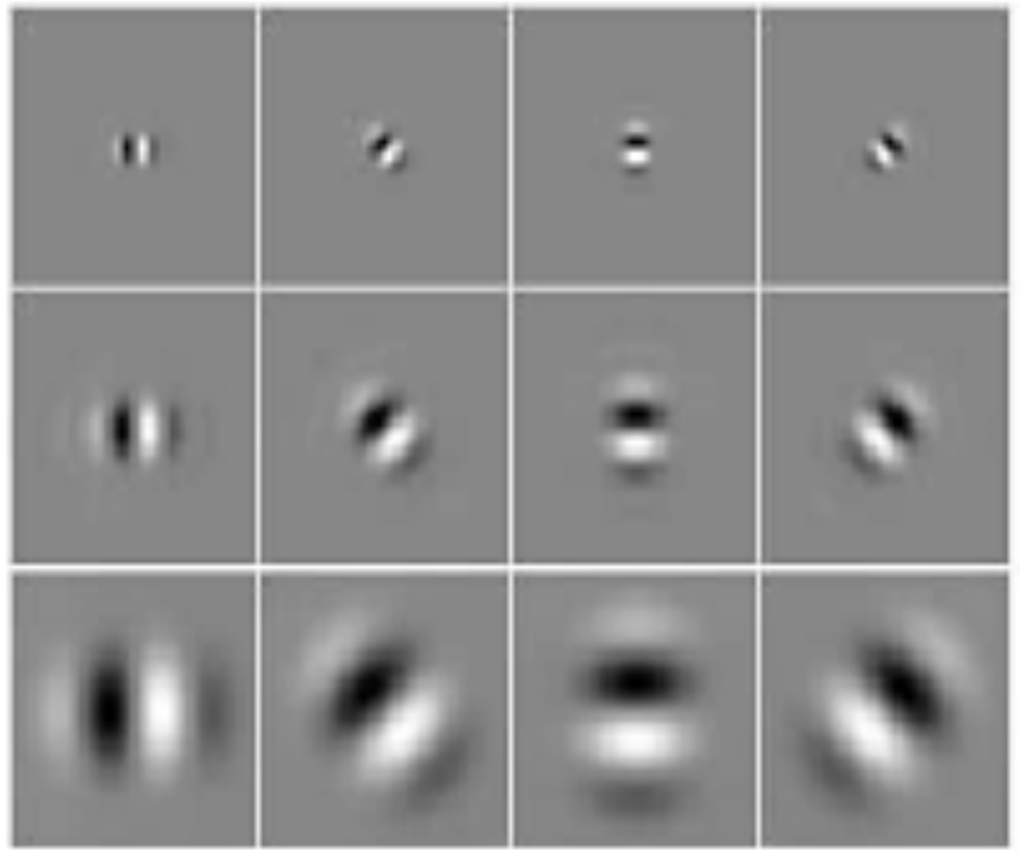
How can we represent texture?

- Generalize this to “orientation histogram”
- Idea #2: Histograms of filter responses
  - One histogram per filter

