MIT CSAIL

6.869: Advances in Computer Vision

Bill Freeman, Antonio Torralba, and Phillip Isola
Oct. 11, 2018

MIT COMPUTER VISION

Lecture 10

Image generation, statistical models, and noise removal

The visual system seems to be tuned to a set of images:

Demo inspired from D. Field

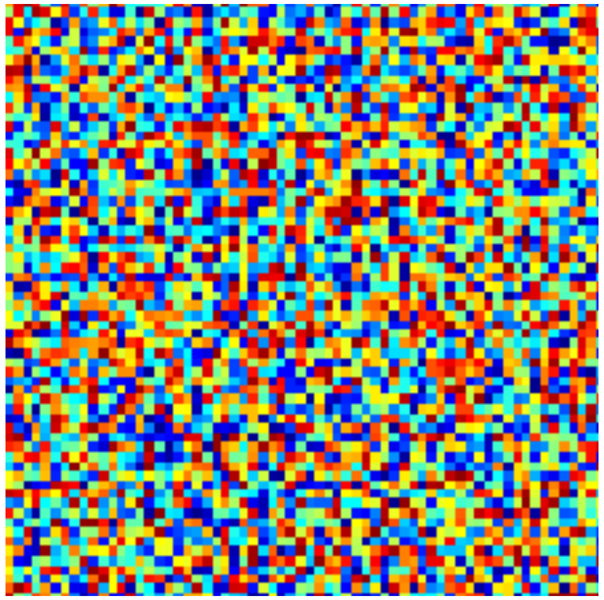
Remember these images

Did you see this image?

Remember these images

Test 2

Did you see this image?



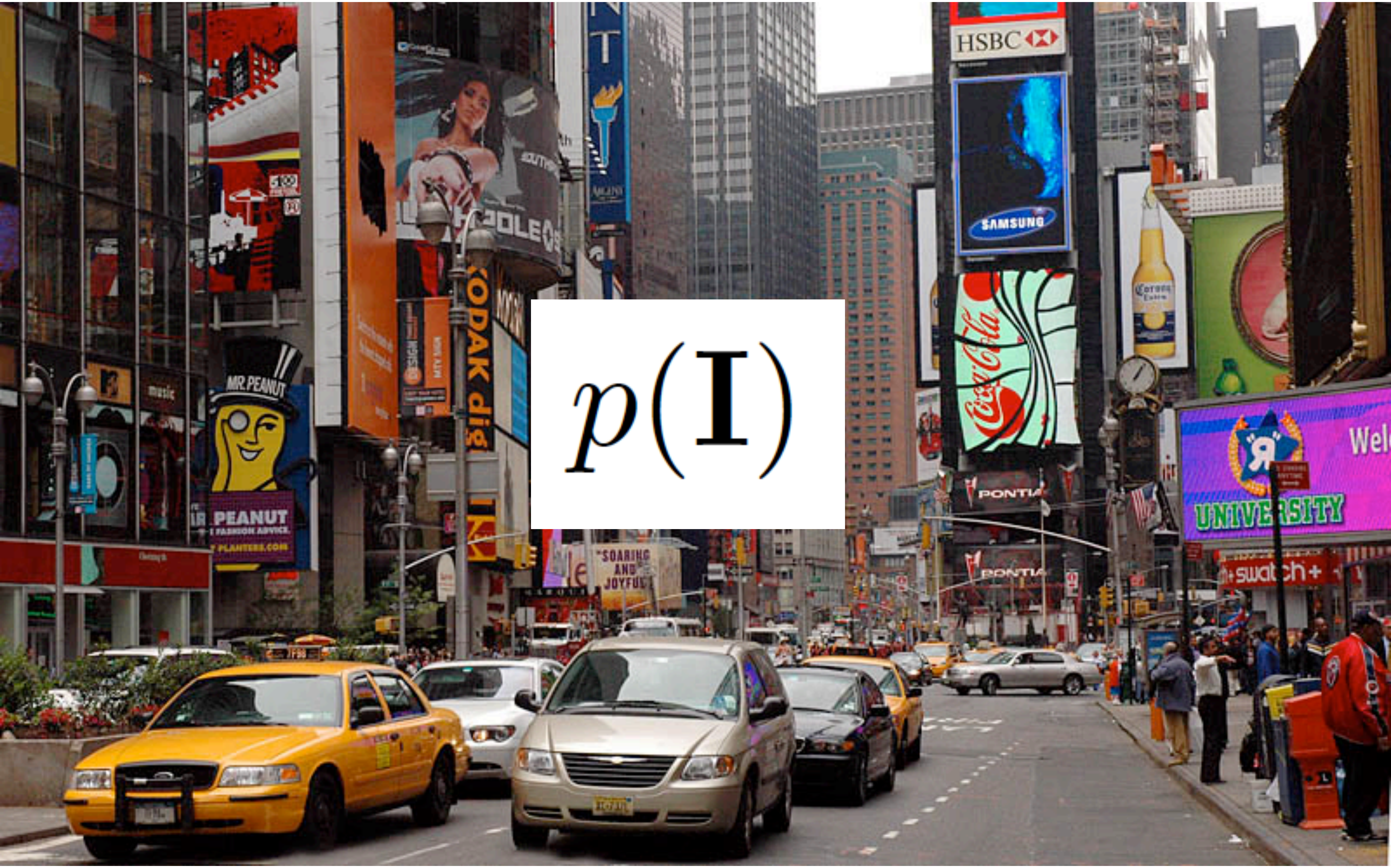
The visual system is tuned to process structures typically found in the world.

Today’s lecture:

3 image models, and

3 corresponding noise removal algorithms

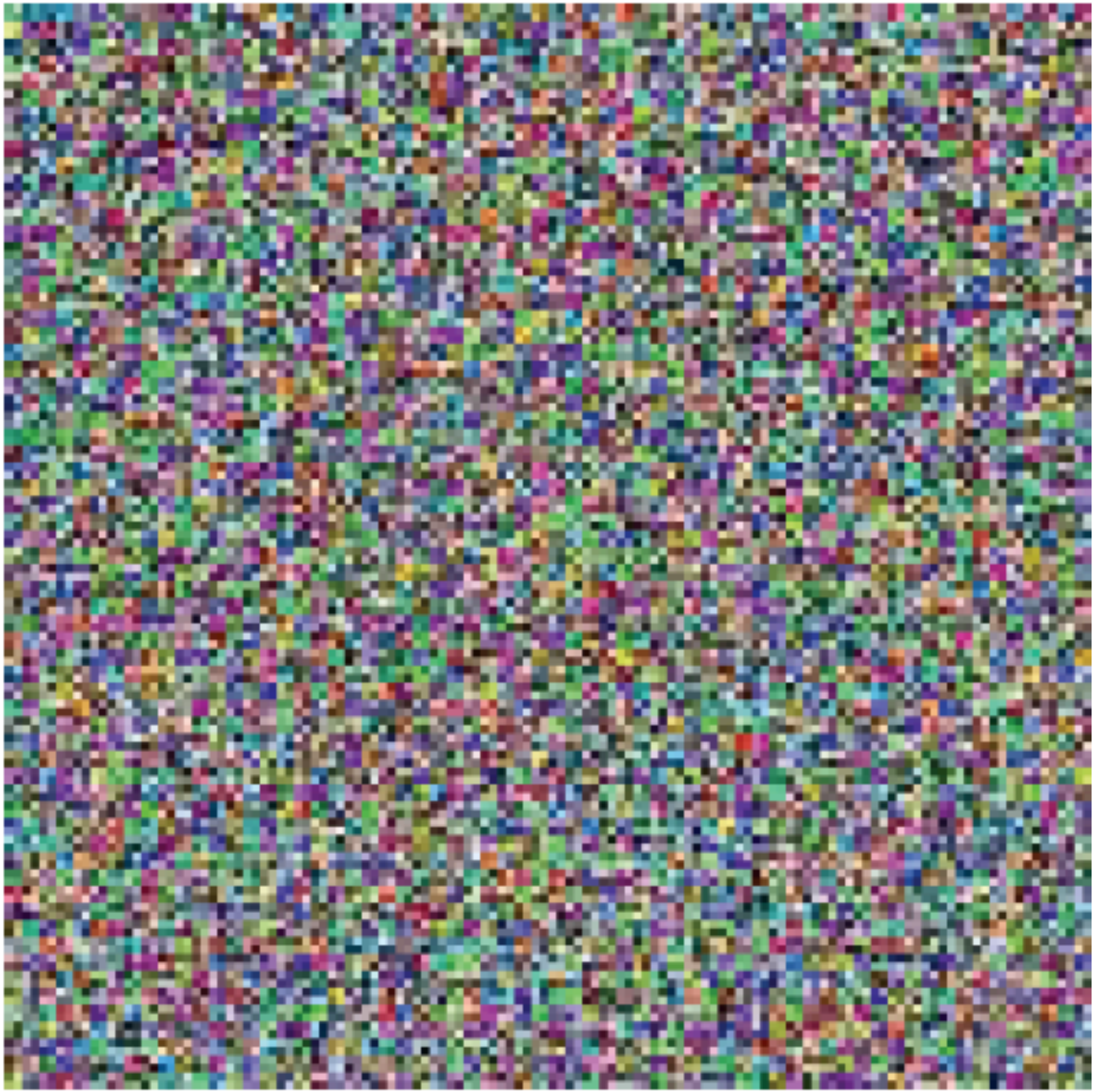
Statistical modeling of images



$$p(\mathbf{I})$$

Visual Worlds

Visual Worlds



Visual Worlds

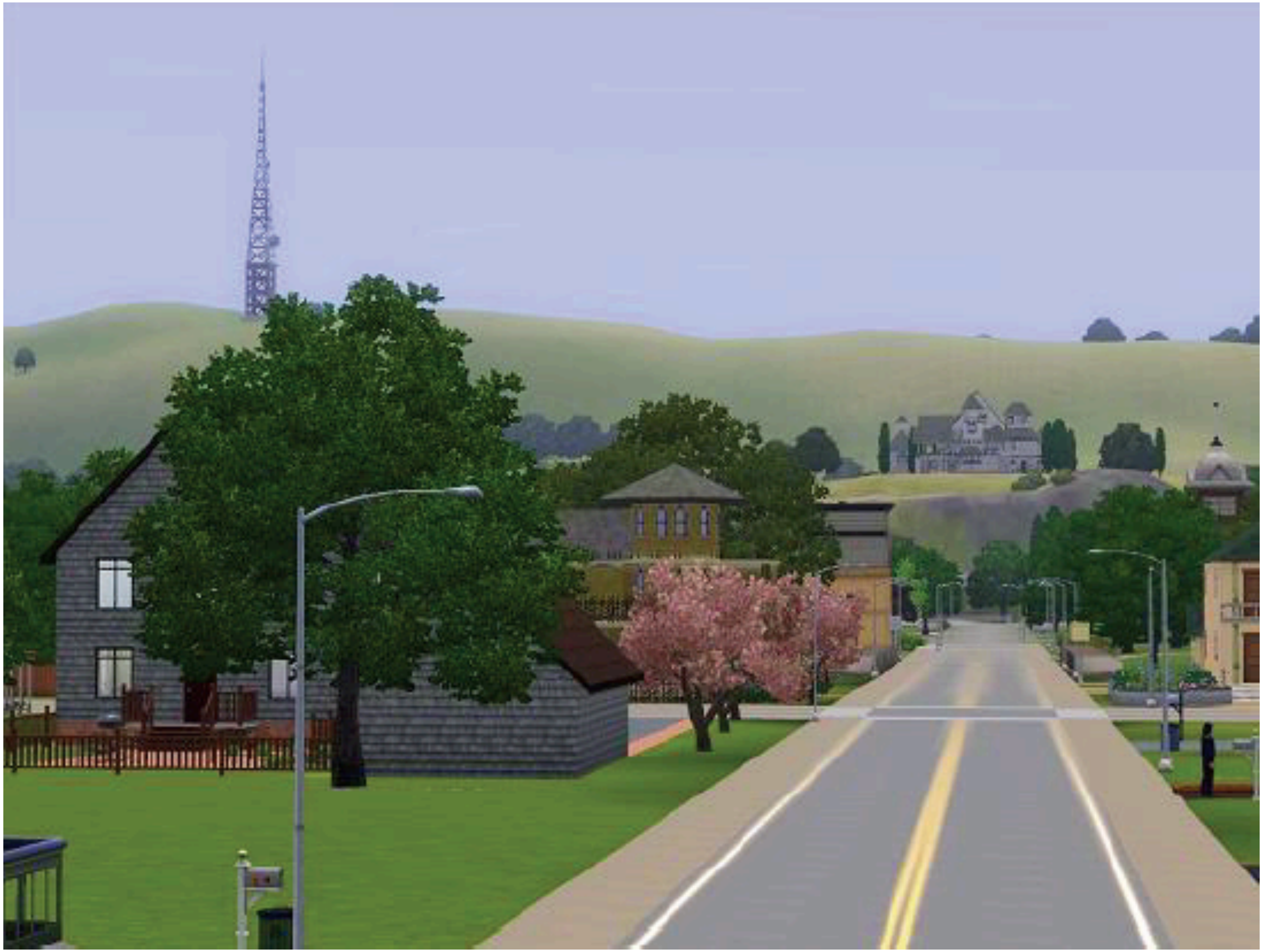
Visual Worlds

Visual Worlds

Visual Worlds

Visual Worlds

Visual Worlds



Visual Worlds

To appear in: Handbook of Video and Image Processing, 2nd edition
ed. Alan Bovik, ©Academic Press, 2005.

4.7 Statistical Modeling of Photographic Images

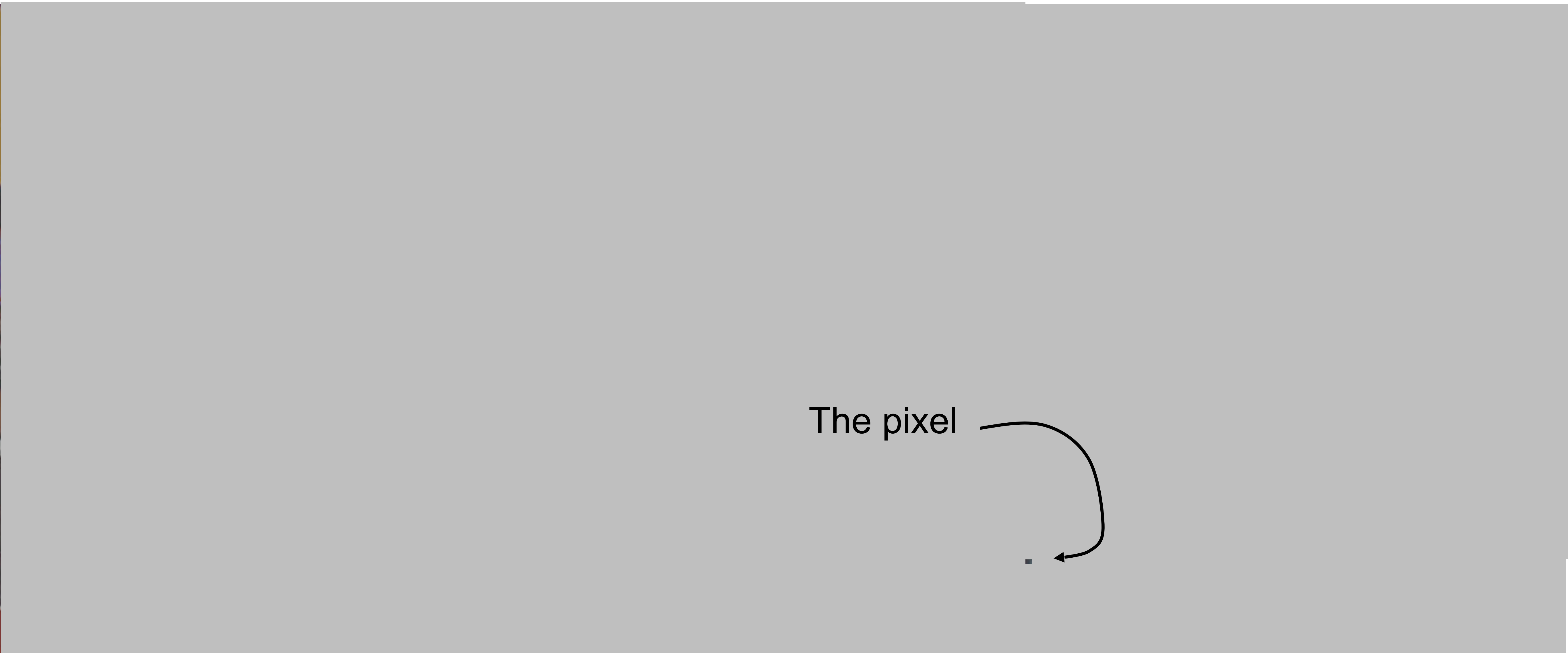
Eero P. Simoncelli

New York University

January 18, 2005

https://pdfs.semanticscholar.org/ee55/814e8705f5e8cf664efb66c31c0ea6372d92.pdf

Statistical modeling of images



$$p(\mathbf{I}) = \prod_{x,y} p(\mathbf{I}(x,y))$$

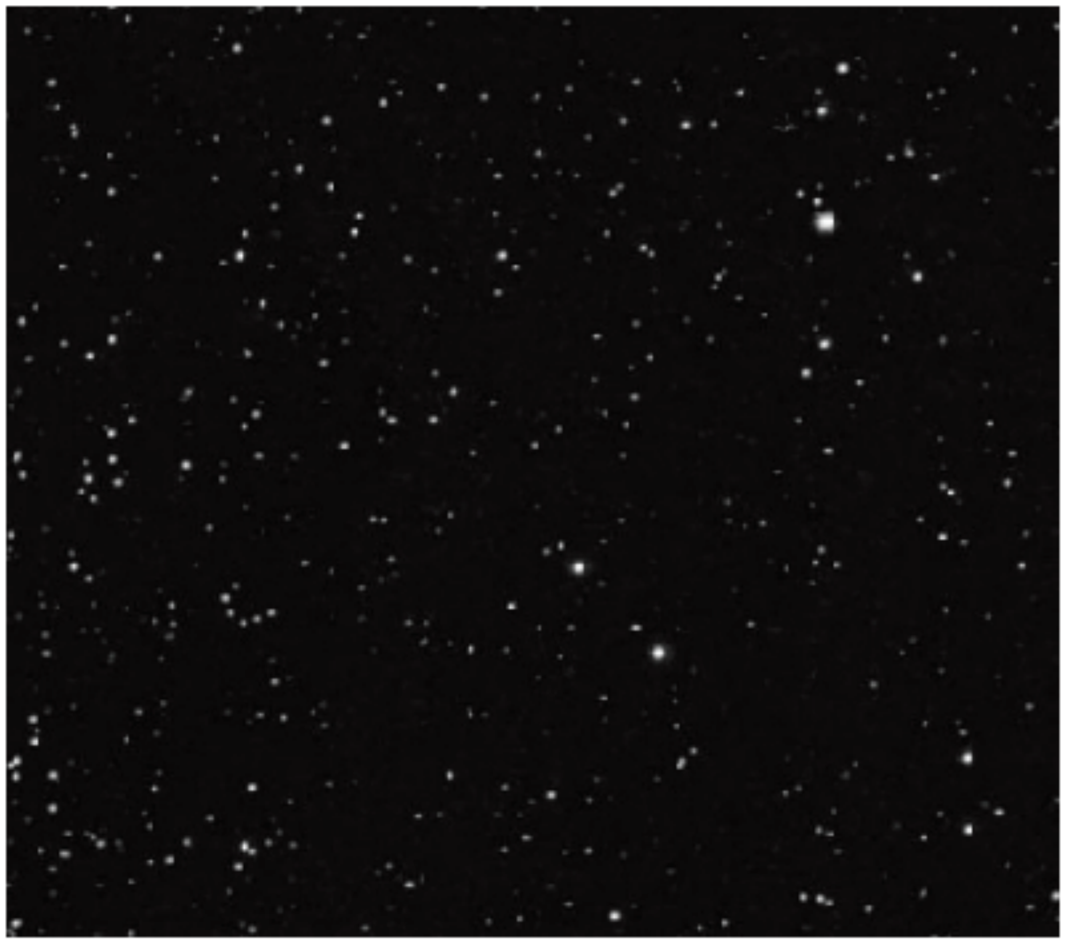
Statistical modeling of images

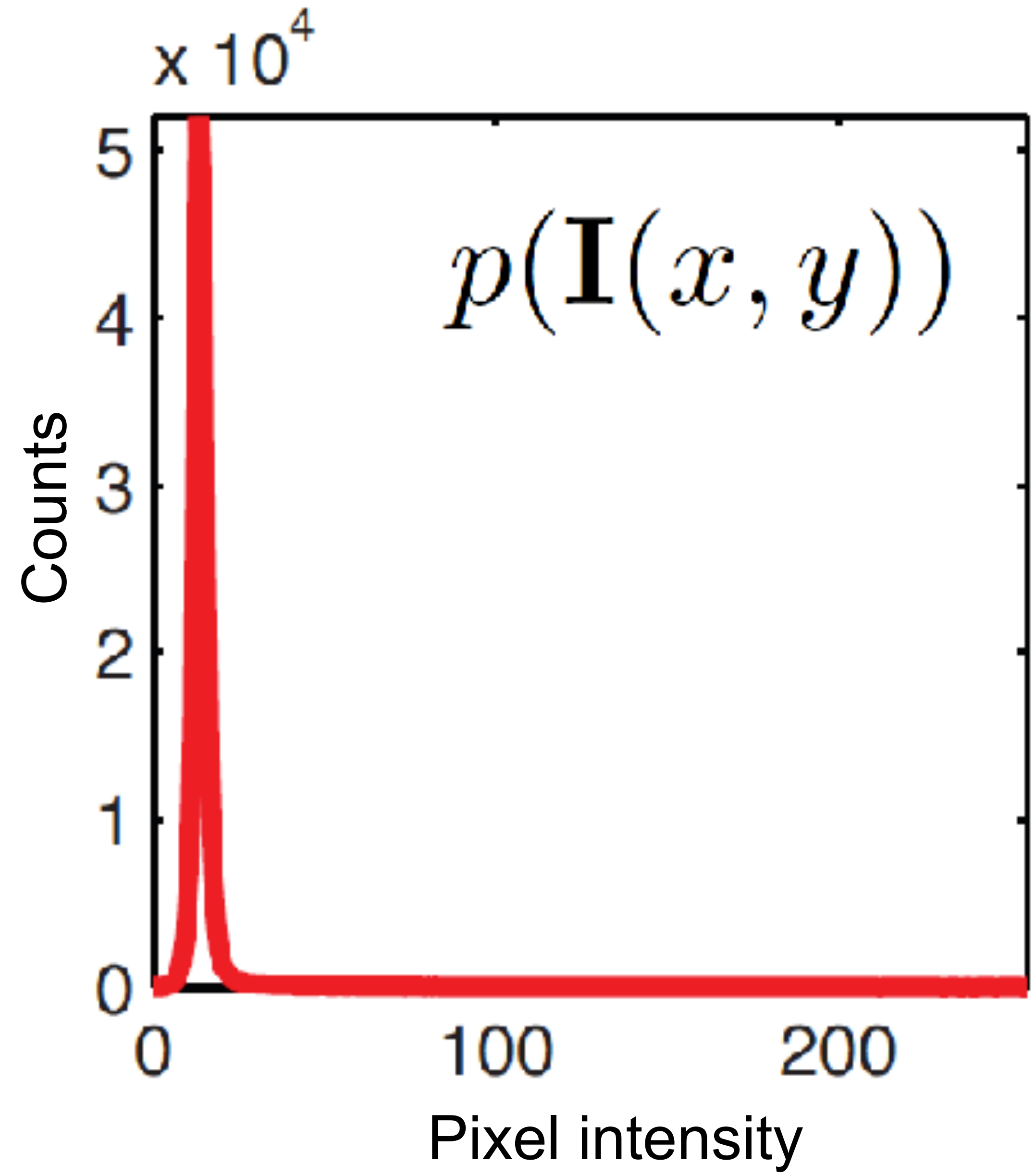
$$p(\mathbf{I}) = \prod_{x,y} p(\mathbf{I}(x,y))$$

Assumptions:
- Independence: All pixels are independent.
- Stationarity: The distribution of pixel intensities does not depend on image location.

$$p(\mathbf{I}) = \prod_{x,y} p(\mathbf{I}(x,y))$$

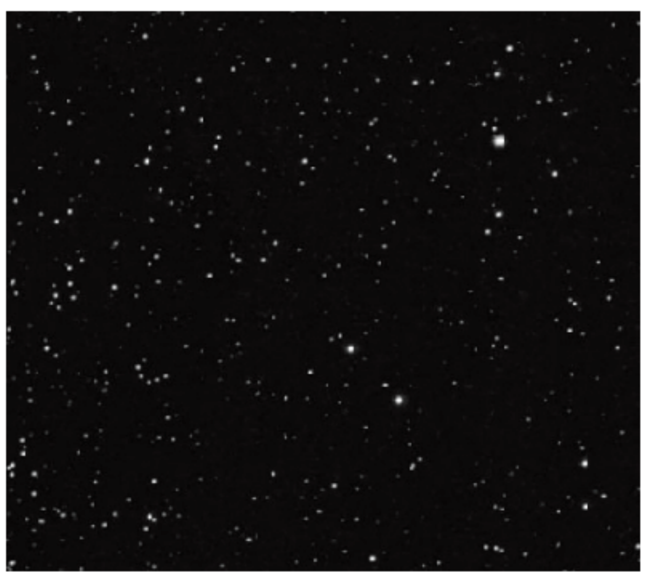
Fitting the model

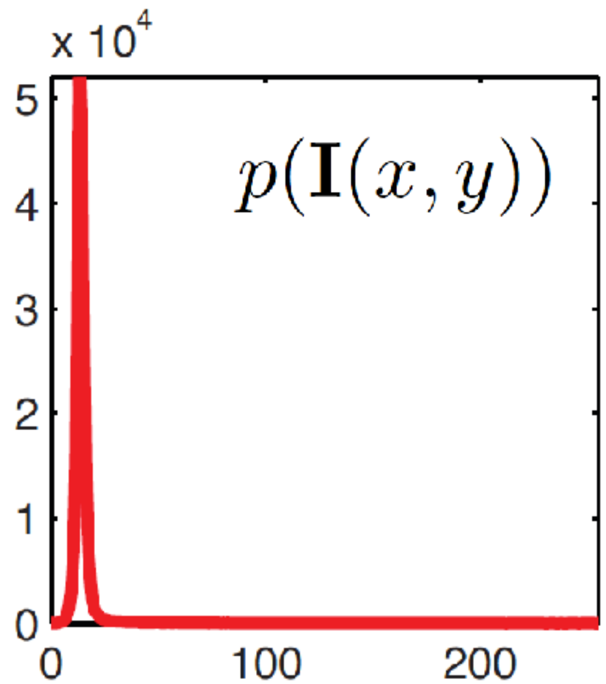




Sampling new images

$$p(\mathbf{I}) = \prod_{x,y} p(\mathbf{I}(x,y))$$

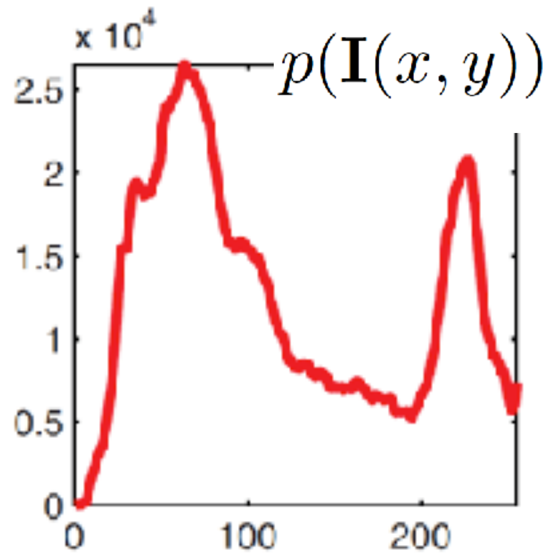


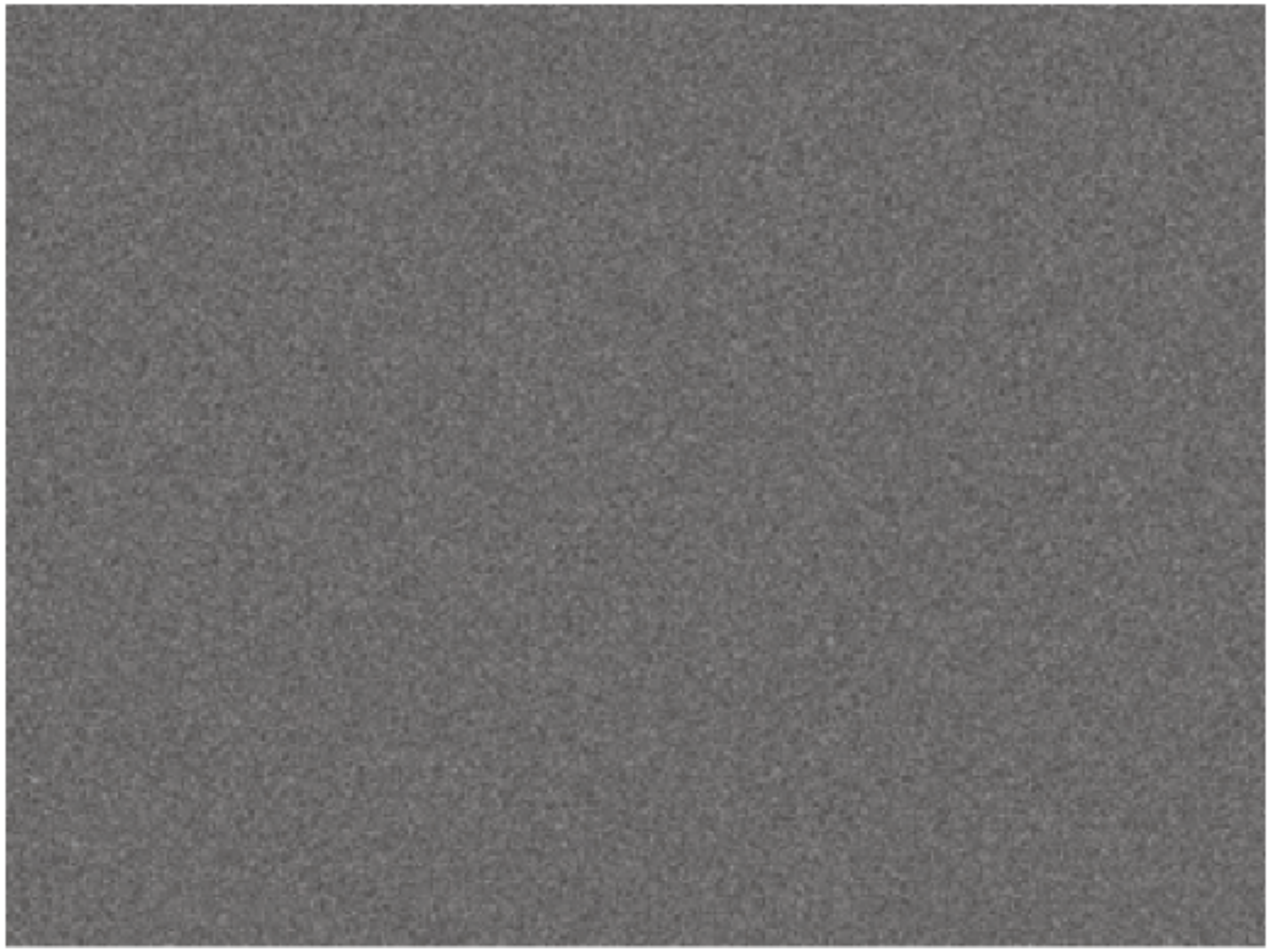


Sample

Sampling new images

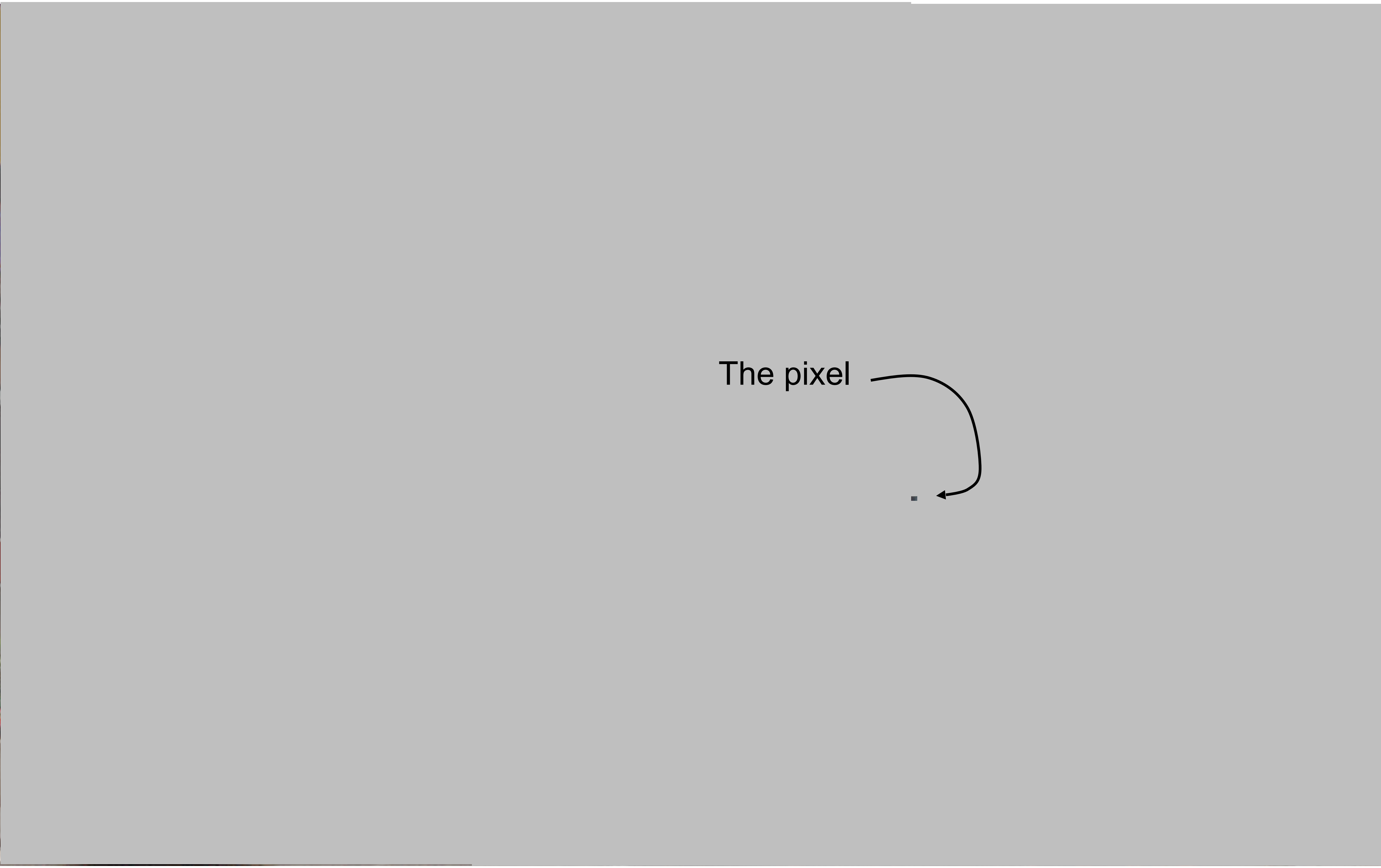
$$p(\mathbf{I}) = \prod_{x,y} p(\mathbf{I}(x,y))$$



