

Signal Processing for Images

Linear Filters: Blur and Gradients



Antonio Carta

2026-09-23

University of Pisa

antonio.cart@unipi.it

Use Convolution and Fourier transform to design **applied image filters**

1. **Blur**: box, Gaussian, binomial
2. **Image derivatives**: gradients and Laplacian

Every filter is a **Linear and Shift-Invariant (LSI)** system, so we analyze it via:

- its impulse response (convolution kernel, spatial view)
- its frequency response (Fourier transform, frequency view)

Blur Filters



Blur filter:

- *spatial view*: average nearby pixel values
- *frequency view*: attenuates high spatial frequencies

Three classic choices:

- **Box filter**: simplest, cheapest
- **Gaussian filter**: smooth, separable, many nice properties
- **Binomial filter**: efficient discrete approximation to Gaussian
- We compare them by: kernel shape, frequency response, and behavior under repeated application

Note: blurring can be implemented with nonlinear operators. We focus on LSI systems today.



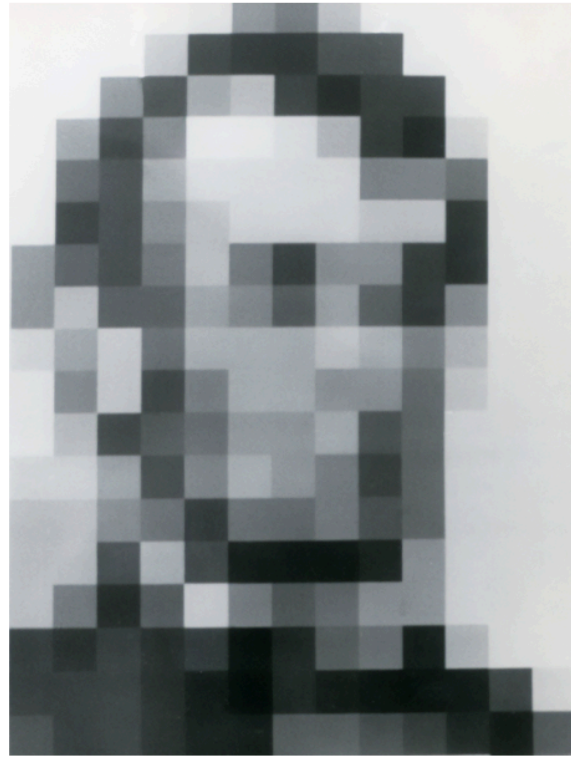
(a) Original noisy image.



(b) Blurred image.

Figure 1: source: visionbook

Blurring Removes Sampling Artifacts



(a) Original Image.



(b) Blurred Image.

Figure 2: source: visionbook

$$\text{box}_{N,M}[n, m] = \begin{cases} 1 & \text{if } -N \leq n \leq N \wedge -M \leq m \leq M \\ 0 & \text{otherwise} \end{cases}$$

- **Intuition**: sum of input pixels around a rectangular area
- This is the unnormalized version
 - we can normalize it to have DC gain: $\sum_n h[n] = 1$ to preserve average brightness
 - Note: DC gain is just the mean value
- **Separable**: 2D box = 1D box along rows * 1D box along columns

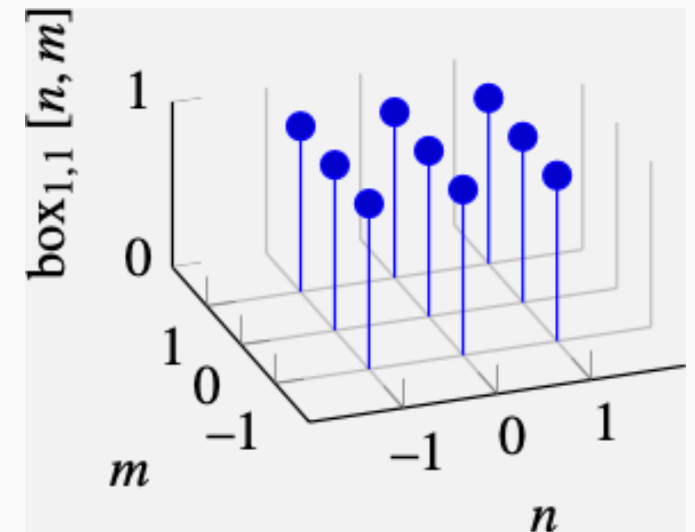


Figure 3: Box kernel. source: visionbook

Fourier of Box Filter

- Notice how the attenuation of high frequencies is not monotonically decreasing.
- Frequency response is a **sinc**: $H(\omega) \propto \frac{\sin((2k+1)\omega/2)}{\sin(\omega/2)}$

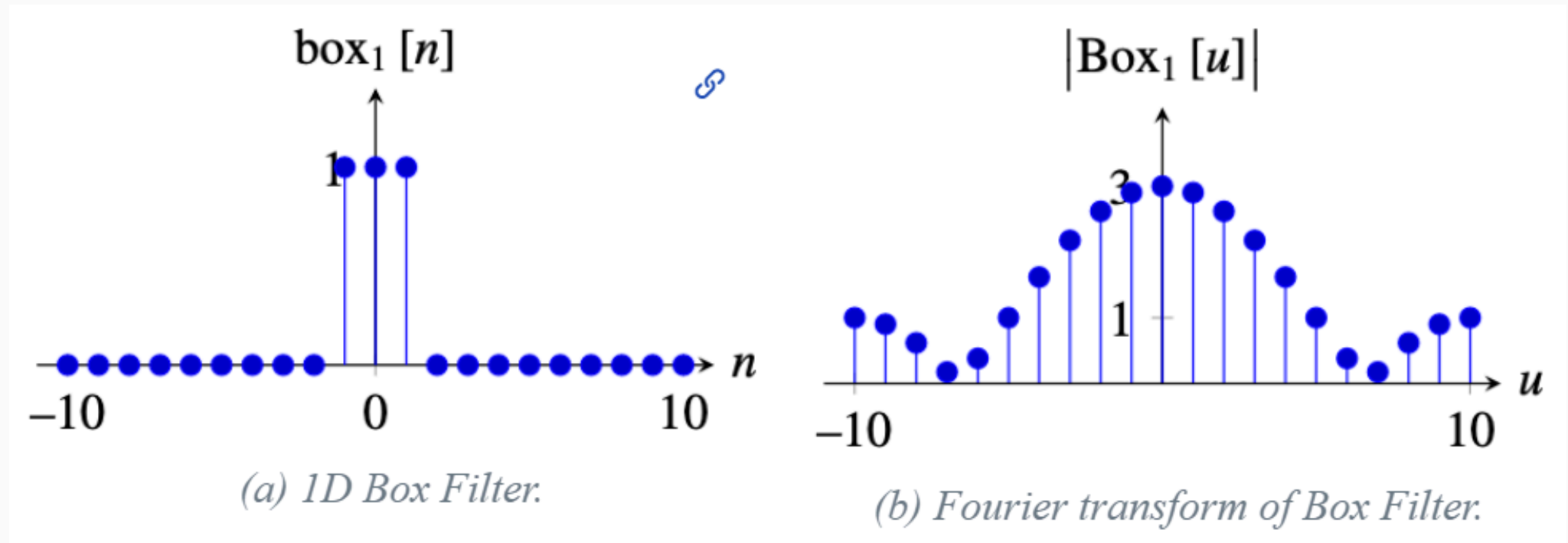


Figure 4: source: visionbook

- even size has a half-pixel shift
- therefore, we usually use odd sizes

- **Non-monotonic**: frequency response has sidelobes $\Rightarrow$ does not cleanly remove high frequencies, can introduce ringing
- Convolution of two box filters does **not** give a larger box ($[1; 1] * [1; 1] = [1; 2; 1]$)
 - Note: having composable blurs is useful for multi-scale representation
- **Visible artifact**: blocky/ringing appearance, especially on sharp edges

1D Gaussian filter:

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

2D Gaussian filter:

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

- Radially symmetric in 2D
- σ controls the spatial scale
- The equations here are normalized such that DC gain is 1

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

Discretization: sample on a grid of size $\approx \pm 3\sigma$ (beyond 3σ , values are negligible)

Discretized Gaussian kernel:

$$g[n, m; \sigma] = \exp\left(-\frac{n^2 + m^2}{2\sigma^2}\right)$$

Note: this is the unnormalized version of the kernel.



*(a) Gaussian filter with
 $\sigma = 2$.*



*(b) Gaussian filter with
 $\sigma = 4$.*



*(c) Gaussian filter with
 $\sigma = 8$.*

Figure 5: source: visionbook

$$\begin{aligned} g[n, m] * \ell[n, m] &= \sum_{k, l} g[n - k, m - l] \ell[k, l] \\ &= \sum_{k, l} \exp\left(-\frac{(n - k)^2 + (m - l)^2}{2\sigma^2}\right) \ell[k, l] \\ &= \sum_k \exp\left(-\frac{(n - k)^2}{2\sigma^2}\right) \left(\sum_l \exp\left(-\frac{(m - l)^2}{2\sigma^2}\right) \ell[k, l]\right) \\ &= g^x * (g^y * \ell[n, m]) \end{aligned}$$

- **Circular Symmetry**: the Gaussian is the only circularly symmetric filter that is separable
- **Self-similar under Fourier transform**: FT of a Gaussian is a Gaussian
 - 1D: $G(w; \sigma) = \exp\left(-\frac{w^2 \sigma^2}{2}\right)$
 - a Gaussian low-pass filter has a Gaussian (smooth, monotonic, no sidelobes) frequency response, fixes the box filter's main flaw
- **Convolution of two Gaussians is a Gaussian**, with variances adding:
 - this the basis behind Gaussian pyramids

$$g_{\sigma_1} * g_{\sigma_2} = g_{\sigma_3}, \quad \sigma_3^2 = \sigma_1^2 + \sigma_2^2$$

- **Heat equation:** g_σ is the solution at time $\sigma^2/2$ of the diffusion equation $\partial u / \partial t = \nabla^2 u$ with a point-source initial condition; blurring with a larger σ is literally “letting heat diffuse longer” (we will reuse this for scale-space)
- **Central limit theorem:** convolving any sufficiently nice kernel with itself many times converges to a Gaussian; this is *why* repeated simple filters (box, binomial) approximate Gaussian blur

- The continuous Gaussian is supported on all of $\mathbb{R}$; any **discrete truncation breaks exact properties**:
 - truncated kernel does not sum to exactly 1 (small DC error)
 - variance addition $\sigma_3^2 = \sigma_1^2 + \sigma_2^2$ holds only approximately
- **Repeated convolution accumulates these small errors**; after many applications, results drift from the ideal continuous Gaussian