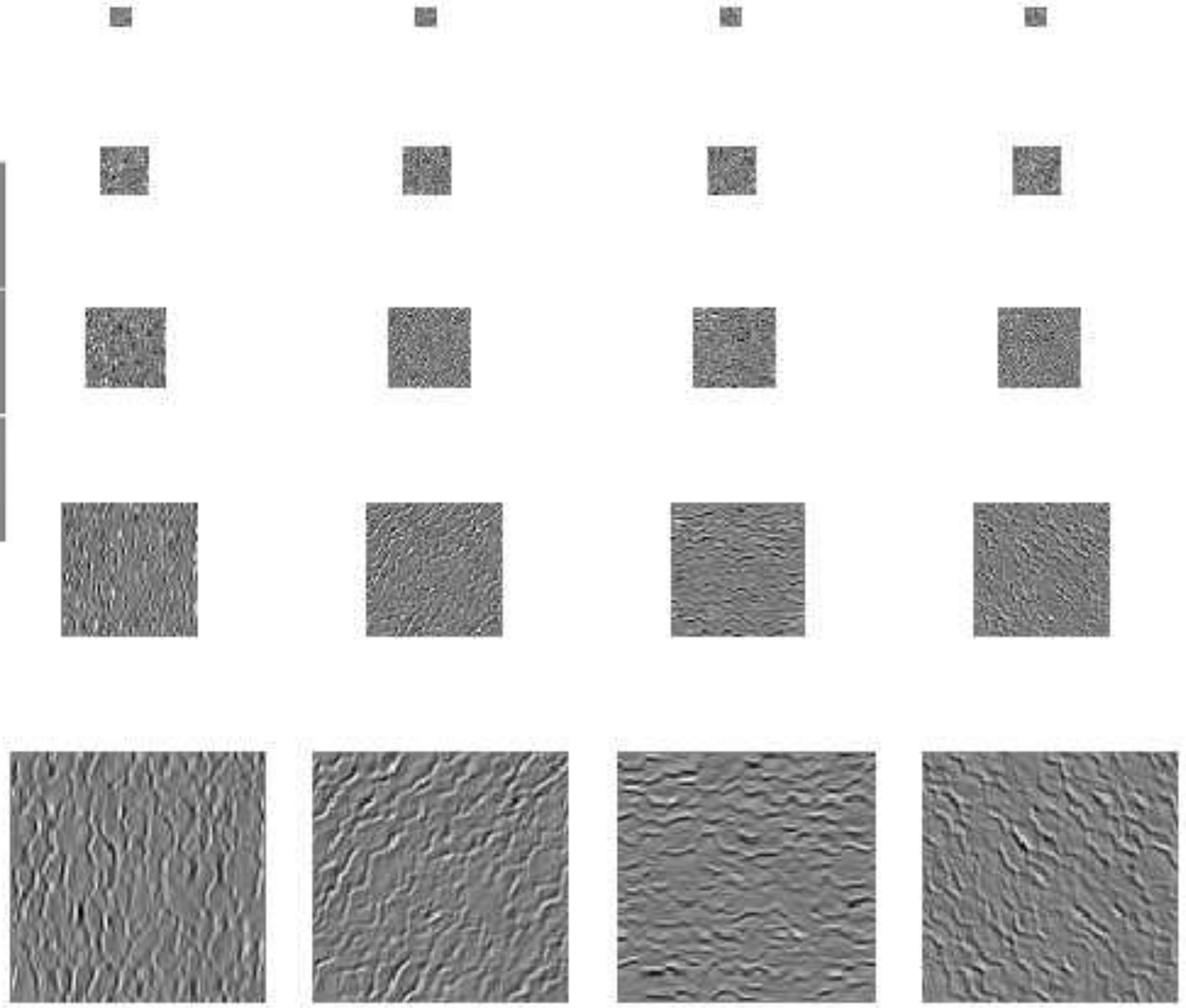
Source: A. Efros

Steerable pyramid decomposition

Filter bank

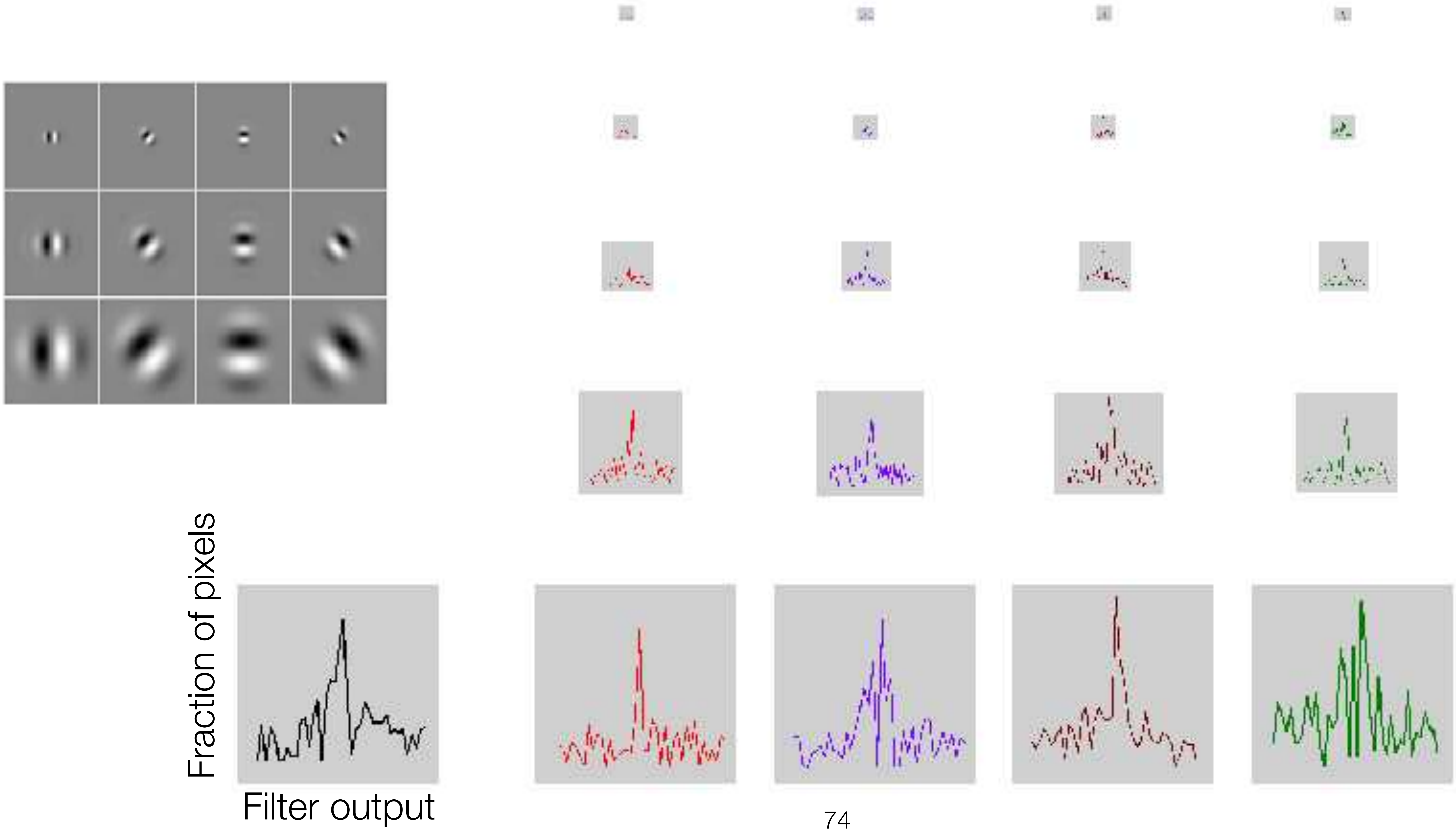


Input image



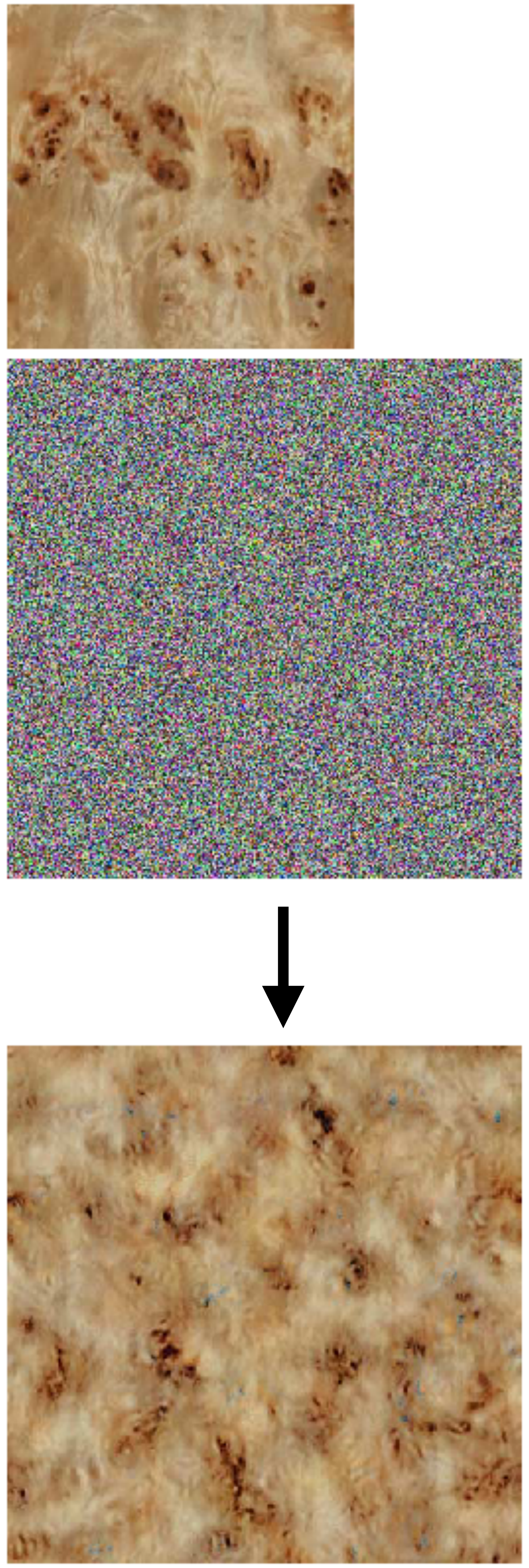
Source: A. Efros

Filter response histograms



Source: A. Efros

Texture synthesis



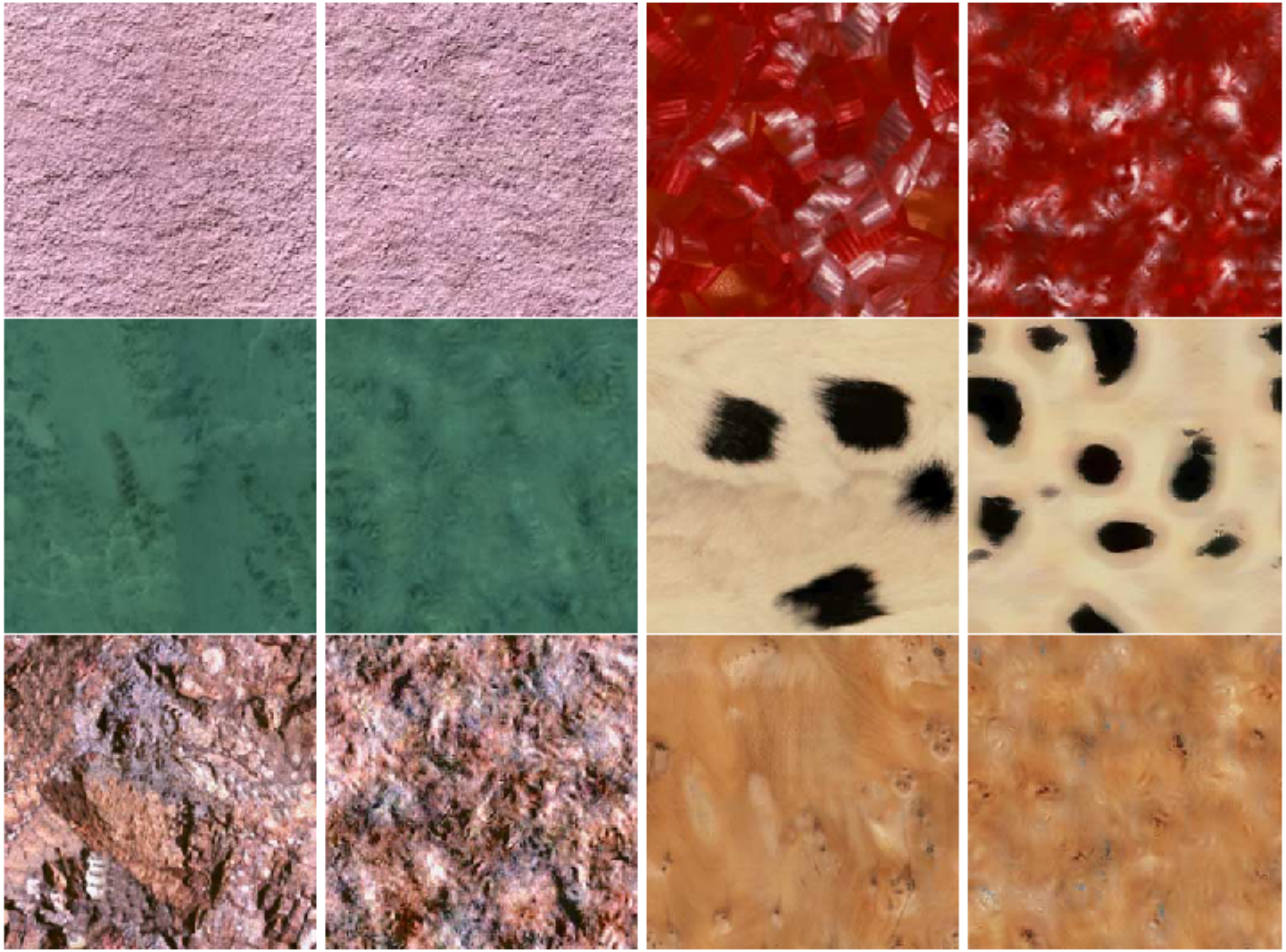
Start with a noise image as output.