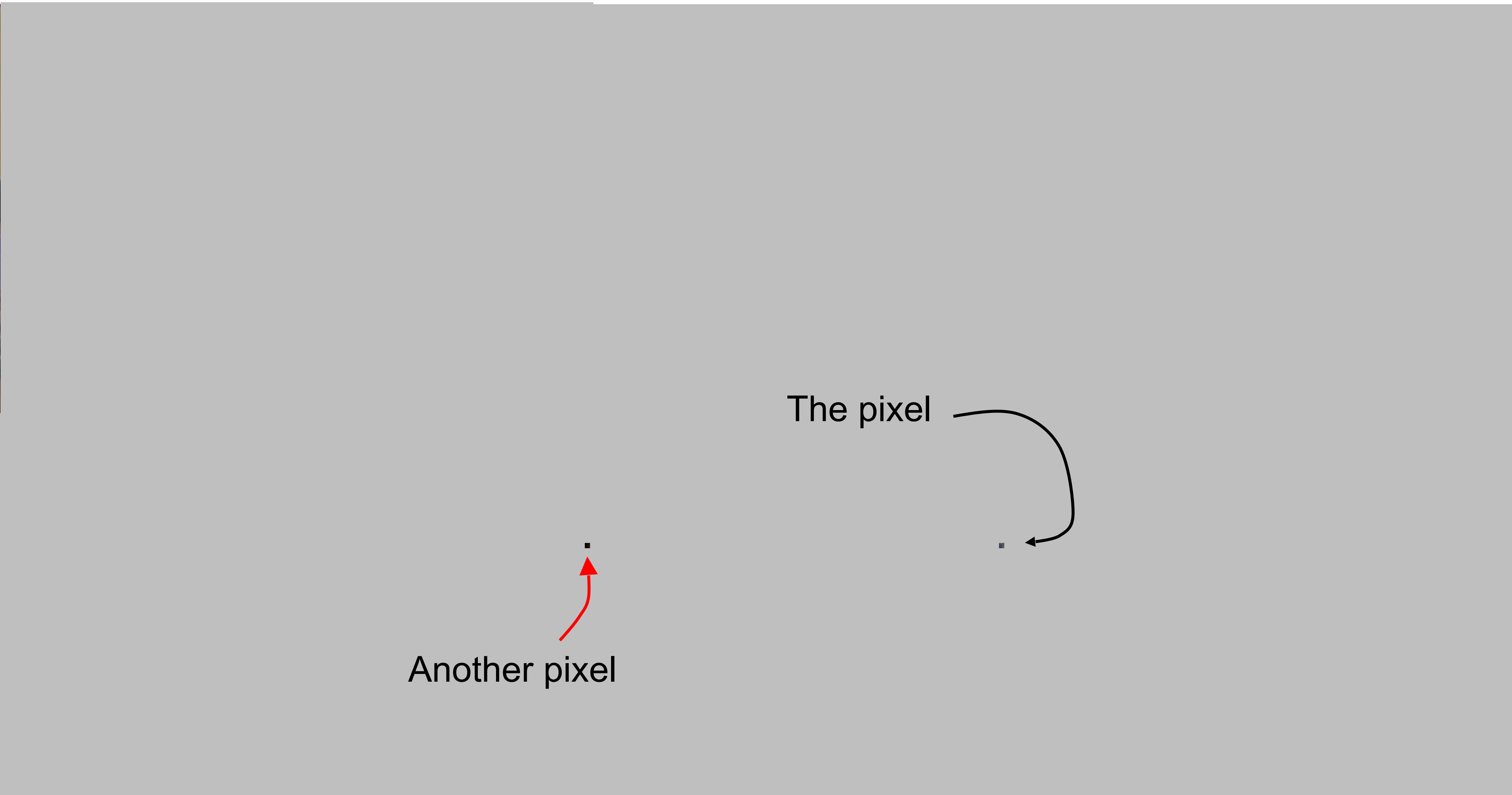


Sample

Statistical modeling of images

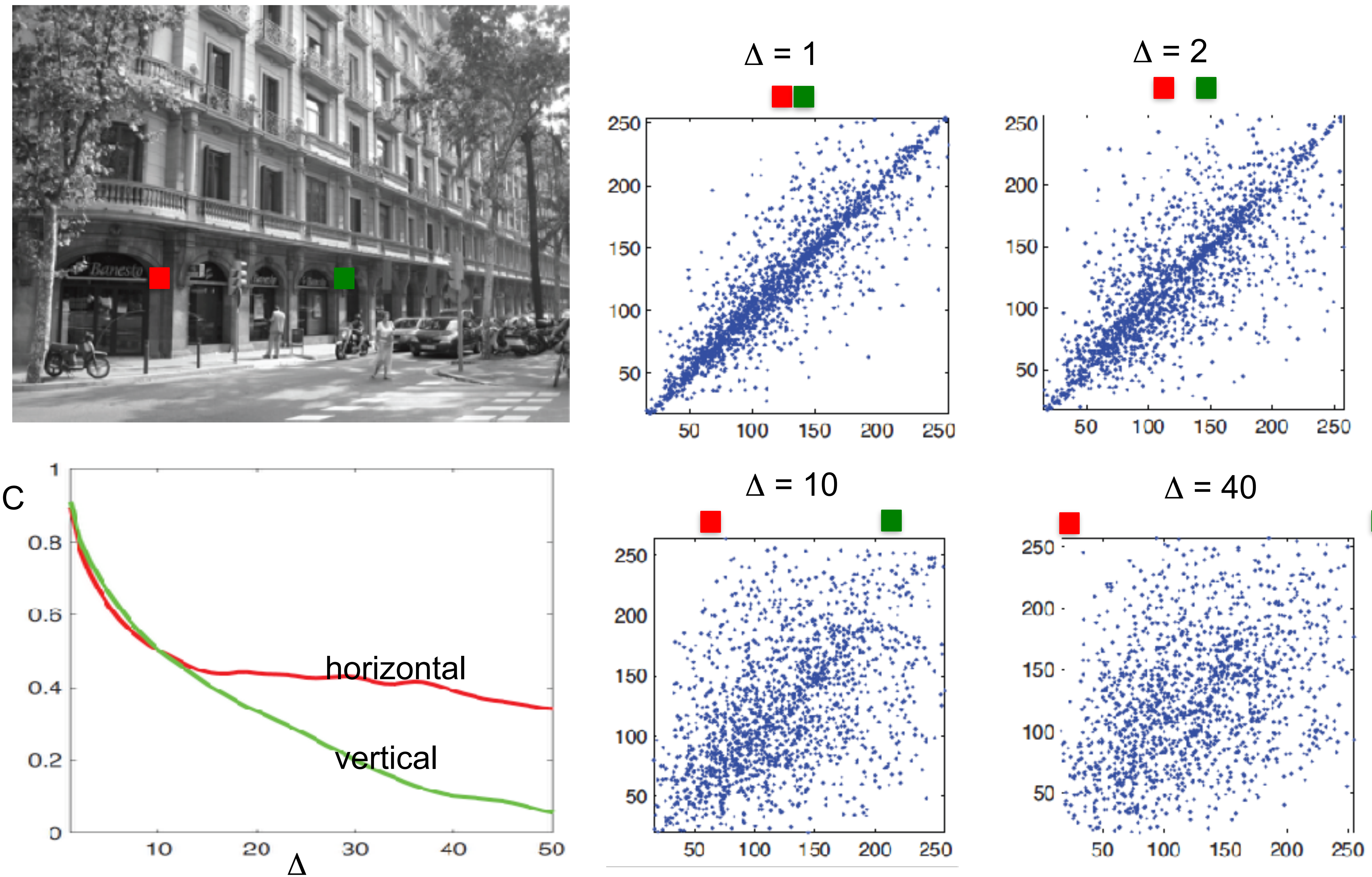


Statistical modeling of images



$$C(\Delta x, \Delta y) = \mathrm{E}[\mathbf{I}(x + \Delta x, y + \Delta y), \mathbf{I}(x, y)]$$

$$C(\Delta x, \Delta y) = \mathrm{E}\left[\mathbf{I}(x + \Delta x, y + \Delta y), \mathbf{I}(x, y)\right]$$



A remarkable property of natural images

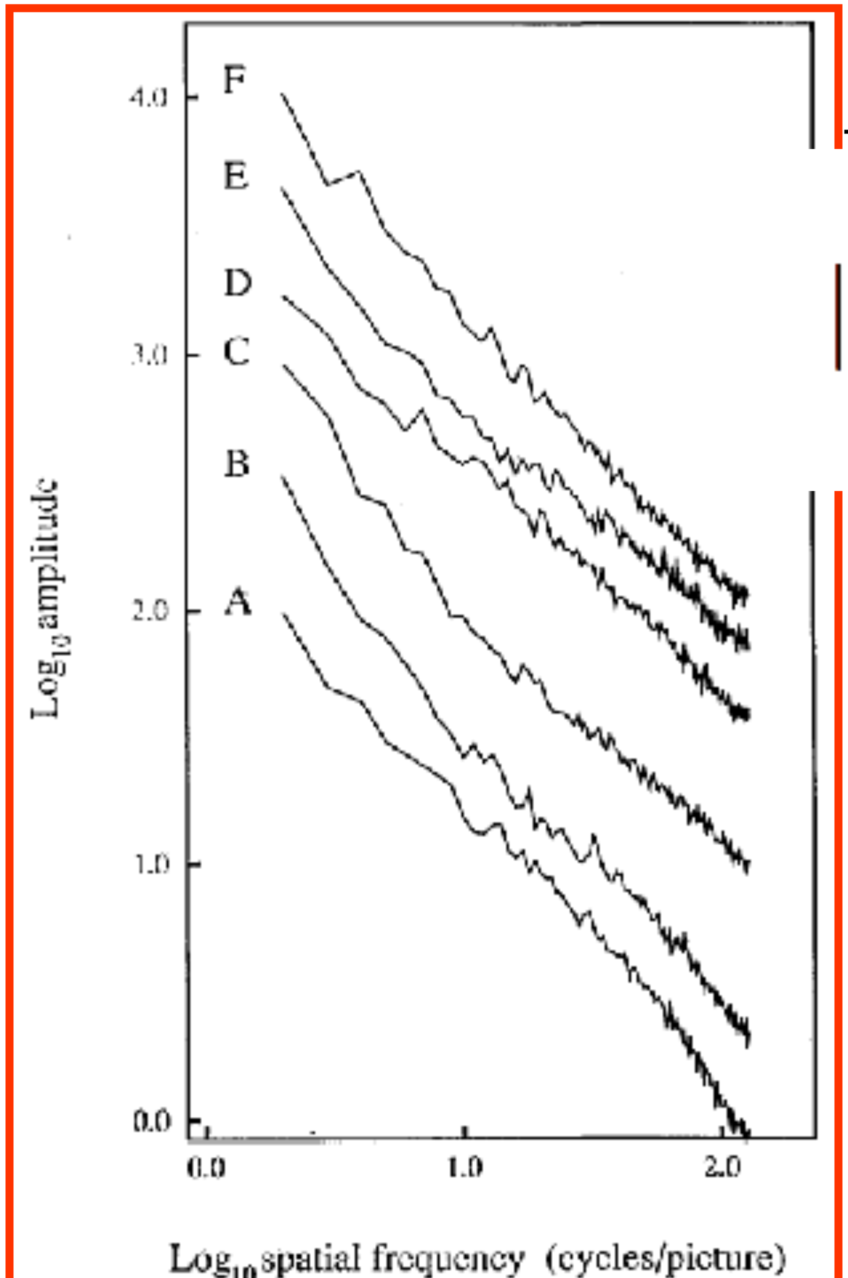
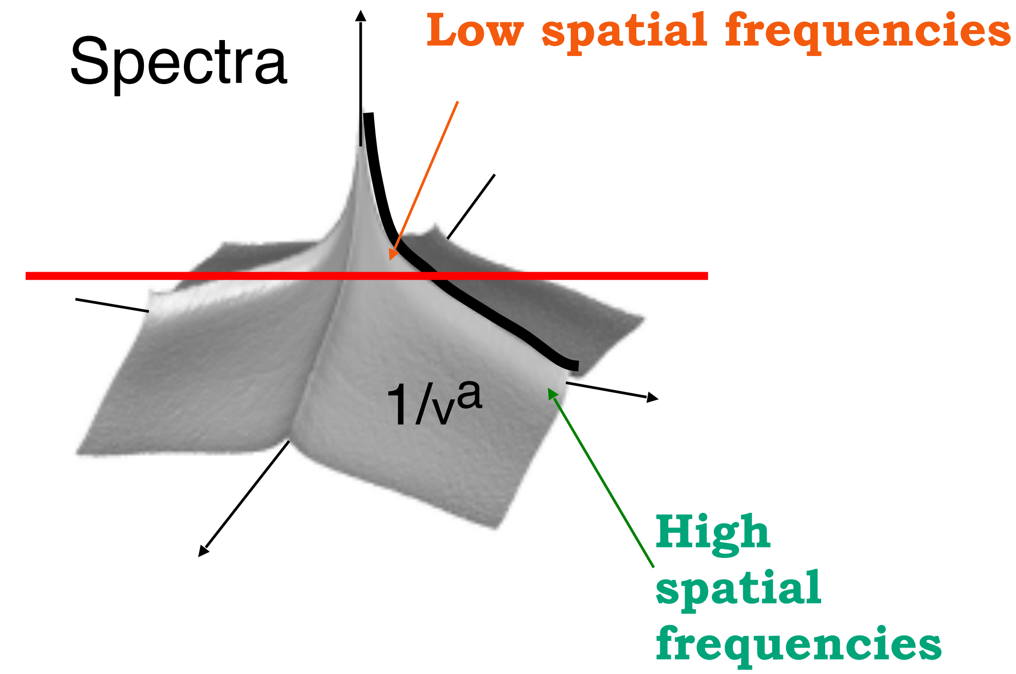
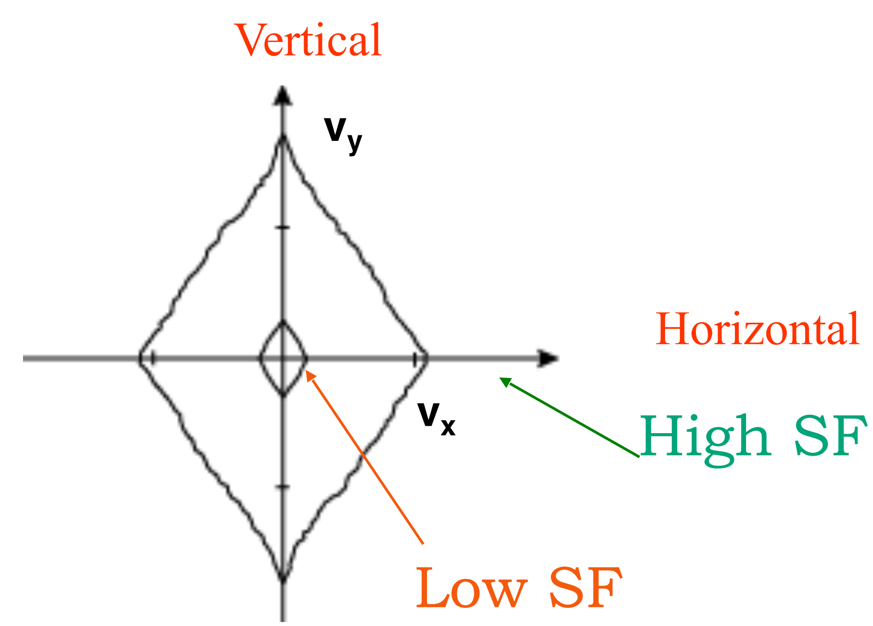


Fig. 8. Amplitude spectra for the six images A–F, averaged across all orientations. The spectra have been shifted up for clarity. On these log–log coordinates the spectra fall off by a factor of roughly $1/f$ (a slope of -1). Therefore the power spectra fall off as $1/f^2$.

Power spectra fall off as

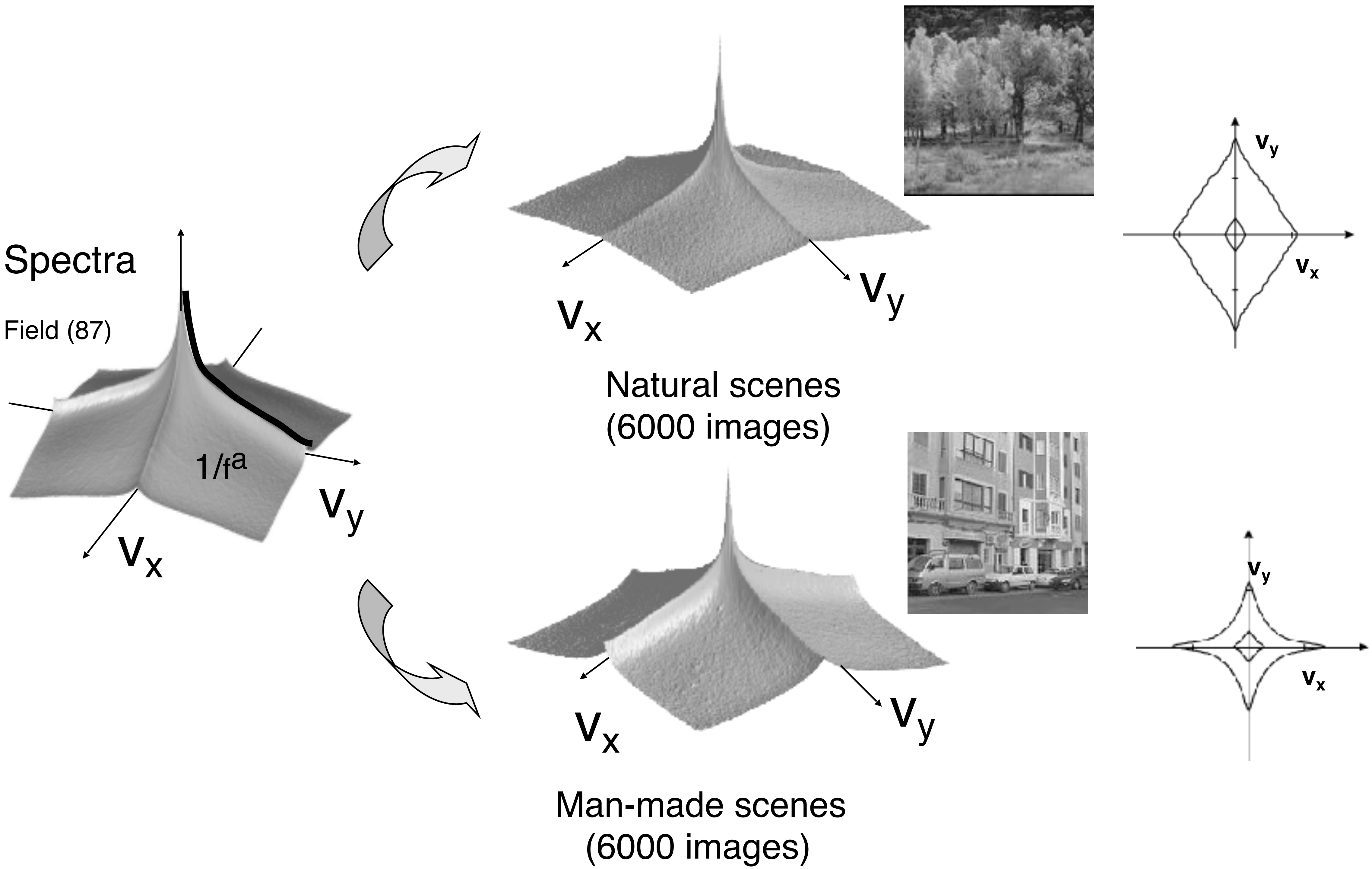
$$|\hat{\mathbf{I}}(v)| \simeq \frac{1}{|v|^{\alpha}}$$





D. J. Field, "Relations between the statistics of natural images and the response properties of cortical cells," J. Opt. Soc. Am. A **4**, 2379- (1987)

A remarkable property of natural images



Torralba and Oliva, *Statistics of Natural Image Categories*. Network: Computation in Neural Systems 14 (2003) 391-412.

Gaussian model

We want a distribution that captures the correlation structure typical of natural images.

Let **C** be the covariance matrix of the image:

$$p(\mathbf{I}) = \exp\left(-\frac{1}{2}\mathbf{I}^T\mathbf{C}^{-1}\mathbf{I}\right) \qquad C = \begin{bmatrix} c_0 & c_1 & c_2 & & \cdots & c_{n-1} \\ c_{n-1} & c_0 & c_1 & c_2 & & \vdots \\ & c_{n-1} & c_0 & c_1 & \ddots & \\ \vdots & \ddots & \ddots & \ddots & & c_2 \\ & & & & & c_1 \\ c_1 & \cdots & & c_{n-1} & & c_0 \end{bmatrix}$$

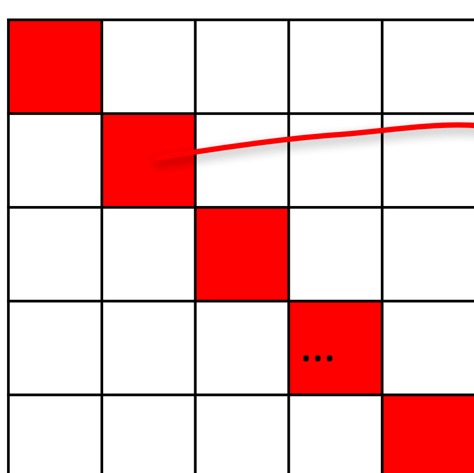
Stationarity assumption: Symmetrical circulant matrix

Diagonalization of circulant matrices: $C = EDE^T$

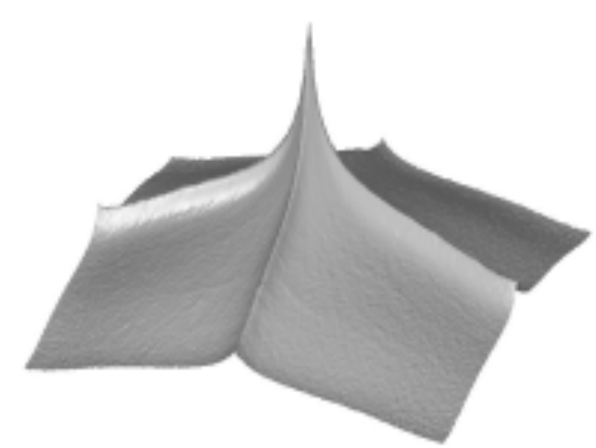
The eigenvectors are the Fourier basis

The eigenvalues are the squared magnitude of the Fourier coefficients

D=

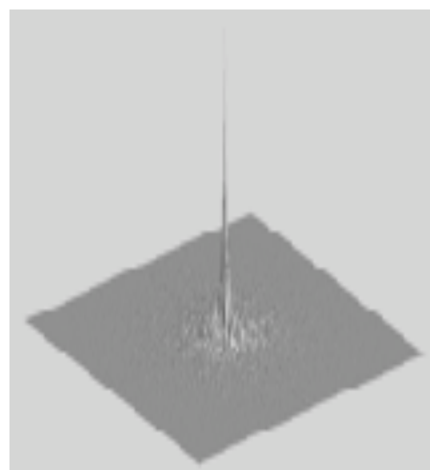


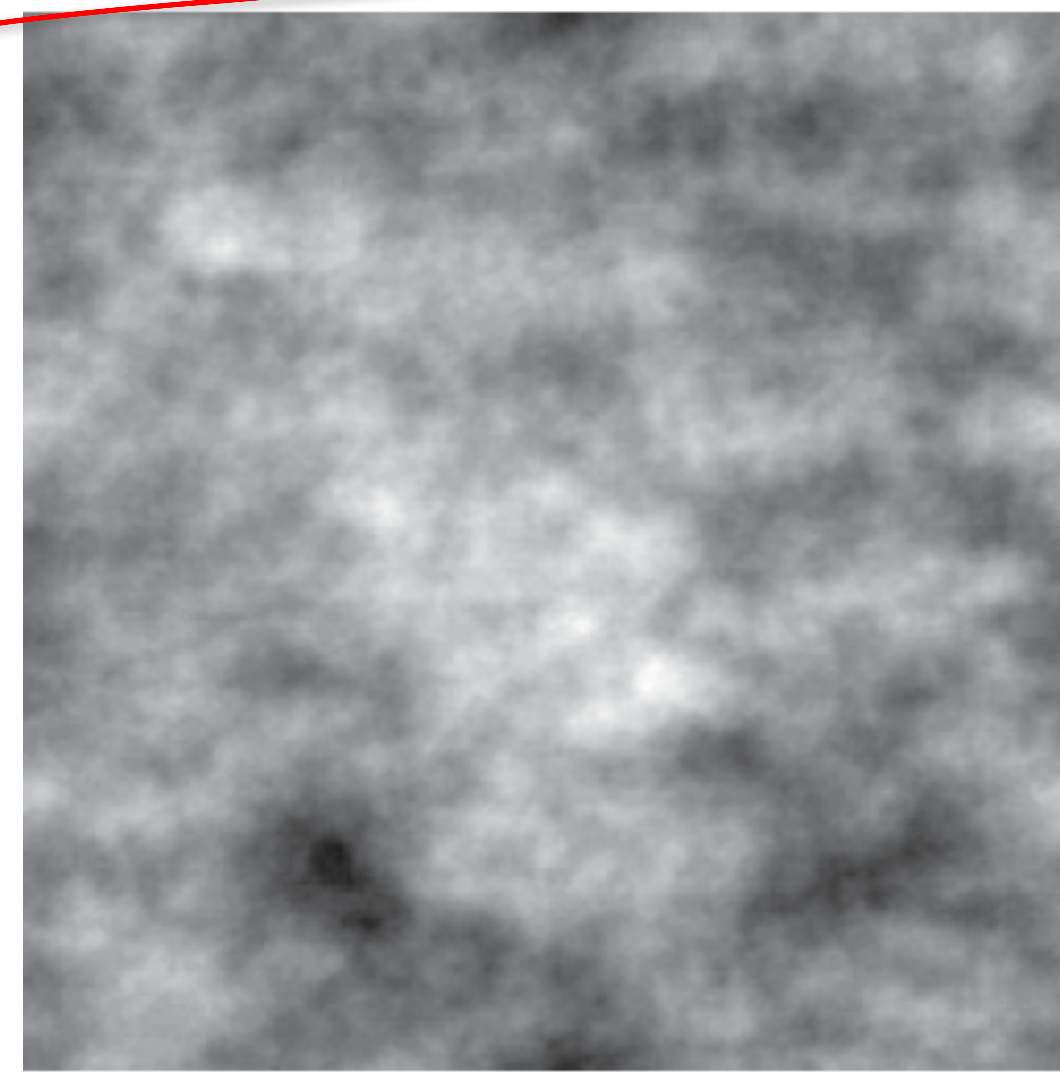
$$\left|\hat{\mathbf{I}}(v)\right|^2 \simeq \frac{1}{|v^2|^{\alpha}}$$



