

计算机视觉

Computer Vision

Lecture 4: Linear Filters

张 超

信息科学技术学院 智能科学系

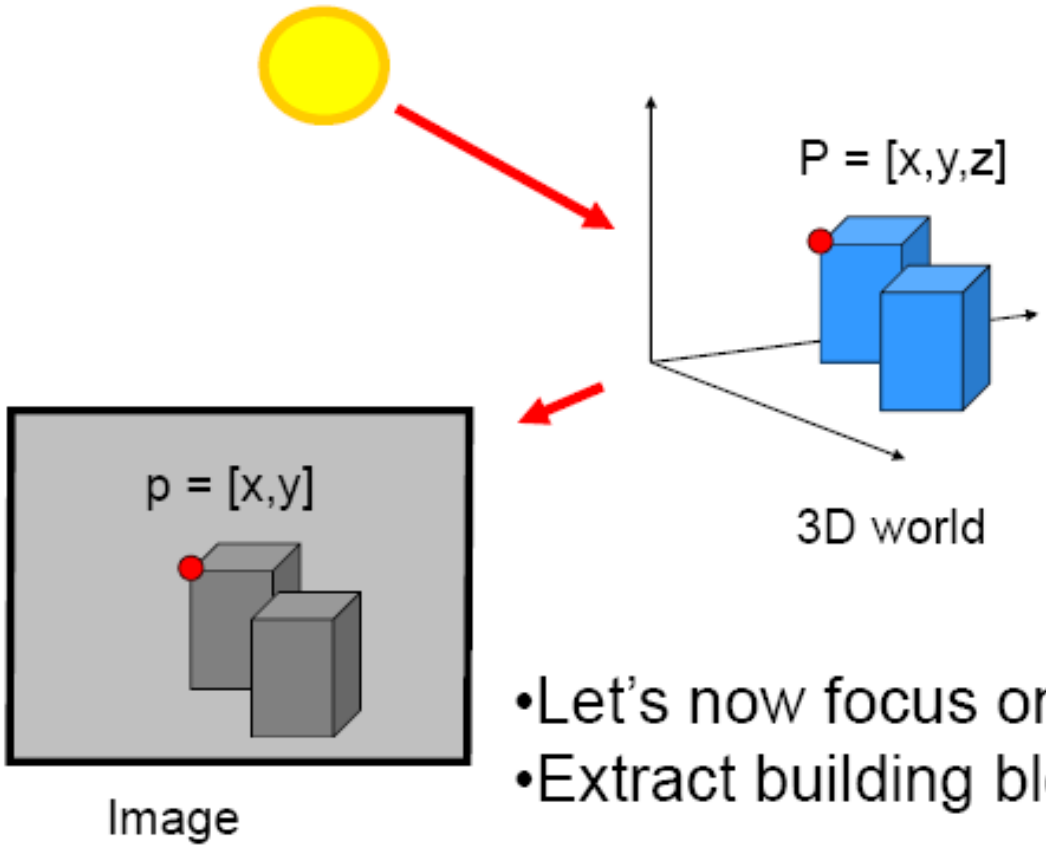
Linear Filters

- Linear filters and convolution
- Fourier Transforms
- Multi-resolution representation

Three views of filtering

- Image filters in spatial domain
  - Filter is a mathematical operation of a grid of numbers
  - Smoothing, sharpening, measuring texture
- Image filters in the frequency domain
  - Filtering is a way to modify the frequencies of images
  - Denoising, sampling, image compression
- Templates and Image Pyramids
  - Filtering is a way to match a template to the image
  - Detection, coarse-to-fine registration

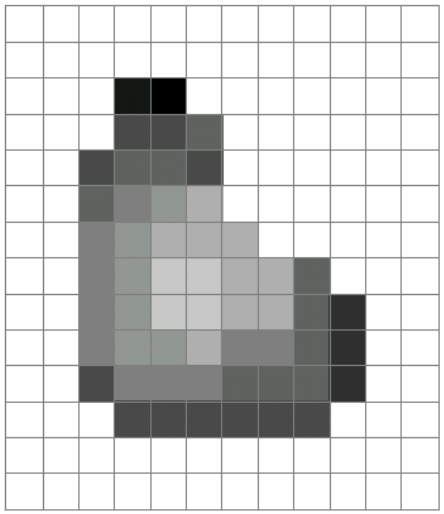
From 3D to 2D



- Let's now focus on 2D
- Extract building blocks

What is an image?

- **A grid of intensity values**

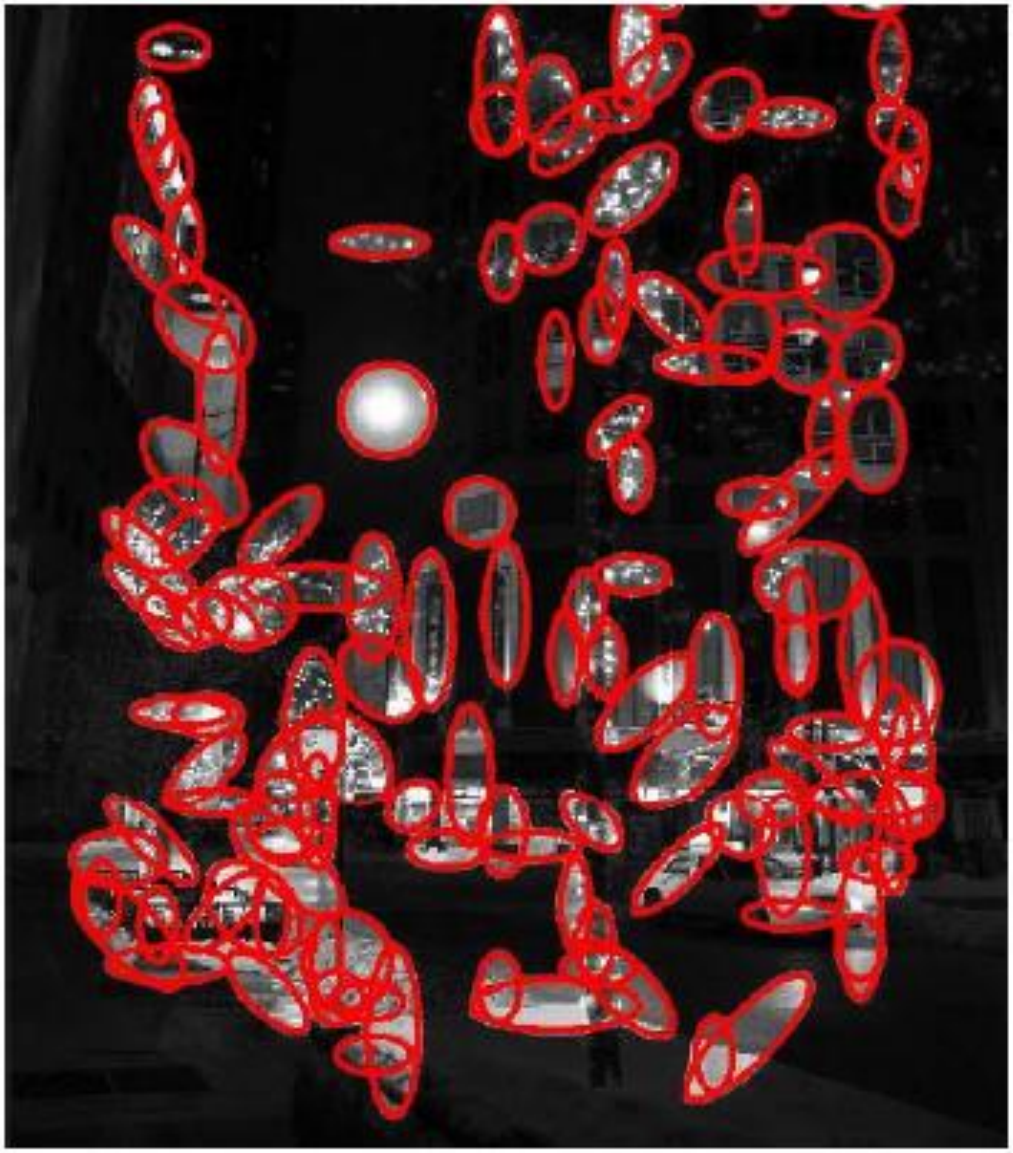


=

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

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

Extract useful building blocks



The big picture…

Feature Detection

e.g. DoG

Feature Description

e.g. SIFT

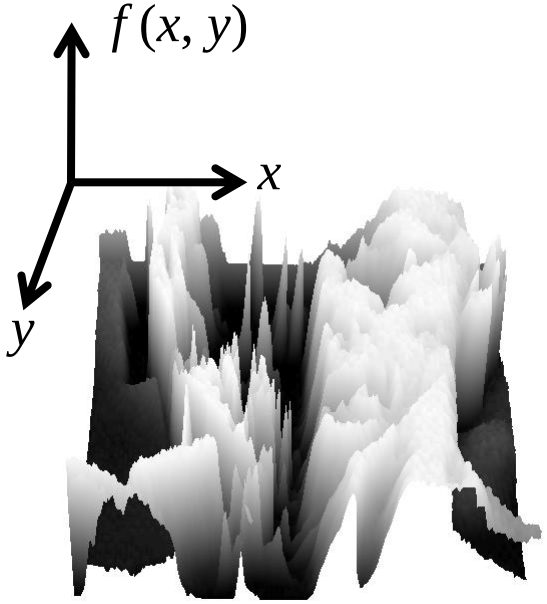
database of local descriptors

Matching / Indexing / Detection

What is an image?

- We can think of a (grayscale) image as a function, f, from $\mathbf{R}^2$ to $\mathbf{R}$:
  - $f(x,y)$ gives the intensity at position (x,y)

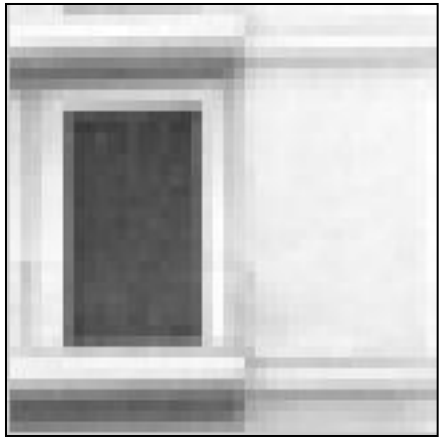
snoop

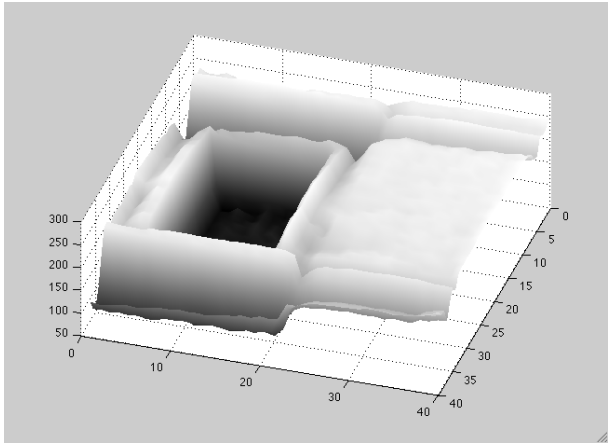


3D view

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

The image as a “surface”





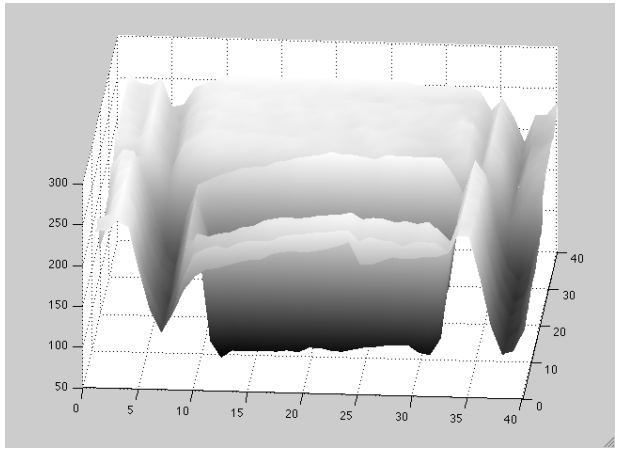
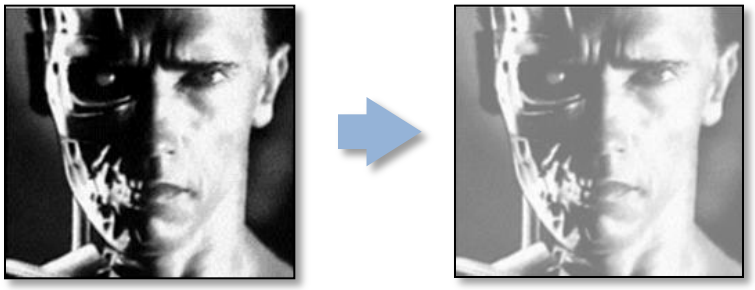
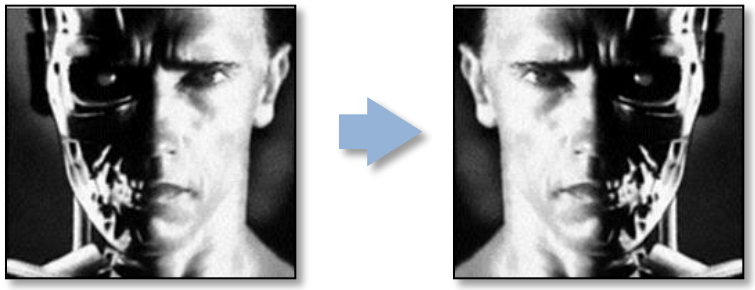


Image transformations

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



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



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

- **We'll talk about a special kind of operator, *convolution* (linear filtering)**

Question: Noise reduction

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

- Take lots of images and average them!
- What's the next best thing?

Source: S. Seitz

Linear filters and convolution

Convolution

- Each pixel in output image is
  - weighted average of window of pixels in input image
  - weights stay the same
  - window centered on pixel
- Important operation
  - Example: smoothing by averaging
  - Example: smoothing by weighted average
  - Example: taking a derivative

Convolution

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

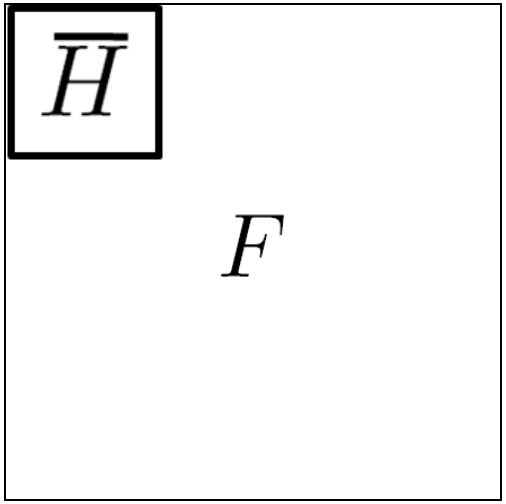
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

This is called a **convolution** operation:

$$G = H * F$$

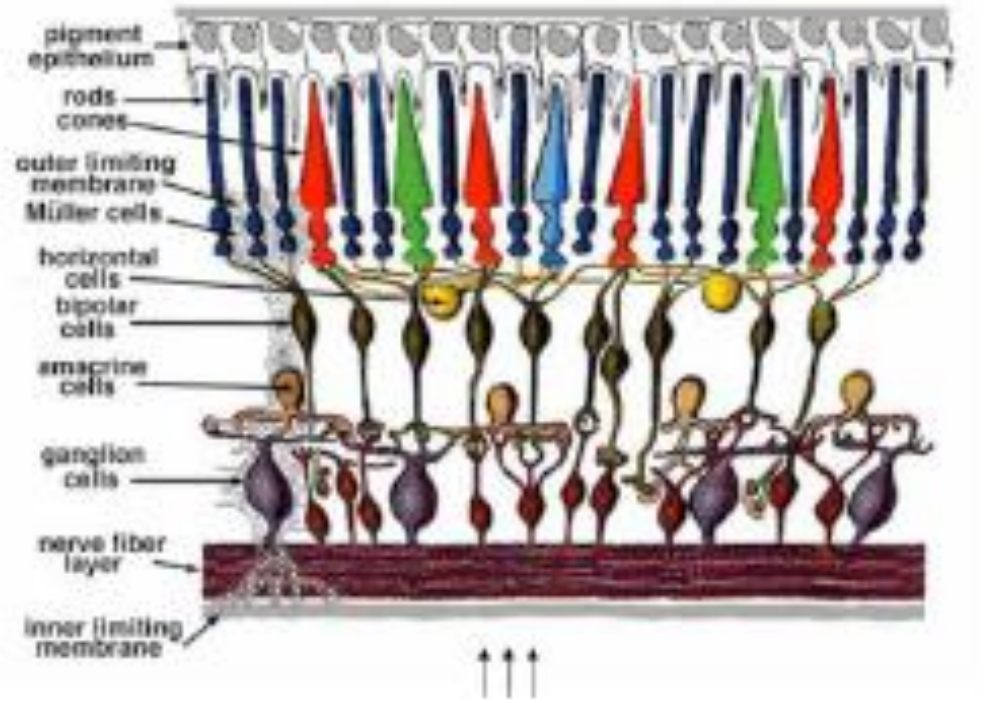
- MATLAB functions: conv2, filter2, imfilter

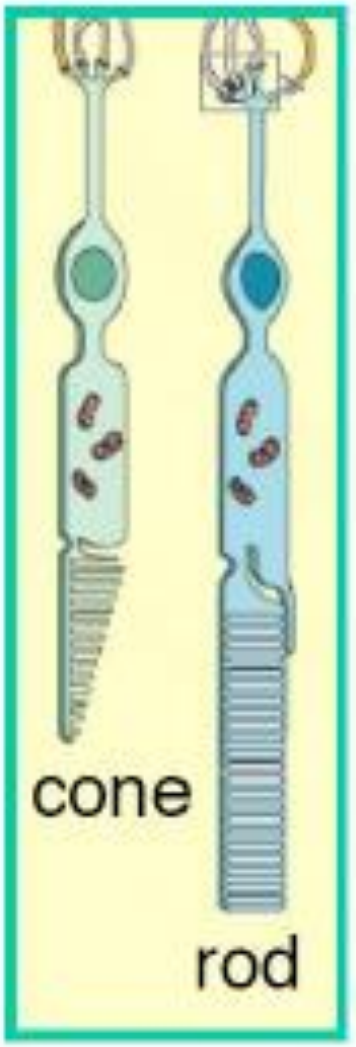
Convolution



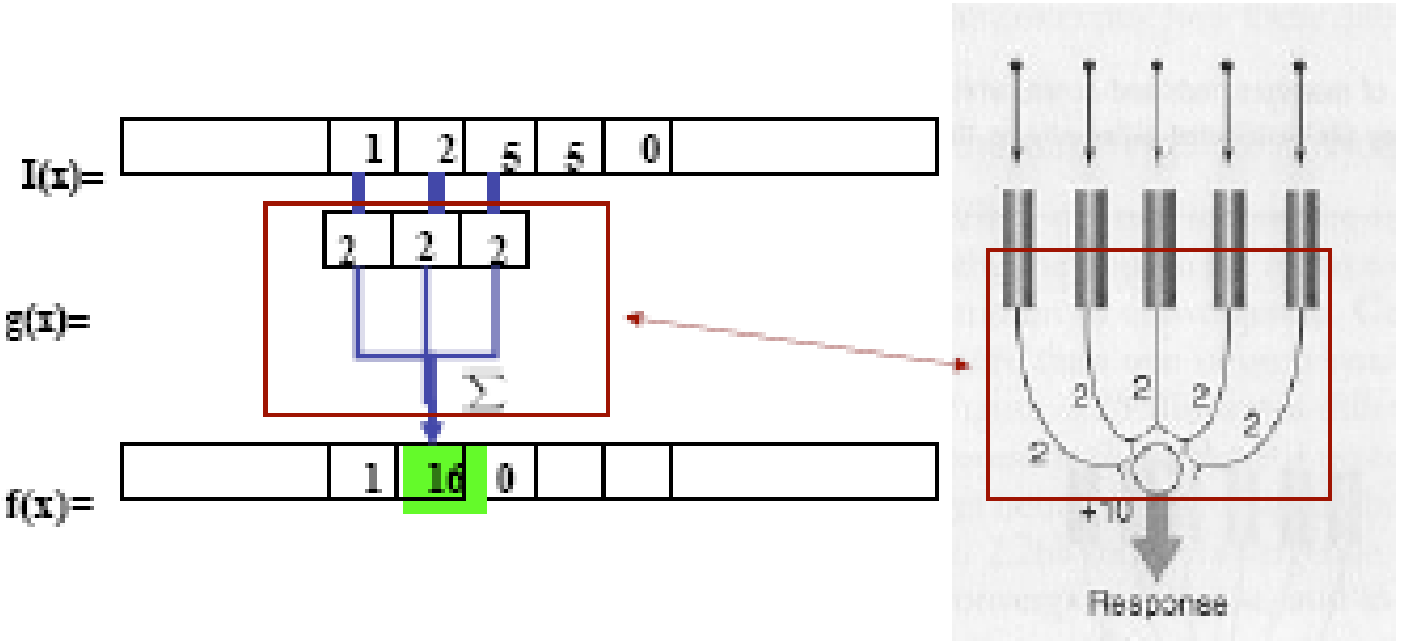
Adapted from F. Durand

Retina up-close

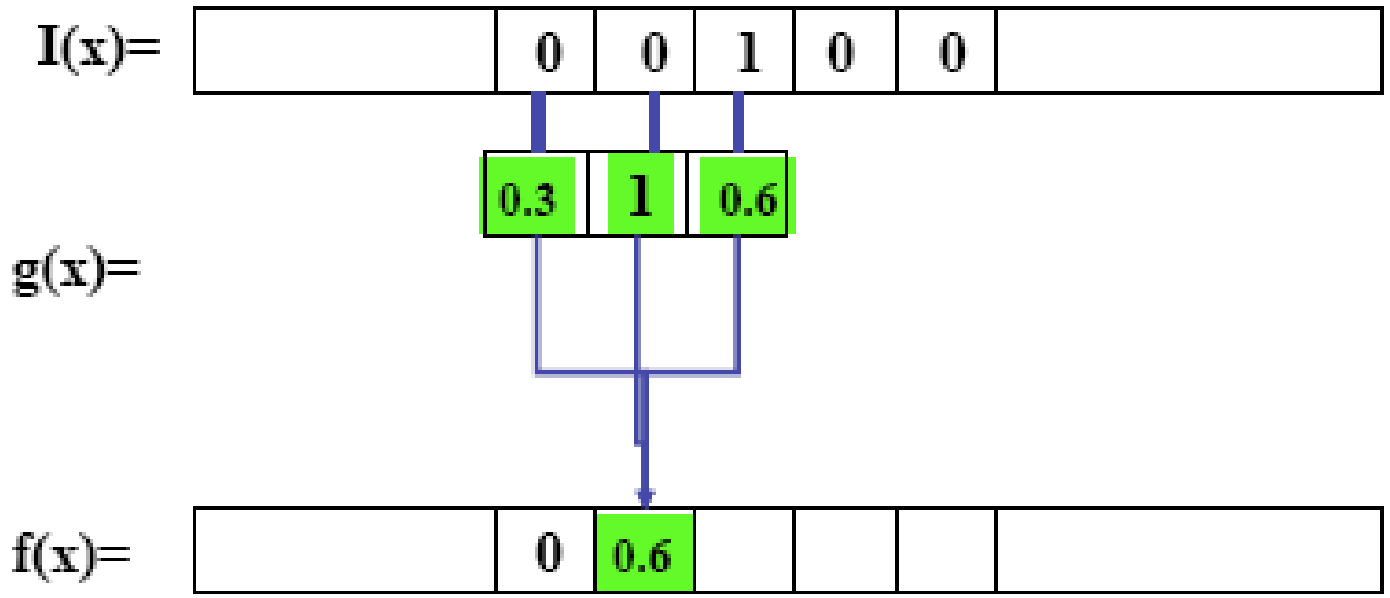


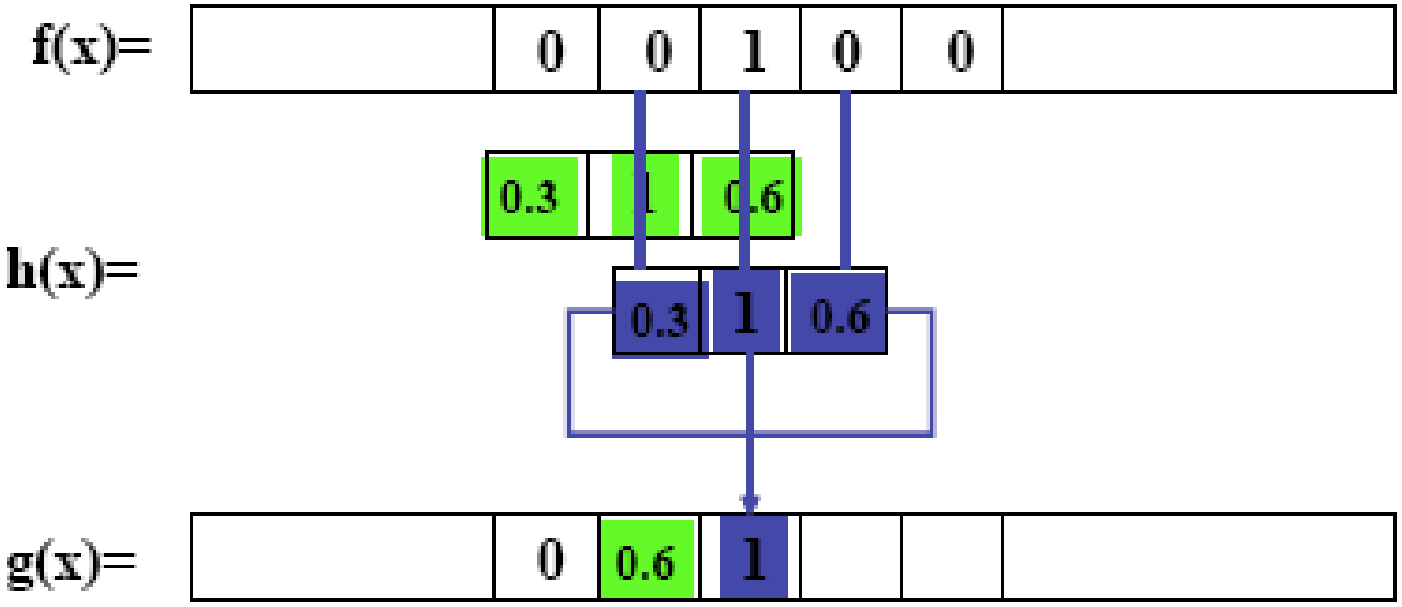


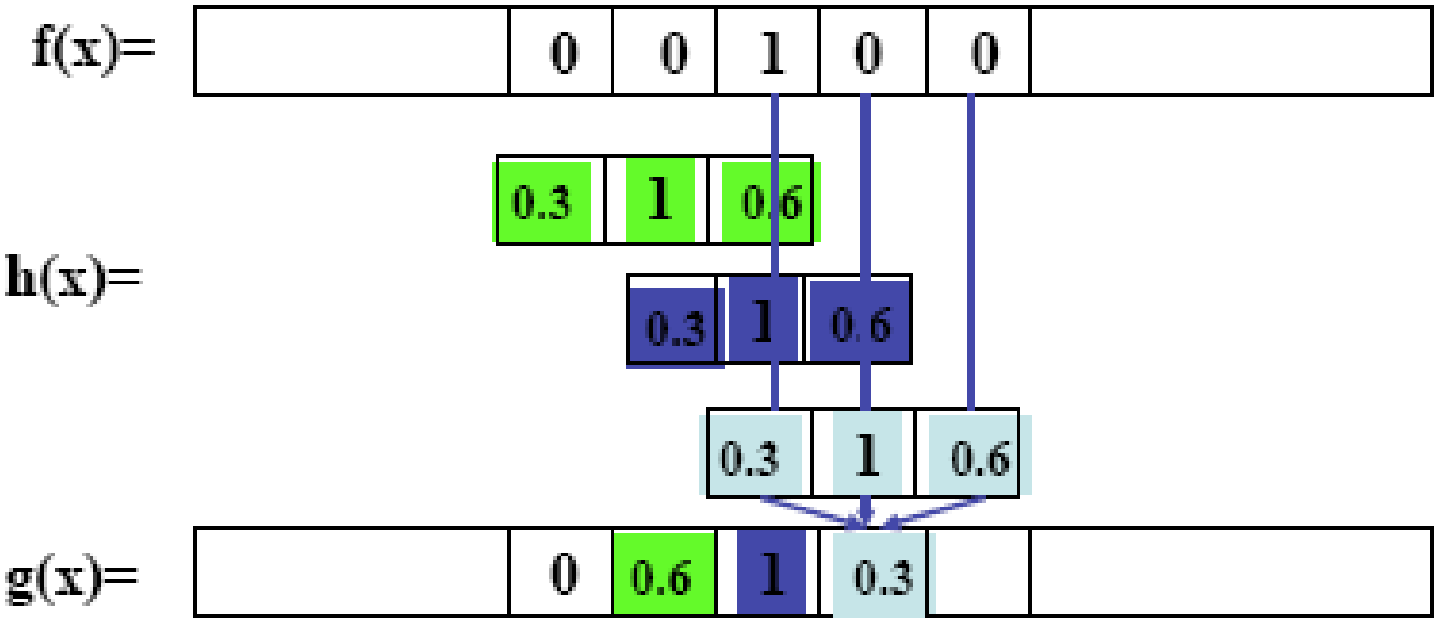
Simple Neural Network



This leads to parallel computation


I(x)=
0
0
1
0
0
0.3
1
0.6
g(x)=
f(x)=
0
0.6


f(x)=
0
0
1
0
0
0.3
1
0.6
h(x)=
0.3
1
0.6
g(x)=
0
0.6
1


f(x)=
0
0
1
0
0
0.3
1
0.6
h(x)=
0.3
1
0.6
0.3
1
0.6
g(x)=
0
0.6
1
0.3

Filters

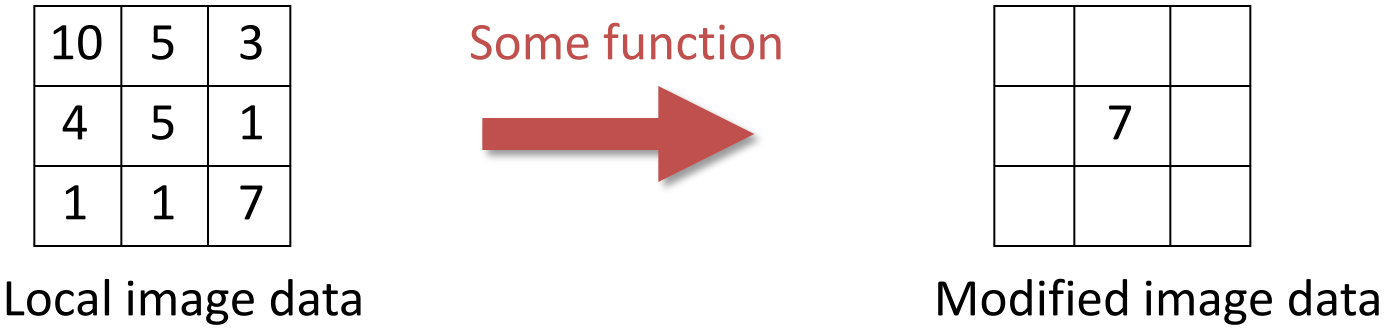
- **Convolution is the same as applying a filter to an image**
  - **Weights are called the "filter kernel"**
  - **Notice I can re-index the filter kernel so it looks like a vector**
  - **If I do so, each output pixel is**
    - **the dot-product of that vector with**
    - **the vector of pixels centered at the output pixel**
- **Important insight**
  - **Filters respond strongly to image windows that look like them**
    - **positive response: just like H**
    - **negative response: like H, but with contrast reversal**

$$R_{ij} = \sum_{u,v} H_{i-u,j-v} F_{u,v}$$

Filter kernel

Image filtering

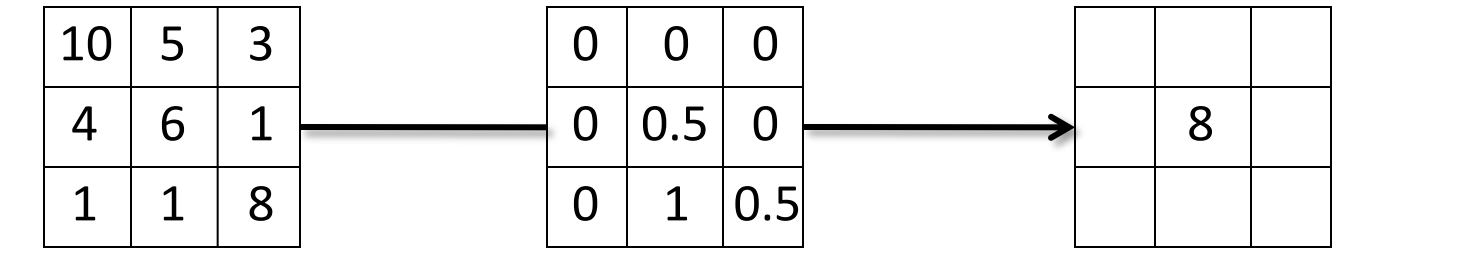
- **Modify the pixels in an image based on some function of a local neighborhood of each pixel**