Iterative algorithm [Heeger & Bergen, 95]:

- Match pixel histogram of output image to input
- Decompose input/output images using a Steerable Pyramid
- Match histograms of input and output pyramids
- Reconstruct image and repeat

Later in the class we'll see a simpler optimization method on neural nets [Gatys et al. 2015]

Source: A. Efros



Source: A. Efros

Source: A. Efros

Failure cases

Figure 7: (Left pair) Inhomogeneous input texture produces blotchy synthetic texture. (Right pair) Homogenous input.

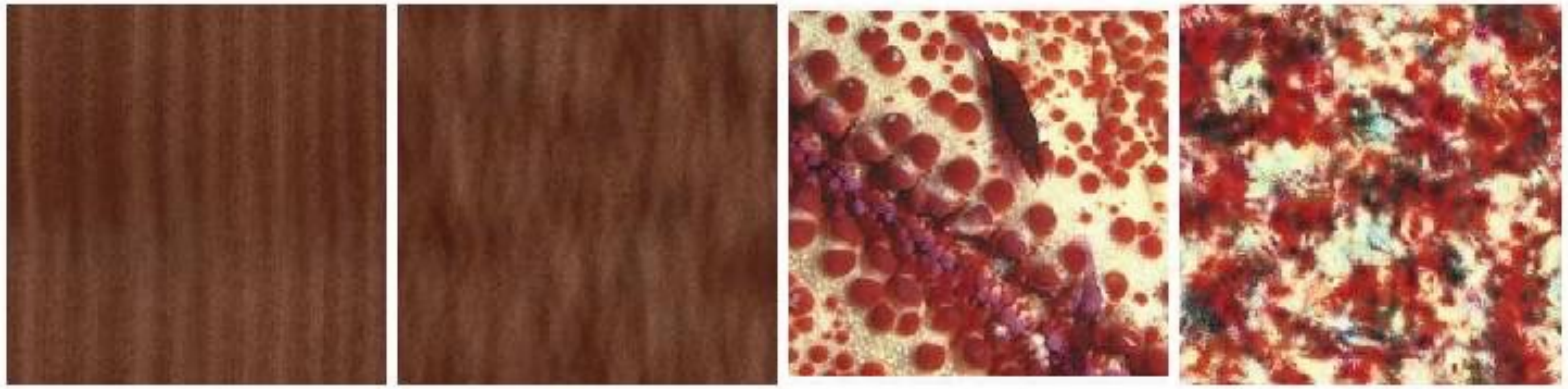


Figure 8: Examples of failures: wood grain and red coral.

Figure 9: More failures: hay and marble.

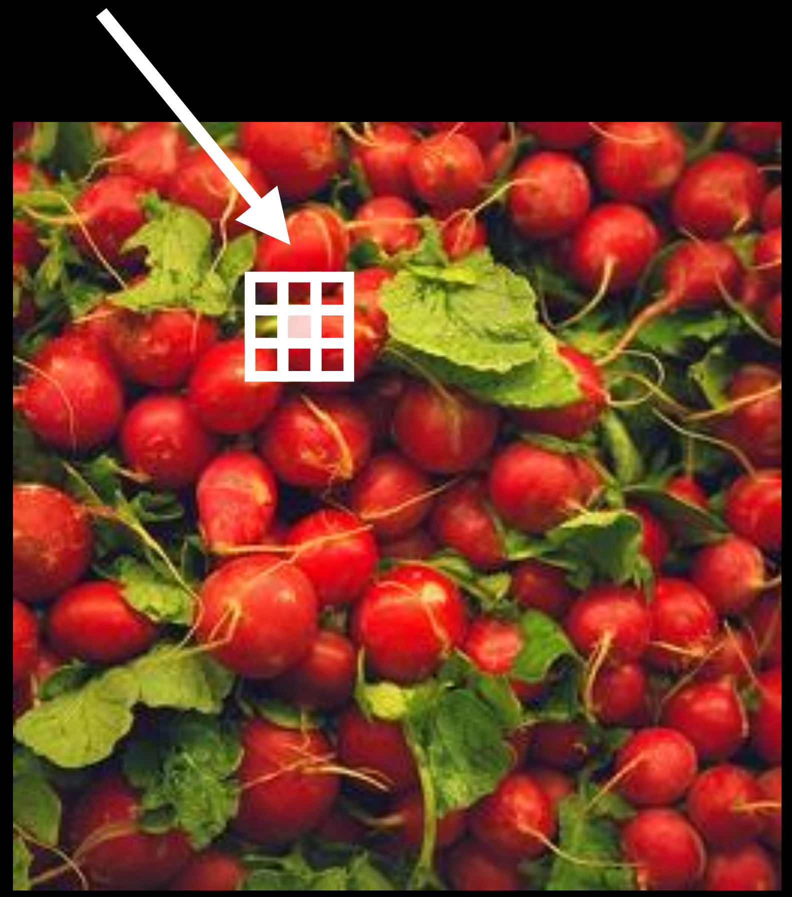
Source: A. Efros

Nonparametric texture synthesis:
who needs pyramids or filters?

Modeling local neighborhoods

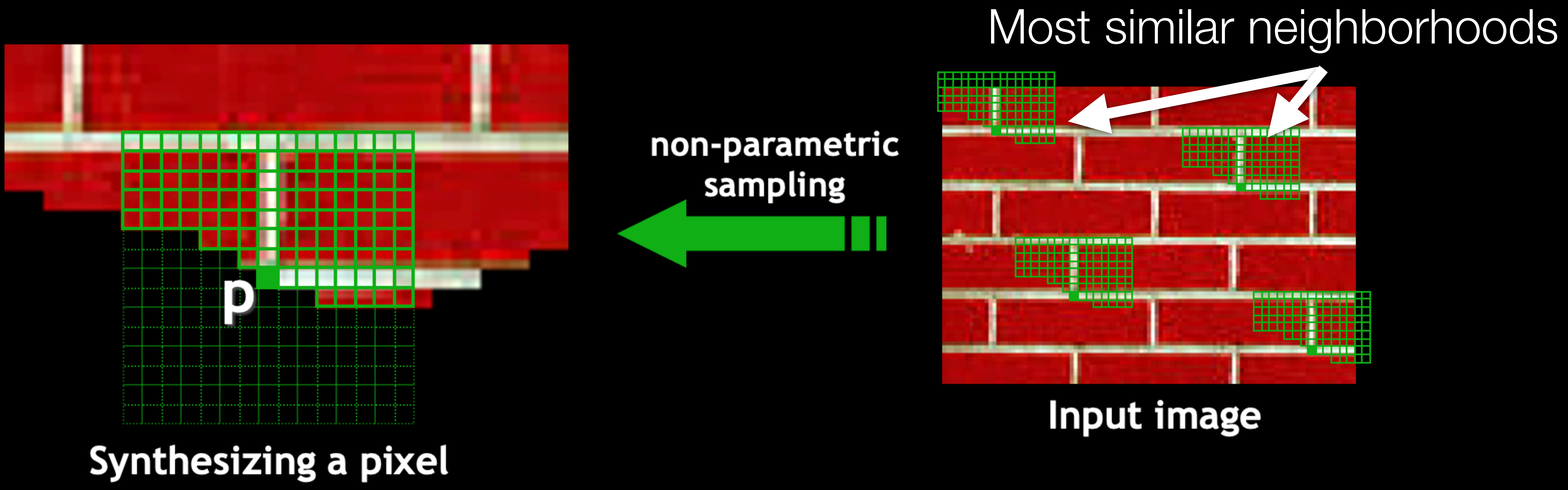
Model $p(\mathbf{p} \,|\, N(\mathbf{p}))$,