Sampling new images

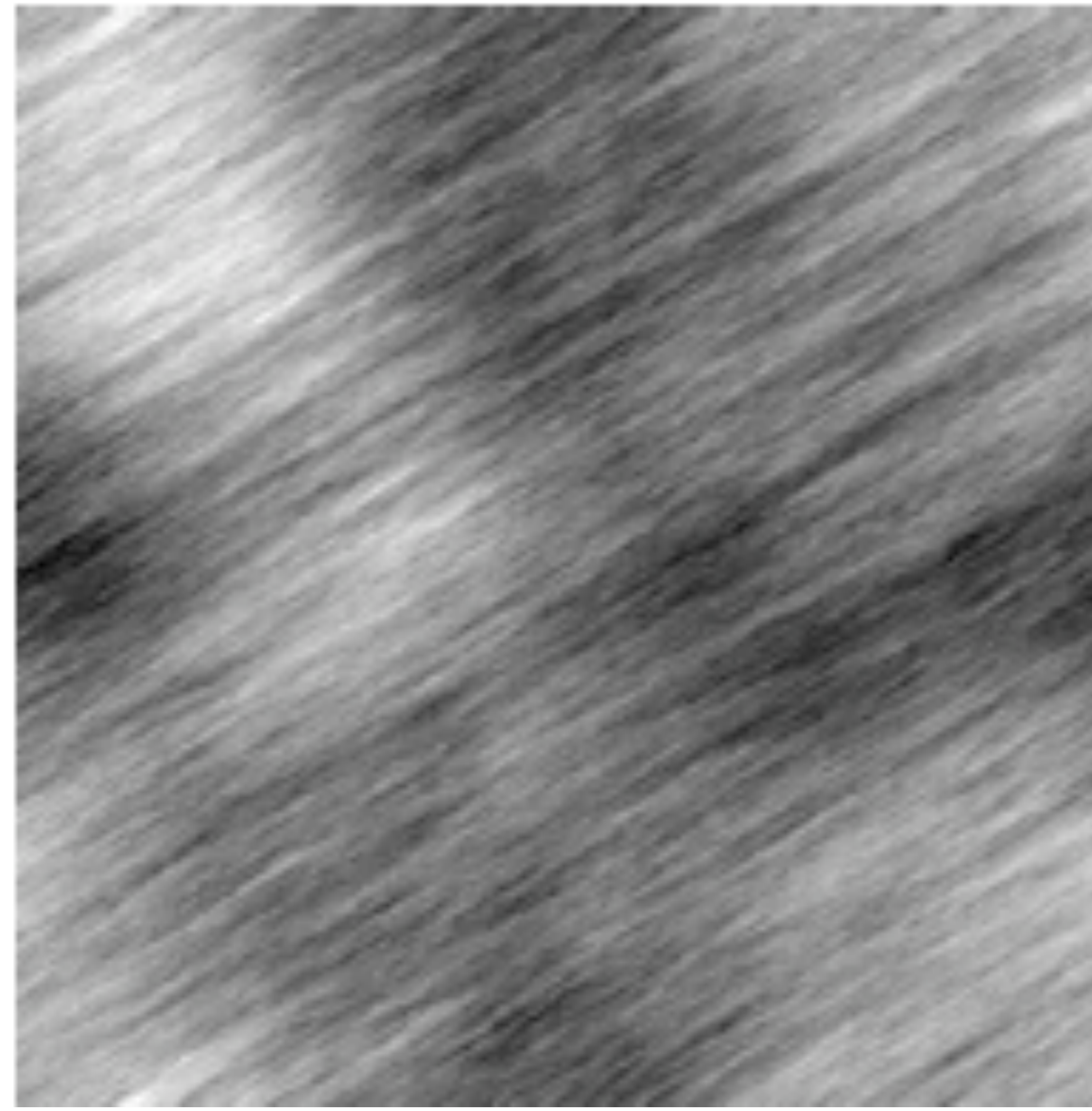
$$p(\mathbf{I}) = \exp\left(-\frac{1}{2}\mathbf{I}^T\mathbf{C}^{-1}\mathbf{I}\right)$$



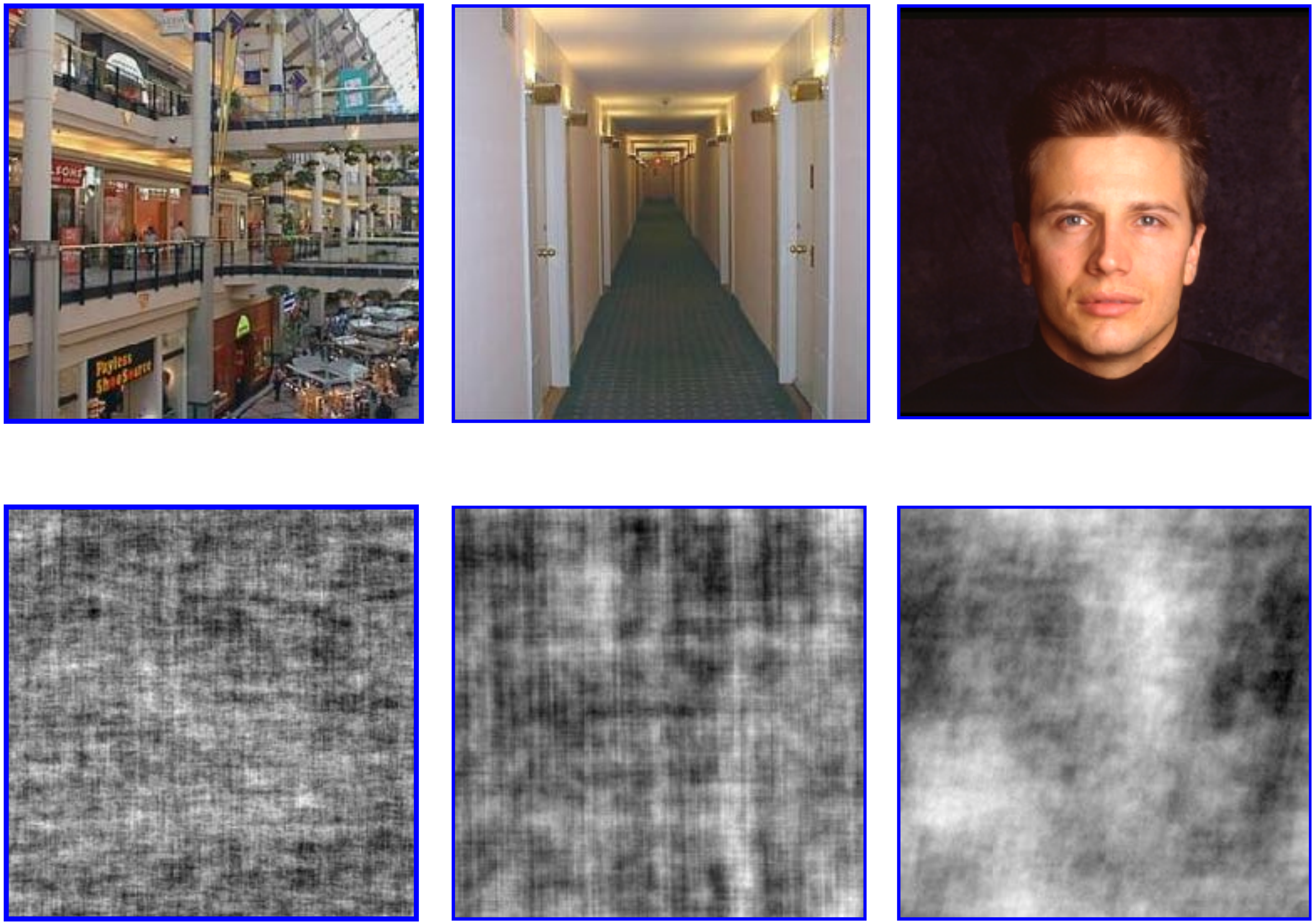


Sample

Sampling new images

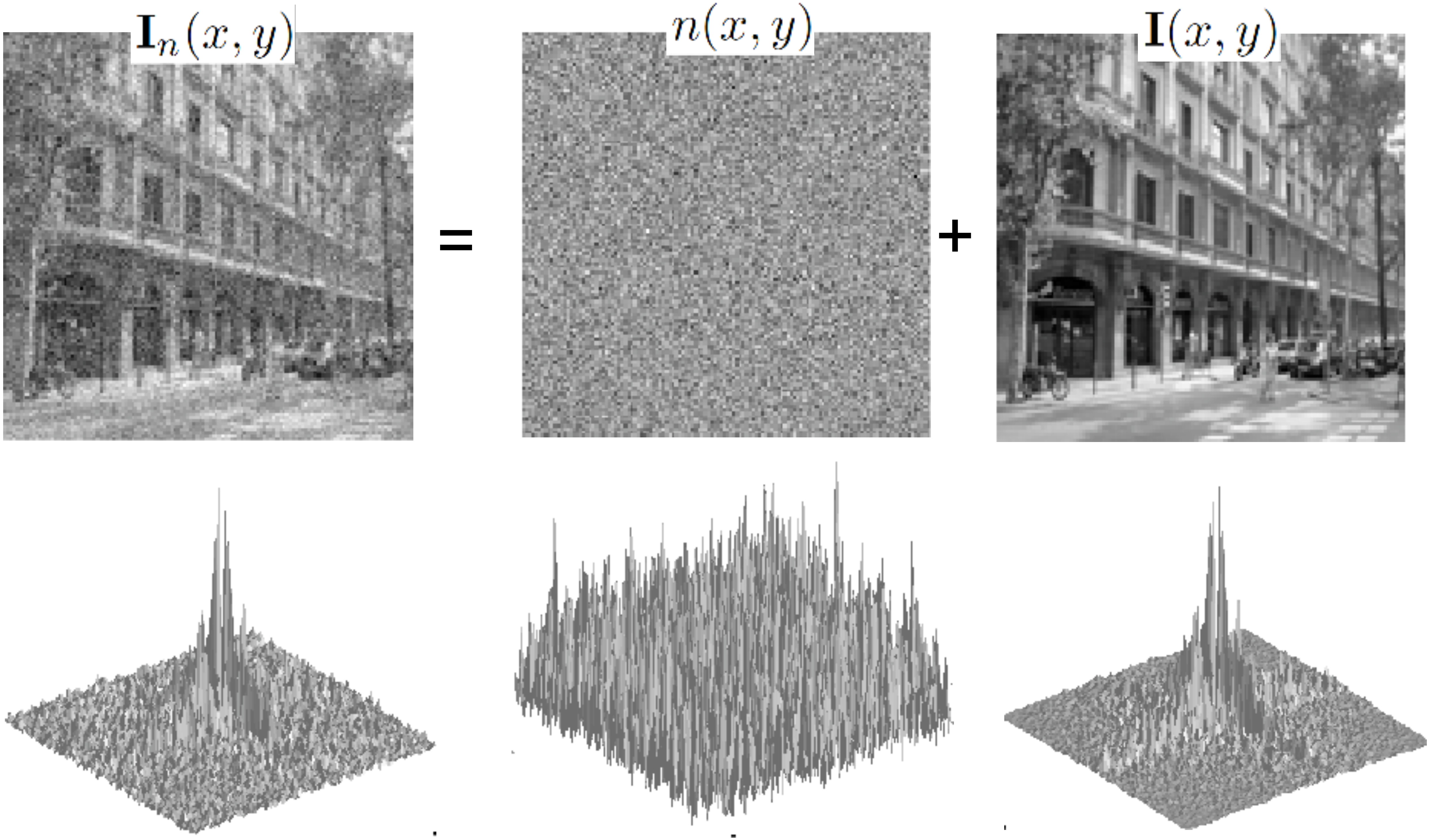


Randomizing the phase (fit the Gaussian image model to each of the images in the top row, then draw another random sample, you get the bottom row)



Denoising

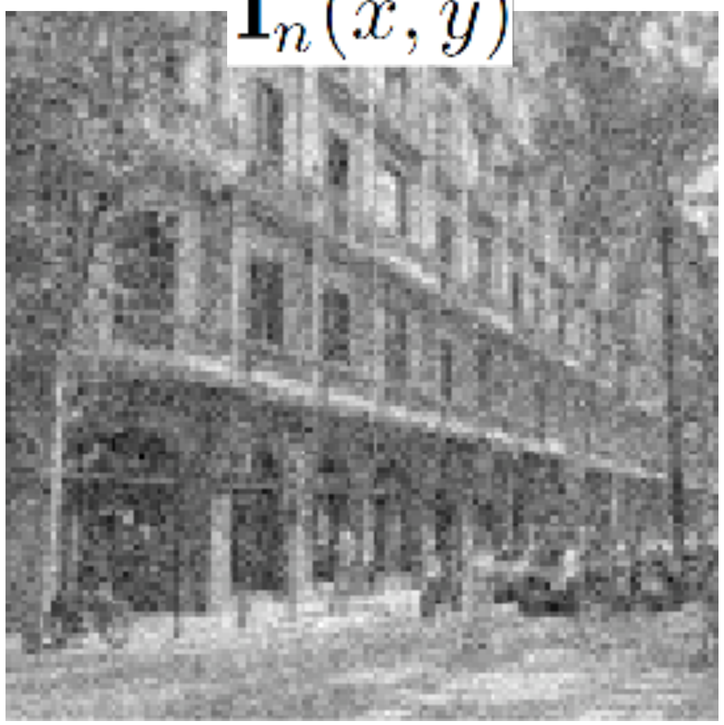
Decomposition of a noisy image



Denoising

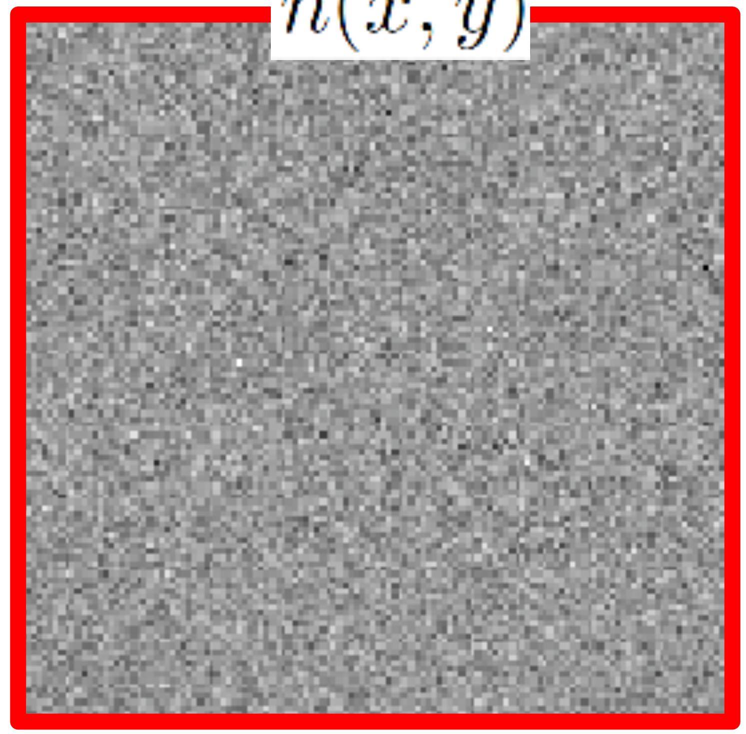
Decomposition of a noisy image

$\mathbf{I}_n(x,y)$



=

$n(x,y)$



+

$\mathbf{I}(x,y)$

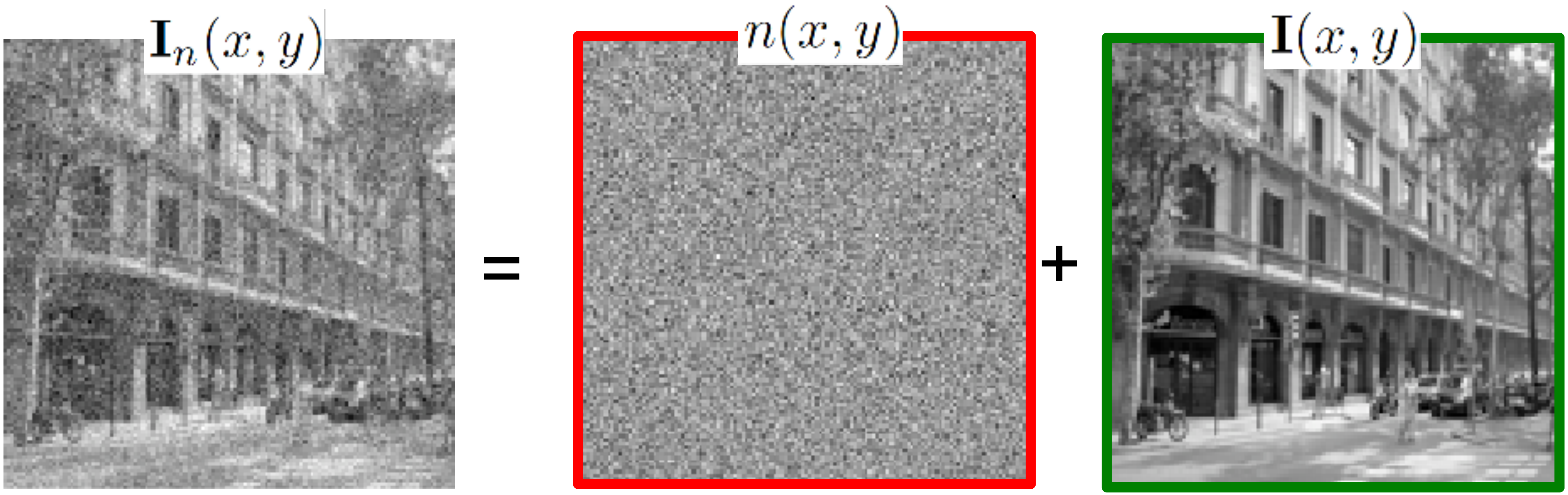
White Gaussian noise: $N(0, \sigma_n^2)$ Natural image

Find I(x,y) that maximizes the posterior (maximum a posteriori, MAP):

$$\max_{\mathbf{I}} p(\mathbf{I}|\mathbf{I}_n) = \max_{\mathbf{I}} \underbrace{p(\mathbf{I}_n|\mathbf{I})}_{\text{likelihood}} \times \underbrace{p(\mathbf{I})}_{\text{prior}}$$

Denoising

Decomposition of a noisy image



White Gaussian noise: $N(0, \sigma_n^2)$ Natural image

Find I(x,y) that maximizes the posterior (maximum a posteriori, MAP):

$$\max_{\mathbf{I}} p(\mathbf{I}|\mathbf{I}_n) = \max_{\mathbf{I}} \underbrace{p(\mathbf{I}_n|\mathbf{I})}_{\text{likelihood}} \times \underbrace{p(\mathbf{I})}_{\text{prior}}$$

$$= \max_{\mathbf{I}} \exp(-\left|\mathbf{I}_n - \mathbf{I}\right|^2 / \sigma_n^2) \times \exp\left(-\frac{1}{2}\mathbf{I}^T\mathbf{C}^{-1}\mathbf{I}\right)$$

Denoising

$$\max_{\mathbf{I}} p(\mathbf{I}|\mathbf{I}_n) = \max_{\mathbf{I}} \quad p(\mathbf{I}_n|\mathbf{I}) \quad \text{x} \quad p(\mathbf{I})$$

likelihood — prior

$$= \max_{\mathbf{I}} \exp(-|\mathbf{I}_n - \mathbf{I}|^2/\sigma_n^2) \text{ x } \exp\left(-\frac{1}{2}\mathbf{I}^T\mathbf{C}^{-1}\mathbf{I}\right)$$

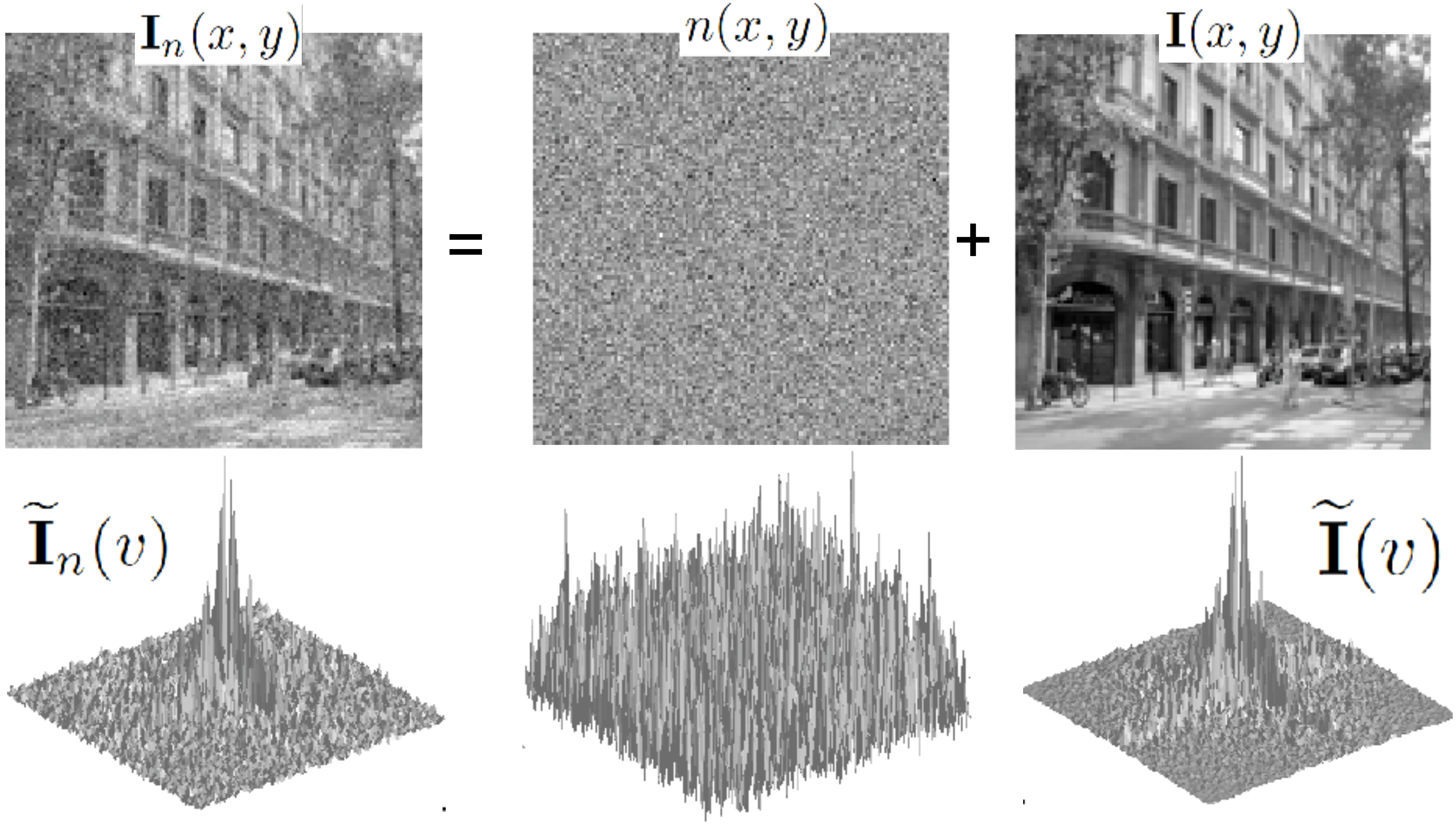
The solution is:

$$\underline{\mathbf{I}} = \mathbf{C}\left(\mathbf{C} + \sigma_n^2 \mathbb{I}\right)^{-1}\mathbf{I}_n$$ (note this is a linear operation)

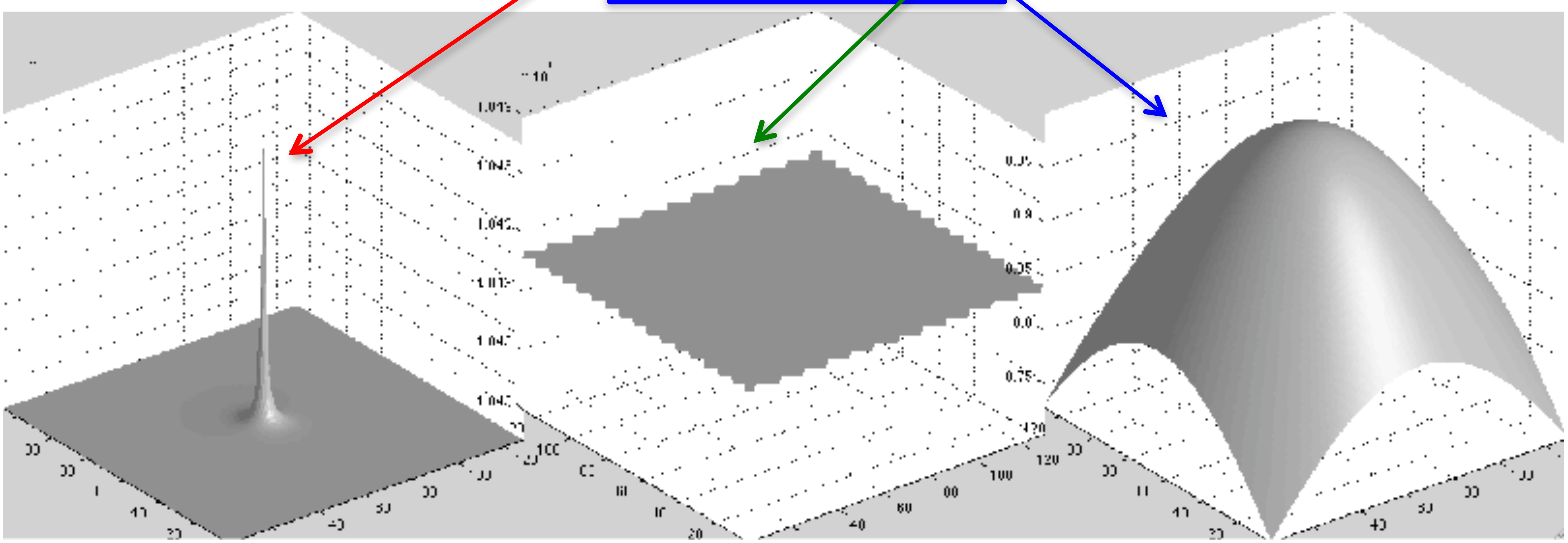
This can also be written in the Fourier domain, with C = EDE^T:

$$\widetilde{\mathbf{I}}(v) = \frac{A/|v|^{2\alpha}}{A/|v|^{2\alpha} + \sigma_n^2}\widetilde{\mathbf{I}}_n(v)$$

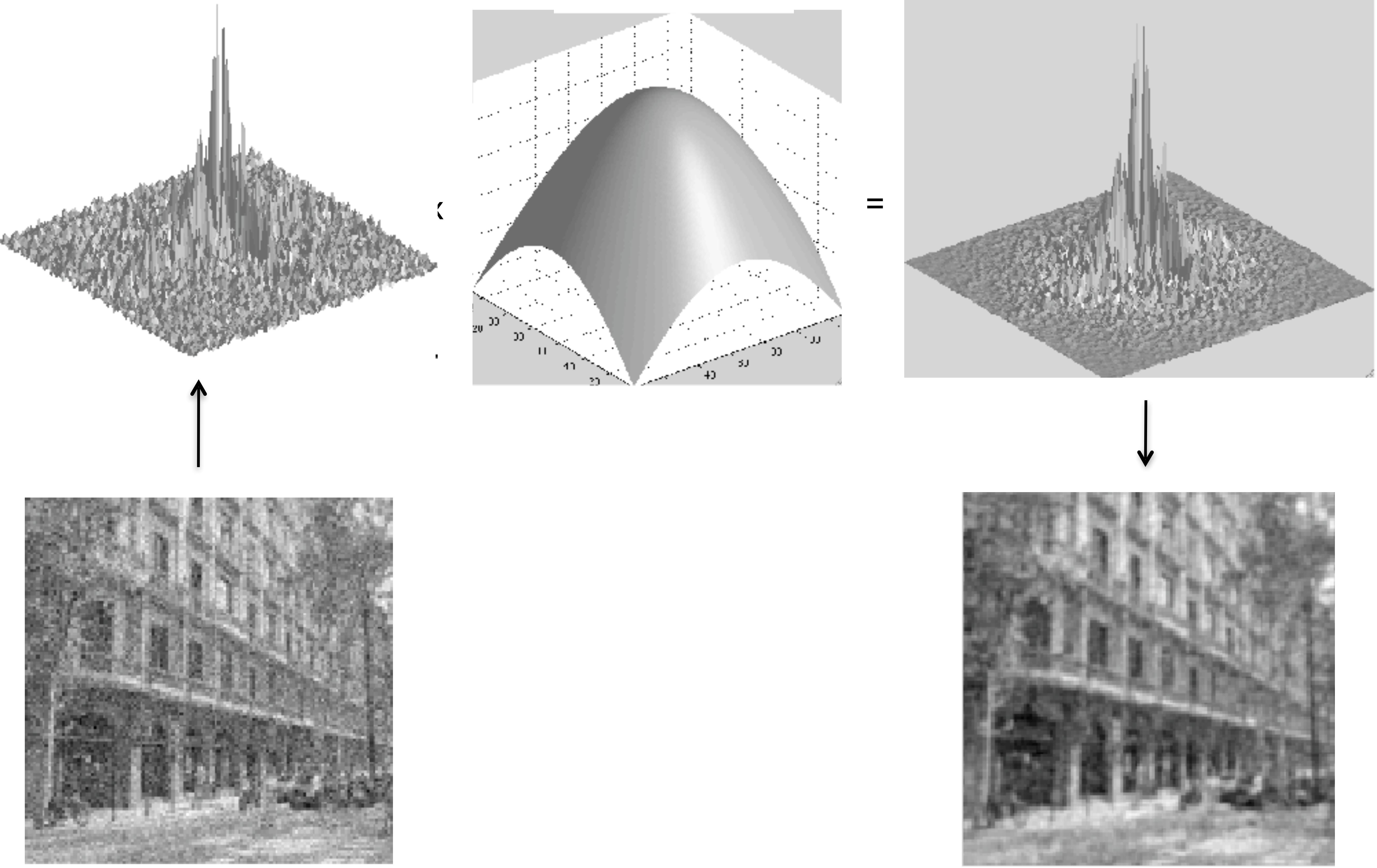
Decomposition of a noisy image



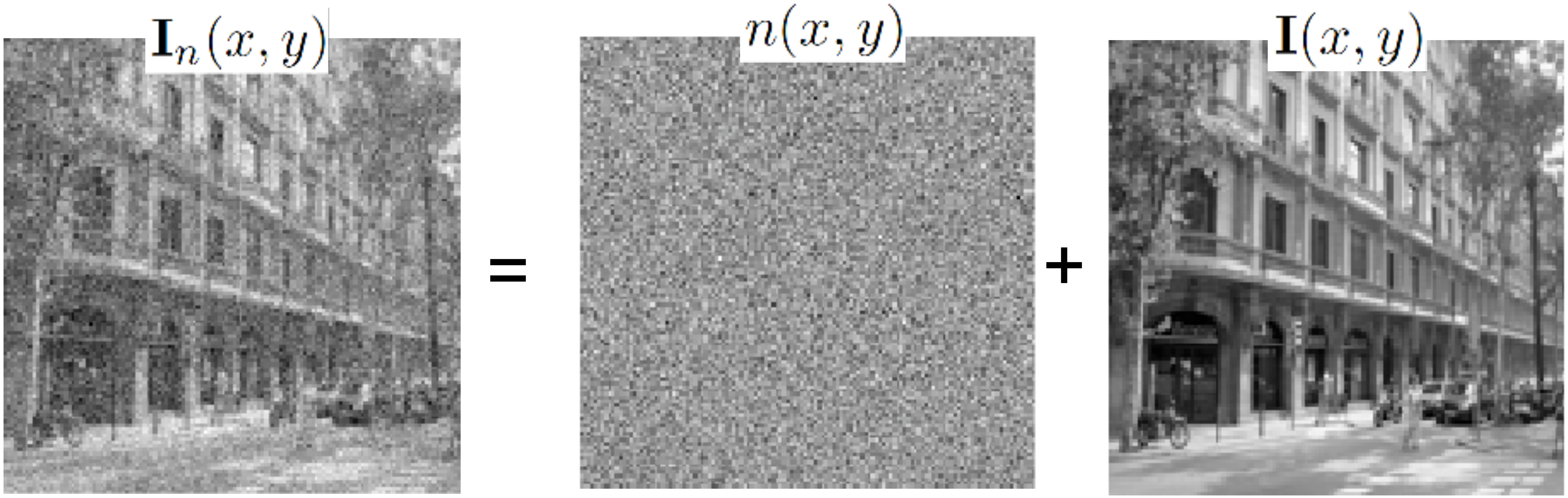
$$\widetilde{\mathbf{I}}(v) = \frac{A/|v|^{2\alpha}}{A/|v|^{2\alpha} + \sigma_n^2} \widetilde{\mathbf{I}}_n(v)$$



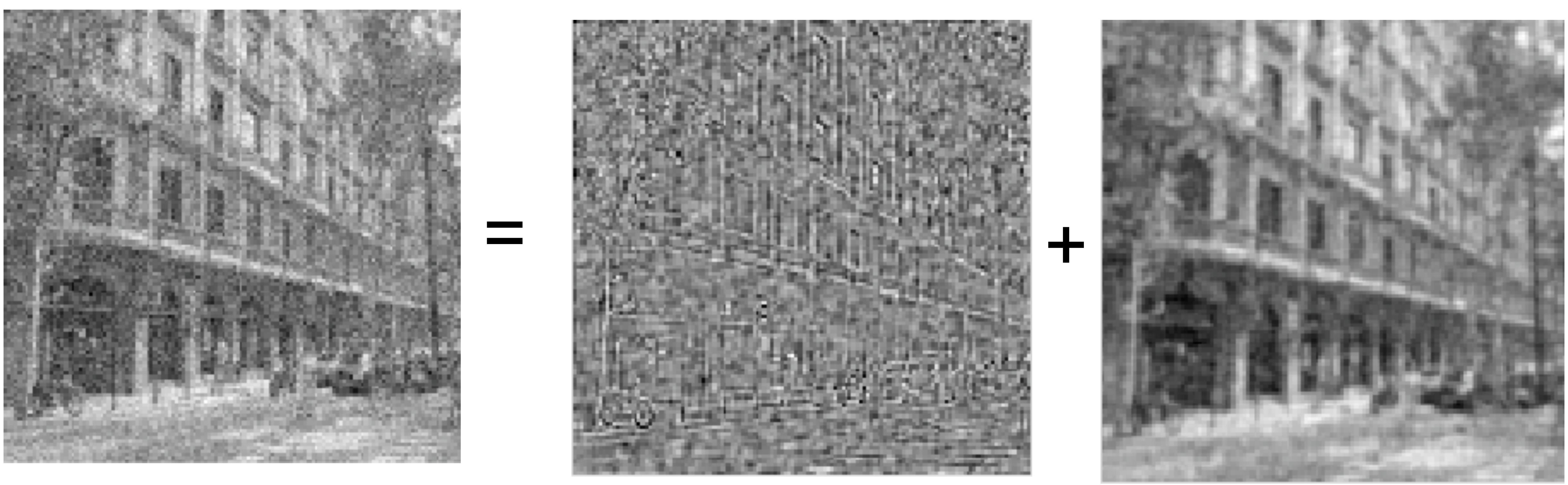
$$\frac{A/|v|^{2\alpha}}{A/|v|^{2\alpha}+\sigma_n^2}$$



The truth:



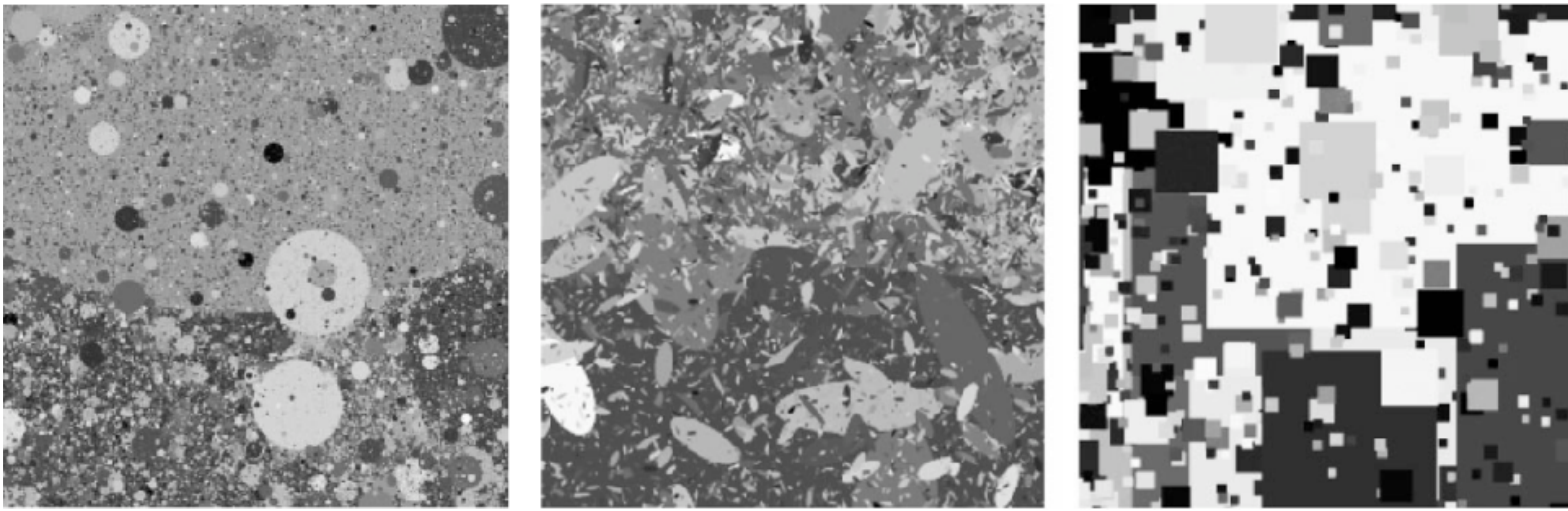
The estimated decomposition:



And we got all this from just modeling the correlation between pairs of pixels!