Source: L. Zhang

Linear filtering

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

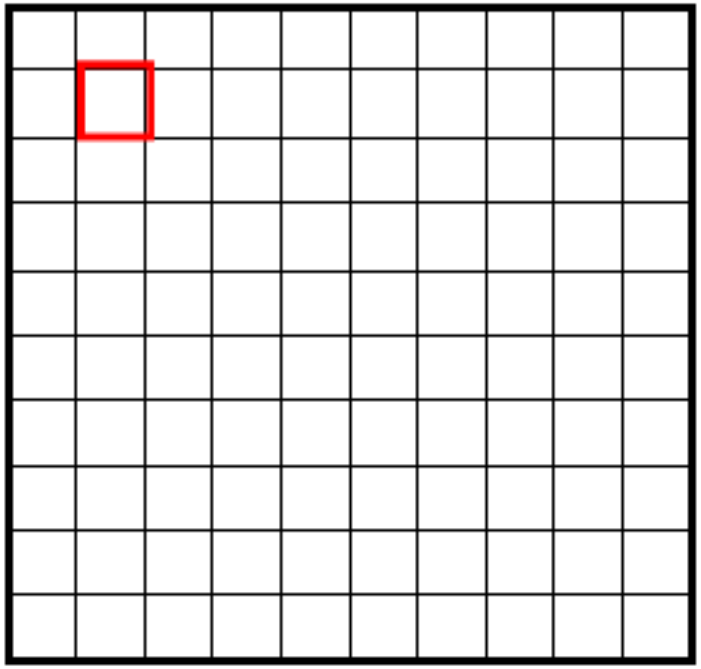


Source: L. Zhang

$\frac{1}{9}$

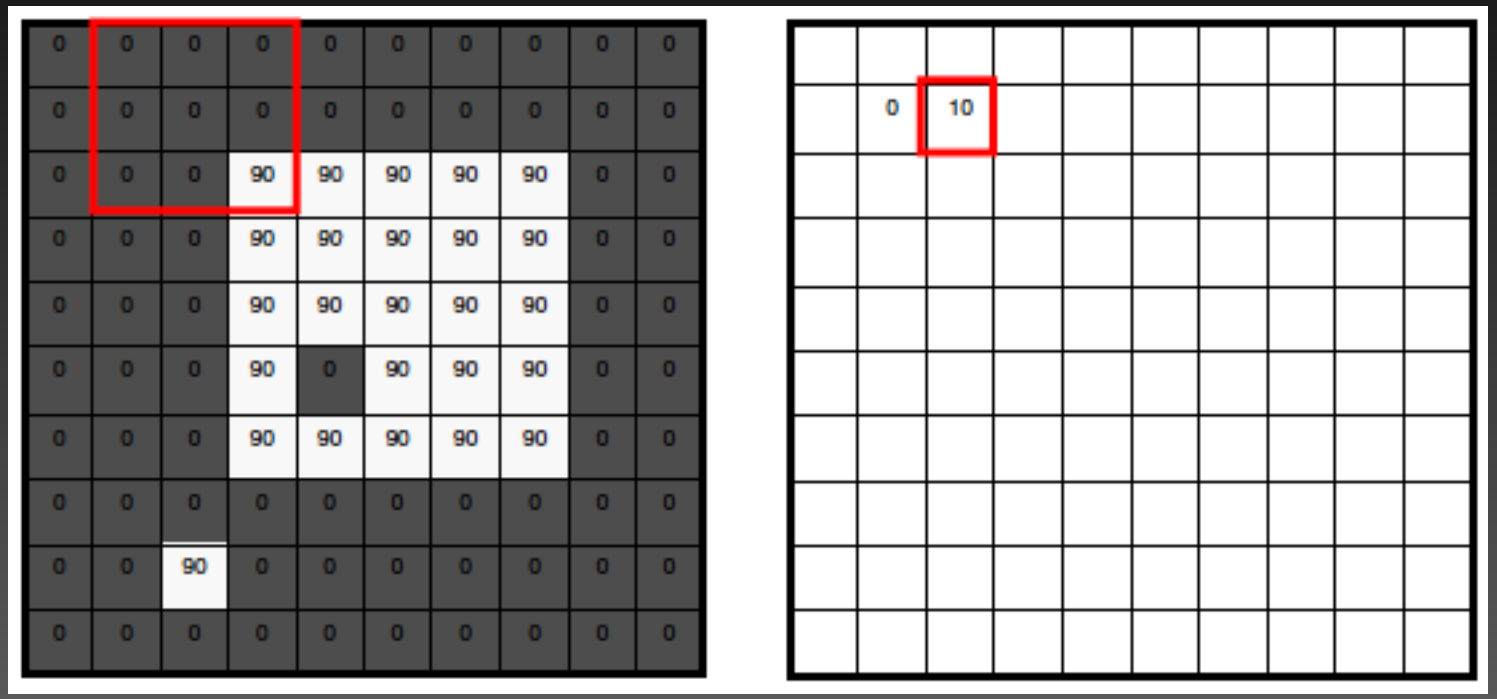
1	1	1
1	1	1
1	1	1

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



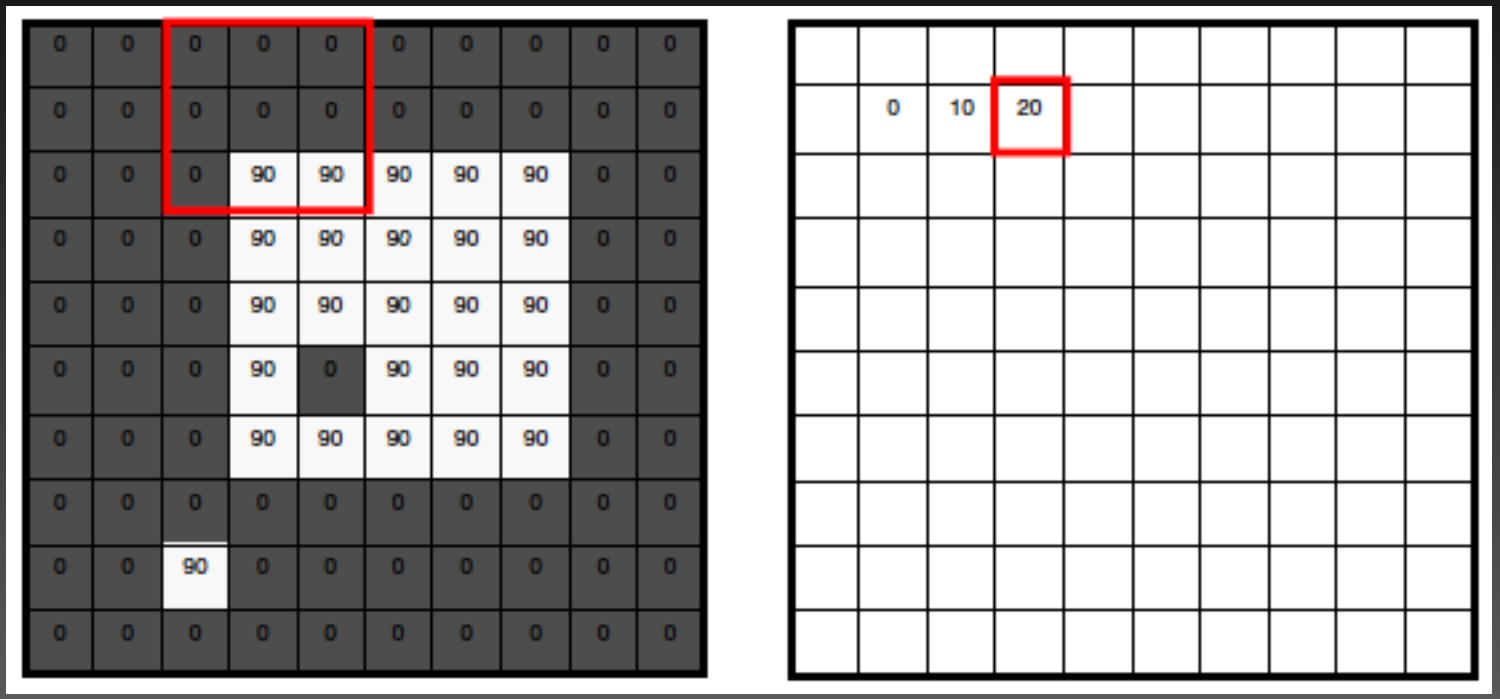
$\frac{1}{9}$

1	1	1
1	1	1
1	1	1



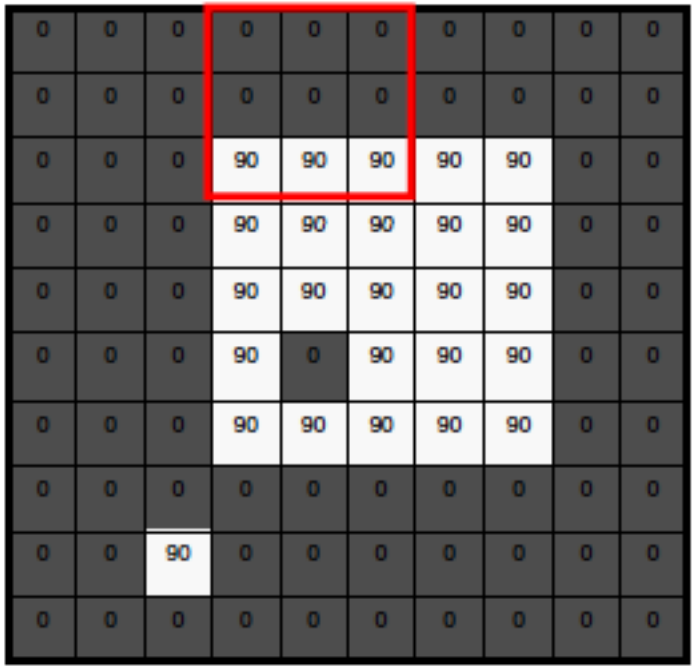
$\frac{1}{9}$

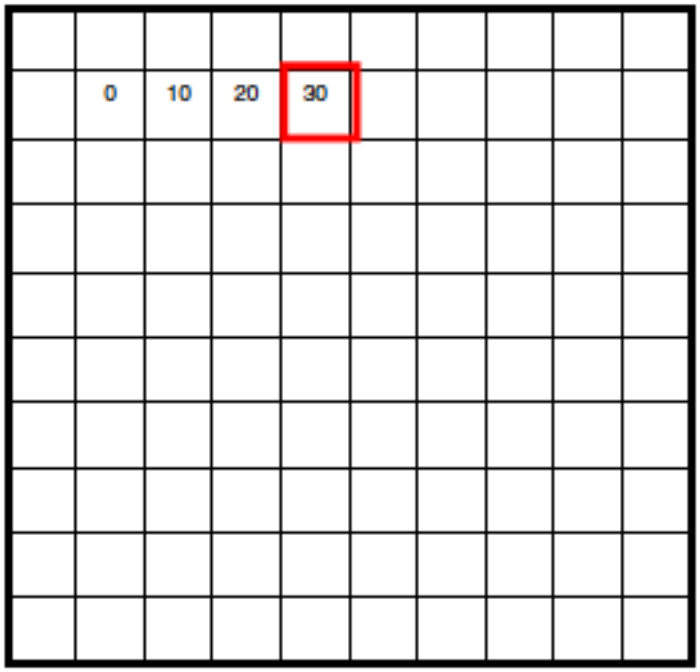
1	1	1
1	1	1
1	1	1



$\frac{1}{9}$

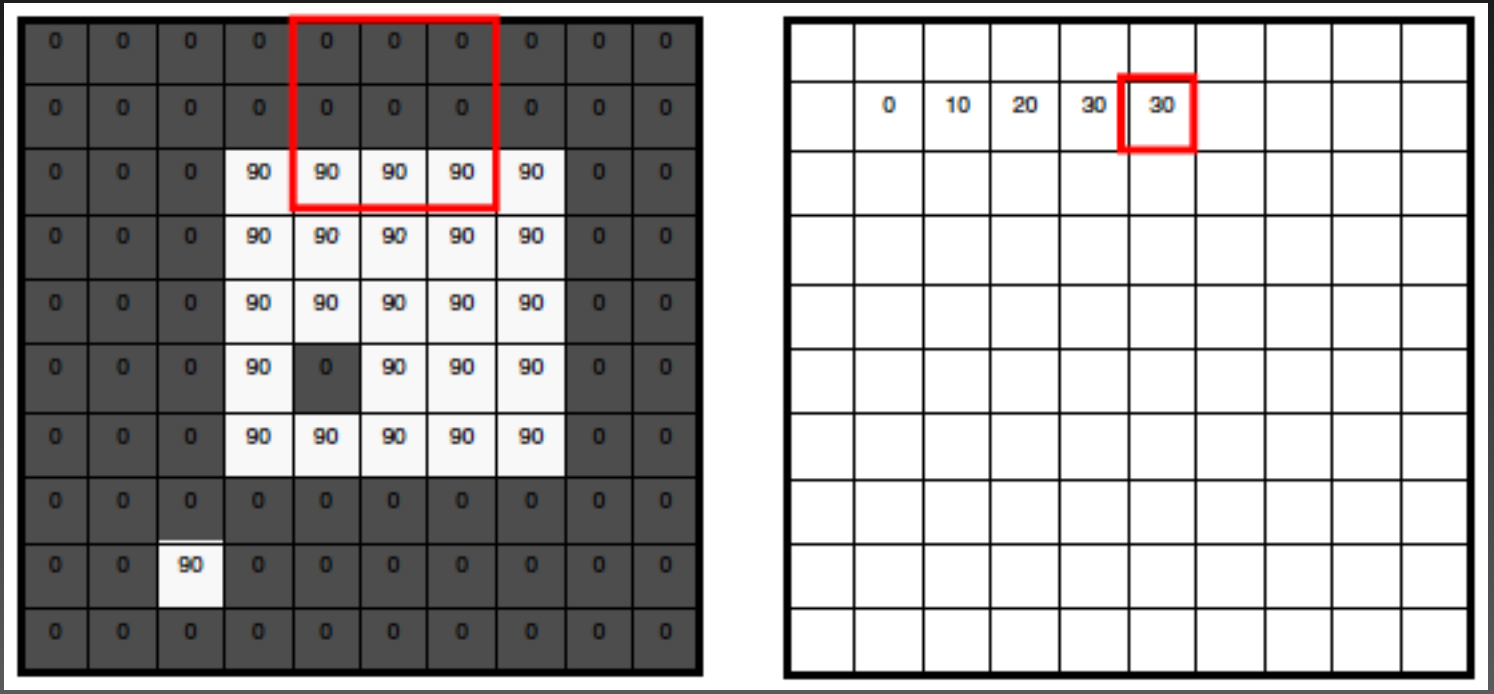
1	1	1
1	1	1
1	1	1





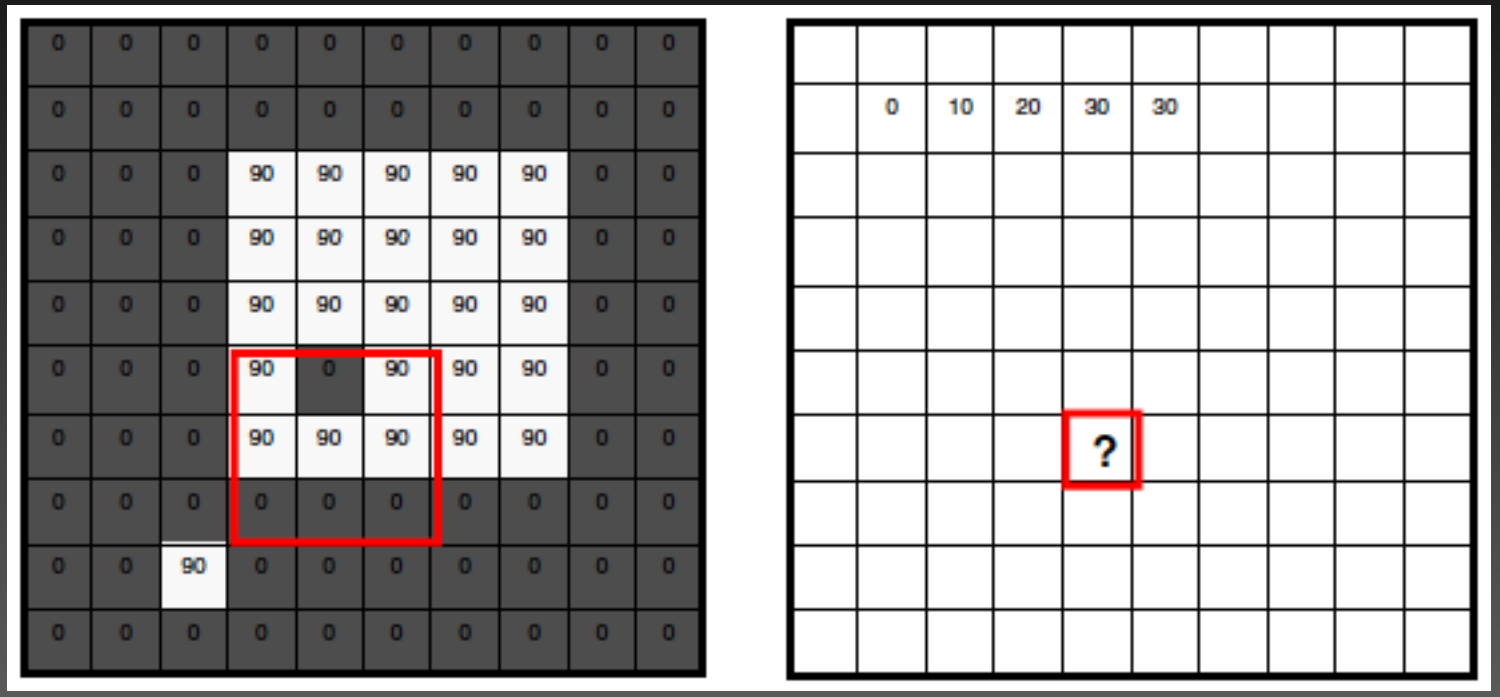
$\frac{1}{9}$

1	1	1
1	1	1
1	1	1



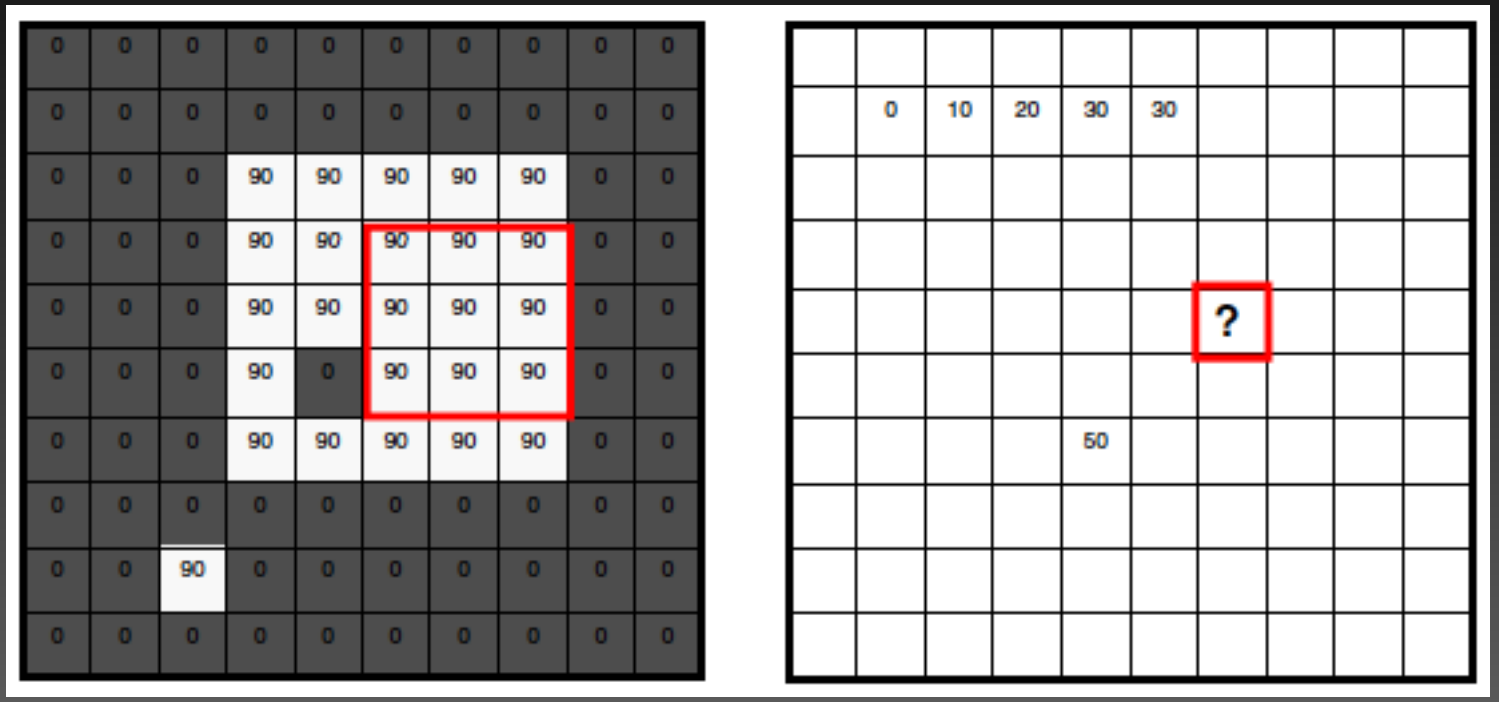
$\frac{1}{9}$

1	1	1
1	1	1
1	1	1



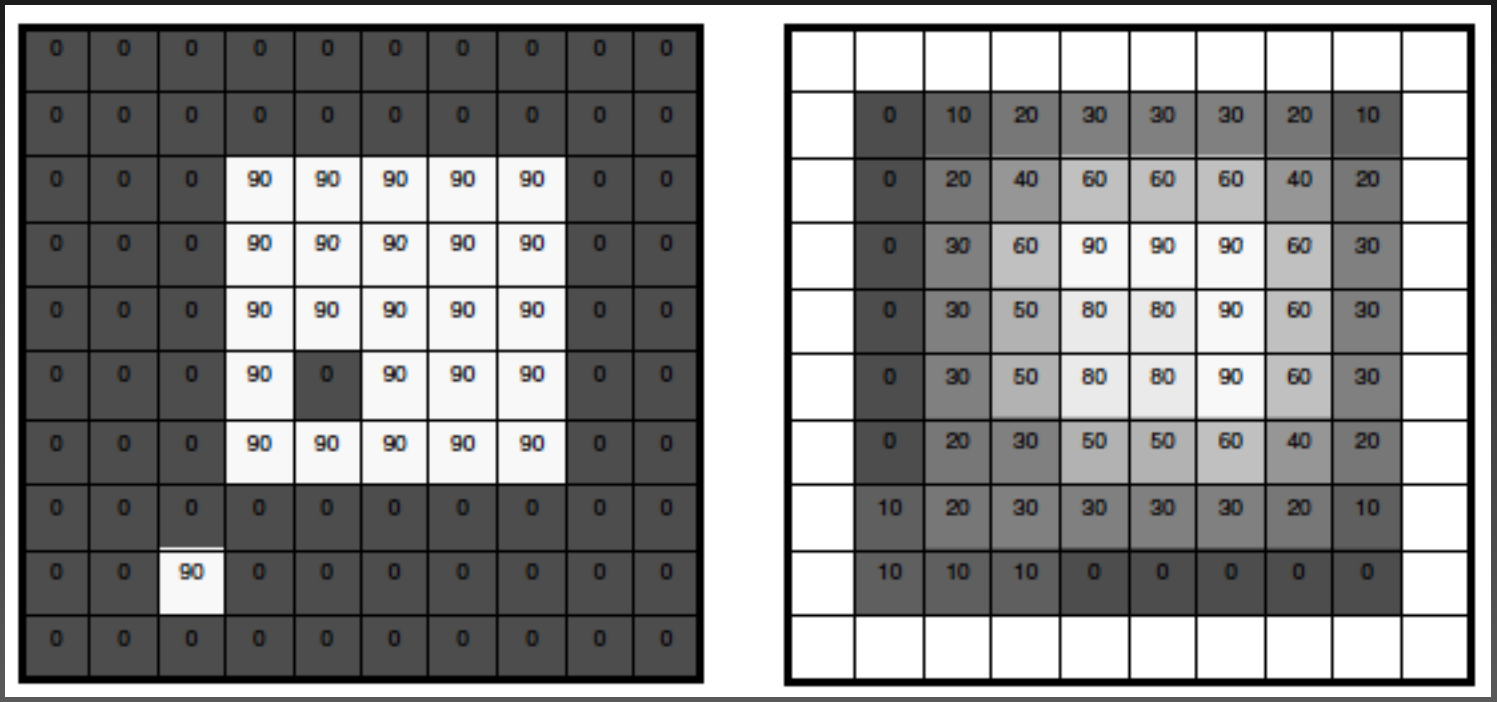
$\frac{1}{9}$

1	1	1
1	1	1
1	1	1



$\frac{1}{9}$

1	1	1
1	1	1
1	1	1



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	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	

What does it do?

- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)

Key properties

- **Linearity:** $\text{filter}(f_1 + f_2) = \text{filter}(f_1) + \text{filter}(f_2)$
- **Shift invariance: same behavior regardless of pixel location:** $\text{filter}(\text{shift}(f)) = \text{shift}(\text{filter}(f))$
- **Theoretical result: any linear shift-invariant operator can be represented as a convolution**

Properties in more detail

- **Commutative:** $a * b = b * a$
  - Conceptually no difference between filter and signal
- **Associative:** $a * (b * c) = (a * b) * c$
  - Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - This is equivalent to applying one filter: a $* (b_1 * b_2 * b_3)$
- **Distributes over addition:** $a * (b + c) = (a * b) + (a * c)$
- **Scalars factor out:** $ka * b = a * kb = k (a * b)$
- **Identity: unit impulse** $e = [\ldots, 0, 0, 1, 0, 0, \ldots]$, $a * e = a$

Linear filters: examples

Original

$*$

0	0	0
0	1	0
0	0	0

$=$

Identical image

Source: D. Lowe

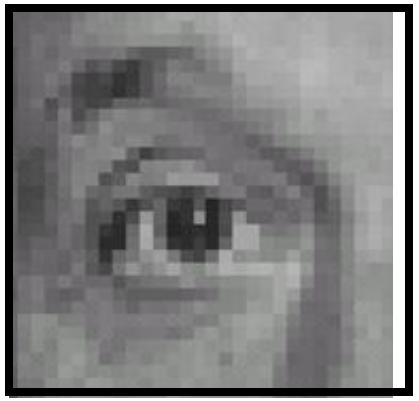
Linear filters: examples

Original

*

0	0	0
1	0	0
0	0	0

=



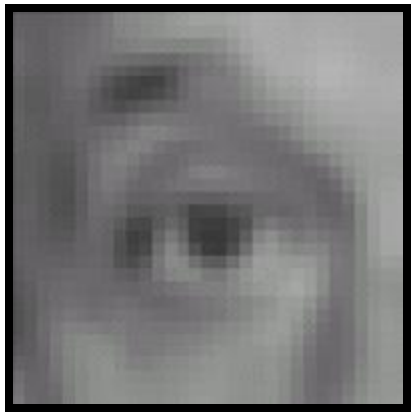
Shifted left
By 1 pixel

Source: D. Lowe

Linear filters: examples

Original

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



Blur (with a mean filter)

Source: D. Lowe

Linear filters: examples

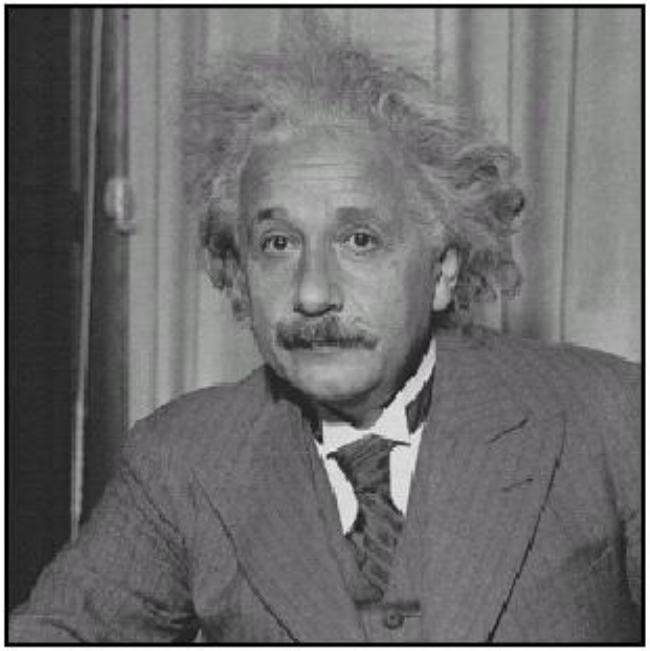
Original

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

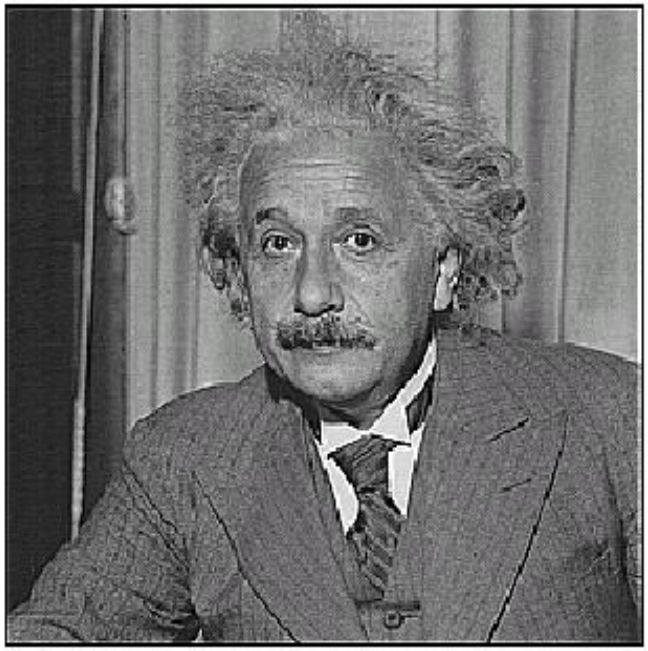
Sharpening filter
(accentuates edges)

Source: D. Lowe

Sharpening



before



after

Source: D. Lowe

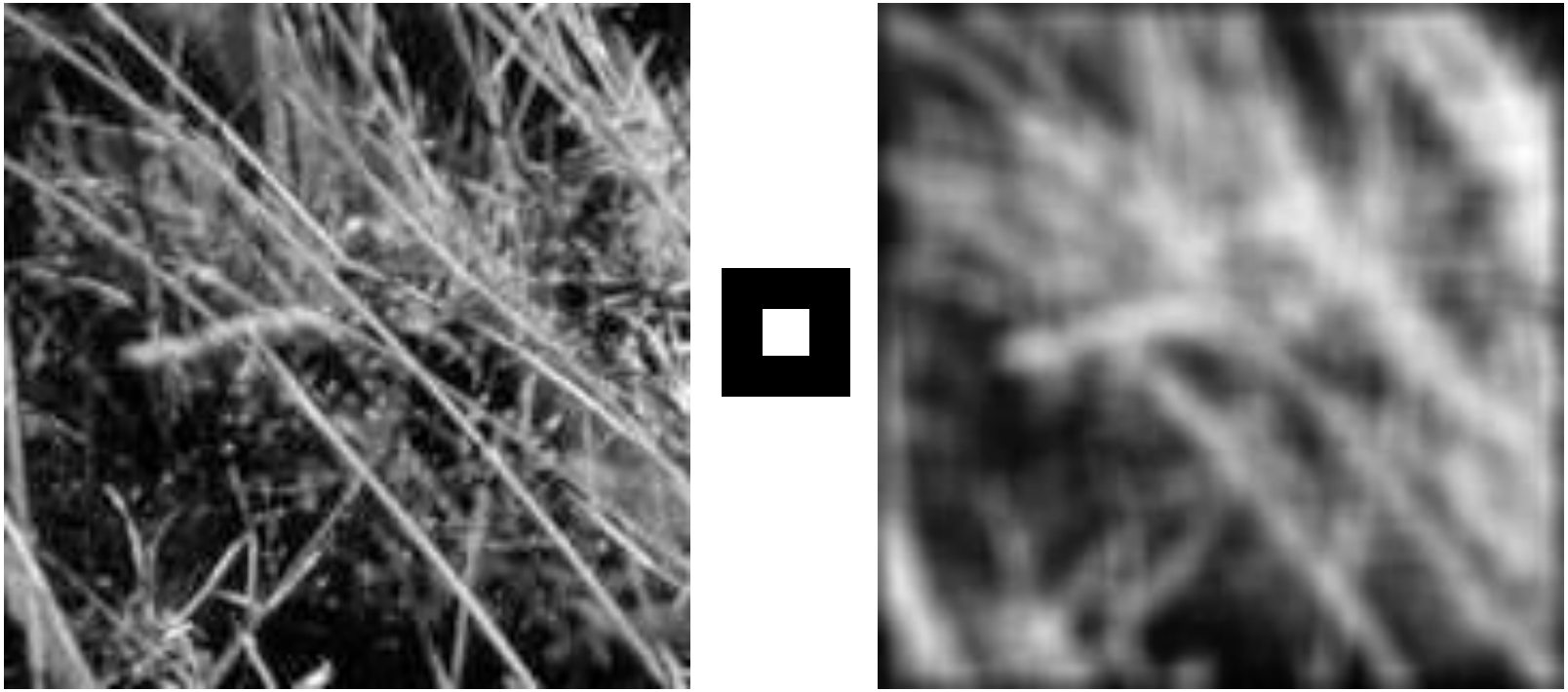
Average Filter

- Mask with positive entries, that sum 1.
- Replaces each pixel with an average of its neighborhood.
- If all weights are equal, it is called a BOX filter.

F

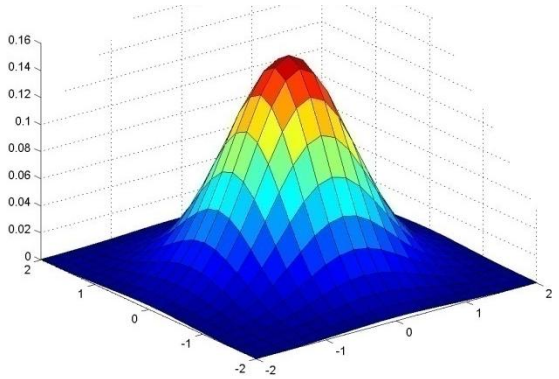
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

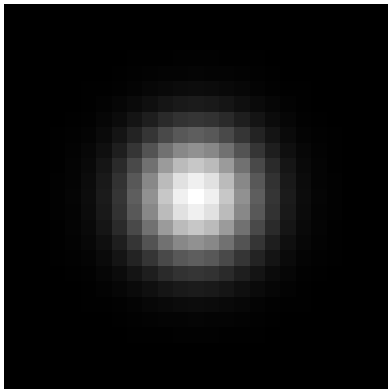
Smoothing with box filter revisited



Source: D. Forsyth

Gaussian Kernel

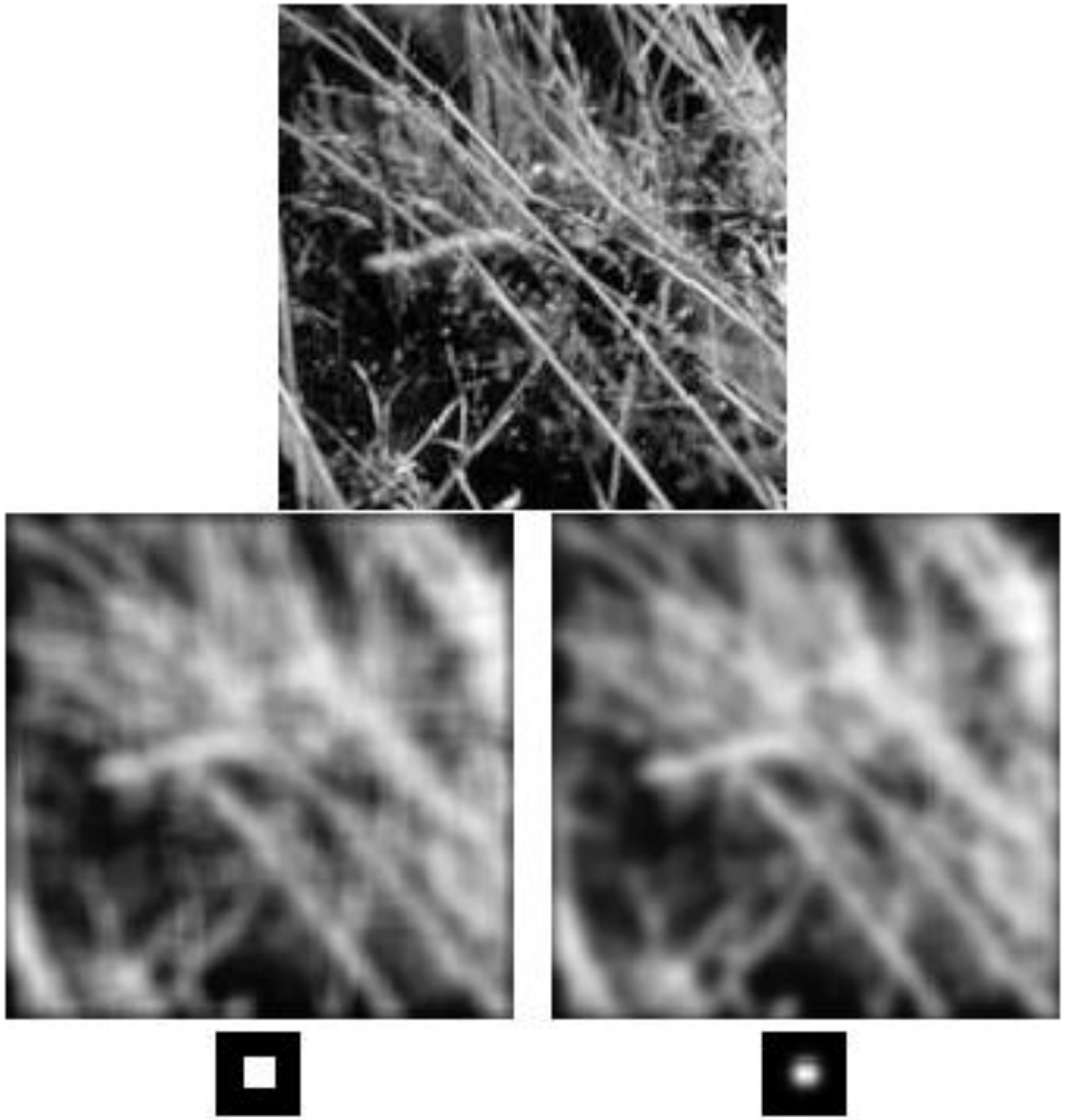




$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

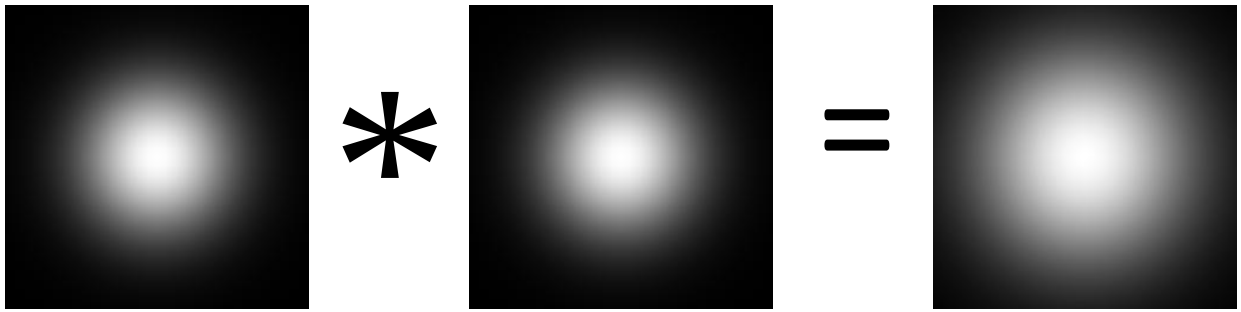
Source: C. Rasmussen

Mean vs. Gaussian filtering



Gaussian filter

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



  - **Convolving two times with Gaussian kernel of width σ = convolving once with kernel of width $\sigma\sqrt{2}$**

Source: K. Grauman

Separability of the Gaussian filter

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

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

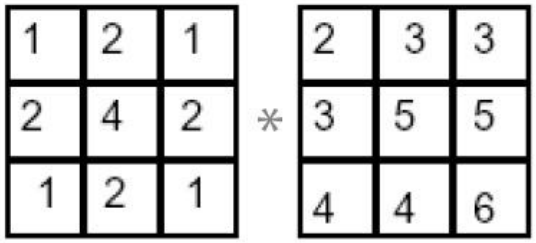
The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

In this case, the two functions are the (identical) 1D Gaussian

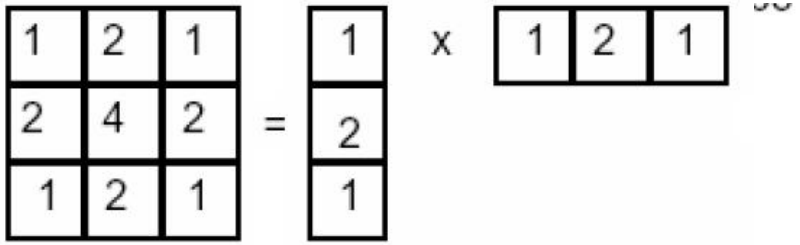
Source: D. Lowe

Separability example

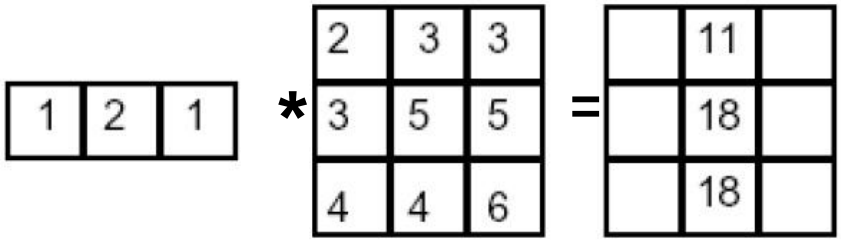
2D convolution
(center location only)



The filter factors into a product of 1D filters:



Perform convolution along rows:



Followed by convolution along the remaining column:

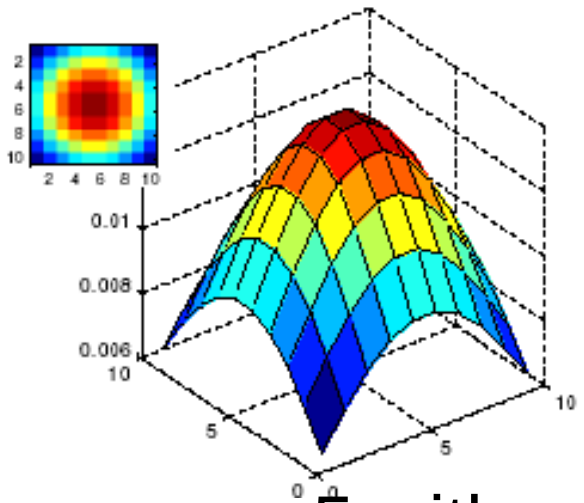
Source: K. Grauman

Why is separability useful?