Probability of pixel given its neighbors



Source: A. Efros

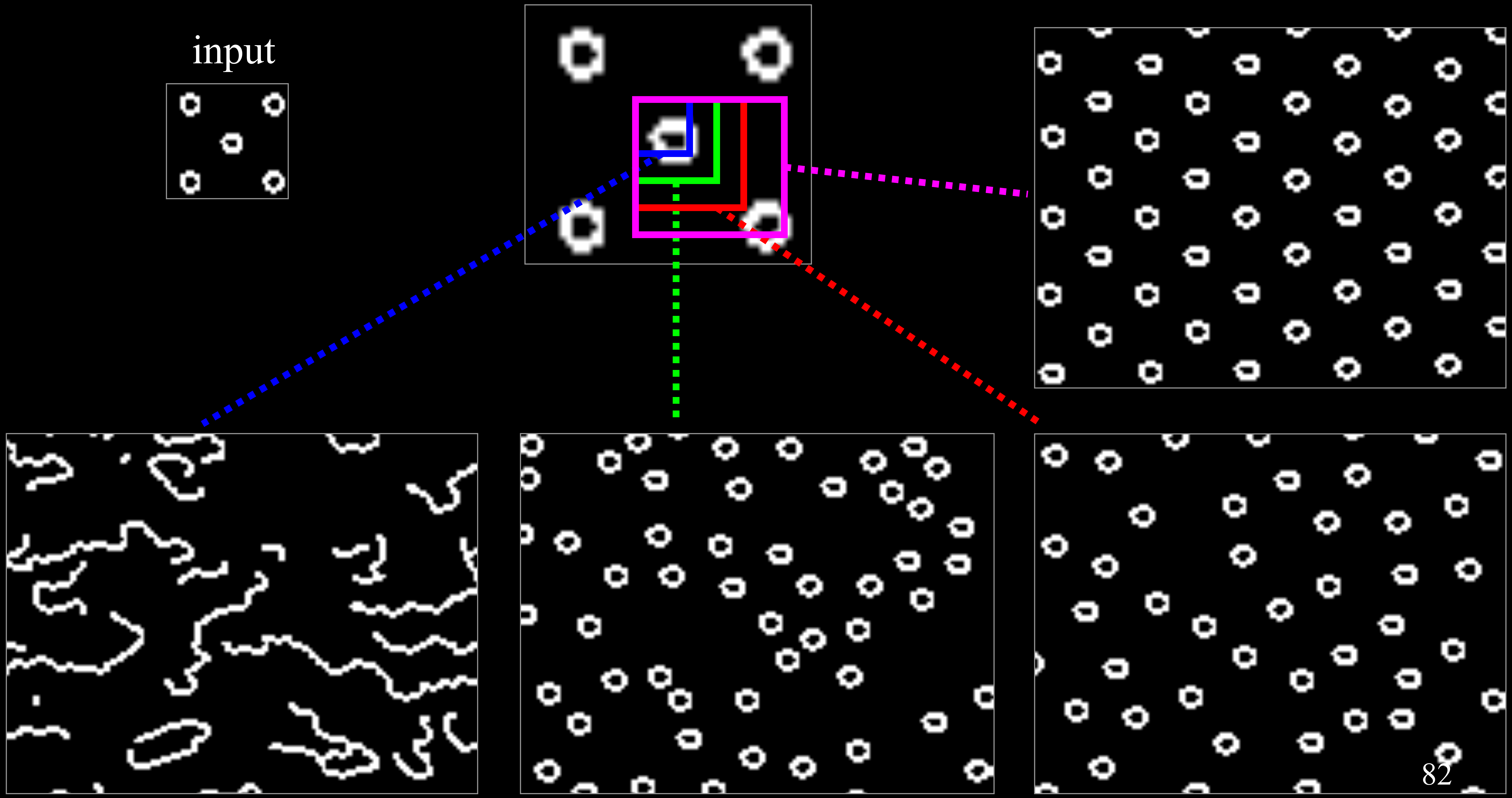
Efros & Leung Algorithm



- Synthesize one pixel at a time. Want to sample: P(**p**|N(**p**)), where N(p) are the already filled-in neighbors
  - Building explicit probability tables is hard
- Instead, we *search the input image* for all similar neighborhoods — that's our distribution for **p**
- To sample from this distribution, just pick one match at random