Dead leaves models

Introduced in the 60’s by Matheron (67) and popularized by Ruderman (97)



From *Lee, Mumford and Huang 2001*

Edges

[-1 1]

g[m,n]

⊗

[-1, 1]

h[m,n]

=

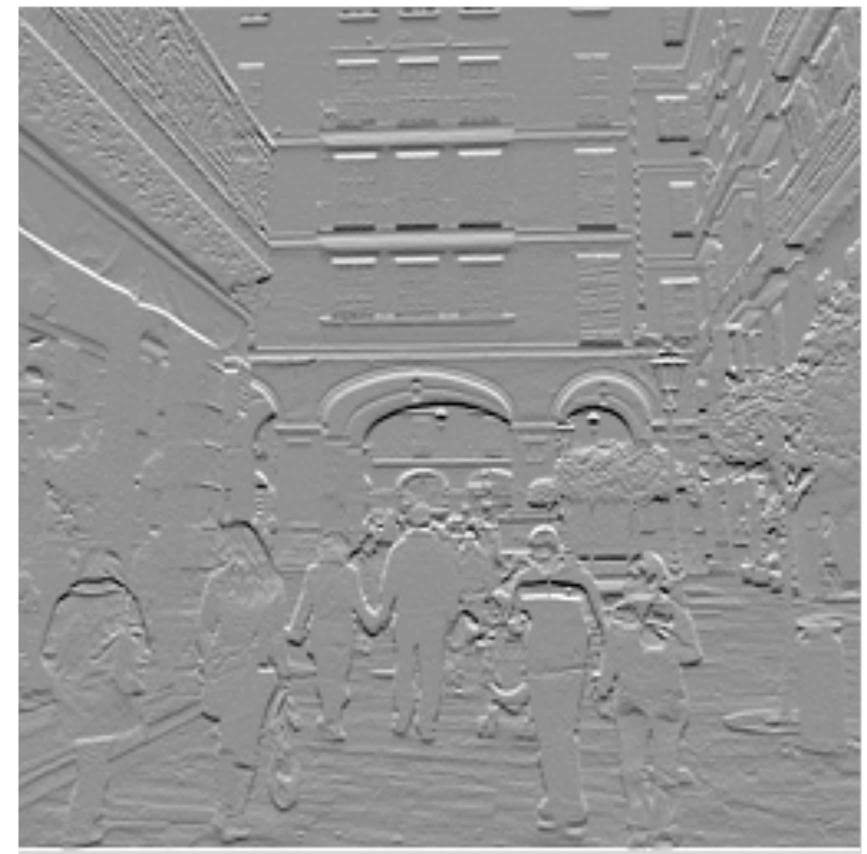
f[m,n]

$[-1\ 1]^T$

$\otimes$

$[-1, 1]^T$ =

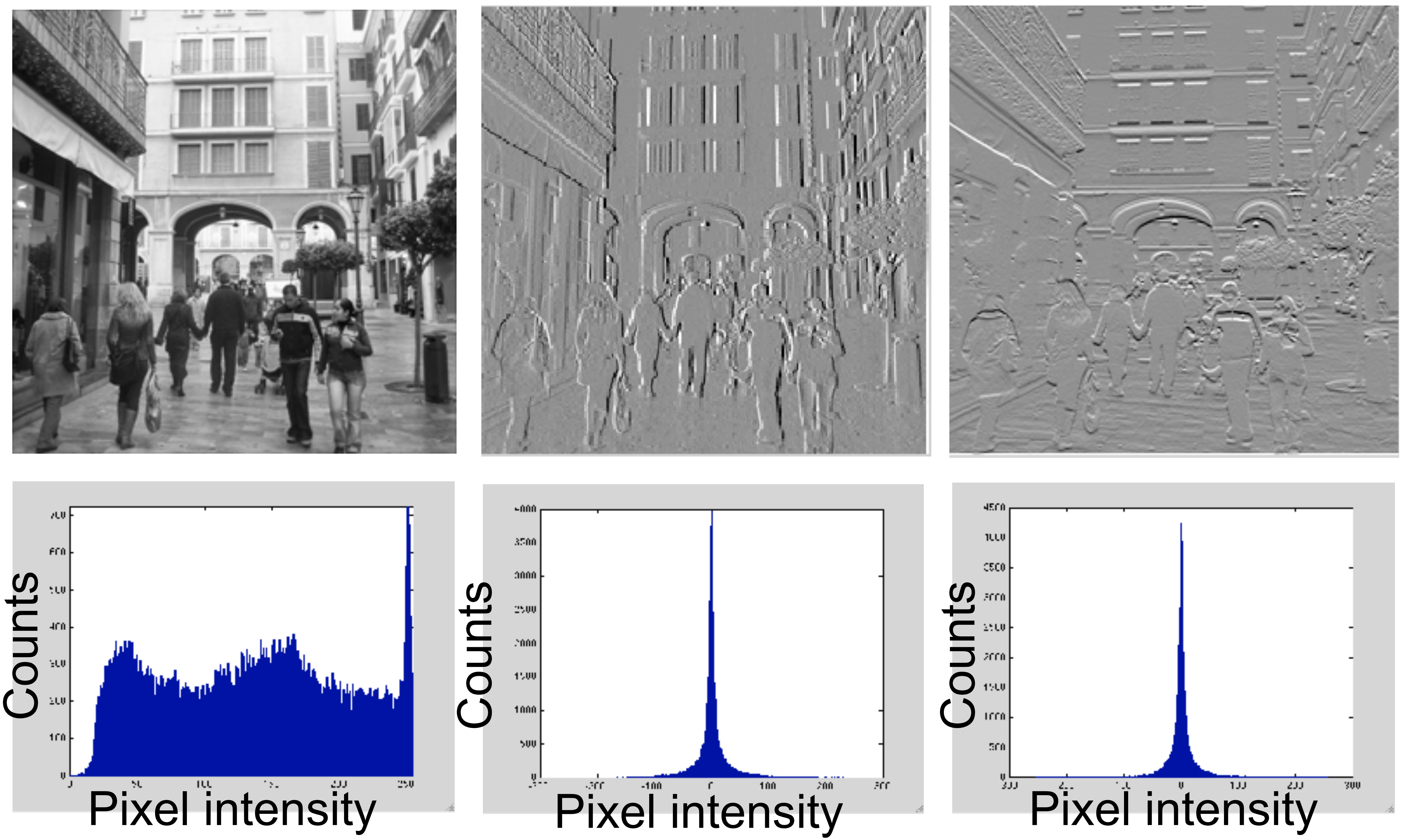
$h[m,n]$



$g[m,n]$

$f[m,n]$

Observation: Sparse filter response



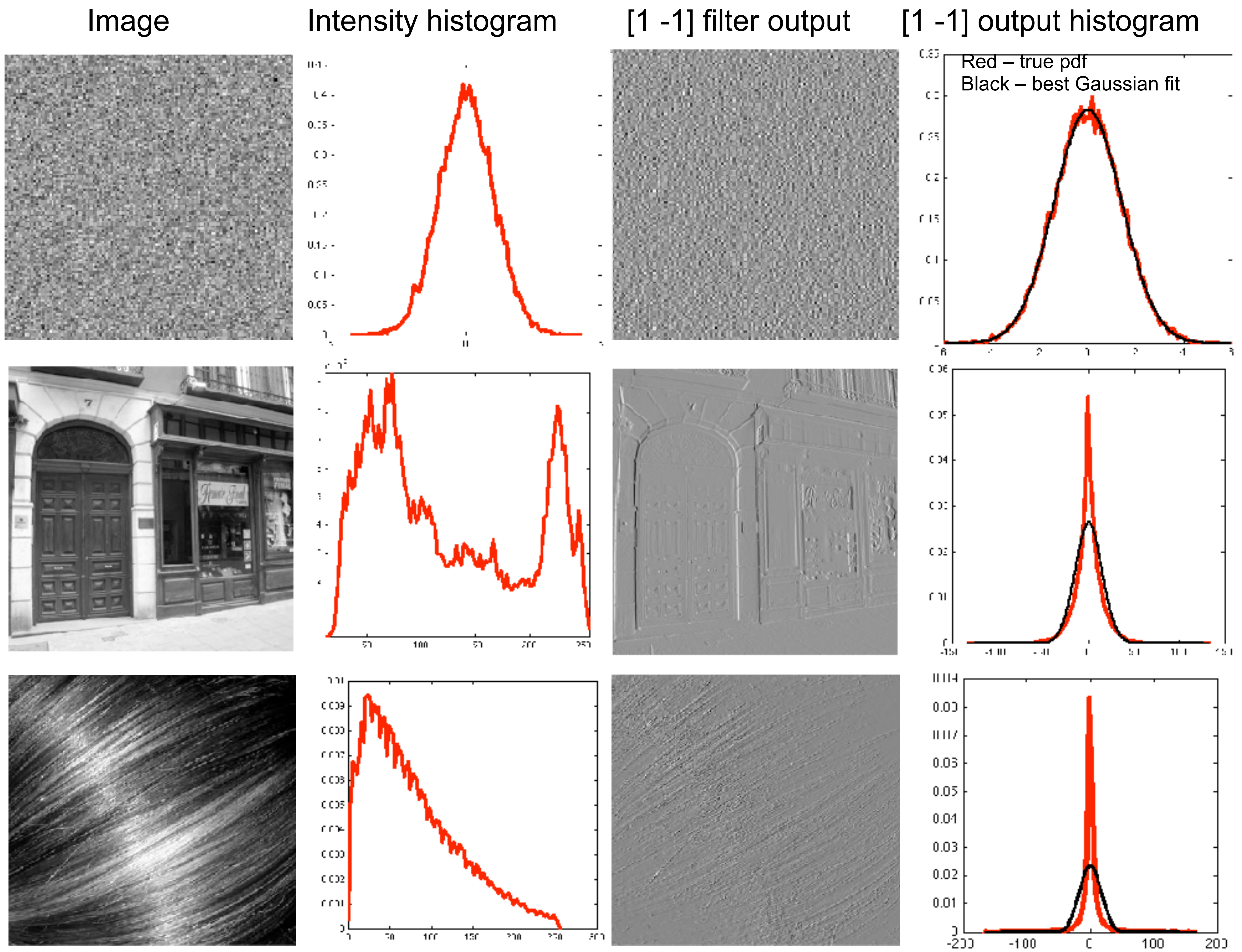
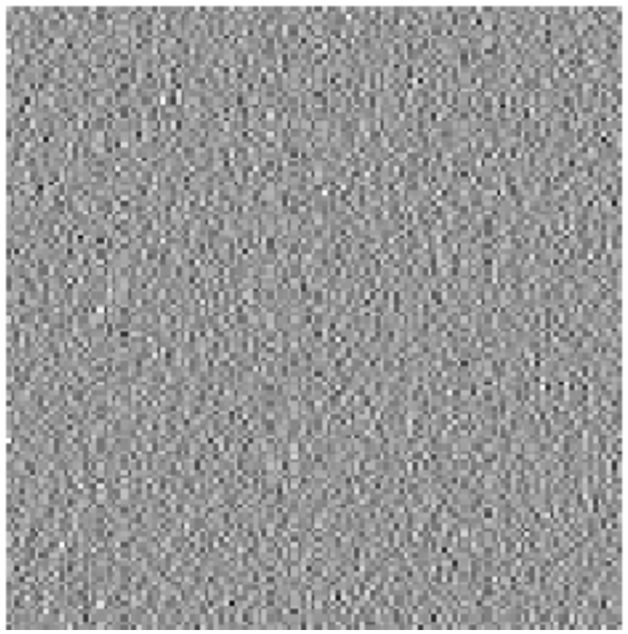
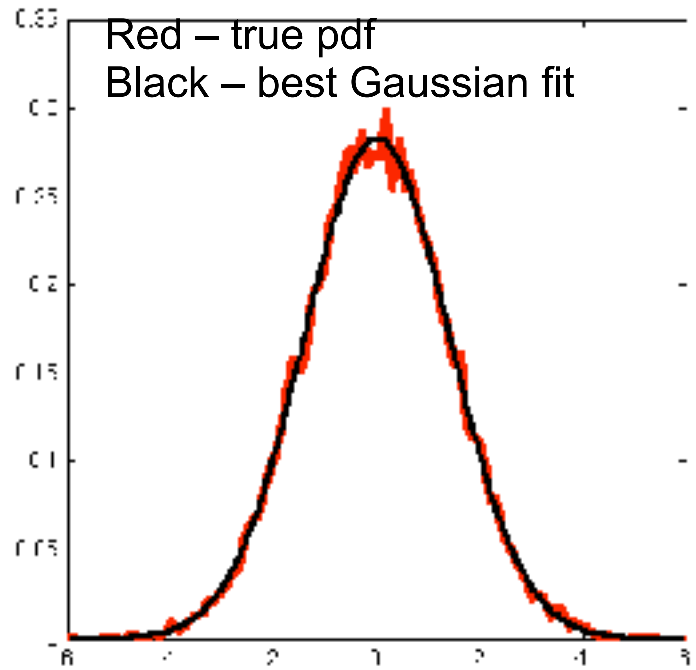


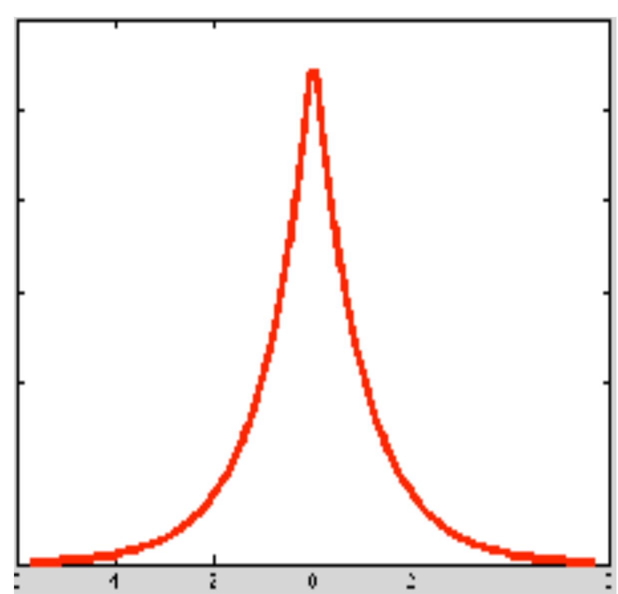
Image
Intensity histogram
[1 -1] filter output
[1 -1] output histogram
Red – true pdf
Black – best Gaussian fit
0.01
0.02
0.03
0.05
-100
0
100

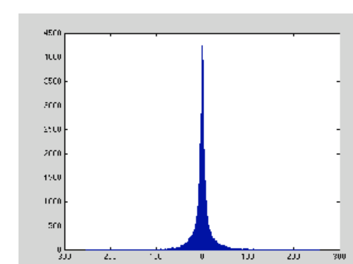
A model for the distribution of filter outputs





$$p(x) = \frac{\exp(-x^2/2\sigma^2)}{\sqrt{2\pi\sigma^2}}$$





$$p(x) = \frac{\exp(-|x/s|^r)}{2s/r\Gamma(1/r)}$$

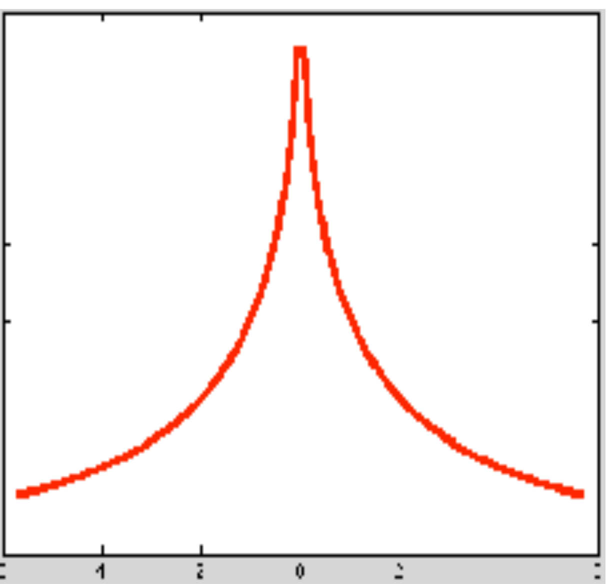
$r \sim 0.8$ (< 2)

Note: this is not a good model for ALL filter outputs

Generalized Gaussian

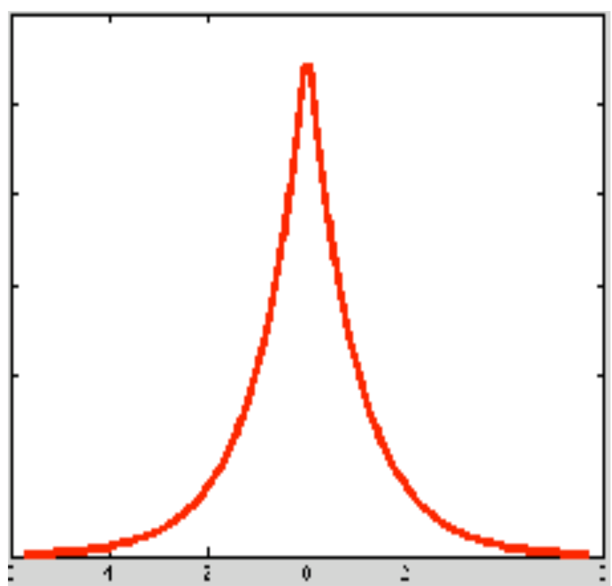
$$p(x) = \frac{\exp(-|x/s|^r)}{2s/r\Gamma(1/r)}$$

r = 0.5



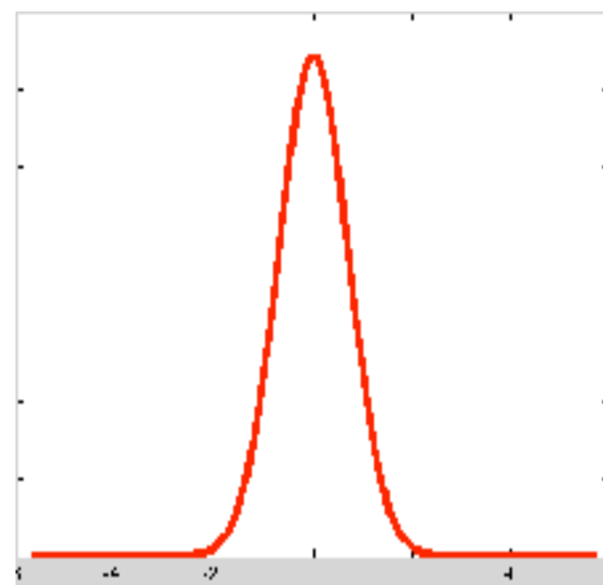
r = 1

Laplacian distribution

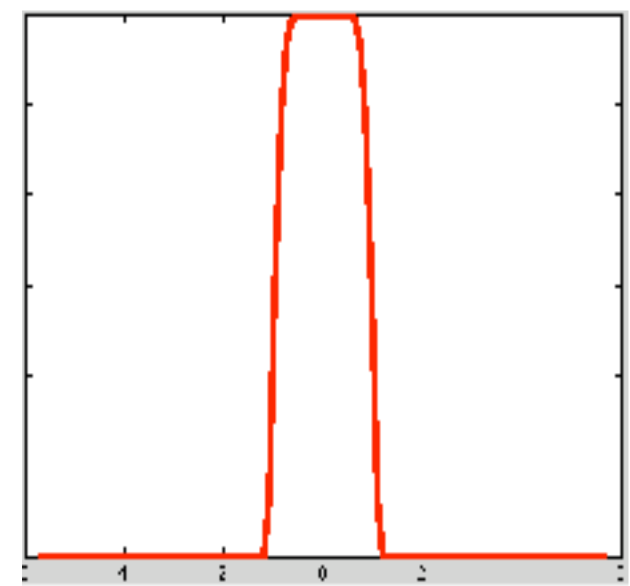


r = 2

Gaussian distribution



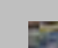
r = 10



Uniform distribution
r -> infinite

The wavelet marginal model

A small neighborhood

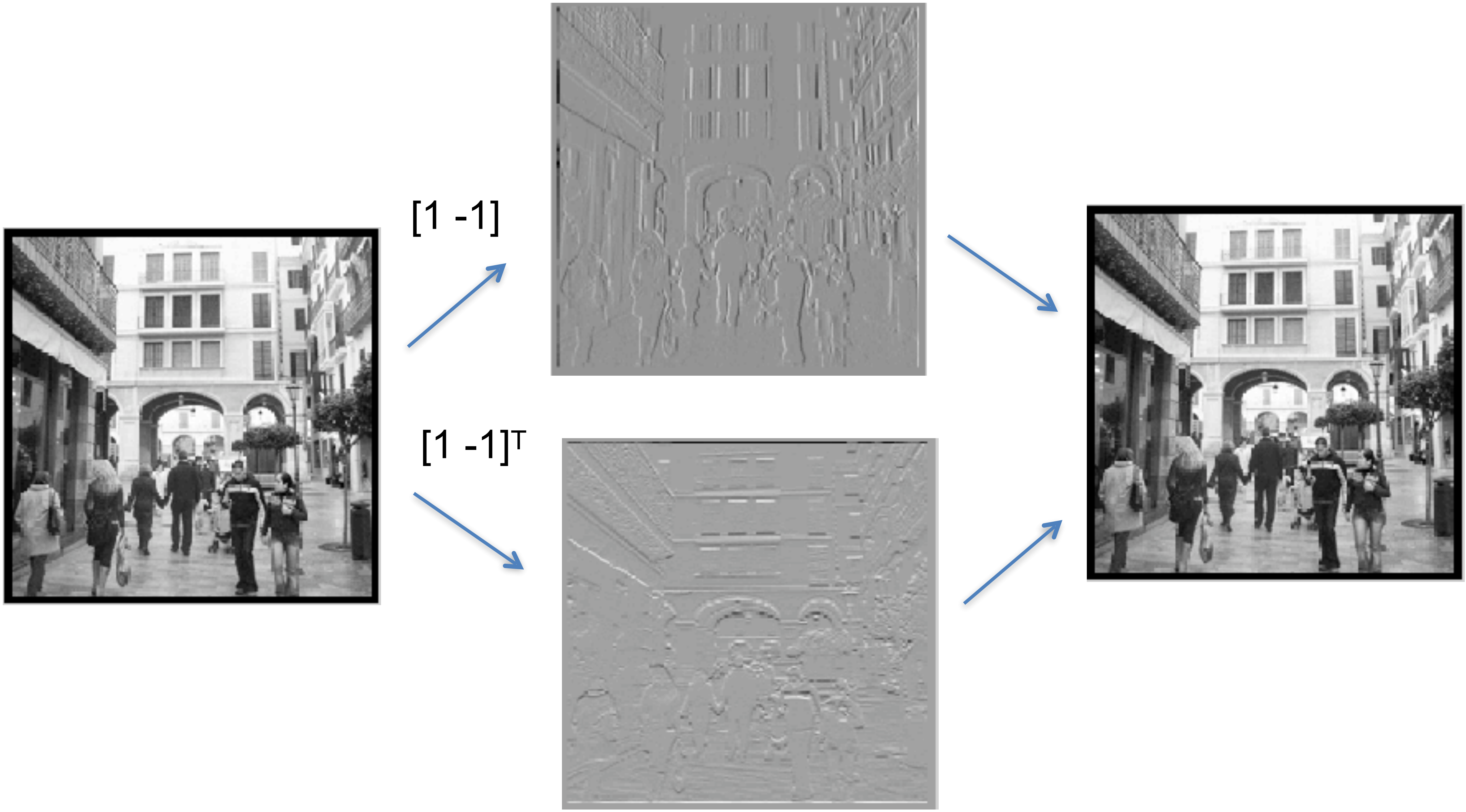


$$p(\mathbf{I}) = \prod_{k} \prod_{x,y} p(h_k(x,y))$$

All pixels and all outputs are independent

Filter outputs

The wavelet marginal model



$$p(\mathbf{I}) = \prod_{k} \prod_{x,y} p(h_k(x,y))$$

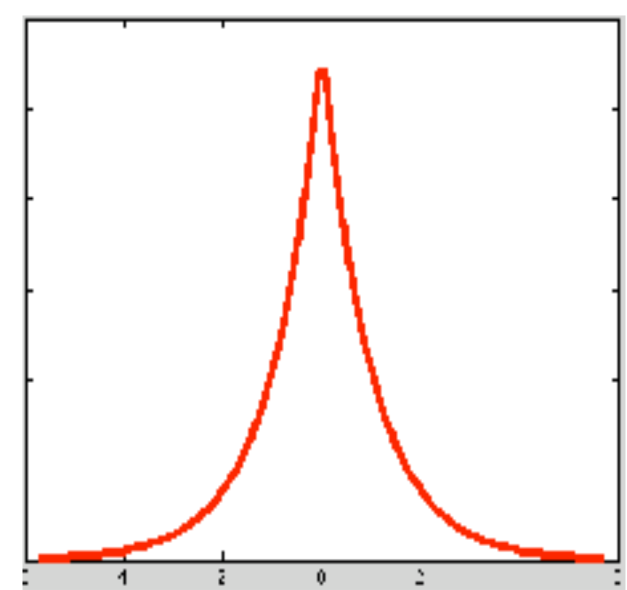
What is the most probable image under the wavelet marginal model?

[1 -1]

[1 -1]^T

$$p(\mathbf{I}) = \prod_{k} \prod_{x,y} p(h_k(x,y))$$

$$p(x) = \frac{\exp(-|x/s|^r)}{2s/r\Gamma(1/r)}$$



Sampling images

Gaussian model

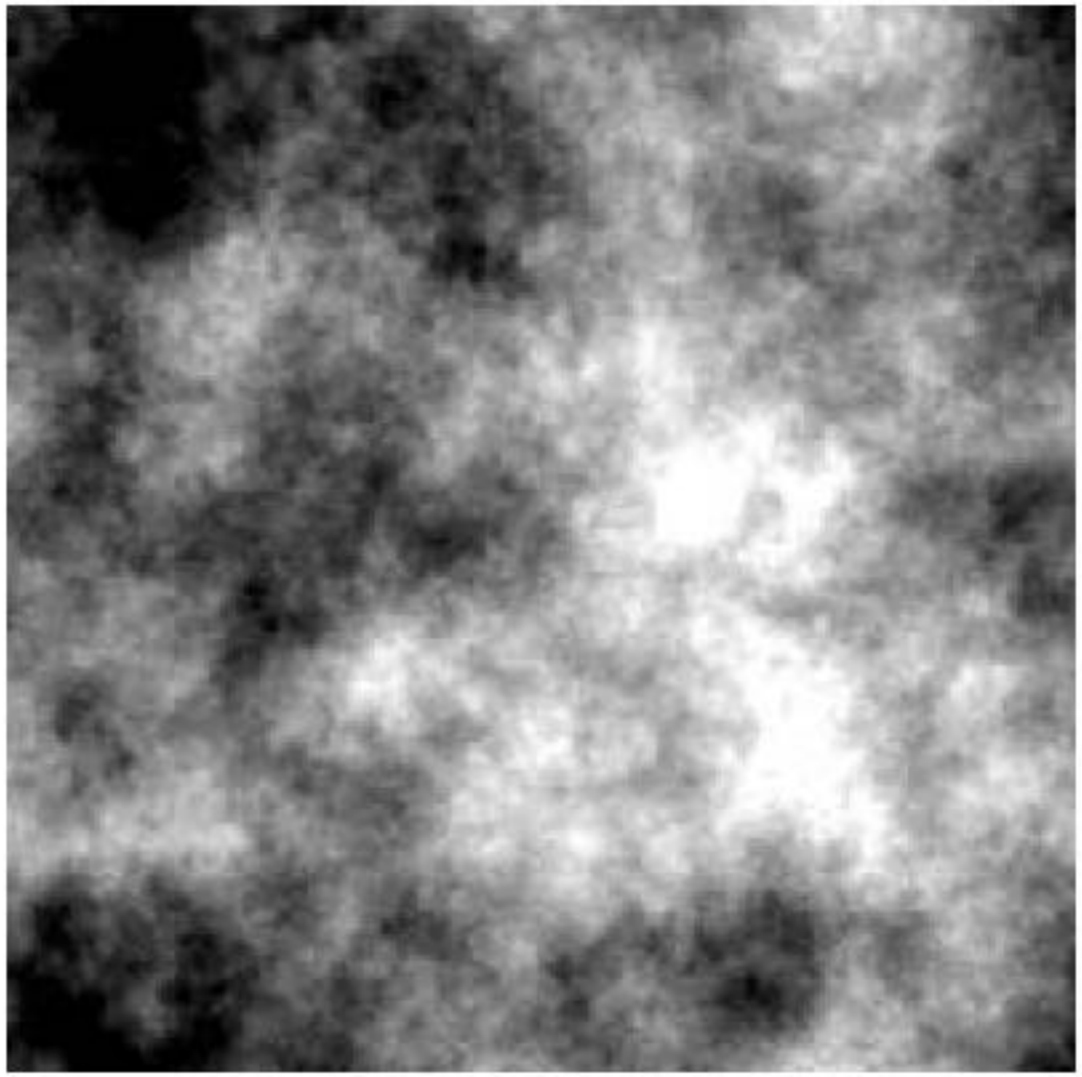


Fig. 3. Example image randomly drawn from the Gaussian spectral model, with $\gamma = 2.0$.