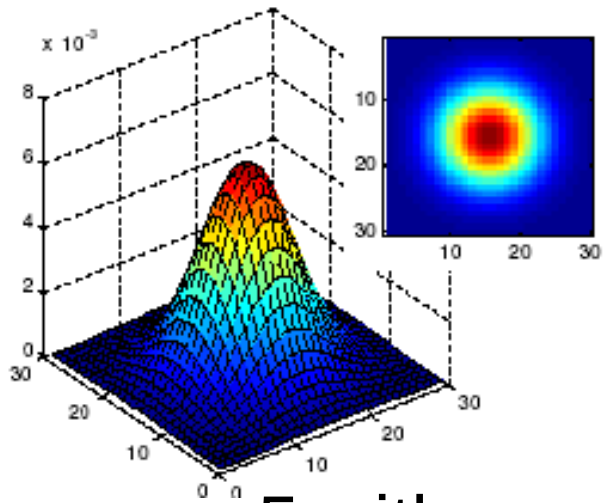
- What is the complexity of filtering an $n \times n$ image with an $m \times m$ kernel?
  - $O(n^2 m^2)$
- What if the kernel is separable?
  - $O(n^2 m)$

Gaussian filters

- What parameters matter here?
- **Size** of kernel or mask
  - Note, Gaussian function has infinite support, but discrete filters use finite kernels



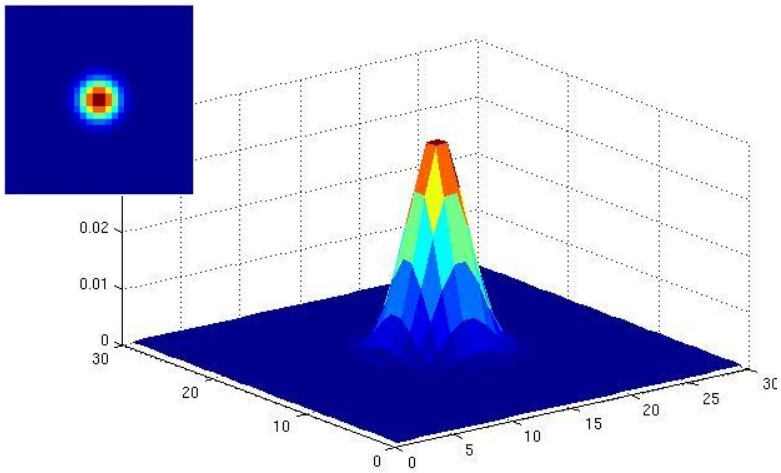
σ = 5 with 10 x 10 kernel



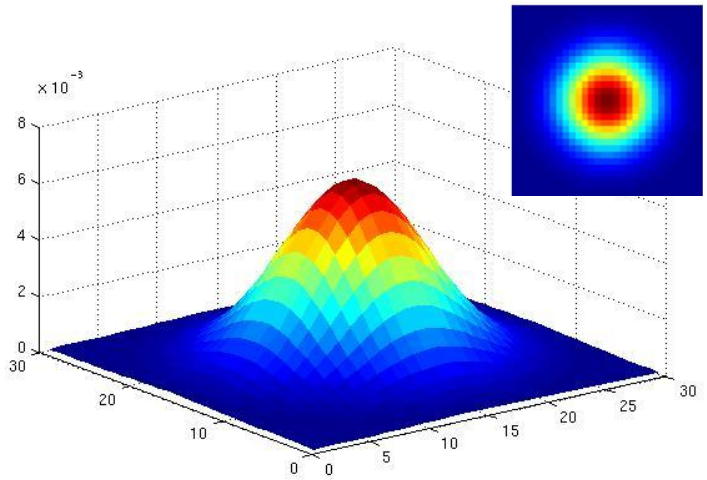
σ = 5 with 30 x 30 kernel

Gaussian filters

- What parameters matter here?
- **Variance** of Gaussian: determines extent of smoothing



σ = 2 with 30 x 30 kernel

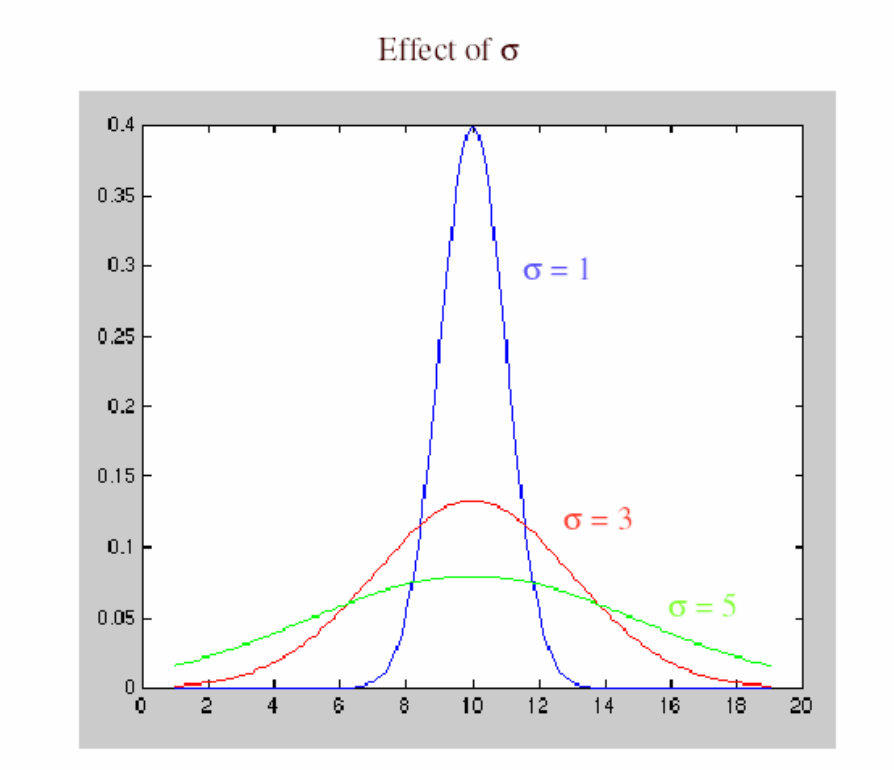


σ = 5 with 30 x 30 kernel

Gaussian filters

How big should the filter be?

- Values at edges should be near zero ← important!
- Rule of thumb for Gaussian: set filter half-width to about 3 σ

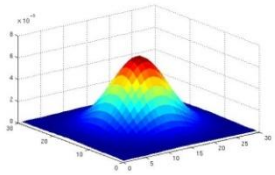


Source: Derek Hoiem

Matlab

```
>> hsize = 10;
>> sigma = 5;
>> h = fspecial('gaussian' hsize, sigma);
```

```
>> mesh(h);
```



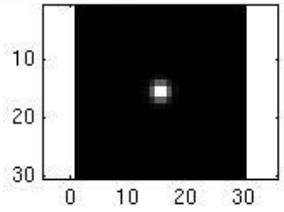
```
>> imagesc(h);
```

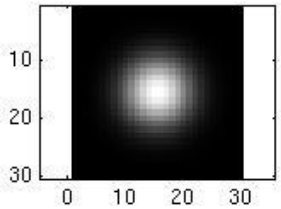
```
>> outim = imfilter(im, h); % correlation
>> imshow(outim);
```

outim

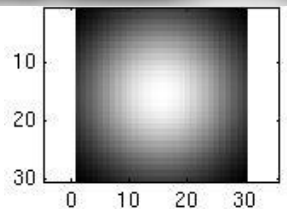
Smoothing with a Gaussian

Parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.





...




```
for sigma=1:3:10
   h = fspecial('gaussian', fsize, sigma);
   out = imfilter(im, h);
   imshow(out);
   pause;
end
```

Noise

Original

Salt and pepper noise

Impulse noise

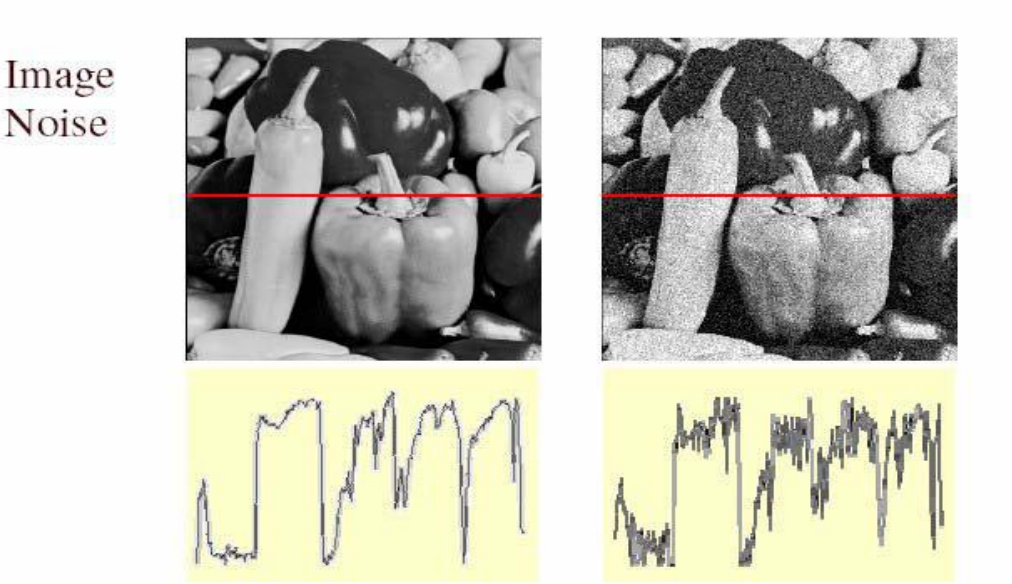
Gaussian noise

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

Source: S. Seitz

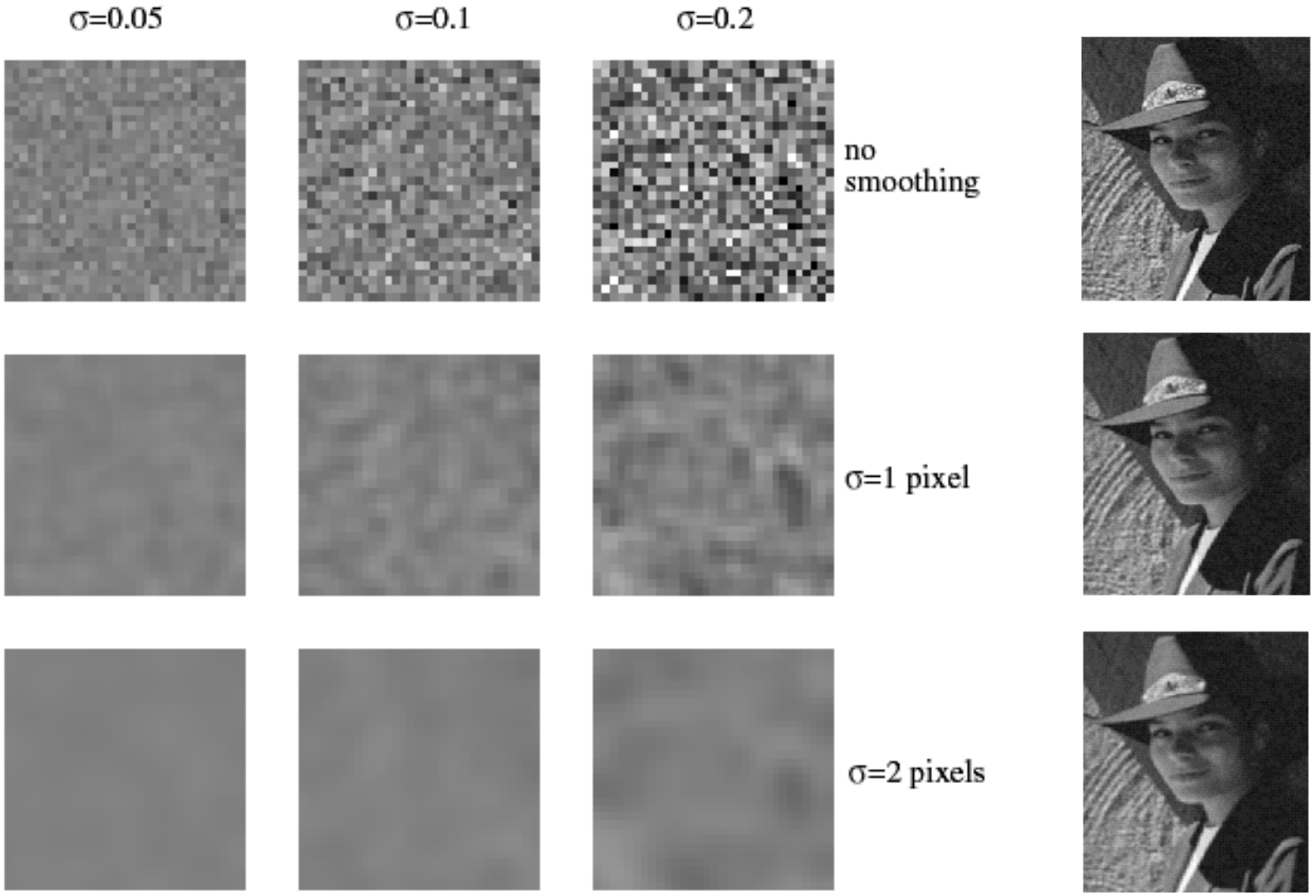
Gaussian noise

- Mathematical model: sum of many independent factors
- Good for small standard deviations
- Assumption: independent, zero-mean noise



Source: M. Hebert

Reducing Gaussian noise



Smoothing with larger standard deviations suppresses noise, but also blurs the image

Reducing salt-and-pepper noise

3x3

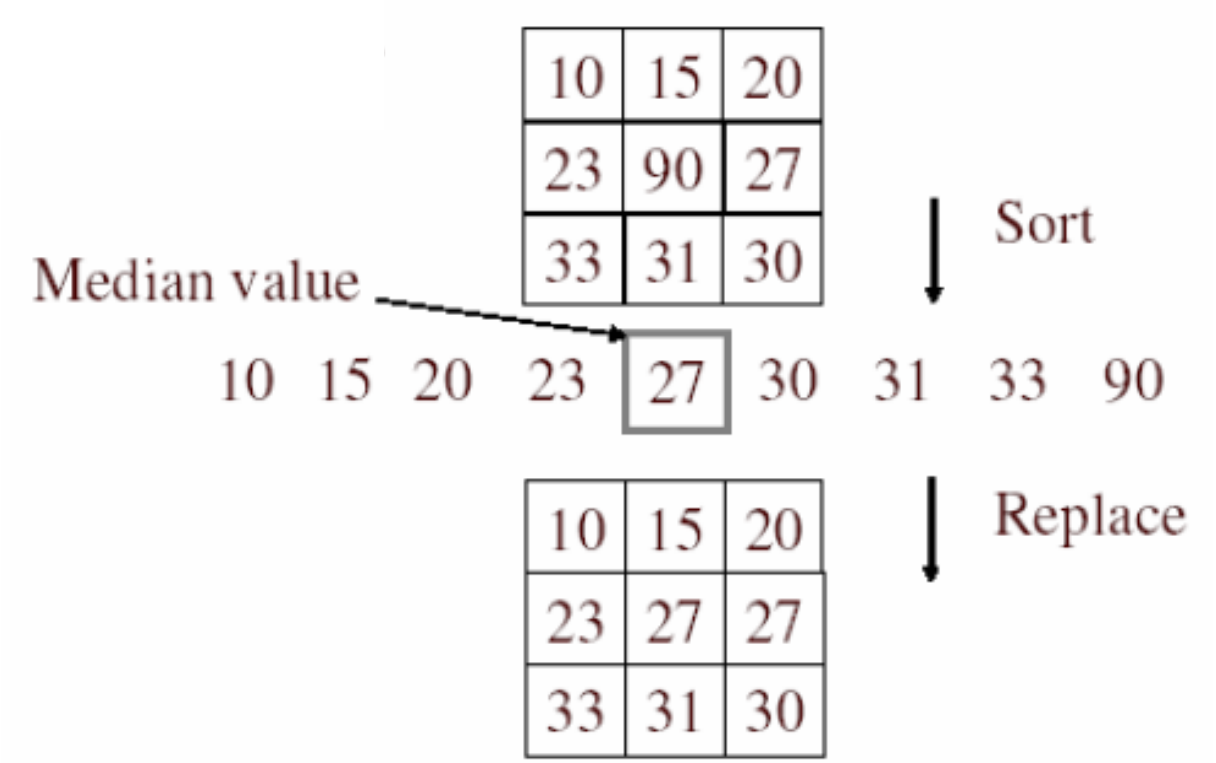
5x5

7x7

- What's wrong with the results?

Alternative idea: Median filtering

- **A median filter operates over a window by selecting the median intensity in the window**

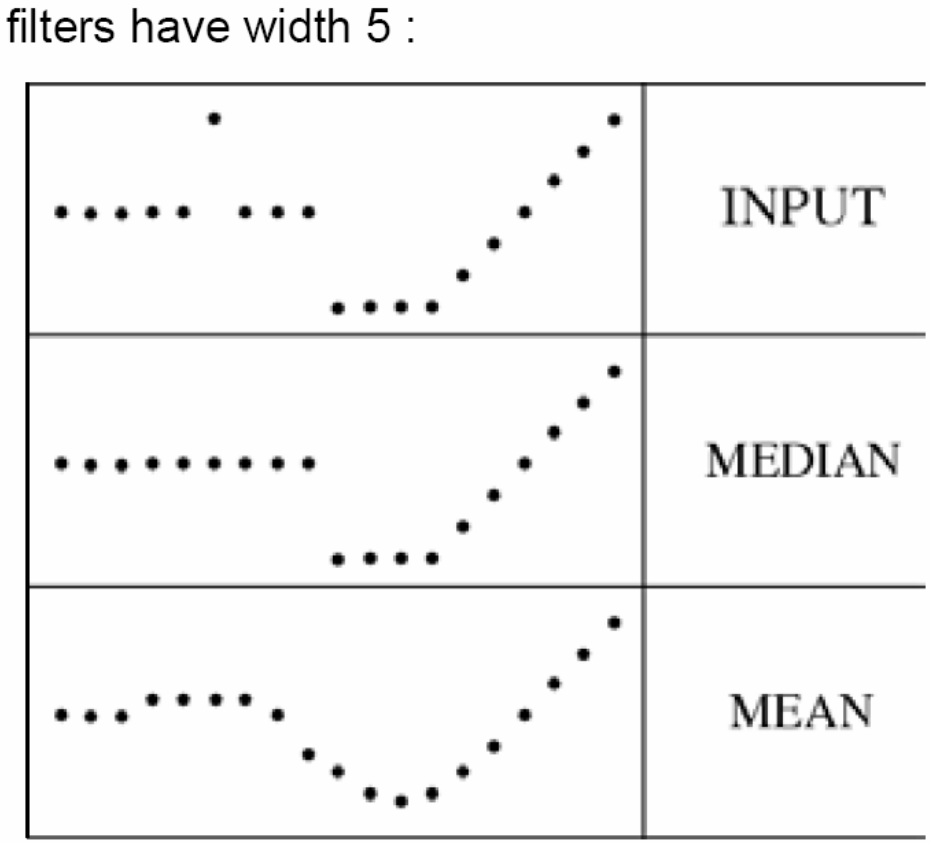


- Is median filtering linear?

Source: K. Grauman

Median filter

- **What advantage does median filtering have over Gaussian filtering?**
  - **Robustness to outliers**

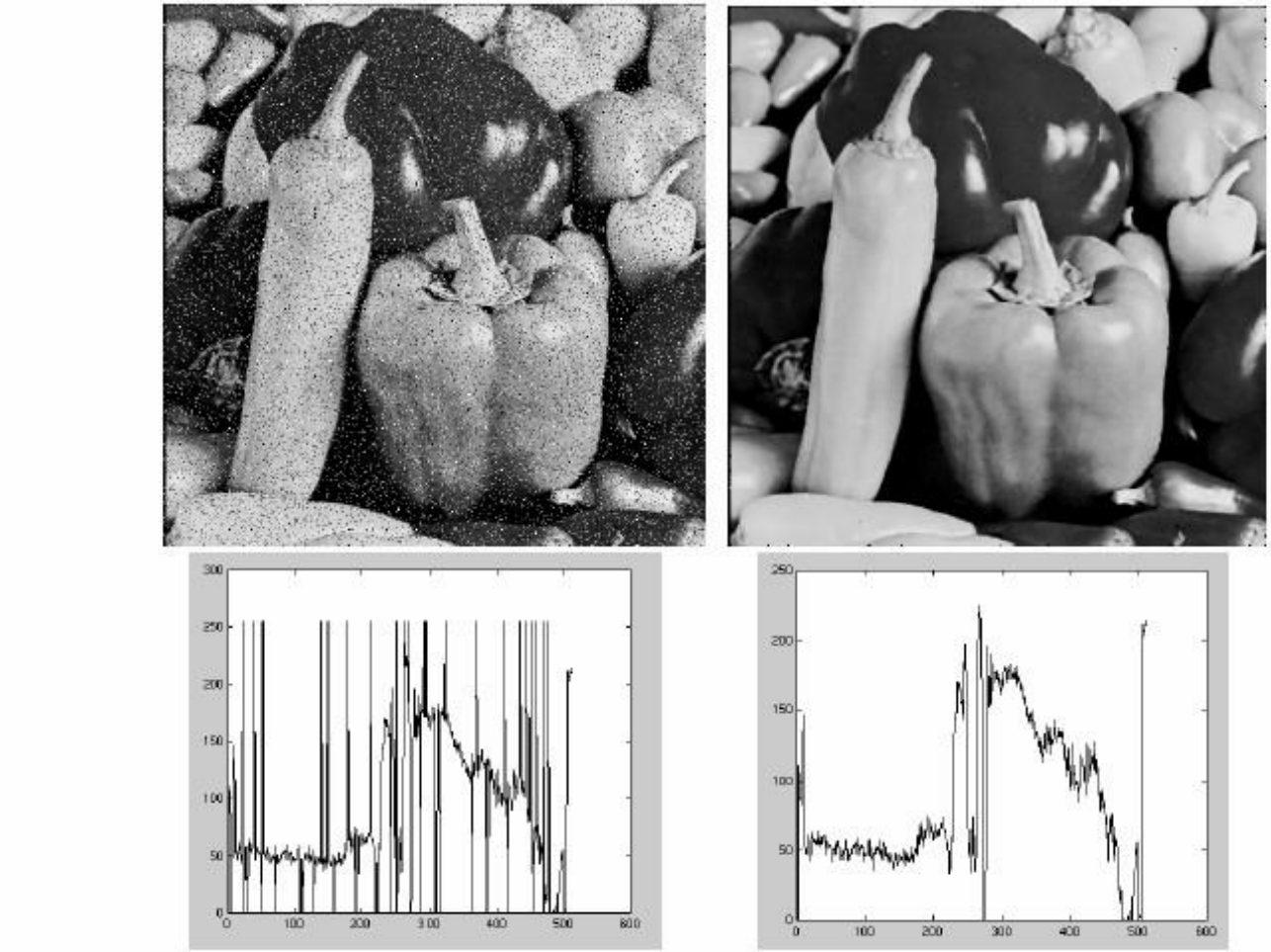


Source: K. Grauman

Median filter

Salt-and-pepper noise

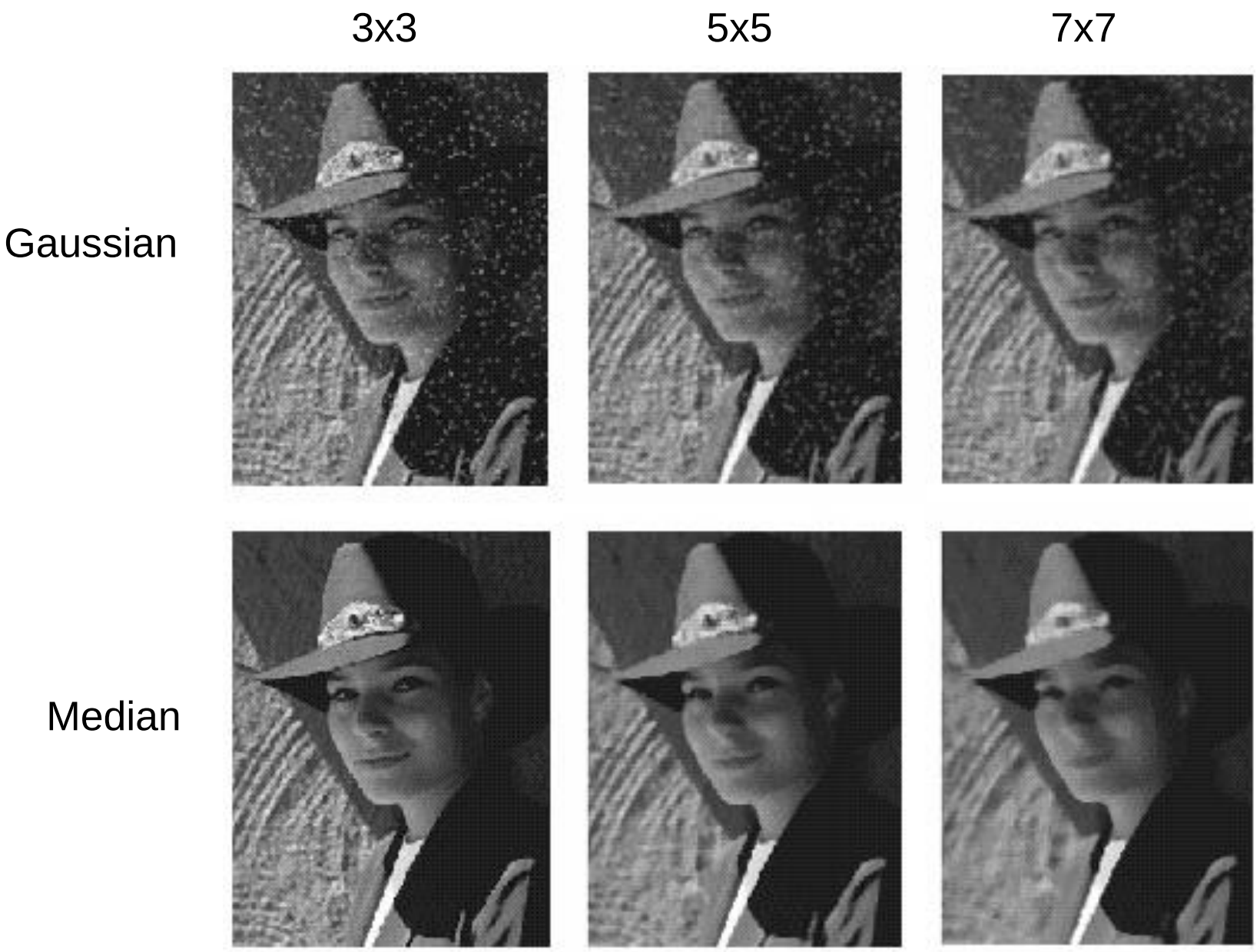
Median filtered



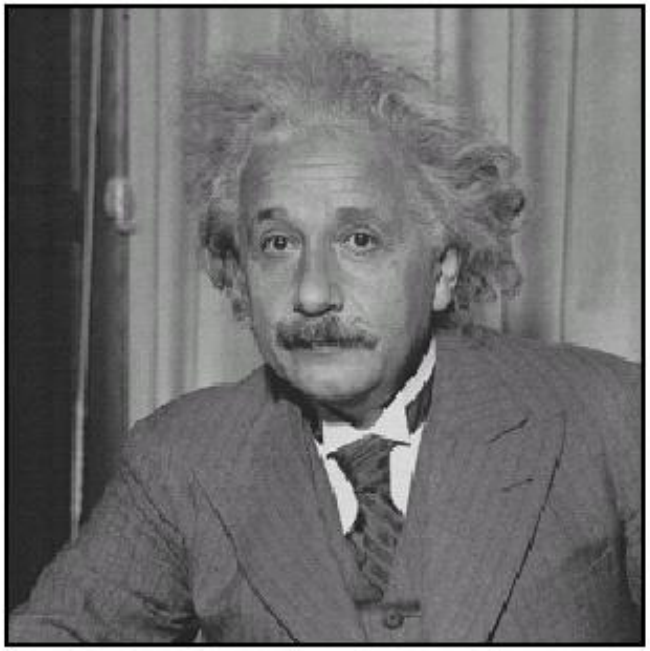
- **MATLAB:** medfilt2(image, [h w])

Source: M. Hebert

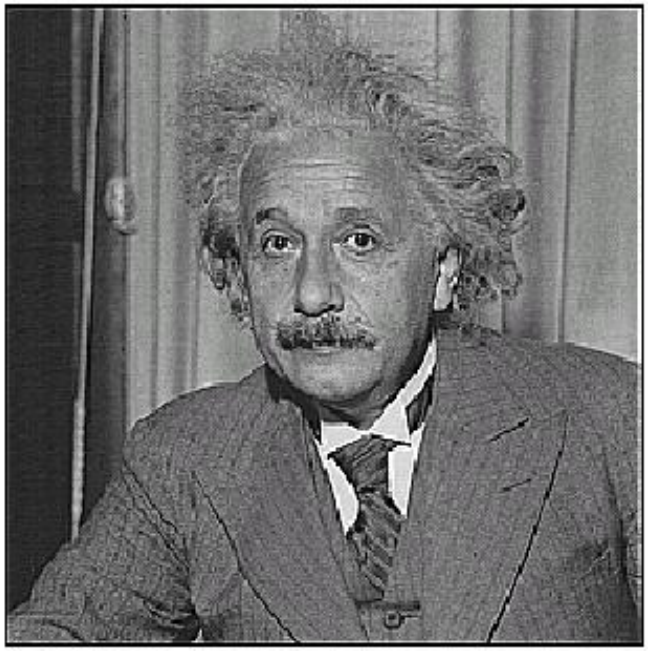
Gaussian vs. median filtering



Sharpening revisited



before



after

Source: D. Lowe

Sharpening revisited

- What does blurring take away?

–

=

Let's add it back:

+ α

=

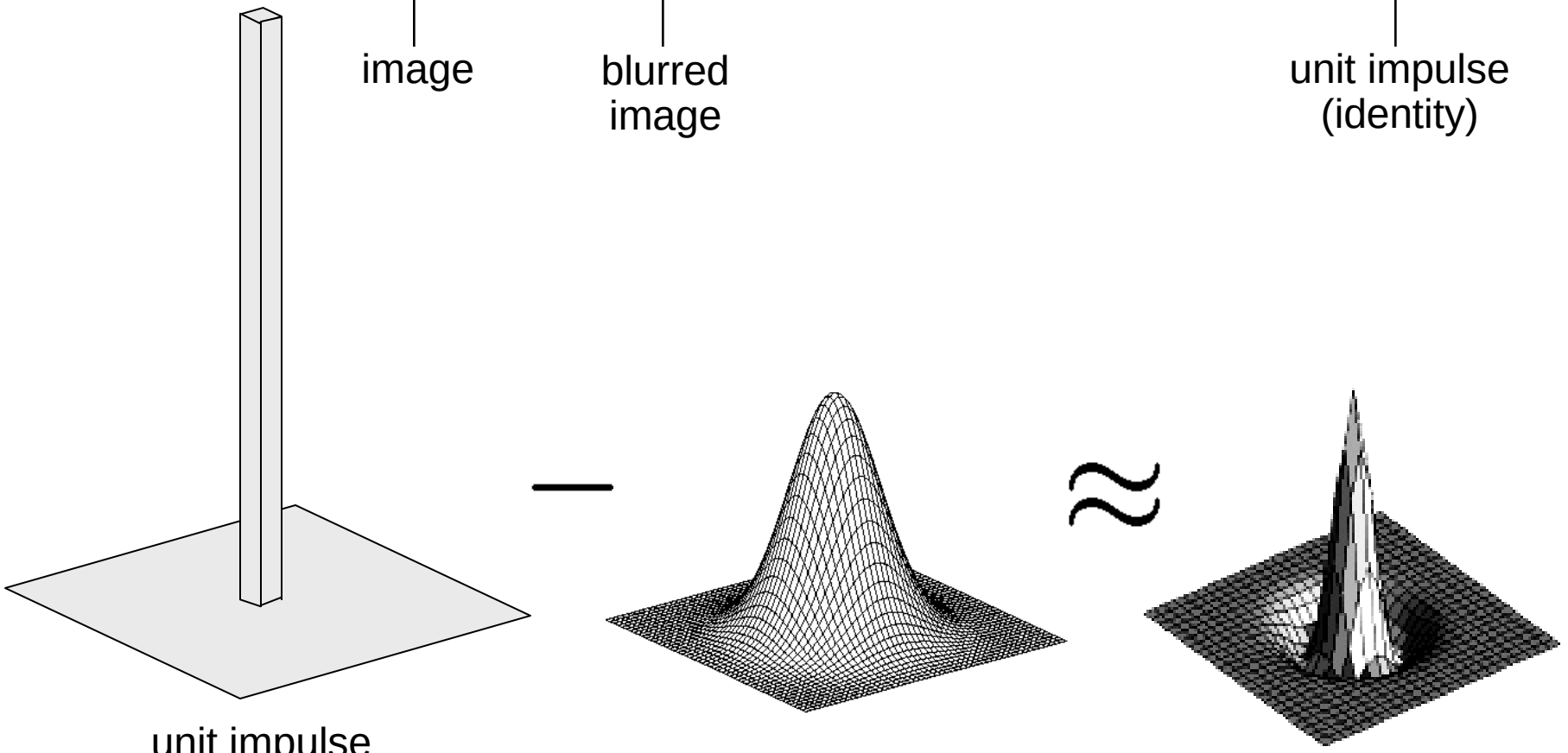
Unsharp mask filter

$$f + \alpha(f - f * g) = (1+\alpha) f - \alpha f * g = f * ((1+\alpha)e - \alpha g)$$

image

blurred image

unit impulse (identity)



unit impulse

Gaussian

Laplacian of Gaussian

Sharpen filter

Finding Simple Patterns with Filters

- A filter is a pattern detector
  - strong filter responses suggest the image pattern "looks like" the filter
- It's not a very good one, because
  - it's linear, so brighter image patches give stronger filter responses
  - it can respond to constant image patches
    - if the mean of the filter weights isn't zero

Normalized Correlation

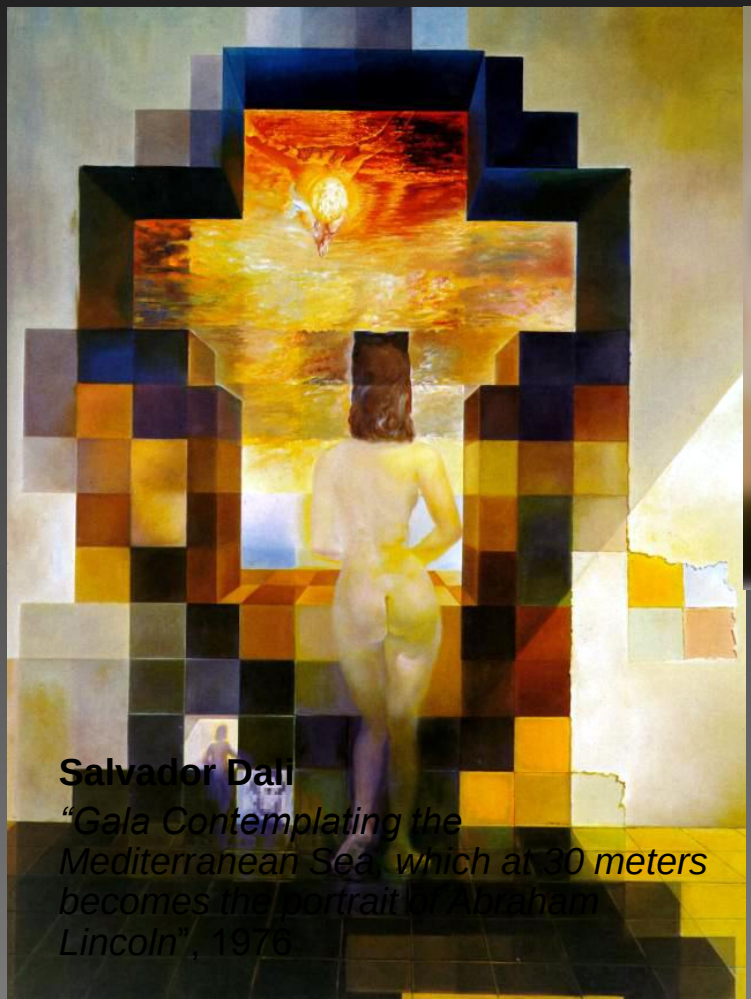
- **Strategy (Normalized correlation):**
  - **use a zero mean filter (subtract the average of the filter weights)**
  - **for each image patch, subtract the mean**
  - **divide the filter response by the magnitude of the result**
  - **Output**
    - **value close to 1 --- image pattern like the filter kernel**
    - **value close to -1 --- image pattern like contrast-reversed filter kernel**
- **This is a great way to build simple, fast pattern detectors**
  - **that require little computation**

FIGURE 4.16: Examples of Freeman *et al.*'s system controlling a television set. Each state is illustrated with what the television sees on the **left** and what the user sees on the **right**. In (**a**), the television is asleep, but a process is watching the user. An open hand causes the television to come on and show its user interface panel (**b**). Focus on the panel tracks the movement of the user's open hand in (**c**), and the user can change channels by using this tracking to move an icon on the screen in (**d**). Finally, the user displays a closed hand in (**e**) to turn off the set. *This figure was originally published as Figure 12 of "Computer Vision for Interactive Computer Graphics," W. Freeman* et al., *IEEE Computer Graphics and Applications, 1998* © *IEEE, 1998.*

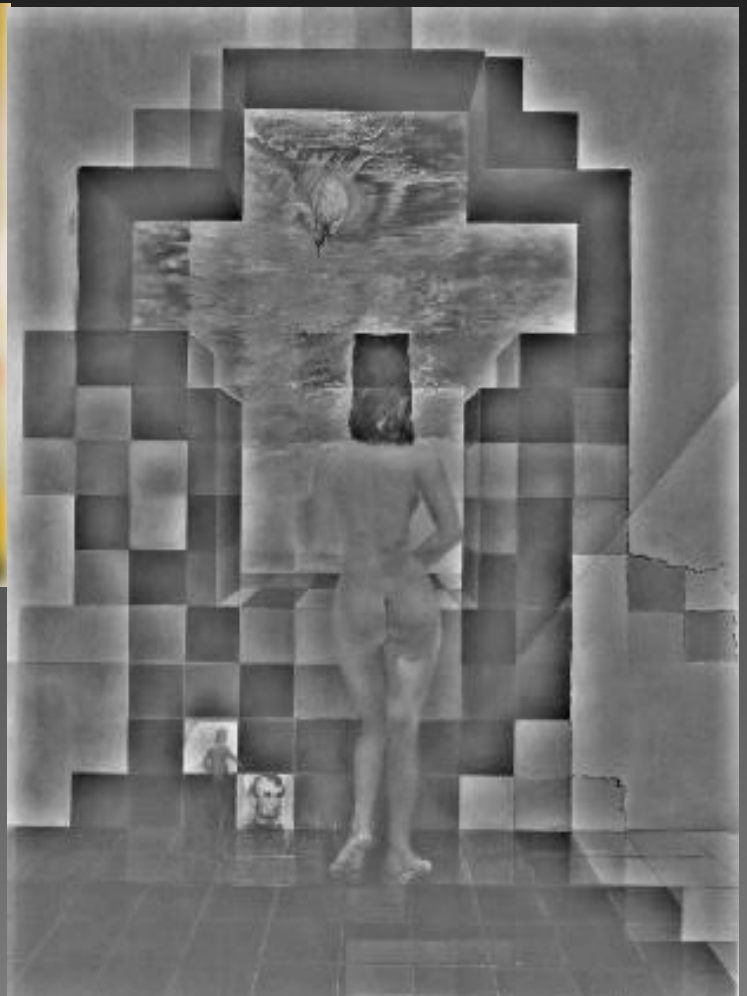
Fourier transform

Fourier Transform

Fourier analysis is a method by which any two dimensional luminance image can be analyzed into the sum of a set of sinusoidal gratings that differ in **spatial frequency**, **orientation**, **amplitude** and **phase**.

