COLLEGE OF ENGINEERING
ELECTRICAL ENGINEERING & COMPUTER SCIENCE
UNIVERSITY OF MICHIGAN

Linear Filters and Image Processing

EECS 598-08 Fall 2014
Foundations of Computer Vision

Instructor: Jason Corso (jjcorso)
web.eecs.umich.edu/~jjcorso/t/598F14

Readings: FP 4, 6.1, 6.4; SZ 3

Date: 9/24/14

Materials on these slides have come from many sources in addition to myself; I am infinitely grateful to these, especially Greg Hager, Silvio Savarese, and Steve Seitz.

Topics

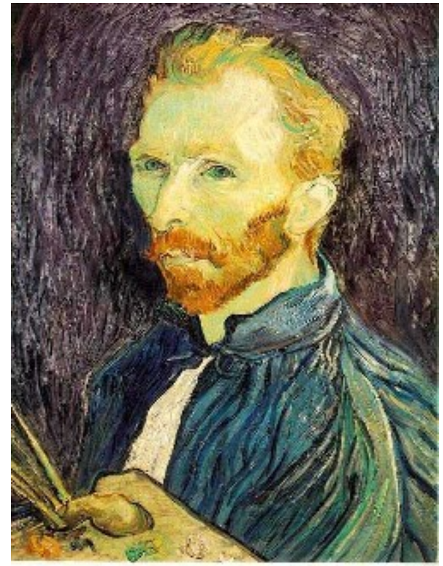
- Linear filters
- Scale-space and image pyramids
- Image denoising
- Representing texture by filters

De-noising

Original

Salt and pepper noise

Super-resolution



In-painting

Image Inpainting, M. Bertalmío et al.
http://www.iua.upf.es/~mbertalmio//restoration.html

Image Inpainting, M. Bertalmío et al.
http://www.iua.upf.es/~mbertalmio//restoration.html

Source: Savarese Slides

Images as functions

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

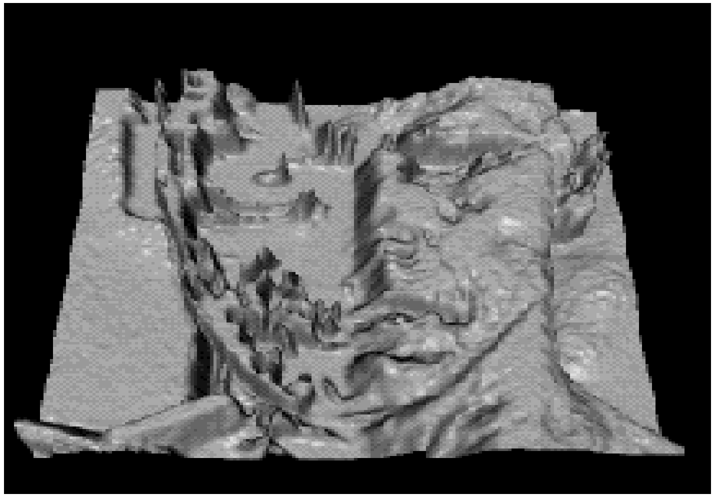
$$f: [a, b] \times [c, d] \rightarrow [0, 1]$$

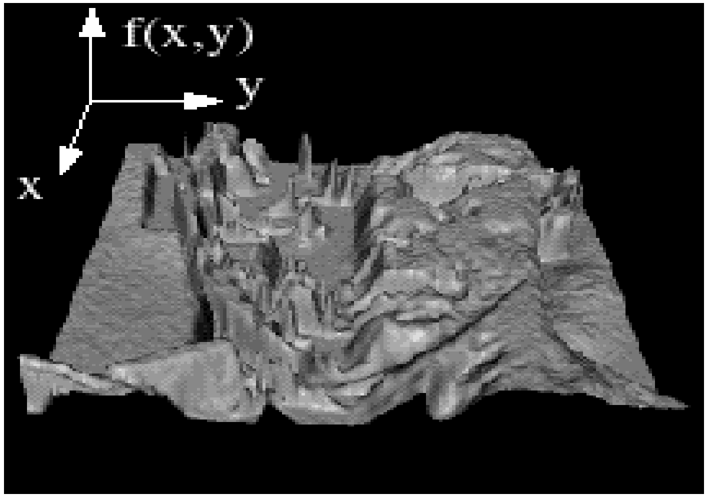
- A color image is just three functions pasted together. We can write this as a "vector-valued" function:

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

Source: Seitz and Szeliski Slides

Images as functions





Source: Seitz and Szeliski Slides

What is a digital image?

- We usually work with **digital** (**discrete**) images:
  - **Sample** the 2D space on a regular grid
  - **Quantize** each sample (round to nearest integer)
- If our samples are Δ apart, we can write this as:

$$f[i,j] = \text{Quantize}\{f(i\Delta, j\Delta)\}$$

- The image can now be represented as a matrix of integer values

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

Source: Seitz and Szeliski Slides

Filtering noise

- How can we “smooth” away noise in an image?

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	100	130	110	120	110	0	0
0	0	0	110	90	100	90	100	0	0
0	0	0	130	100	90	130	110	0	0
0	0	0	120	100	130	110	120	0	0
0	0	0	90	110	80	120	100	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Source: Seitz and Szeliski Slides

Mean filtering

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

Source: Seitz and Szeliski Slides

Source: Seitz and Szeliski Slides

Mean filtering

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Source: Seitz and Szeliski Slides

Cross-correlation filtering

- As an equation: Assume the window is (2k+1)x(2k+1):

$$G[i,j] = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]$$

- We can generalize this idea by allowing different weights for different neighboring pixels:

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

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

$$G = H \otimes F$$

- H is called the **filter, kernel,** or **mask**.

Mean kernel

- What's the kernel for a 3x3 mean filter?

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x, y]$

$H[u, v]$

Source: Seitz and Szeliski Slides

Gaussian filtering

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

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x, y]$

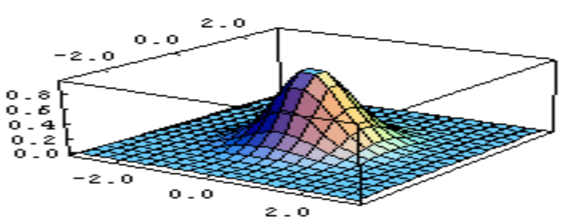
$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$H[u, v]$

- This kernel is an approximation of a Gaussian function:

$$H[u, v] = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{u^2 + v^2}{2\sigma^2}\right)$$



- What happens if you increase σ ?

Source: Seitz and Szeliski Slides

Separability of the Gaussian filter

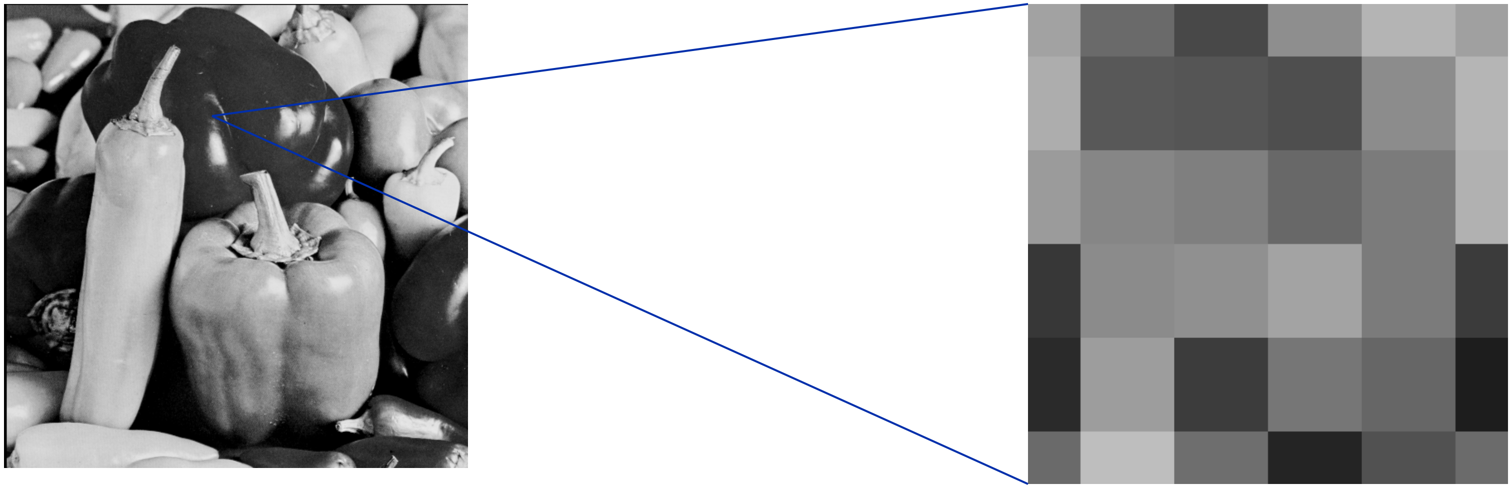
- The Gaussian function (2D) can be expressed as the product of two one-dimensional functions in each coordinate axis.
  - They are identical functions in this case.

$$
\begin{aligned}
H[u,v] &= \frac{1}{2\pi\sigma^2}\exp\left(-\frac{u^2+v^2}{2\sigma^2}\right) \\
&= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{u^2}{2\sigma^2}\right)\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{v^2}{2\sigma^2}\right)\right)
\end{aligned}
$$

- What are the implications for filtering?

Source: D. Lowe

IMAGE NOISE



Cameras are not perfect sensors *and*
Scenes never quite match our expectations

Source: G Hager Slides

Noise Models

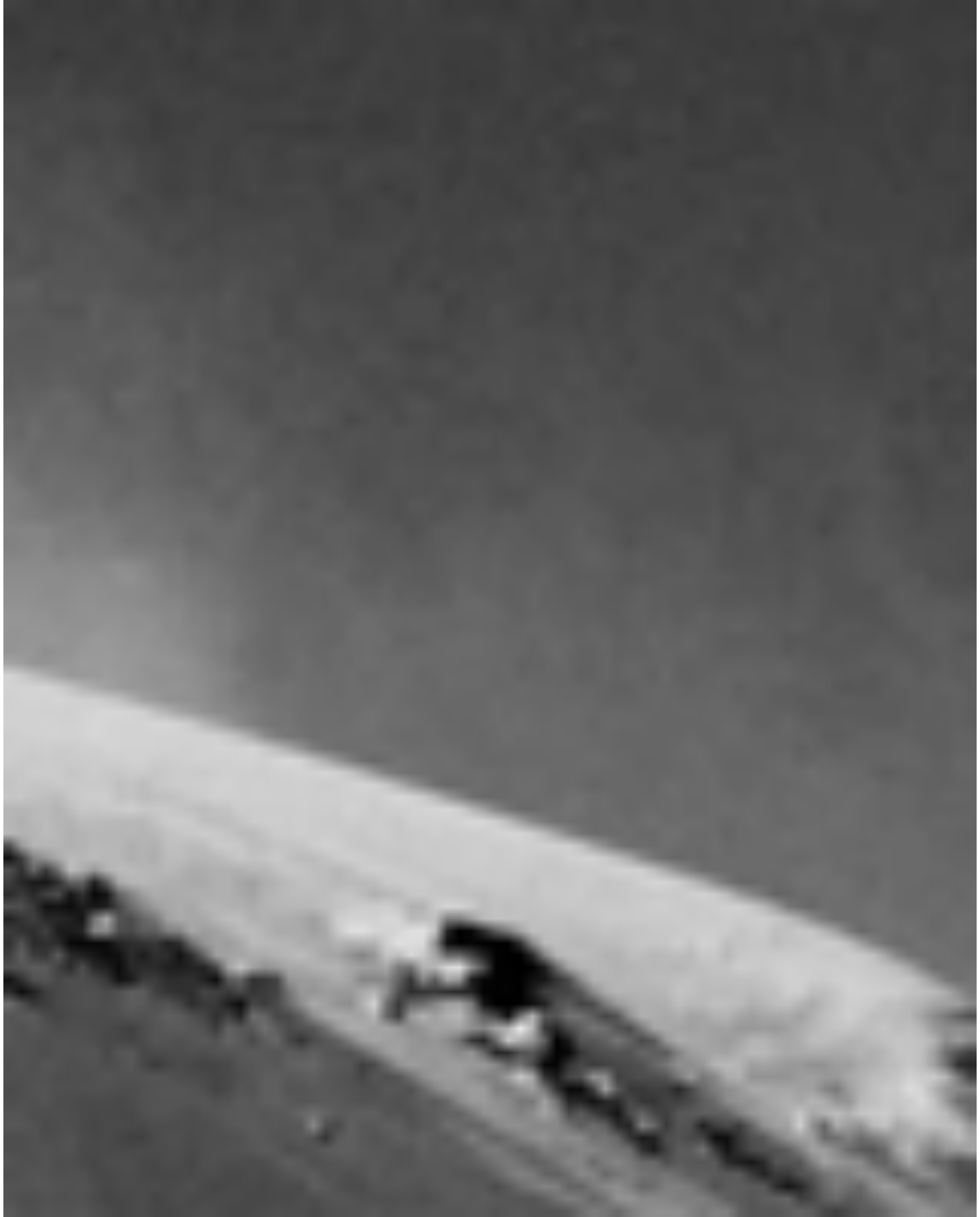
- Noise is commonly modeled using the notion of "additive white noise."
  - Images: I(u,v,t) = I*(u,v,t) + n(u,v,t)
  - Note that n(u,v,t) is independent of n(u',v',t') unless u'=u,u'=u,t'=t.
  - Typically we assume that n (noise) is independent of image location as well --- that is, it is i.i.d
  - Typically we assume the n is zero mean, that is E[n(u,v,t)]=0
- A typical noise model is the Gaussian (or normal) distribution parametrized by π and σ

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

- This implies that no two images of the same scene are ever identical

Source: G Hager Slides

Gaussian
Noise:
sigma=1

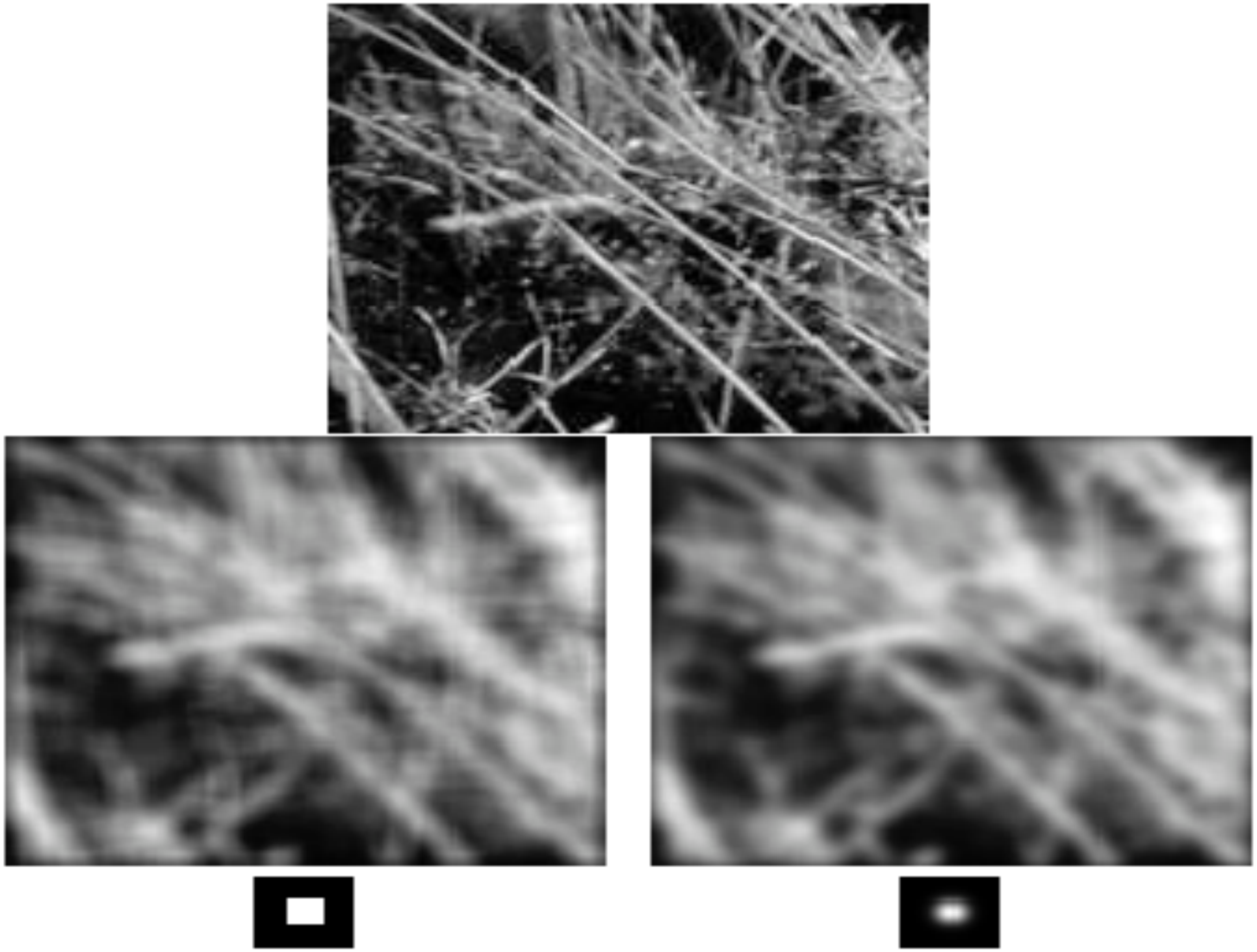


Source: G Hager Slides

Gaussian Noise: sigma=16

Source: G Hager Slides

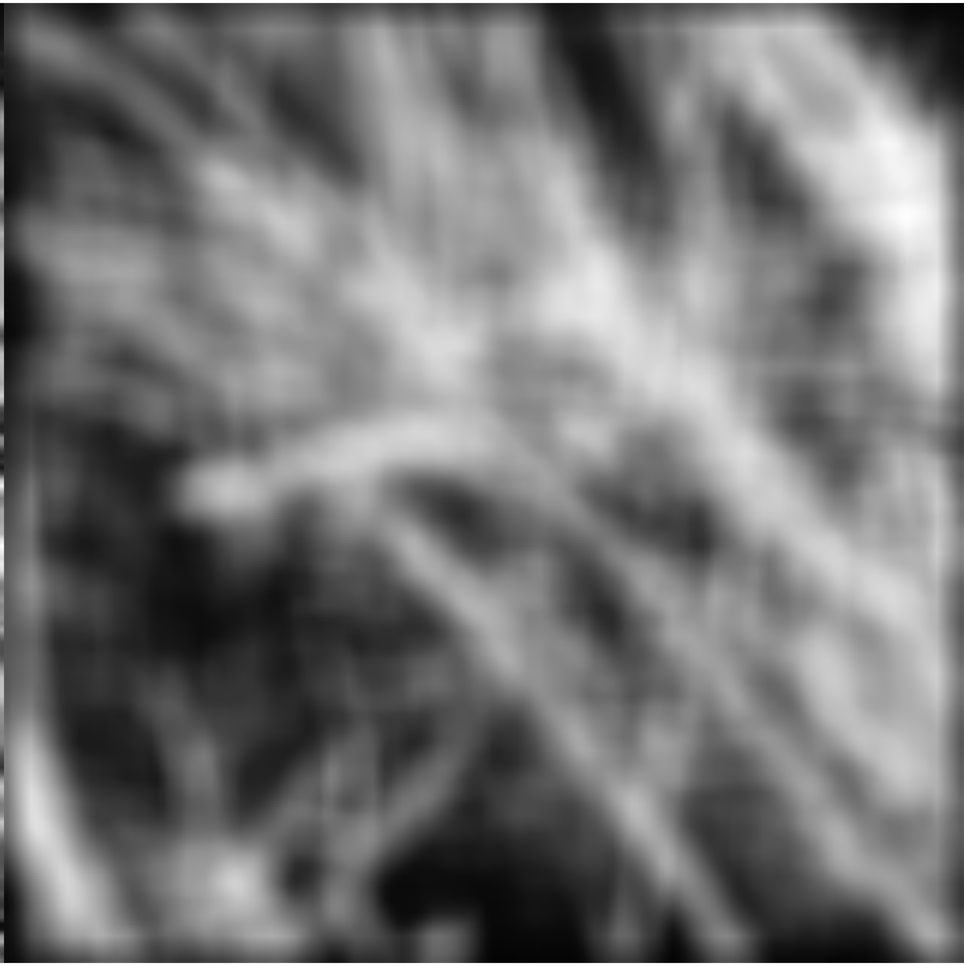
Mean vs. Gaussian filtering



Source: Seitz and Szeliski Slides

Smoothing by Averaging

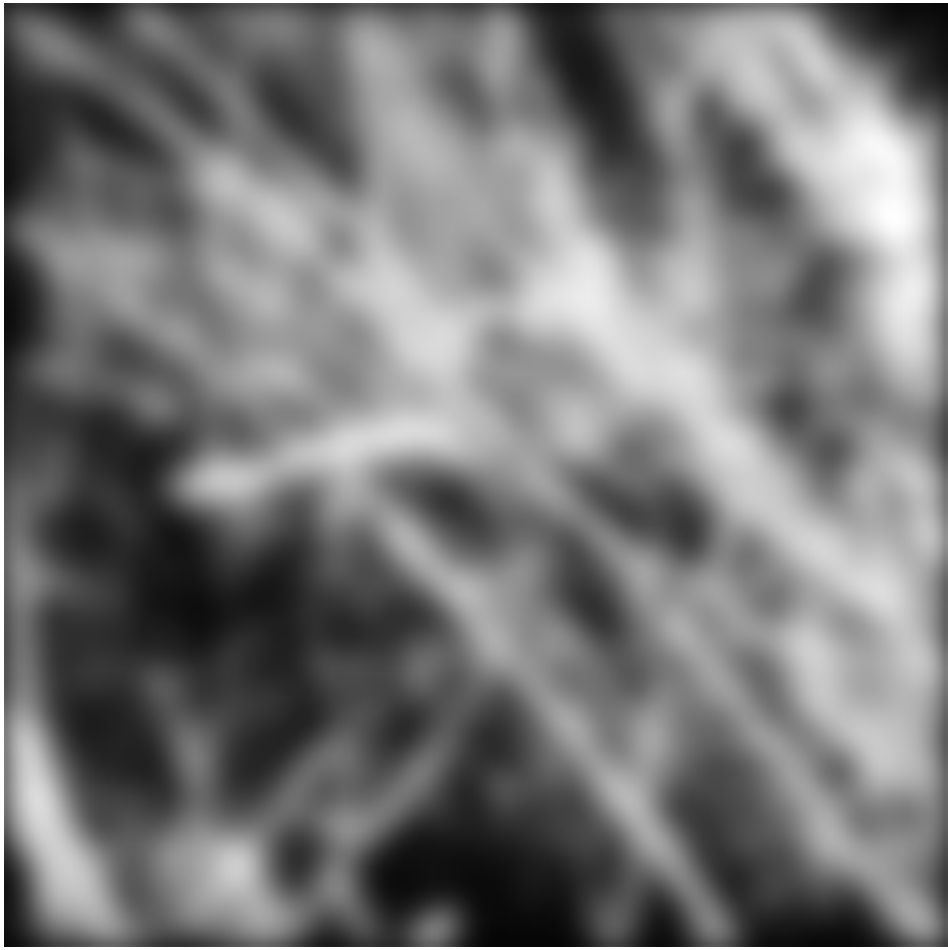
Kernel: 