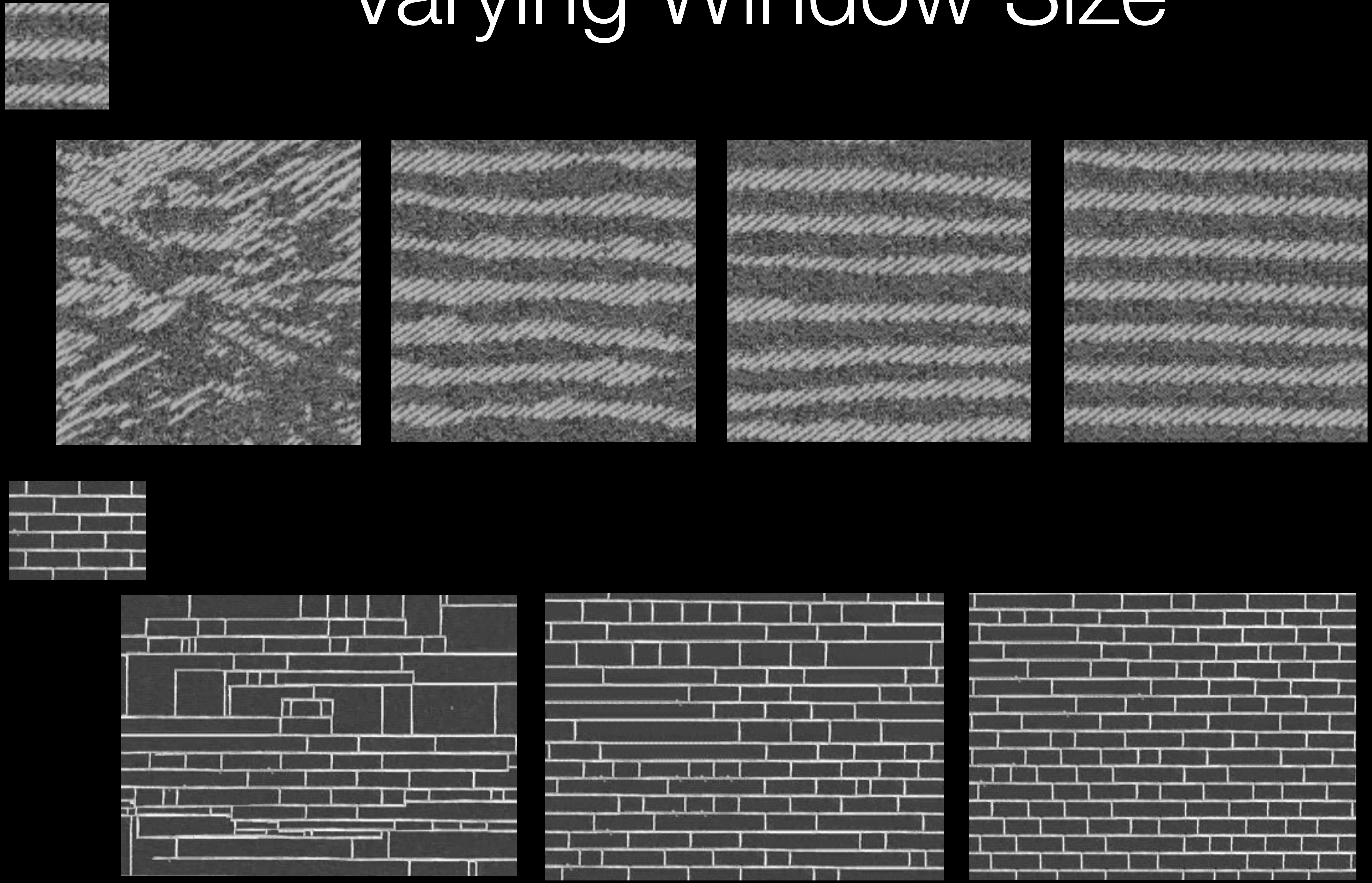
Source: A. Efros

Neighborhood Window



Source: A. Efros

Varying Window Size

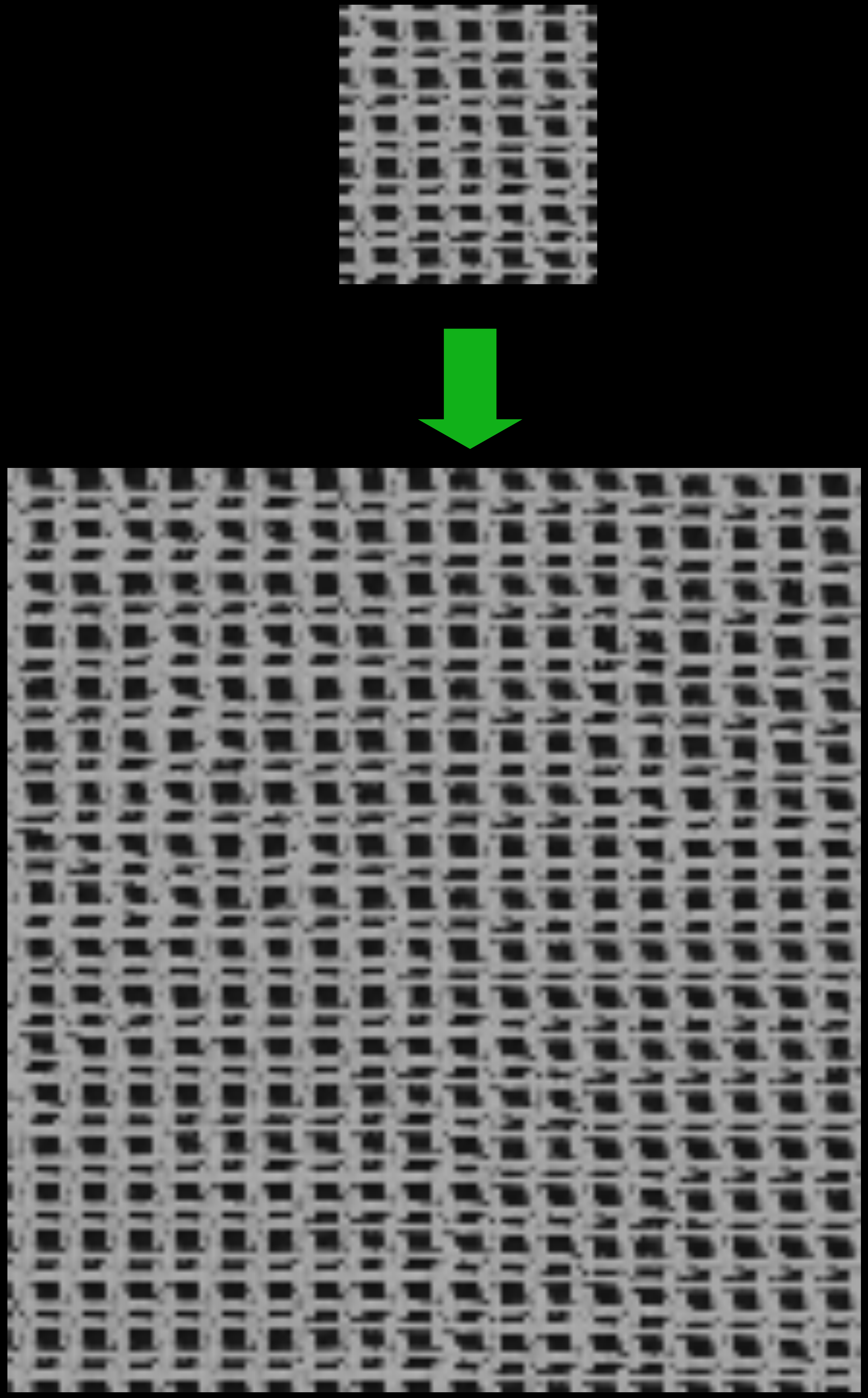


Increasing window size

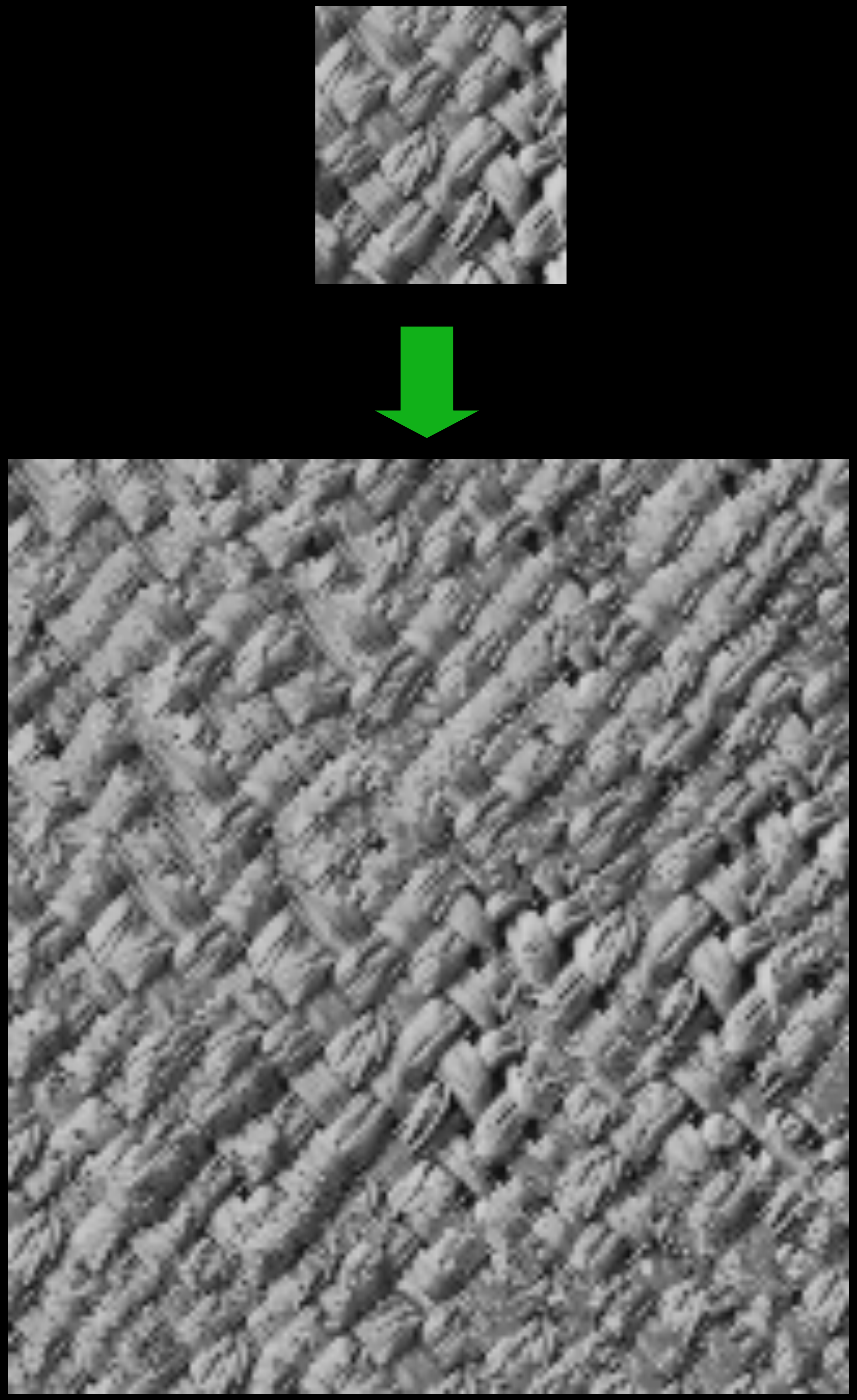