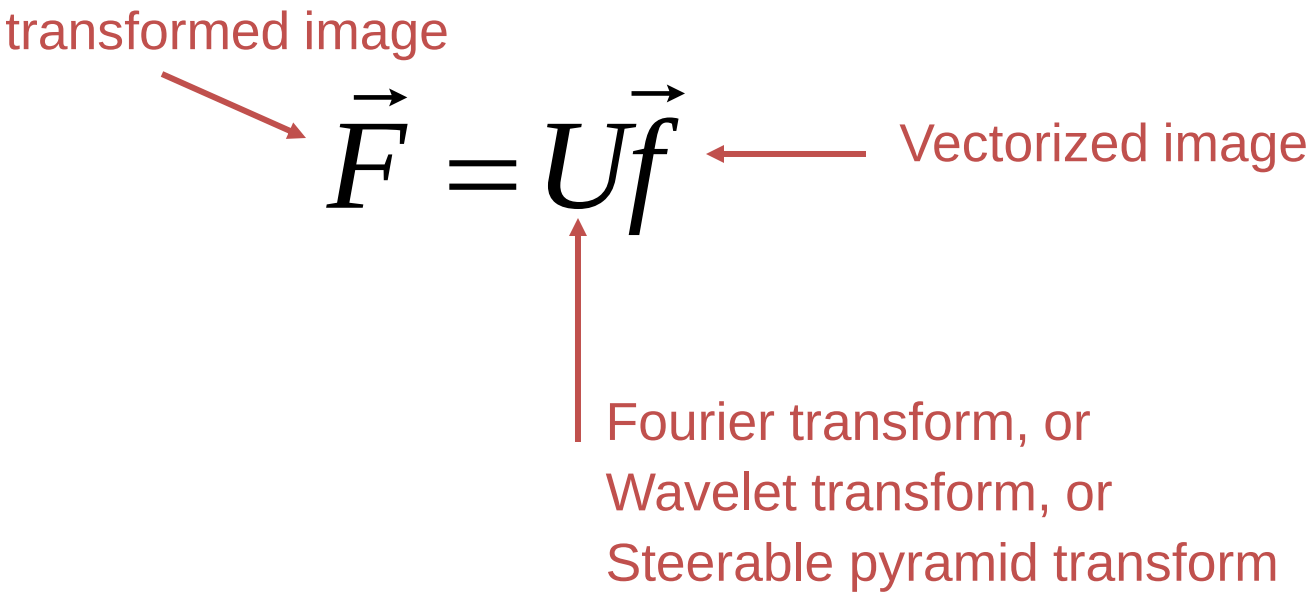


Salvador Dali
"Gala Contemplating the Mediterranean Sea, which at 30 meters becomes the portrait of Abraham Lincoln", 1976



Linear image transformations

- **In analyzing images, it's often useful to make a change of basis.**



Self-inverting transforms

Same basis functions are used for the inverse transform

$$\vec{f} = U^{-1}\vec{F}$$

$$= U^{+}\vec{F}$$

U transpose and complex conjugate

An example of such a transform: the Discrete Fourier transform

Forward transform

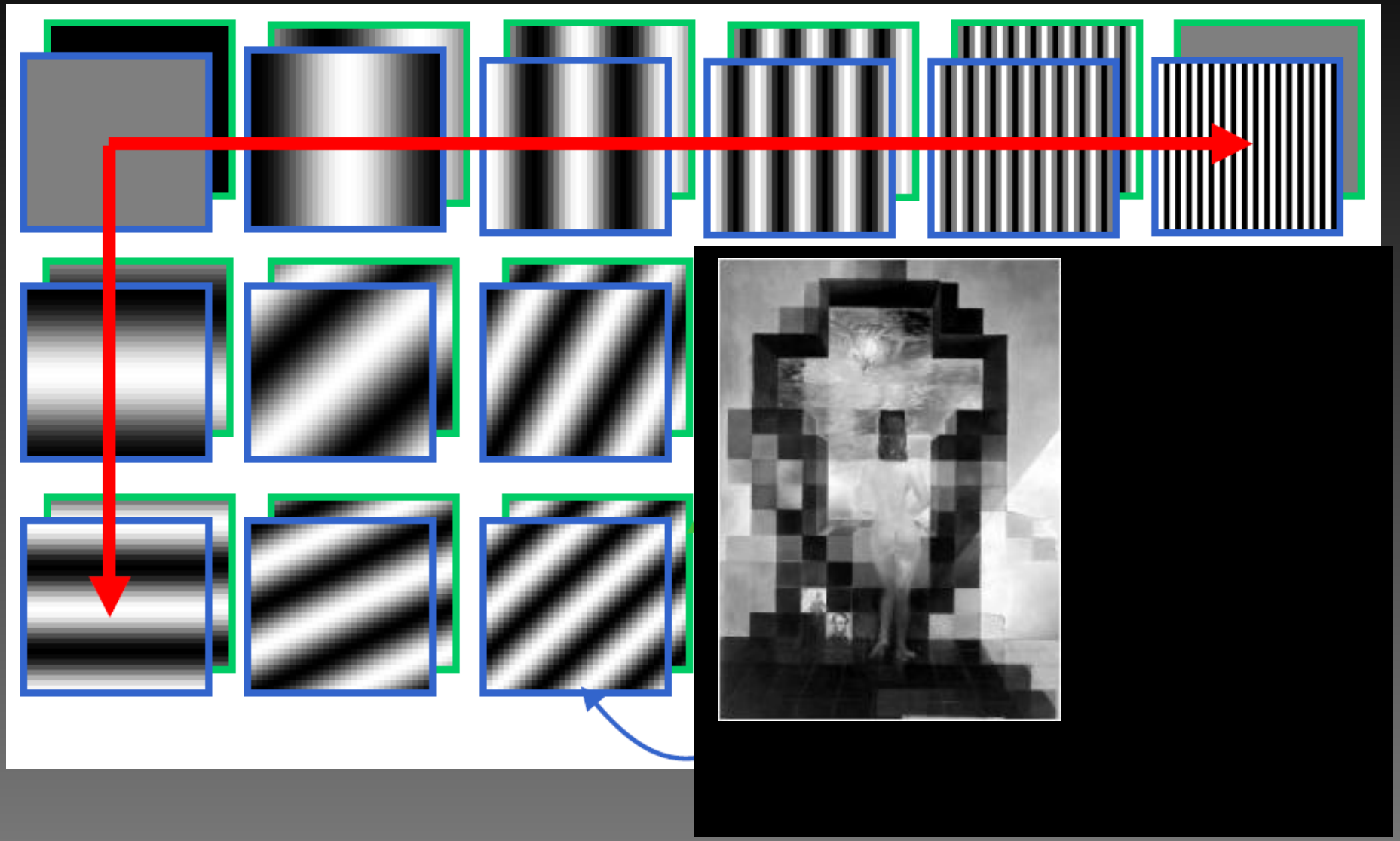
$$F[m,n] = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f[k,l] e^{-\pi i \left(\frac{km}{M} + \frac{\ln}{N} \right)}$$

Inverse transform

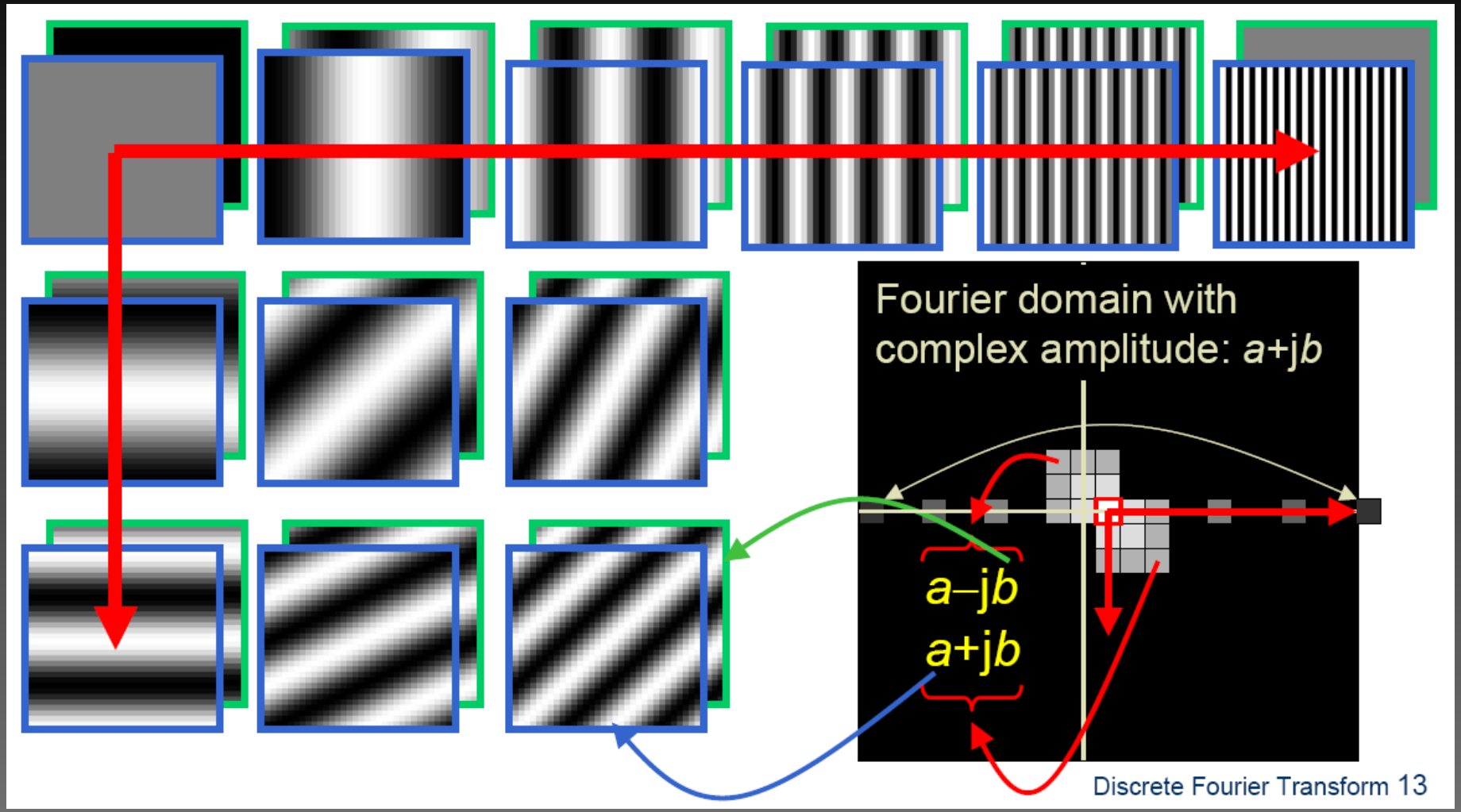
$$f[k,l] = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} F[m,n] e^{+\pi i \left(\frac{km}{M} + \frac{\ln}{N} \right)}$$

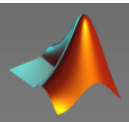
Sinusoidal gratings as the “primitives” of an image

A nice set of basis: Teases away fast vs. slow changes in the image.



Sinusoidal gratings as the “primitives” of an image



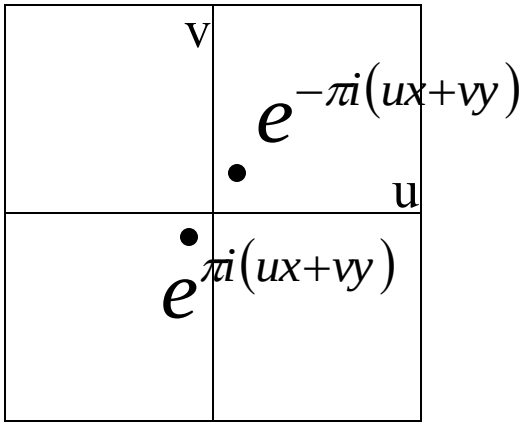


imagesc(log(abs(fftshift(fft2(im)))));

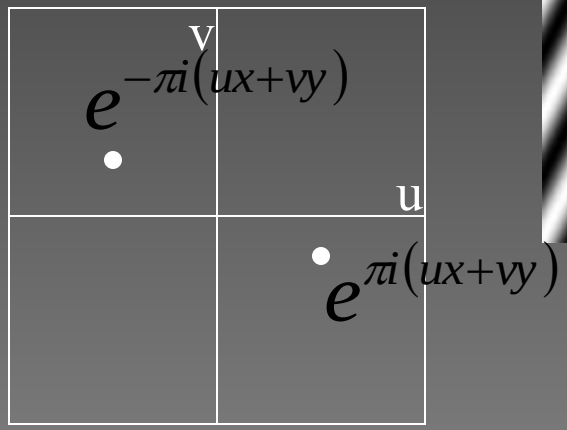
Slide from A. Efros

To get some sense of what basis elements look like, we plot a basis element --- or rather, its real part ---

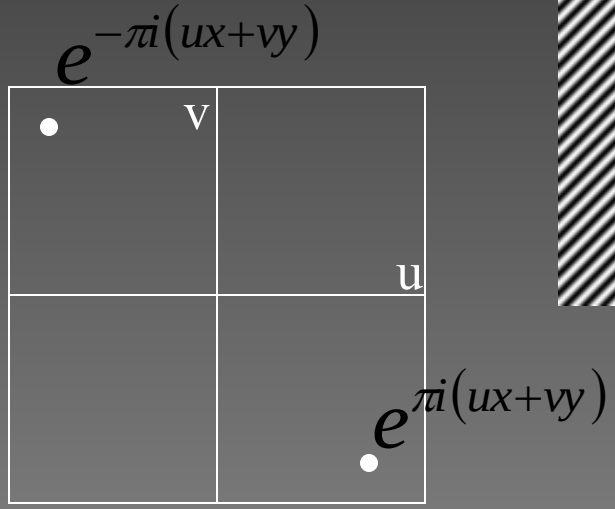
as a function of x,y for some fixed u, v. We get a function that is constant when (ux+vy) is constant. The magnitude of the vector (u, v) gives a frequency, and its direction gives an orientation. The function is a sinusoid with this frequency along the direction, and constant perpendicular to the direction.



Here u and v are larger than in the previous slide.

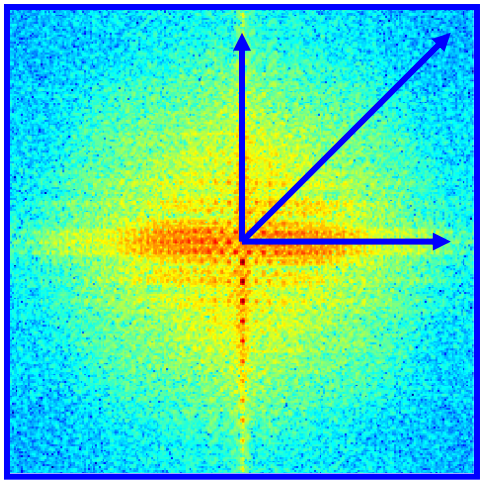


And larger still...



How to interpret a Fourier Spectrum

Vertical orientation



0

45 deg.

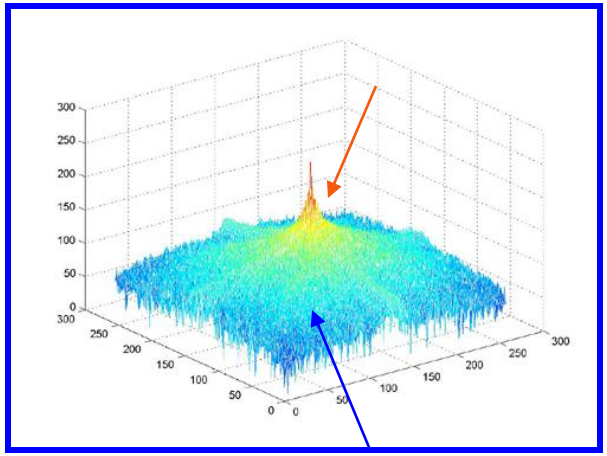
Horizontal orientation

0

f_{max}

fx in cycles/image

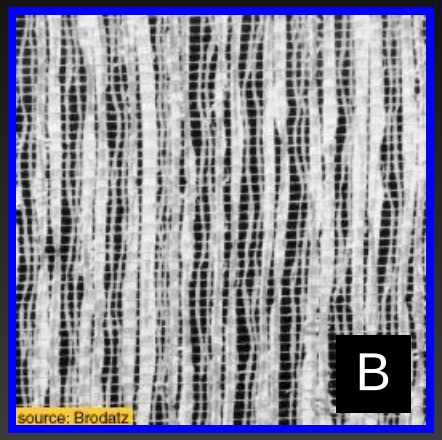
Low spatial frequencies

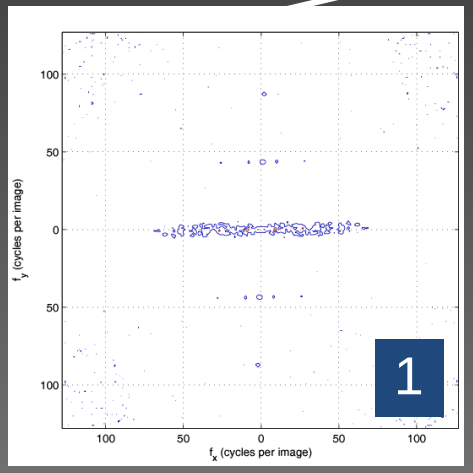


High spatial frequencies

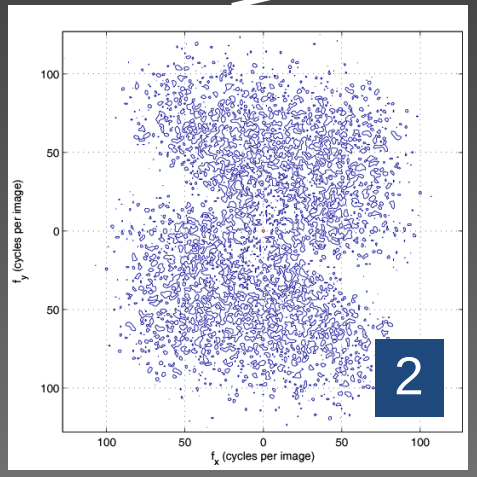
Log power spectrum

Fourier Amplitude Spectrum

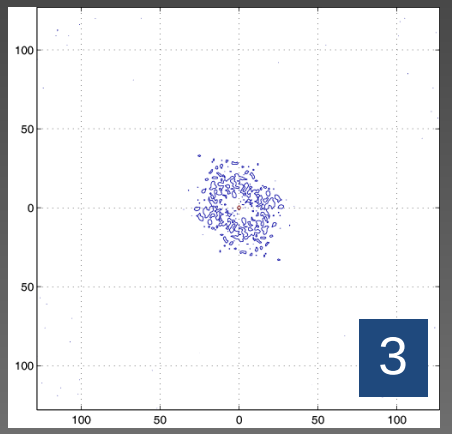




fx(cycles/image pixel size)



fx(cycles/image pixel size)



fx(cycles/image pixel size)

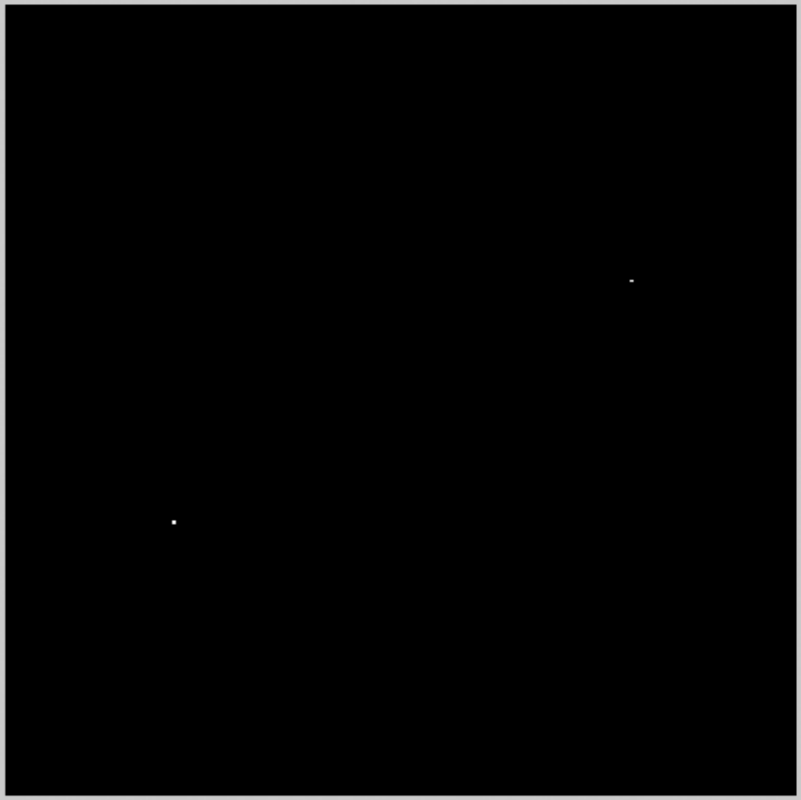
Two examples of image synthesis with Fourier basis

First: randomly sample the Fourier coefficients of an image and reconstruct from those.

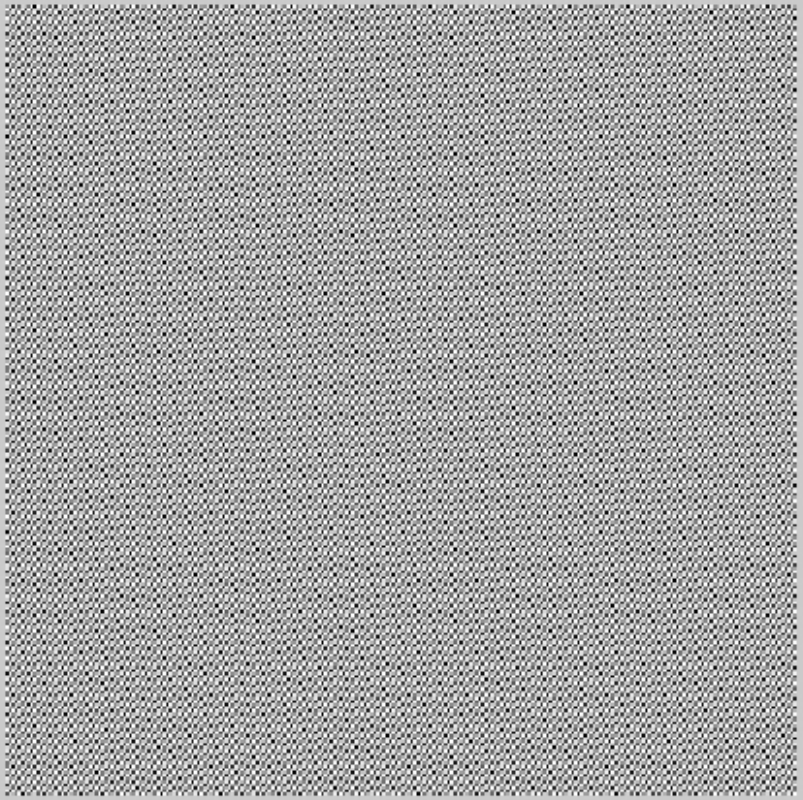
Second: sample Fourier coefficients in descending order of amplitude.

2

2



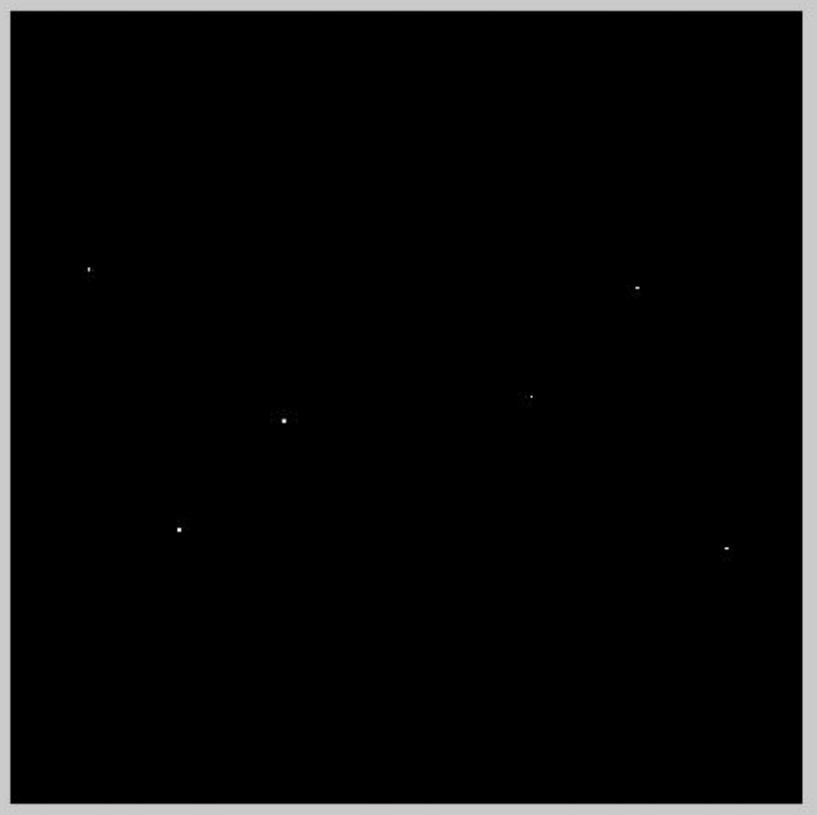
#1: Range [0, 1]
Dims [256, 256]



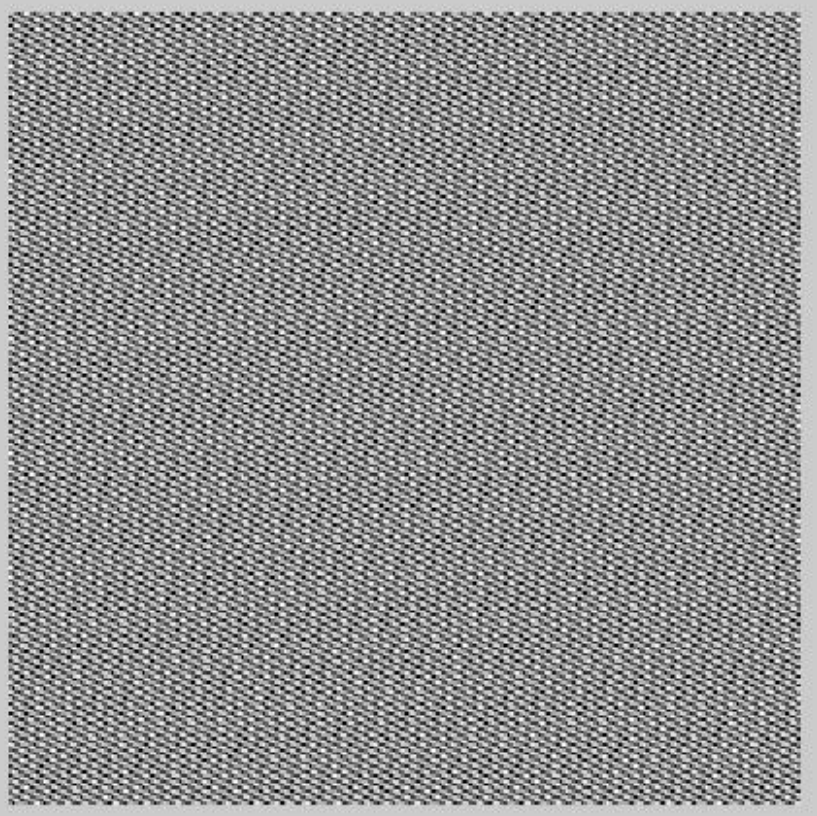
#2: Range [0.000109, 0.0267]
Dims [256, 256]

6

6



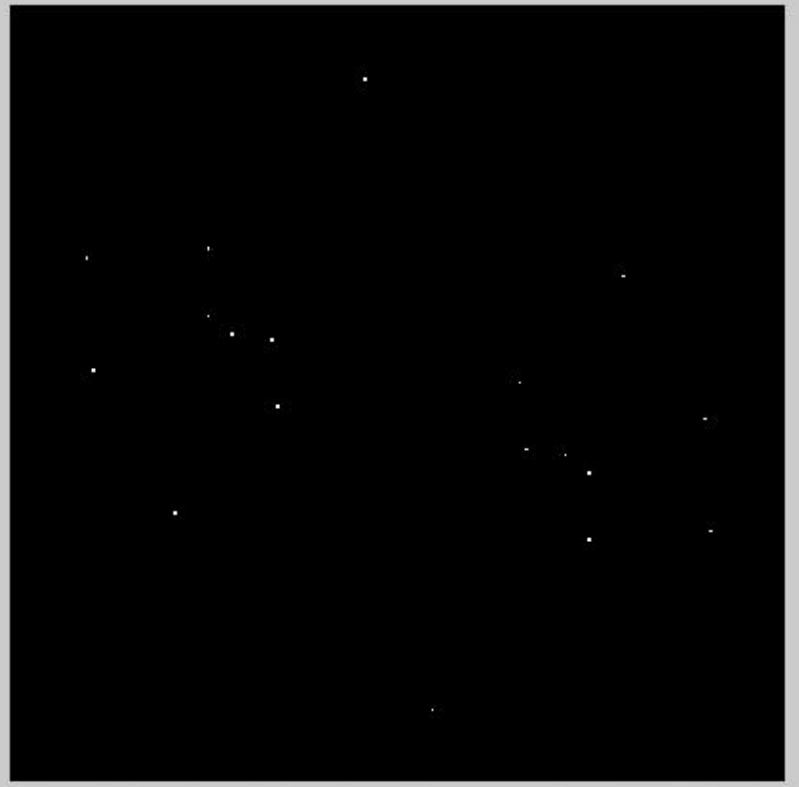
#1: Range [0, 1]
Dims [256, 256]



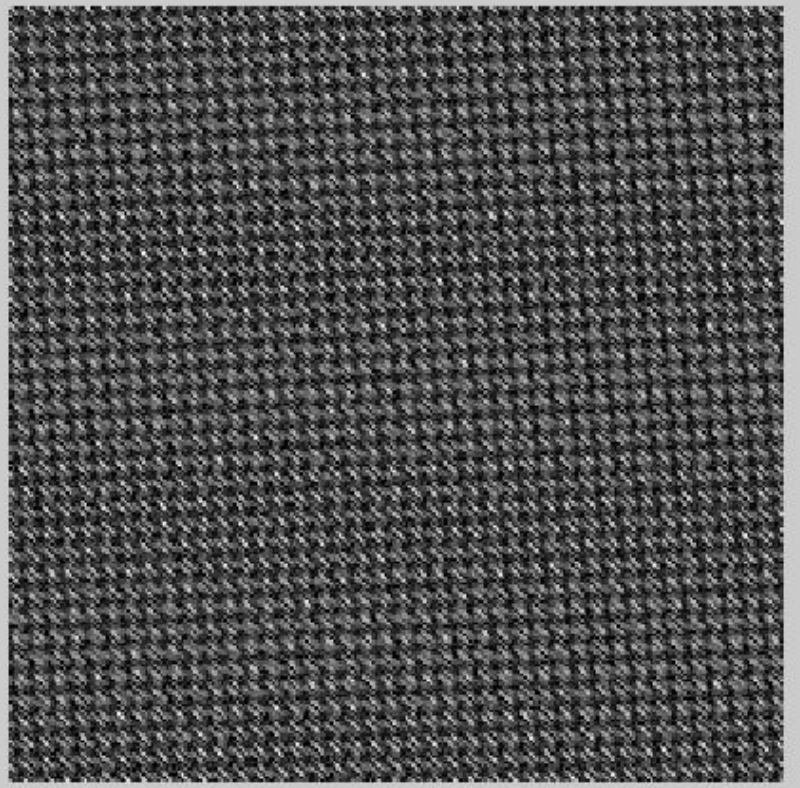
#2: Range [1.89e-007, 0.226]
Dims [256, 256]

18

18



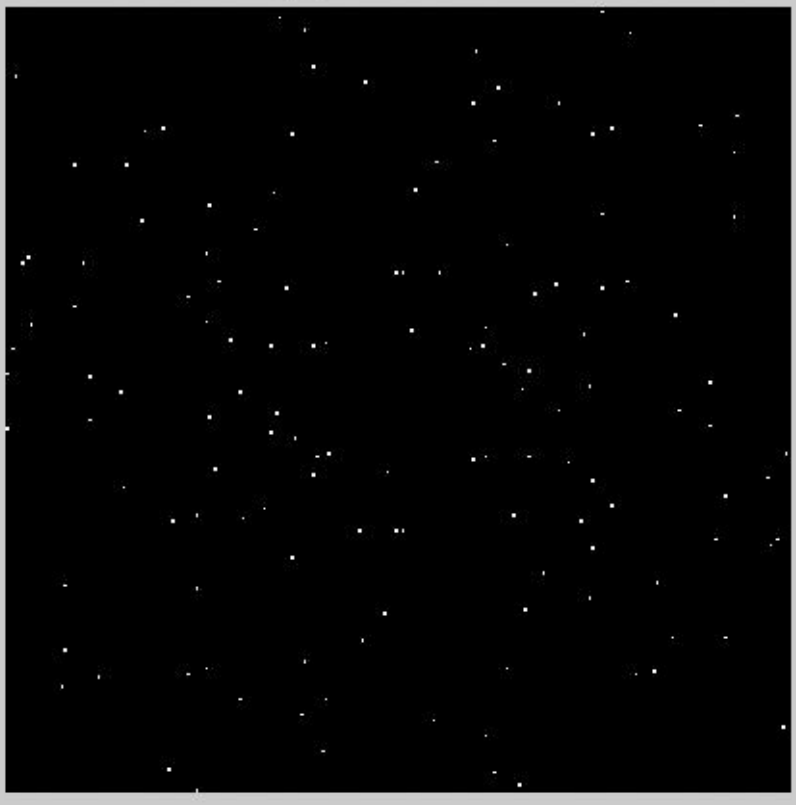
#1: Range [0, 1]
Dims [256, 256]



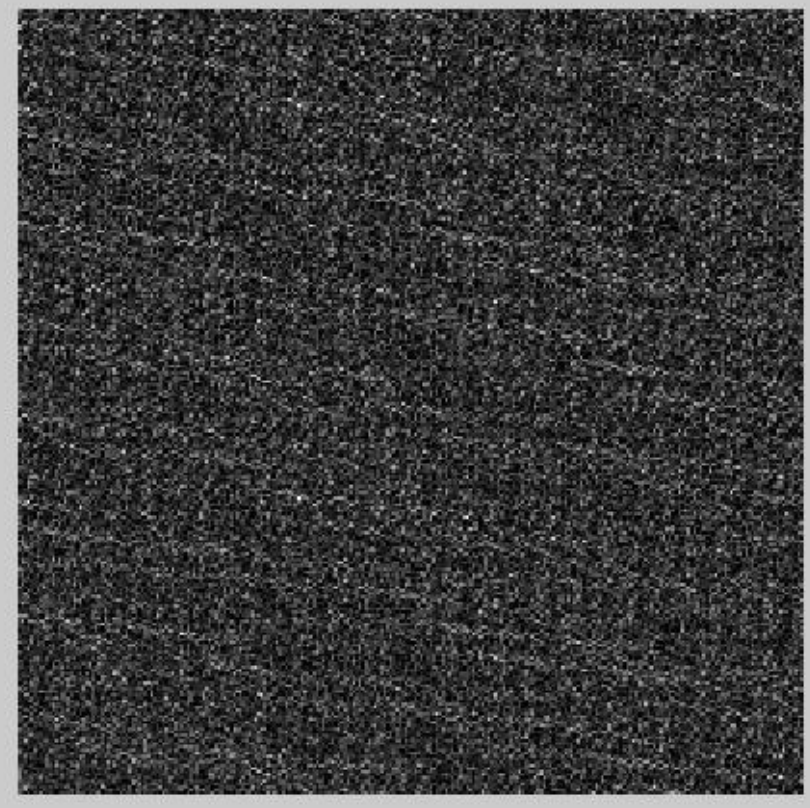
#2: Range [4.79e-007, 0.503]
Dims [256, 256]

136

136



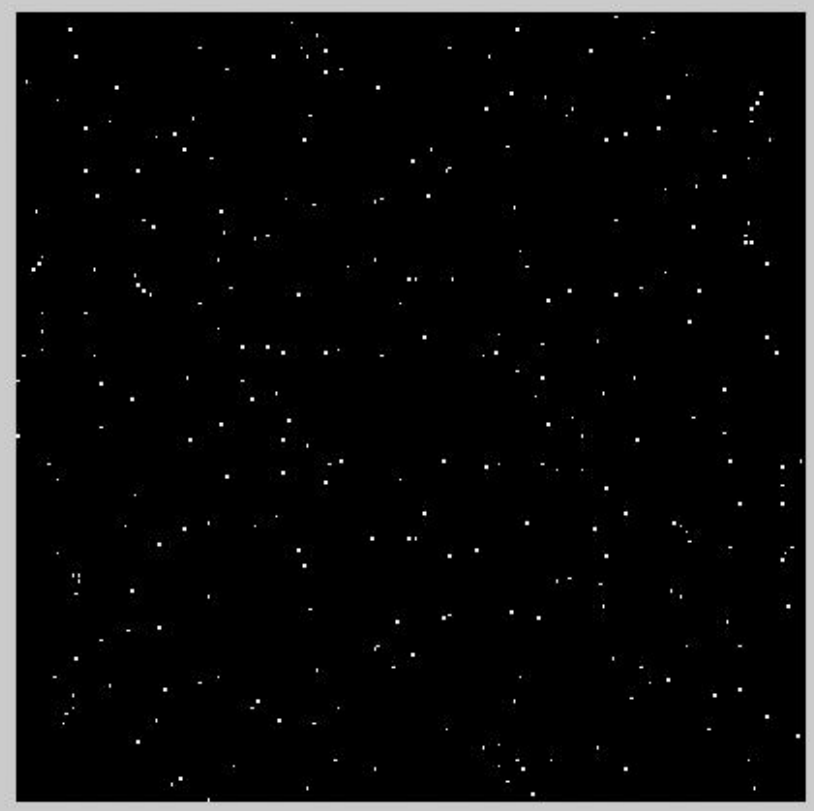
#1: Range [0, 1]
Dims [256, 256]



#2: Range [8.25e-006, 3.48]
Dims [256, 256]

282

282

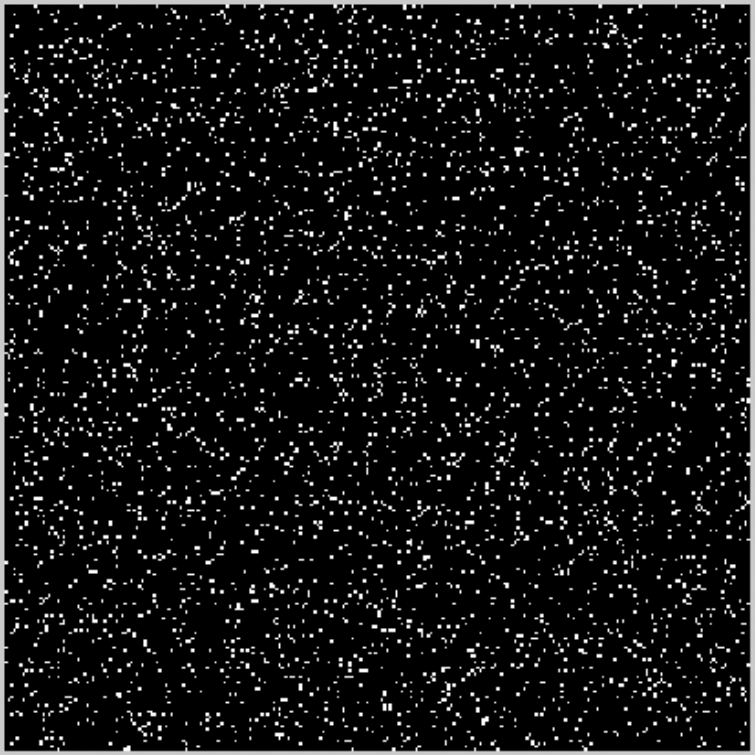


#1: Range [0, 1]
Dims [256, 256]

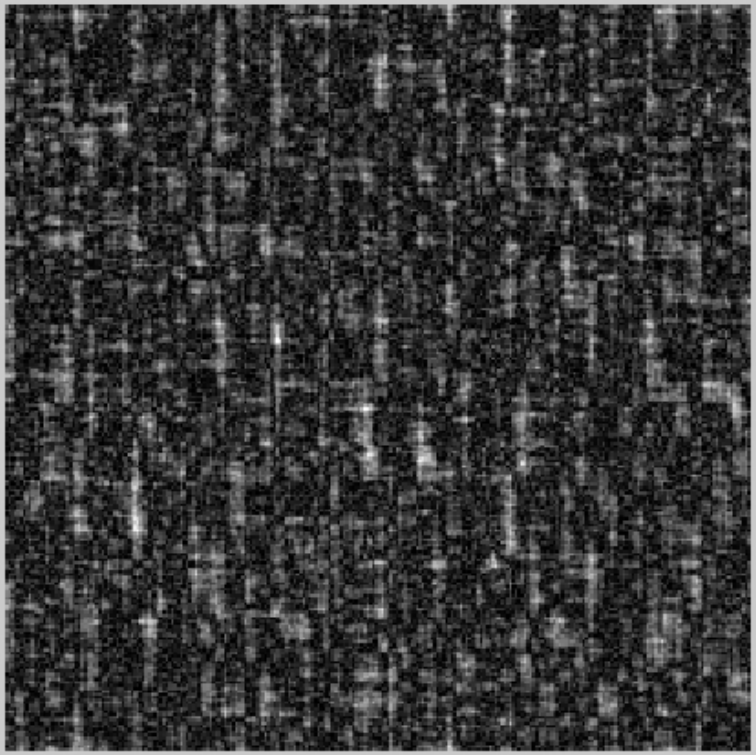
#2: Range [1.39e-005, 5.88]
Dims [256, 256]

4052.