Wavelet marginal model

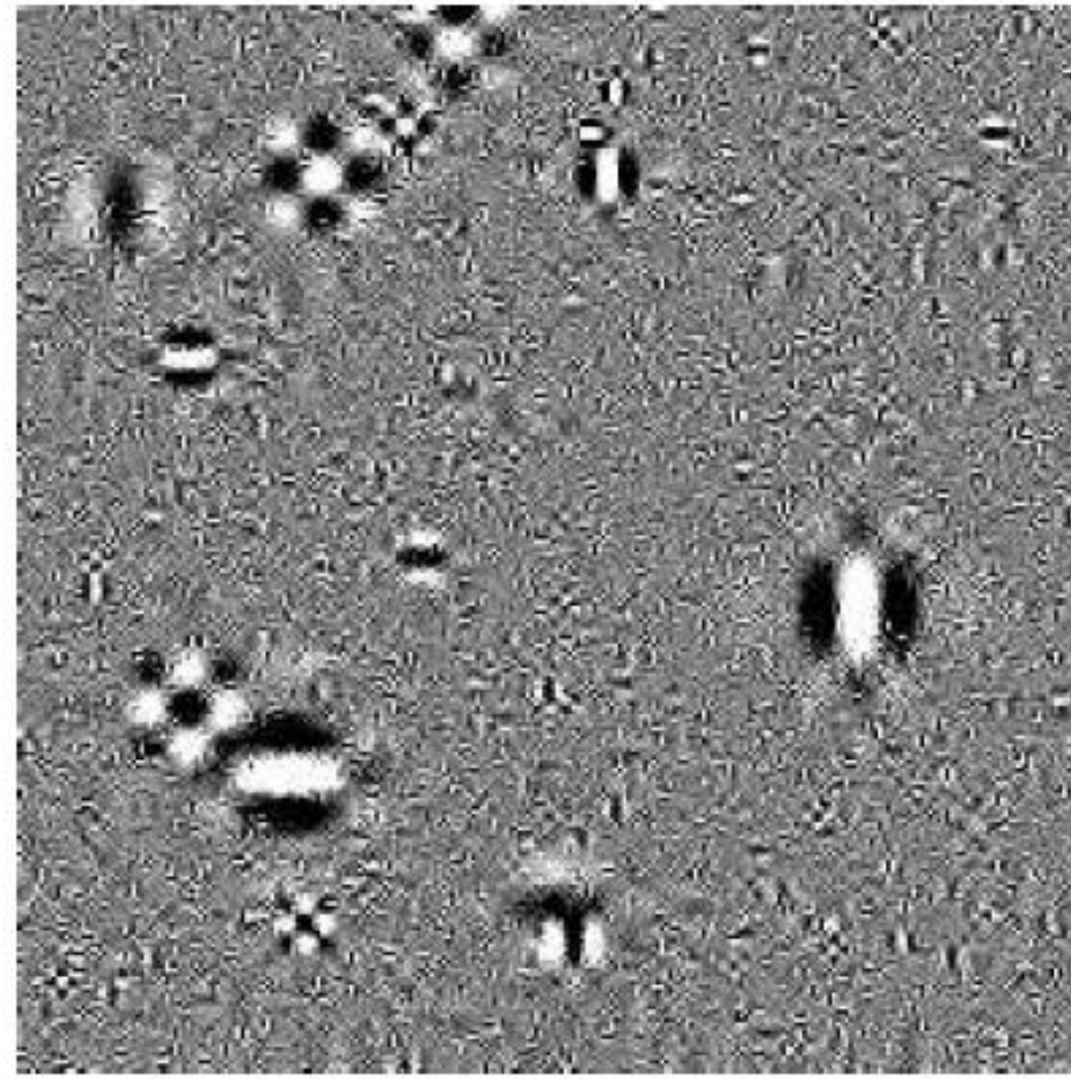
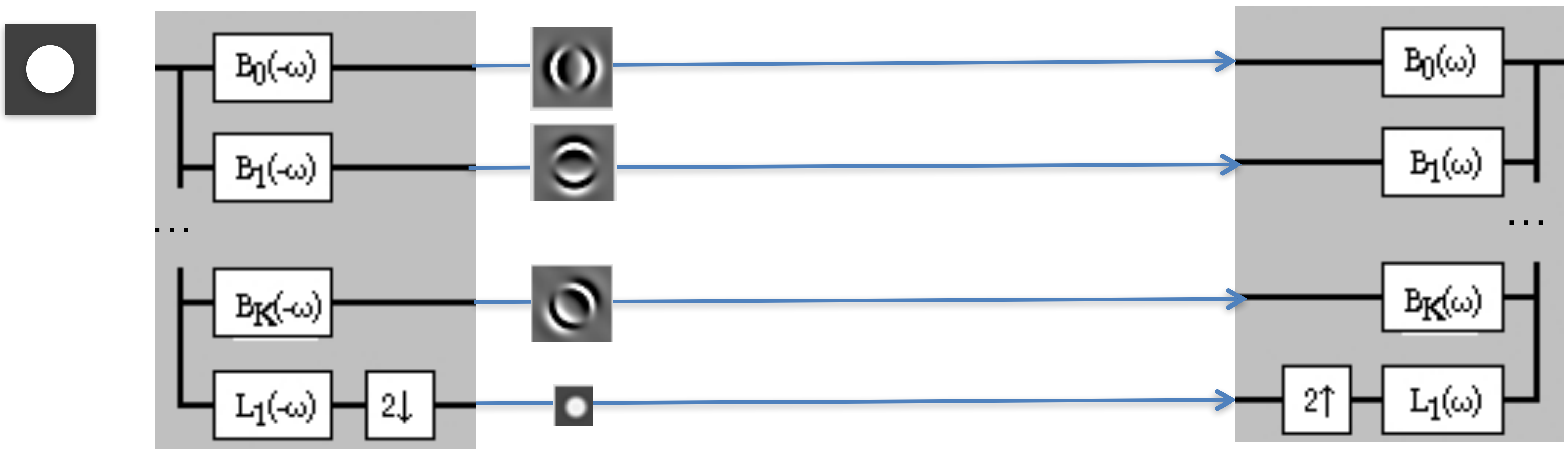


Fig. 6. A sample image drawn from the wavelet marginal model, with subband density parameters chosen to fit the image of Fig. 7.

Steerable Pyramid

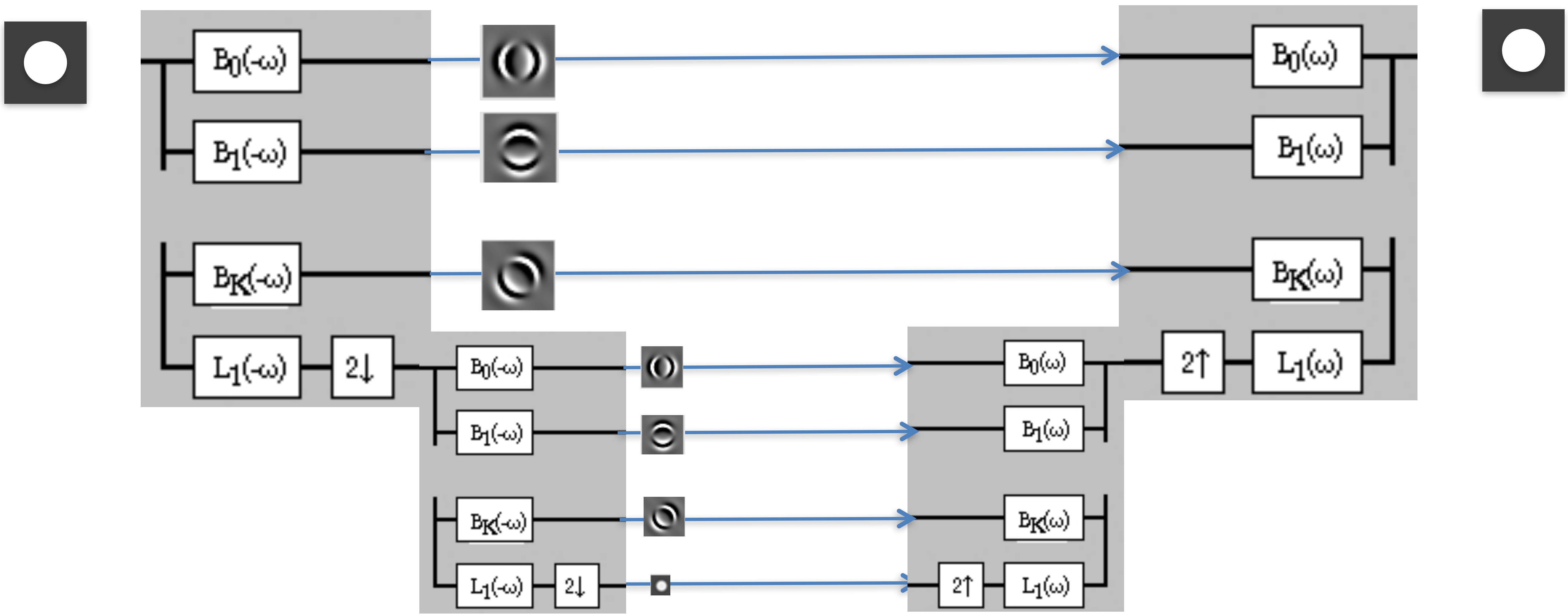
Decomposition **Reconstruction**



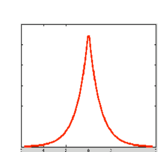
Images from: http://www.cis.upenn.edu/~eero/steerpyr.html

Steerable Pyramid

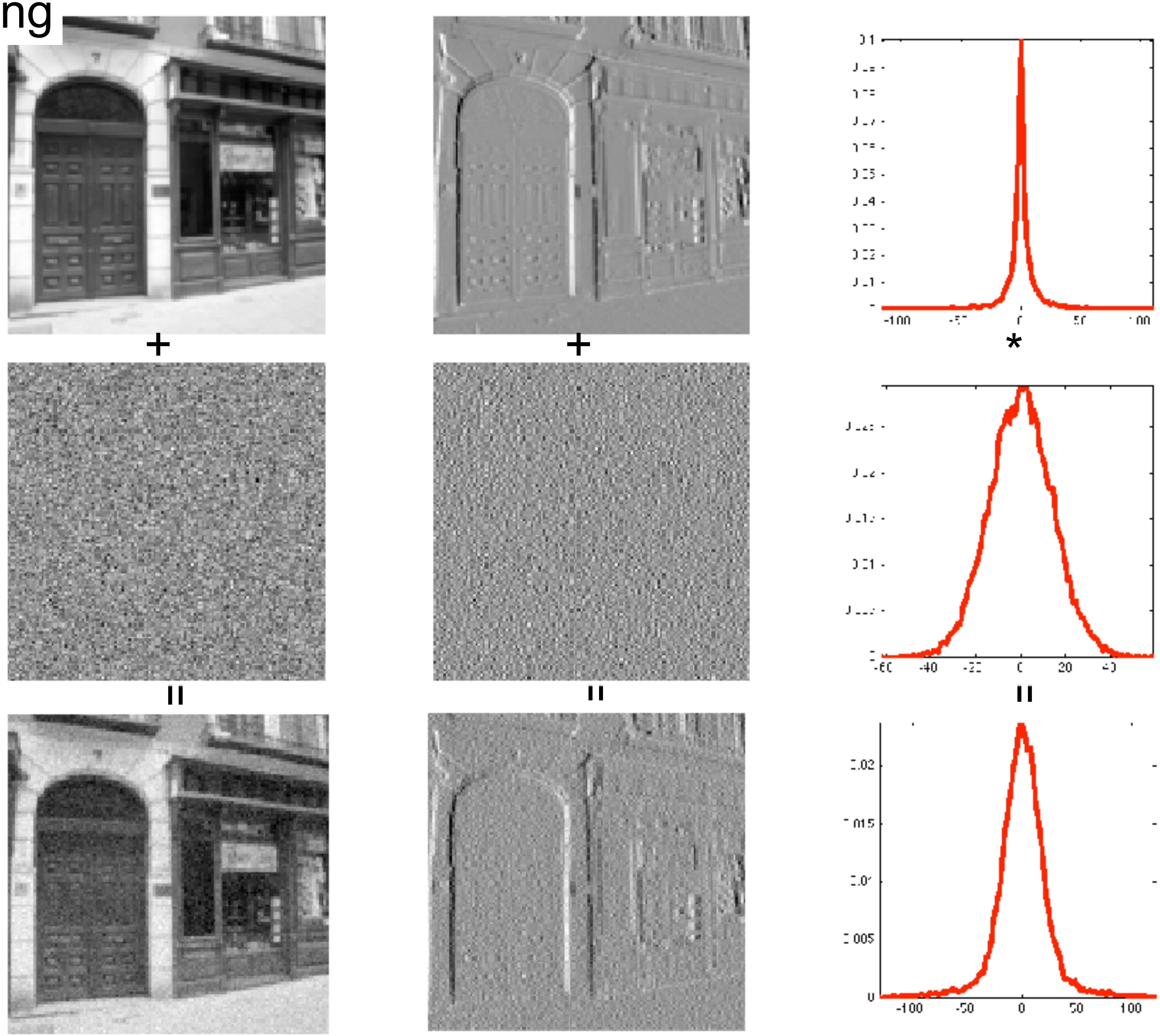
Decomposition **Reconstruction**



$$p(\mathbf{I}) = \prod_{k} \prod_{x,y} p(h_k(x,y))$$



Denoising



White Gaussian noise

Noisy image

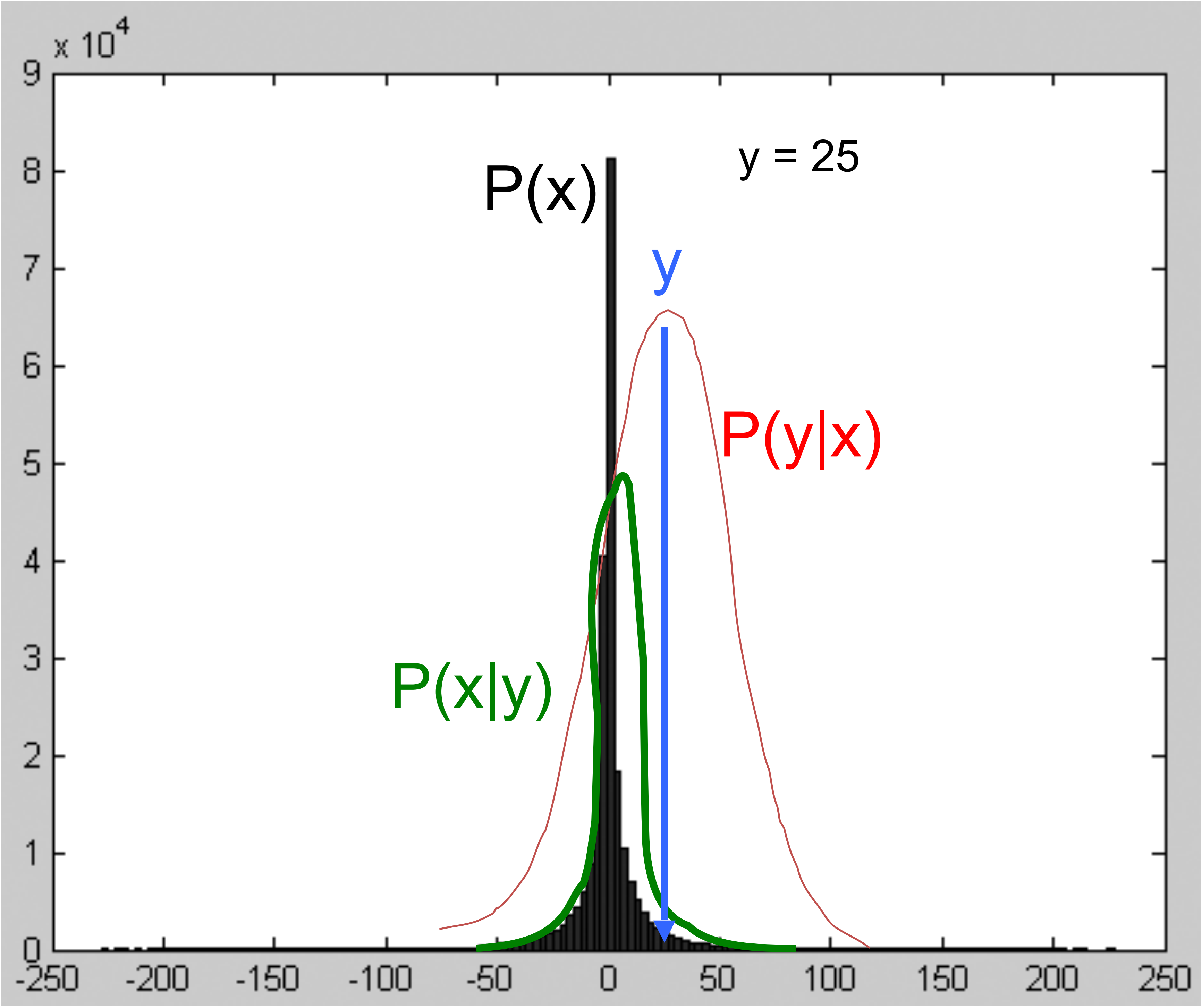
Denoising with the marginal wavelet model

Let y = noise-corrupted observation: y = x+n, with n ~ gaussian.

Let x = bandpassed image value before adding noise.

By Bayes theorem

P(x|y) ~ P(y|x) P(x)

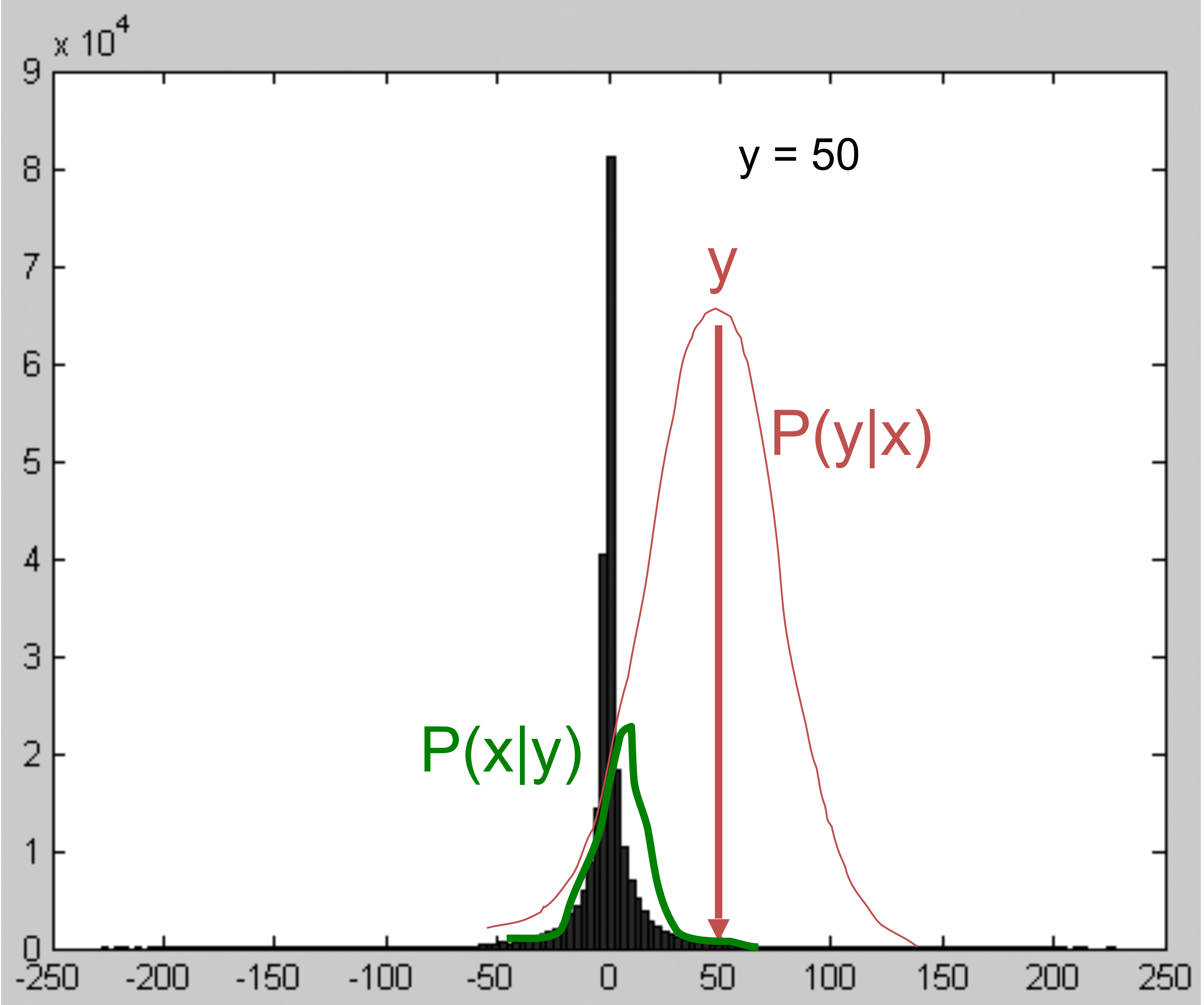


Denoising with the marginal wavelet model

Let x = bandpassed image value before adding noise.
Let y = noise-corrupted observation.

By Bayes theorem

P(x|y) ~ P(y|x) P(x)

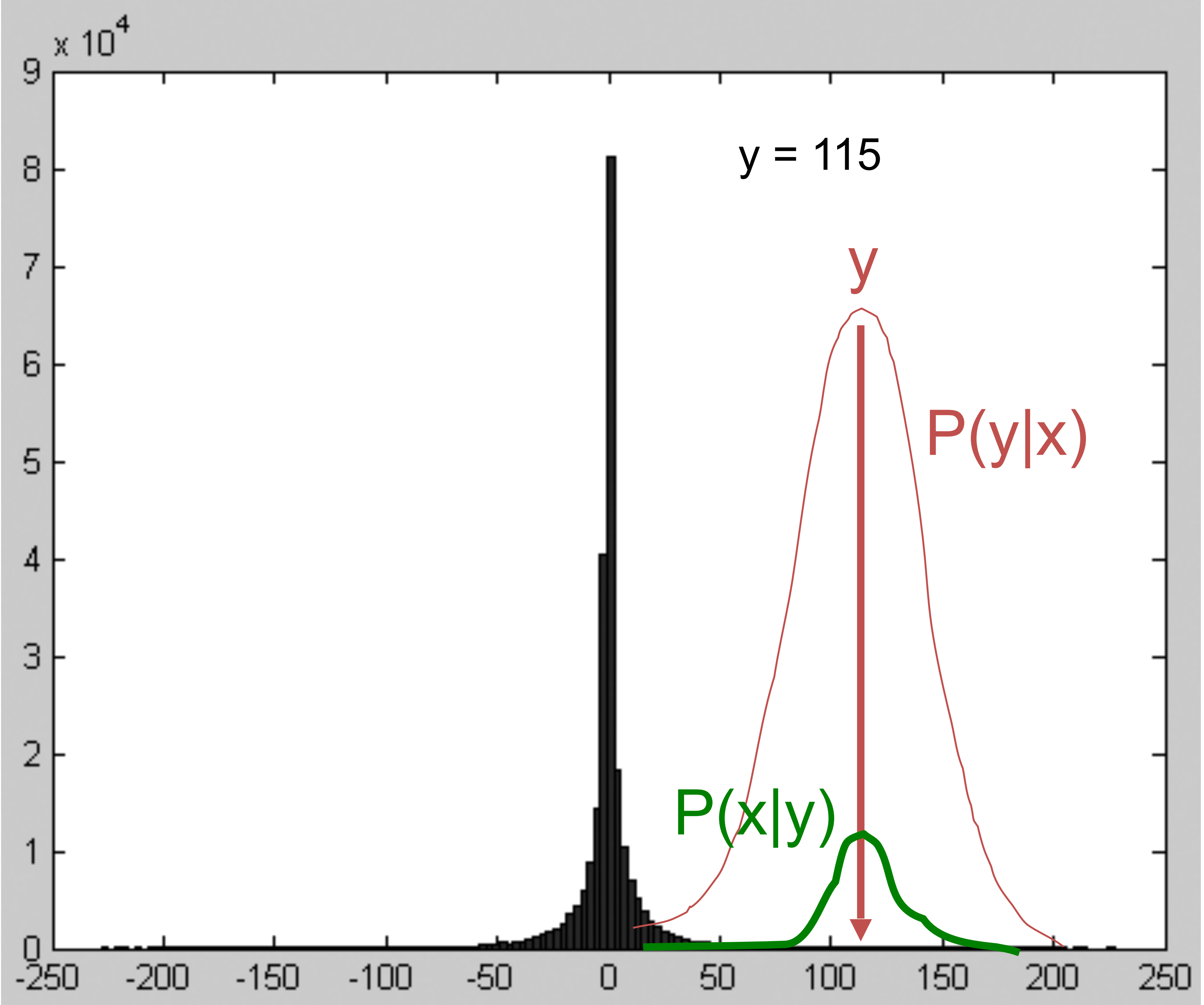


Denoising with the marginal wavelet model

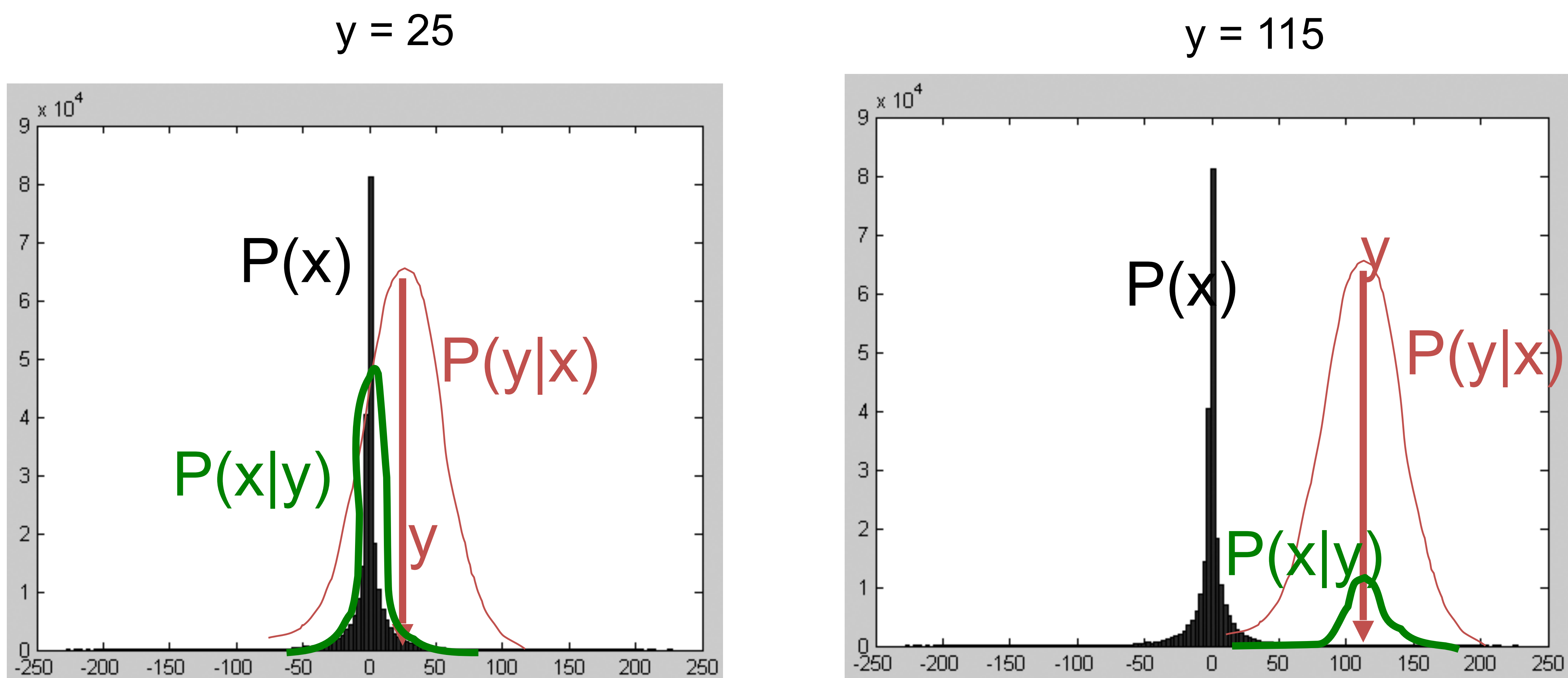
Let x = bandpassed image value before adding noise.
Let y = noise-corrupted observation.