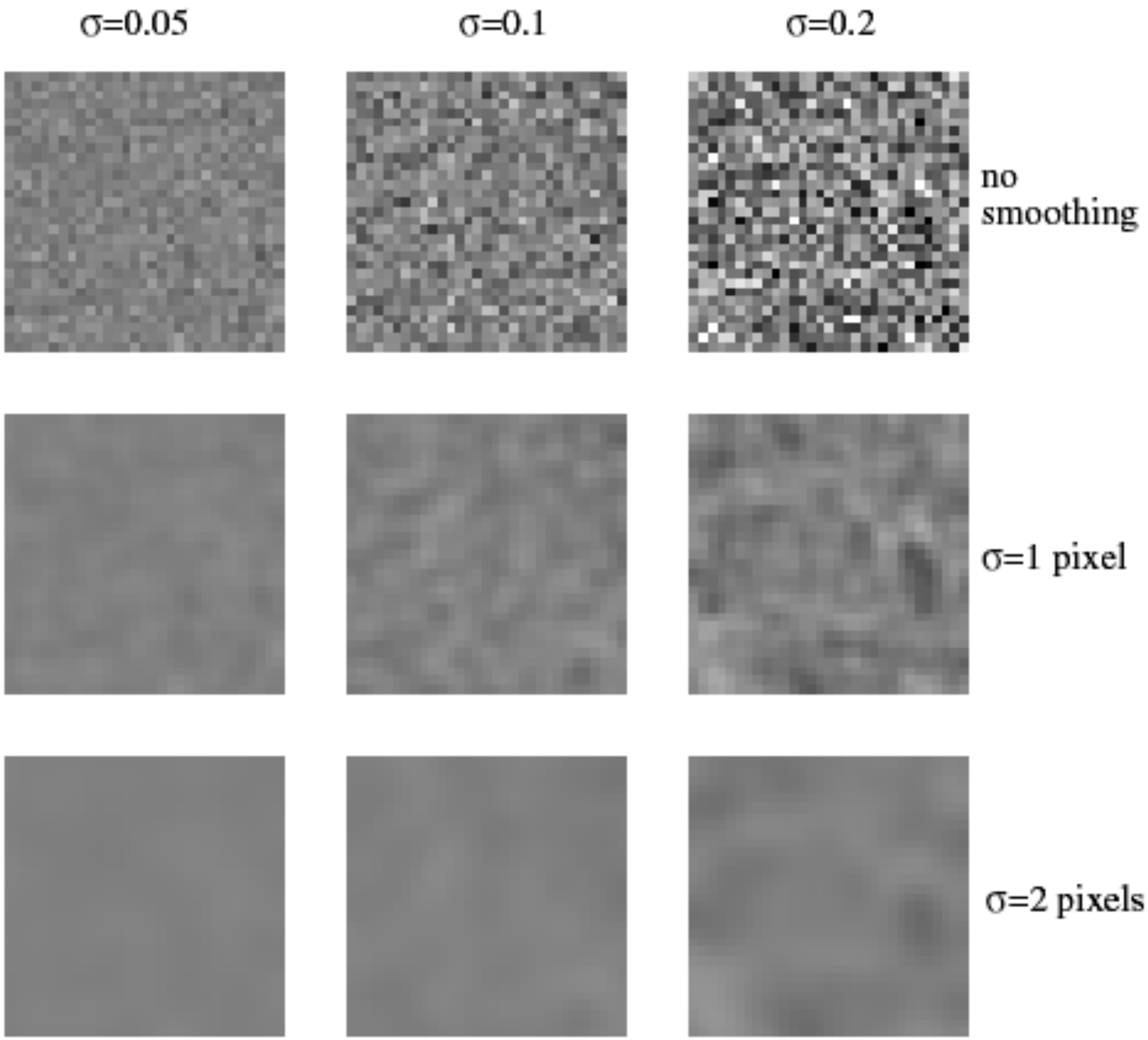
Source: G Hager, D. Kriegman Slides

Smoothing with a Gaussian

Kernel: 



Source: G Hager, D. Kriegman Slides



The effects of smoothing
Each row shows smoothing with gaussians of different width; each column shows different realizations of an image of gaussian noise.

Source: G Hager, D. Kriegman Slides

Properties of Noise Processes

- Properties of temporal image noise:

Mean $\mu(i,j) = \Sigma\, I(u,v,t)/n$

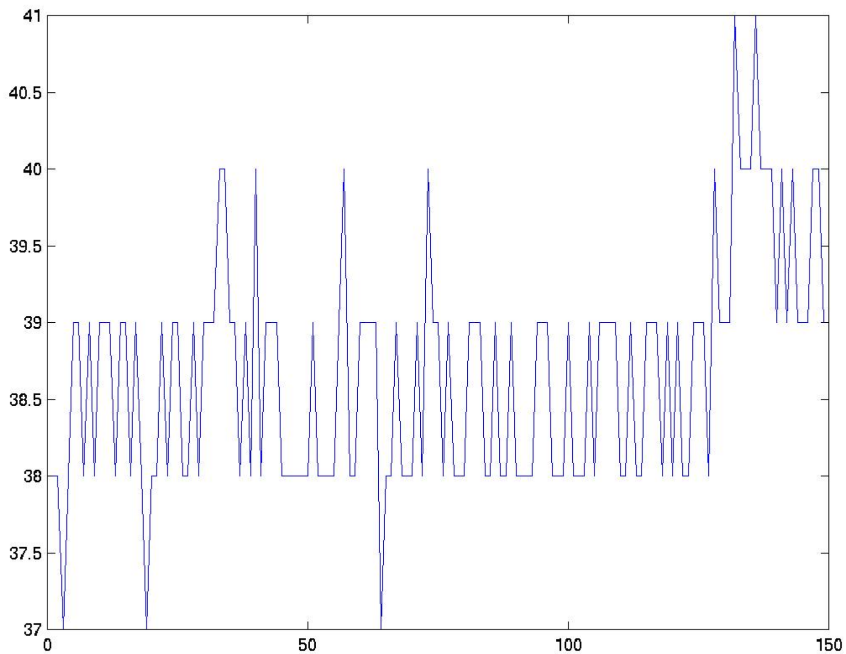
Standard Deviation $\sigma_{i,j} = \text{Sqrt}(\Sigma\, (\mu(\iota,\varphi) - I(u,v,t))^2/n \;)$

Signal-to-noise Ratio $\dfrac{\mu\,(i,j)}{\sigma_{i,j}}$

Source: G Hager Slides

Image Noise

- An experiment: take several images of a static scene and look at the pixel values



mean = 38.6
std = 2.99

Snr = 38.6/2.99 ≈ 13
max snr = 255/3 ≈ 85

Source: G Hager Slides

PROPERTIES OF TEMPORAL IMAGE NOISE

(i.e., successive images)

- If standard deviation of grey values at a pixel is s for a pixel for a single image, then the laws of statistics states that for independent sampling of grey values, for a temporal average of n images, the standard deviation is:

$$\frac{\sigma}{\mathrm{Sqrt}(n)}$$

- For example, if we want to double the signal to noise ratio, we could average 4 images.

Source: G Hager Slides

Temporal vs. Spatial Noise

- It is common to assume that:
  - spatial noise in an image is consistent with the temporal image noise
  - the spatial noise is independent and identically distributed

- Thus, we can think of a neighborhood of the image itself as approximated by an additive noise process

- Averaging is a common way to reduce noise
  - instead of temporal averaging, how about spatial?
- For example, for a pixel in image I at i,j

$$I'(i,j) = 1/9 \sum_{i'=i-1}^{i+1} \sum_{j'=j-1}^{j+1} I(i',j')$$

Correlation and Convolution

- Correlation: $G = H \otimes F$

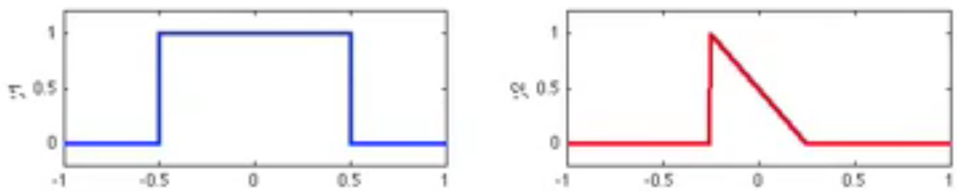
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] F[i+u, j+v]$$

- Convolution: $G = H * F$

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

Impulse Response Function

Correlation and Convolution



Source: https://www.youtube.com/watch?v=Ma0YONjMZLI

Convolution: Shift Invariant Linear Systems

- Commutative: $F * H = H * F$
  - Conceptually no difference between filter and signal
- Associative: $F * (H * L) = (F * H) * L$
  - Often apply several filters in sequence: $(((F * H_1) * H_2 * H_3)$
  - This is equivalent to applying one filter: $F * (H_1 * H_2 * H_3)$
- Linearity / Distributes over addition:

$$F * (H_1 + H_2) = (F * H_1) + (F * H_2)$$

- Scalars factor out: $kF * H = F * kH = k(F * H)$
- Shift-Invariance: $H * \mathrm{Shift}(F) = \mathrm{Shift}(H * F)$
- Identity: unit impulse

$F * e = F$

$e =$

•0	•0	•0
•0	•1	•0
•0	•0	•0

Source: Savarese Slides

Convolution: Properties

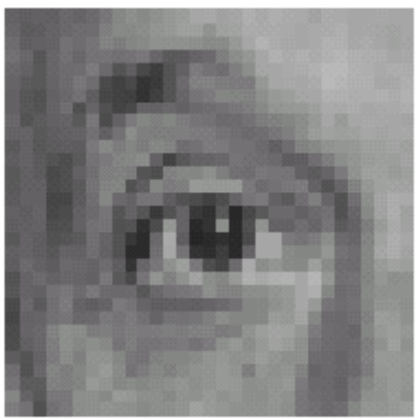
- **Linearity:** filter(f_1 + f_2) = filter(f_1) + filter(f_2)

- **Shift invariance:** filter (shift (f)) = shift (filter (f))

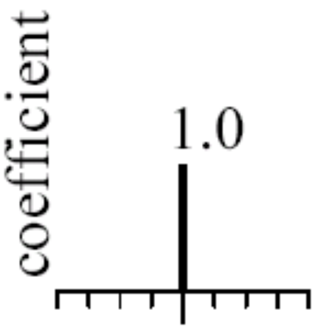
  (same behavior regardless of pixel location)

- Theoretany linear shift-invariant operator can be represented as a convolutionical result:

Linear Filtering: Status Check!



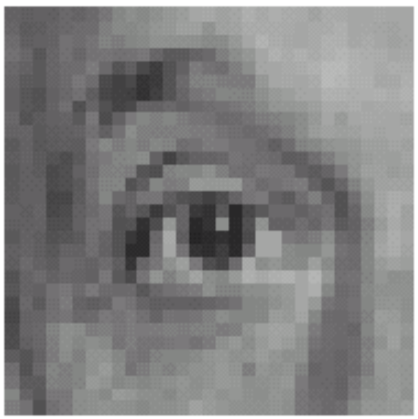
original



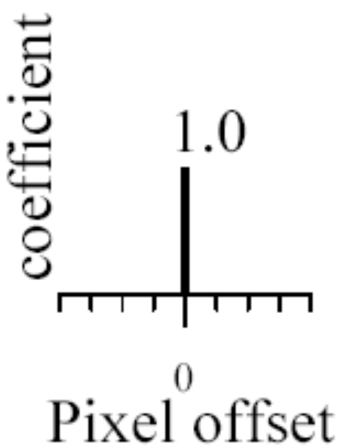
?

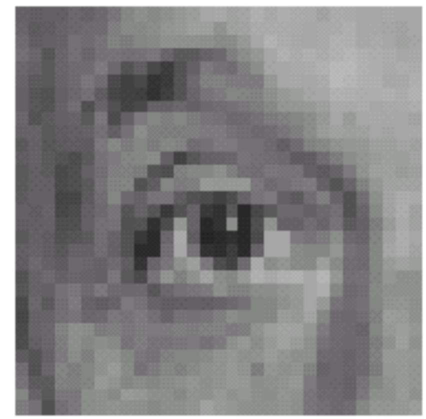
Source: B. Freeman Slides

Linear filtering (warm-up slide)



original

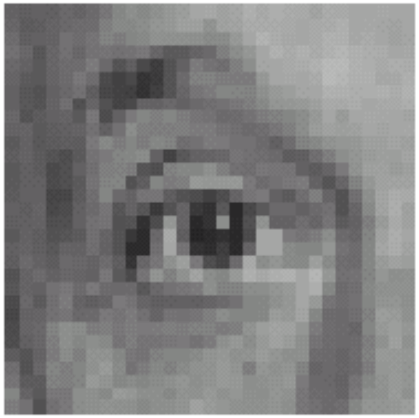




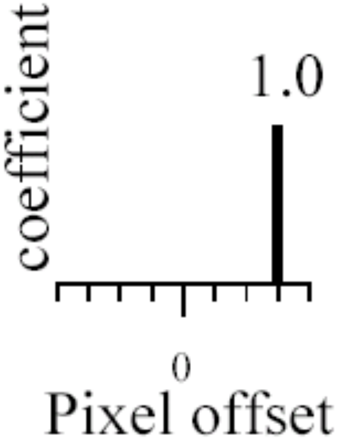
Filtered
(no change)

Source: B. Freeman Slides

Linear filtering



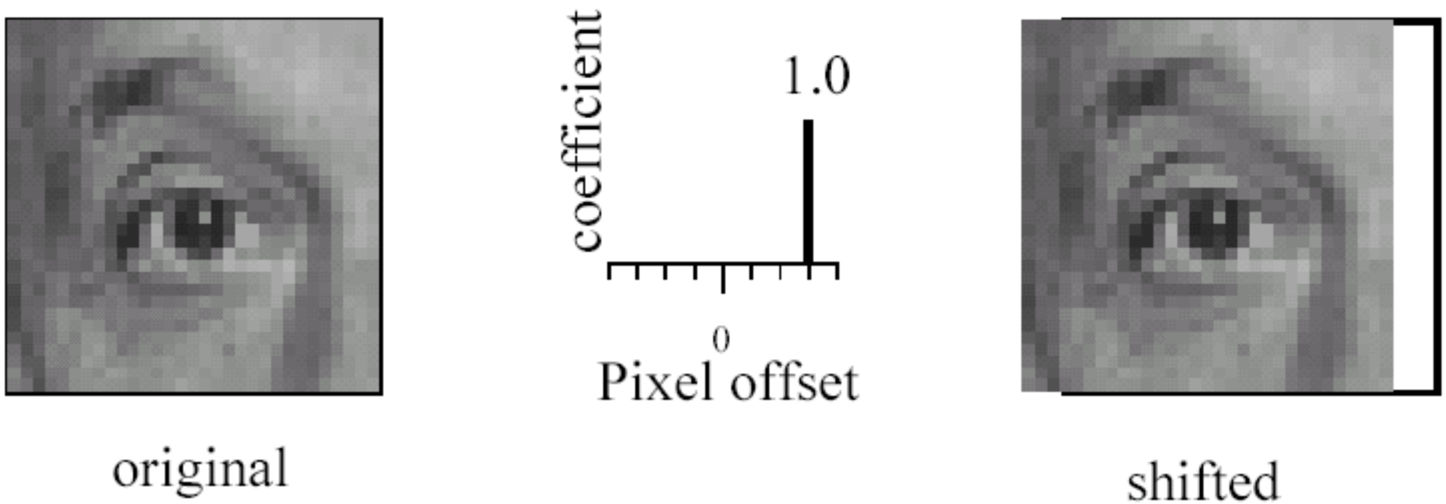
original



?

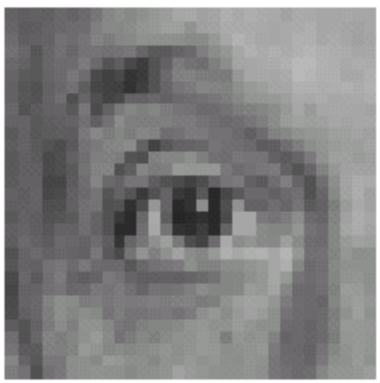
Source: B. Freeman Slides

shift

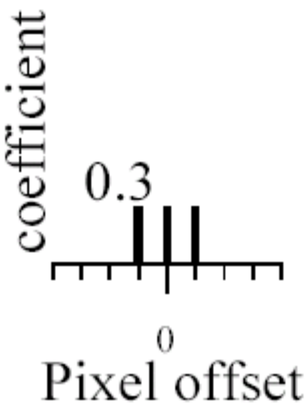


Source: B. Freeman Slides

Linear filtering

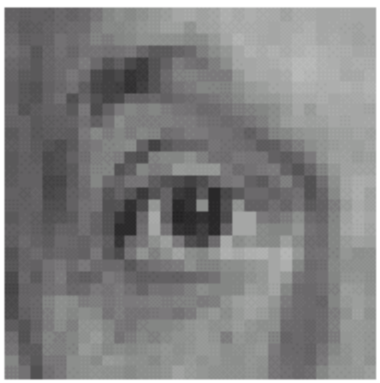


original

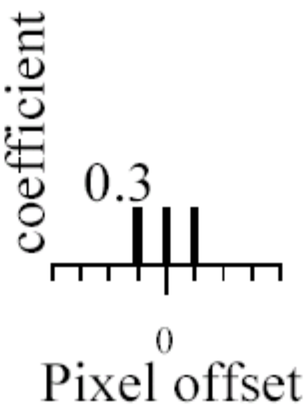


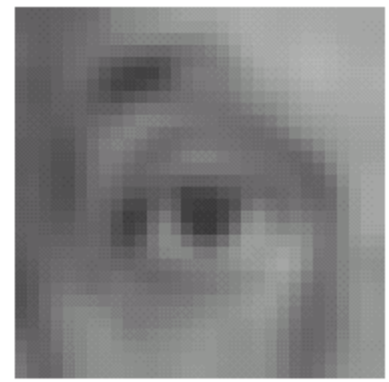
?