4052



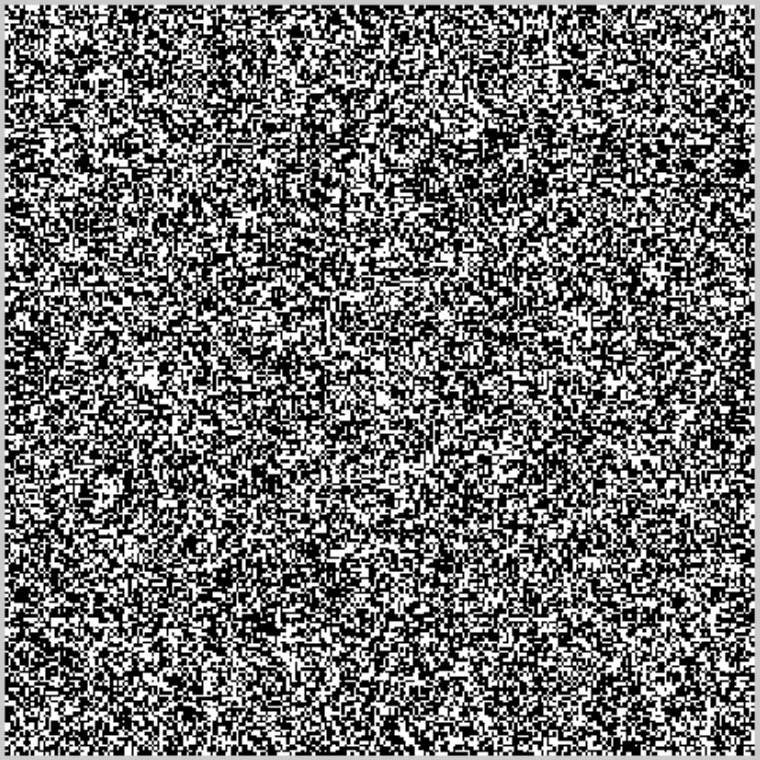
#1: Range [0, 1]
Dims [256, 256]



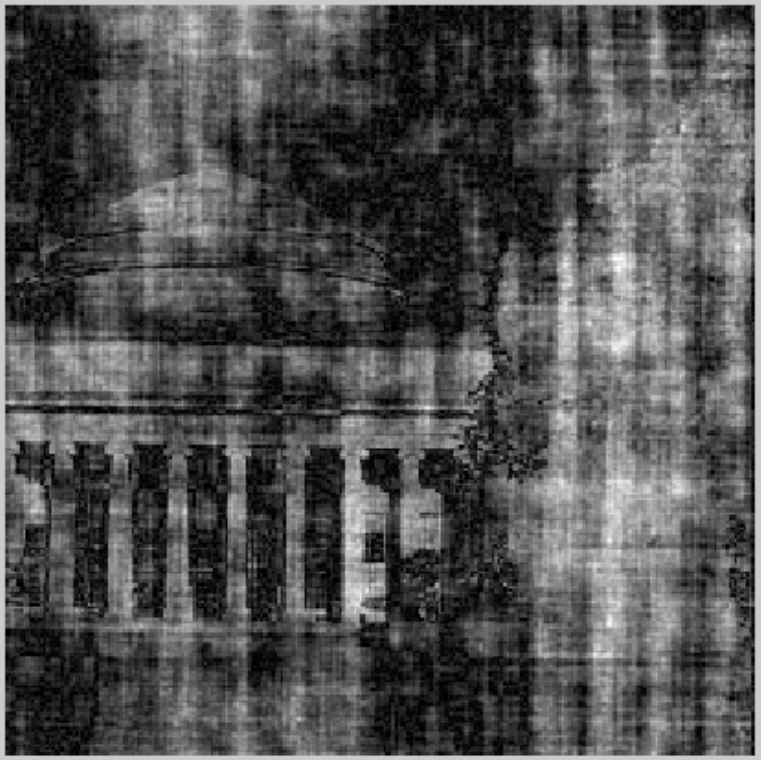
#2: Range [0.000556, 37.7]
Dims [256, 256]

28743

28743



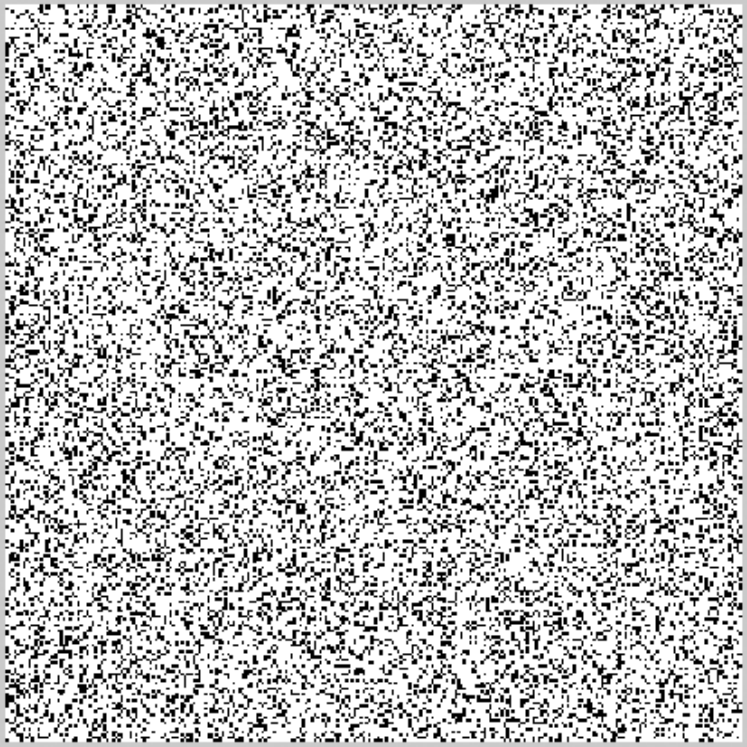
#1: Range [0, 1]
Dims [256, 256]



#2: Range [0.00109, 146]
Dims [256, 256]

49190.

49190



#1: Range [0, 1]
Dims [256, 256]

#2: Range [0.00758, 294]
Dims [256, 256]

65536.

65536.

#1: Range [0.5, 1.5]
Dims [256, 256]

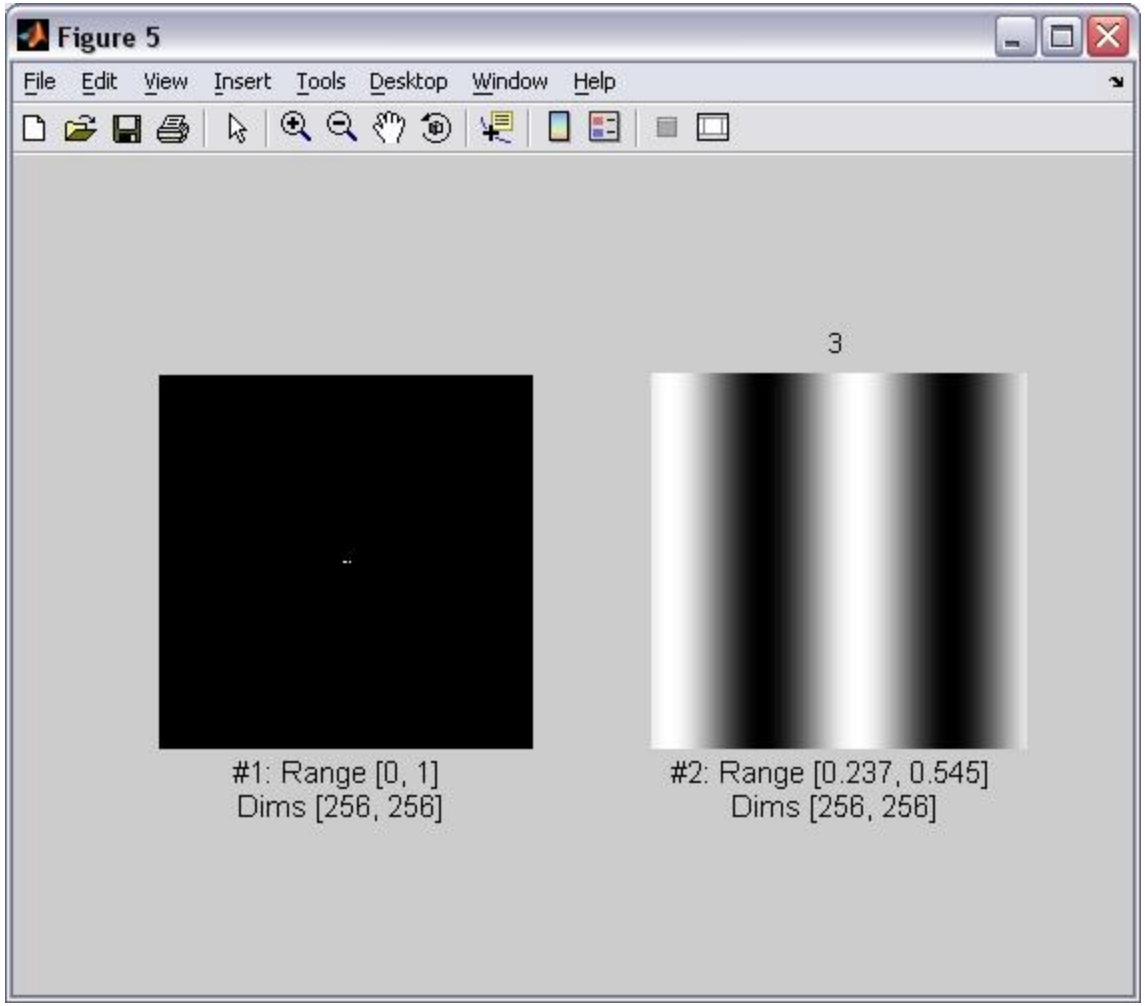
#2: Range [4.43e-015, 255]
Dims [256, 256]

Two examples of image synthesis with Fourier basis

First: randomly sample the Fourier coefficients of an image and reconstruct from those.

Second: sample Fourier coefficients in descending order of amplitude.

3



Now, an analogous sequence of images, but selecting Fourier components in descending order of magnitude.

5

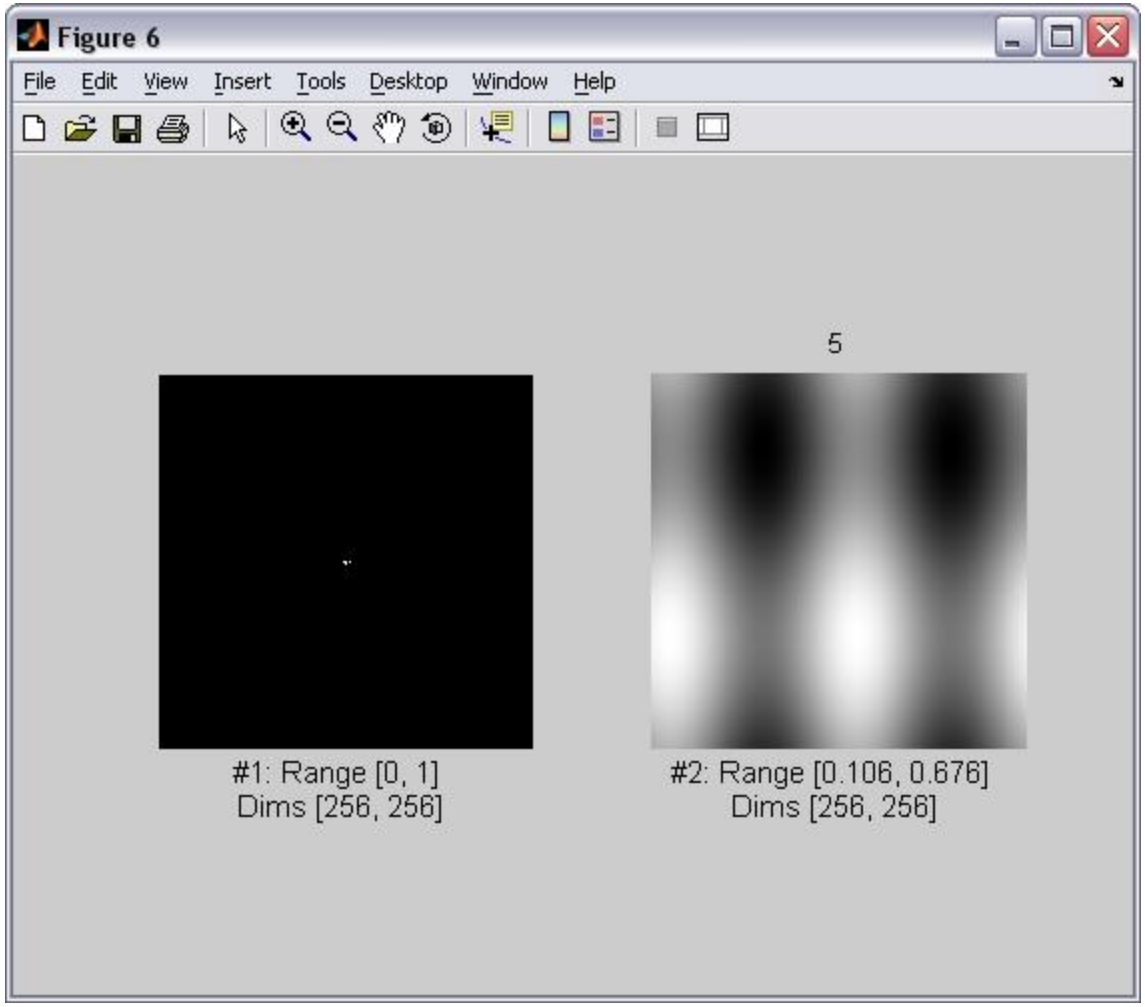


Figure 6
File Edit View Insert Tools Desktop Window Help
5
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.106, 0.676]
Dims [256, 256]

9

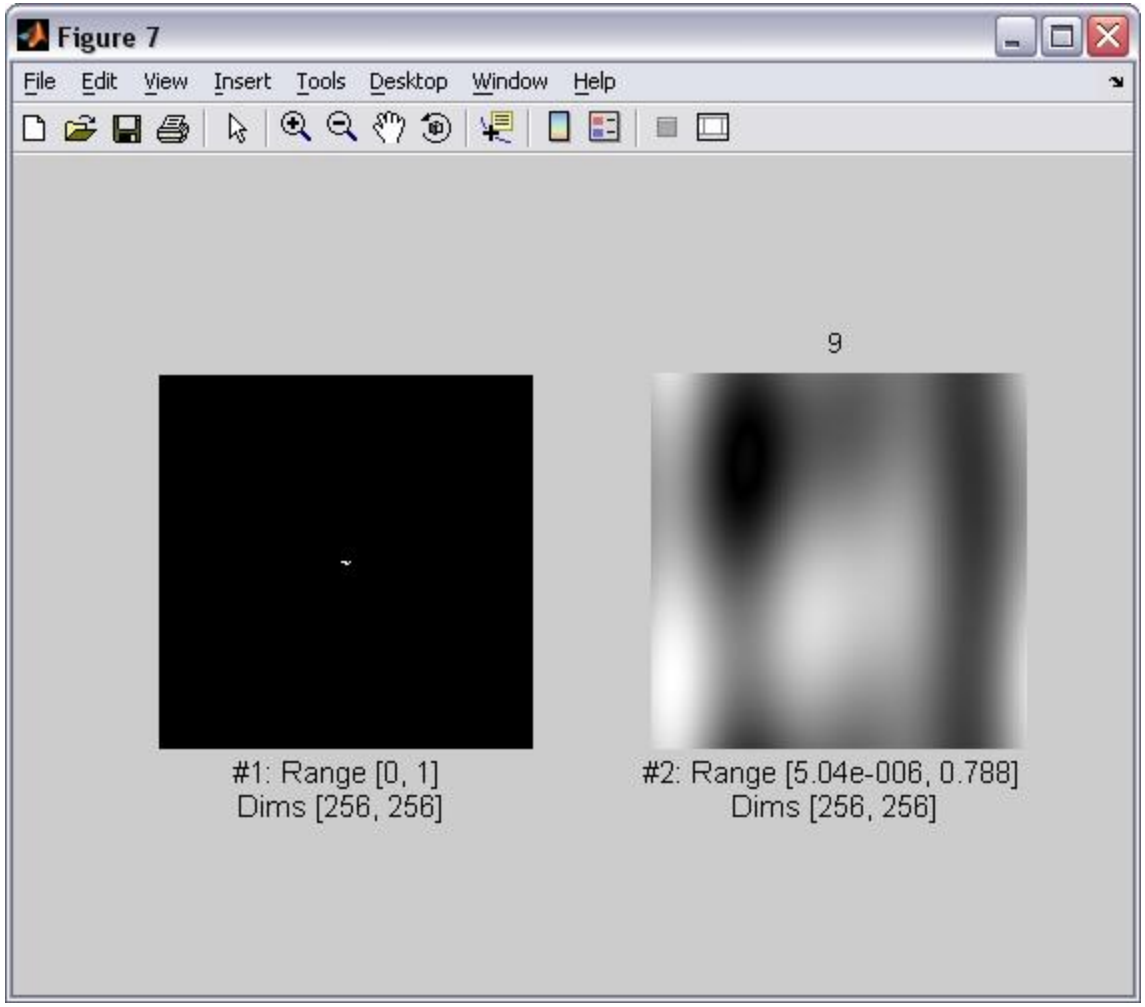


Figure 7
File Edit View Insert Tools Desktop Window Help
9
#1: Range [0, 1]
Dims [256, 256]
#2: Range [5.04e-006, 0.788]
Dims [256, 256]

17

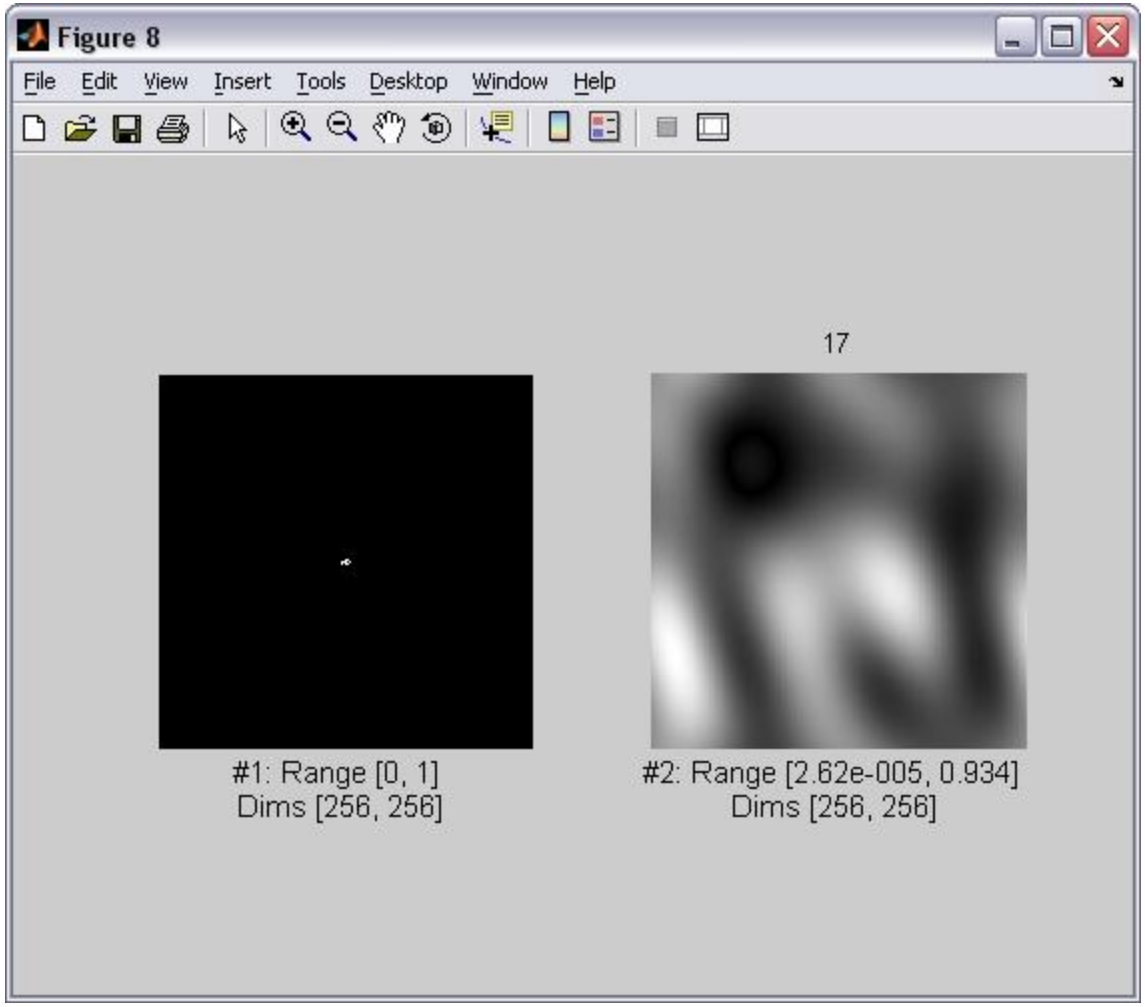


Figure 8
File Edit View Insert Tools Desktop Window Help
17
#1: Range [0, 1]
Dims [256, 256]
#2: Range [2.62e-005, 0.934]
Dims [256, 256]

33

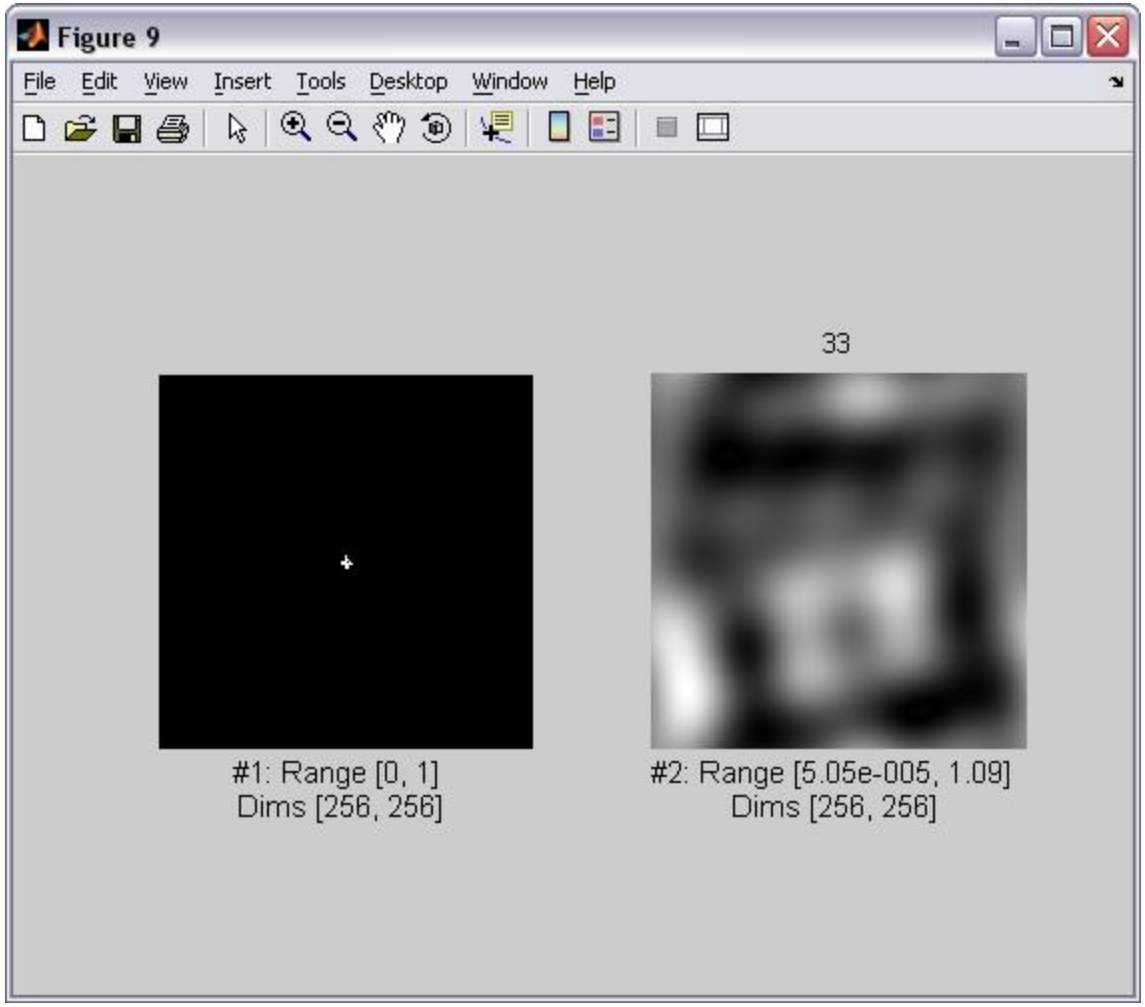


Figure 9
File Edit View Insert Tools Desktop Window Help
33
#1: Range [0, 1]
Dims [256, 256]
#2: Range [5.05e-005, 1.09]
Dims [256, 256]

65

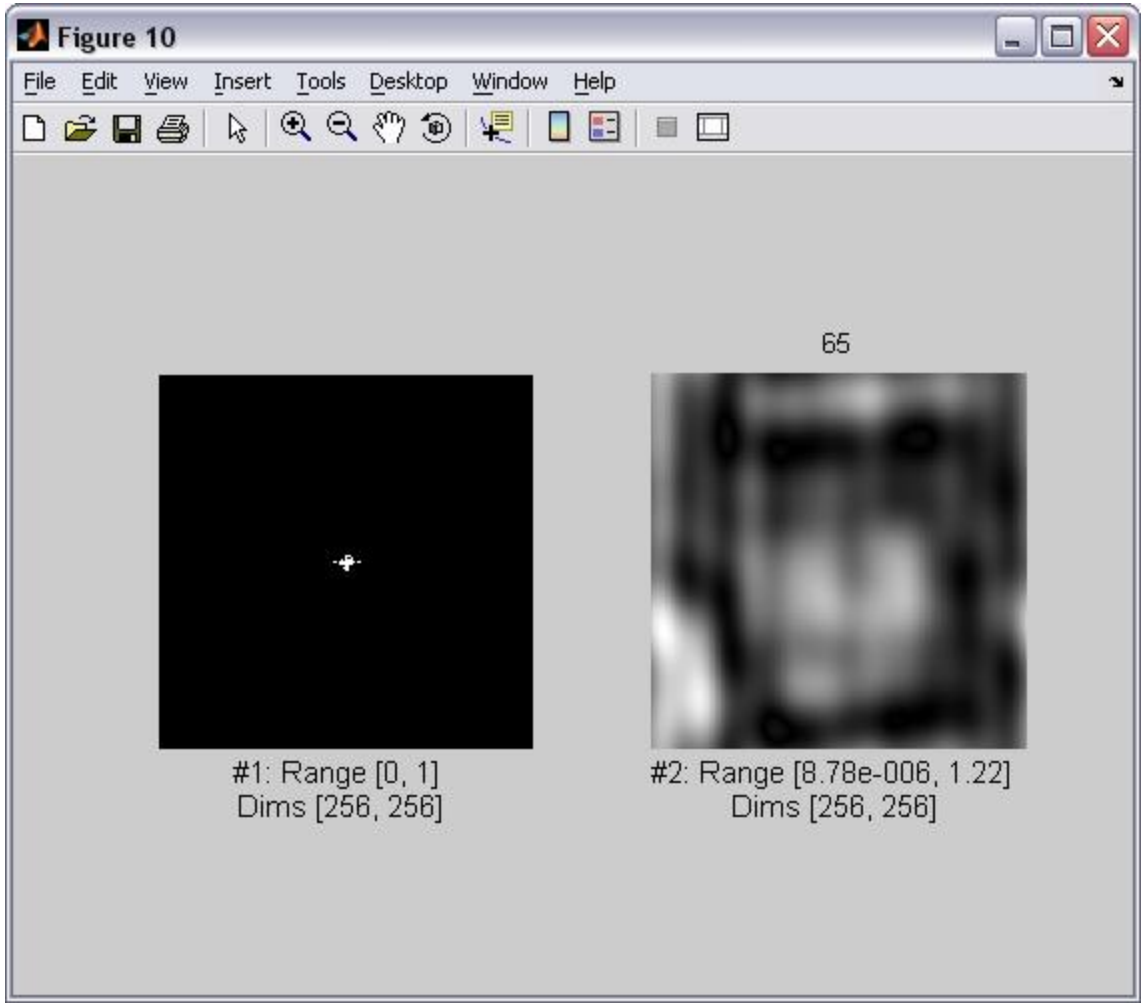


Figure 10
File Edit View Insert Tools Desktop Window Help
65
#1: Range [0, 1]
Dims [256, 256]
#2: Range [8.78e-006, 1.22]
Dims [256, 256]

129

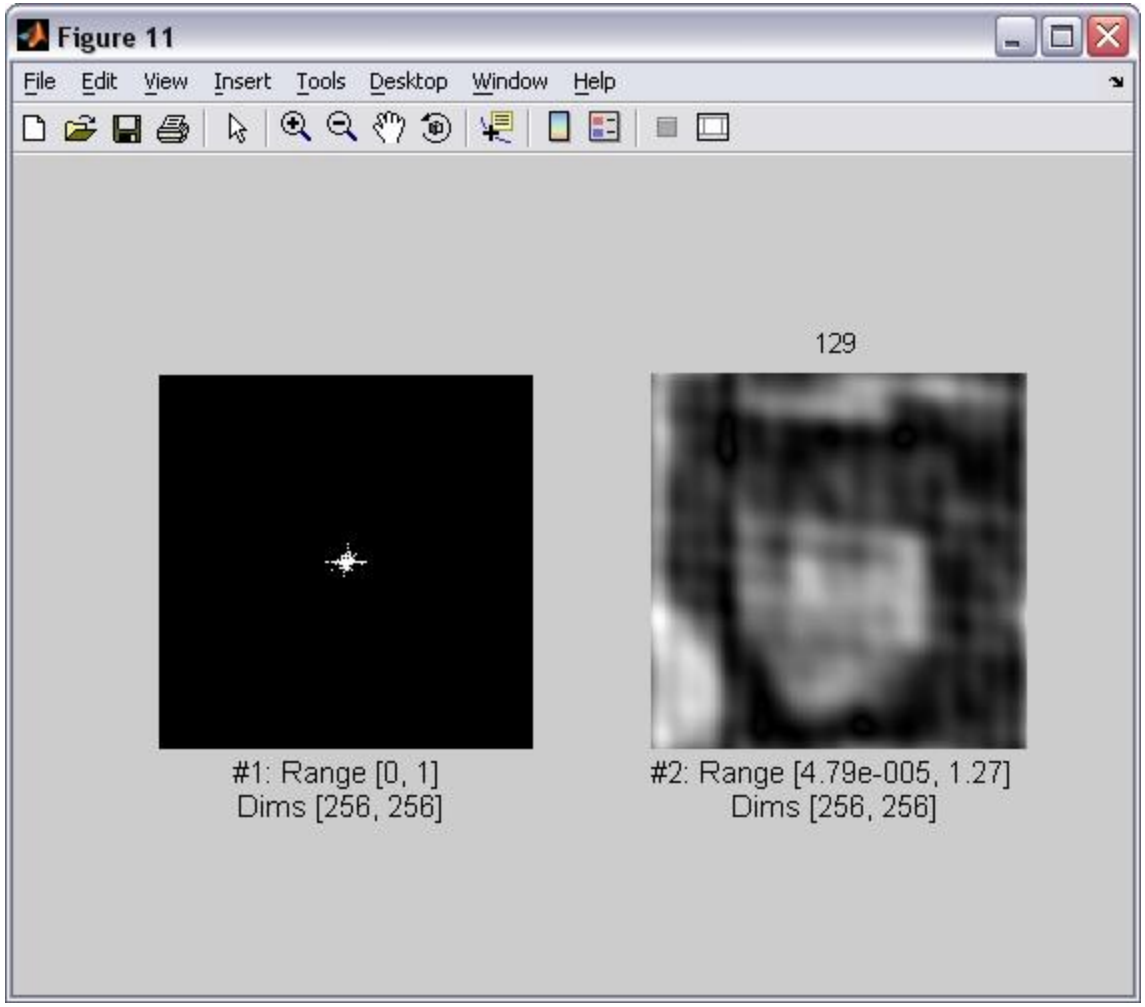


Figure 11
File Edit View Insert Tools Desktop Window Help
129
#1: Range [0, 1]
Dims [256, 256]
#2: Range [4.79e-005, 1.27]
Dims [256, 256]

257

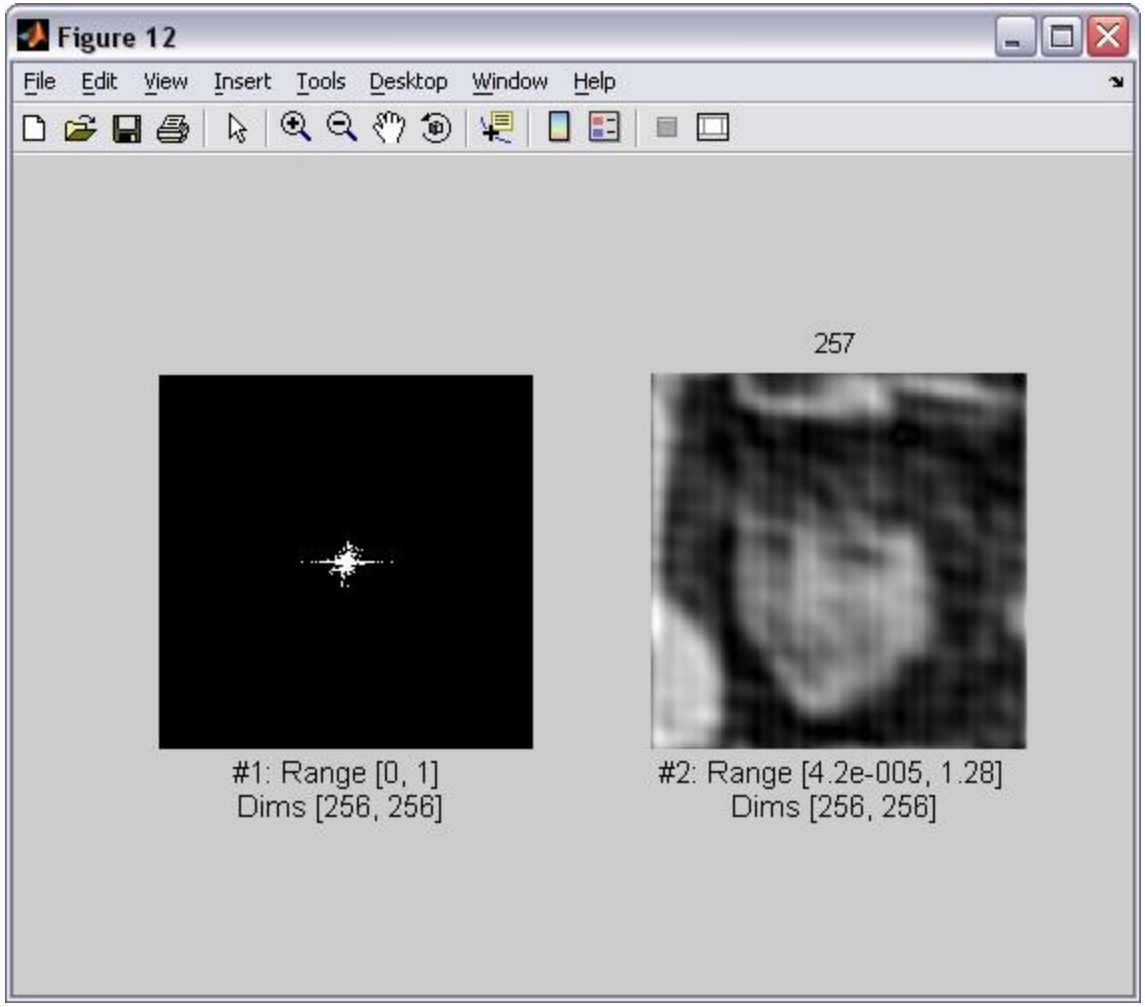


Figure 12
File Edit View Insert Tools Desktop Window Help
257
#1: Range [0, 1]
Dims [256, 256]
#2: Range [4.2e-005, 1.28]
Dims [256, 256]

513

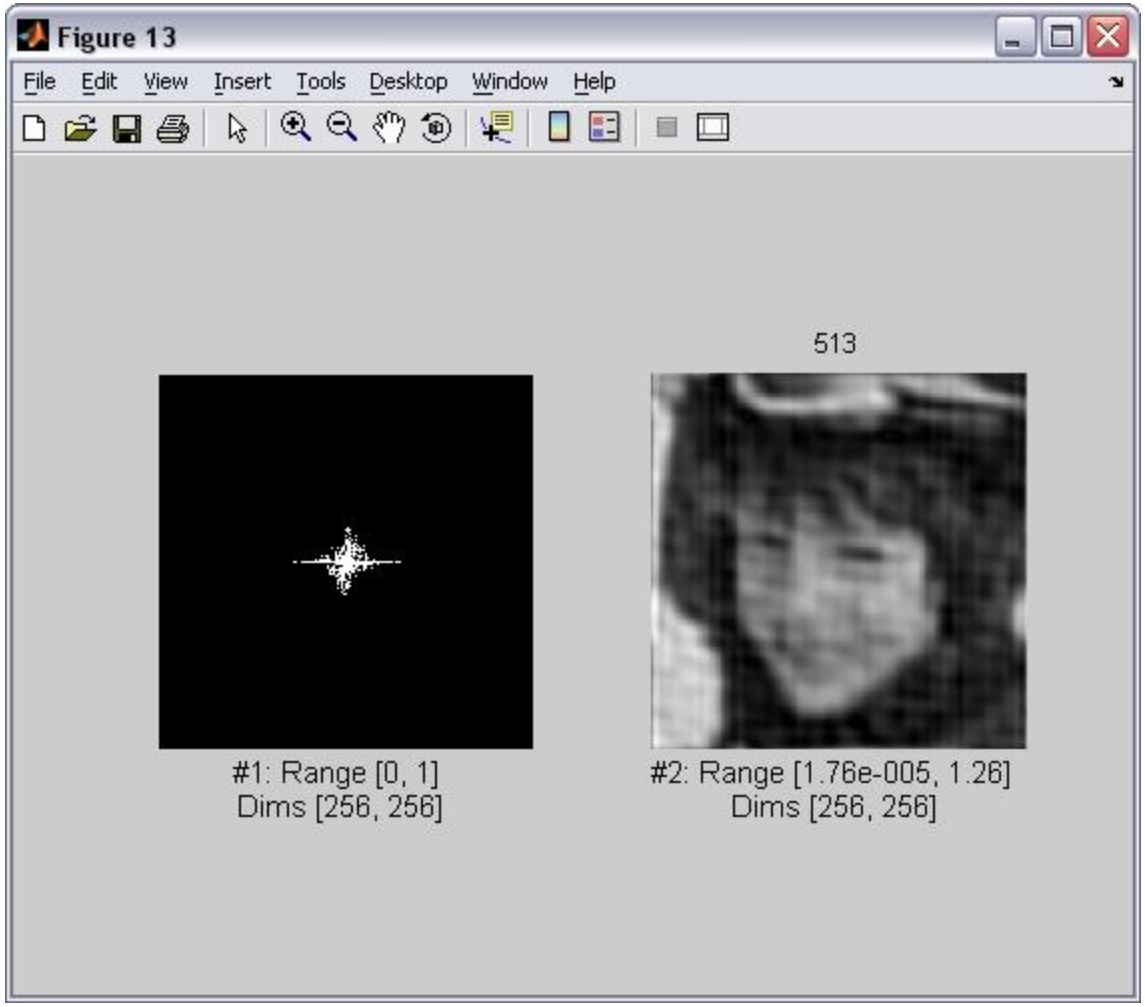


Figure 13
File Edit View Insert Tools Desktop Window Help
513
#1: Range [0, 1]
Dims [256, 256]
#2: Range [1.76e-005, 1.26]
Dims [256, 256]