By Bayes theorem

P(x|y) ~ P(y|x) P(x)



Denoising with the marginal wavelet model



For small y: probably it is due to noise and y should be set to 0
For large y: probably it is due to an image edge and it should be kept untouched

MAP estimate, $\hat{x}$, as function of observed coefficient value, y

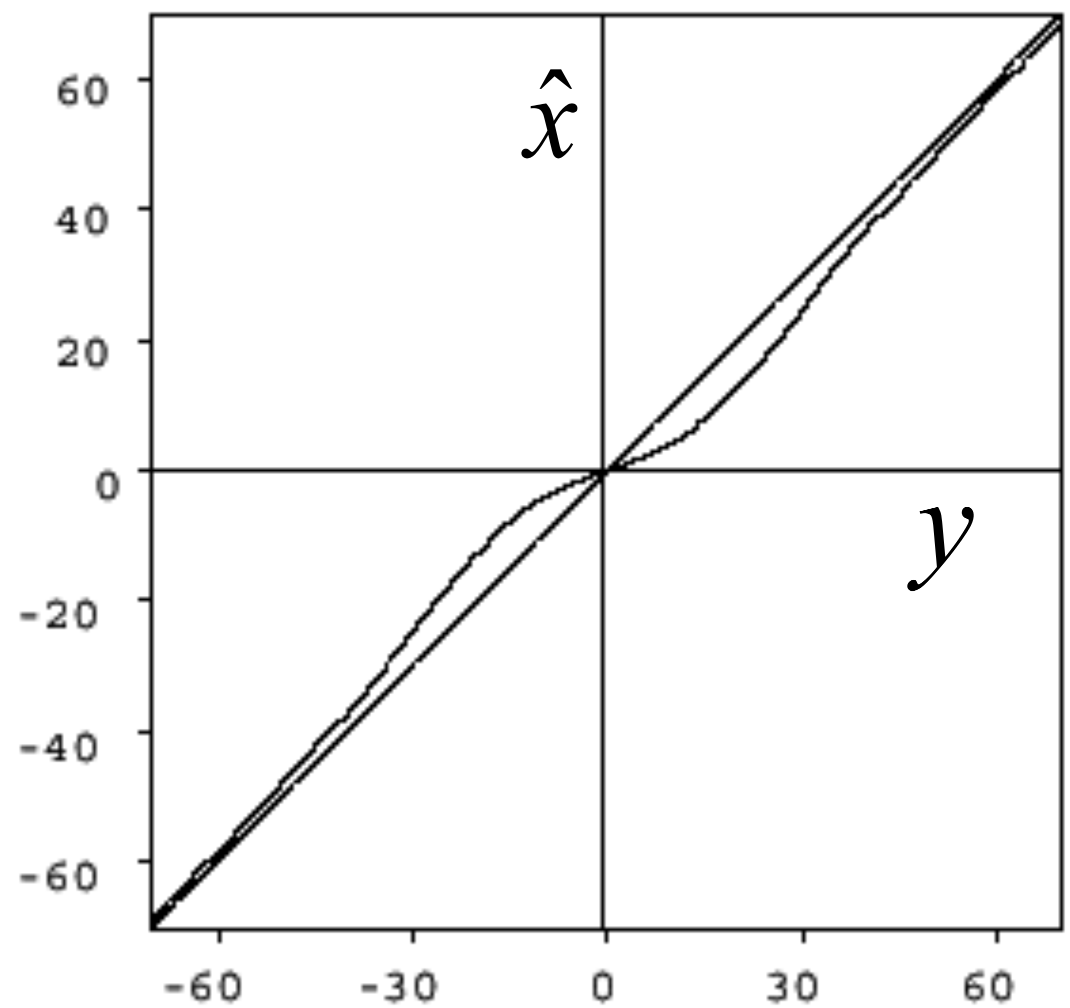
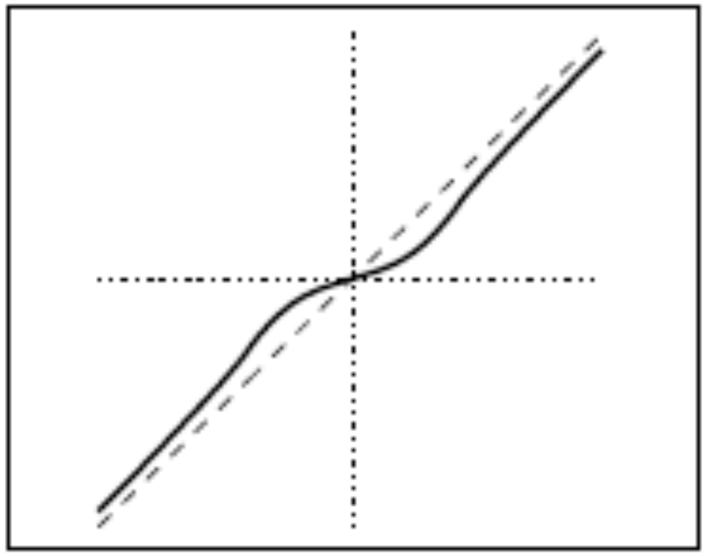
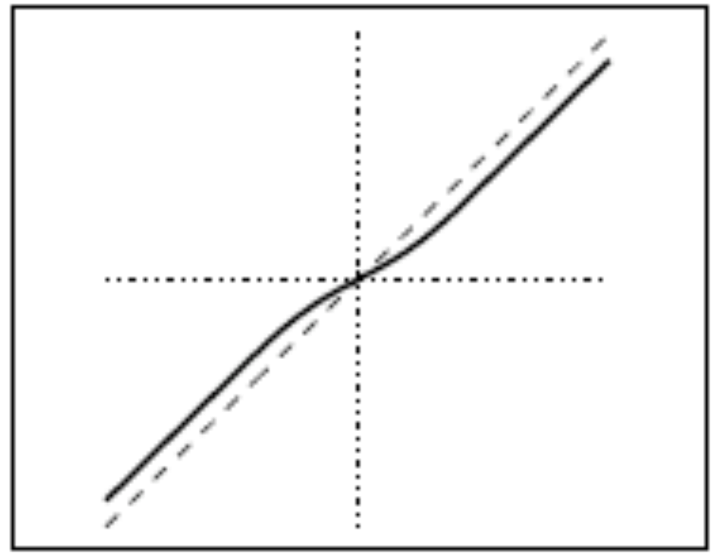


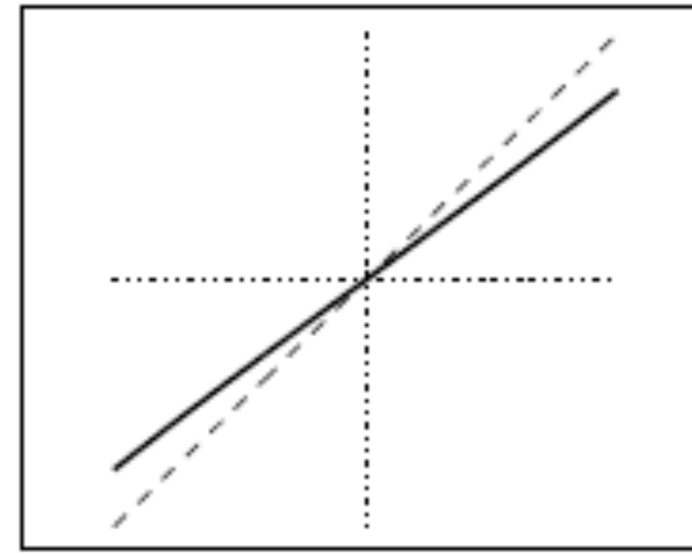
Figure 2: Bayesian estimator (symmetrized) for the signal and noise histograms shown in figure 1. Superimposed on the plot is a straight line indicating the identity function.

http://www-bcs.mit.edu/people/adelson/pub_pdfs/simoncelli_noise.pdf

Simoncelli and Adelson, Noise Removal via Bayesian Wavelet Coring



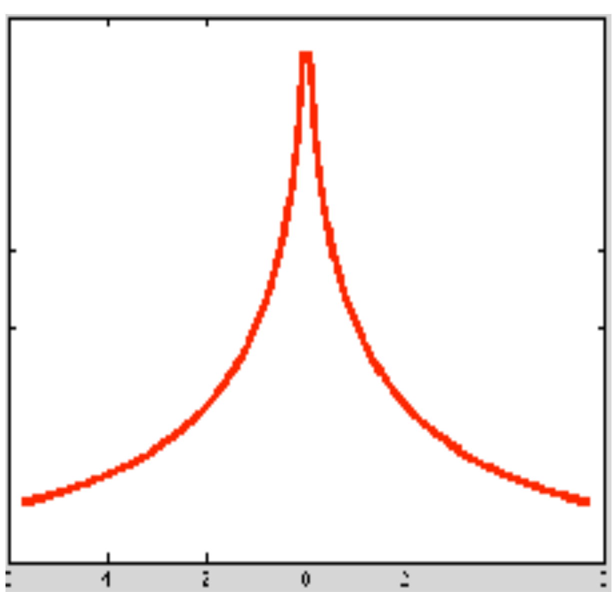


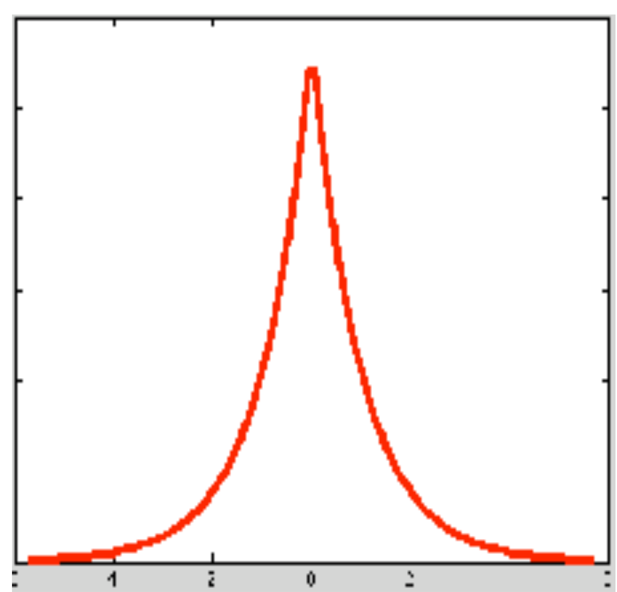


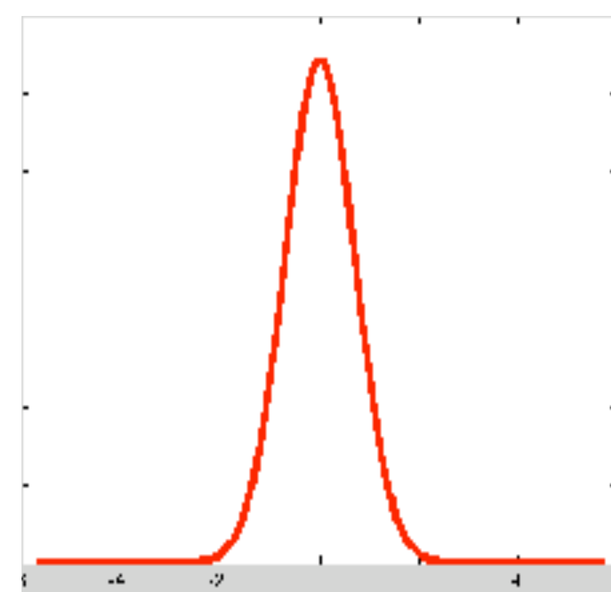
r = 0.5

r = 1
Laplacian distribution

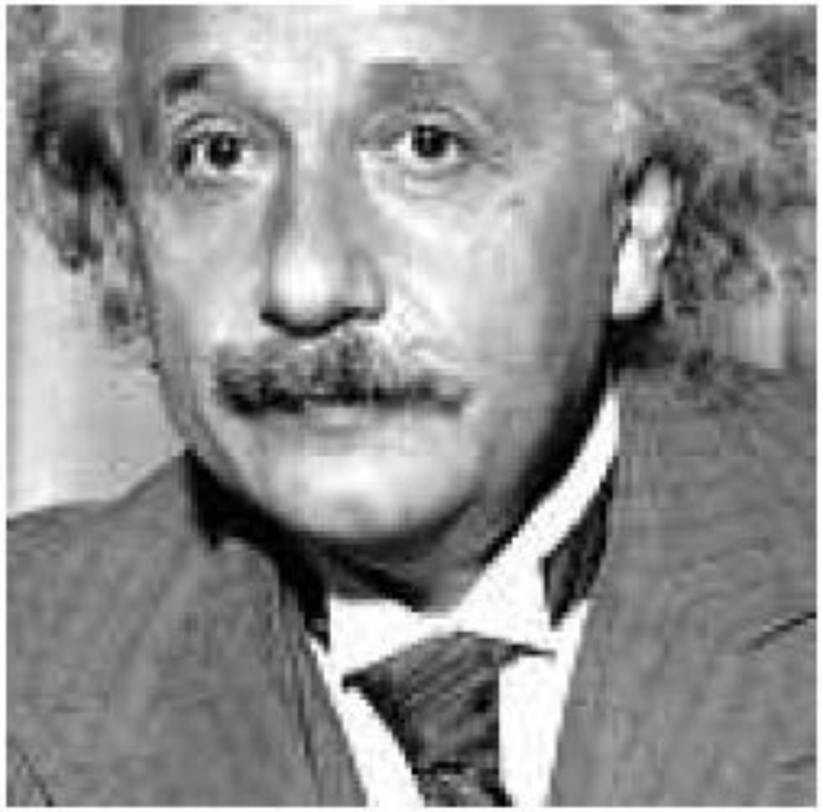
r = 2
Gaussian distribution



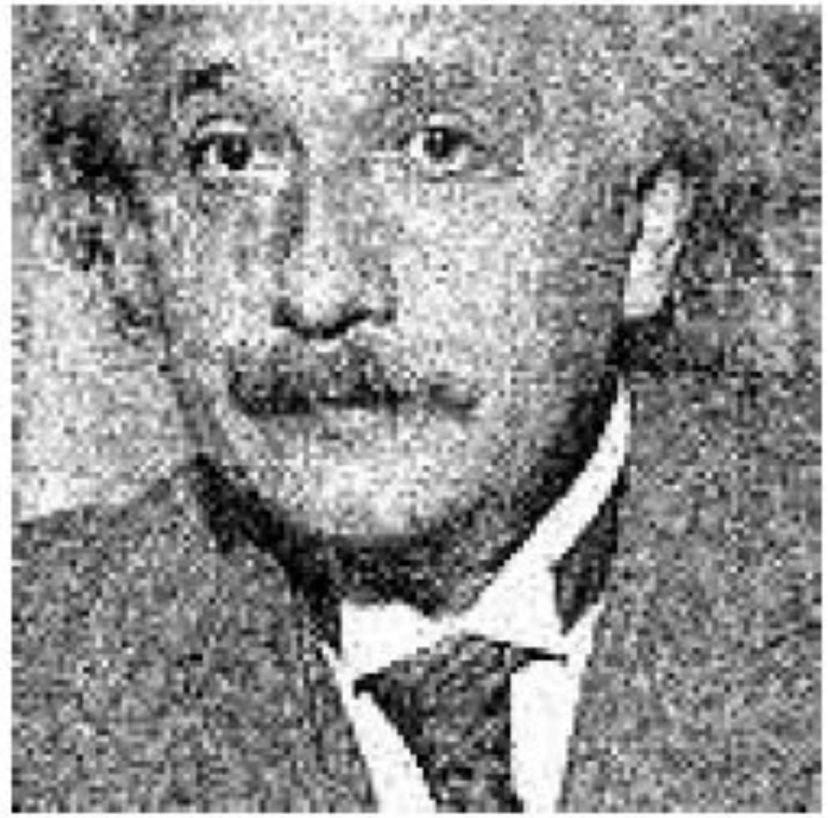




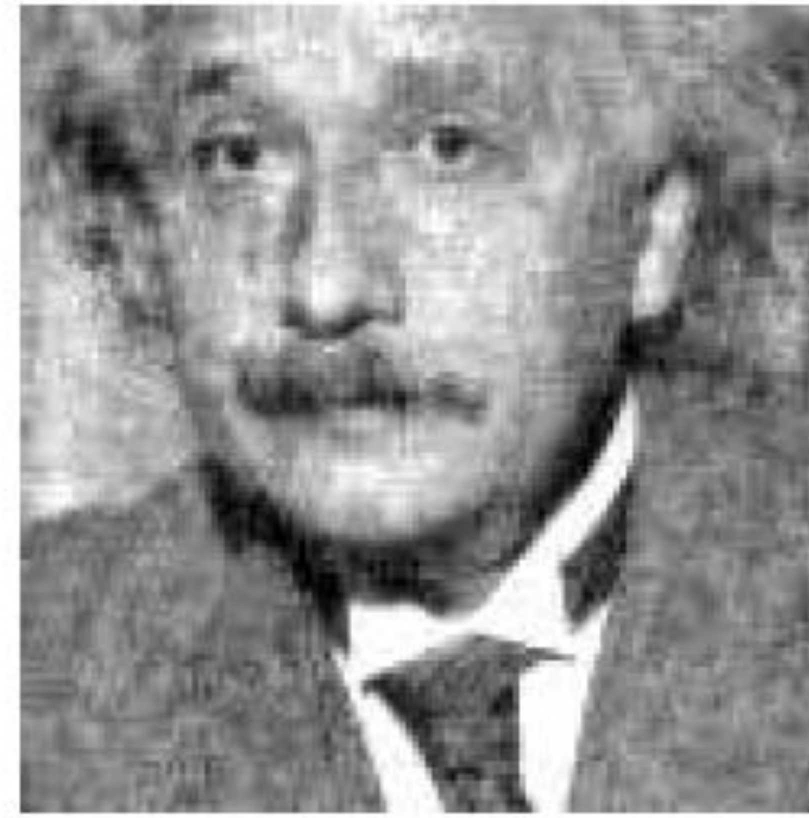
original

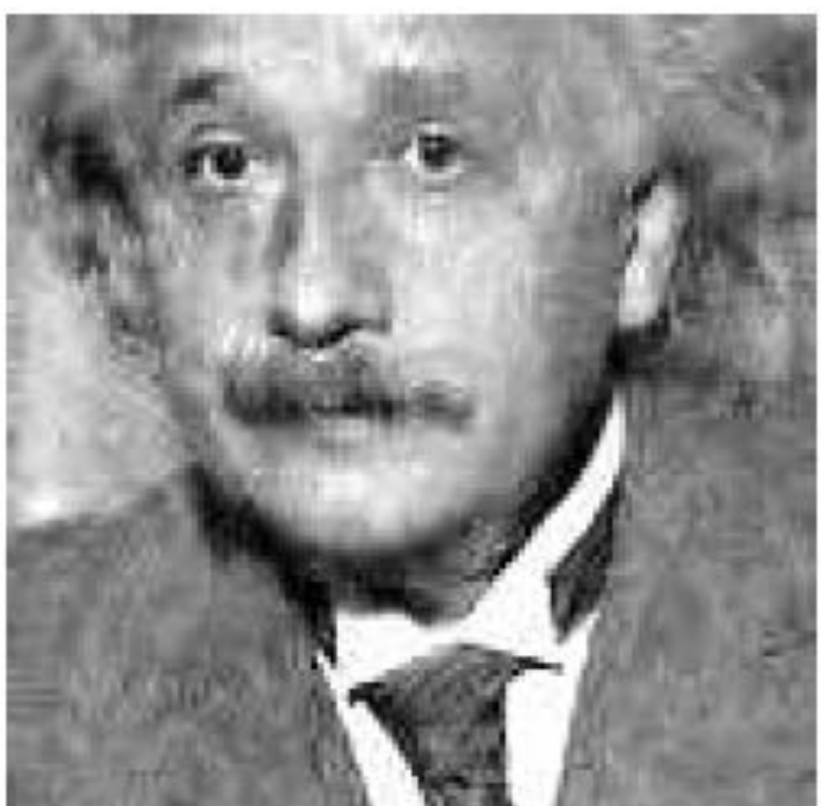


With Gaussian noise of std. dev. 21.4 added, giving PSNR=22.06



(1) Denoised with Gaussian model, PSNR=27.87





(2) Denoised with wavelet marginal model, PSNR=29.24

http://www.cns.nyu.edu/pub/eero/simoncelli05a-preprint.pdf

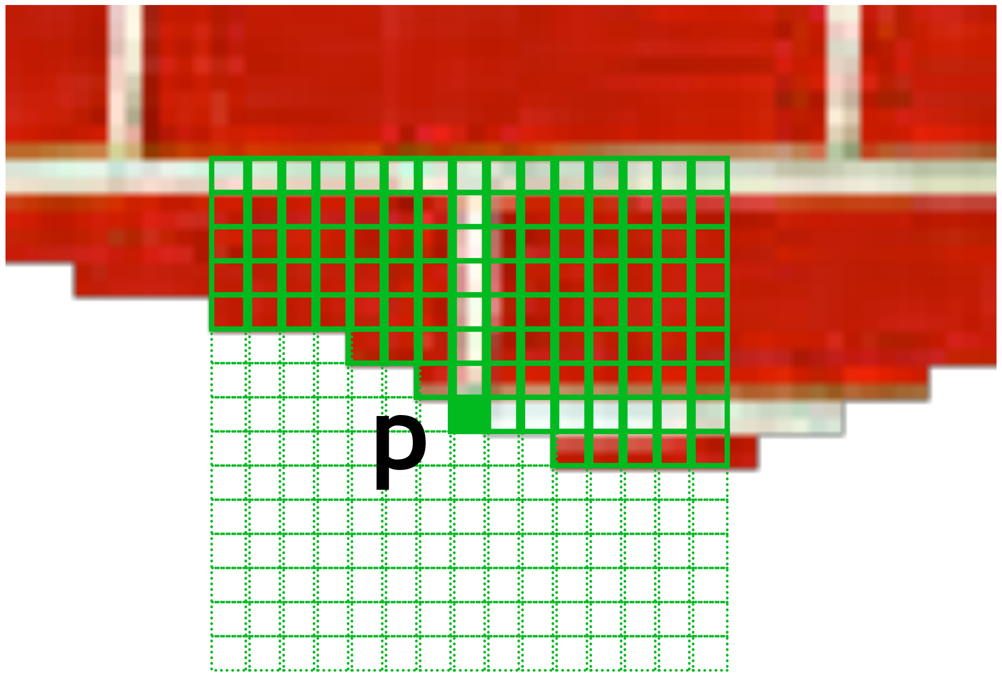
Non-parametric image modeling and noise removal

IEEE International Conference on Computer Vision, Corfu, Greece, September 1999

Texture Synthesis by Non-parametric Sampling

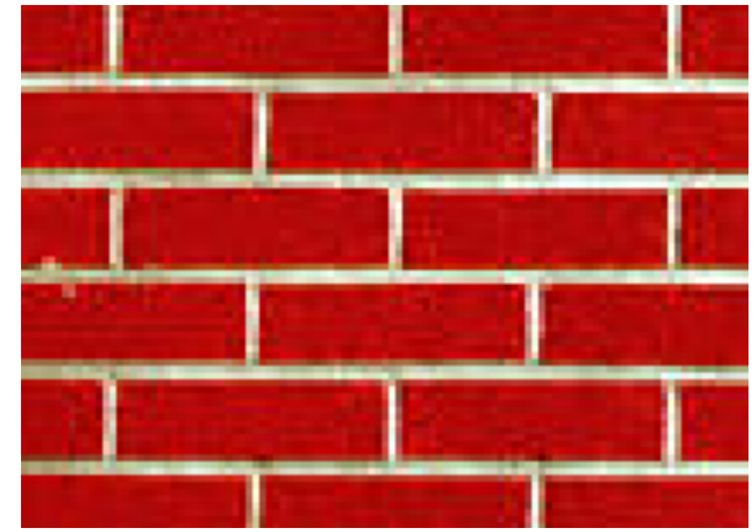
Alexei A. Efros and Thomas K. Leung
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720-1776, U.S.A.
{efros,leungt}@cs.berkeley.edu

Efros & Leung Algorithm



Synthesizing a pixel

non-parametric sampling



Input image

Assuming Markov property, compute P(**p**|N(**p**))

- – Building explicit probability tables is infeasible
- –Instead, we *search the input image* for all similar neighborhoods — that’s our pdf for **p**
- –To sample from this pdf, just pick one match at random

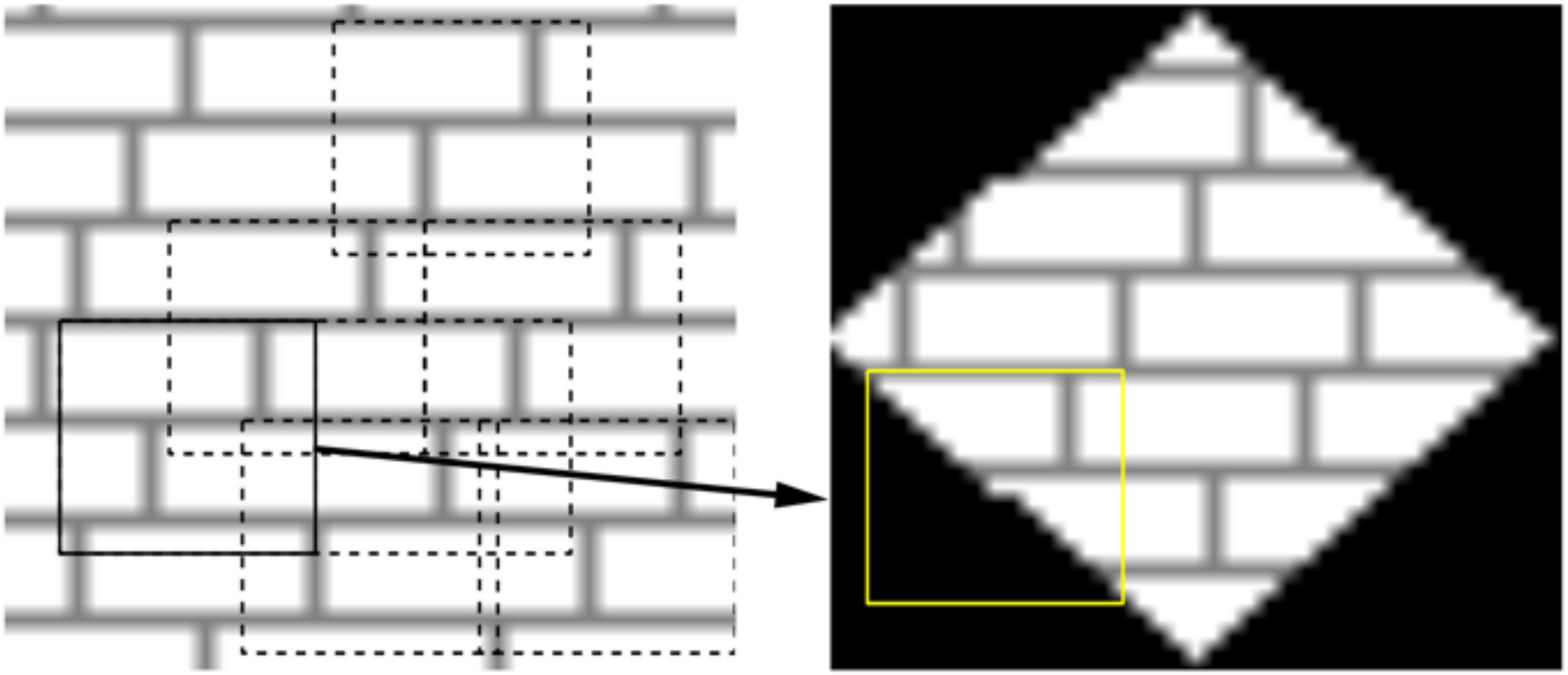
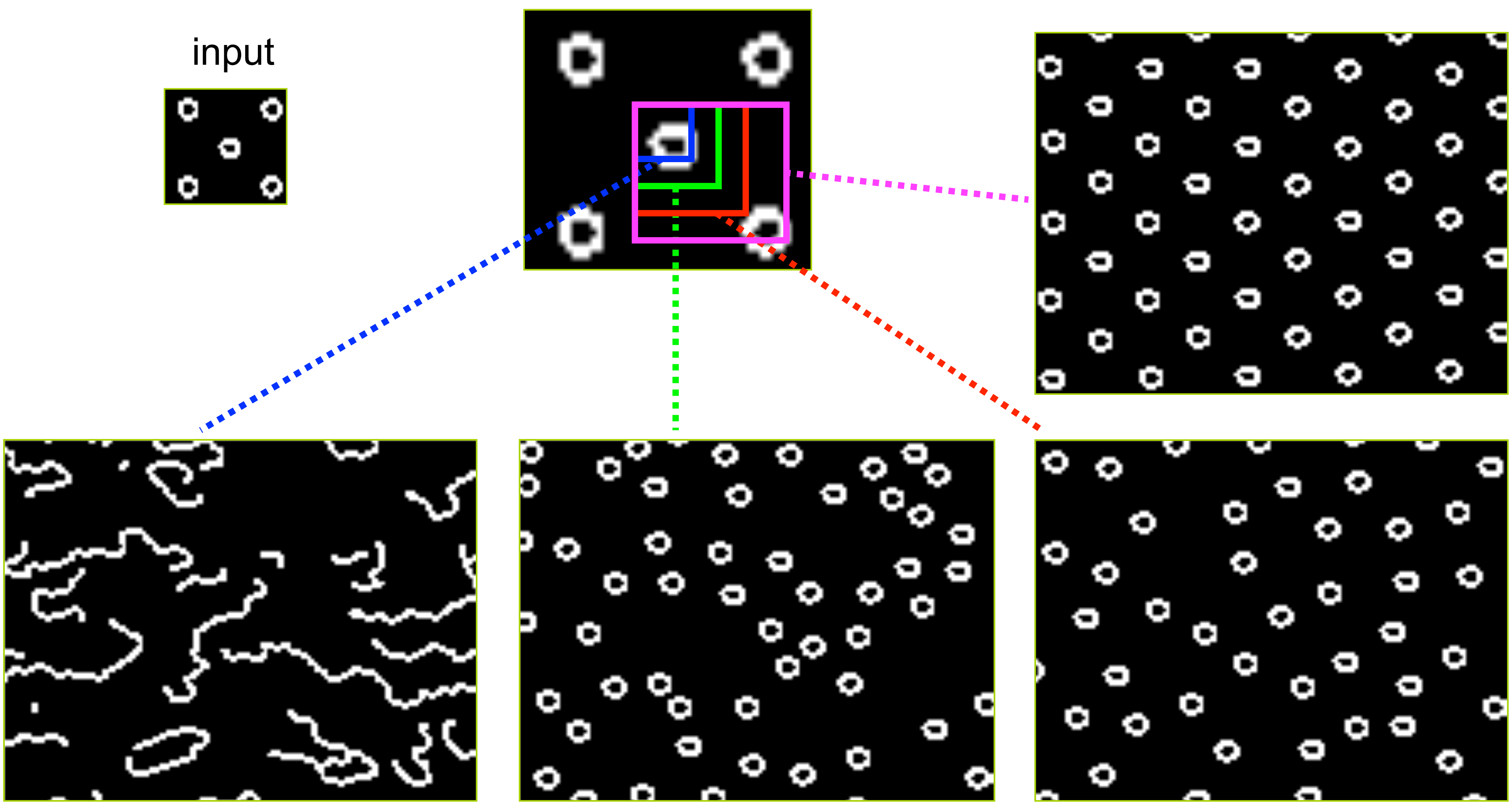
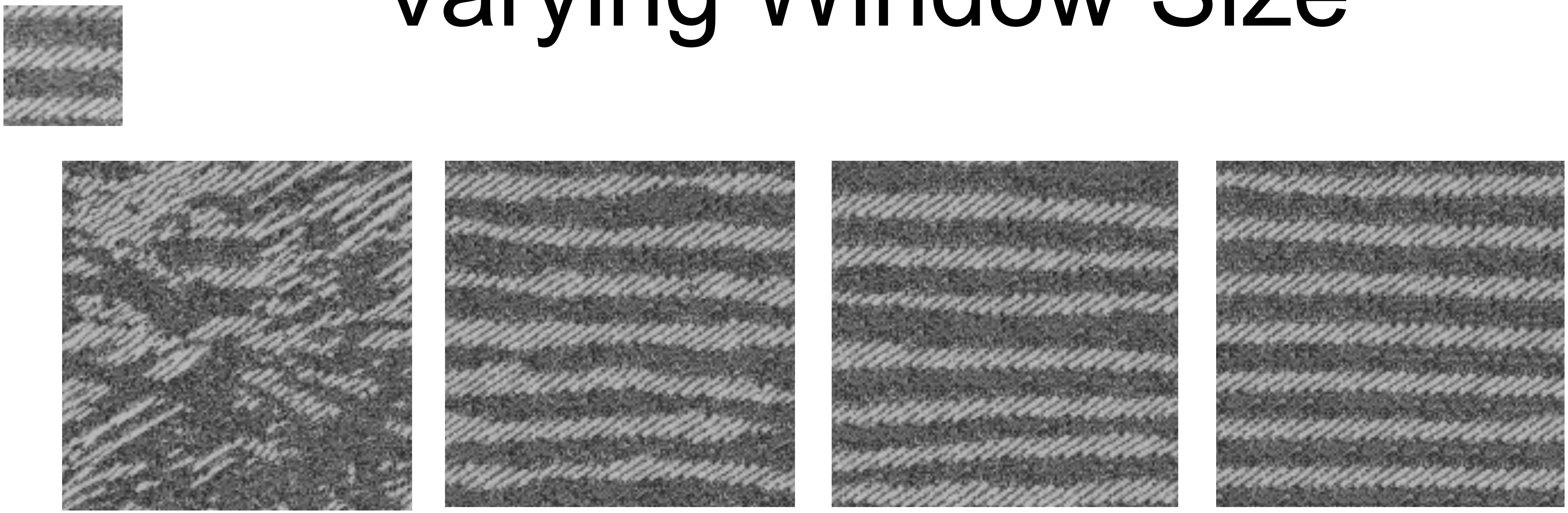


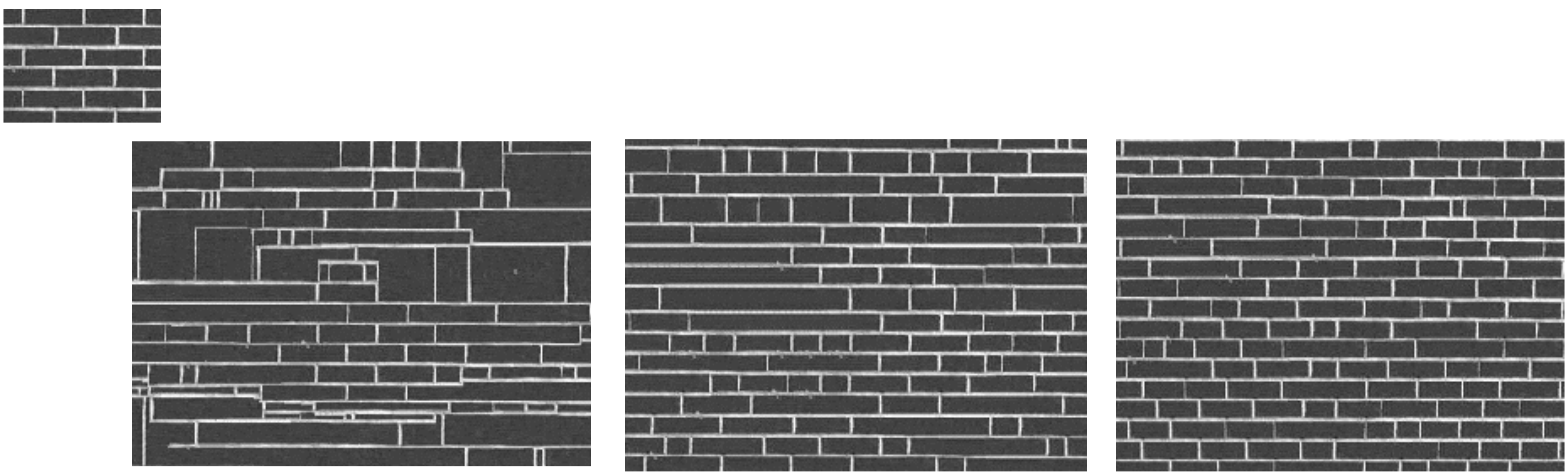
Figure 1. Algorithm Overview. Given a sample texture image (left), a new image is being synthesized one pixel at a time (right). To synthesize a pixel, the algorithm first finds all neighborhoods in the sample image (boxes on the left) that are similar to the pixel's neighborhood (box on the right) and then randomly chooses one neighborhood and takes its center to be the newly synthesized pixel.