The **binomial filter** fixes these limitations with well-behaved discrete arithmetic

- $$[1, 1] * [1, 1] = [1, 2, 1], \quad [1, 2, 1] * [1, 1] = [1, 3, 3, 1], \dots$$
- n -fold convolution of $[1, 1]$ gives the binomial coefficients $\binom{n}{k}$, $k = 0, \dots, n$
 - All coefficients are **integers**, exact arithmetic, no rounding error

b_0						1														$\sigma_0^2 = 0$
b_1						1		1												$\sigma_1^2 = 1/4$
b_2					1		2		1											$\sigma_2^2 = 1/2$
b_3				1		3		3		1										$\sigma_3^2 = 3/4$
b_4			1		4		6		4		1									$\sigma_4^2 = 1$
b_5		1		5		10		10		5		1								$\sigma_5^2 = 5/4$
b_6		1	6		15		20		15		6		1							$\sigma_6^2 = 3/2$
b_7	1		7		21		35		35		21		7		1					$\sigma_7^2 = 7/4$
b_8	1	8		28		56		70		56		28		8		1				$\sigma_8^2 = 2$

- DC gain = 2^n (sum of binomial coefficients), normalize by dividing by 2^n
- Variance: $\sigma^2 = n/4$, grows linearly with the number of $[1, 1]$ convolutions
- **Composition rule**: $b_n * b_m = b_{n+m}$, exact, no approximation (unlike truncated Gaussians)
- $n = 2$ gives the ubiquitous **3-tap kernel** $[1, 2, 1]/4$
- Frequency response is **monotonically decreasing** (no sidelobes, like Gaussian)
- symmetric kernel
- $[1, -1, 1, -1, \dots]$ (the highest possible frequency) is perfectly annihilated by $[1, 2, 1]$: $1 - 2 + 1 = 0$
 - this does not happen with the odd-sized box filter

- Separable construction: $B_{n(x,y)} = b_{n(x)} b_{n(y)}$
- The standard 3×3 binomial kernel ($n = 2$):

$$\frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

- fast, exact, separable smoothing, very useful for image pyramids (next lecture) and efficient on-device/hardware filtering

Box vs. Gaussian vs. Binomial

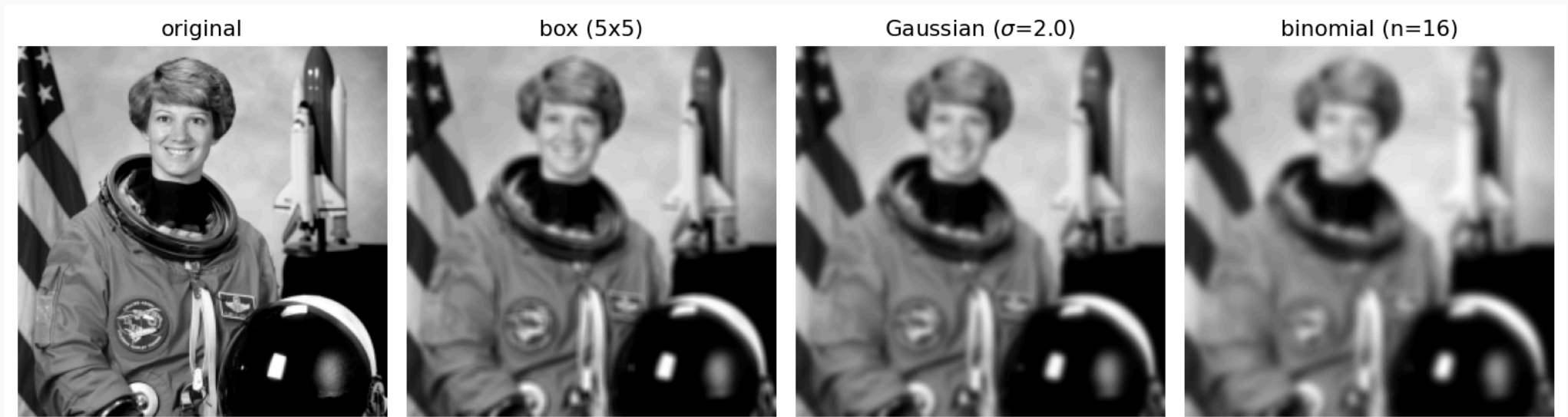


Figure 7: Original image filtered with a 5×5 box, a Gaussian ($\sigma = 2$), and a binomial filter (16 applications of $[1, 2, 1]/4$)

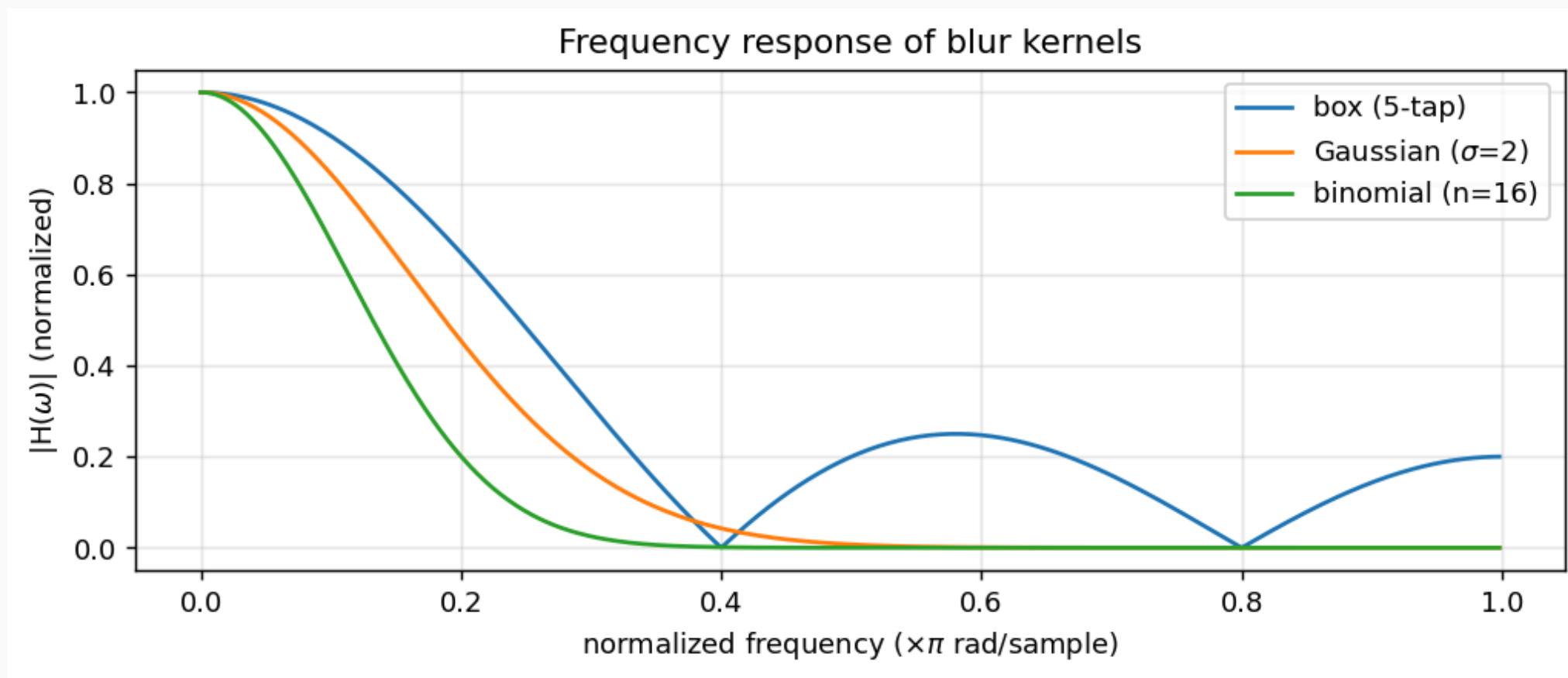
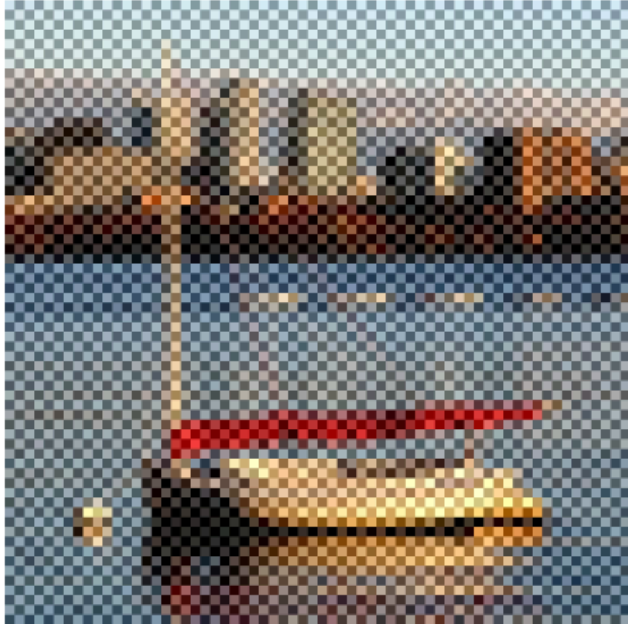
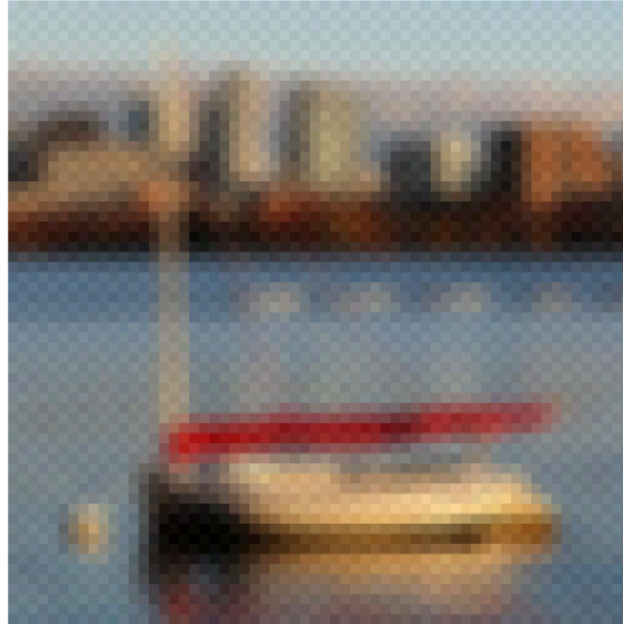


Figure 8: Frequency responses: box has sidelobes (ringing); Gaussian and binomial are smooth and monotonic

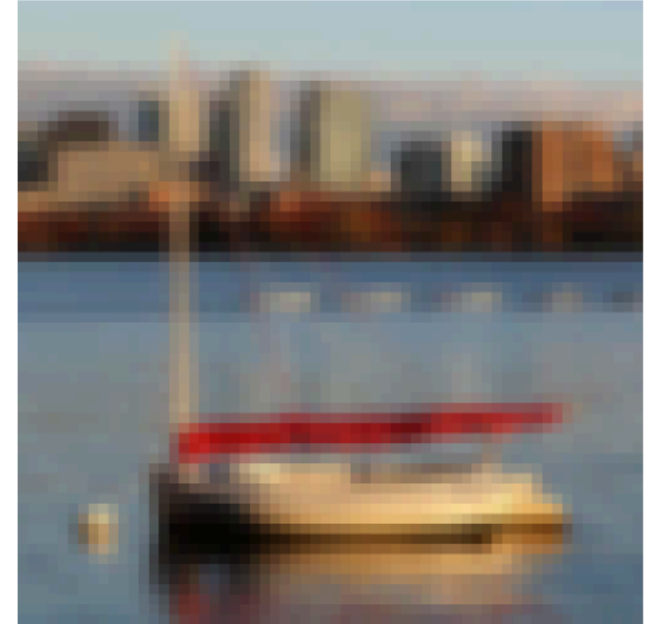
Noisy Box vs Binomial



(a) Noisy Image.