1025

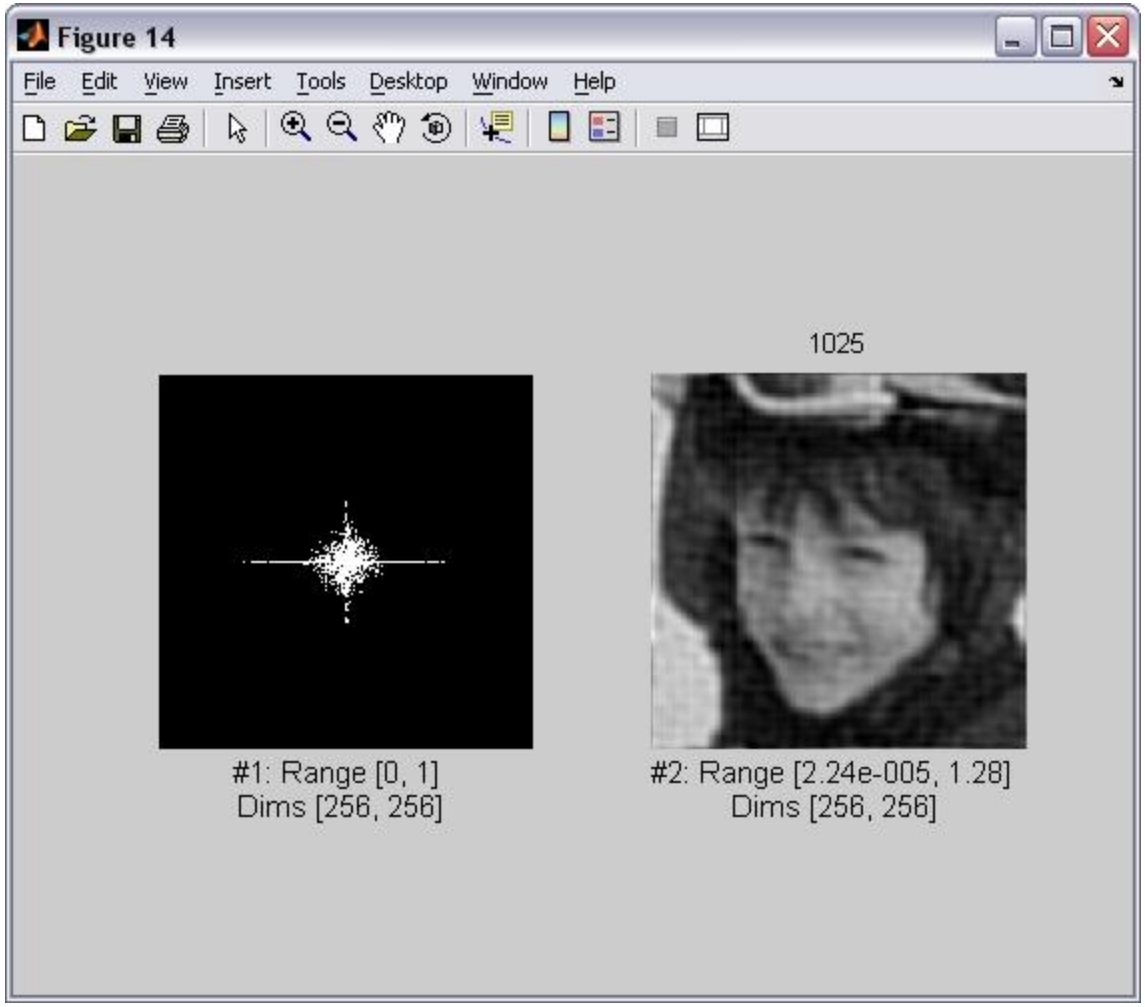


Figure 14
File Edit View Insert Tools Desktop Window Help
1025
#1: Range [0, 1]
Dims [256, 256]
#2: Range [2.24e-005, 1.28]
Dims [256, 256]

2049

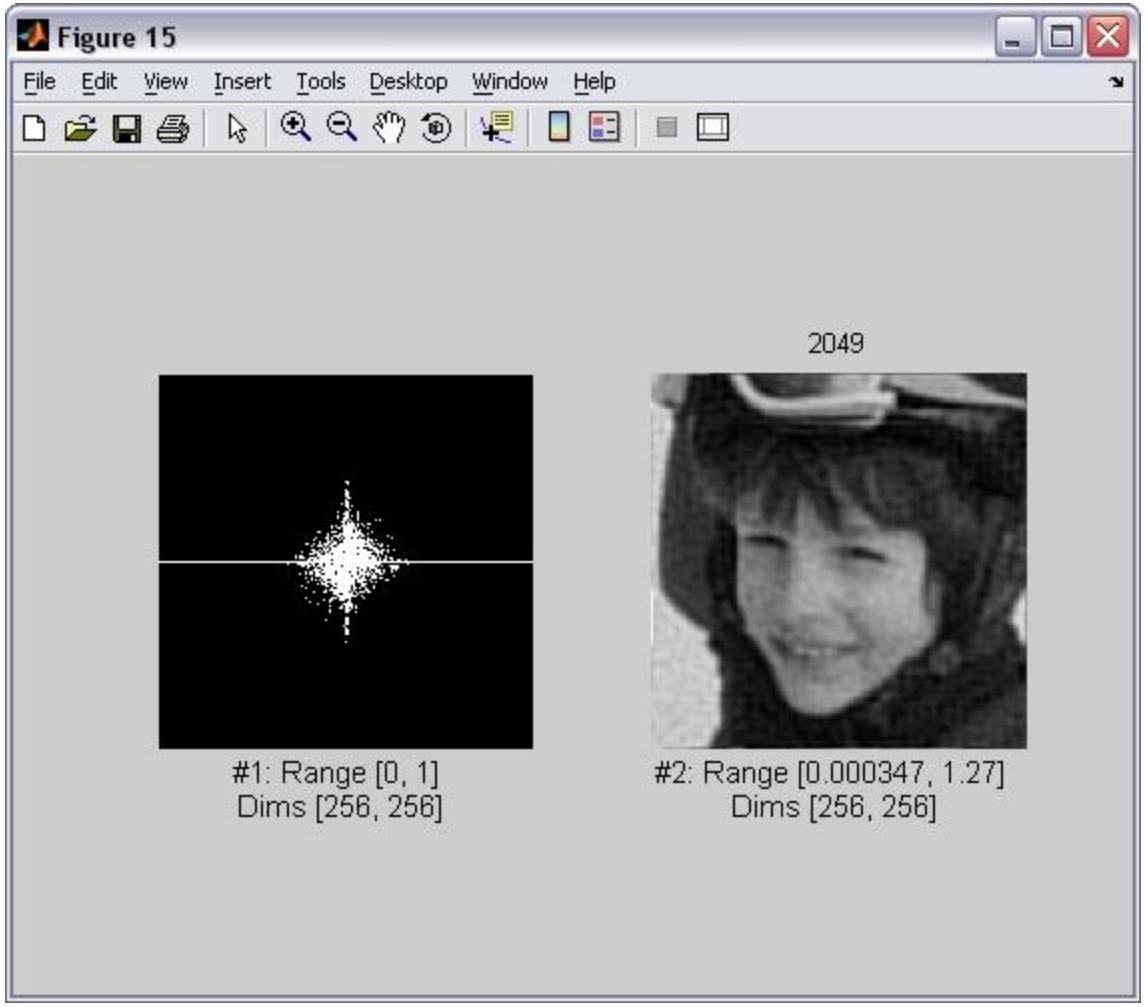


Figure 15
File Edit View Insert Tools Desktop Window Help
2049
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.000347, 1.27]
Dims [256, 256]

4097

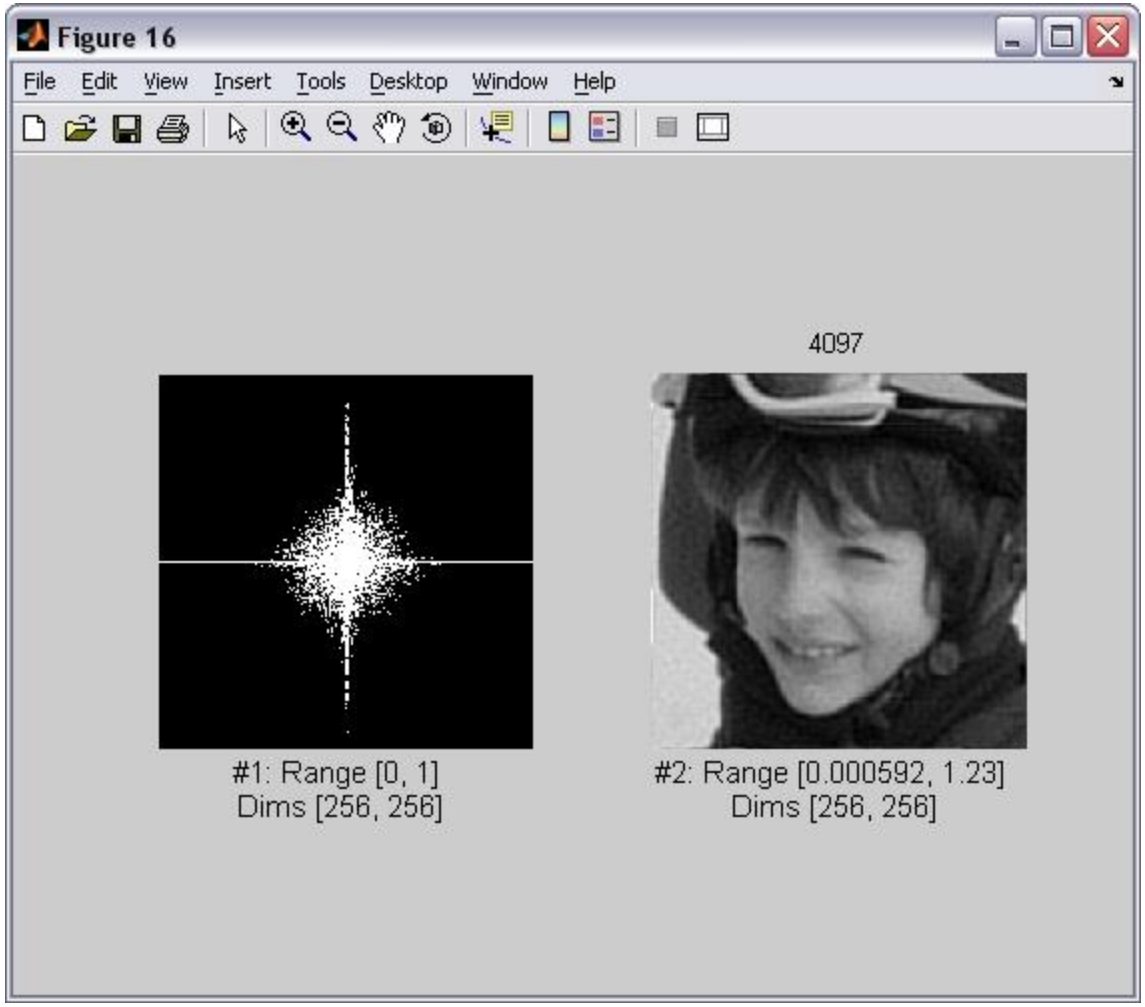


Figure 16
File Edit View Insert Tools Desktop Window Help
4097
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.000592, 1.23]
Dims [256, 256]

8193

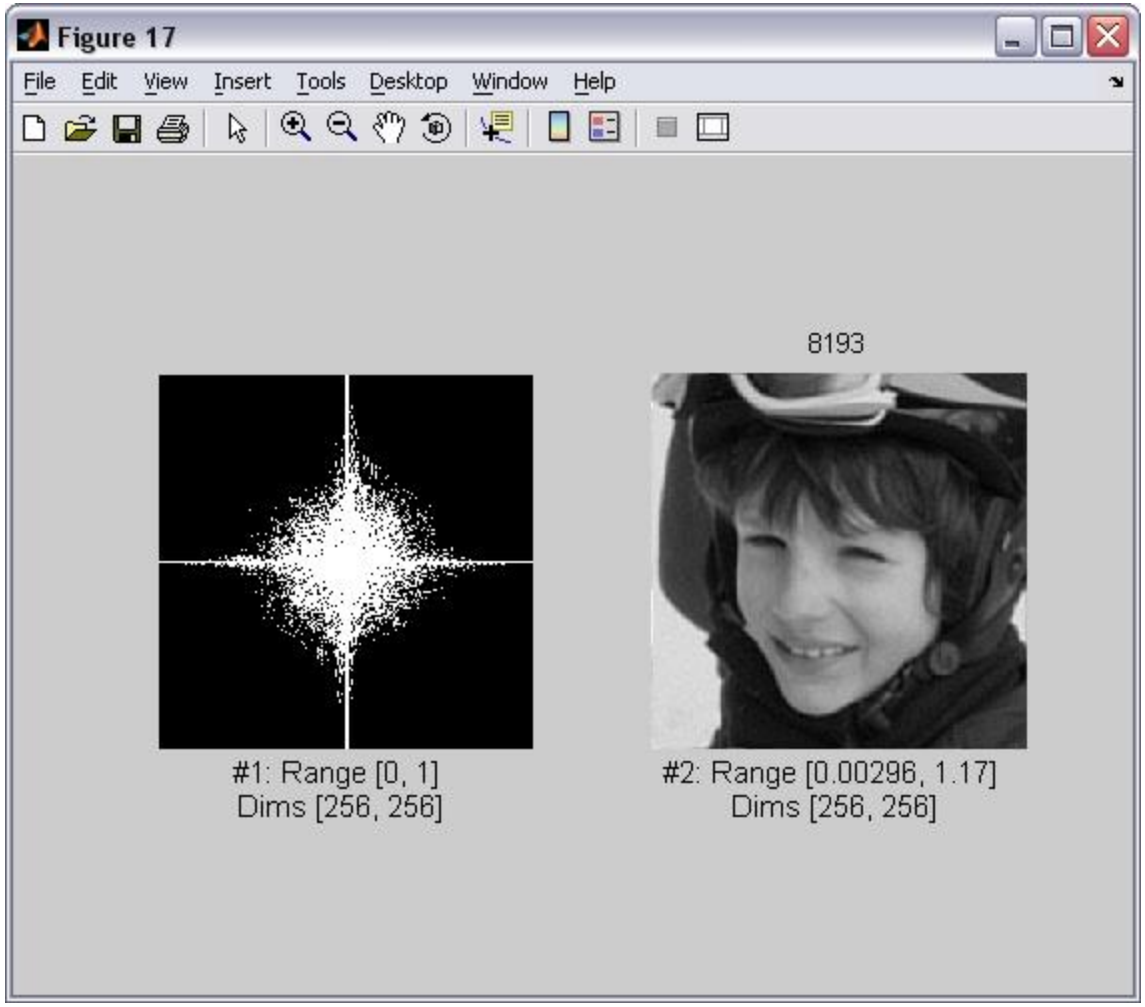


Figure 17
File Edit View Insert Tools Desktop Window Help
8193
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.00296, 1.17]
Dims [256, 256]

16385

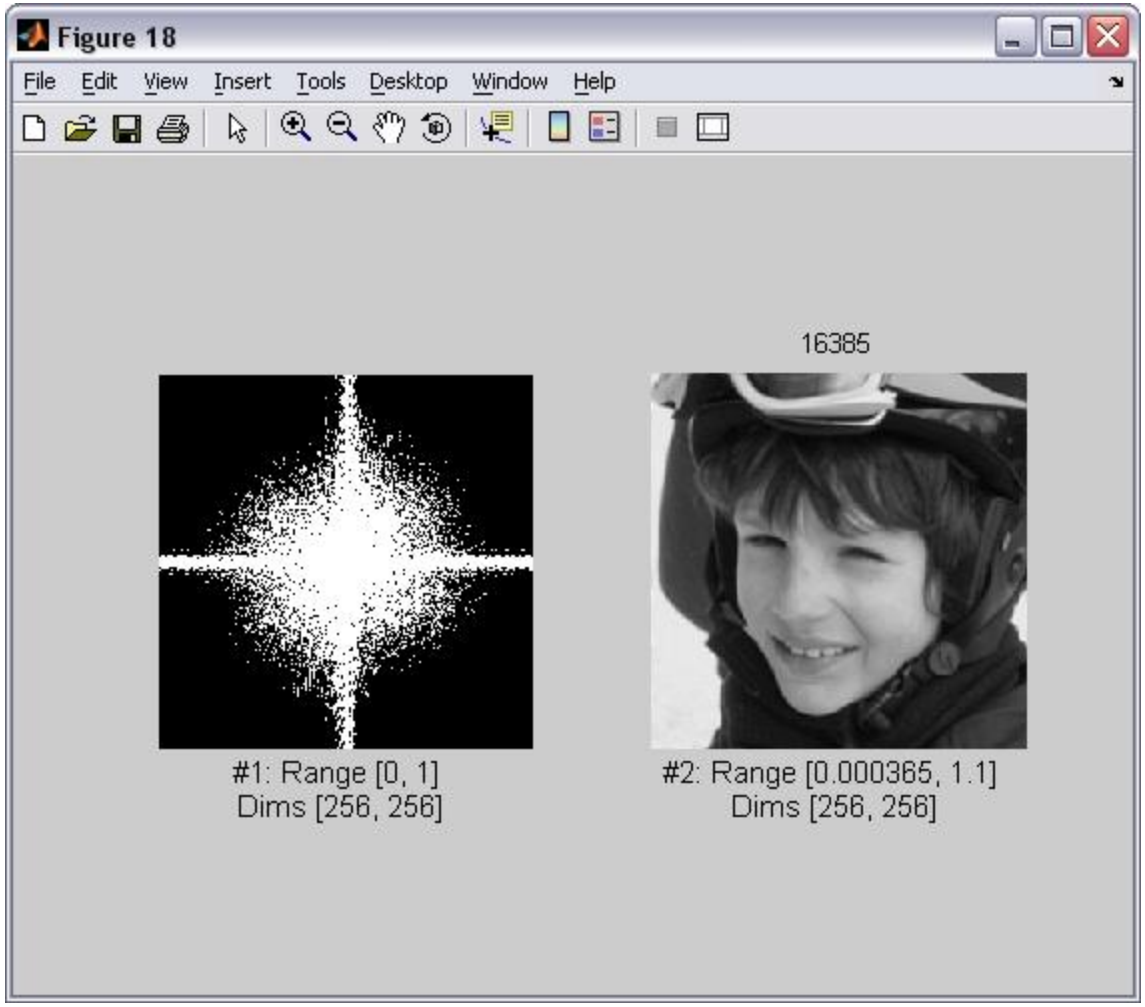


Figure 18
File Edit View Insert Tools Desktop Window Help
16385
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.000365, 1.1]
Dims [256, 256]

32769

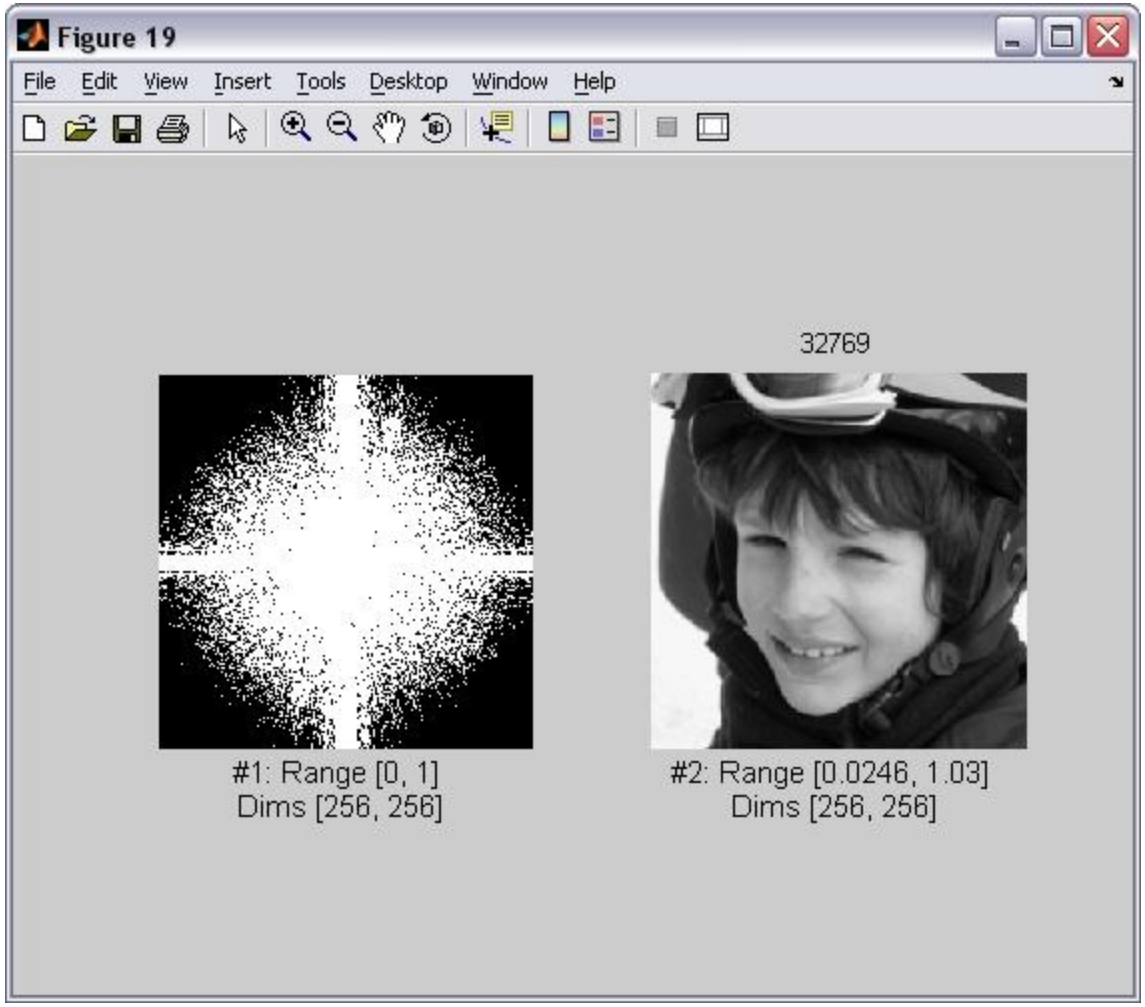


Figure 19
File Edit View Insert Tools Desktop Window Help
32769
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.0246, 1.03]
Dims [256, 256]

65536

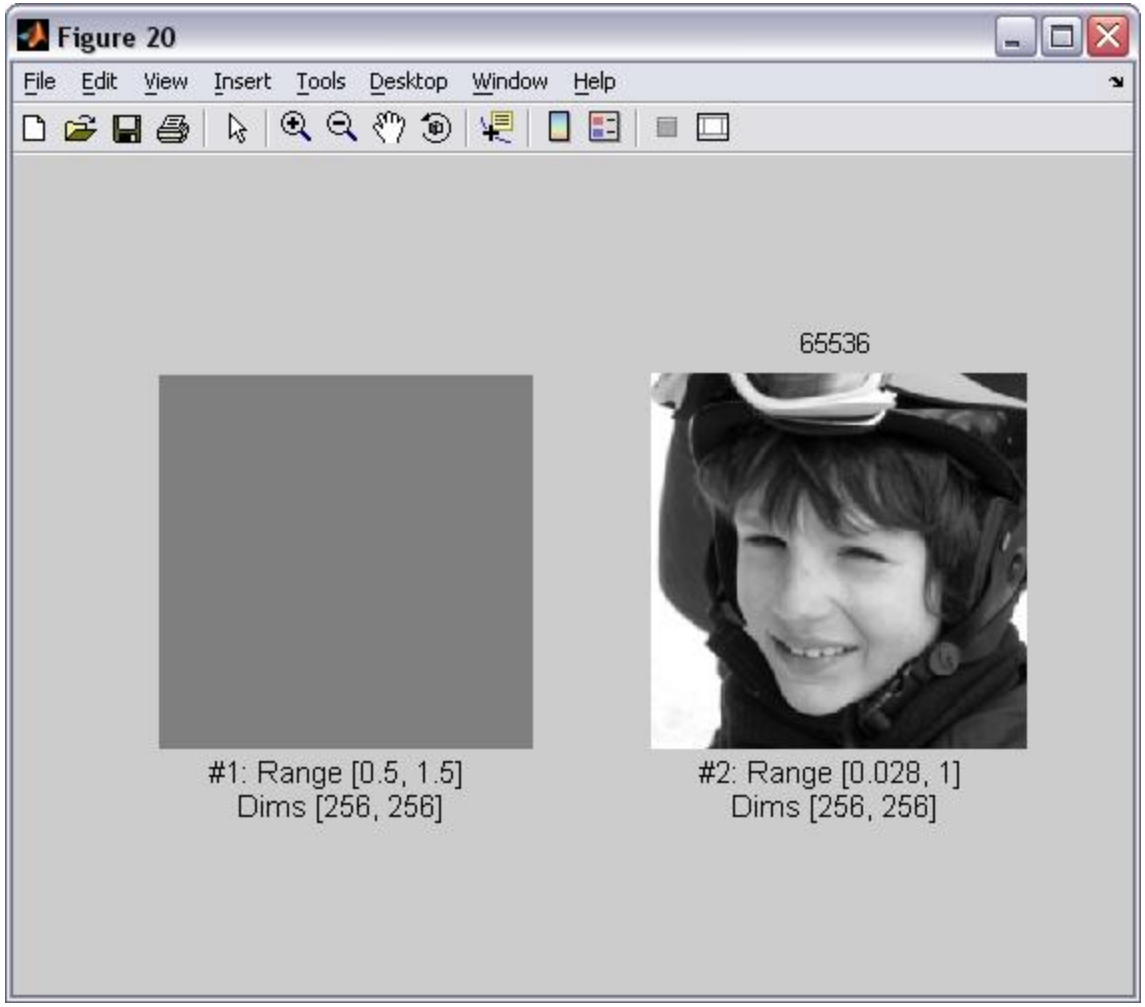
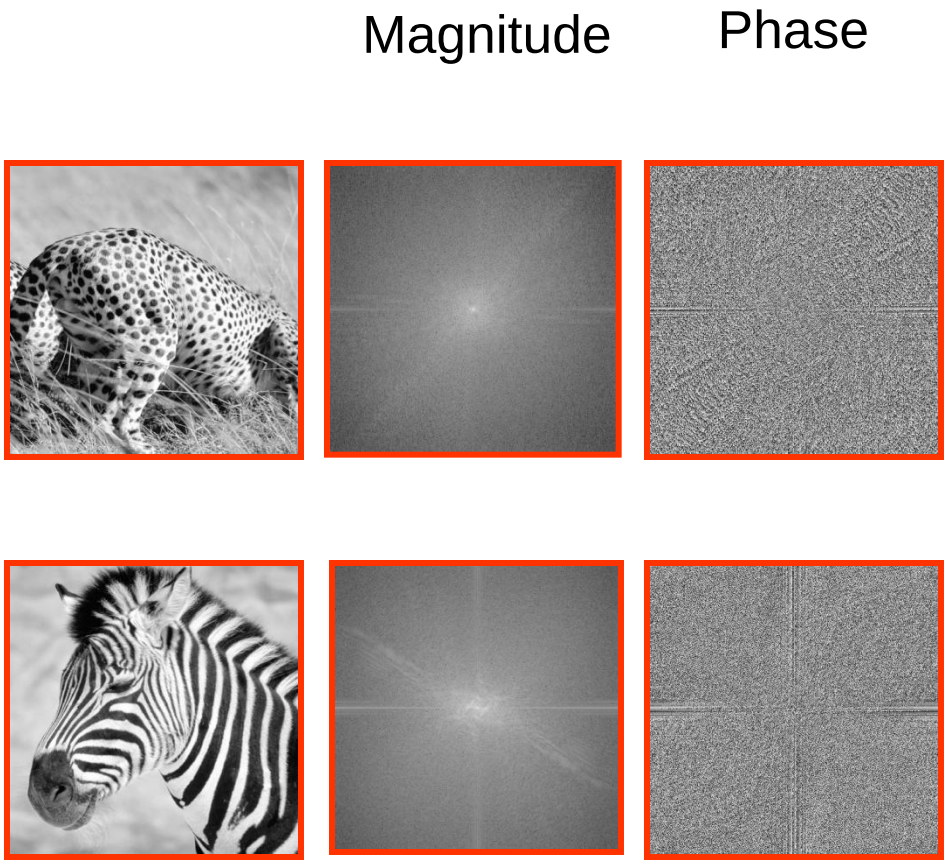


Figure 20
File Edit View Insert Tools Desktop Window Help
65536
#1: Range [0.5, 1.5]
Dims [256, 256]
#2: Range [0.028, 1]
Dims [256, 256]

Fourier Transform

- Fourier transform of a real function is complex
  - difficult to plot, visualize
  - instead, we can think of the phase and magnitude of the transform
- Phase is the phase of the complex transform
- Magnitude is the magnitude of the complex transform

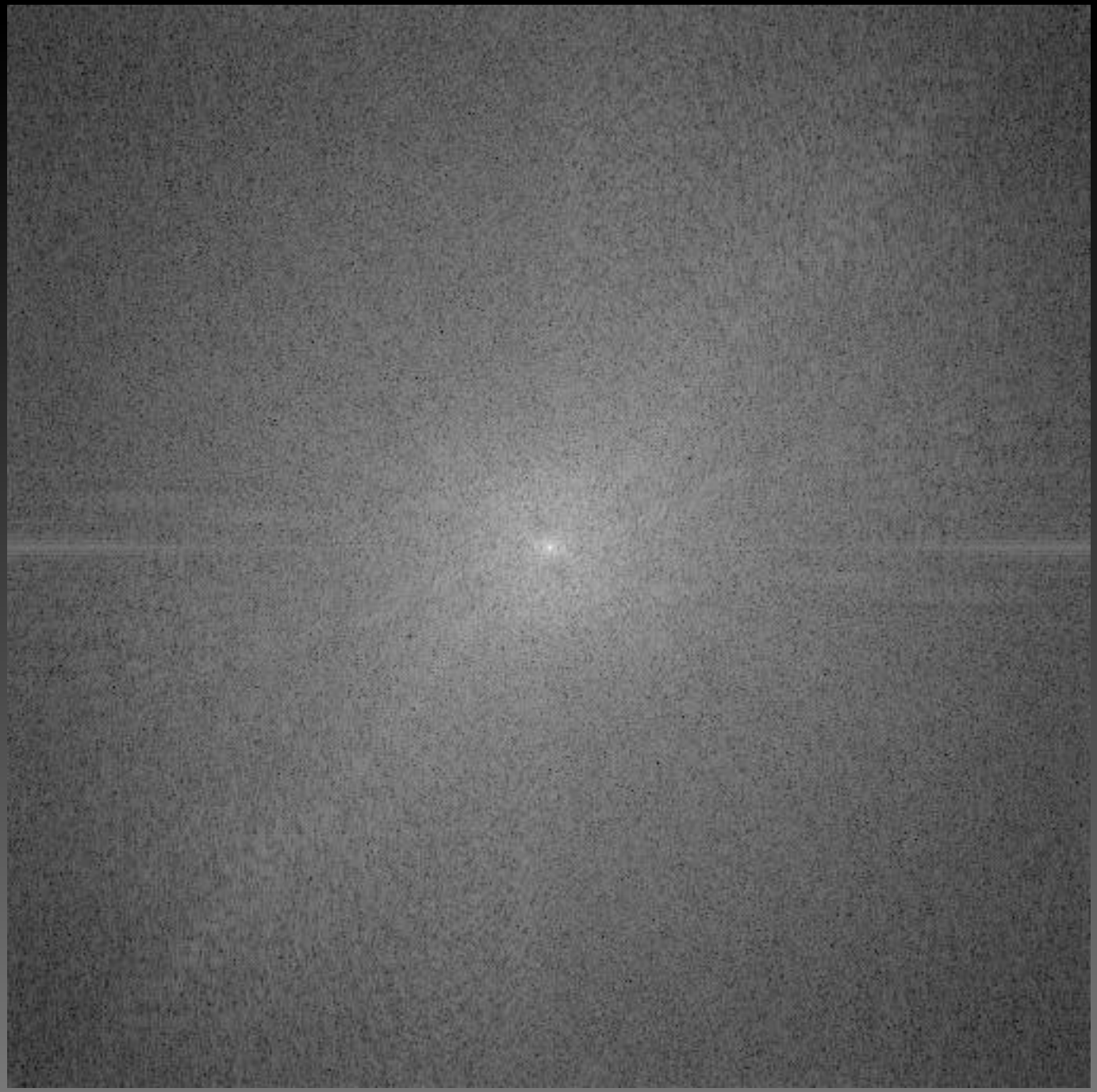


Phase and Magnitude

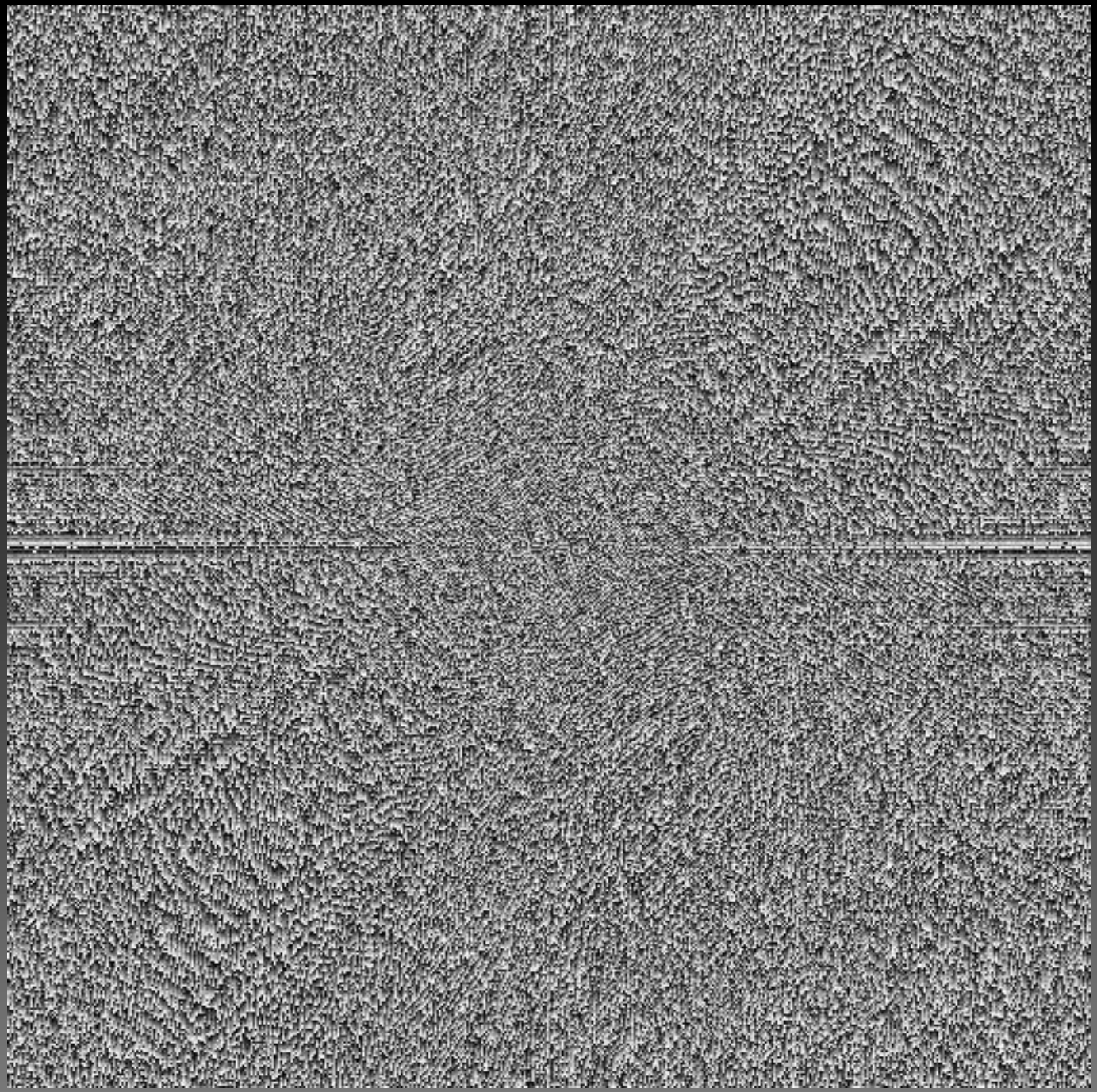
- **Curious fact**
  - **all natural images have about the same magnitude transform**
  - **hence, phase seems to matter, but magnitude largely doesn't**

- **Demonstration**
  - **Take two pictures, swap the phase transforms, compute the inverse - what does the result look like?**