Source: A. Efros

Synthesis Results

french canvas



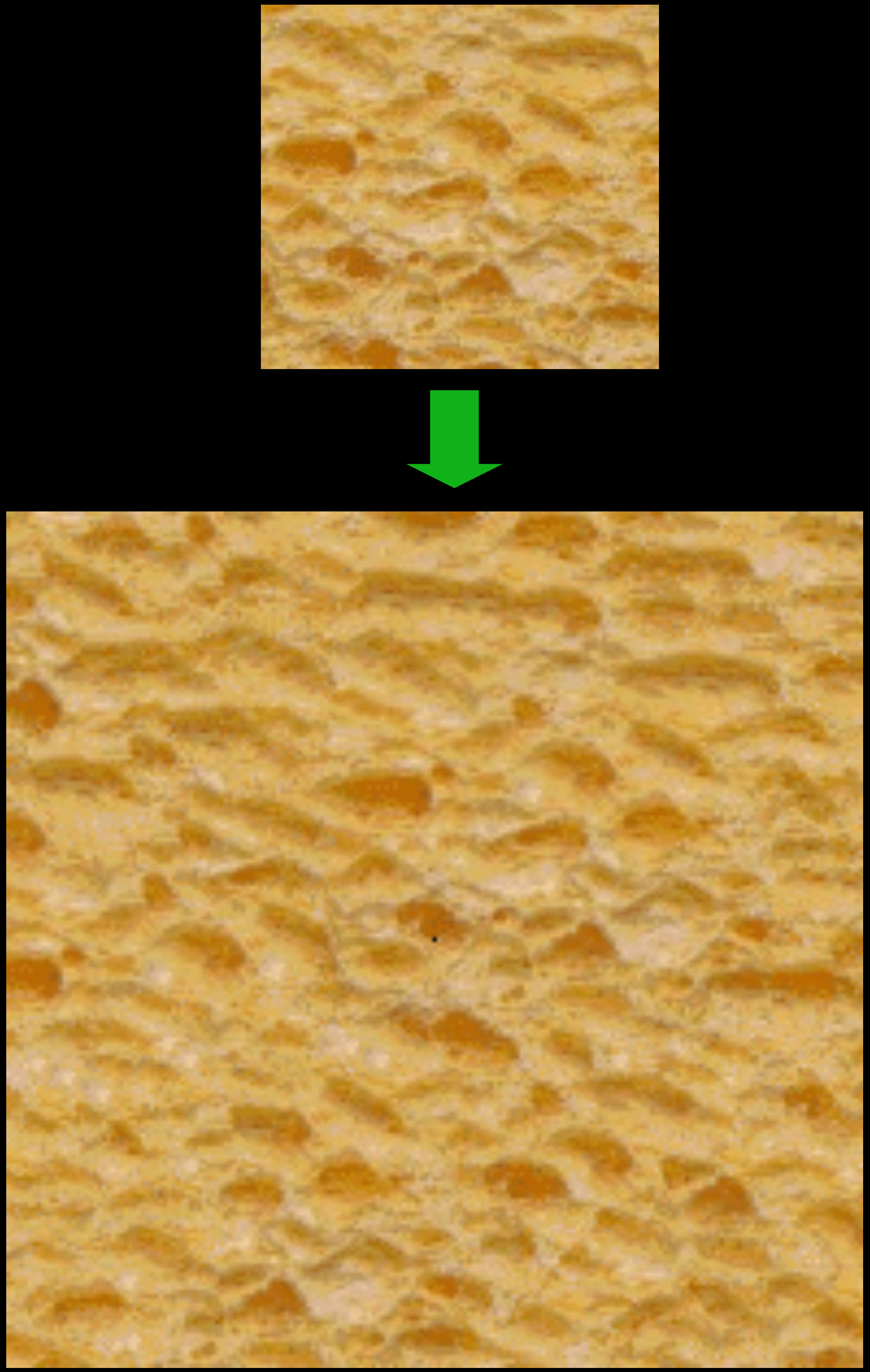
rafia weave



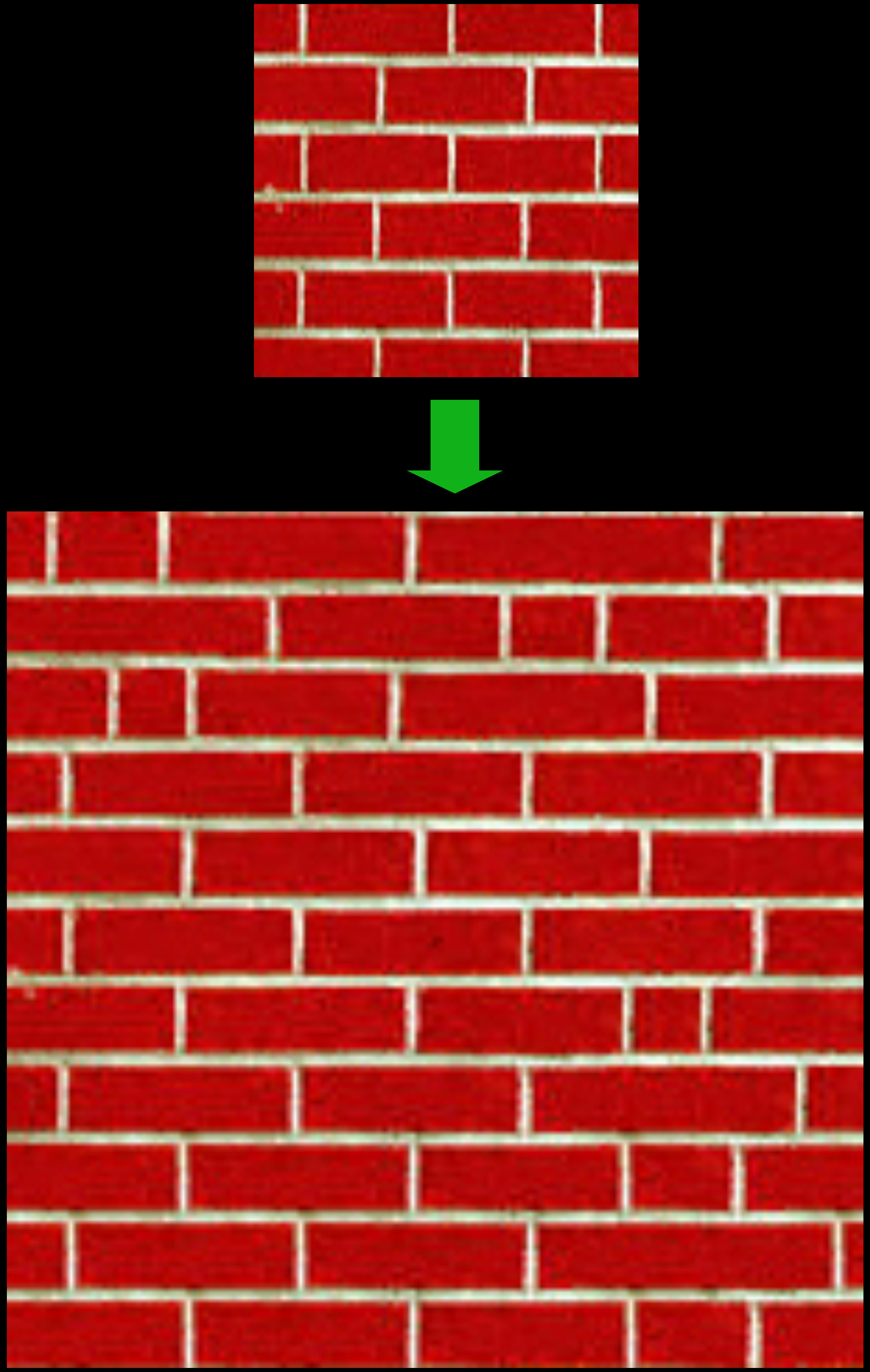
Source: A. Efros

More Results

white bread