Blurring



original

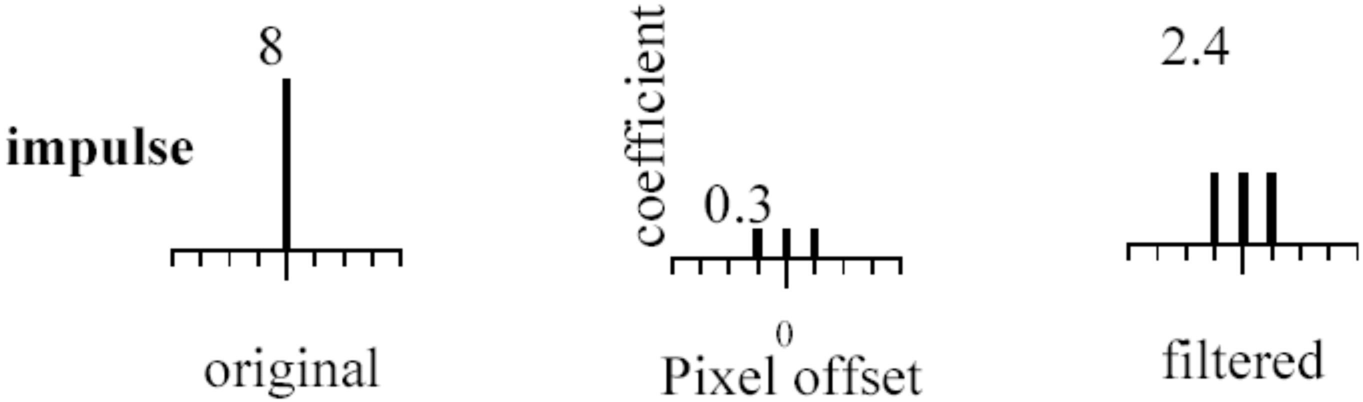




Blurred (filter applied in both dimensions).

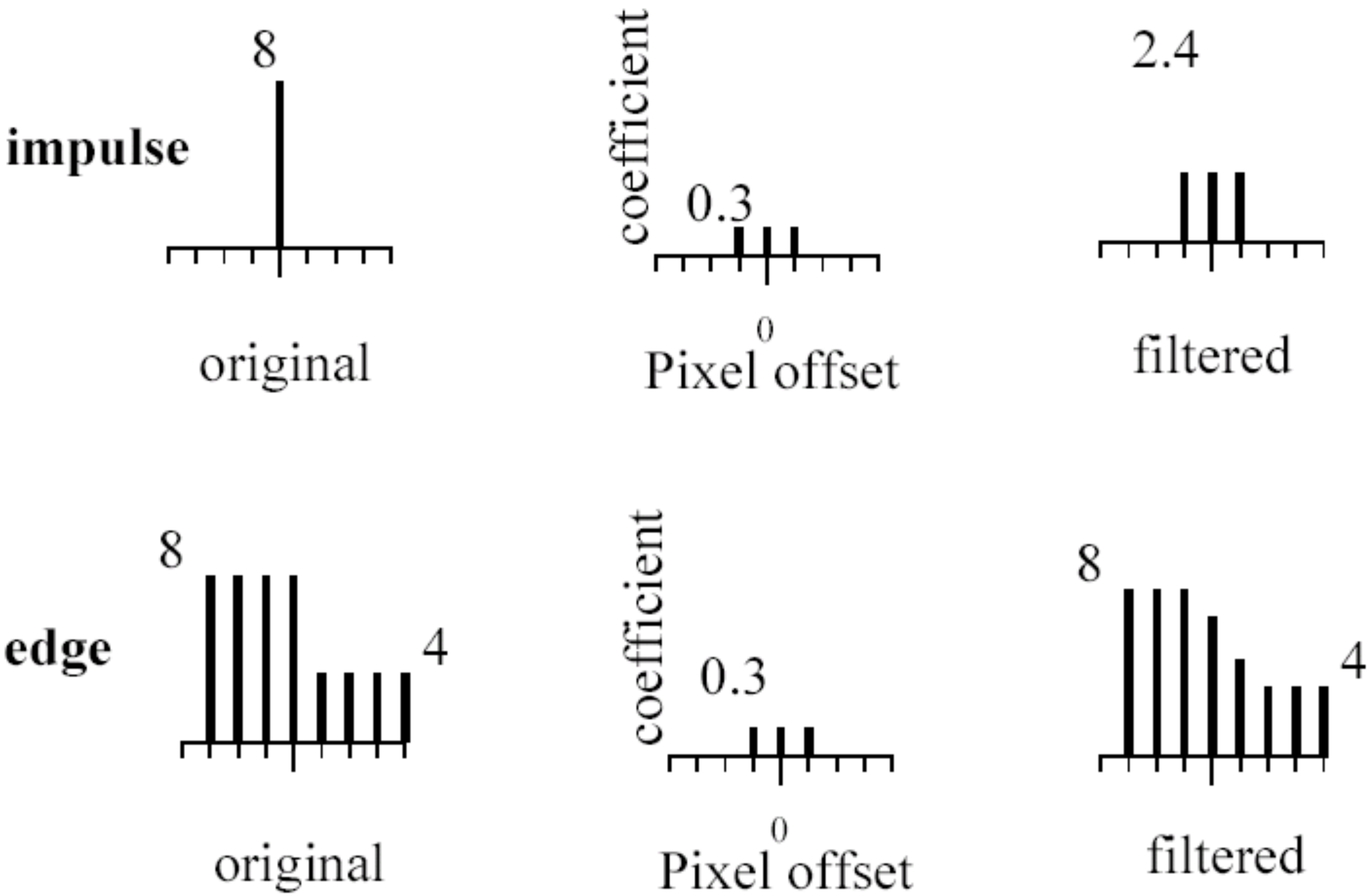
Source: B. Freeman Slides

Blur examples



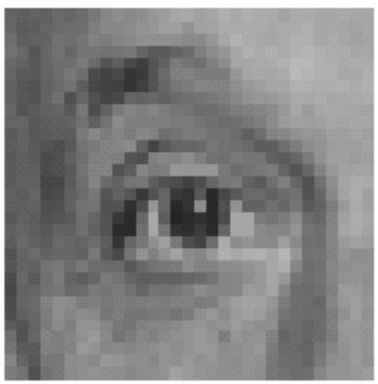
Source: B. Freeman Slides

Blur examples

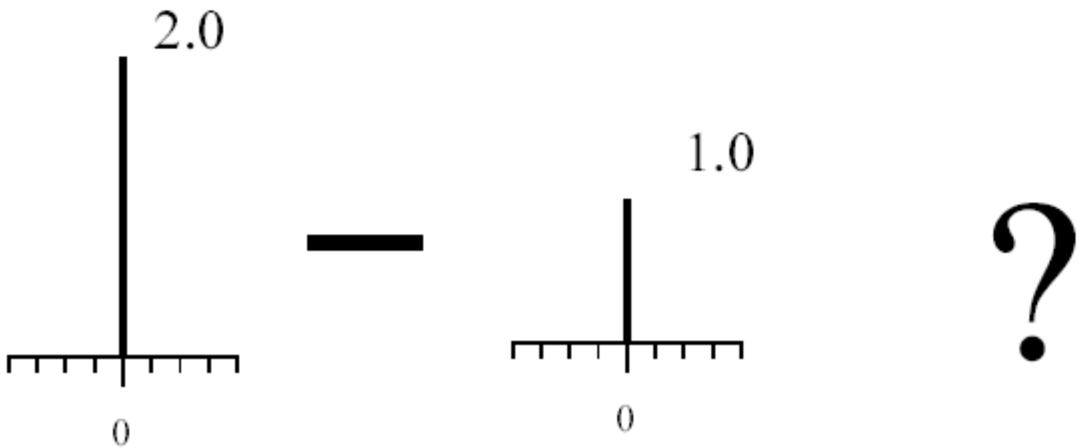


Source: B. Freeman Slides

Linear filtering (warm-up slide)

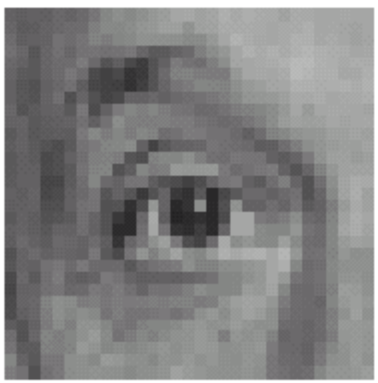


original

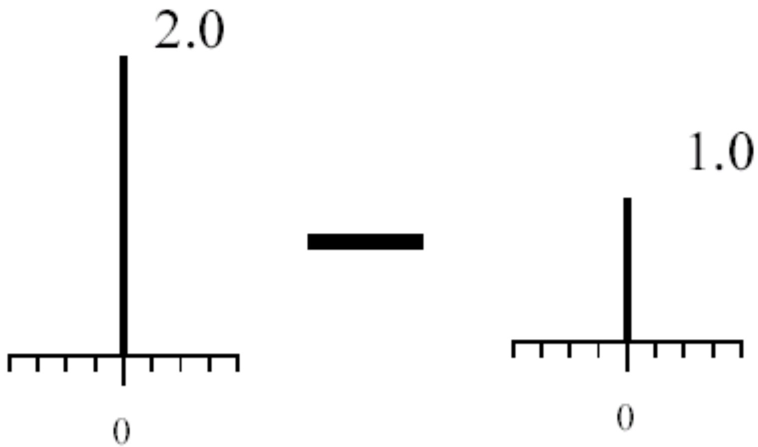


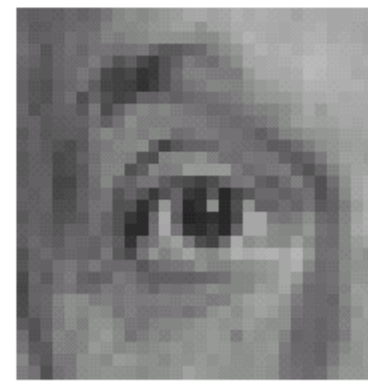
Source: B. Freeman Slides

Linear filtering (no change)



original

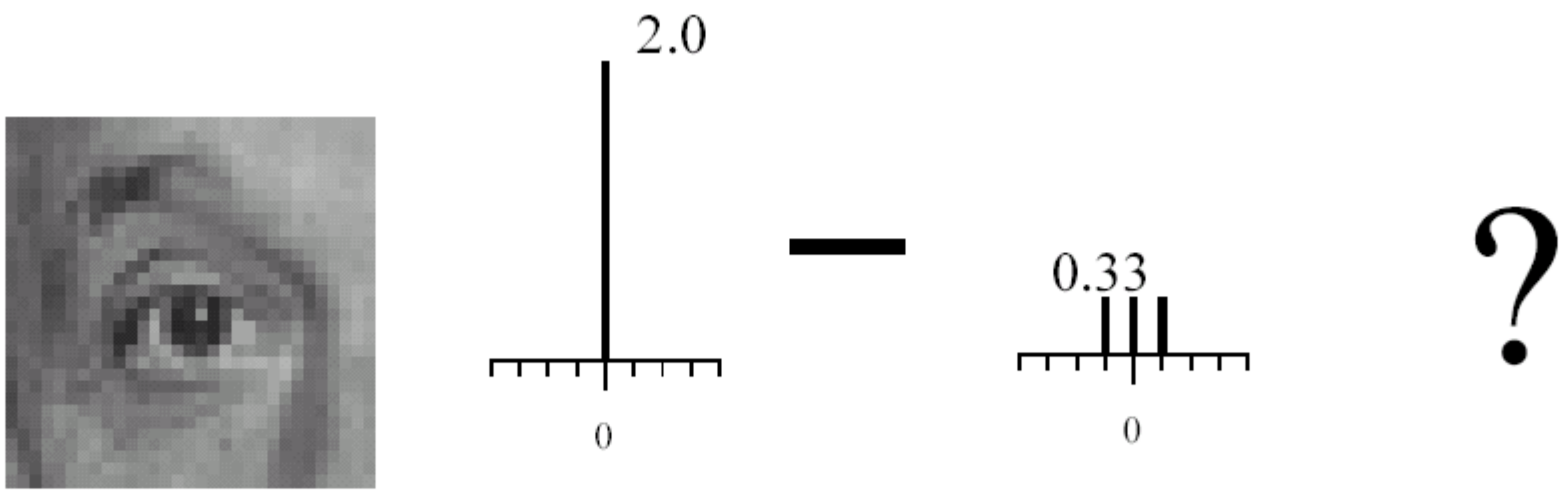




Filtered
(no change)

Source: B. Freeman Slides

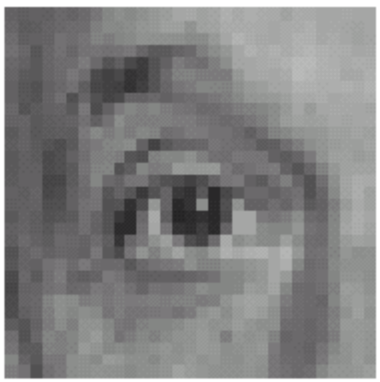
Linear filtering



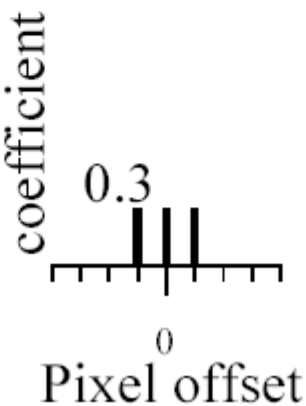
original

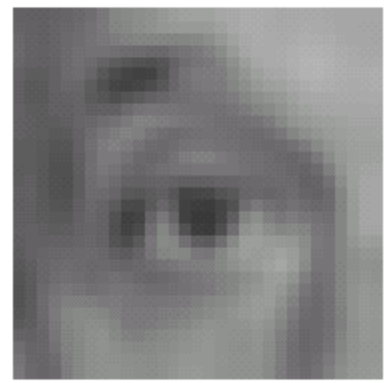
Source: B. Freeman Slides

(remember blurring)



original

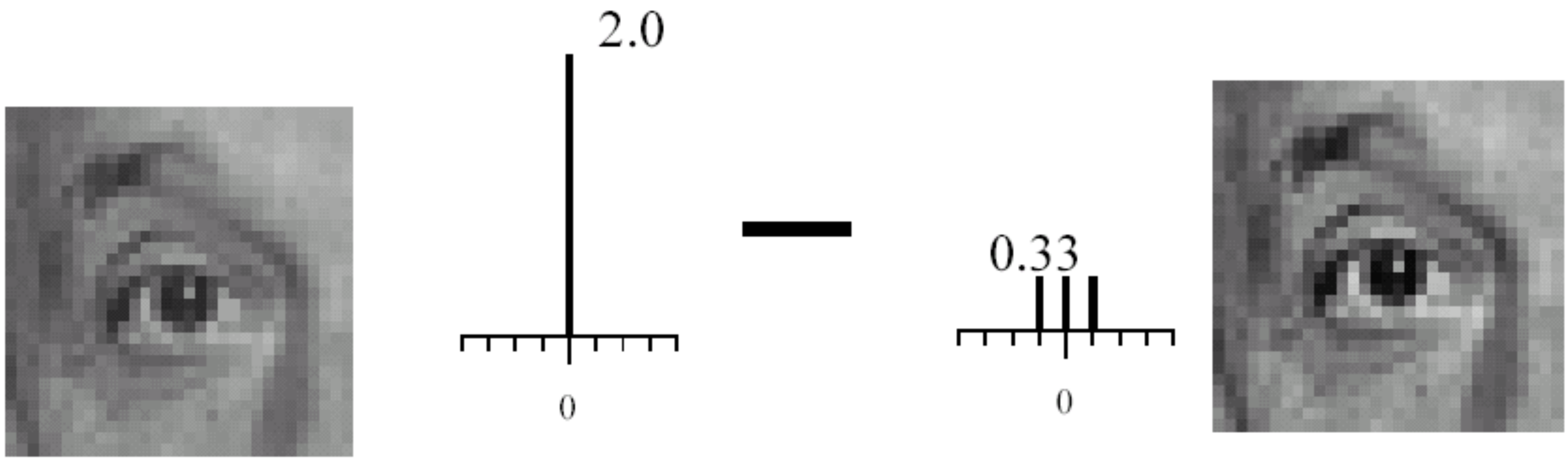




Blurred (filter applied in both dimensions).

Source: B. Freeman Slides

Sharpening

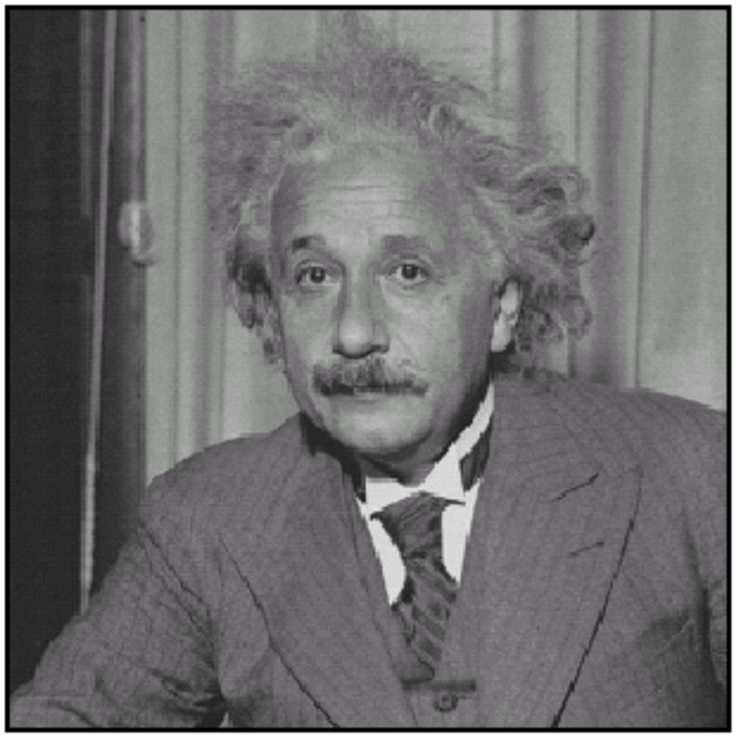


original

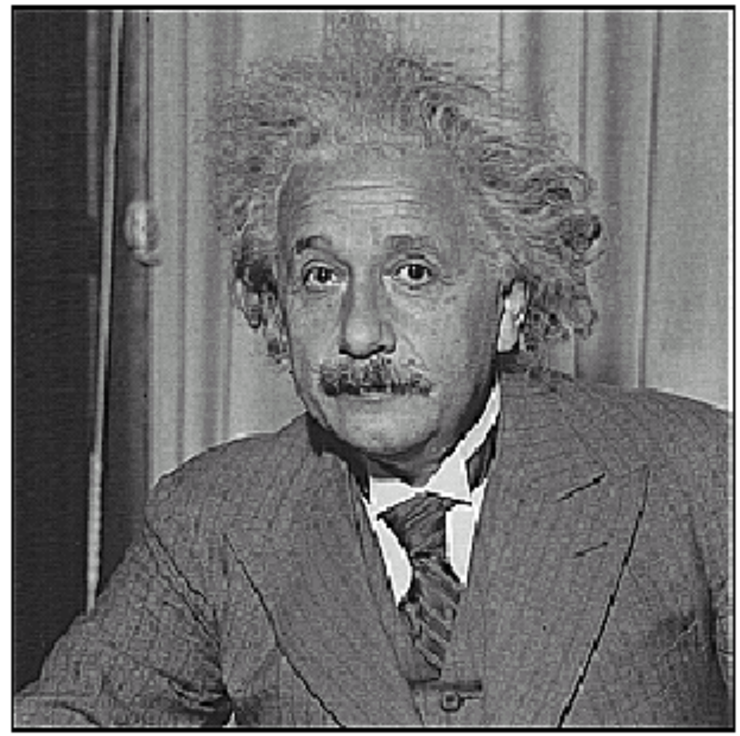
Sharpened original

Source: B. Freeman Slides

Sharpening



before



after

Source: B. Freeman Slides

What does blurring take away?