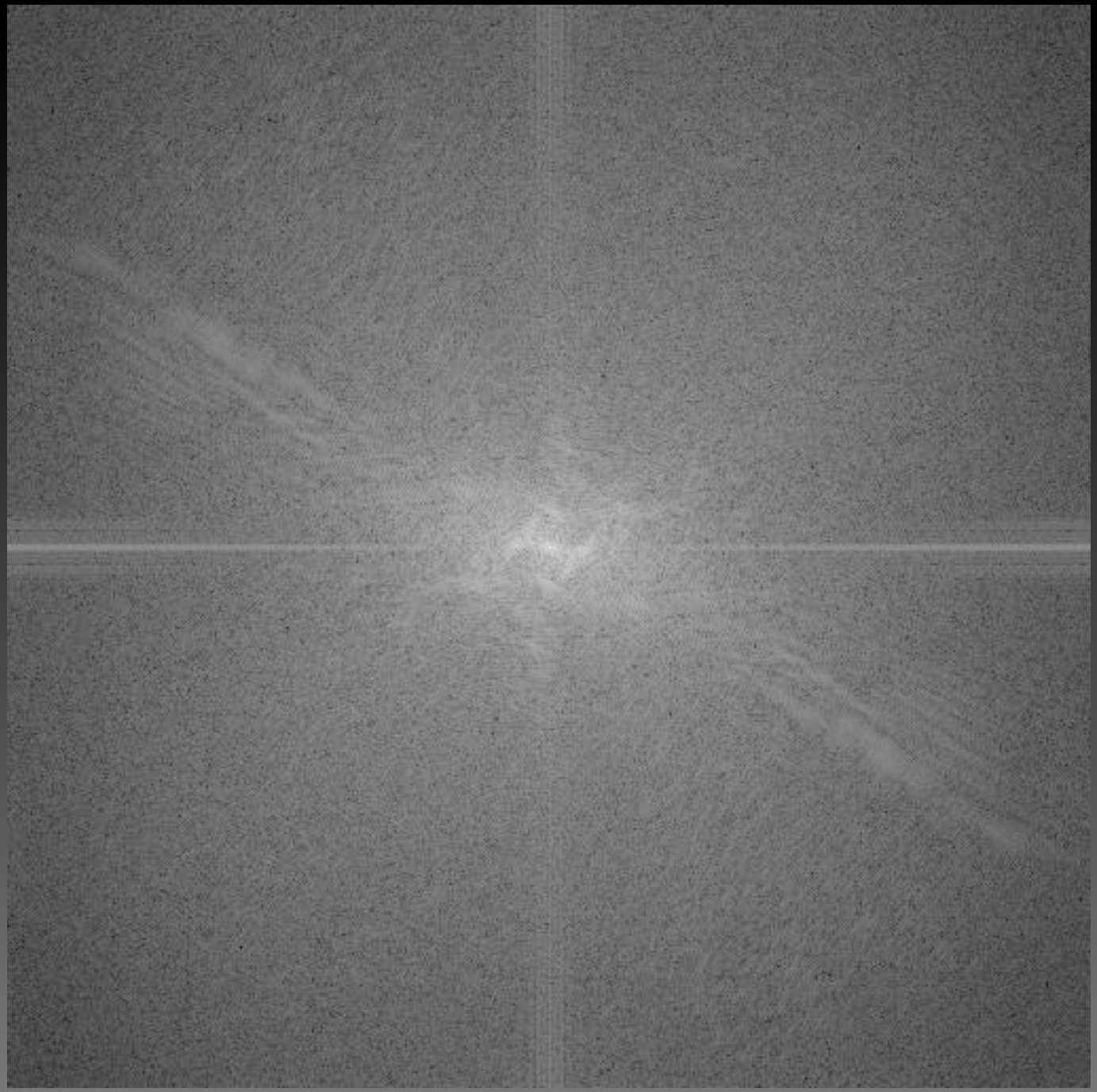
This is the magnitude transform of the cheetah pic



This is the phase transform of the cheetah pic

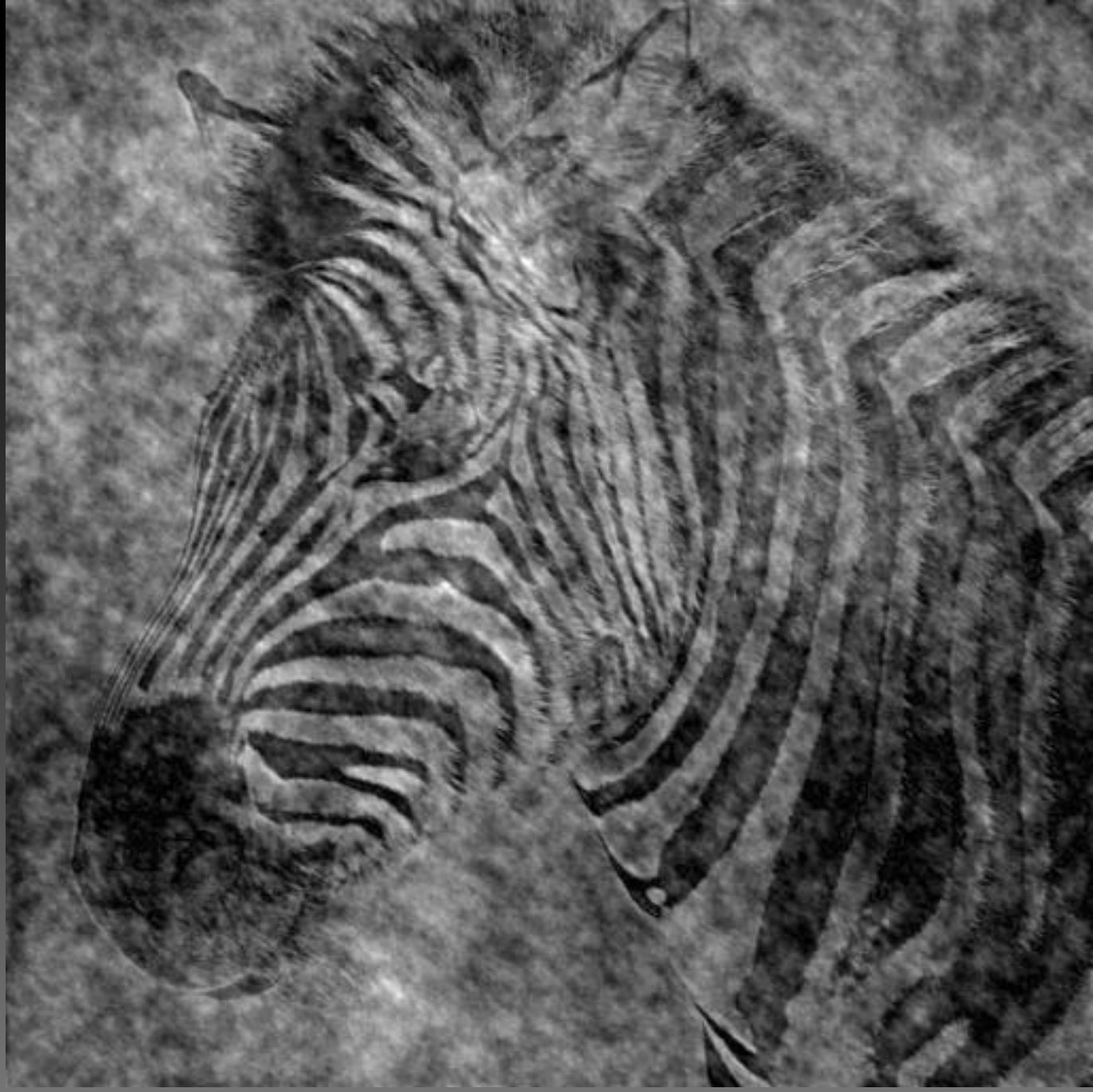


This is the magnitude transform of the zebra pic

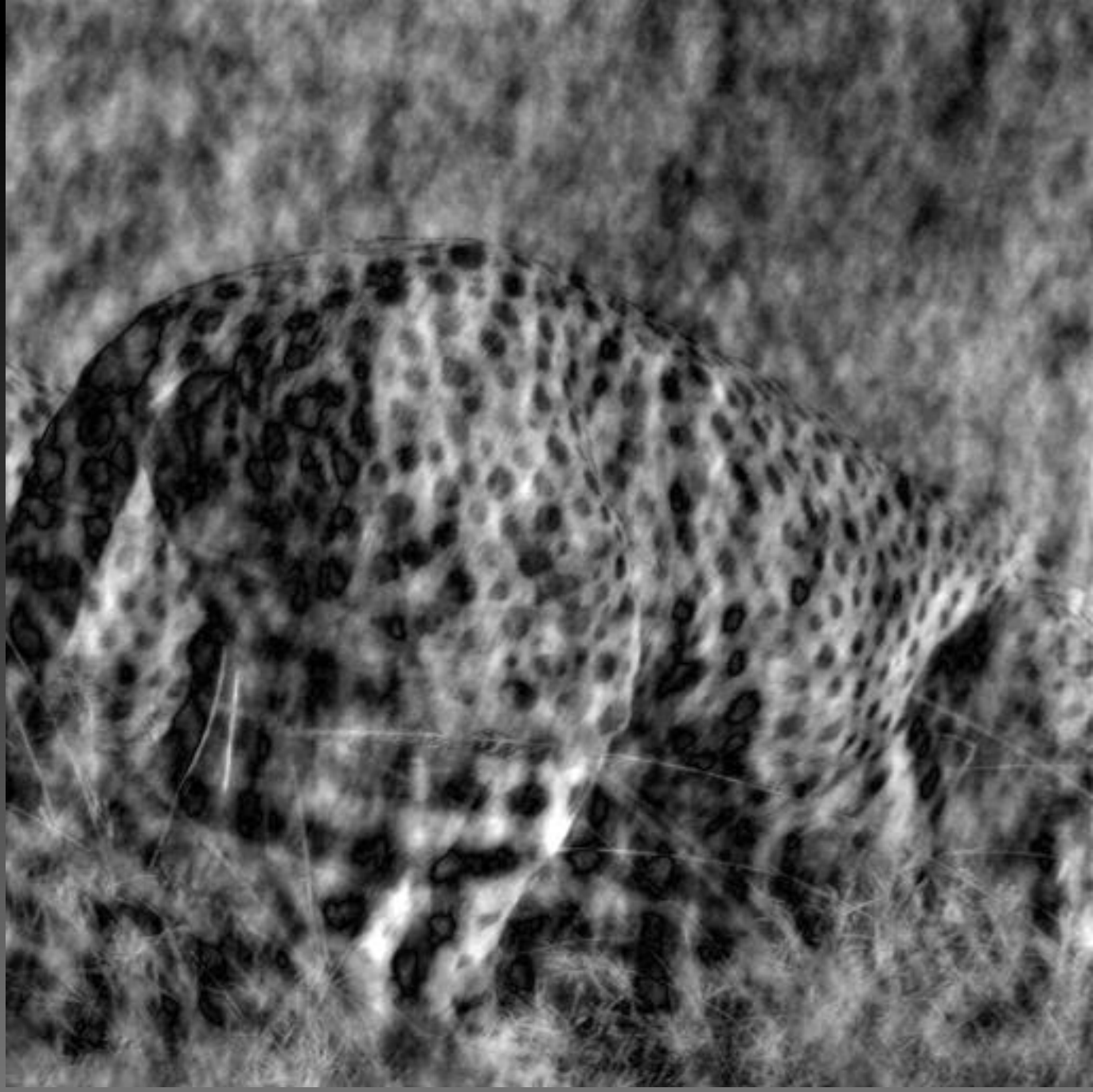


This is the phase transform of the zebra pic

Reconstruction with zebra phase, cheetah magnitude



Reconstruction with cheetah phase, zebra magnitude



Phase and Magnitude

- The magnitude of natural images can often be quite similar, one to another. But magnitude **encodes statistics of orientation at all spatial scales**.
- The phase carry the information of **where the image contours are**, by specifying how the phases of the sinusoids must line up in order to create the observed contours and edges.

Image with cheetah phase (and zebra magnitude)

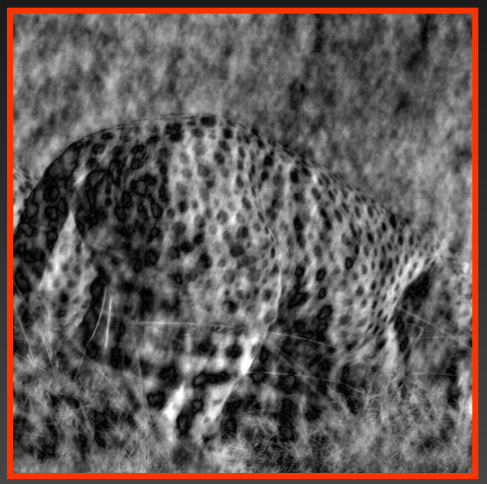
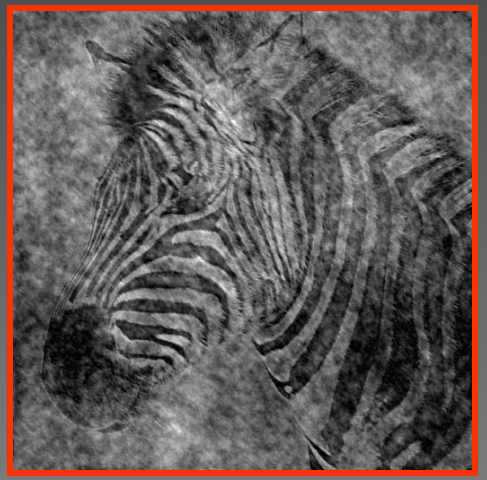
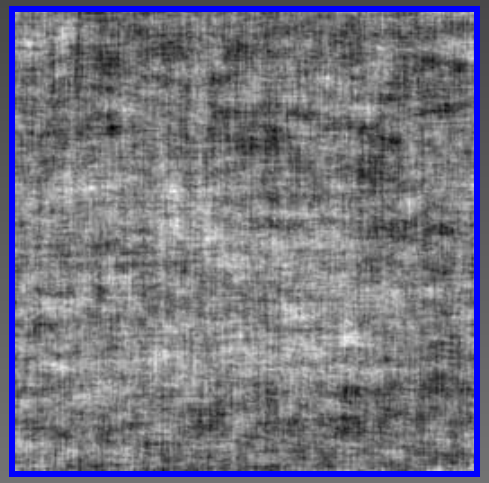
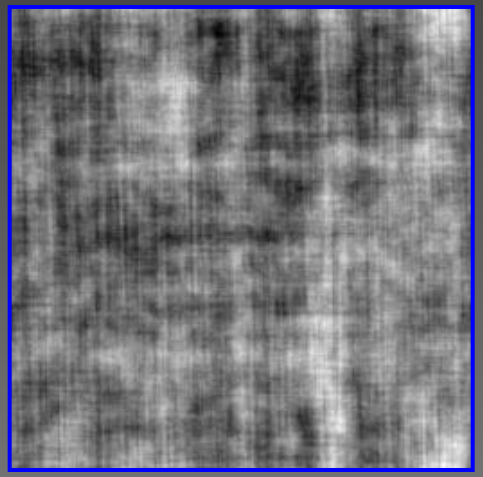
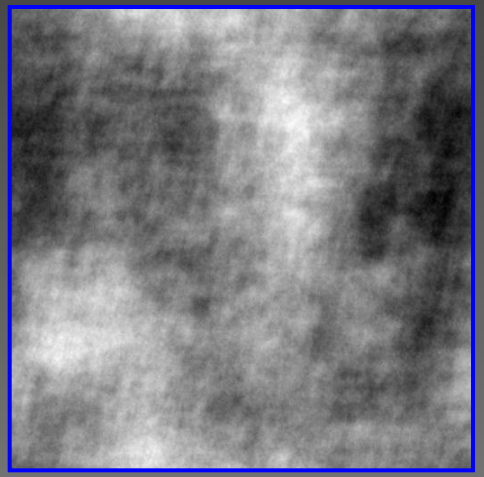


Image with zebra phase (and cheetah magnitude)



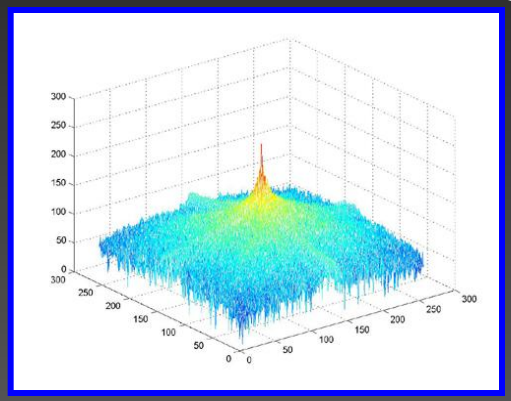
Randomizing the phase



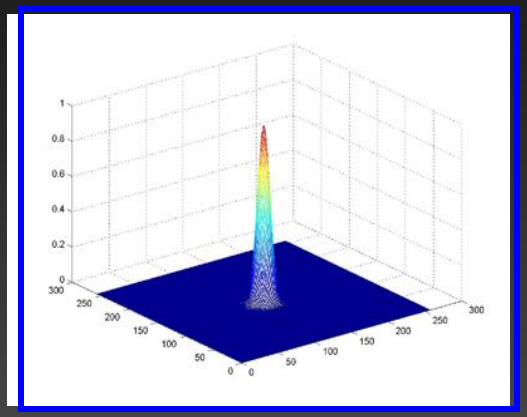
Use of Fourier Spectrum : Filtering

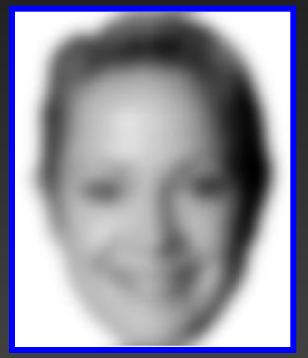
Fourier space



*

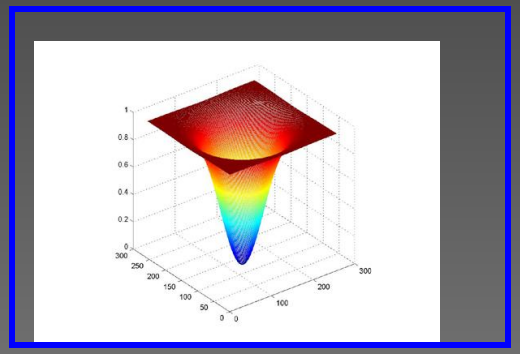
Low pass Filter



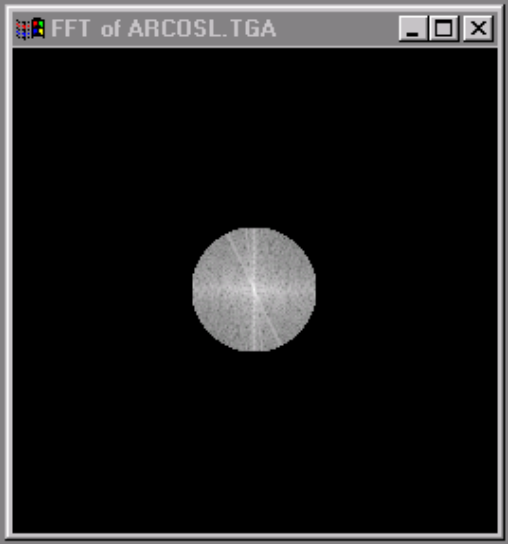


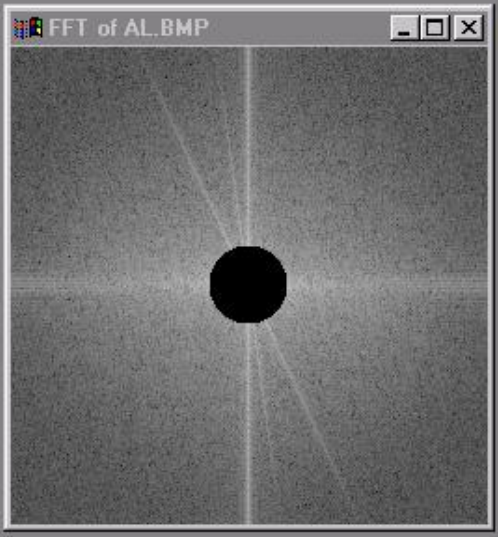
High pass filter

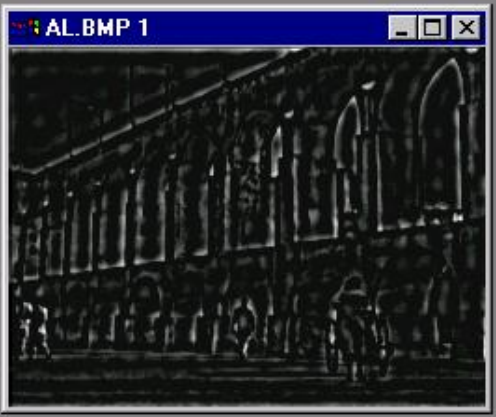
*



Low and High Pass filtering





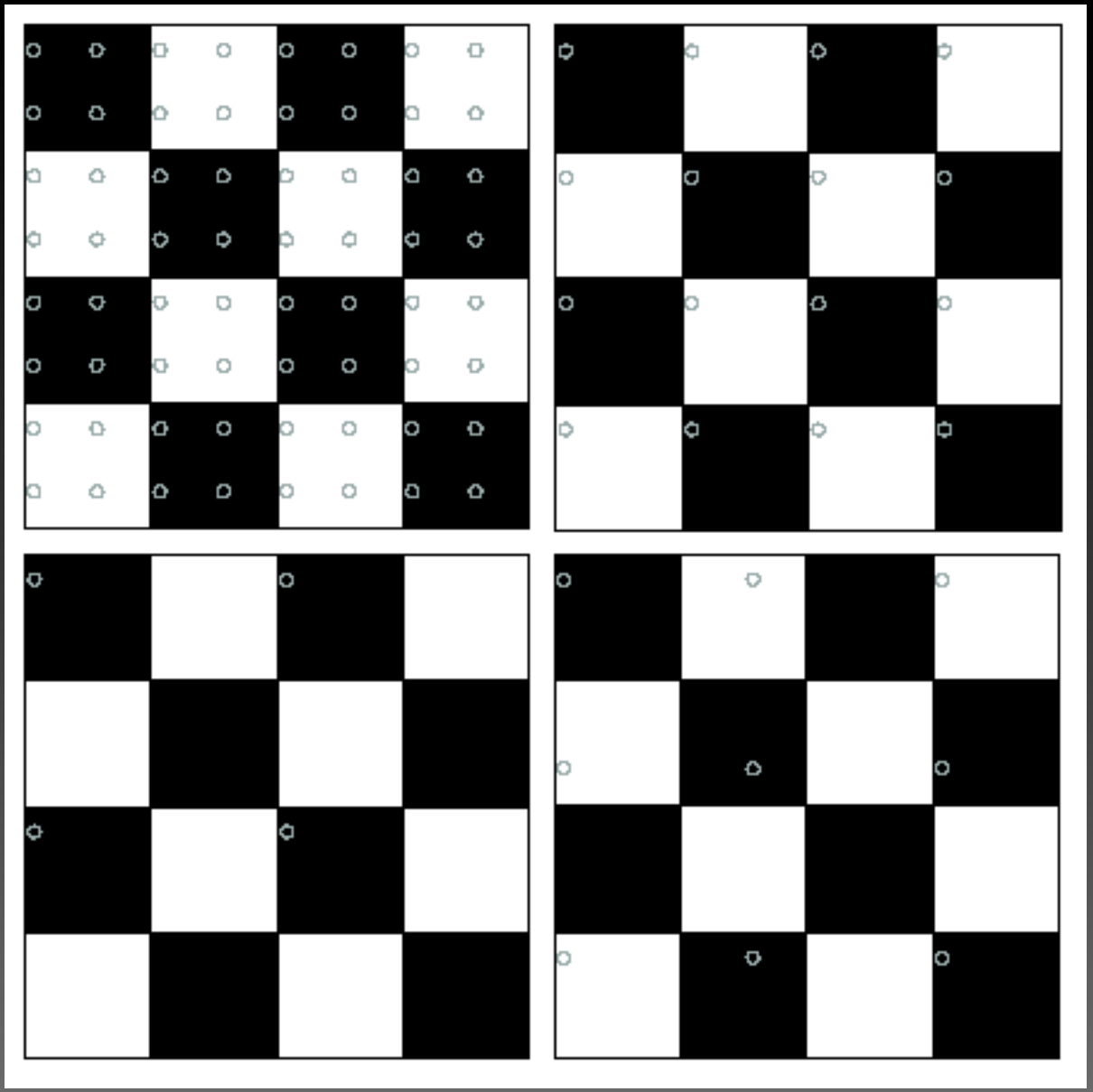


The Fourier transform of a sampled signal

$$F\left(\text{Sample}_{2D}\left(f(x,y)\right)\right) = F\left(f(x,y)\sum_{i=-\infty}^{\infty}\sum_{i=-\infty}^{\infty}\delta(x-i,y-j)\right)$$

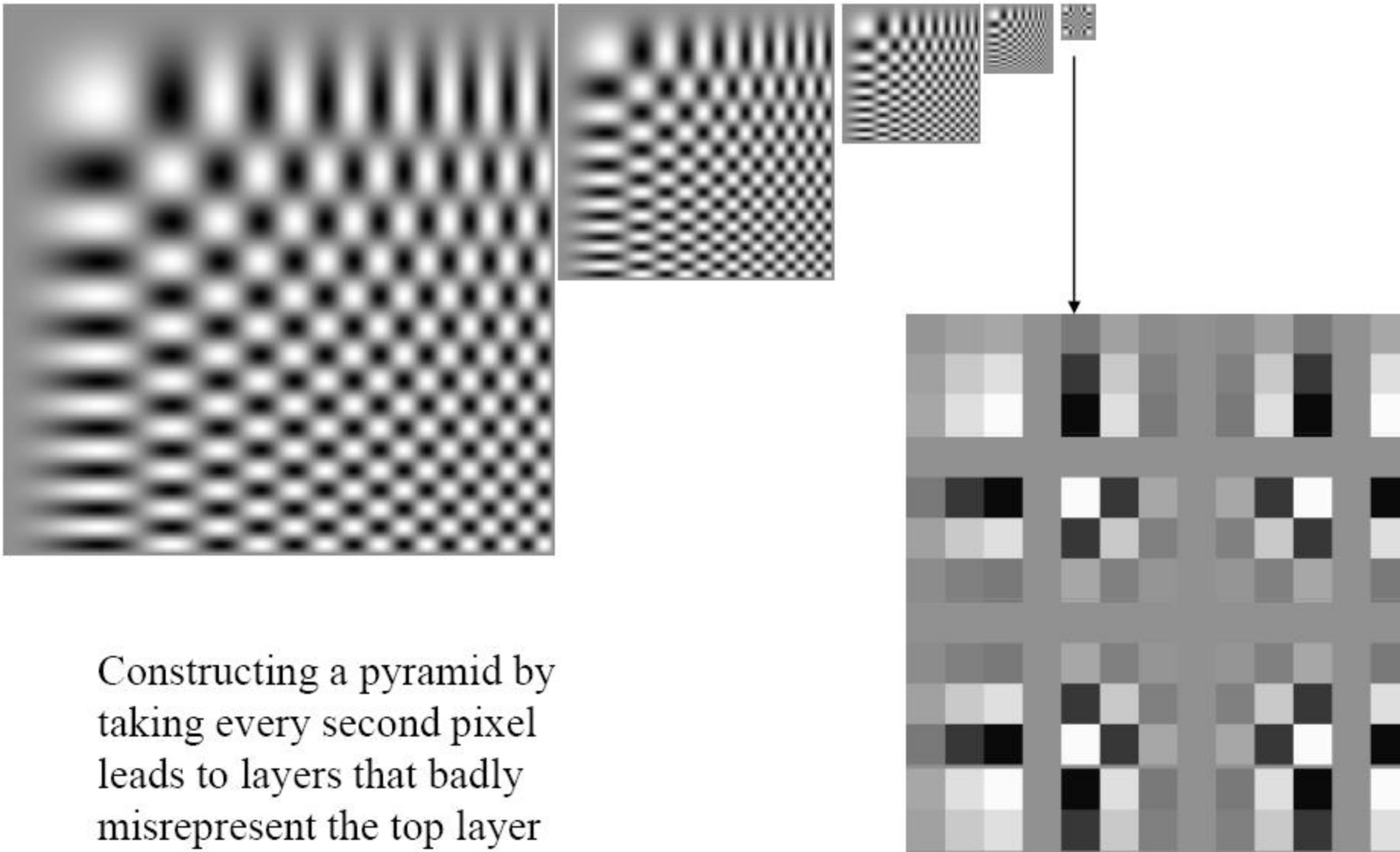
$$= F\left(f(x,y)\right) ** F\left(\sum_{i=-\infty}^{\infty}\sum_{i=-\infty}^{\infty}\delta(x-i,y-j)\right)$$

$$= \sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty}F(u-i,v-j)$$



Resample the checkerboard by taking one sample at each circle. In the case of the top left board, new representation is reasonable.
Top right also yields a reasonable representation.
Bottom left is all black (dubious) and bottom right has checks that are too big.

Sampling and aliasing



Constructing a pyramid by taking every second pixel leads to layers that badly misrepresent the top layer

Sampling in 1D

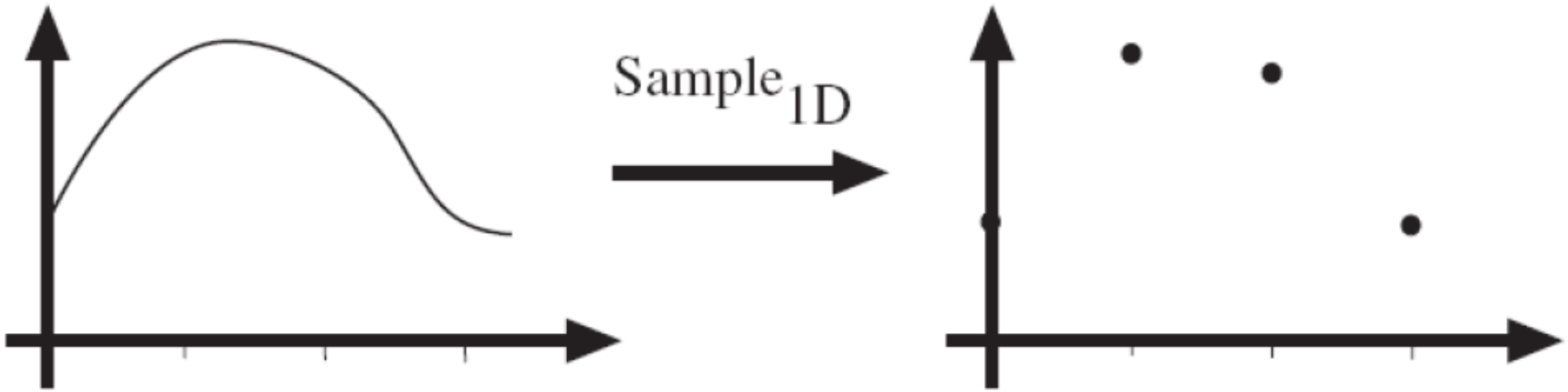
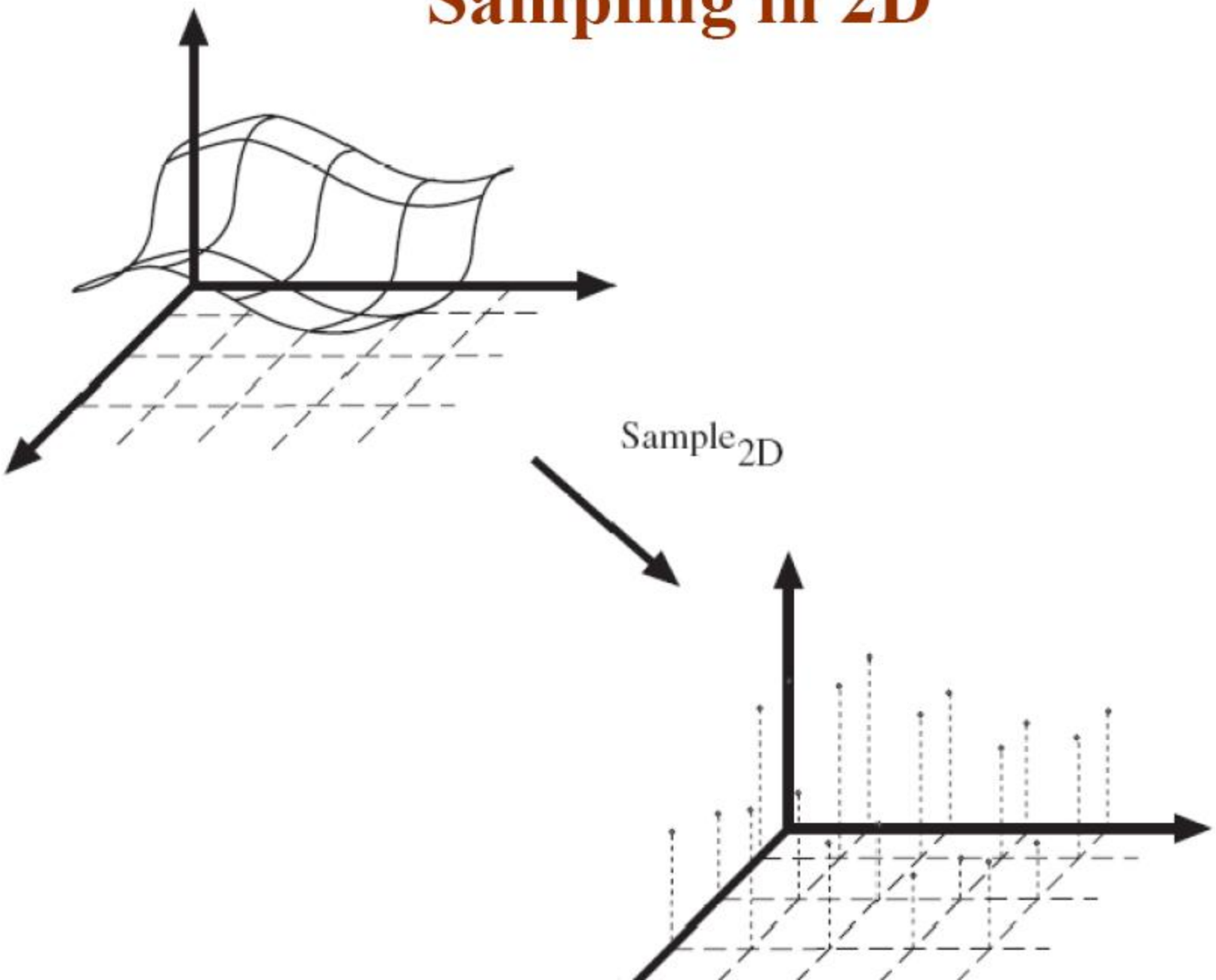
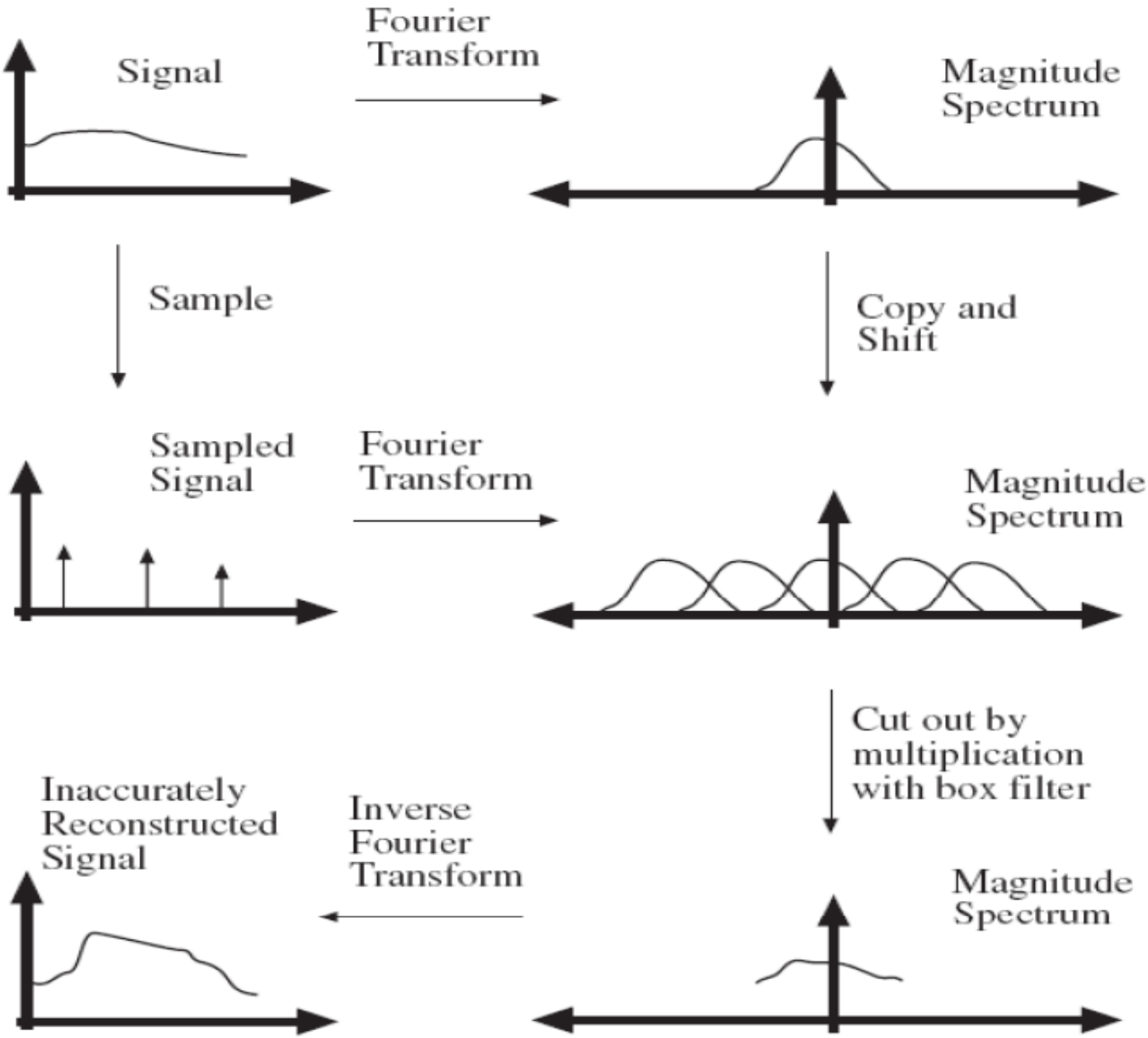
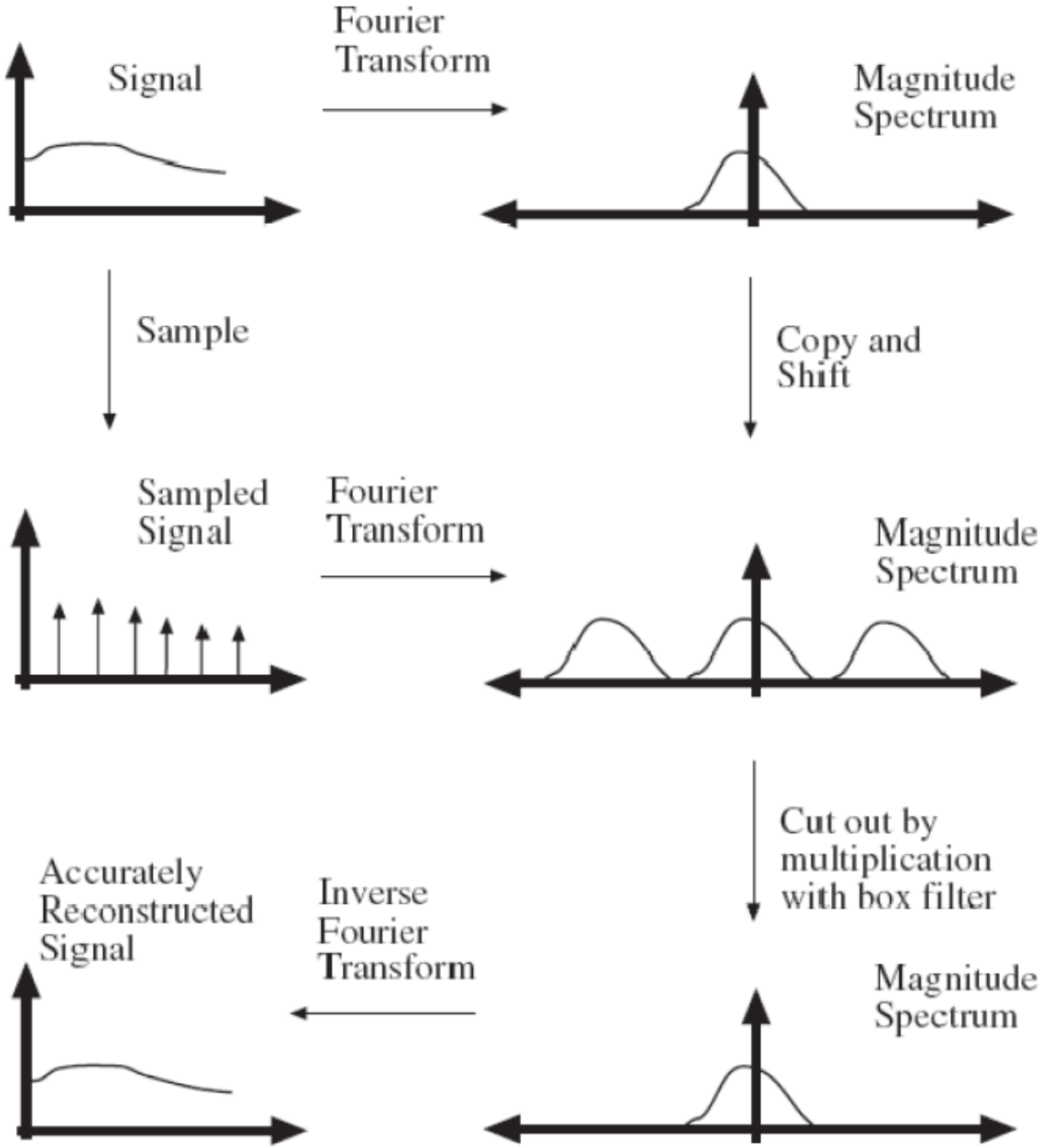


FIGURE 7.8: Sampling in 1D takes a function, and returns a vector whose elements are values of that function at the sample points, as the top figures show. For our purposes, it is enough that the sample points be integer values of the argument. We allow the vector to be infinite dimensional, and have negative as well as positive indices.