Neighborhood Window



Varying Window Size

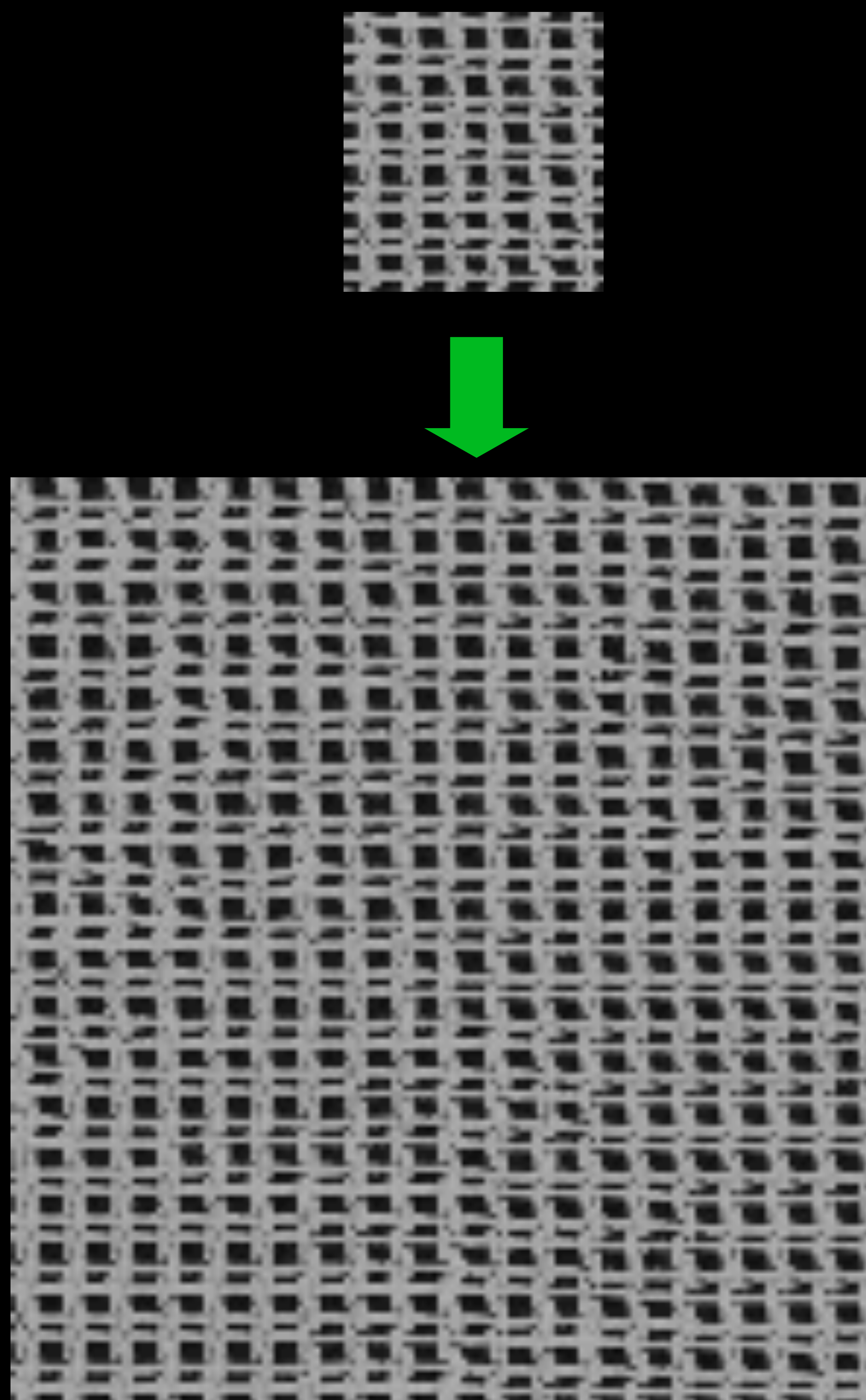




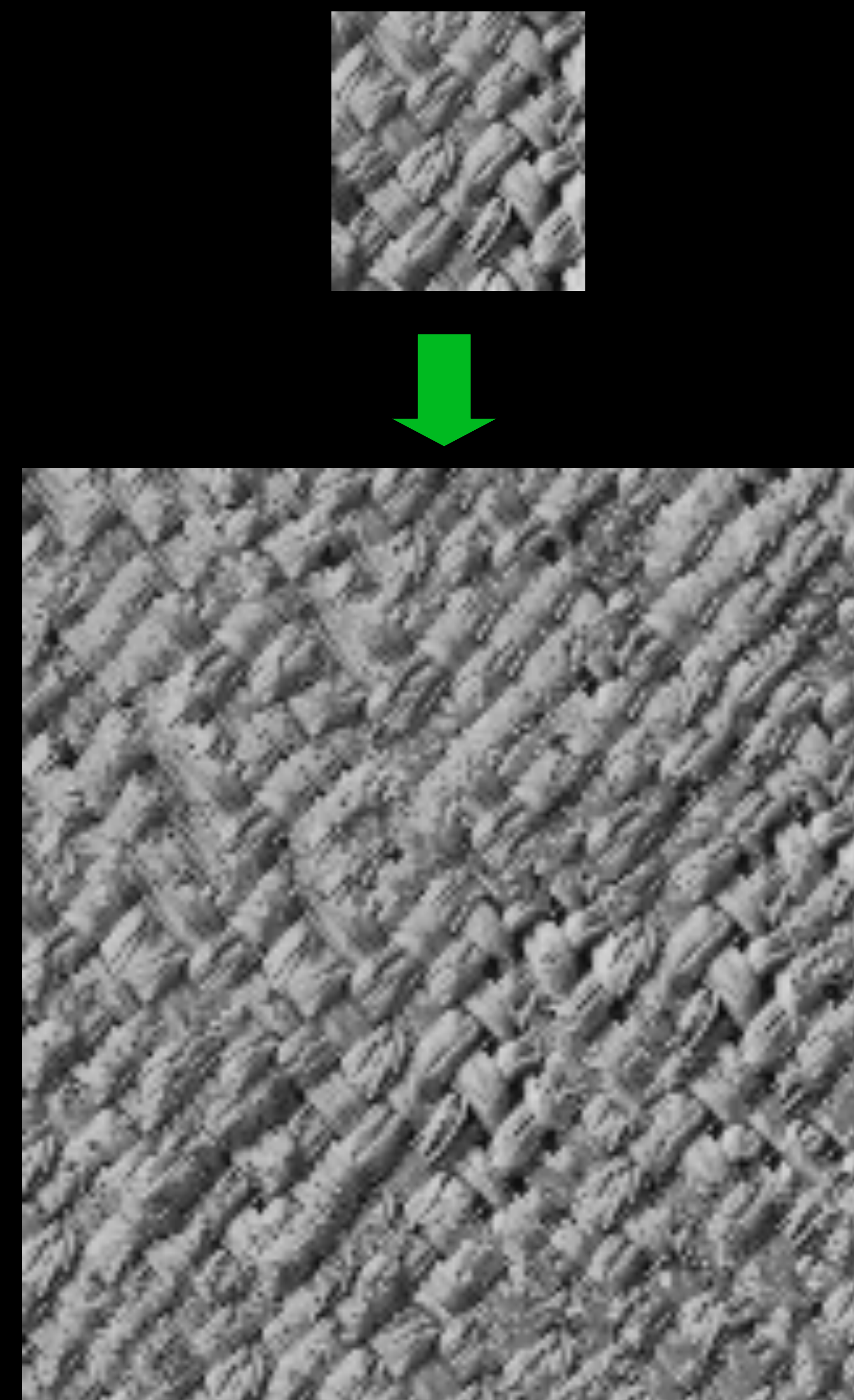
Increasing window size

Synthesis Results

french canvas

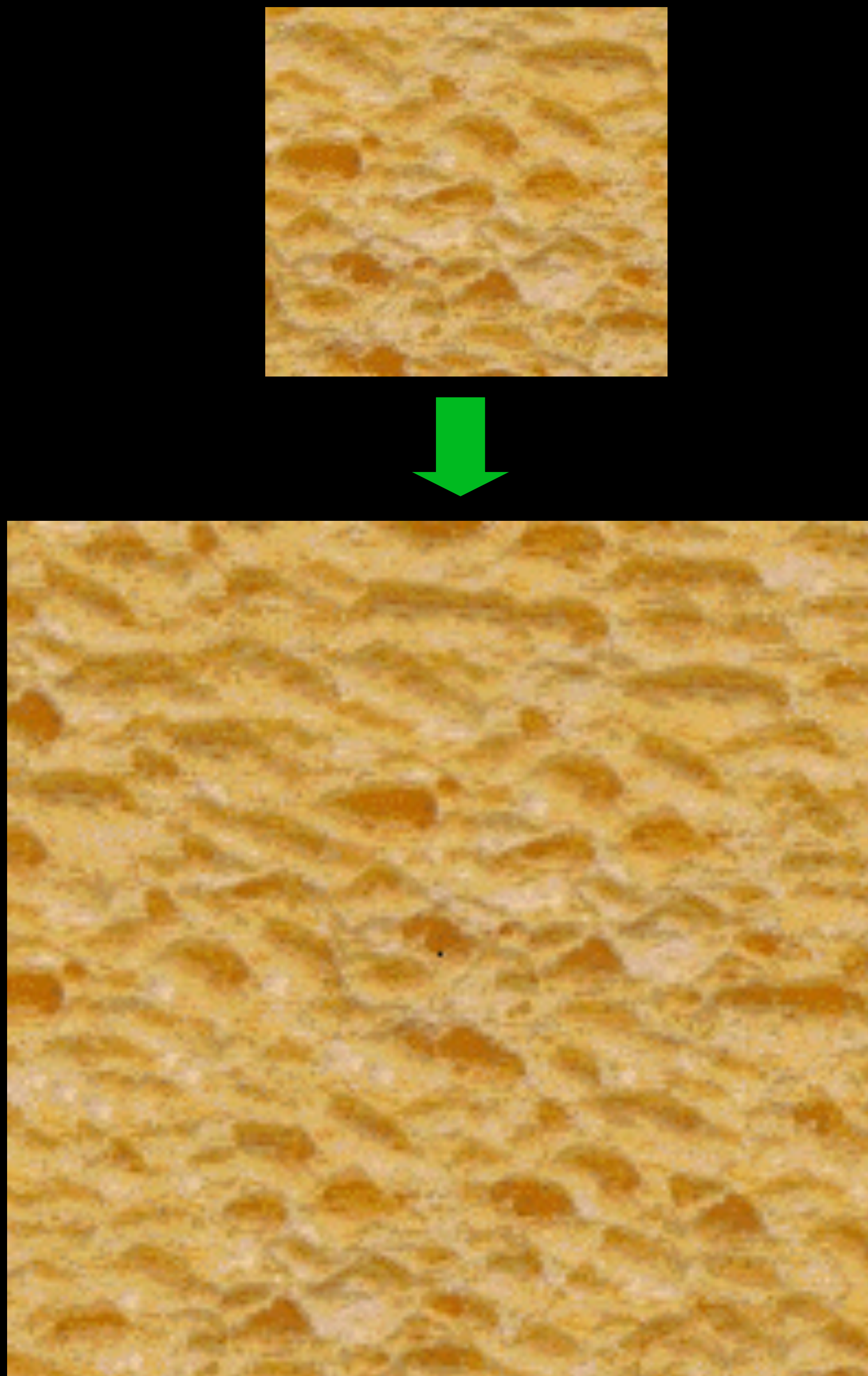


rafia weave

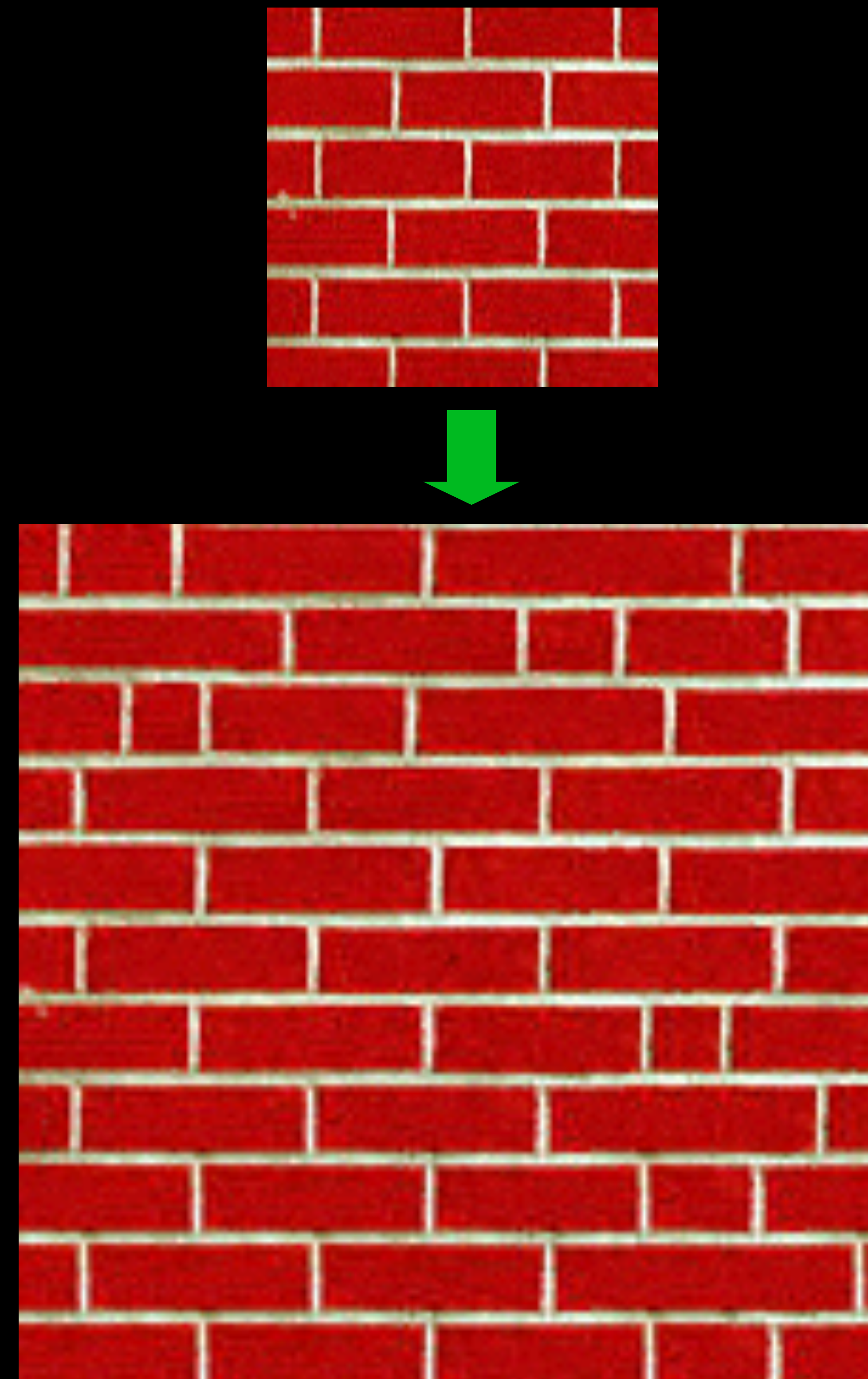


More Results

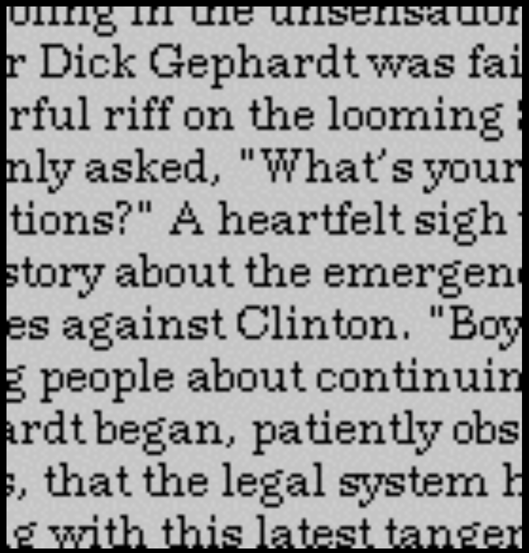
white bread

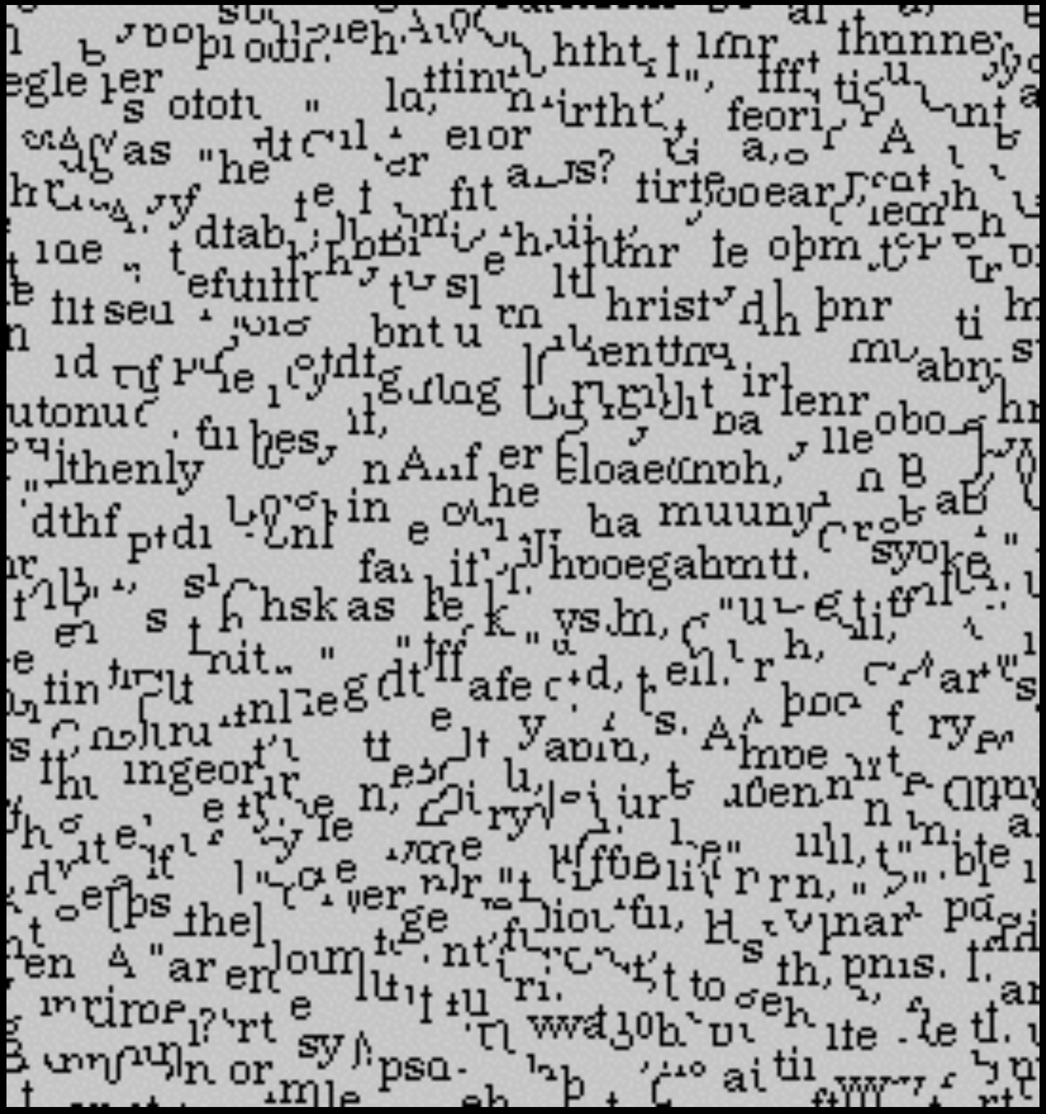


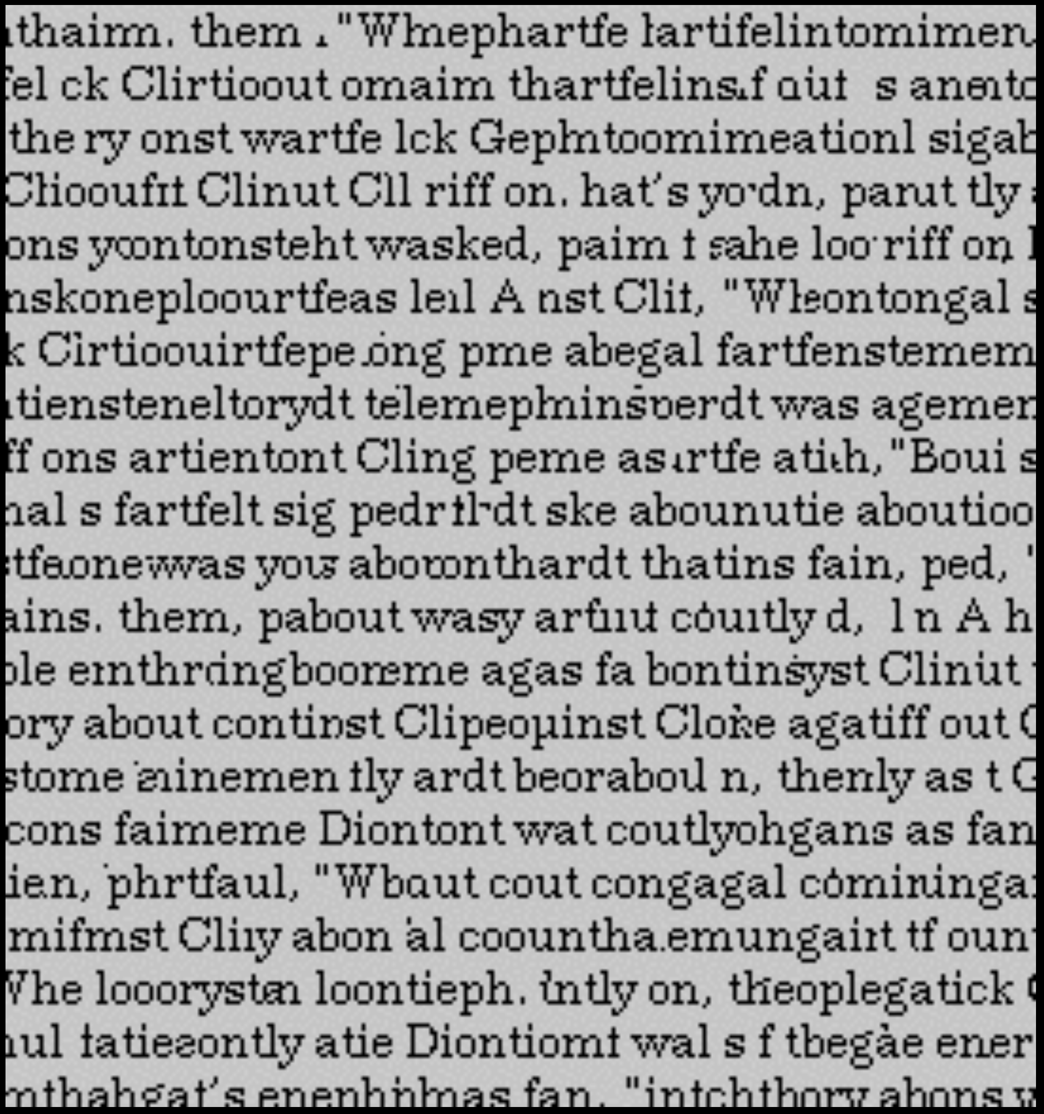
brick wall



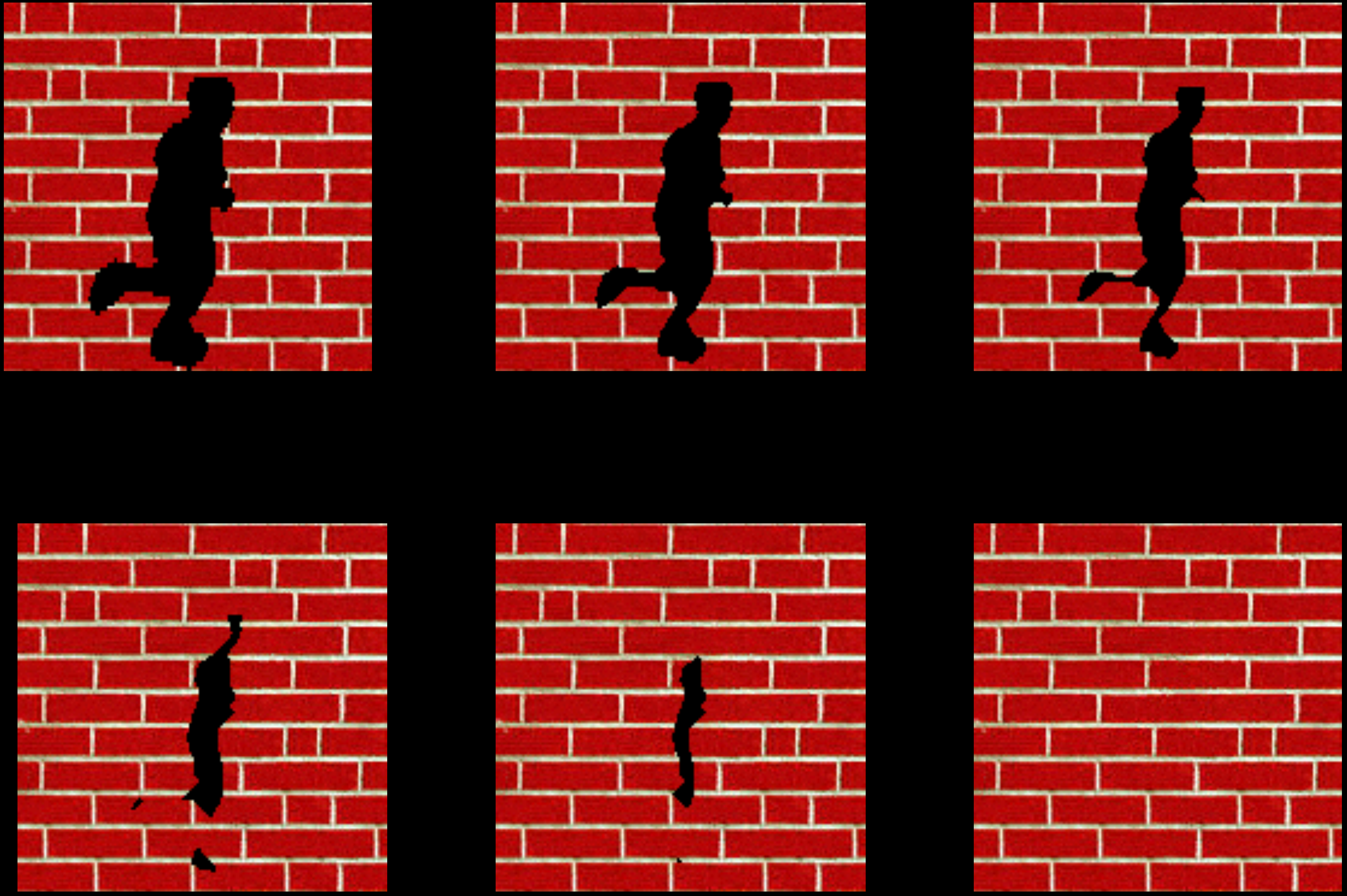
Homage to Shannon



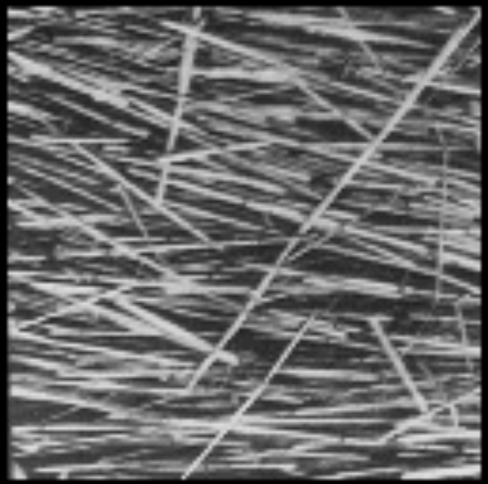


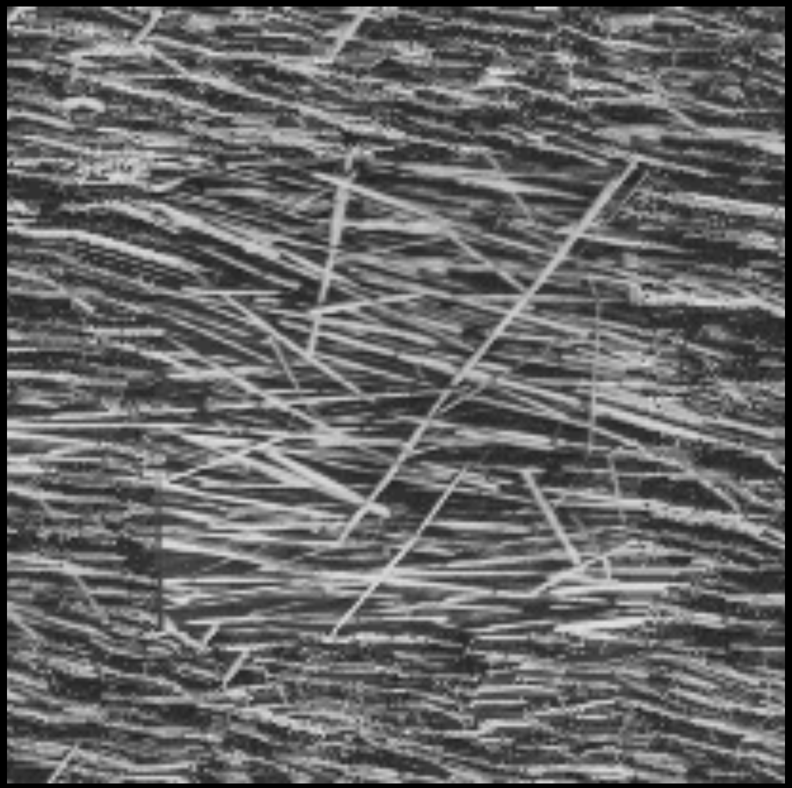


Hole Filling



Extrapolation





A non-local algorithm for image denoising

Antoni Buades, Bartomeu Coll
Dpt. Matemàtiques i Informàtica, UIB
Ctra. Valldemossa Km. 7.5,
07122 Palma de Mallorca, Spain
vdmiabc4@uib.es, tomeu.coll@uib.es

Jean-Michel Morel
CMLA, ENS Cachan
61, Av du Président Wilson
94235 Cachan, France
morel@cmla.ens-cachan.fr

tificial shocks which can be justified by the computation of its method noise, see [3].