–

=

- Let’s add it back:

+ a

=

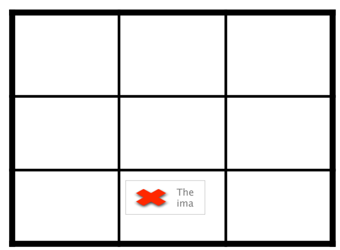
Source: Savarese Slides

Image gradient

- How can we differentiate a *digital* image $F(x, y)$?
  - Option 1: reconstruct a continuous image, f, then take gradient
  - Option 2: take discrete derivative (finite difference)

$$\frac{\partial f}{\partial x}[x, y] \approx F[x+1, y] - F[x, y]$$

How would you implement this as a cross-correlation?



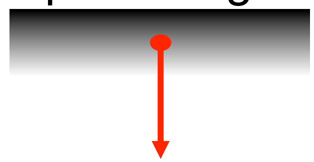
Source: Seitz and Szeliski Slides

Image gradient

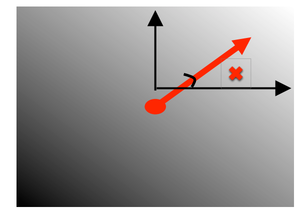
$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

It points in the direction of most rapid change in intensity

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



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



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

The gradient direction is given by:

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

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

The *edge strength* is given by the gradient magnitude

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

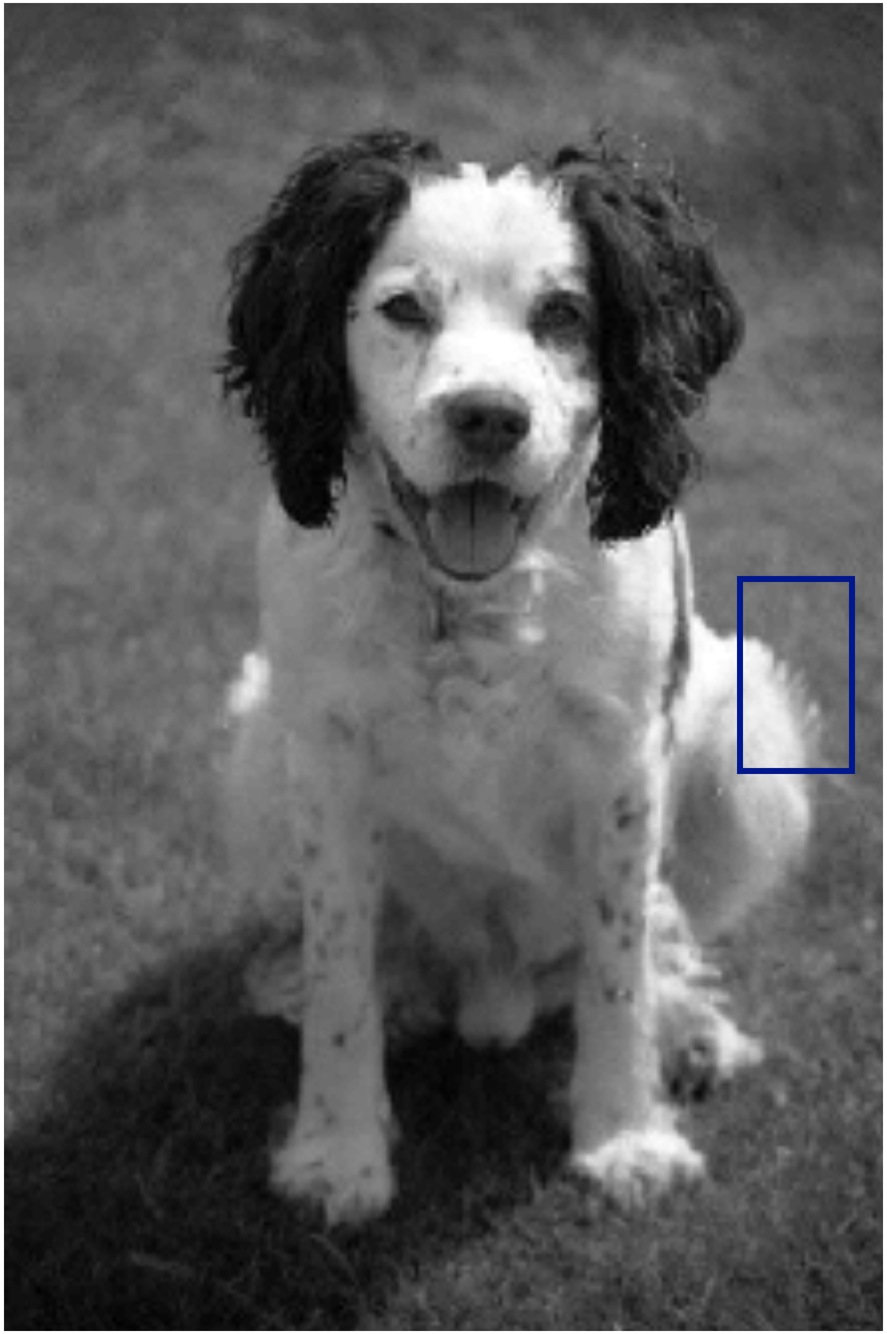
Source: Seitz and Szeliski Slides

Physical causes of edges

1. Object boundaries
2. Surface normal discontinuities
3. Reflectance (albedo) discontinuities
4. Lighting discontinuities

Source: G Hager Slides

Object Boundaries

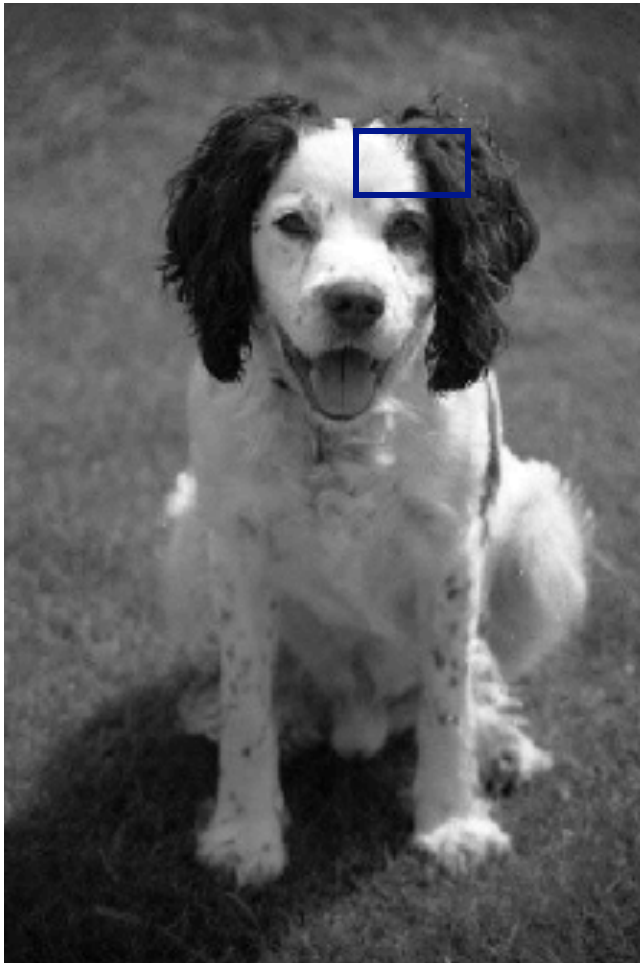


Source: G Hager Slides

Surface normal discontinuities

Source: G Hager Slides

Boundaries of material properties

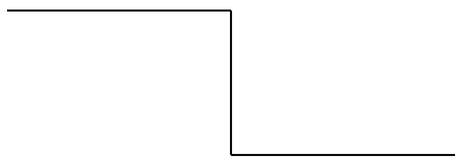


Source: G Hager Slides

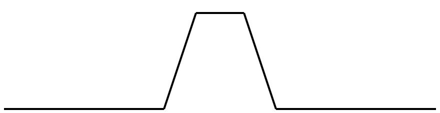
Boundaries of lighting

Source: G Hager Slides

Edge Types

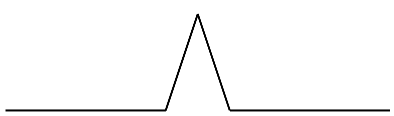


Step



Ridge

Which of these do you suppose a derivative filter detects best?



Roof

Source: G Hager Slides

Some Other Interesting Kernels

The Roberts Operator

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

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

The Prewitt Operator

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

Source: G Hager Slides

Some Other Interesting Kernals

The Sobel Operator
$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

The Laplacian Operator
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} or \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

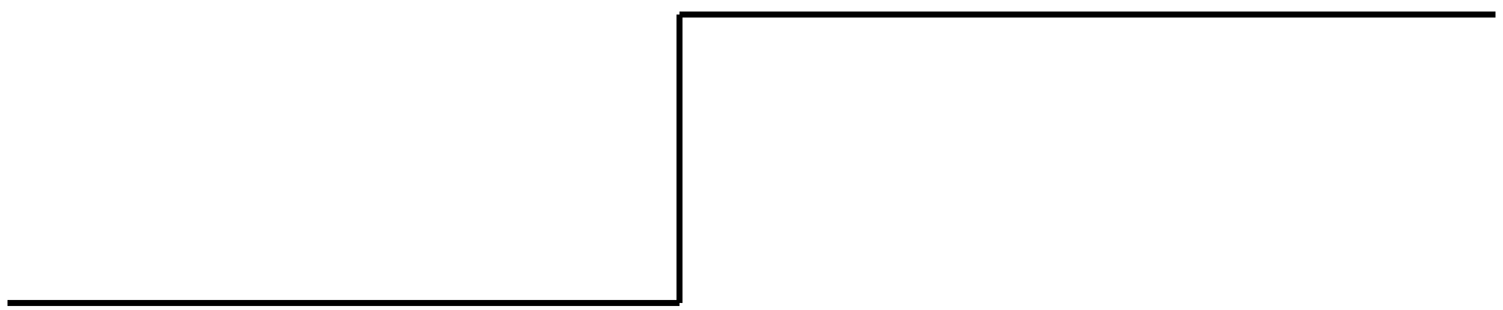
A good exercise: derive the Laplacian from 1-D derivative filters.

Note the Laplacian is rotationally symmetric!

Source: G Hager Slides

Edge is Where Change Occurs 1D

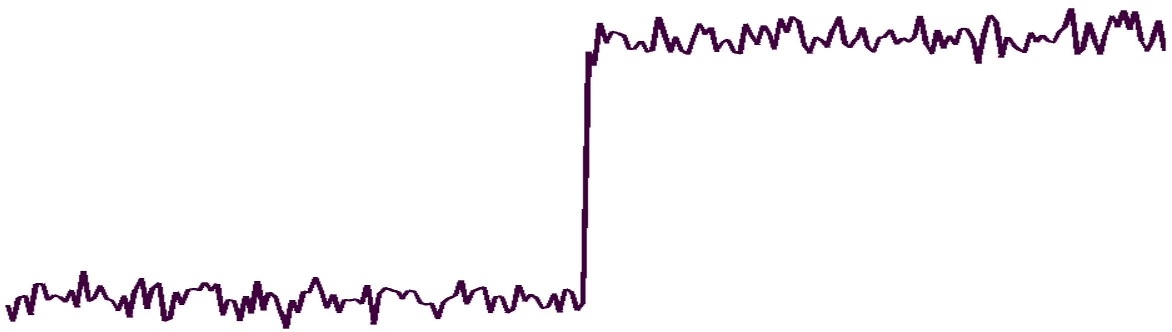
- Change is measured by derivative in 1D
  - Biggest change, derivative has maximum magnitude
  - Or 2nd derivative is zero.



Source: G Hager Slides

Noisy Step Edge

- Derivative is high everywhere.
- Must smooth before taking gradient.



Source: G Hager Slides

Smoothing Plus Derivatives

- One problem with differences is that they by definition reduce the signal to noise ratio.

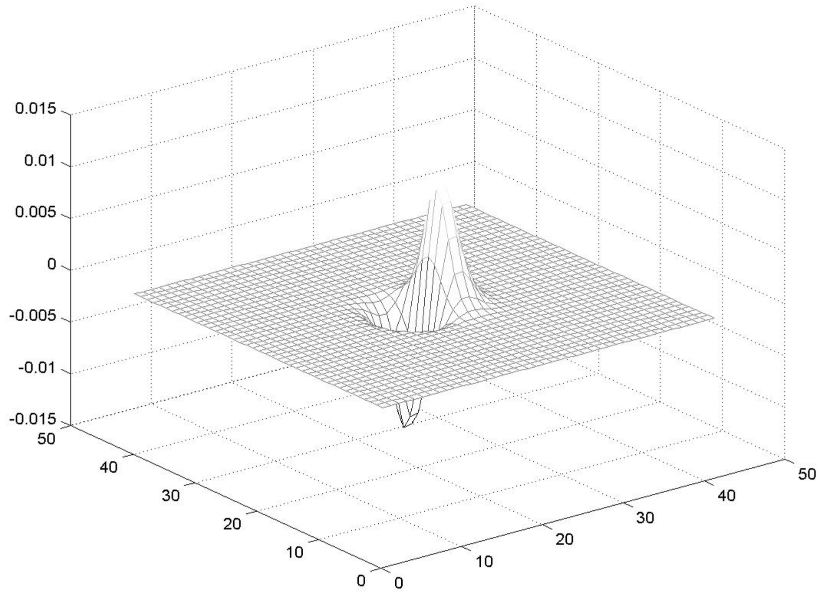
- Recall smoothing operators (the Gaussian!) reduce noise.

- Hence, an obvious way of getting clean images with derivatives is to combine derivative filtering and smoothing: e.g.

$$(F * G) * D_x = F * (G * D_x)$$

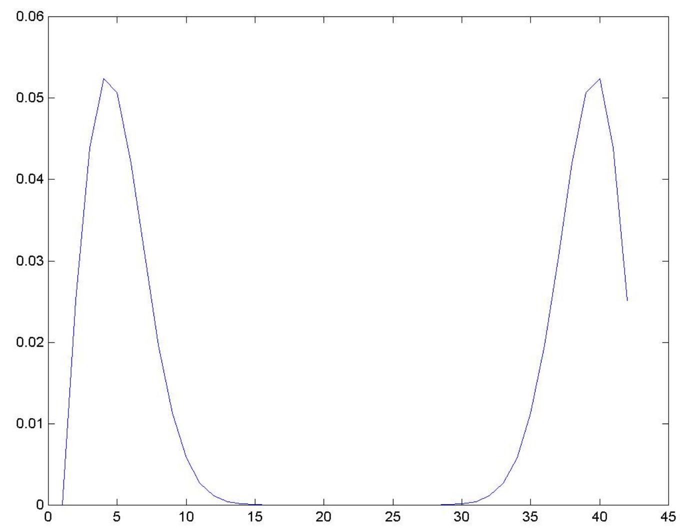
Source: G Hager Slides

The Fourier Spectrum of DOG



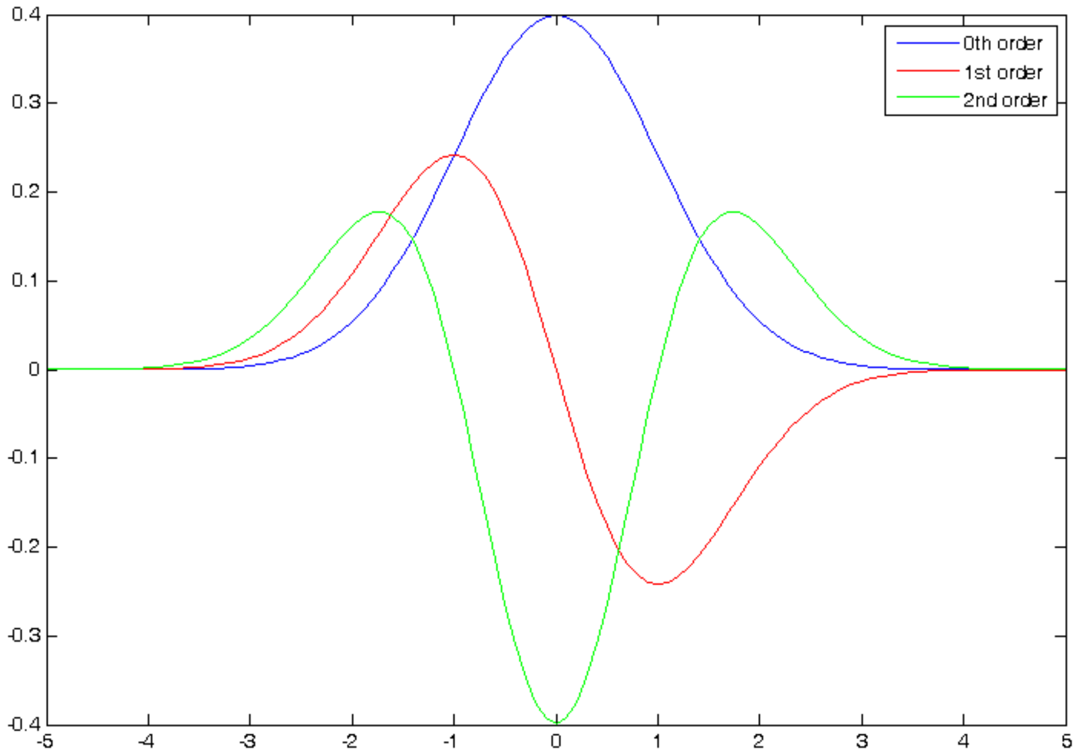
Derivative of a Gaussian

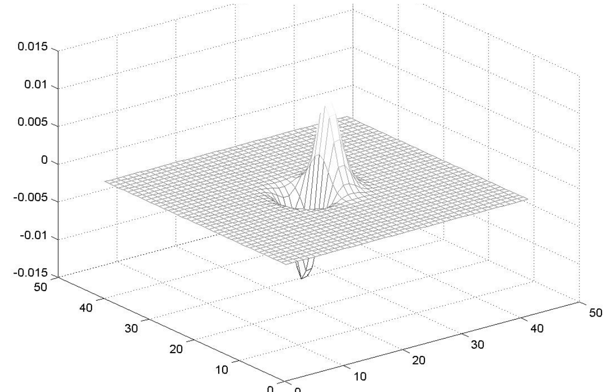
PS of central slice



Source: G Hager Slides

The DoG: Derivative of a Gaussian





Properties of the DoG operator

- Now, going back to the directional derivative:
  - $D_u(f(x,y)) = f_x(x,y)u_1 + f_y(x,y)u_2$
- Now, including a Gaussian convolution, we see
  - $D_u[G*I] = D_u[G]*I = [u_1G_x + u_2G_y]*I = u_1G_y*I + u_2G_x*I$
- The two components $I*G_x$ and $I*G_y$ are the *image gradient*
- Note the directional derivative is maximized in the direction of the gradient
- (note some authors use DoG as “Difference of Gaussian” which we’ll run into soon ....)