Sampling in 2D




Signal
Fourier Transform
Magnitude Spectrum
Sample
Copy and Shift
Sampled Signal
Fourier Transform
Magnitude Spectrum
Cut out by multiplication with box filter
Inaccurately Reconstructed Signal
Inverse Fourier Transform
Magnitude Spectrum


Signal
Fourier Transform
Magnitude Spectrum
Sample
Copy and Shift
Sampled Signal
Fourier Transform
Magnitude Spectrum
Cut out by multiplication with box filter
Accurately Reconstructed Signal
Inverse Fourier Transform
Magnitude Spectrum

Multiscale representations

Multiscale representations

- Find template matches at all scales
  - e.g., when finding hands or faces, we don't know what size they will be in a particular image
  - Template size is constant, but image size changes
- Efficient search for correspondence
  - look at coarse scales, then refine with finer scales
  - much less cost, but may miss best match
- Examining all levels of detail
  - Find edges with different amounts of blur
  - Find textures with different spatial frequencies (levels of detail)

Image information occurs at all spatial scales

Scale invariance

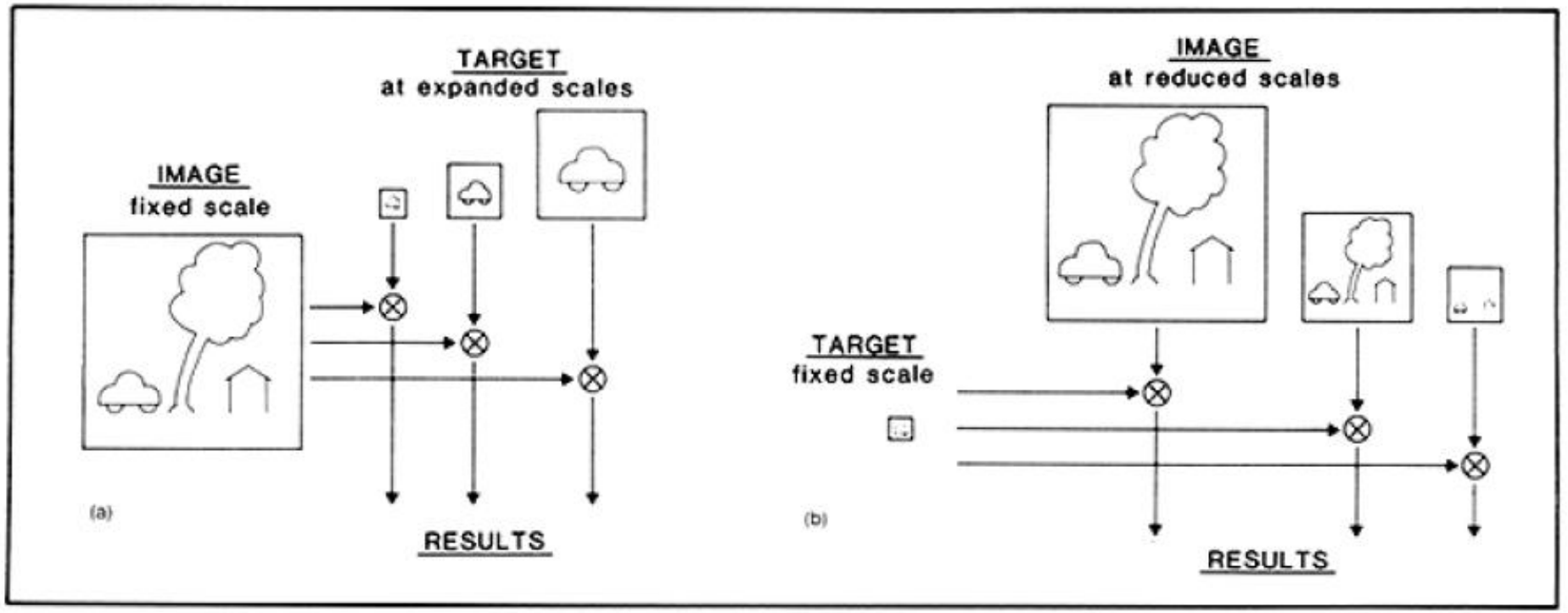


Fig. 1. Two methods of searching for a target pattern over many scales. In the first approach, (a), copies of the target pattern are constructed at several expanded scales, and each is convolved with the original image. In the second approach, (b), a single copy of the target is convolved with copies of the image reduced in scale. The target should be just large enough to resolve critical details The two approaches should give equivalent results, but the second is more efficient by the fourth power of the scale factor (image convolutions are represented by 'O').

Pyramid methods in image processing
Adelson et al, 84

The Gaussian pyramid

- **Create each level from previous one:**
  - **smooth and sample**
- **Smooth with gaussians, because**
  - **a gaussian*gaussian=another gaussian**
- **Gaussians are low pass filters, so representation is redundant.**

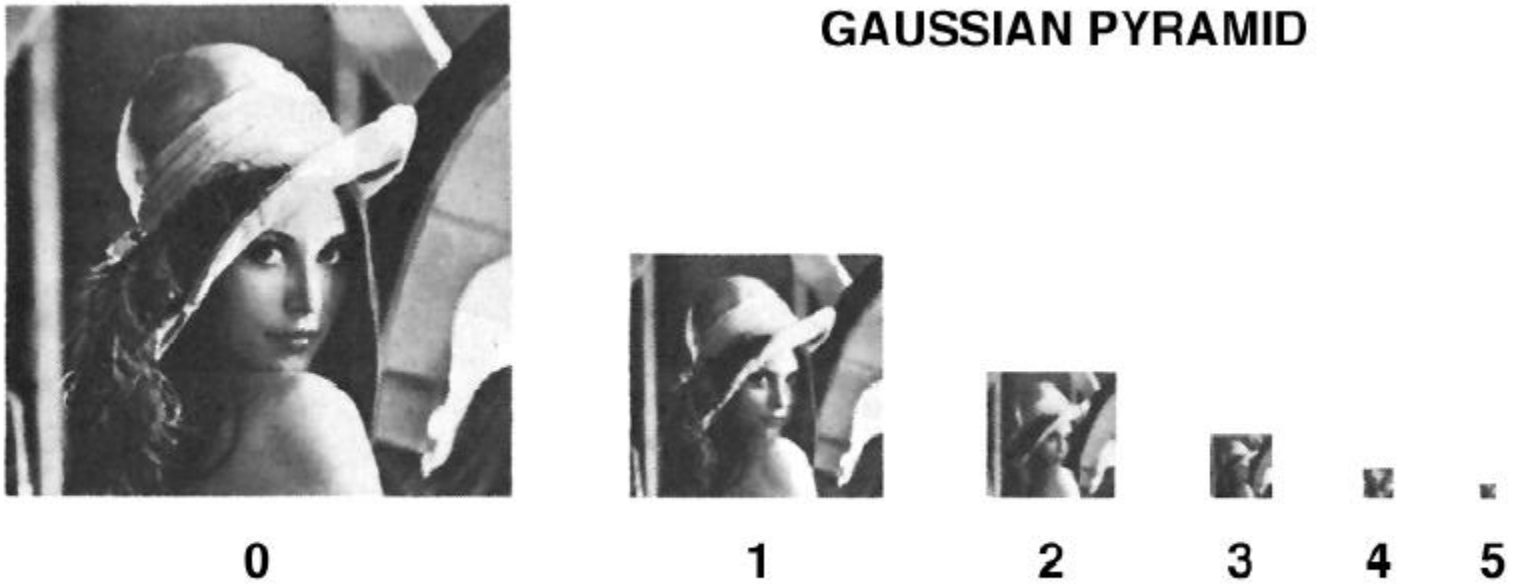


Fig. 4. First six levels of the Gaussian pyramid for the "Lady" image The original image, level 0, meusures 257 by 257 pixels and each higher level array is roughly half the dimensdons of its predecessor. Thus, level 5 measures just 9 by 9 pixels.

http://www-bcs.mit.edu/people/adelson/pub_pdfs/pyramid83.pdf

IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-31, NO. 4, APRIL 1983

512 256 128 64 32 16 8

Some desirable properties for a blur kernel

- **Positivity:** $h(m) >= 0$
- **Symmetry:** $h(m) = h(-m)$
- **Unimodality:** $h(m) >= h(m+1)$ **for** $m >= 0$
- **Normalized:** $\Sigma h(m) = 1$
- **Equal contribution:** $\Sigma h(2m) = \Sigma h(2m+1)$

Some kernels that verify this are:

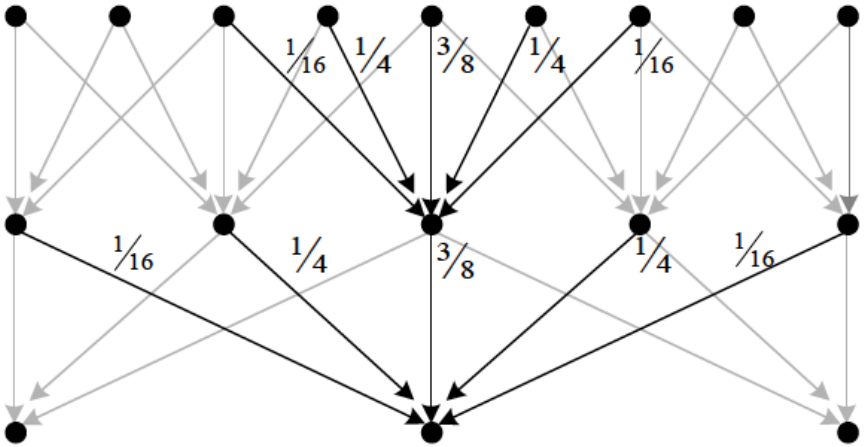
[½, ½]

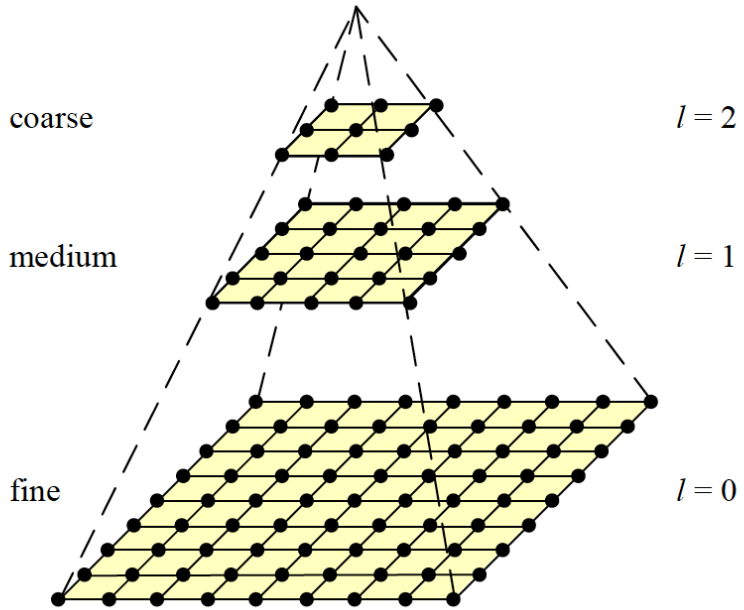
For length 5:

[¼, ½, ¼] ¼-a/2, ¼, a, ¼, ¼-a/2]

The value of **a** that best approximates a Gaussian filter is 0.4

Gaussian Pyramid





http://www-bcs.mit.edu/people/adelson/pub_pdfs/pyramid83.pdf

IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-31, NO. 4, APRIL 1983

Convolution and subsampling as a matrix multiply (1-d case)

$$x_2 = G_1 x_1$$

$$G_1 =$$

1	4	6	4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	4	6	4	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	4	6	4	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	4	6	4	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	4	6	4	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	4	6	4	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	4	6	4	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	4	6	4	1	0

(Normalization constant of 1/16 omitted for visual clarity.)

Next pyramid level

$$x_3 = G_2 x_2$$

$$G_2 = \begin{matrix} 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 \end{matrix}$$

The combined effect of the two pyramid levels

$$x_3 = G_2 G_1 x_1$$

$$G_2 G_1 = \begin{matrix}
1 & 4 & 10 & 20 & 31 & 40 & 44 & 40 & 31 & 20 & 10 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 4 & 10 & 20 & 31 & 40 & 44 & 40 & 31 & 20 & 10 & 4 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 10 & 20 & 31 & 40 & 44 & 40 & 30 & 16 & 4 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 10 & 20 & 25 & 16 & 4 & 0
\end{matrix}$$

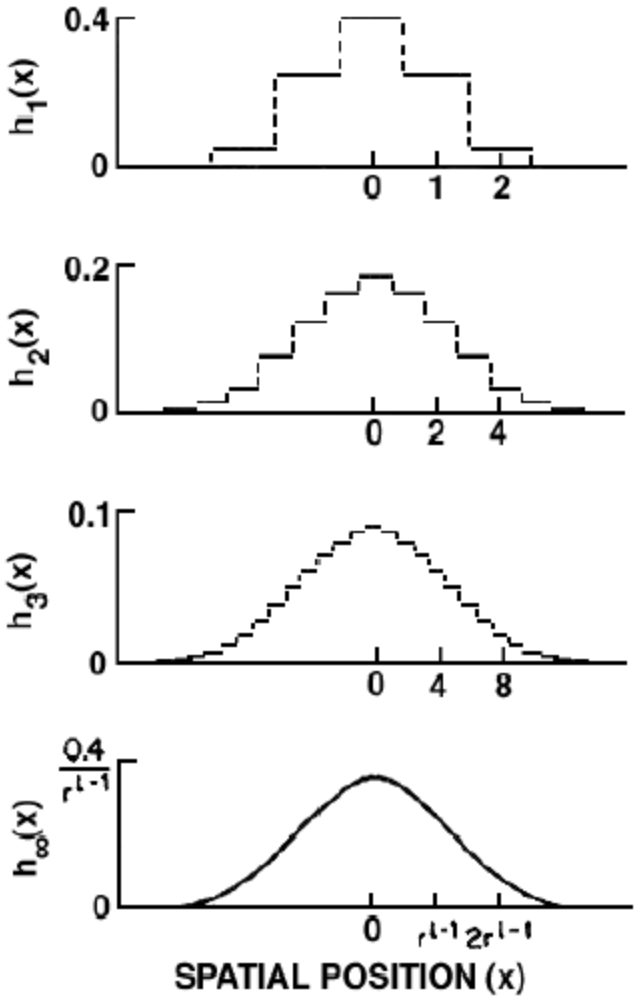
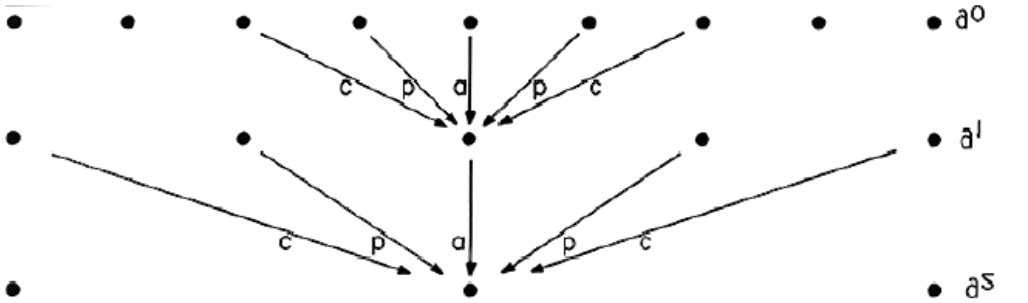


Fig. 2. The equivalent weighting functions $h_l(x)$ for nodes in levels 1, 2, 3, and infinity of the Gaussian pyramid. Note that axis scales have been adjusted by factors of 2 to aid comparison Here the parameter a of the generating kernel is 0.4, and the resulting equivalent weighting functions closely resemble the Gaussian probability density functions.



http://www-bcs.mit.edu/people/adelson/pub_pdfs/pyramid83.pdf

IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-31, NO. 4, APRIL 1983

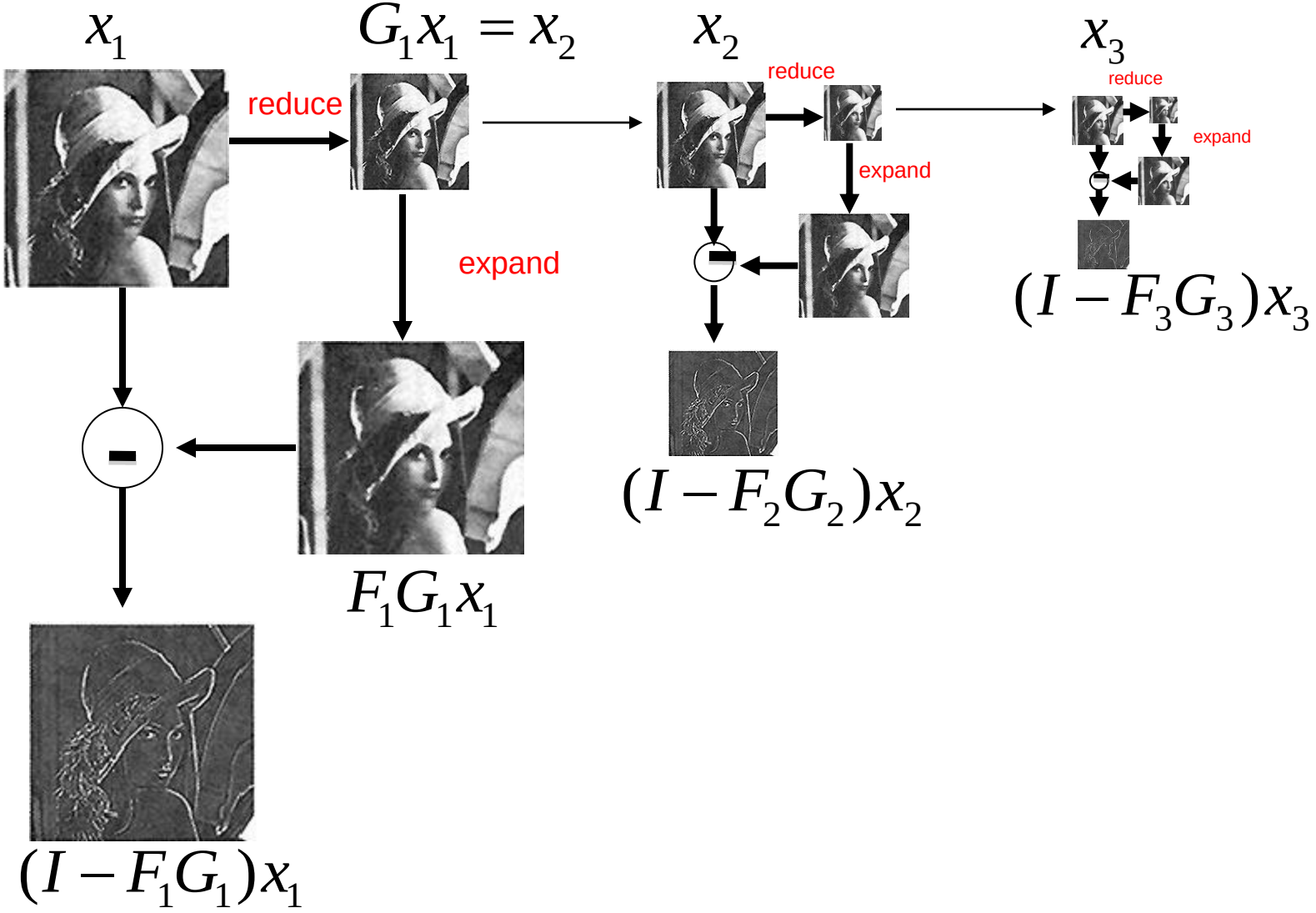
Gaussian pyramids used for

- **up- or down- sampling images.**
- **Multi-resolution image analysis**
  - **Look for an object over various spatial scales**
  - **Coarse-to-fine image processing: form blur estimate or the motion analysis on very low-resolution image, upsample and repeat. Often a successful strategy for avoiding local minima in complicated estimation tasks.**

The Laplacian Pyramid

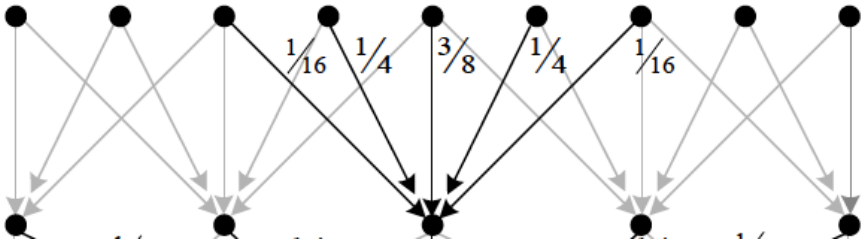
- **Synthesis**
  - **Compute the difference between upsampled Gaussian pyramid level and Gaussian pyramid level.**
  - **band pass filter - each level represents spatial frequencies (largely) unrepresented at other level.**
- **Applications:**
  - **Texture synthesis; Image compression; Noise removal; Image blending**

Laplacian pyramid algorithm

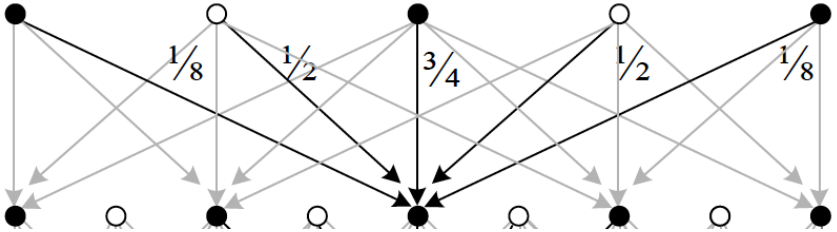


Reduce and expand

Reduce



Expand



http://www-bcs.mit.edu/people/adelson/pub_pdfs/pyramid83.pdf

IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-31, NO. 4, APRIL 1983

Showing, at full resolution, the information captured at each level of a Gaussian (top) and Laplacian (bottom) pyramid.

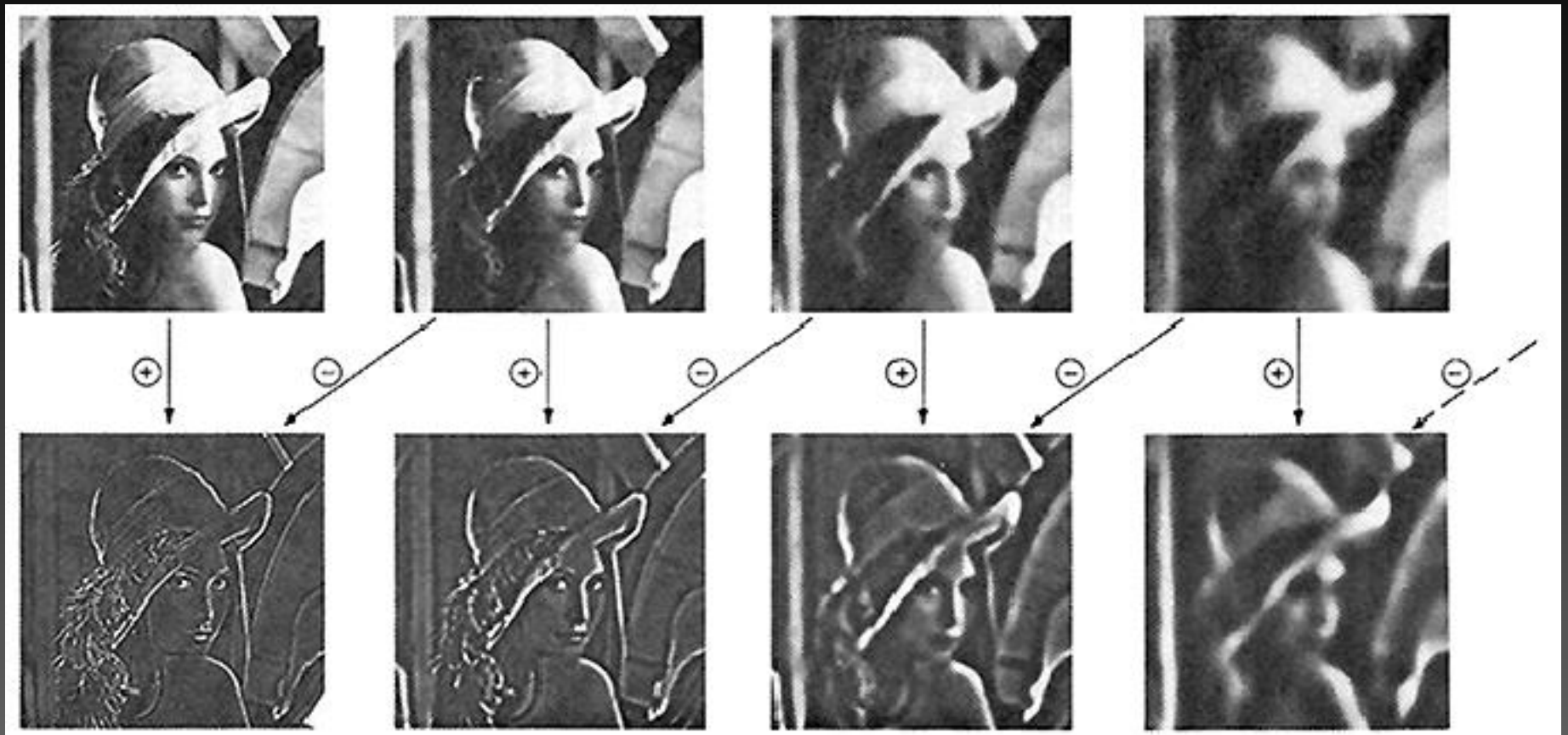


Fig 5. First four levels of the Gaussian and Laplacian pyramid. Gaussian images, upper row, were obtained by expanding pyramid arrays (Fig. 4) through Gaussian interpolation. Each level of the Laplacian pyramid is the difference between the corresponding and next higher levels of the Gaussian pyramid.

http://www-bcs.mit.edu/people/adelson/pub_pdfs/pyramid83.pdf

IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-31, NO. 4, APRIL 1983

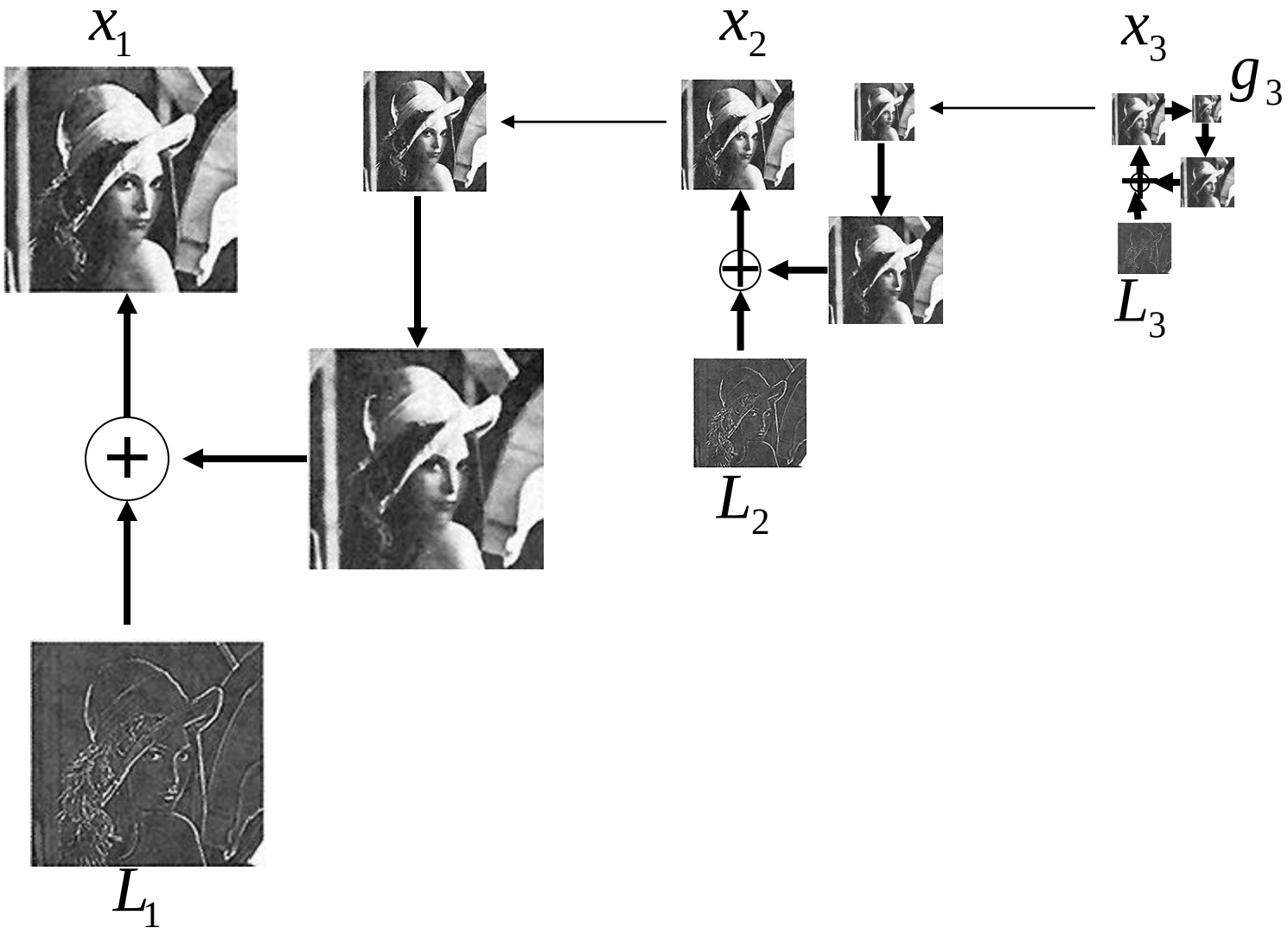
Laplacian pyramid reconstruction algorithm: recover x_1 from L_1, L_2, L_3 and x_4

G# is the blur-and-downsample operator at pyramid level #
F# is the blur-and-upsample operator at pyramid level #

Laplacian pyramid elements:
L1 = (I – F1 G1) x1
L2 = (I – F2 G2) x2
L3 = (I – F3 G3) x3
x2 = G1 x1
x3 = G2 x2
x4 = G3 x3

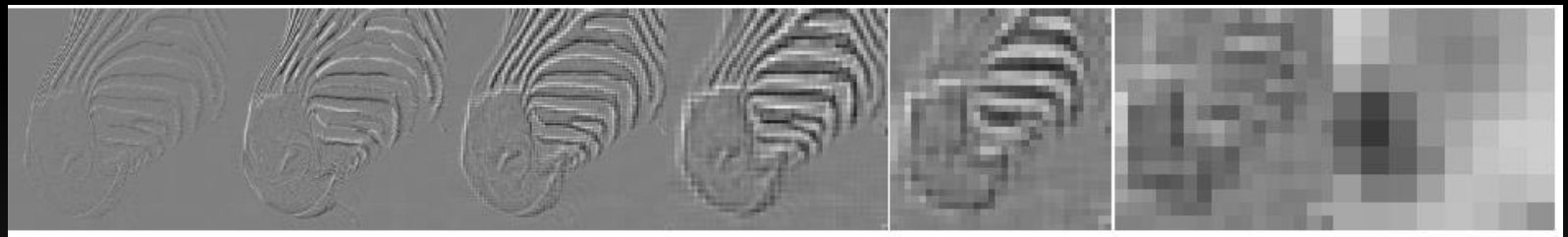
Reconstruction of original image (x1) from Laplacian pyramid elements:
x3 = L3 + F3 x4
x2 = L2 + F2 x3
x1 = L1 + F1 x2

Laplacian pyramid reconstruction algorithm: recover x_1 from L_1, L_2, L_3 and g_3



512 256 128 64 32 16 8

Gaussian pyramid



512 256 128 64 32 16 8

Laplacian pyramid

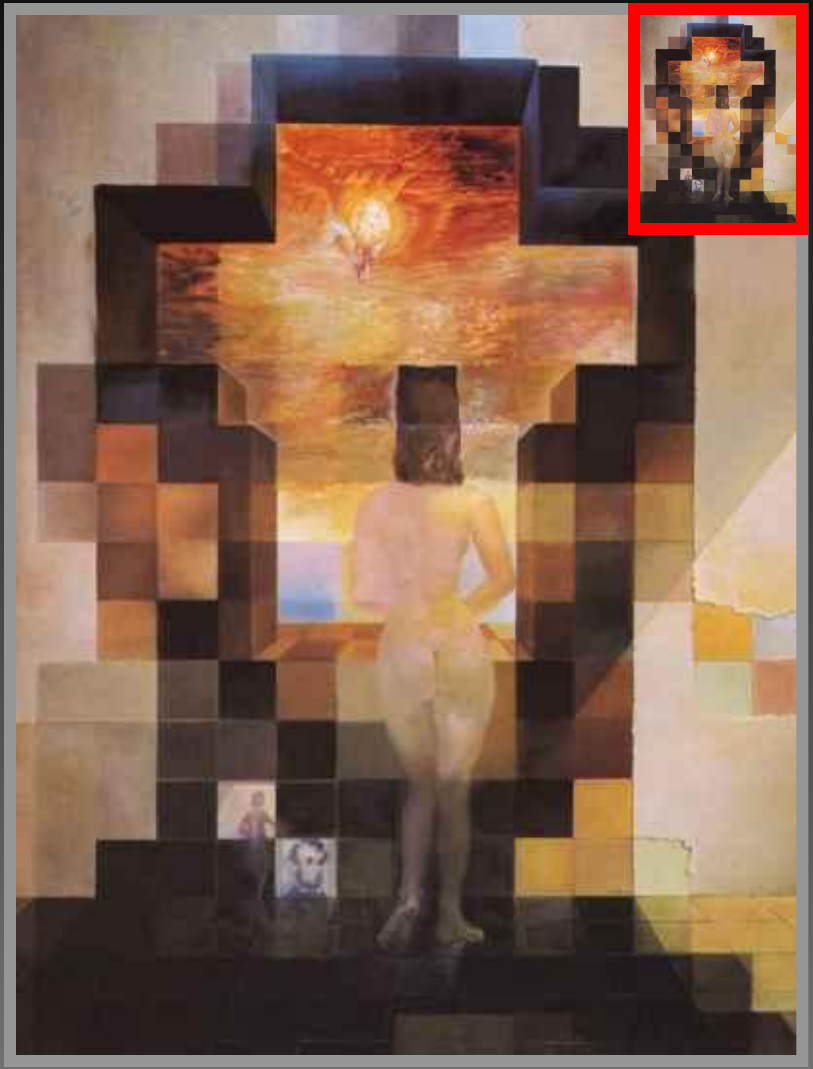
Why use these representations?

- Handle real-world size variations with a constant-size vision algorithm.
- Remove noise
- Analyze texture
- Recognize objects
- Label image features
- Image priors can be specified naturally in terms of wavelet pyramids.

Visual Perception

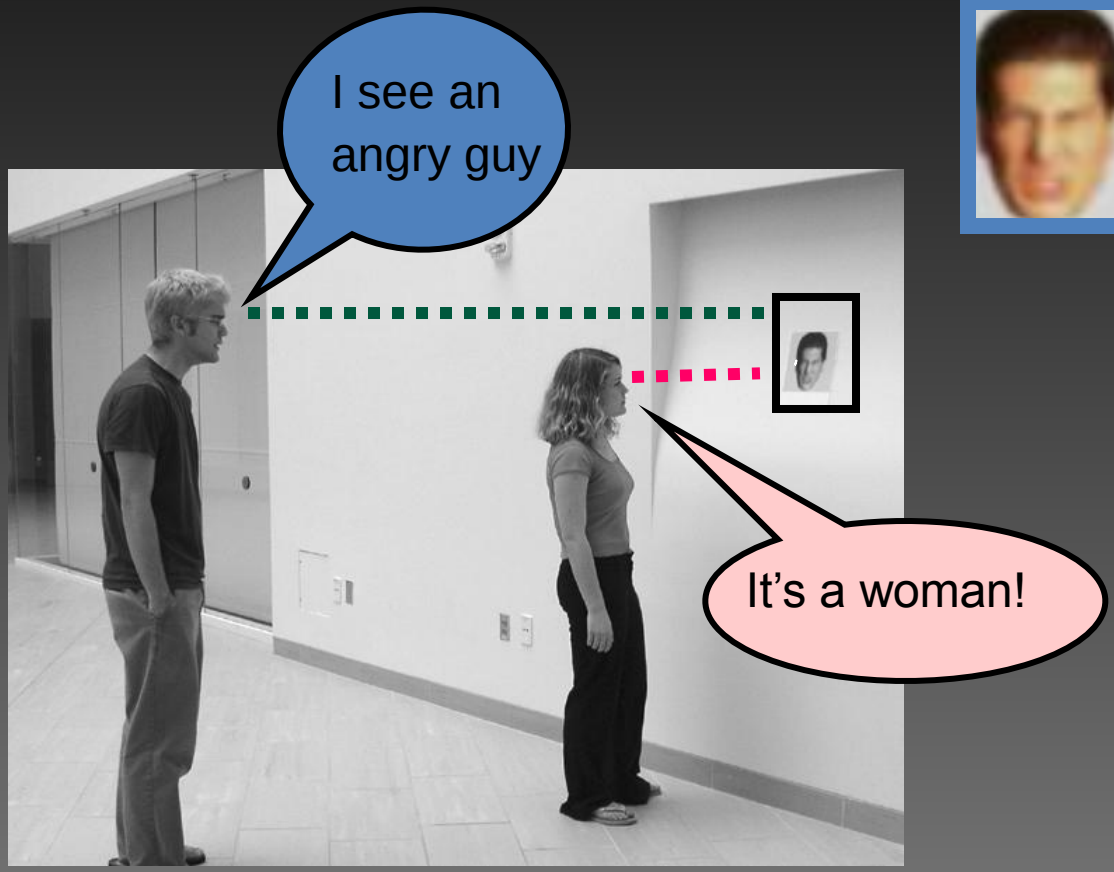
For computer vision systems to “understand art” other properties of human visual perception must be taken into account

Salvador Dali



Hybrid Images

What you see…



From Far Away

Up Close

Advances in multi-resolution perception

- **We tend to constantly reinterpret the representation if we are gaining information**

- **We tend to stick with our first grouping interpretation if we are losing information. The hysteresis effect is modulated by the strength of perceptual grouping across scales.**

The visual system is designed to integrate across multi resolution in a way that provides **the most accurate interpretation** of the world

as we move through it.

Readings

- **Chapter 4**