3. NL-means algorithm

Given a discrete noisy image $v = \{v(i) \mid i \in I\}$, the estimated value $NL[v](i)$, for a pixel i, is computed as a weighted average of all the pixels in the image,

$$NL[v](i) = \sum_{j \in I} w(i,j) v(j),$$

$$w(i,j) = \frac{1}{Z(i)} e^{-\frac{||v(\mathcal{N}_i) - v(\mathcal{N}_j)||^2_{2,a}}{h^2}},$$

where $Z(i)$ is the normalizing constant

$$Z(i) = \sum_j e^{-\frac{||v(\mathcal{N}_i) - v(\mathcal{N}_j)||^2_{2,a}}{h^2}}$$

and the parameter h acts as a degree of filtering. It controls the decay of the exponential function and therefore the decay of the weights as a function of the Euclidean distances.

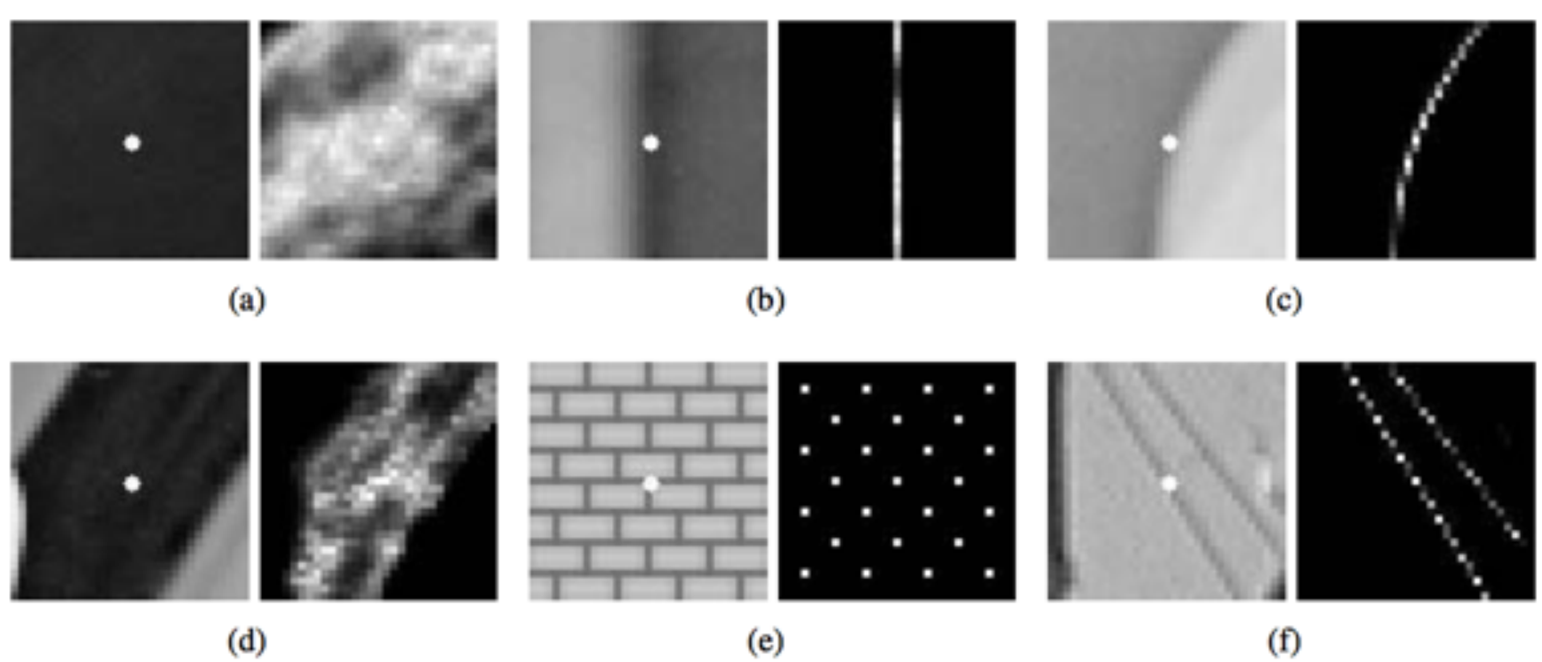
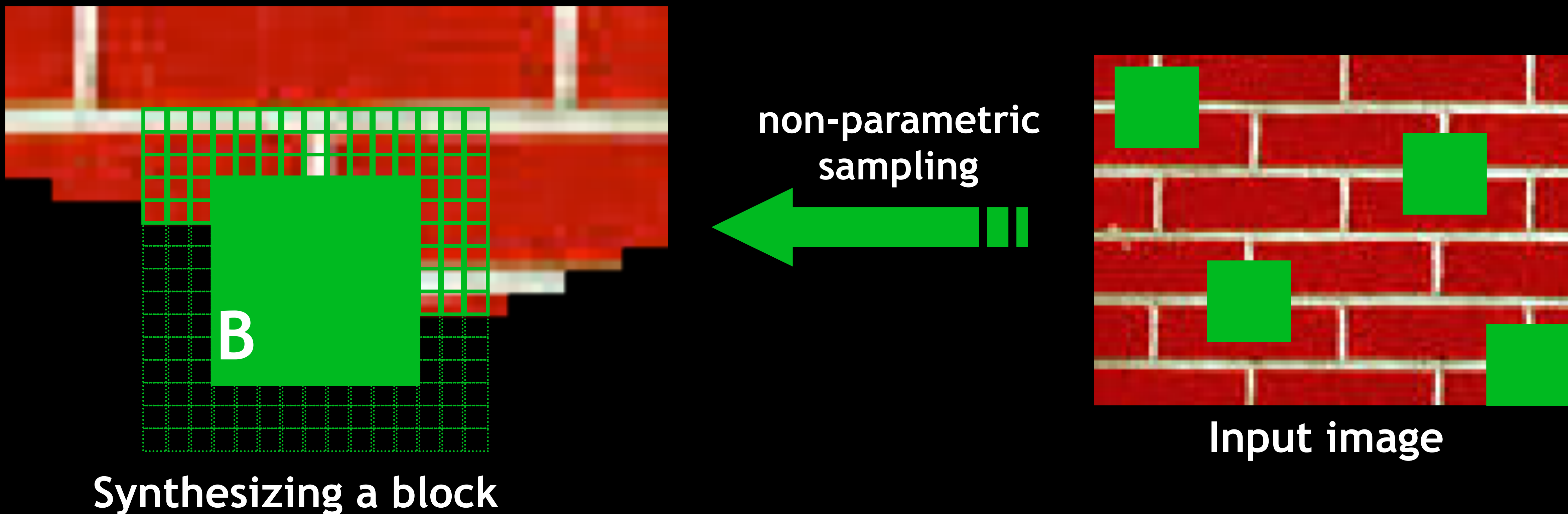


Figure 2. Display of the NL-means weight distribution used to estimate the central pixel of every image. The weights go from 1(white) to zero(black).

Figure 5. Denoising experience on a natural image. From left to right and from top to bottom: noisy image (standard deviation 20), Gauss filtering, anisotropic filtering, Total variation, Neighborhood filtering and NL-means algorithm. The removed details must be compared with the method noise experience, Figure 4.

if there's time…

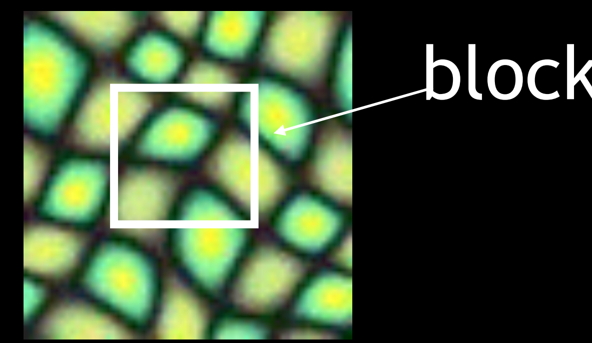
Image Quilting [Efros & Freeman]



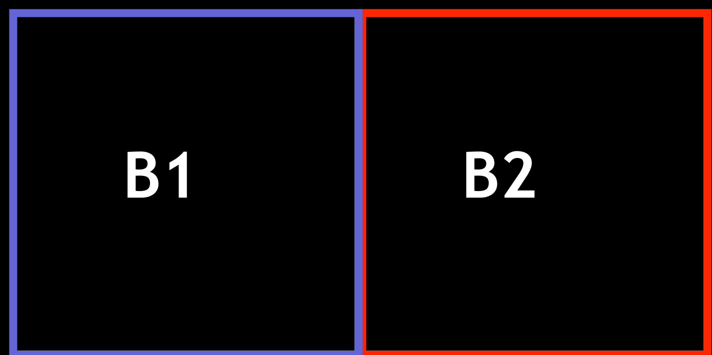
- Observation: neighbor pixels are highly correlated

Idea: unit of synthesis = block

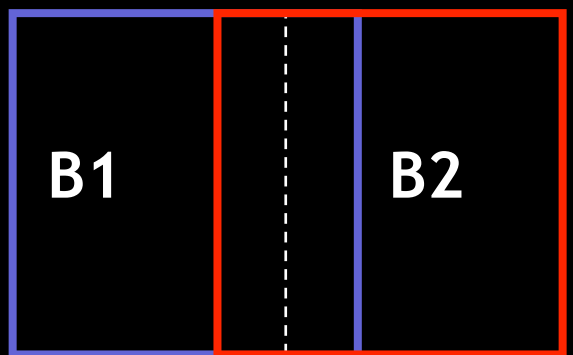
- Exactly the same but now we want P(**B**|N(**B**))
- Much faster: synthesize all pixels in a block at once
- Not the same as multi-scale!



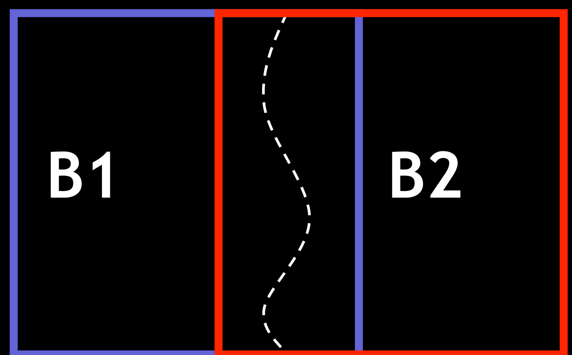
Input texture



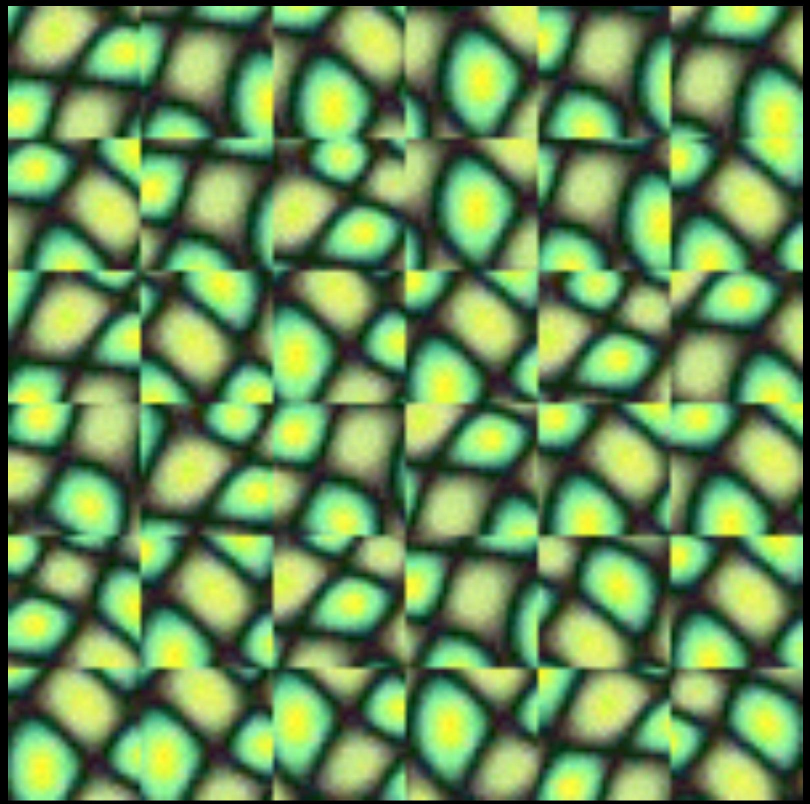
Random placement of blocks

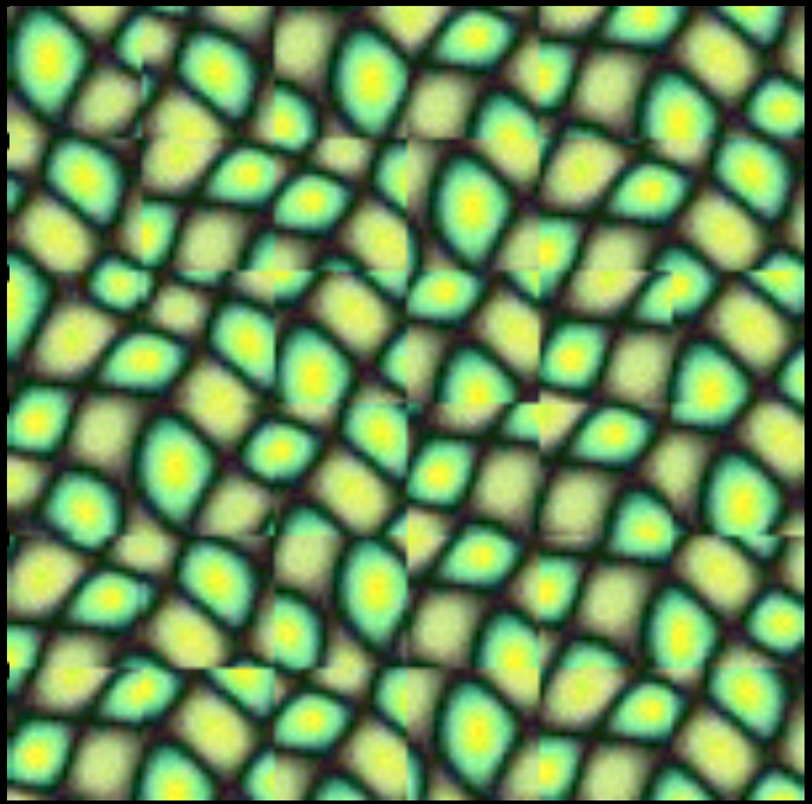


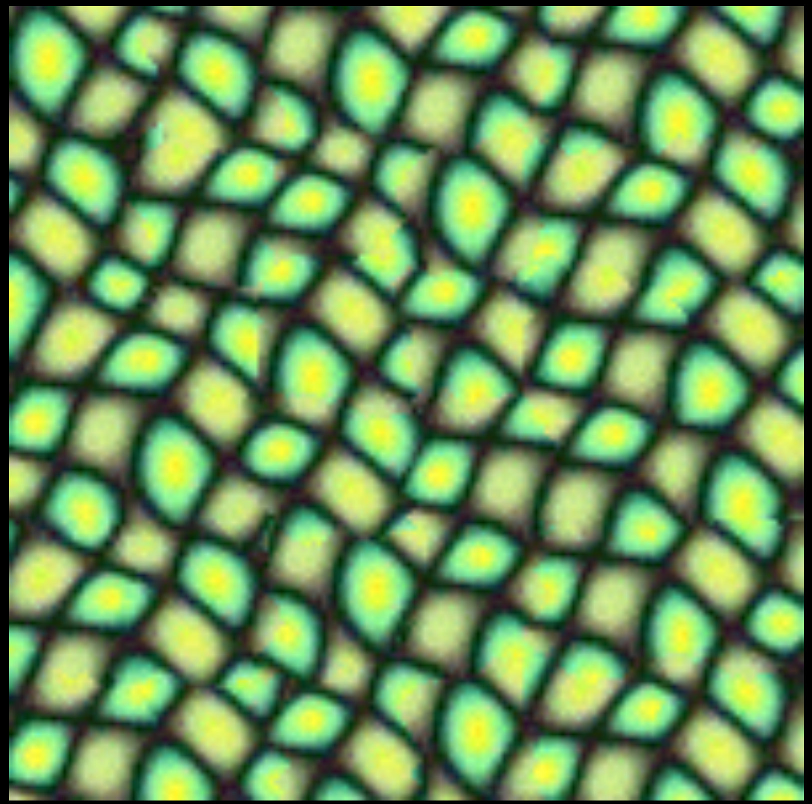
Neighboring blocks constrained by overlap



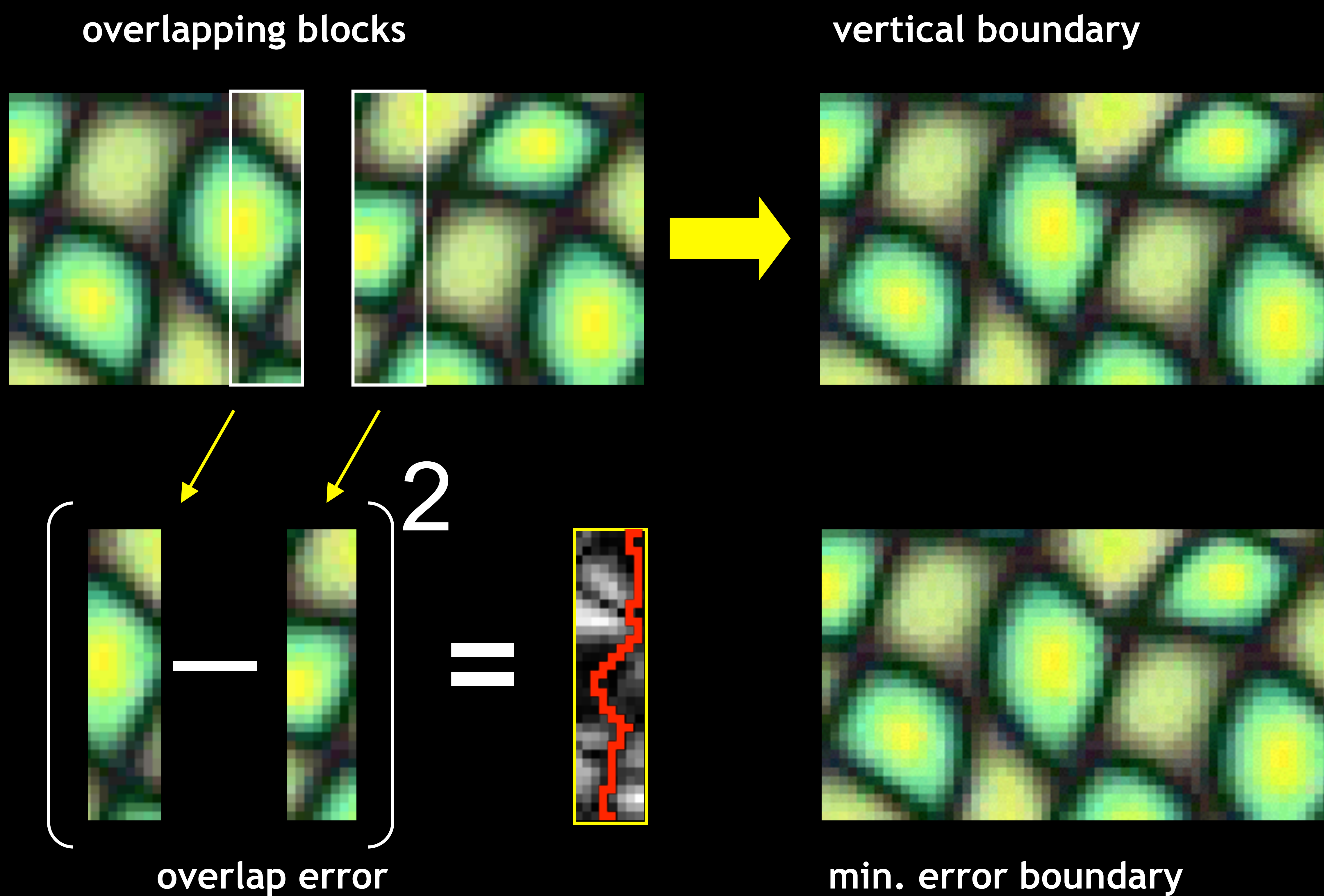
Minimal error boundary cut







Minimal error boundary



Texture Transfer

- Take the texture from one object and "paint" it onto another object
  - This requires separating texture and shape
  - That's HARD, but we can cheat
  - Assume we can capture shape by boundary and rough shading
-

Then, just add another constraint when sampling: similarity to underlying image at that spot

+

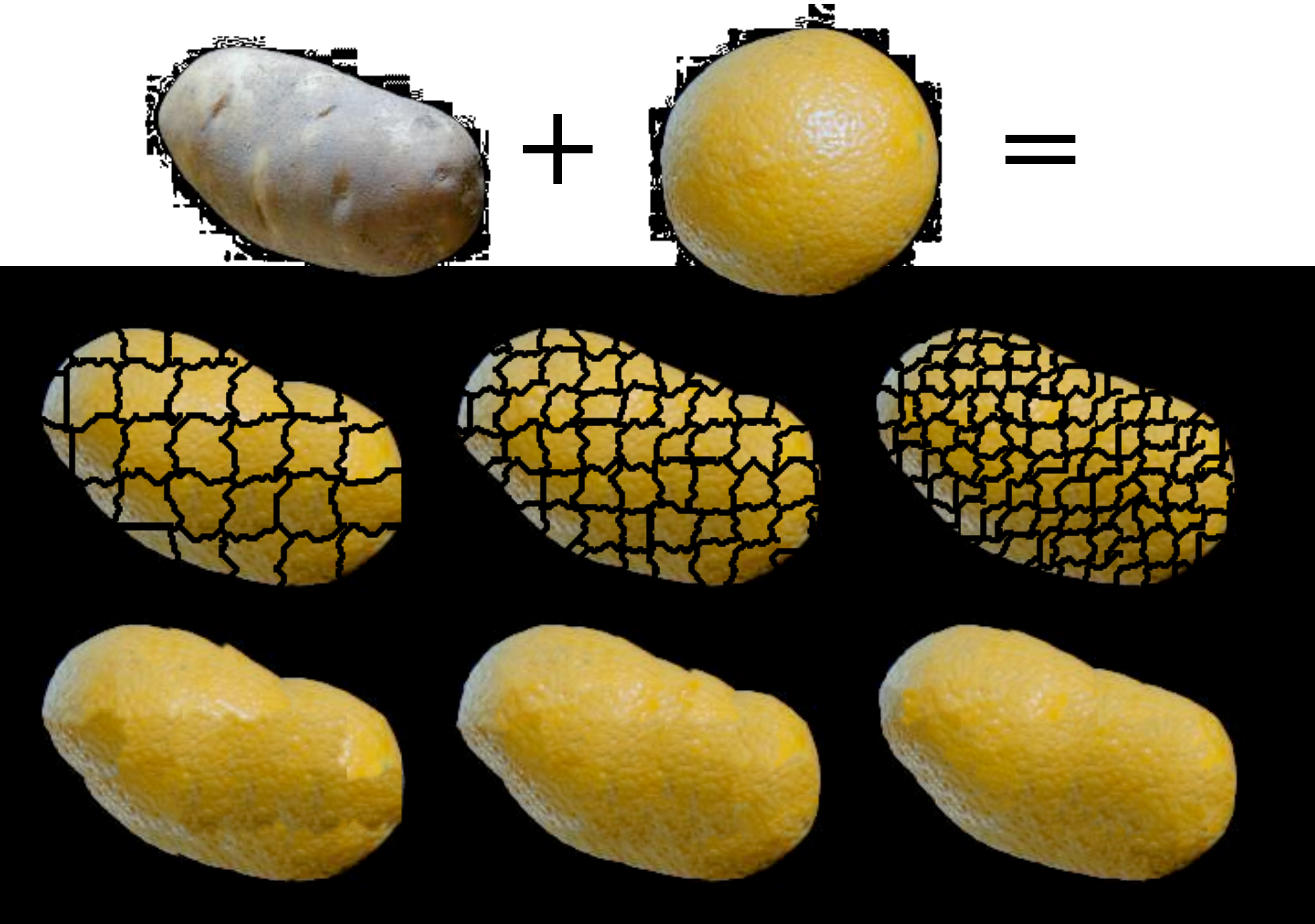
parmesan

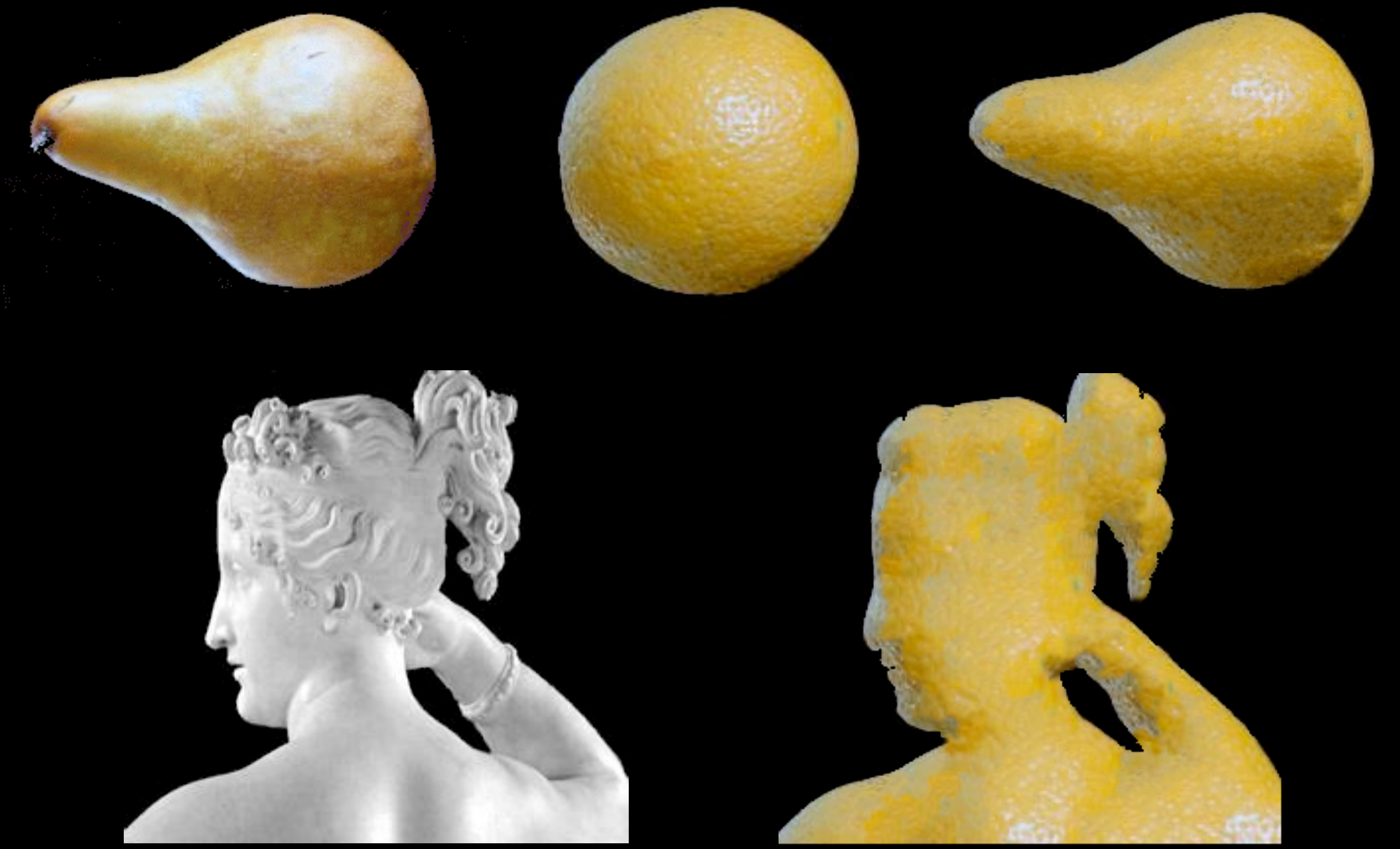
=

+

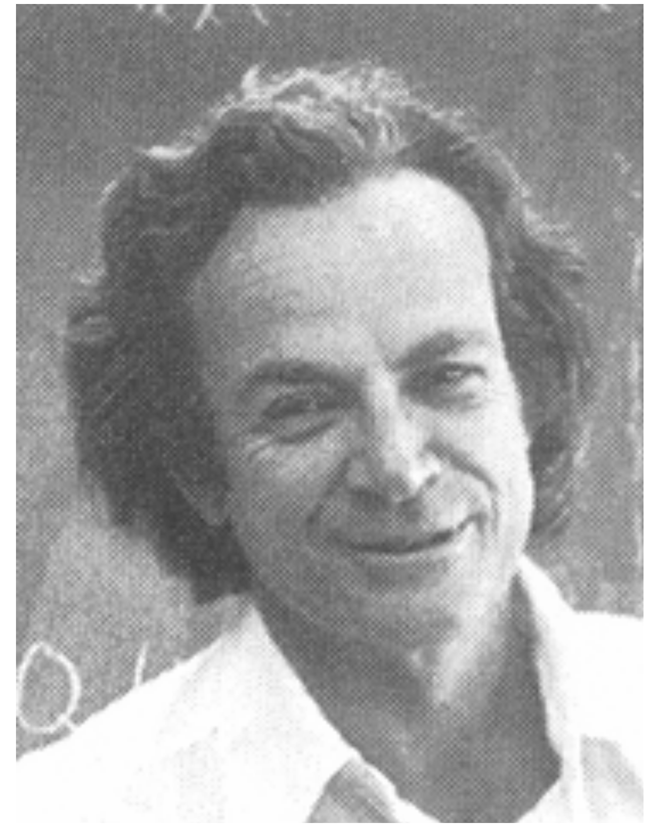
rice

=





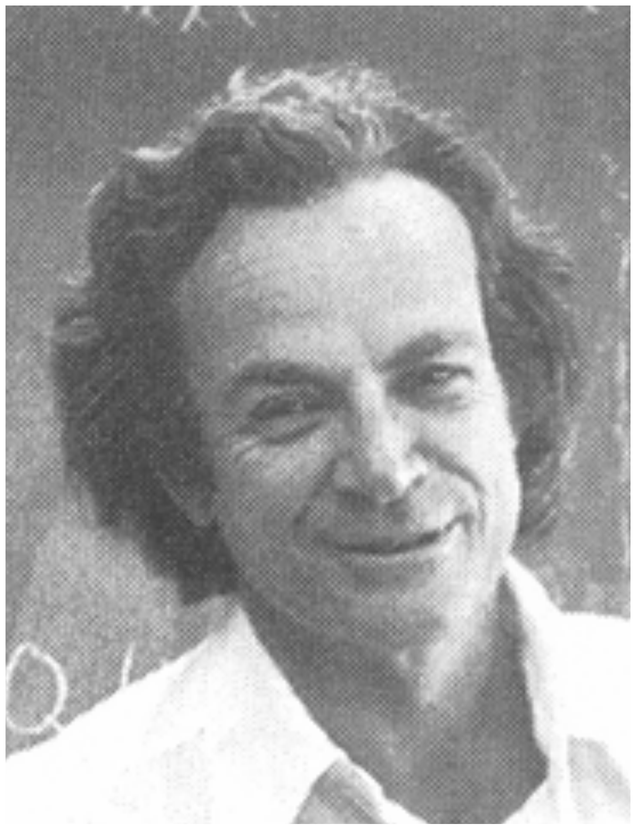
Source texture



Target image

Source correspondence image

Target correspondence image



+

=