(b) Box Filtered Image.



(c) Binomial Filtered Image.

Figure 9: source: visionbook

Practical Recommendations:

- **Box**: fast but adds artifacts
- **Gaussian**: optimal frequency-domain behavior in continuous domain; “breaks” in the discrete domain
- **Binomial**: fast and keeps the properties of Gaussian blur in the discrete domain

Intuition:

- in **space**, blur averages pixels and removes noise
- in **frequency**, blur attenuates high-frequency components

Image Derivatives

Why Derivatives?

- Image derivatives capture **changes**: edges, boundaries, texture gradients
- The operator computing $\partial I / \partial x$ and $\partial I / \partial y$ is **linear and shift-invariant**
 - therefore we can implement it with a convolution
- Applications: edge/boundary detection, shape-from-shading and other 3D recovery cues, image editing, sharpening, illumination/reflectance separation

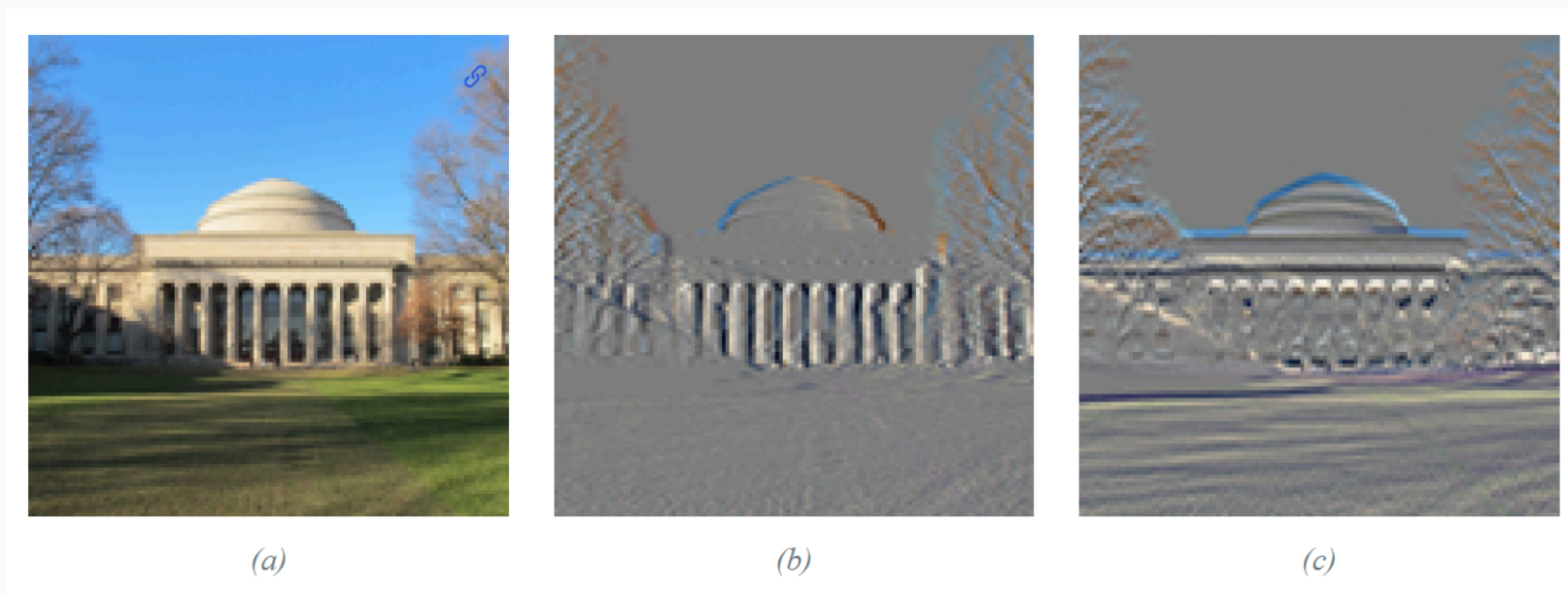


Figure 10: source: visionbook

Continuous derivative: $f'(x) = \lim_{h \rightarrow 0} (f(x+h) - f(x))/h$.

Two standard discrete approximations:

$$d_0 = [1, -1] \qquad d_1 = \frac{1}{2}[1, 0, -1]$$

- d_0 : forward difference, 2-tap, half-sample shift (result lies between input samples)
- d_1 : central difference, 3-tap, no shift, but skips the center sample

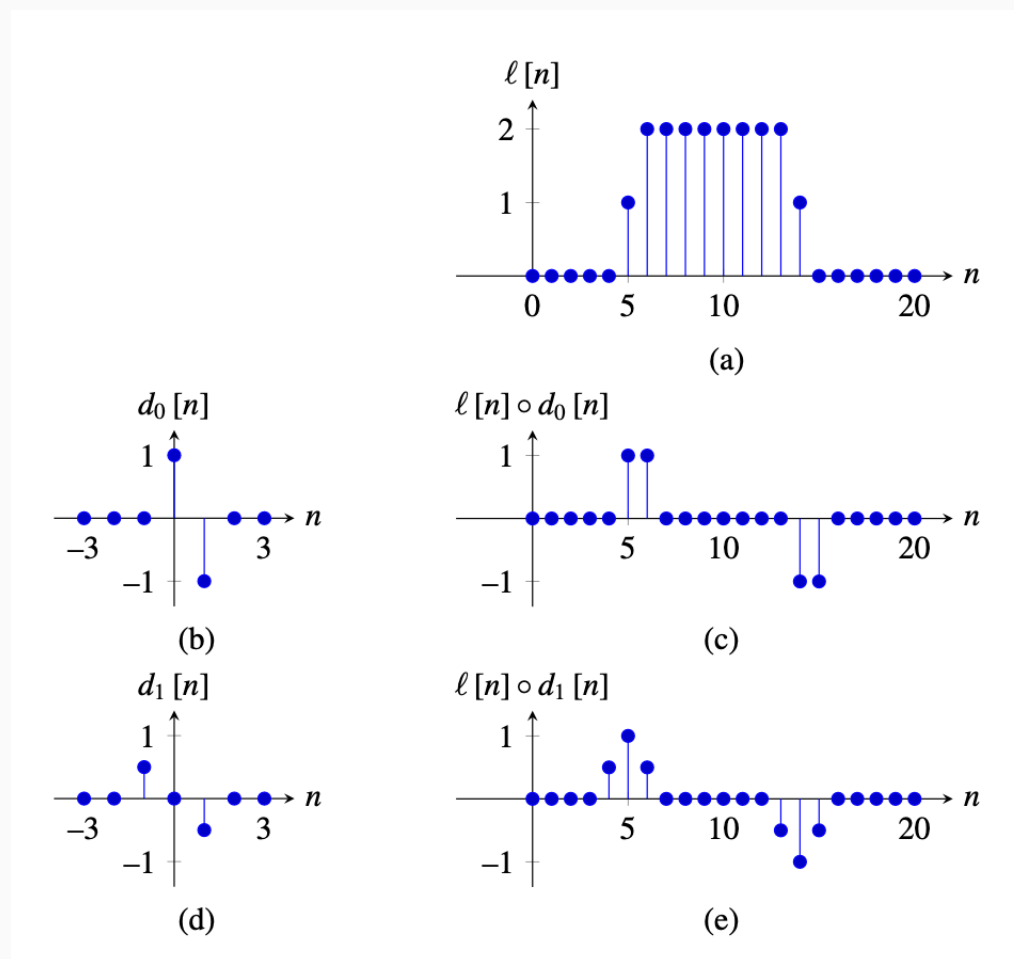


Figure 11: source: visionbook

Frequency Response

- the black line is the ideal frequency response of the derivative
- both d_0 and d_1 are good approximations only for low frequencies
- d_0 has a better magnitude approximation but it causes a phase shift