Source: G Hager Slides

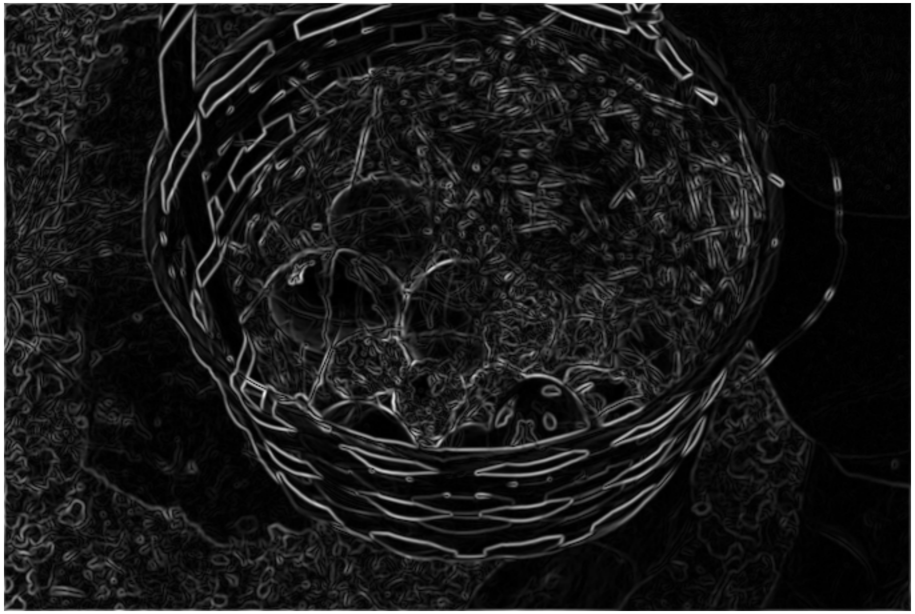
Algorithm: Simple Edge Detection

1. Compute $I_x = I_g * (G(\sigma) * G(\sigma)' * [1,-1;1,-1])$
2. Compute $I_y = I_g * (G(\sigma) * G(\sigma)' * [1,-1;1,-1]')$
3. Compute $I_{mag} = \mathrm{sqrt}(I_x .* I_x + I_y .* I_y)$
4. Threshold: $I_{res} = I_{mag} > \tau$

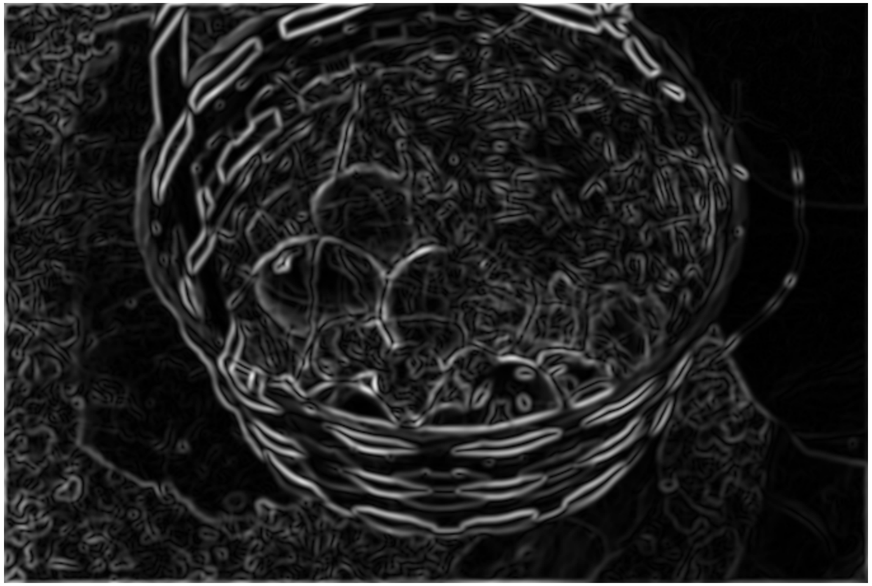
It is interesting to note that if we wanted an edge detector for a specific direction of edges, we can simply choose the appropriate projection (weighting) of the component derivatives.

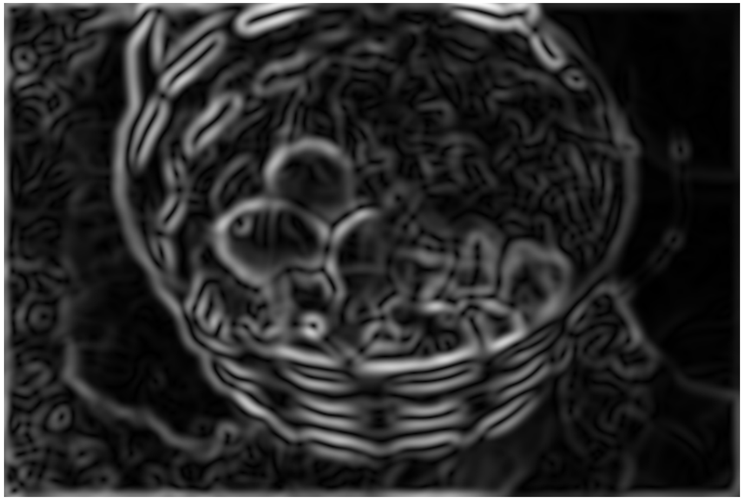
Example

sigma = 1



sigma = 2





sigma = 5

Source: G Hager Slides

Limitations of Linear Operators on Impulsive Noise

Source: G Hager Slides

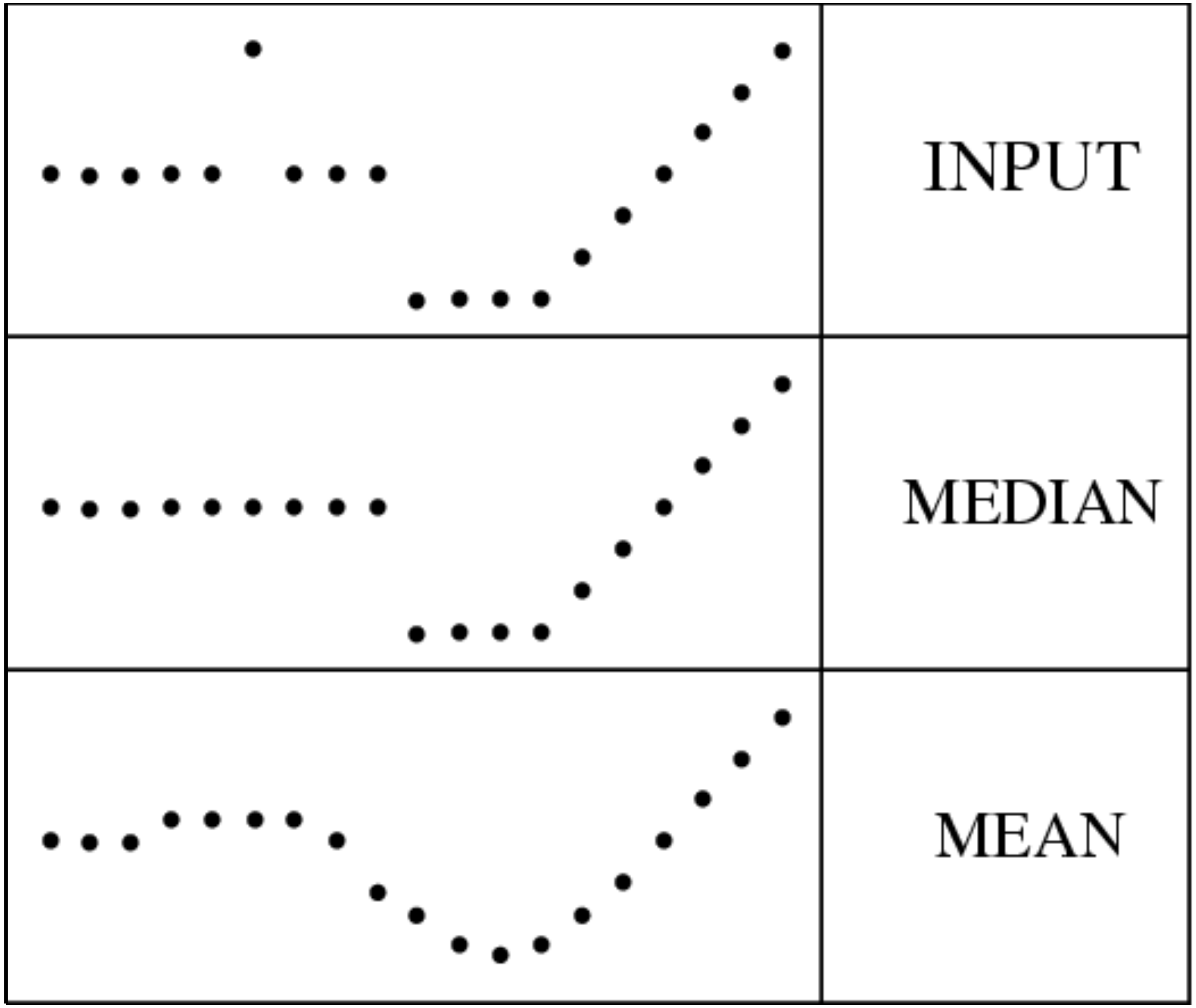
Nonlinear Filtering: The Median Filter

Suppose I look at the local statistics and replace each pixel with the *median* of its neighbors:

Source: G Hager Slides

Median Filtering Example

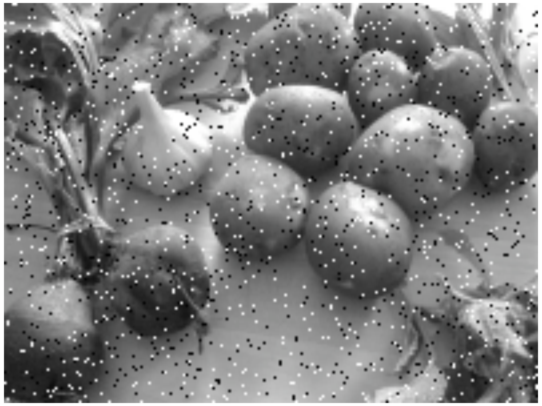
filters have width 5 :



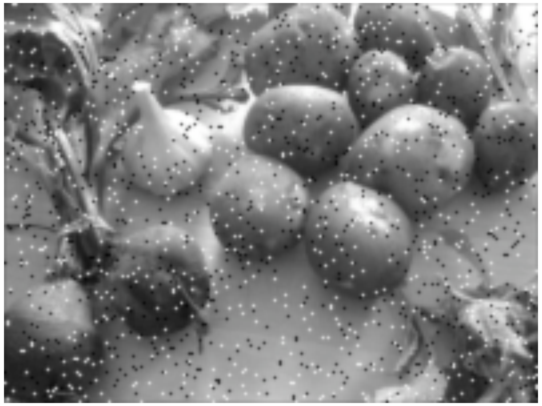
Source: G Hager Slides

Median Filtering: Example

Original



Salt and Pepper



Gaussian Filter

Median Filter

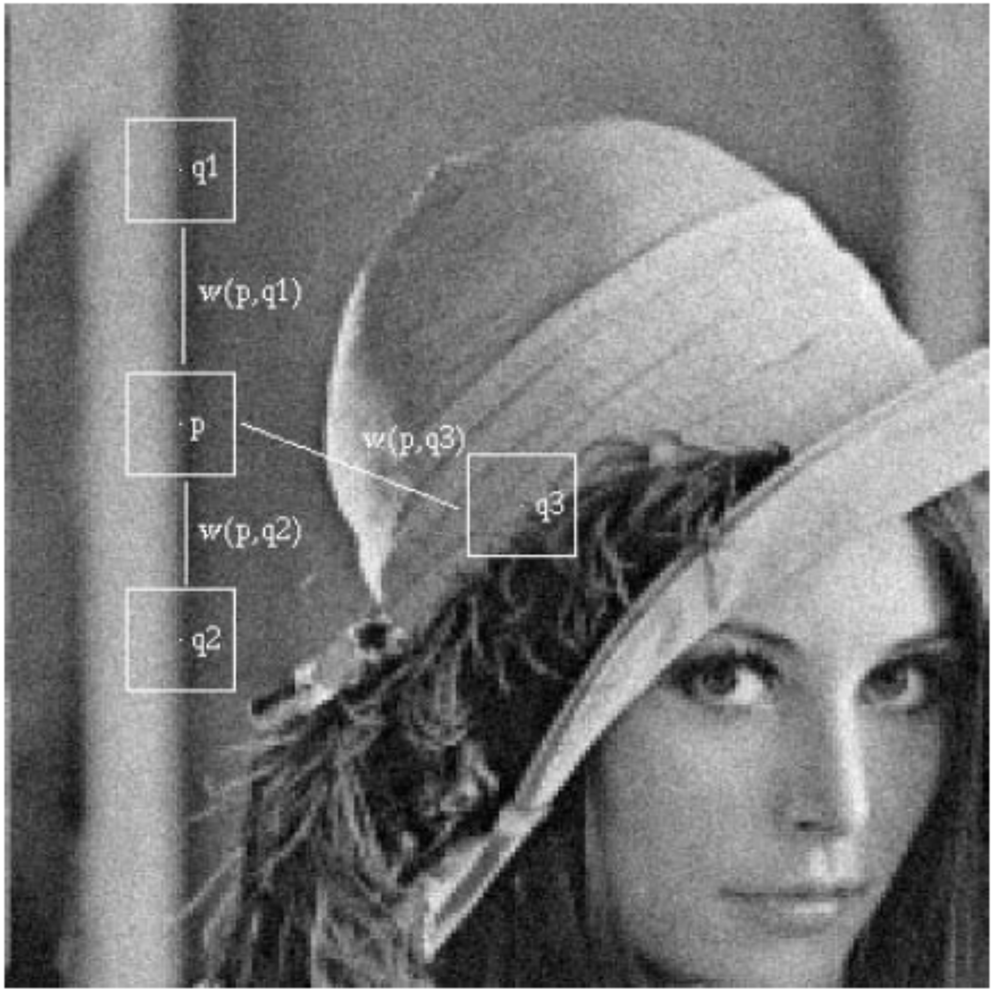
Source: G Hager Slides

Non-local Means for Image Denoising

$$S(i) = \sum_j w(i,j) v(j)$$

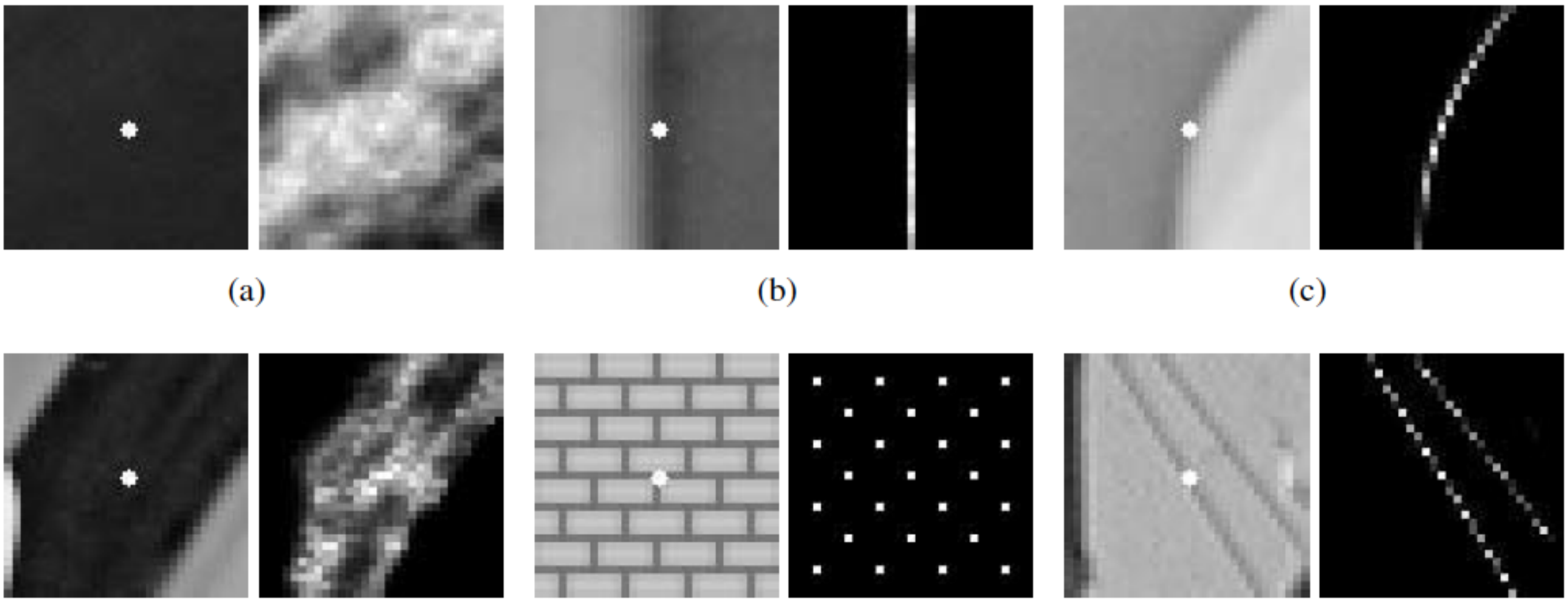
Similarity Between Two Locations

Typically, the Euclidean distance in a Gaussian kernel.



Paper/Source: Buades, Coll, Morel. “A non-local algorithm for image denoising” CVPR 2005.

NL Means Weight Distribution



(a) (b) (c)

(d) (e) (f)

Paper/Source: Buades, Coll, Morel. “A non-local algorithm for image denoising” CVPR 2005.

NL Means Example Result

Paper/Source: Buades, Coll, Morel. “A non-local algorithm for image denoising” CVPR 2005.

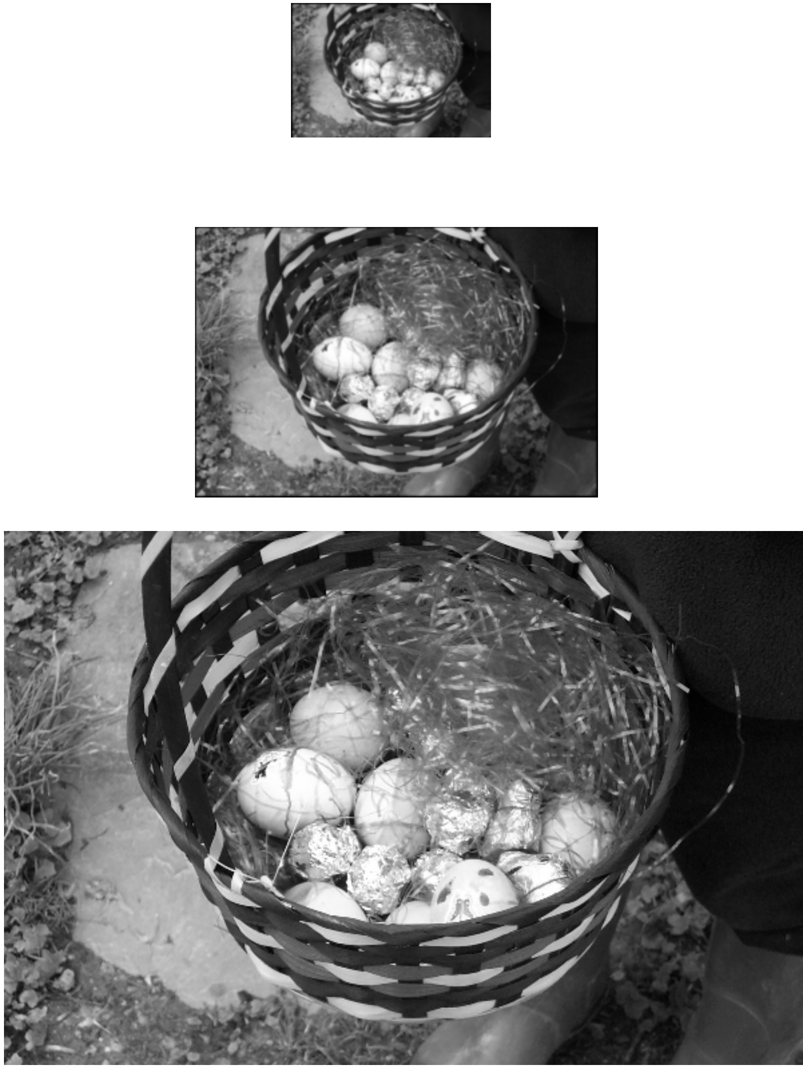
Filter Pyramids

- Recall we can always filter with $\mathcal{G}(\sigma)$ for any σ
- As a result, we can think of a continuum of filtered images as σ grows.
  – This is referred to as the "scale space" of the images. We will see this show up several times.
- As a related note, suppose I want to subsample images
  – Subsampling reduces the highest frequencies
  – Averaging reduces noise
  – Pyramids are a way of doing both

Source: G Hager Slides

Gaussian Pyramid

- Algorithm:
  - 1. Filter with $\mathcal{G}(\sigma = 1)$
  - 2. Resample at every other pixel
  - 3. Repeat