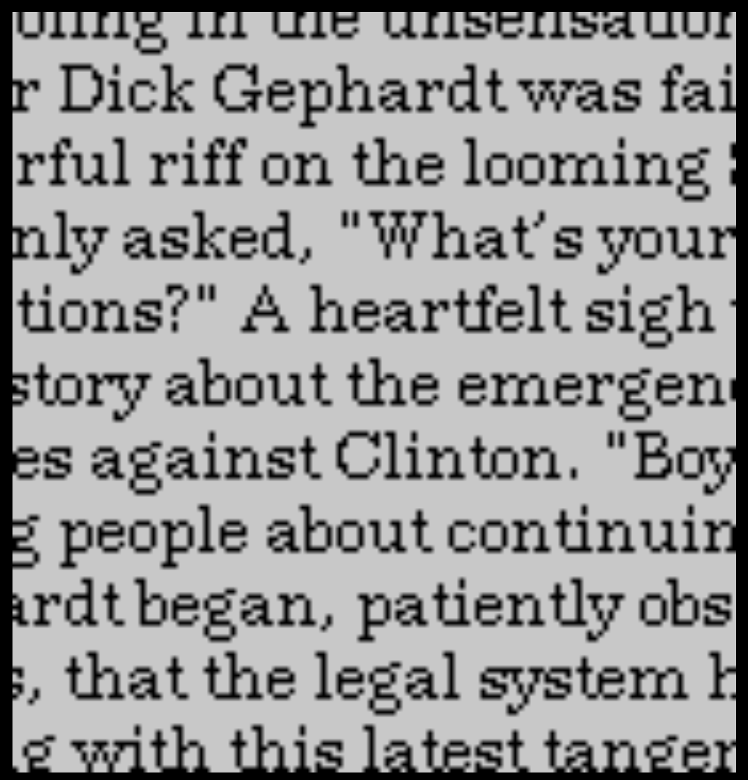


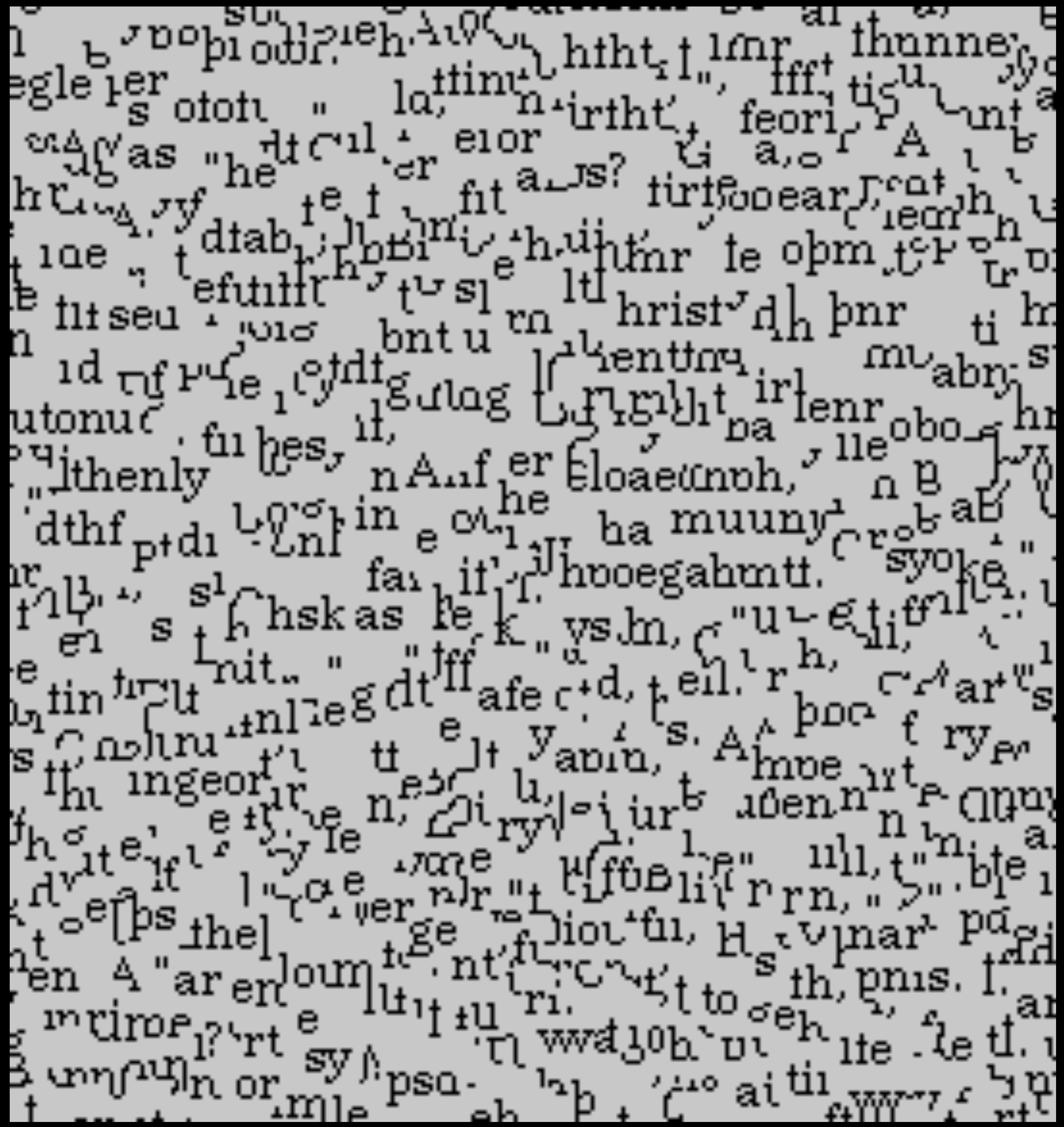
brick wall

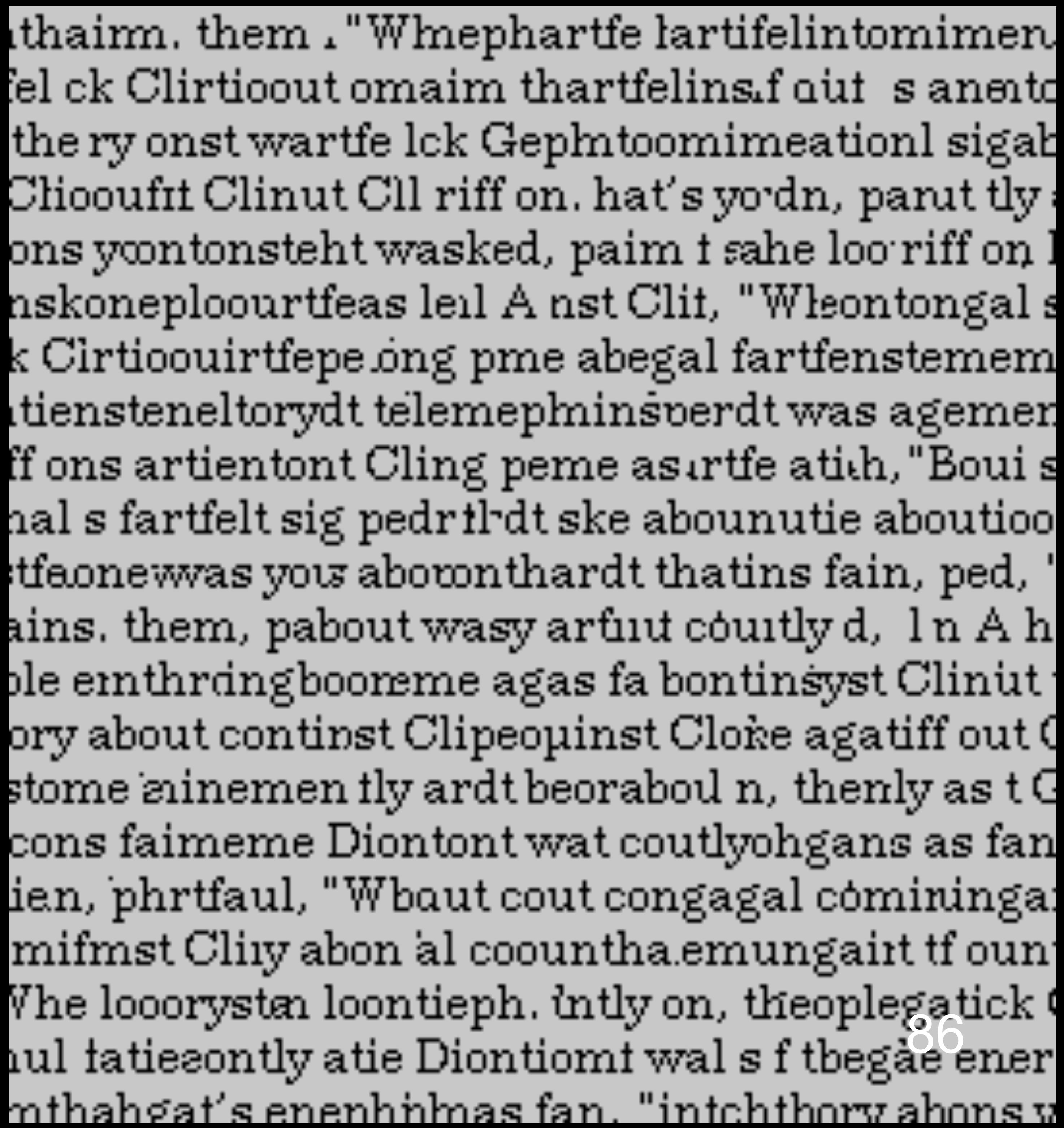


Source: A. Efros

Homage to Shannon

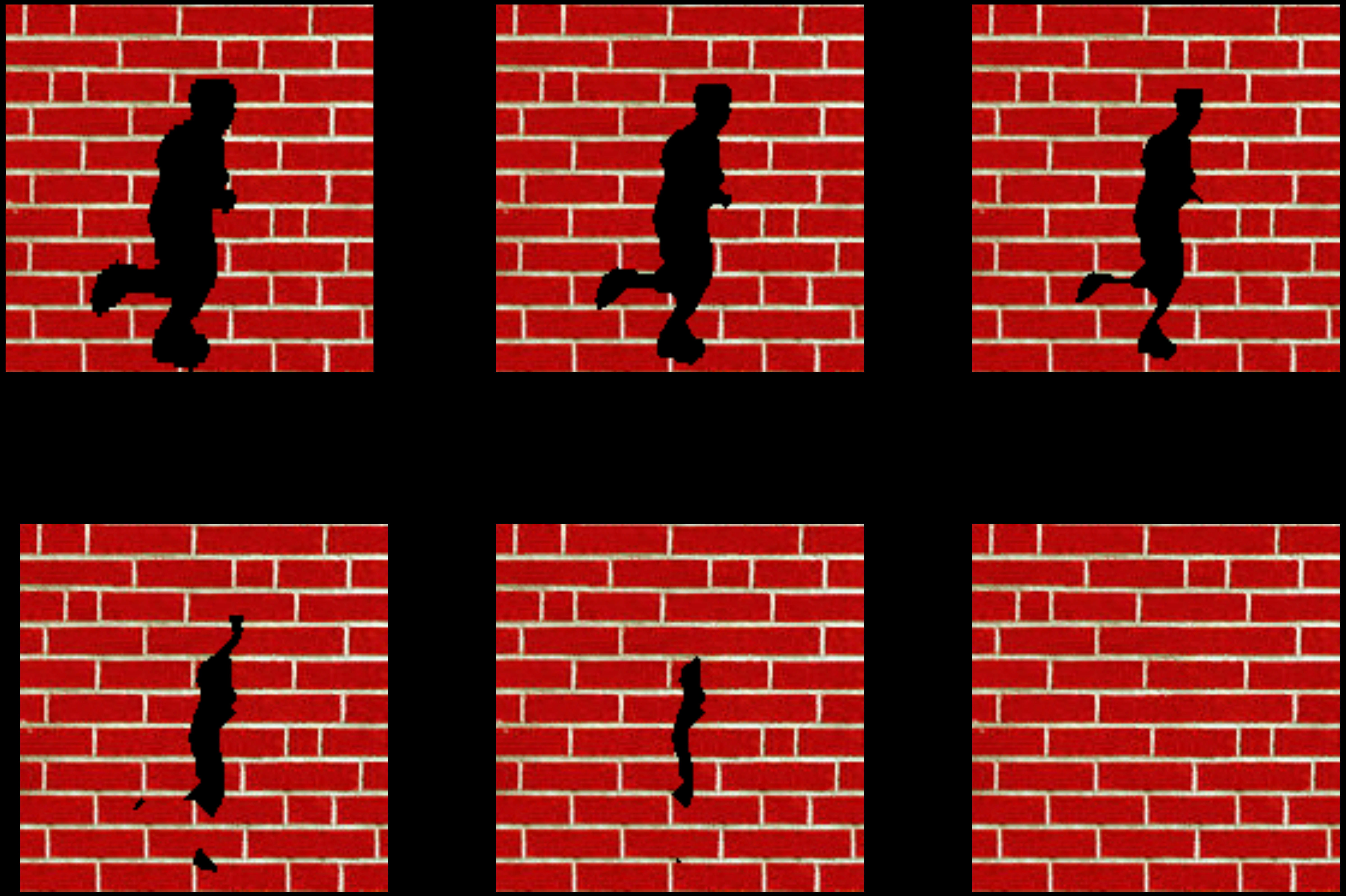