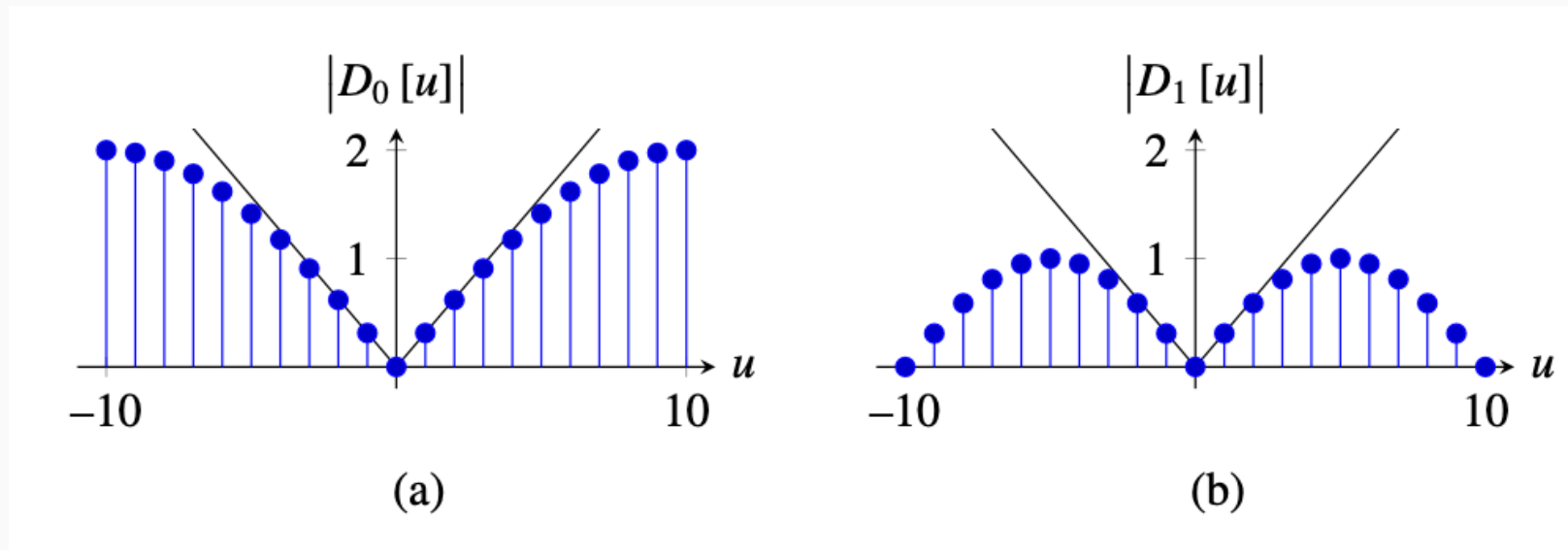


Figure 12: source: visionbook

in 2D, we can compute x and y derivatives separately. With d_0 :

$$d_0^x = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

$$d_0^y = (1 \quad -1)$$

Note that the outputs are misaligned:

- d_0^x is shifted half a pixel in x
- d_0^y is shifted half a pixel in y

- we have a discrete approximation of the derivative
- it's fast
- it's imperfect: shifted and scaled compared to the true derivative

Gradient-Based Image Representation

Three reasons for using gradients as image representations:

- gradients highlight some structure such as edges
- gradients are “almost” invertible
- motivating example: image editing

For simplicity, we work in 1D. We can write the matrix corresponding to the convolution with $d_0 = (1 \ -1)$

$$D_0 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix} \quad D_0^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

In this case, inversion is exact! but we are assuming zero padding

If we have a valid padding (i.e., we keep only the pixels where all the inputs are inside the image):

$$D_0 = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

we lost the first row. We cannot invert D_0 because it has N inputs and $N - 1$ outputs.

We can compute the pseudo inverse:

$$D_0^+ = \frac{1}{5} \begin{pmatrix} -4 & -3 & -2 & -1 \\ 1 & -3 & -2 & -1 \\ 1 & 2 & -2 & -1 \\ 1 & 2 & 3 & -1 \\ 1 & 2 & 3 & 4 \end{pmatrix}$$

We lost the DC component (mean of the image).

$$d_0 = (1 \ -1)$$

$$\ell = [1, 1, 2, 2, 0]$$

$$r = D_0 \ell = [0, -1, 0, 2]$$

$$\hat{\ell} = D_0^+ r = [-0.2, -0.2, 0.8, 0.8, -1.2]$$

the mean of the image is $\frac{1+1+2+2+0}{5} = 1.2$

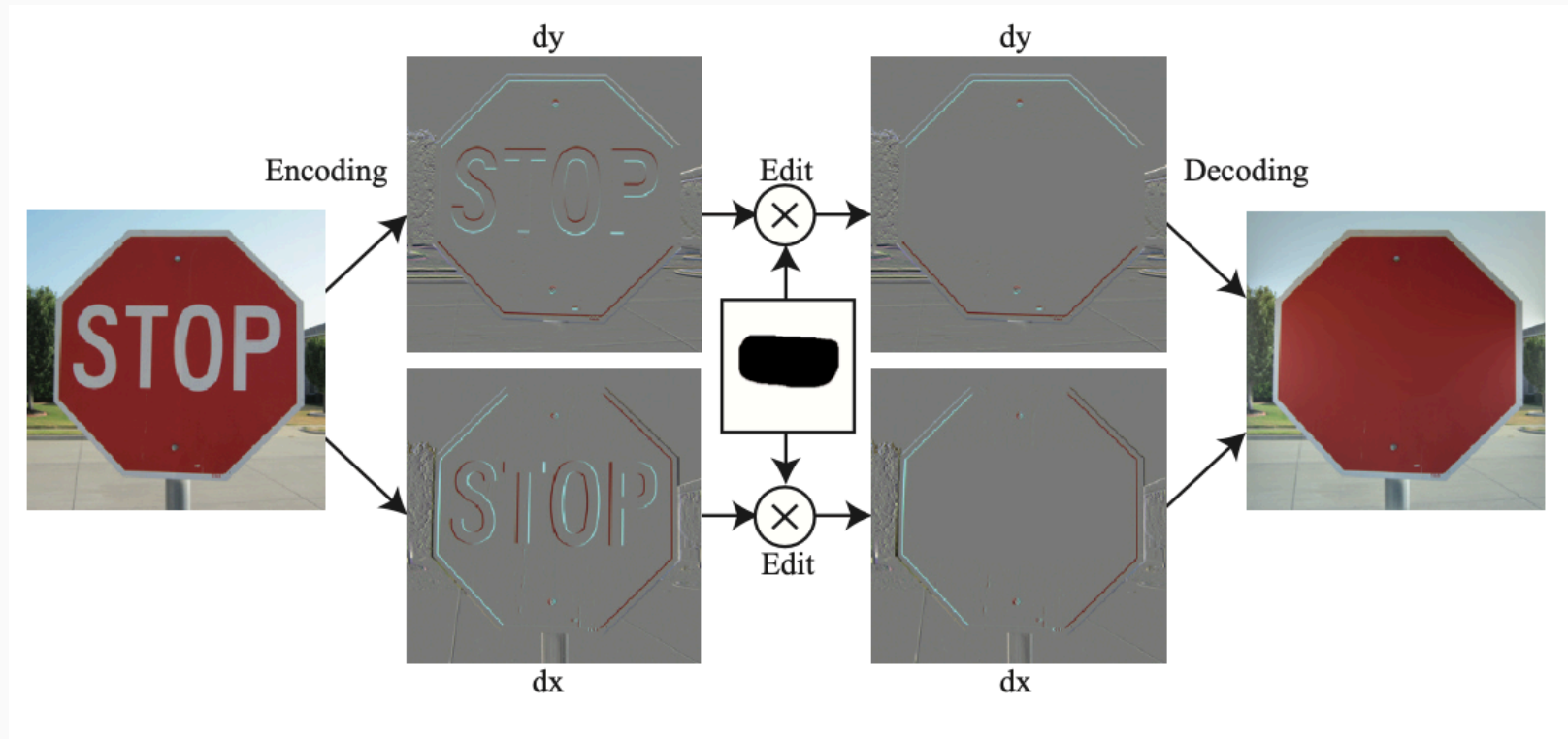


Figure 13: source: visionbook

Notice how we didn't have to specify anything about colors or orientation.

Blurred Derivatives



Blurred Derivatives

- Noise is amplified by derivatives
- example: if I have i.i.d. Gaussian noise with variance σ^2 in each pixel, convolution with $d_0 = \begin{pmatrix} 1 & -1 \end{pmatrix}$ results in noise with variance $2\sigma^2$ (sum of Gaussians)
- there is also a scaling/sampling problem. When we design detectors, we need to pick a desired “scale/resolution”

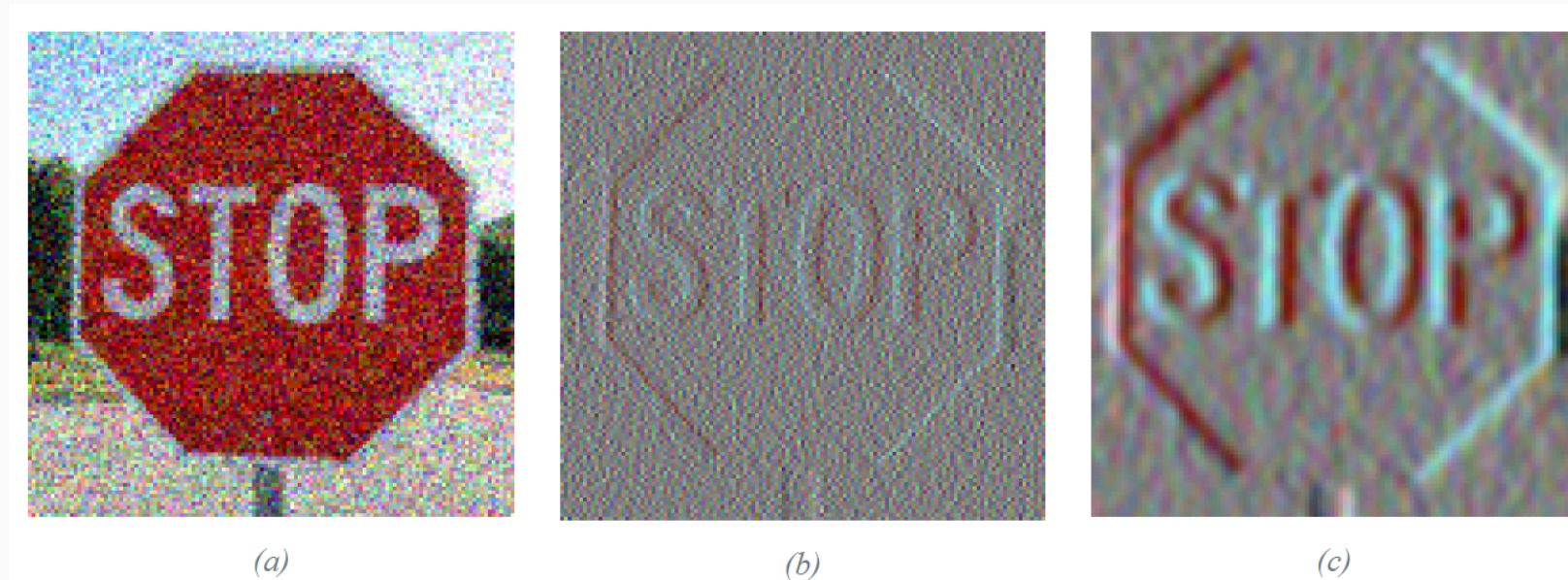


Figure 14: source: visionbook

Blurred derivative: compute the derivative, then blur the result to remove noise (high frequencies)

Derivative and convolution commute:

$$\frac{\partial \ell(x, y)}{\partial x} * g(x, y) = \ell(x, y) * \frac{\partial g(x, y)}{\partial x}$$

- Therefore, to implement a blurred derivative, we just need to design an appropriate kernel by computing the derivative of a blur kernel