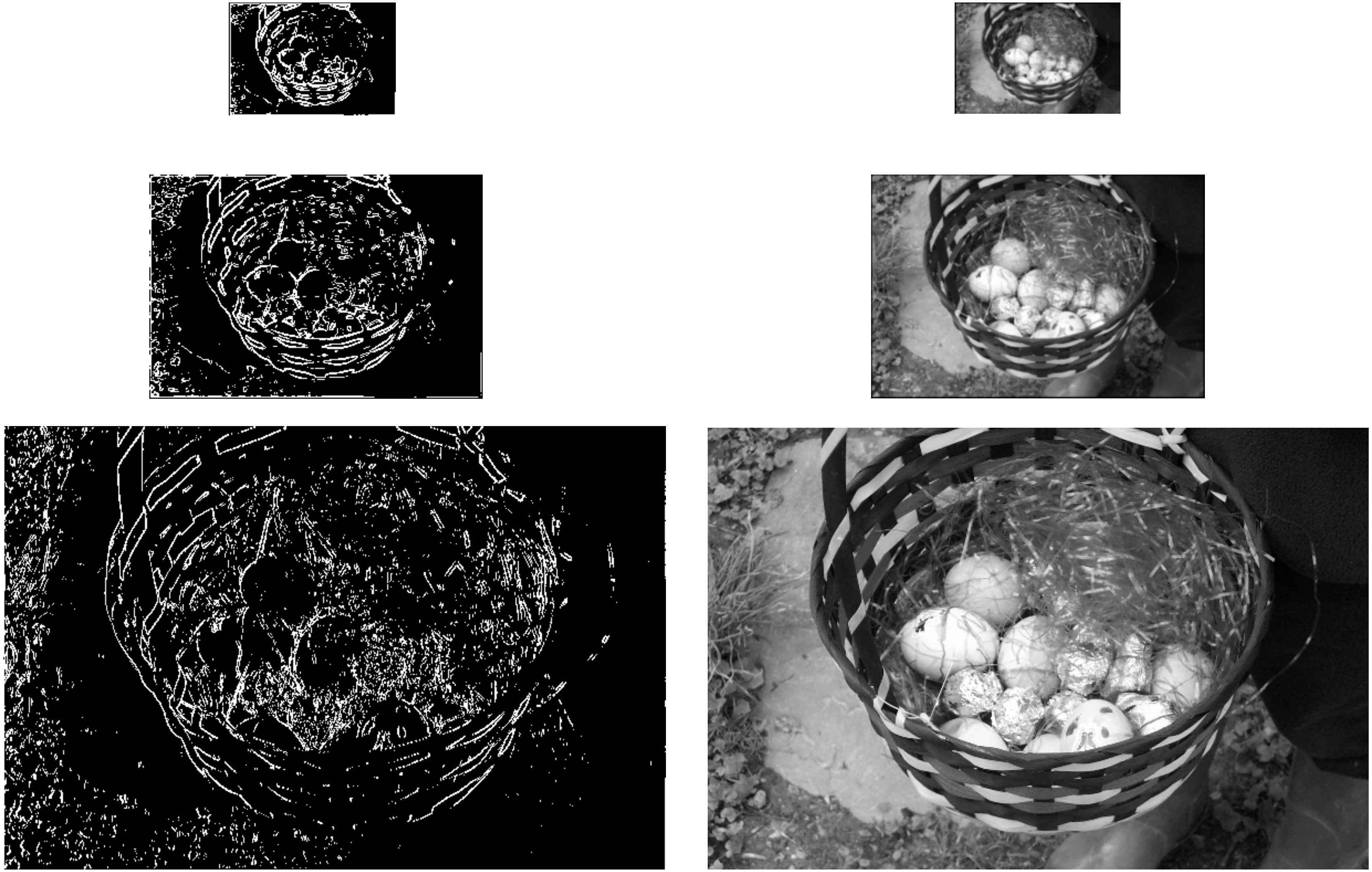
Source: G Hager Slides

Laplacian Pyramid Algorithm

- Create a Gaussian pyramid by successive smoothing with a Gaussian and down sampling
- Set the coarsest layer of the Laplacian pyramid to be the coarsest layer of the Gaussian pyramid
- For each subsequent layer n+1, compute

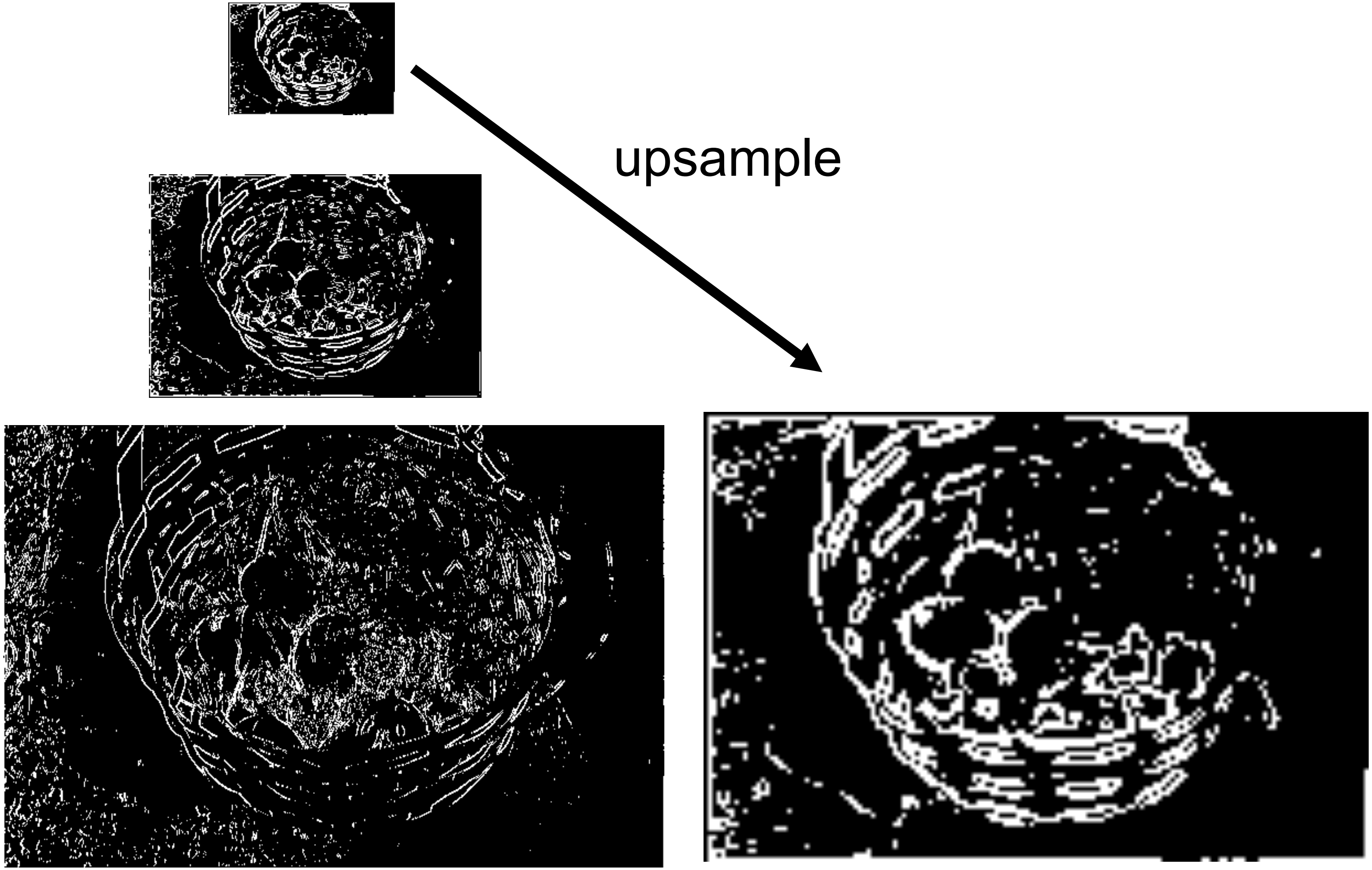
$$L(n+1) = G(n+1) = \mathrm{Upsample}(G(n))$$

Laplacian of Gaussian Pyramid



Source: G Hager Slides

Laplacian of Gaussian Pyramid



Source: G Hager Slides

Understanding Convolution

- Another way to think about convolution is in terms of how it changes the *frequency distribution* in the image.

- Recall the *Fourier* representation of a function

  - $F(u) = \int f(x)\, e^{-2\pi i u x}\, dx$
  - recall that $e^{-2\pi i u x} = \cos(2\pi u x) - i \sin(2 \pi u x)$
  - Also we have $f(x) = \int F(u)\, e^{2\pi i u x}\, du$
  - $F(u) = |F(u)|\, e^{i \Phi(u)}$
    - a decomposition into magnitude ($|F(u)|$) and phase $\Phi(u)$
    - If $F(u) = a + i b$ then
    - $|F(u)| = (a^2 + b^2)^{1/2}$ and $\Phi(u) = \text{atan2}(a,b)$

  - $|F(u)|^2$ is the *power spectrum*

- Questions: what function takes many many many terms in the Fourier expansion?

Source: G Hager, D. Kriegman Slides

Understanding Convolution

Discrete Fourier Transform (DFT)

$$F[u,v] \equiv \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} I[x,y]e^{\frac{-2\pi\ j}{N}(xu+yv)}$$

Inverse DFT

$$I[x,y] \equiv \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F[u,v]e^{\frac{+2\pi\ j}{N}(ux+vy)}$$

Implemented via the "Fast Fourier Transform" algorithm (FFT)

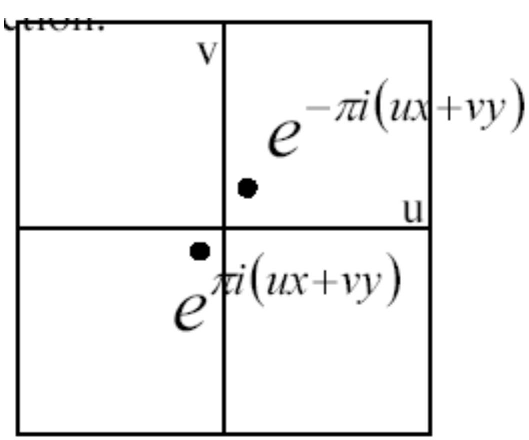
Source: G Hager, D. Kriegman Slides

Fourier basis element

$$e^{-i2\pi(ux+vy)}$$

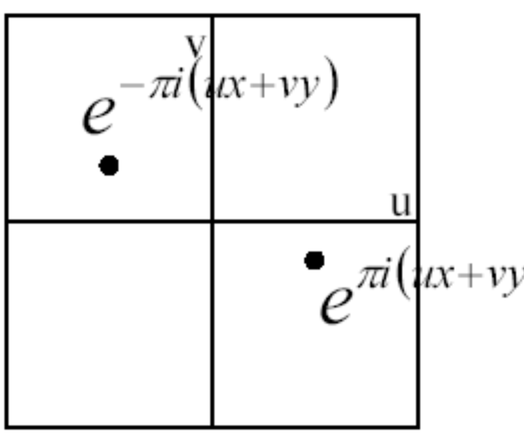
Transform is sum of orthogonal basis functions

Vector (u,v)
- Magnitude gives frequency
- Direction gives orientation.



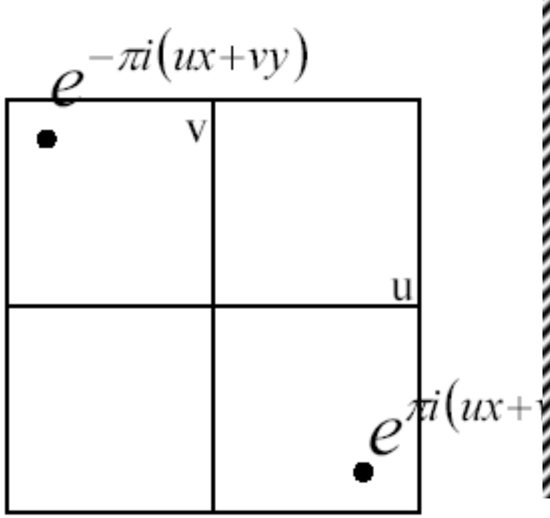
Source: G Hager, D. Kriegman Slides

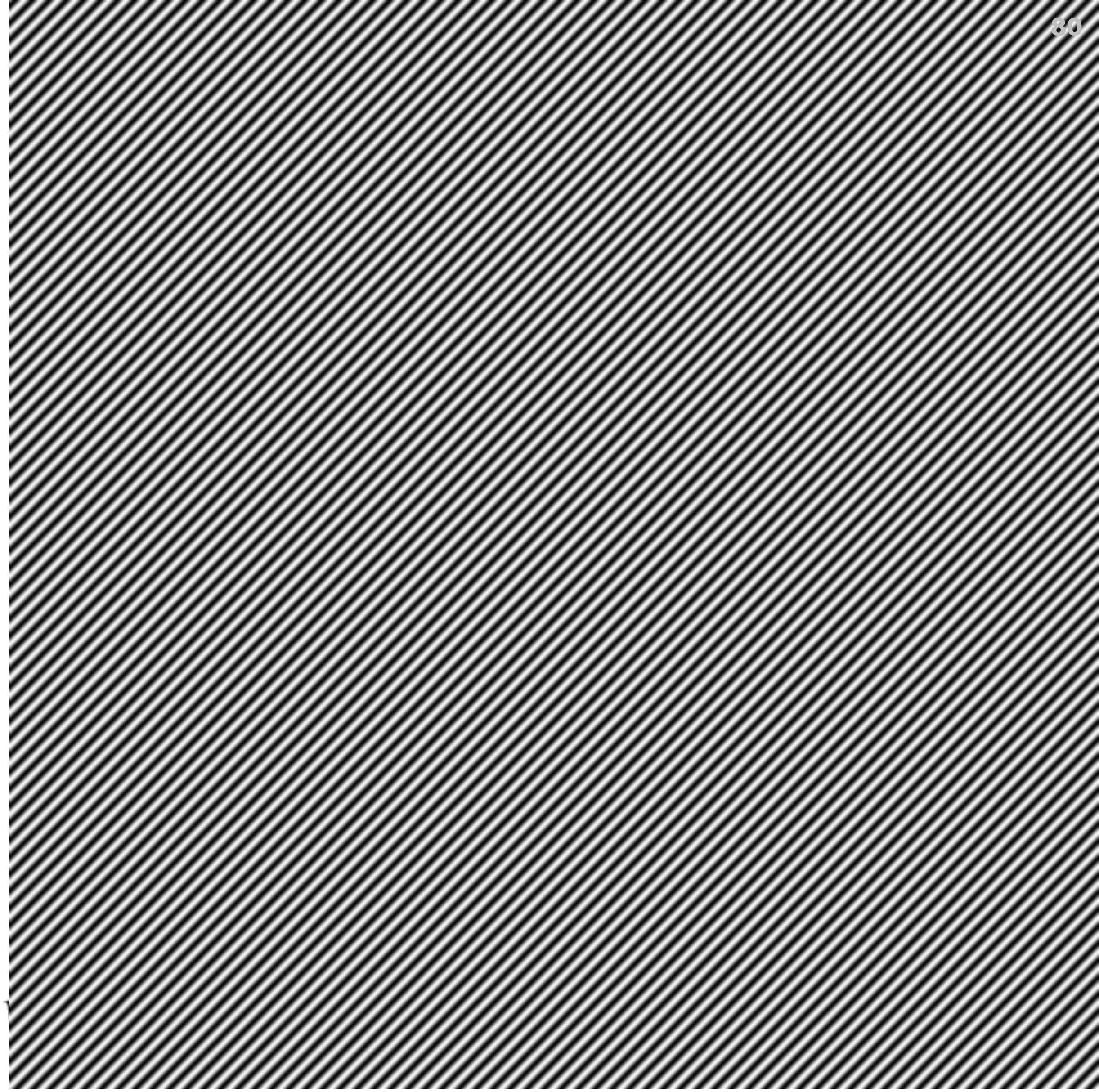
Here u and v are larger than in the previous slide.



Source: G Hager, D. Kriegman Slides

And larger still...

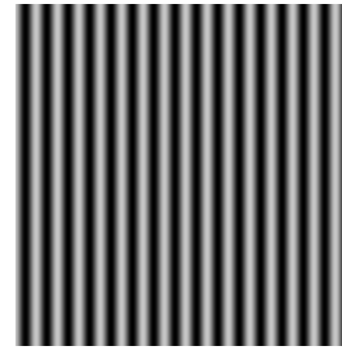




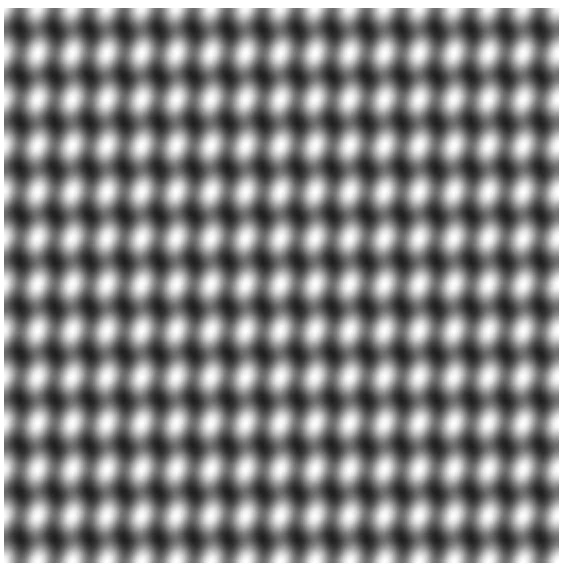
Source: G Hager, D. Kriegman Slides

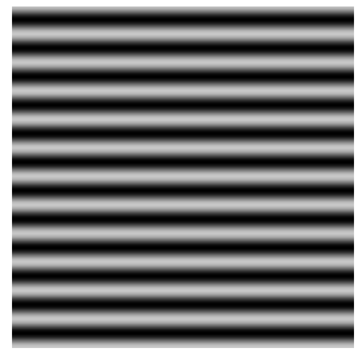
The Fourier “Hammer”

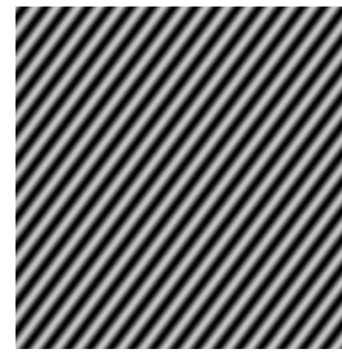
“Power Spectrum” →



Linear Combination:

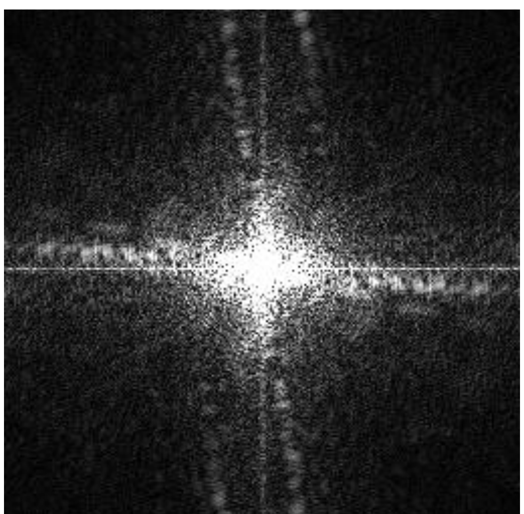




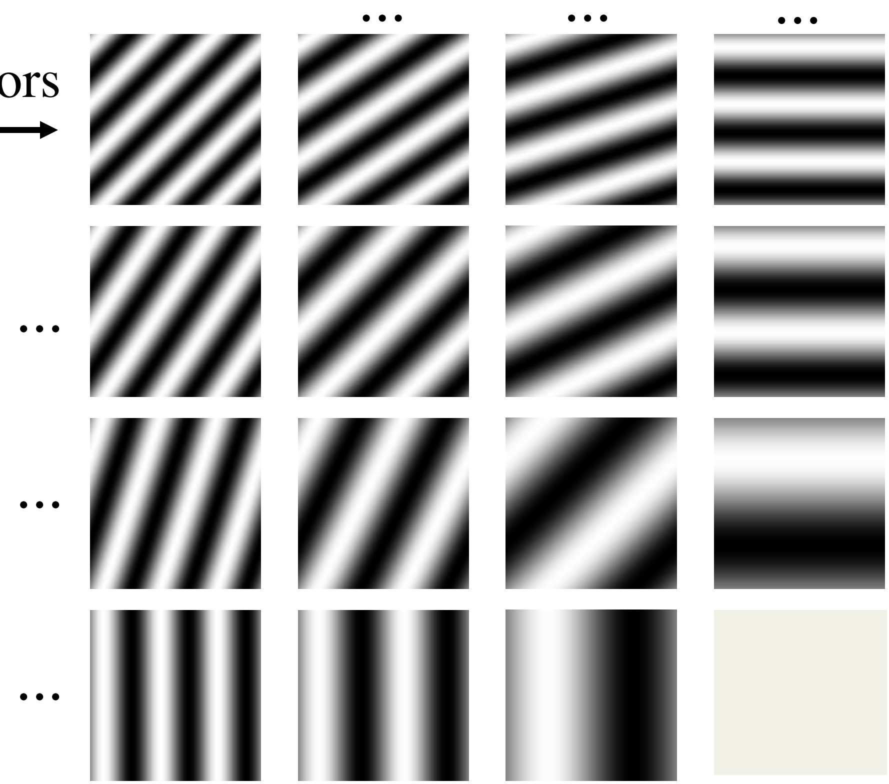


Basis vectors

Frequency Decomposition



All Basis Vectors



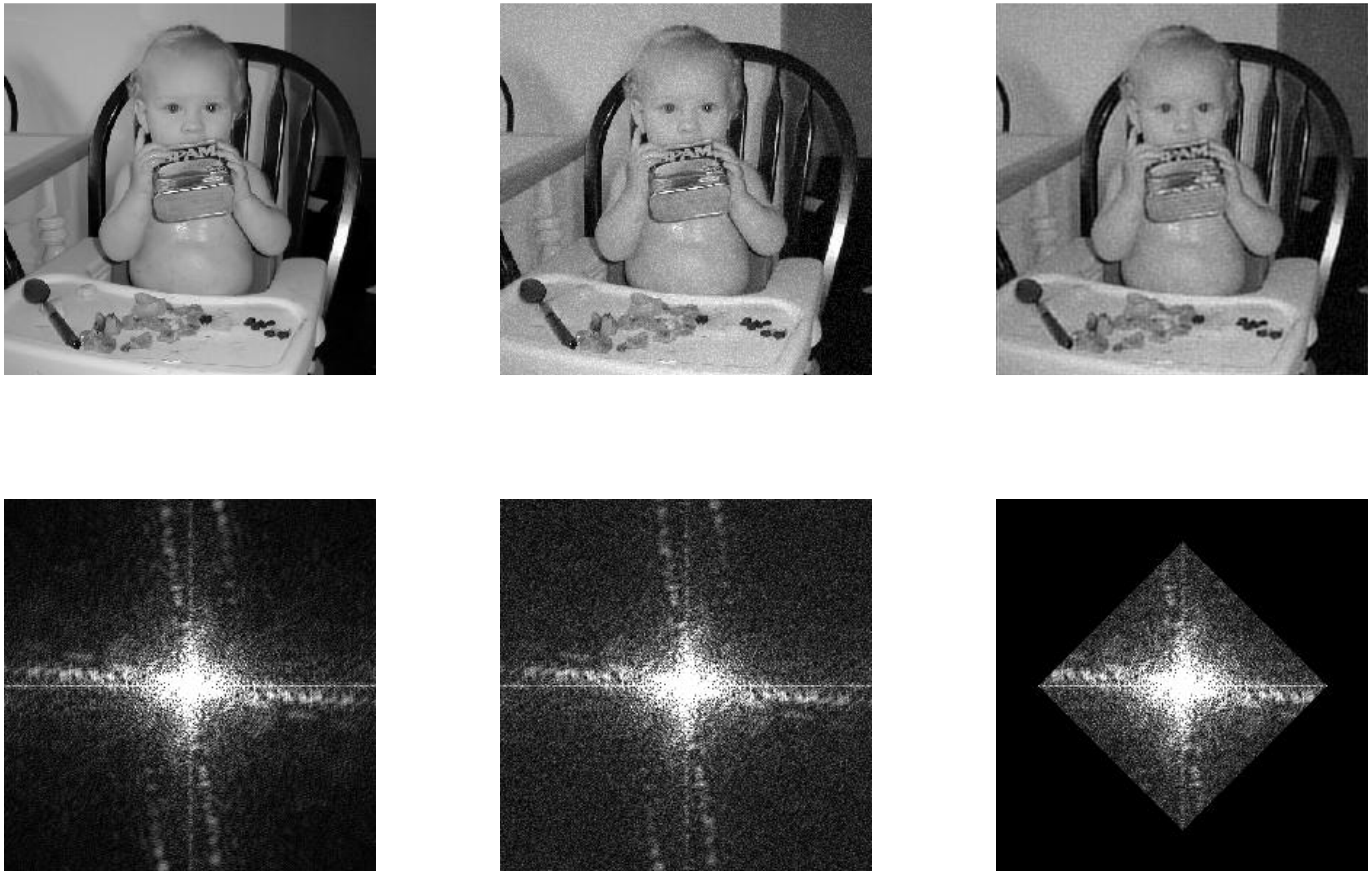
Example

intensity ~ that frequency's coefficient

Source: G Hager, D. Kriegman Slides

Using Fourier Representations

Smoothing

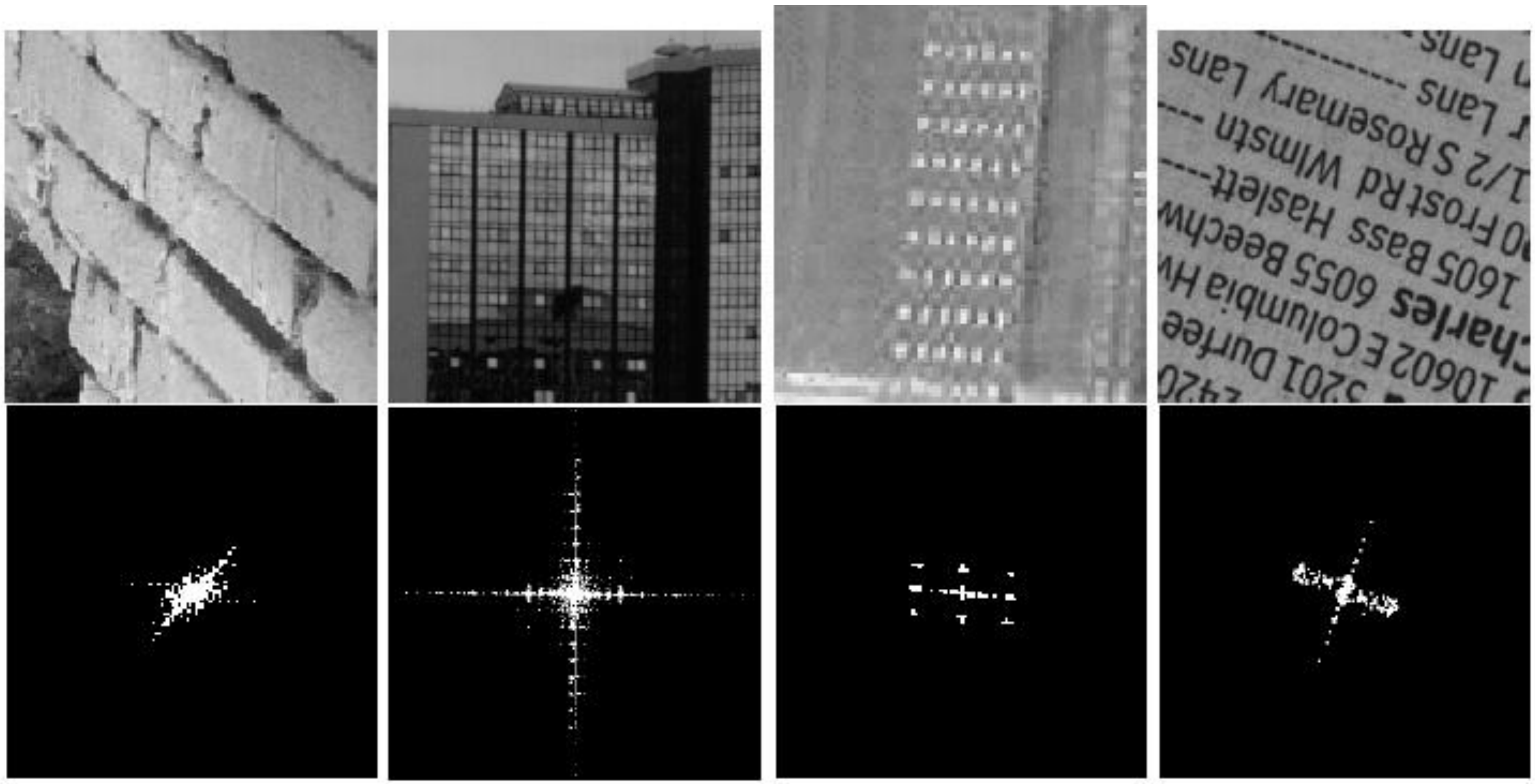


<u>Data Reduction</u>: only use *some* of the existing frequencies

Source: G Hager, D. Kriegman Slides

Using Fourier Representations

Dominant Orientation



Limitations: not useful for local segmentation

Source: G Hager, D. Kriegman Slides

Phase and Magnitude

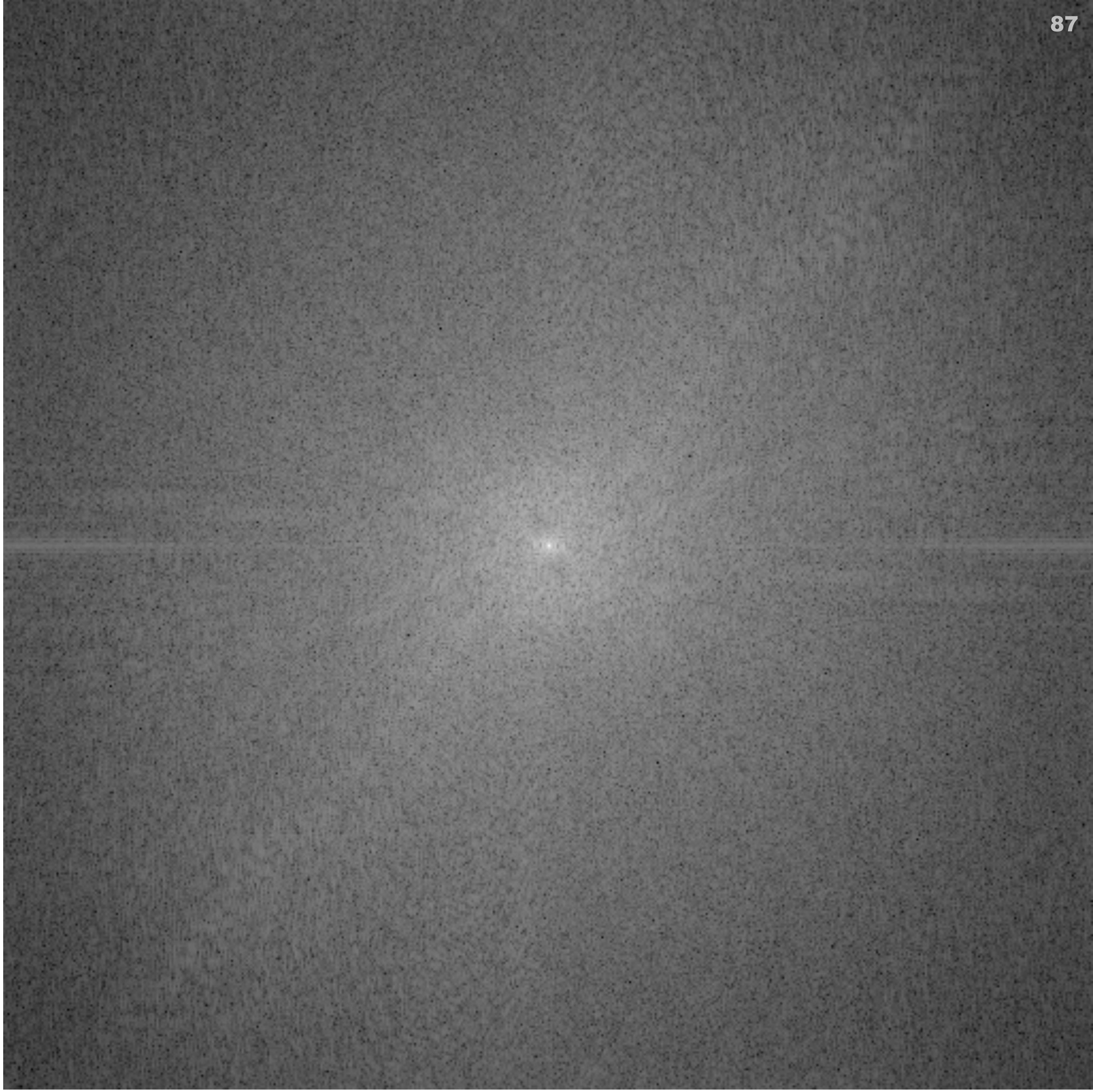
$$e^{it} = \cos t + i \sin t$$

- Fourier transform of a real function is complex with real (R) and imaginary (I) components
  - difficult to plot, visualize
  - instead, we can think of the phase and magnitude of the transform
- Phase is the phase of the complex transform
  - p(u) = atan(I(u)/R(u))
- Magnitude is the magnitude of the complex transform
  - m(u) = sqrt(R^2(u) + I^2(u))
- 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?