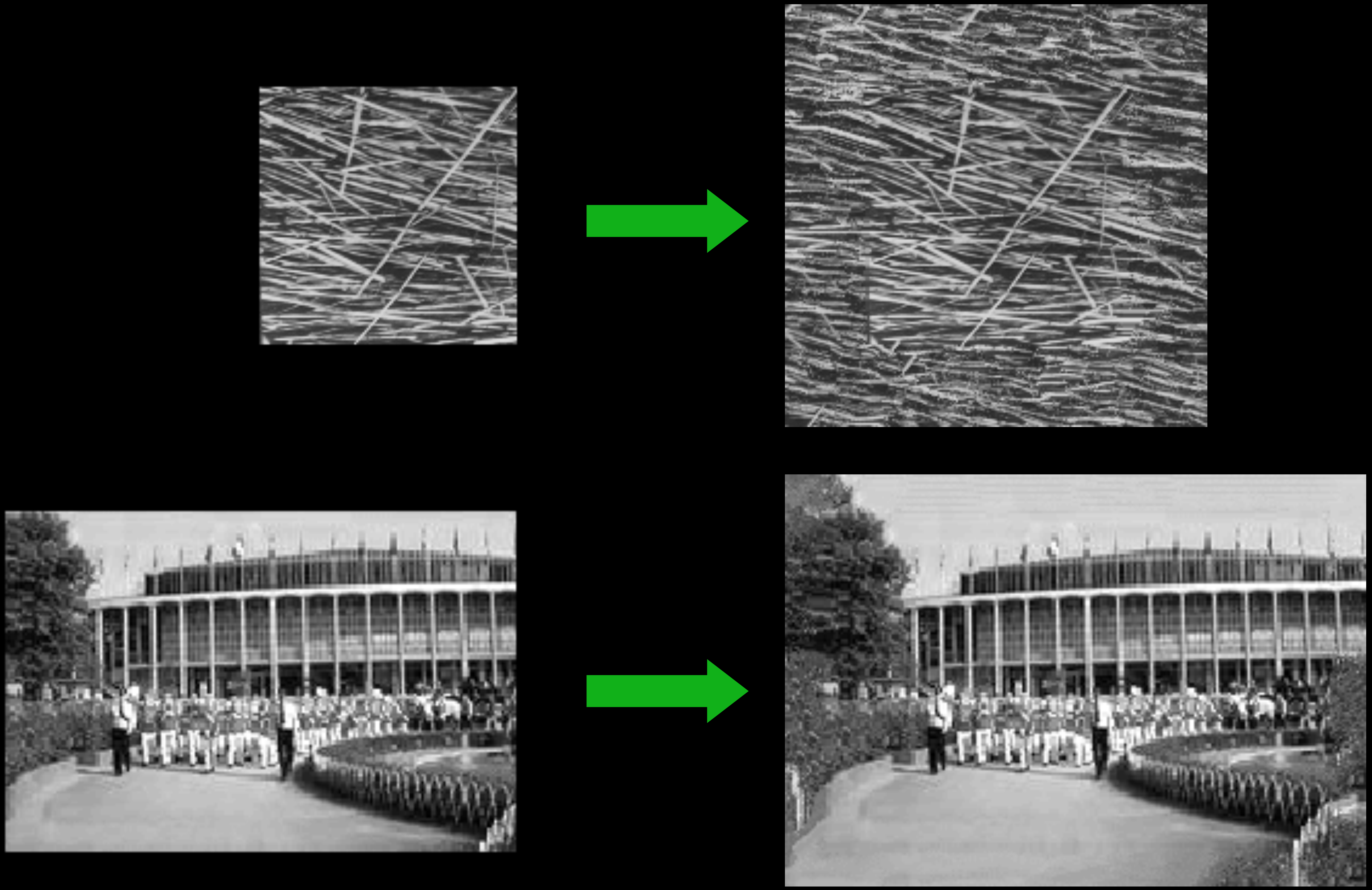
Source: A. Efros

Hole Filling (a.k.a. Inpainting)



Source: A. Efros

Extrapolation



Source: A. Efros

Today

- Image pyramids
- Image statistics
- Texture synthesis