Gaussian derivatives smooth first, then differentiate:

$$g'_{\sigma(x)} = -\frac{x}{\sigma^2} g_{\sigma(x)}$$

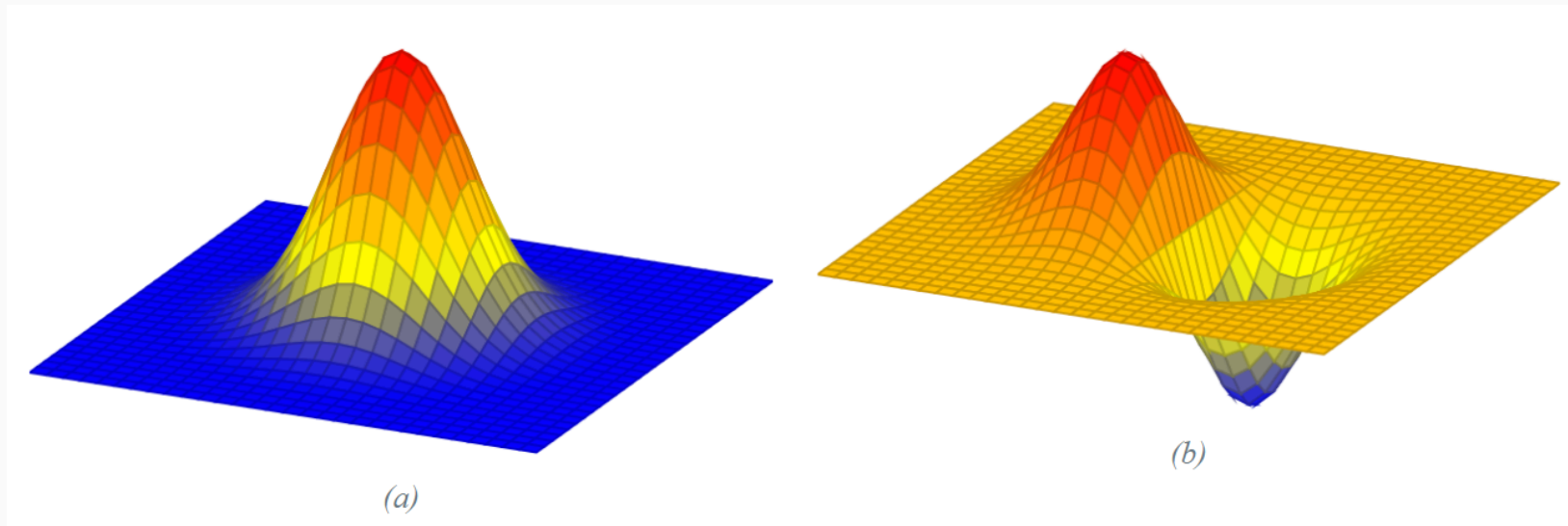


Figure 15: source: visionbook

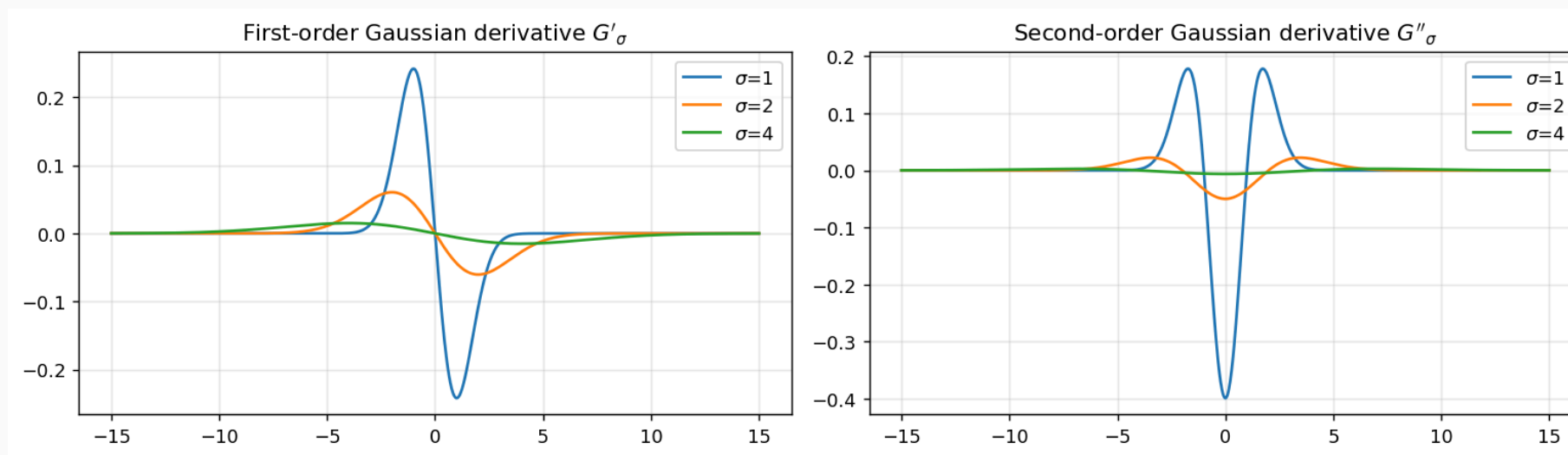


Figure 16: First- and second-order Gaussian derivative kernels at $\sigma = 1, 2, 4$

Derivatives are not scale-invariant!

- Larger $\sigma \Rightarrow$ wider kernel, responds to **coarser** changes, less sensitive to noise
- Smaller $\sigma \Rightarrow$ responds to **fine** detail, more sensitive to noise
- Choice of σ = choice of the **scale** at which we measure change

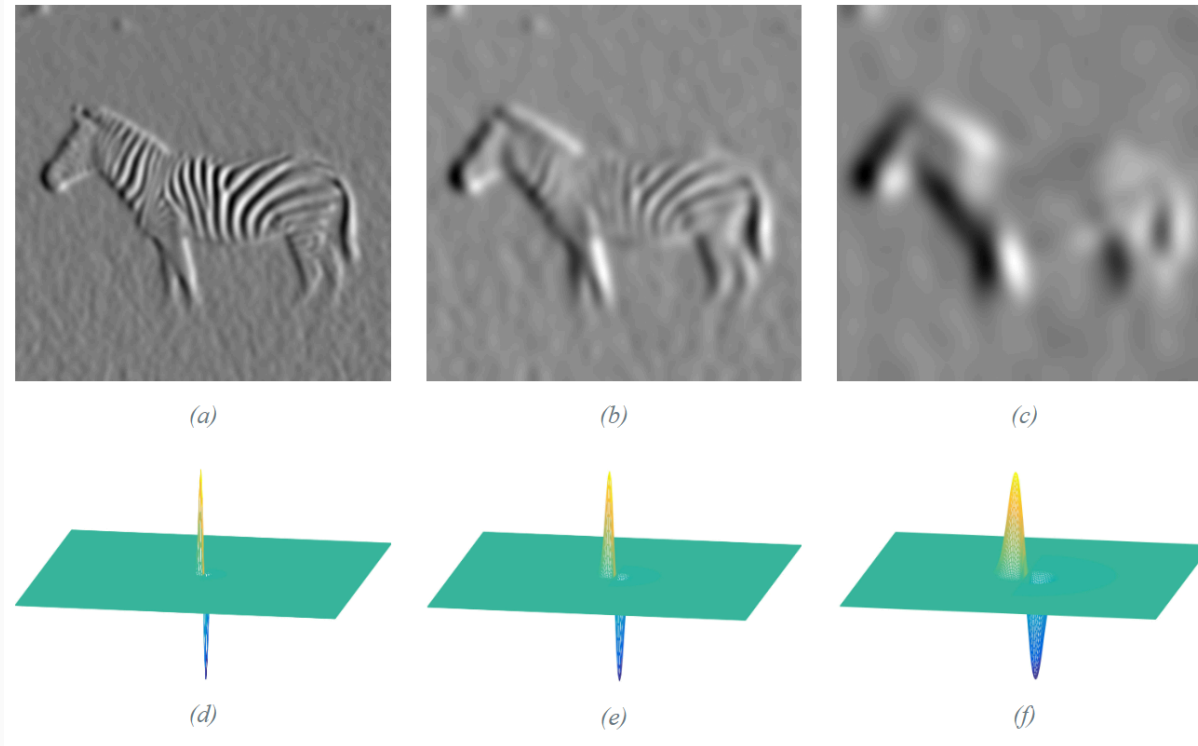


Figure 17: First- and second-order Gaussian derivative kernels at $\sigma = 1, 2, 4$. source: visionbook

- General n -th order 1D Gaussian derivative: $g_{\sigma}^{(n)}(x) = H_n(x/\sigma)g_{\sigma}(x)$ where H_n is (up to scaling) a Hermite polynomial
- Second-order:

$$g_{\sigma(x)}'' = \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2} \right) g_{\sigma(x)}$$

- 2D separable derivatives: products of 1D derivative/smoothing kernels, e.g.

$$\partial_{xx}g_{\sigma(x,y)} = g_{\sigma(x)}''g_{\sigma(y)}, \quad \partial_{xy}g_{\sigma(x,y)} = g_{\sigma(x)}'g_{\sigma(y)}'$$

These form a systematic family (“Gaussian derivative filter bank”), full set of local image structure measurements at a given scale.