Source: G Hager, D. Kriegman Slides

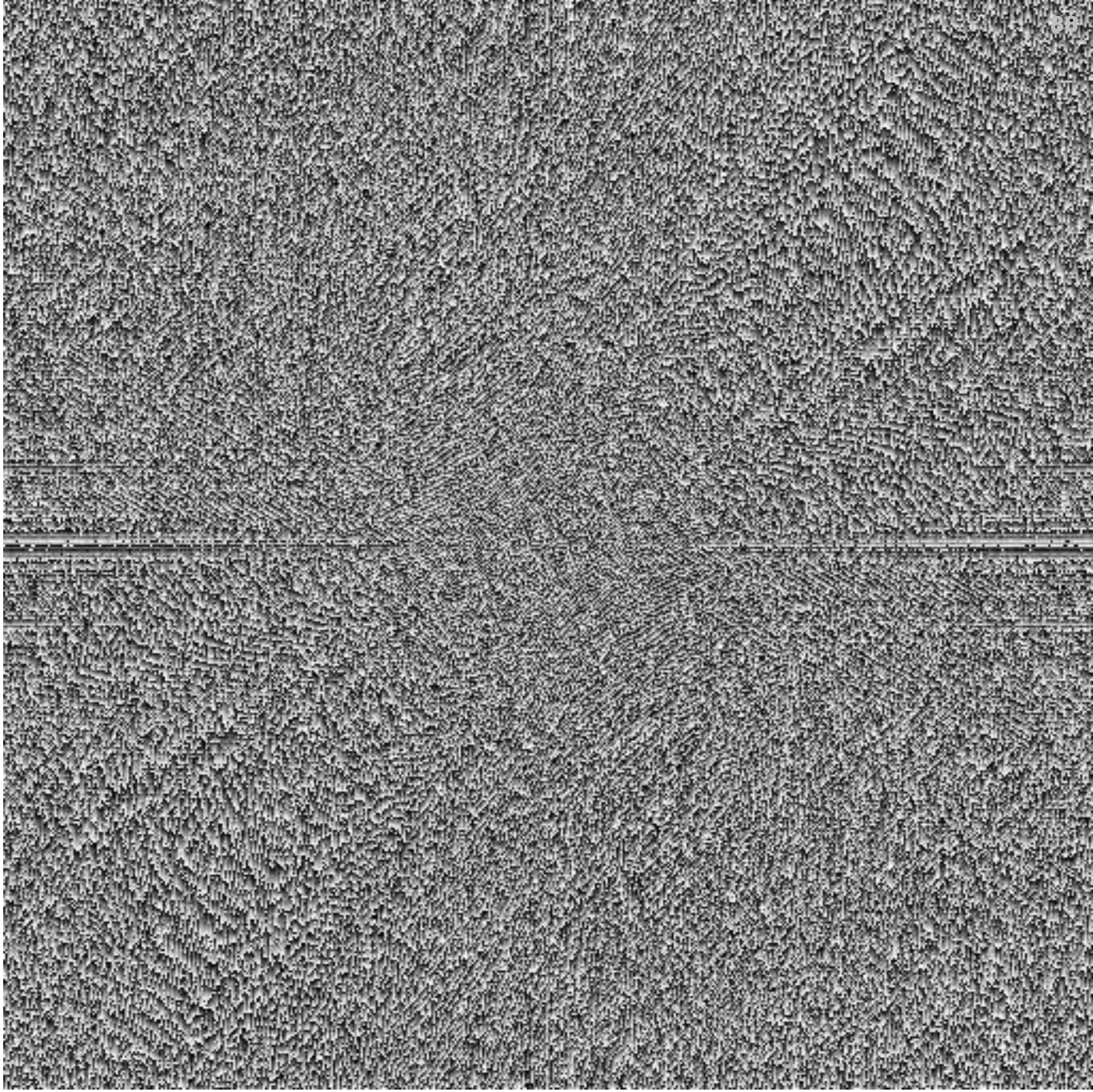
Source: G Hager, D. Kriegman Slides

This is the magnitude transform of the cheetah pic



Source: G Hager, D. Kriegman Slides

This is the phase transform of the cheetah pic



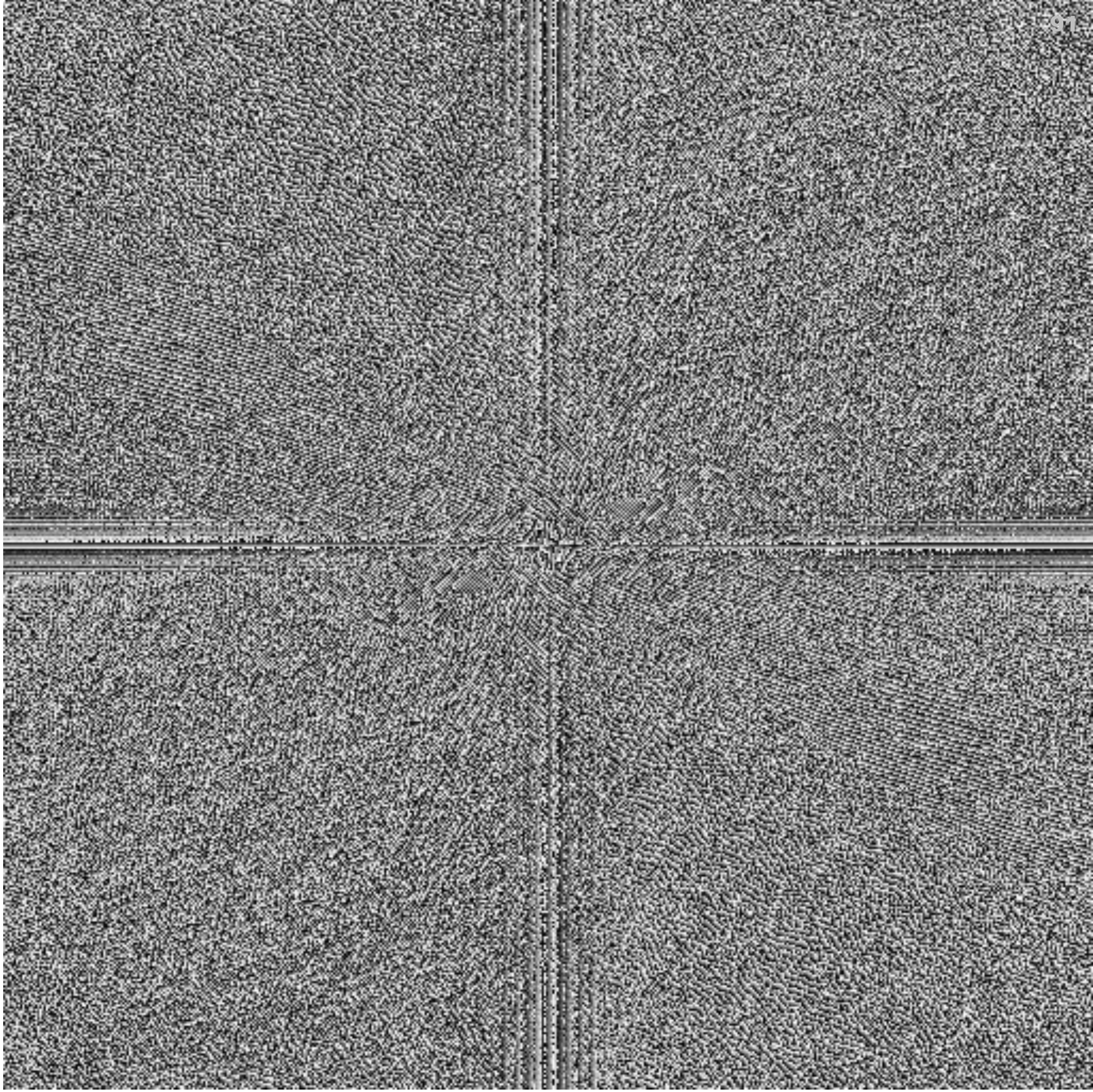
Source: G Hager, D. Kriegman Slides


Source: G Hager, D. Kriegman Slides

This is the
magnitude
transform
of the
zebra pic

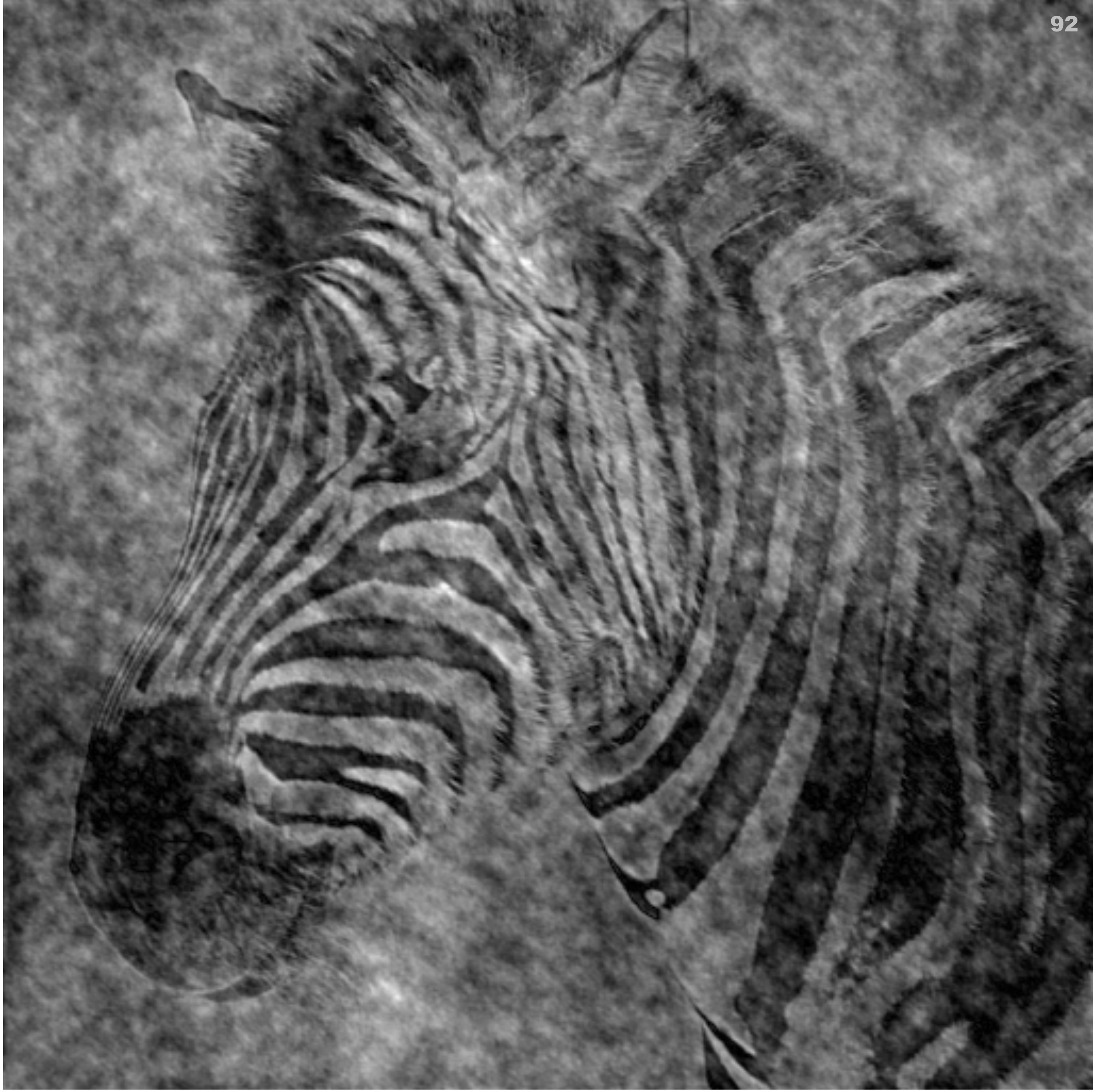
Source: G Hager, D. Kriegman Slides

This is the phase transform of the zebra pic



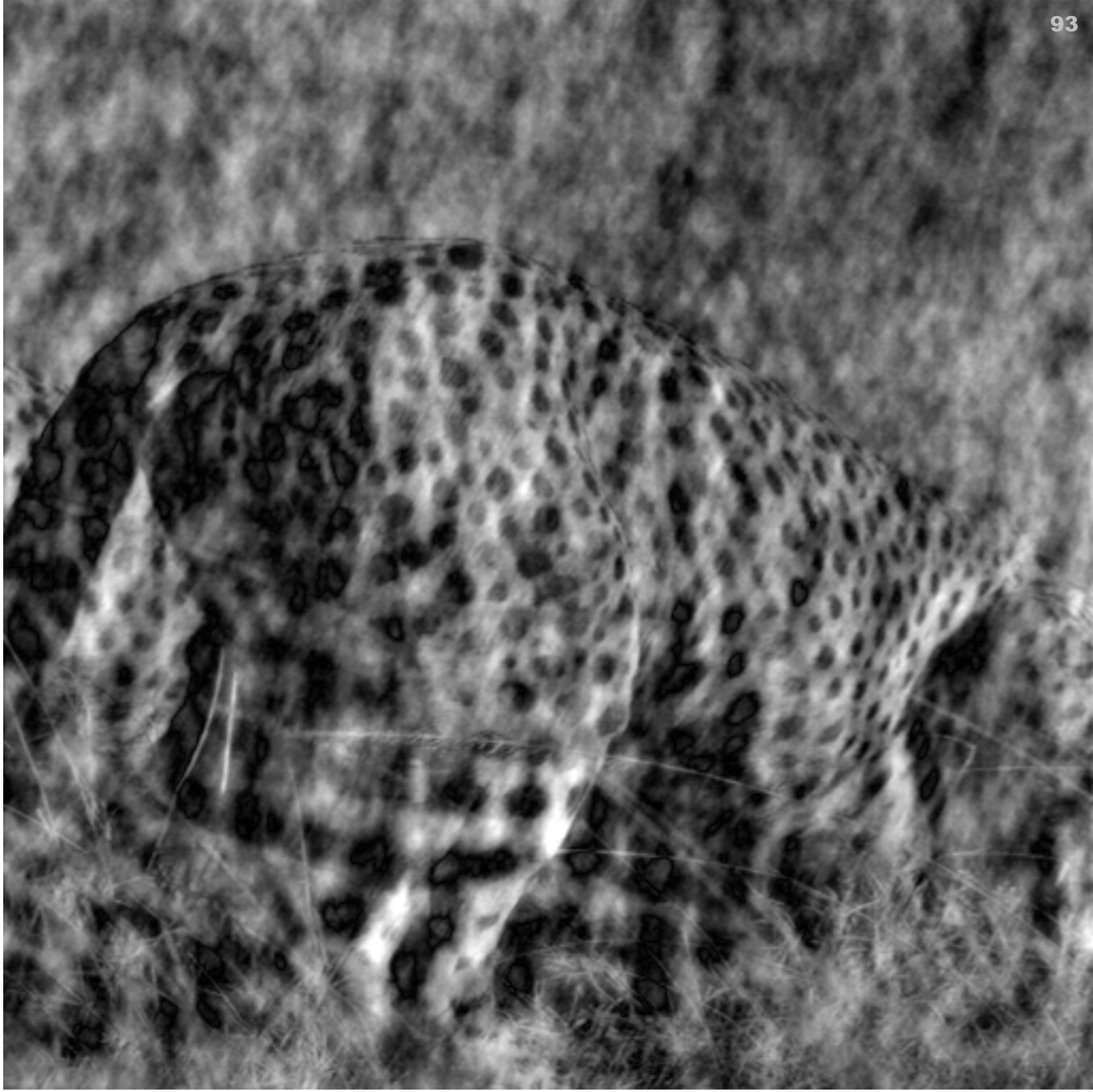
Source: G Hager, D. Kriegman Slides

Reconstruction with zebra phase, cheetah magnitude



Source: G Hager, D. Kriegman Slides

Reconstruction with cheetah phase, zebra magnitude



Source: G Hager, D. Kriegman Slides

The Fourier Transform and Convolution

- If H and G are images, and F(.) represents Fourier transform, then

$$F(H*G) = F(H)F(G)$$

- Thus, one way of thinking about the properties of a convolution is by thinking of how it modifies the frequencies of the image to which it is applied.

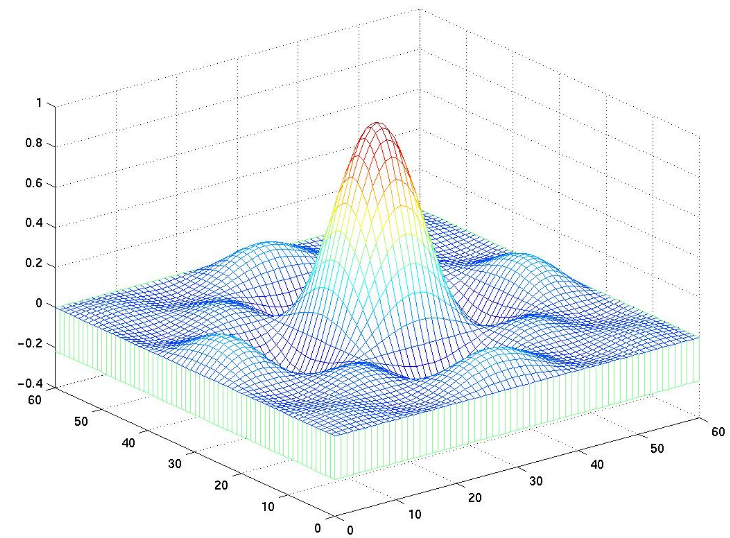
- In particular, if we look at the power spectrum, then we see that convolving image H by G attenuates frequencies where G has low power, and amplifies those which have high power.

- This is referred to as the **Convolution Theorem**

Source: G Hager, D. Kriegman Slides

The Properties of the Box Filter

F(mean filter) =



Thus, the mean filter enhances low frequencies
but also has “side lobes” that admit higher frequencies

Source: G Hager, D. Kriegman Slides

The Gaussian Filter: A Better Noise Reducer

- Ideally, we would like an averaging filter that removes (or at least attenuates) high frequencies beyond a given range

- It is not hard to show that the FT of a Gaussian is again a Gaussian.
  - What does this imply? FT($e^{-\alpha x^2}$) = $\sqrt{\frac{\pi}{\alpha}} \cdot e^{-\frac{(\pi\xi)^2}{\alpha}}$

- Note that in general, we truncate --- a good general rule is that the width (w) of the filter is at least such that w > 5 σ. Alternatively we can just stipulate that the width of the filter determines σ (or vice-versa).

- Note that in the discrete domain, we truncate the Gaussian, thus we are still subject to ringing like the box filter.

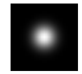
Source: G Hager, D. Kriegman Slides

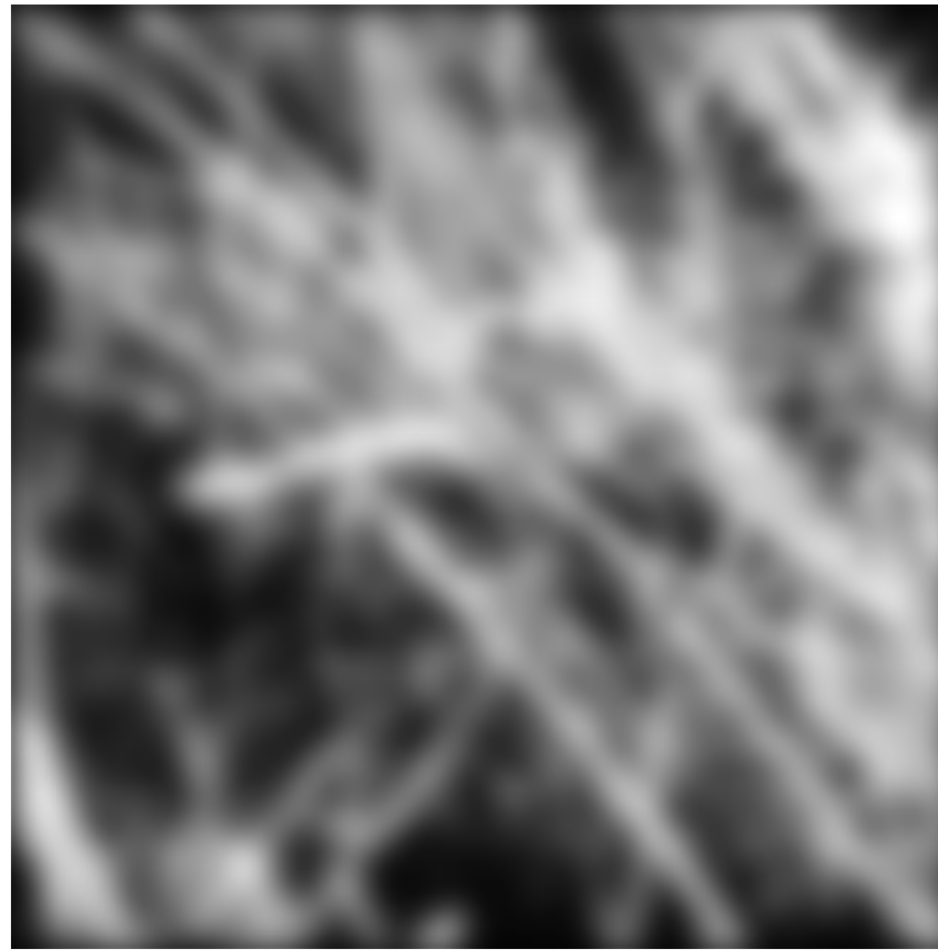
Smoothing by Averaging

Kernel: 

Source: G Hager, D. Kriegman Slides

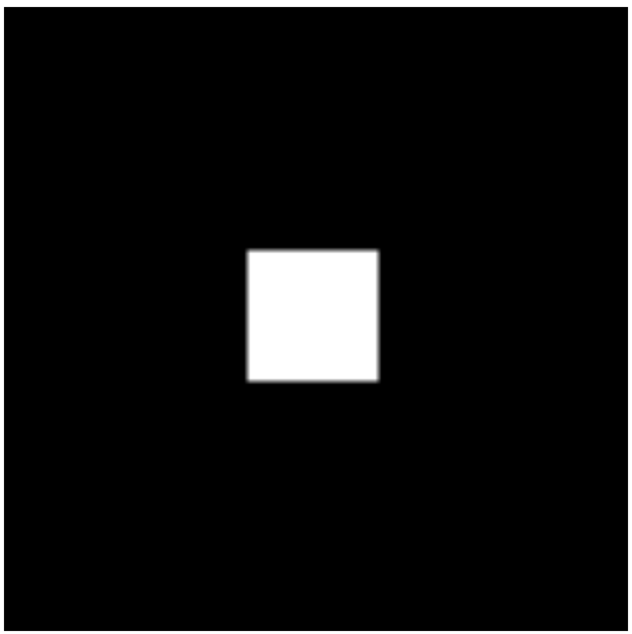
Smoothing with a Gaussian

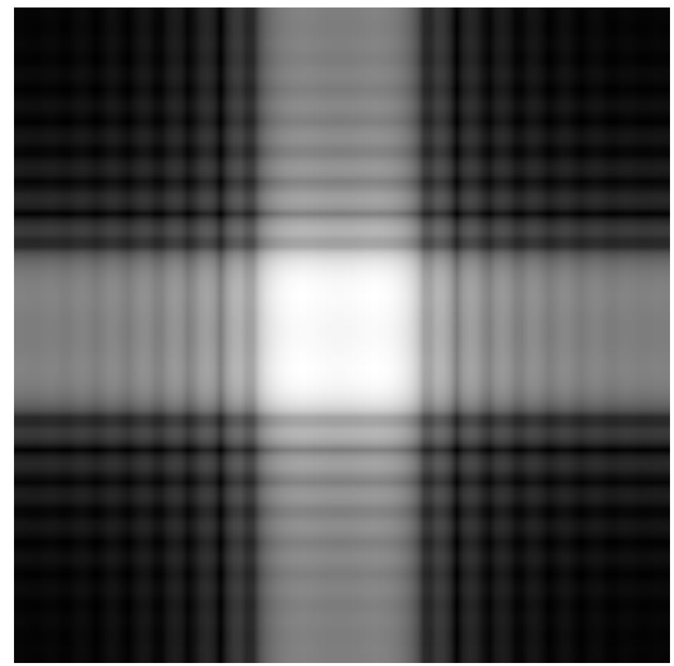
Kernel: 



Source: G Hager, D. Kriegman Slides

Why Not a Frequency Domain Filter?

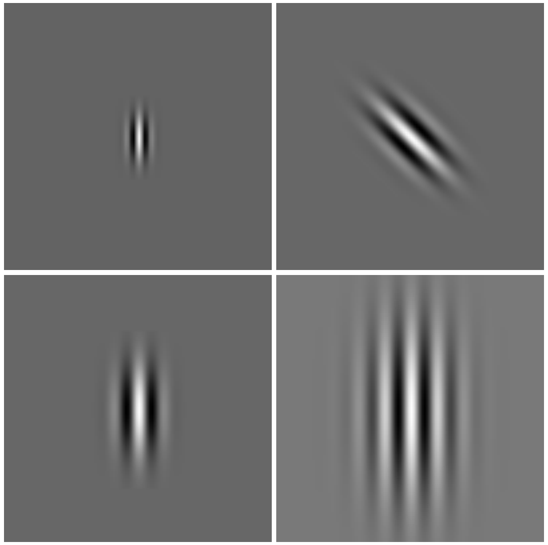




Gabor Filters

- Fourier decompositions are a way of measuring "texture" properties of an image, but they are global
- Gabor filters are a "local" way of getting image frequency content

g(x,y) = s(x,y) w(x,y) == a "sin" and a "weight"



$s(x,y) = \exp(-i\,(2\pi\,(x\,u + y\,v)))$
$w(x,y) = \exp(-1/2\,(x^2 + y^2)/\sigma^2)$

Now, we have several choices to make:

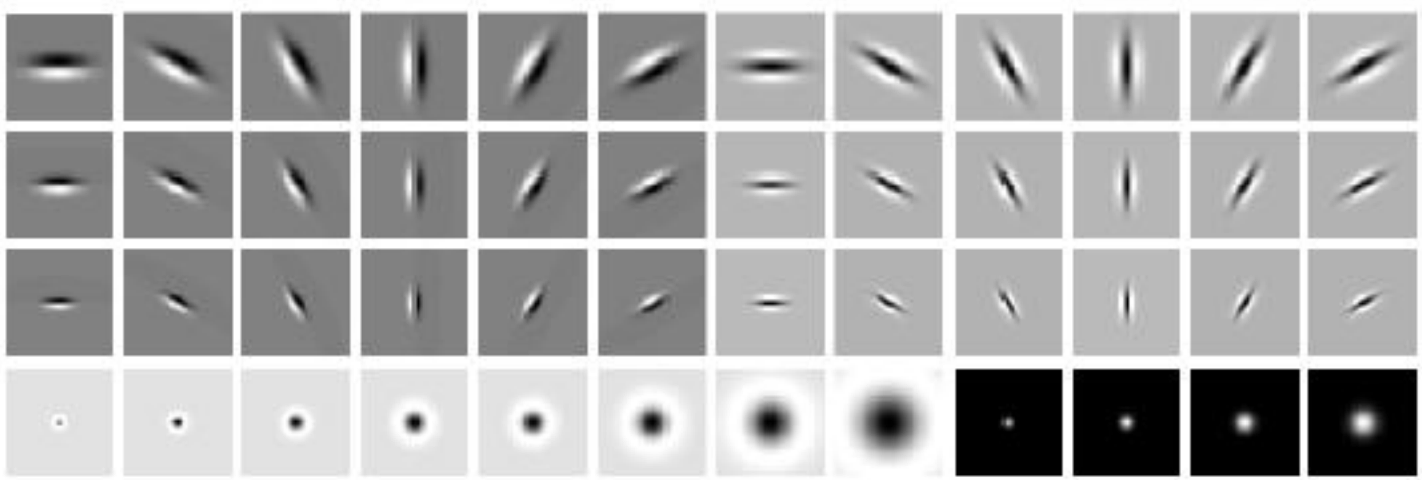
1. u and v defines frequency and orientation
2. σ defines scale (or locality)

Thus, Gabor filters for texture can be computationally expensive as we often must compute many scales, orientations, and frequencies

Source: G Hager, D. Kriegman Slides

Filtering for Texture

- The Leung-Malik (LM Filter): set of edge and bar filters plus Gaussian and Laplacian of Gaussian



Source: G Hager, D. Kriegman Slides

Next Lecture: Local Image Features

- Readings: FP 5; SZ 4.2, 4.3