Higher-Order Gaussian Derivatives

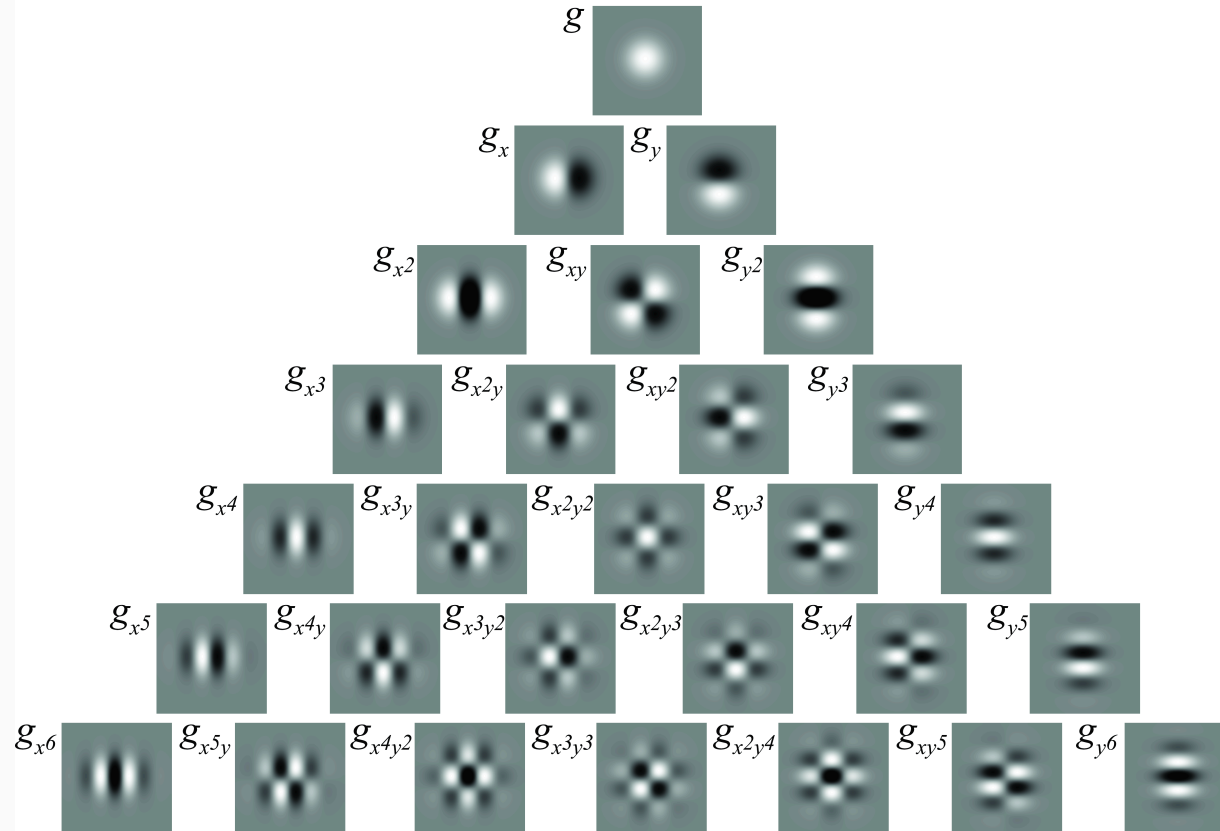


Figure 18: Higher-order Gaussian kernels. source: visionbook

- Just as binomial filters approximate the Gaussian, **differences of binomial filters approximate Gaussian derivatives**
- **Sobel–Feldman operator** = separable composition of a binomial smoothing kernel $([1, 2, 1])$ in one direction and a derivative kernel $([1, 0, -1])$ in the other:

$$S_x = [1, 2, 1]^T * [1, 0, -1] = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

- Gradient vector: $\nabla \ell(x, y) = \left(\frac{\partial \ell(x, y)}{\partial x}, \frac{\partial \ell(x, y)}{\partial y} \right)$
 - two derivative images (x and y)
- For Gaussian derivatives: $\nabla \ell \circ g = \nabla g \circ \ell = \left(g_{x(x, y)}, g_{y(x, y)} \right) \circ \ell$
- **Directional derivative** along unit vector t are expressed as linear combinations of $\frac{\partial \ell(x, y)}{\partial x}$ and $\frac{\partial \ell(x, y)}{\partial y}$:

$$\frac{\partial \ell(x, y)}{\partial t} \circ g = \cos(\theta) g_{x(x, y)} \circ \ell(x, y) + \sin(\theta) g_{y(x, y)} \circ \ell(x, y)$$

Example: Sobel

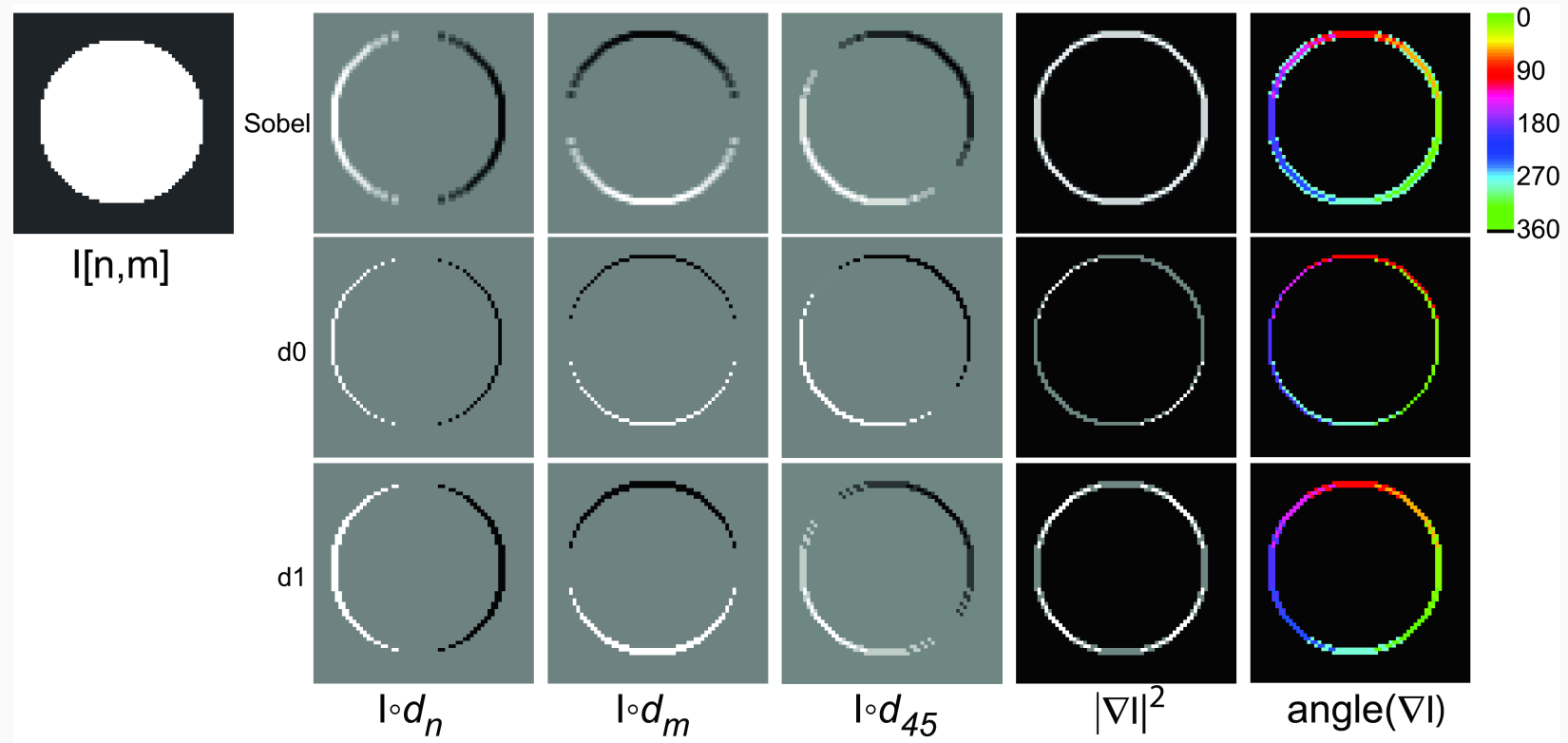


Figure 19: Gradient magnitude and orientation for different image derivatives. source: visionbook

Comparing d_0 , d_1 , Sobel, and a true Gaussian derivative. The Sobel operator gives the most rotationally invariant gradient magnitude, but it is blurrier.

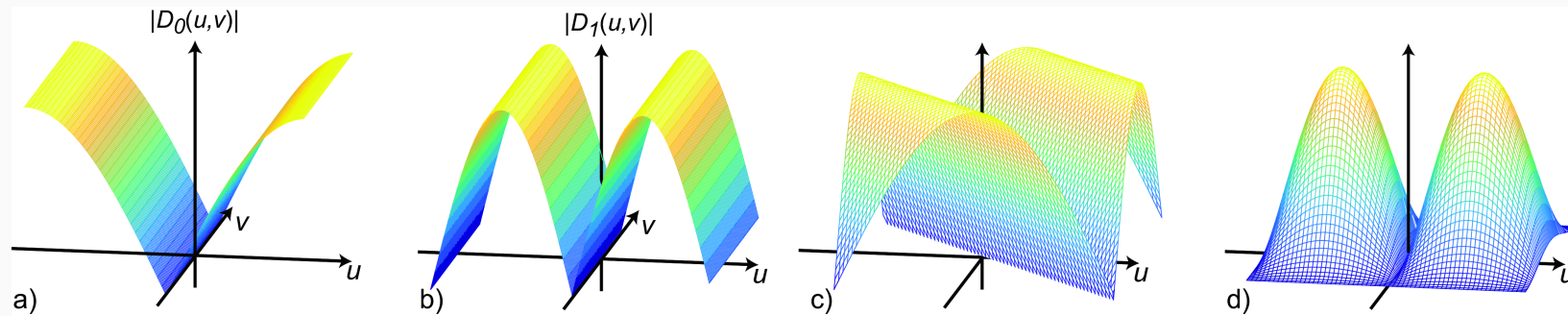


Figure 20: Frequency response for image derivatives: d0, d1, Robert cross operator, Sobel-Feldman. source: visionbook



Figure 21: Image, gradient magnitude, and gradient orientation.

- High magnitude: edges/boundaries
- Orientation is perpendicular to edge direction (a cue used by some feature descriptors)
- Consequence: we can build orientation detectors with directional derivatives

- derivatives depend on the chosen direction. As a result, we may need to compute several derivatives at different orientations to extract features
- The **Laplacian** is a second-order, **rotationally invariant** operator:

$$\nabla^2 l = \frac{\partial^2 l}{\partial x^2} + \frac{\partial^2 l}{\partial y^2}$$

- similarly to the derivative, this operator is sensitive to noise, so we generally want to blur it

Laplacian of Gaussian (LoG)

- Combines Gaussian smoothing + Laplacian: $\nabla^2 l \circ g_\sigma = \nabla^2 g_\sigma \circ l$
- $\text{LoG}_\sigma = \nabla^2 g_\sigma$, a “Mexican hat” shaped kernel
- scale-dependence due to the Gaussian parameter σ

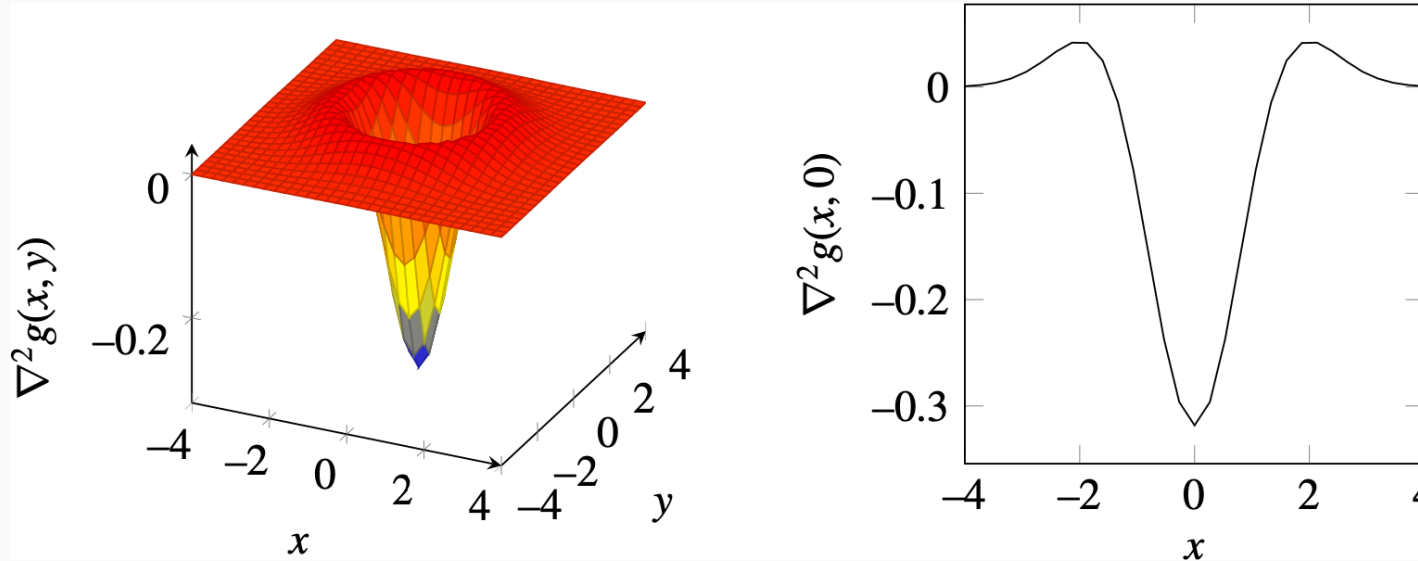


Figure 22: LoG kernel. source: visionbook

for $\sigma > 1$ the LoG is a band-pass filter.

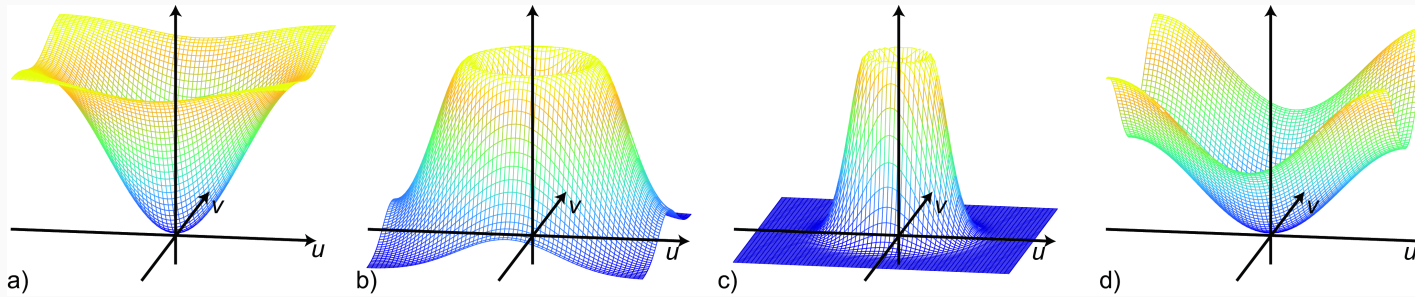


Figure 23: LoG FFT. source: visionbook

(a) $\sigma = \frac{1}{2}$; (b) $\sigma = 1$; (c) $\sigma = 2$; and (d) DFT of the five-point discrete approximation:

$$\nabla_5^2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Laplacian: $\nabla^2 l = \frac{\partial^2 l}{\partial x^2} + \frac{\partial^2 l}{\partial y^2}$

- in 1D, we can approximate the Laplacian as $[1, -2, 1]$
- this is obtained by $d_0 \circ d_0 = [1, -1] \circ [1, -1]$

Laplacian: $\nabla^2 l = \frac{\partial^2 l}{\partial x^2} + \frac{\partial^2 l}{\partial y^2}$

- Discrete 5-point approximation:

$$\nabla^2 l[m, n] \approx l[m + 1, n] + l[m - 1, n] + l[m, n + 1] + l[m, n - 1] - 4l[m, n]$$

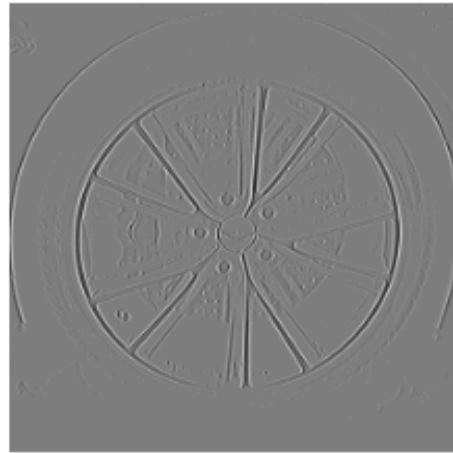
- Measures local **curvature**: positive in dark regions surrounded by bright, negative in bright regions surrounded by dark, ≈ 0 in flat/linear regions

$$\nabla_5^2 = \begin{bmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \end{bmatrix}$$

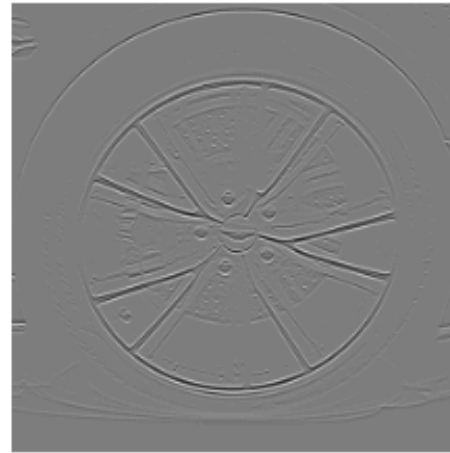
- it is sum-separable: equivalent to compute the 1D kernel in the x and y direction and summing the two results.



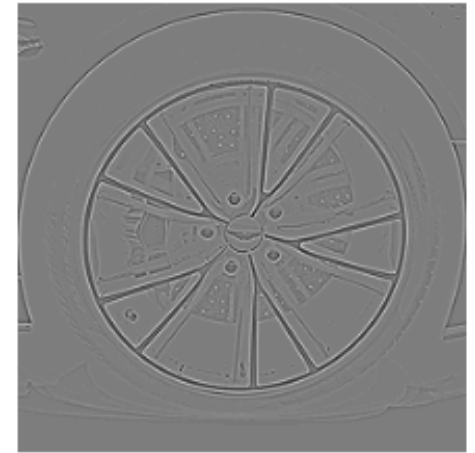
(a)



(b)



(c)



(d)

Figure 24: (b) 2nd-order derivative along x; (c) 2nd-order derivative along y; (d) Laplacian.

Source: visionbook

compared to derivatives, the Laplacian filter:

- It is rotationally invariant
- It measures curvature. If the image contains a linear trend the derivative will be non-zero despite having no boundaries, while the Laplacian will be zero
- Edges can be located as the zero-crossings in the Laplacian output (not very reliable)
- Zero crossings of an image form closed contours

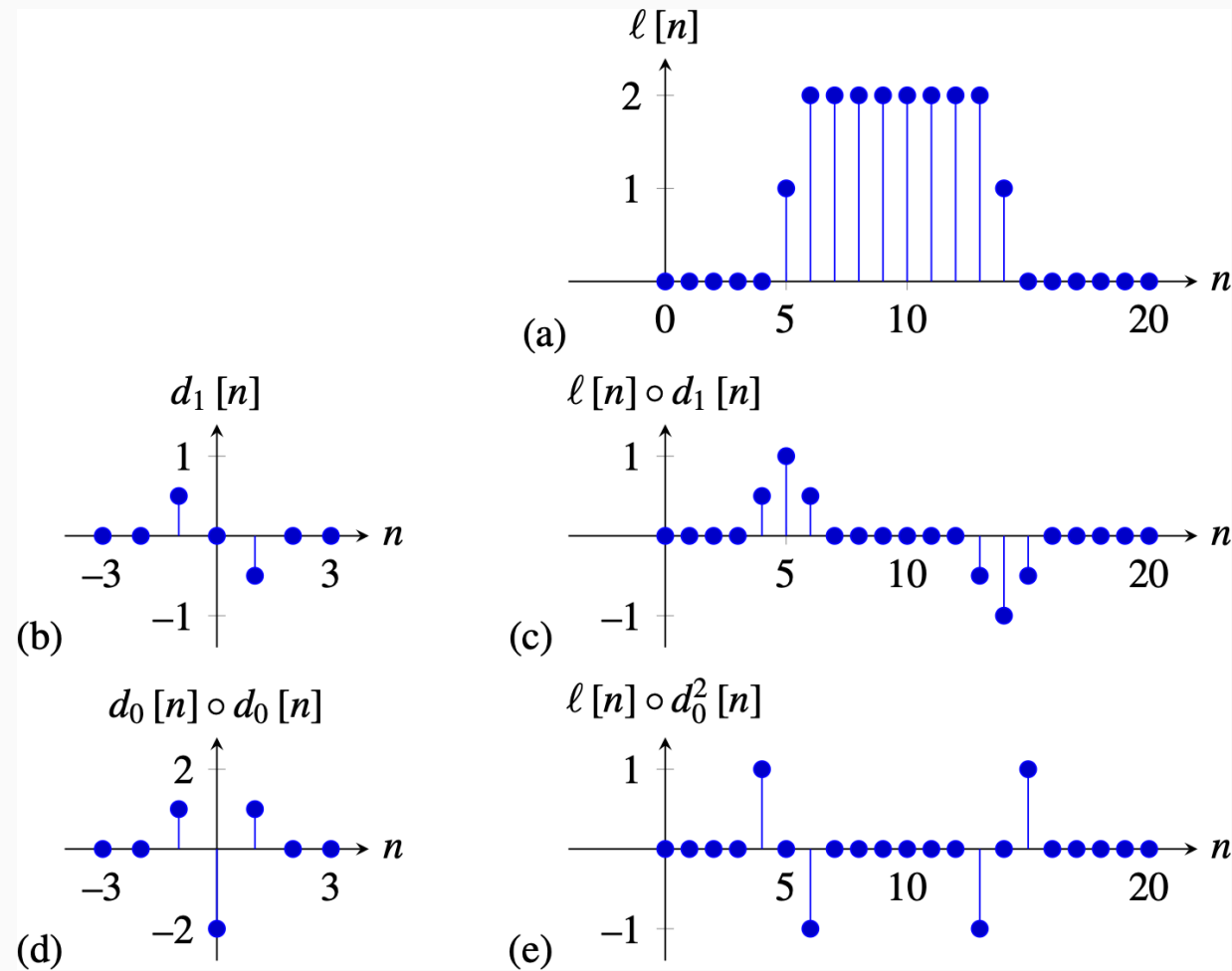


Figure 25: Zero-crossings of LoR correspond to maxima in the derivative. source: visionbook

- derivatives highlight edges and structure
- derivatives are oriented detectors
- noise requires blur, which introduces a scale parameter
- LoG as a rotationally-invariant representation
 - LoG is not necessarily better, just a different representation with its own set of tradeoffs
 - we will use the LoG in multi-scale representation and for SIFT

Early Visual System

- The **early visual system** is the “perception” component in our brain: from the eye to the primary visual cortex. It does the basic image processing
- To motivate the filters we just studied we show its connection with the LoG

- Human contrast sensitivity is well modeled by combining a Laplacian (edge enhancement) with a Gaussian (smoothing):

$$h = -\nabla^2 g_\sigma + \lambda g_\sigma$$

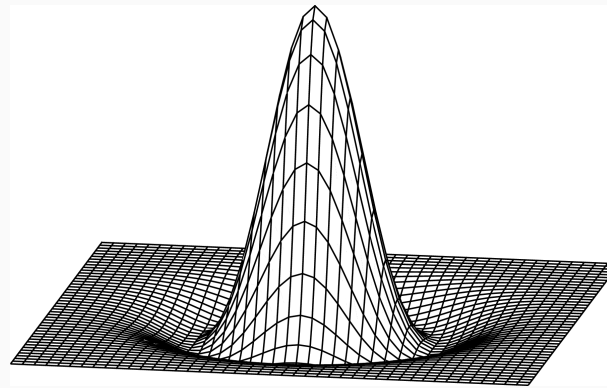


Figure 26: Kernel. source: visionbook

- Its frequency response peaks at intermediate spatial frequencies and matches measured human **contrast sensitivity function**

Contrast Sensitivity Function (Campbell and Robson)

- In the image, the amplitude of the waves varies vertically, but it is constant horizontally
- we perceive the middle of the image as waves but not the extremes. This is an illusion due to the contrast sensitivity of our visual system
- The FFT of $h = -\nabla^2 g_\sigma + \lambda g_\sigma$ matches well our perception

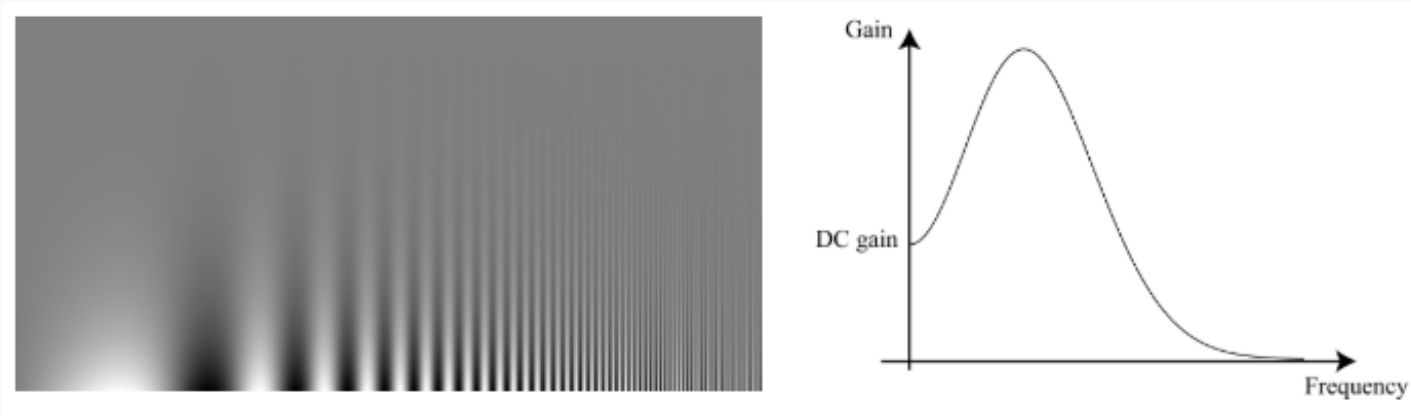


Figure 27: source: visionbook

- **Vasarely illusions** ((a), (c)) are made of nested squares. The bright/dark diagonals are an illusion (apparent brightness gradients/halos at flat-color boundaries), a side effect of band-pass filtering by the visual system
- $h = -\nabla^2 g_\sigma + \lambda g_\sigma$
- $l * h$ matches well our perception ((b), (d))

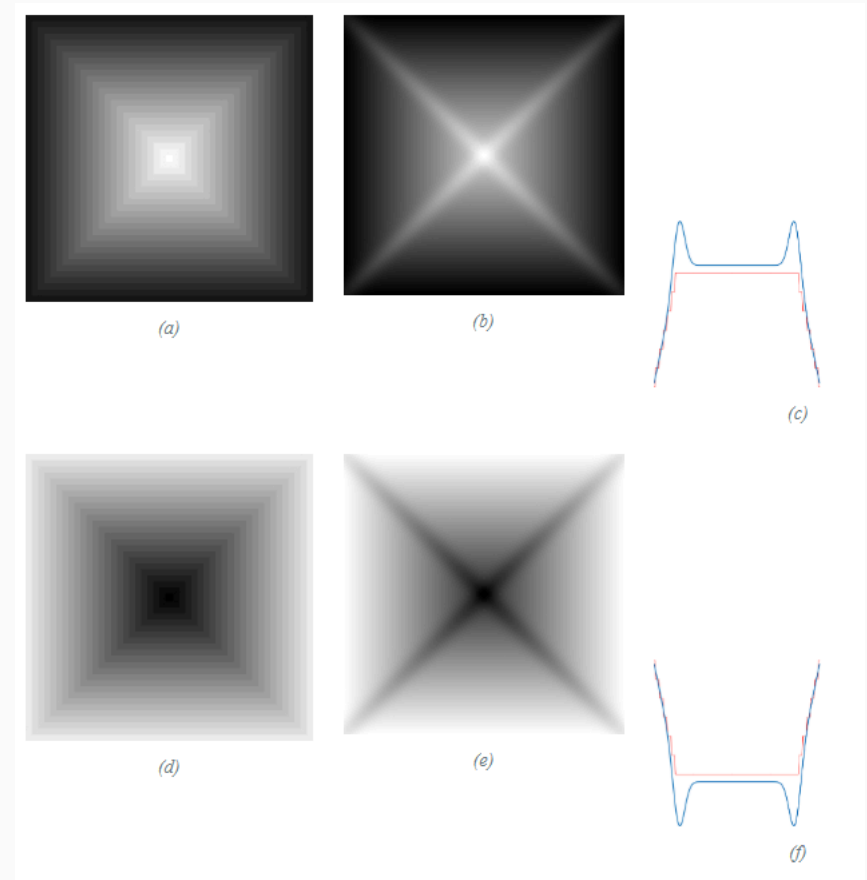


Figure 28: source: visionbook

Sharpening



- Idea: enhance high frequencies by **subtracting a blurred copy**:

$$l_{\text{sharp}} = 2l - (g_{\sigma} * l) = l + (l - g_{\sigma} * l)$$

- $(l - g_{\sigma} * l)$ is a high-pass version of l ; adding it back amplifies edges/texture
- Equivalent to convolving with $h = 2\delta - g_{\sigma}$, a filter with frequency response $2 - \hat{g}_{\sigma}(\omega)$, which is > 1 at high frequencies
- Repeated/iterated application progressively increases sharpness (and eventually noise/ringing)

$$l_{\text{sharp}} = 2l - (g_{\sigma} * l) = l + (l - g_{\sigma} * l)$$

Kernel:

$$\text{sharpening filter} = \begin{bmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{bmatrix} - \frac{1}{16} \begin{bmatrix} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \end{bmatrix}$$

Example 1

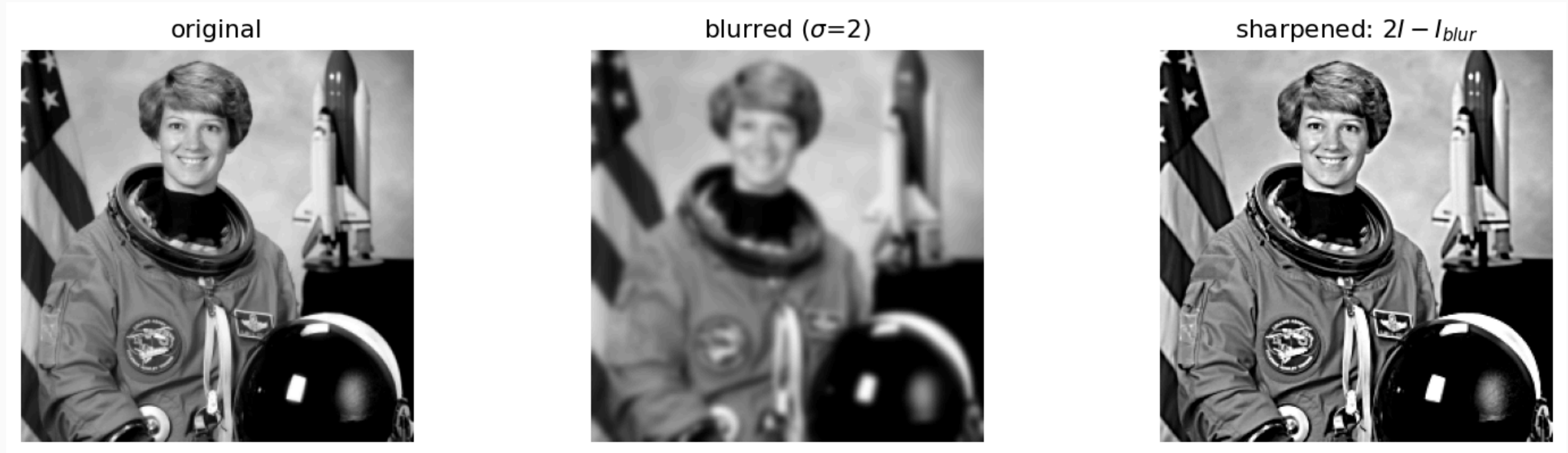


Figure 29: Original, Gaussian-blurred ($\sigma = 2$), and sharpened result.

Example 2

<https://visionbook.mit.edu/derivatives.html#sharpening-filter>

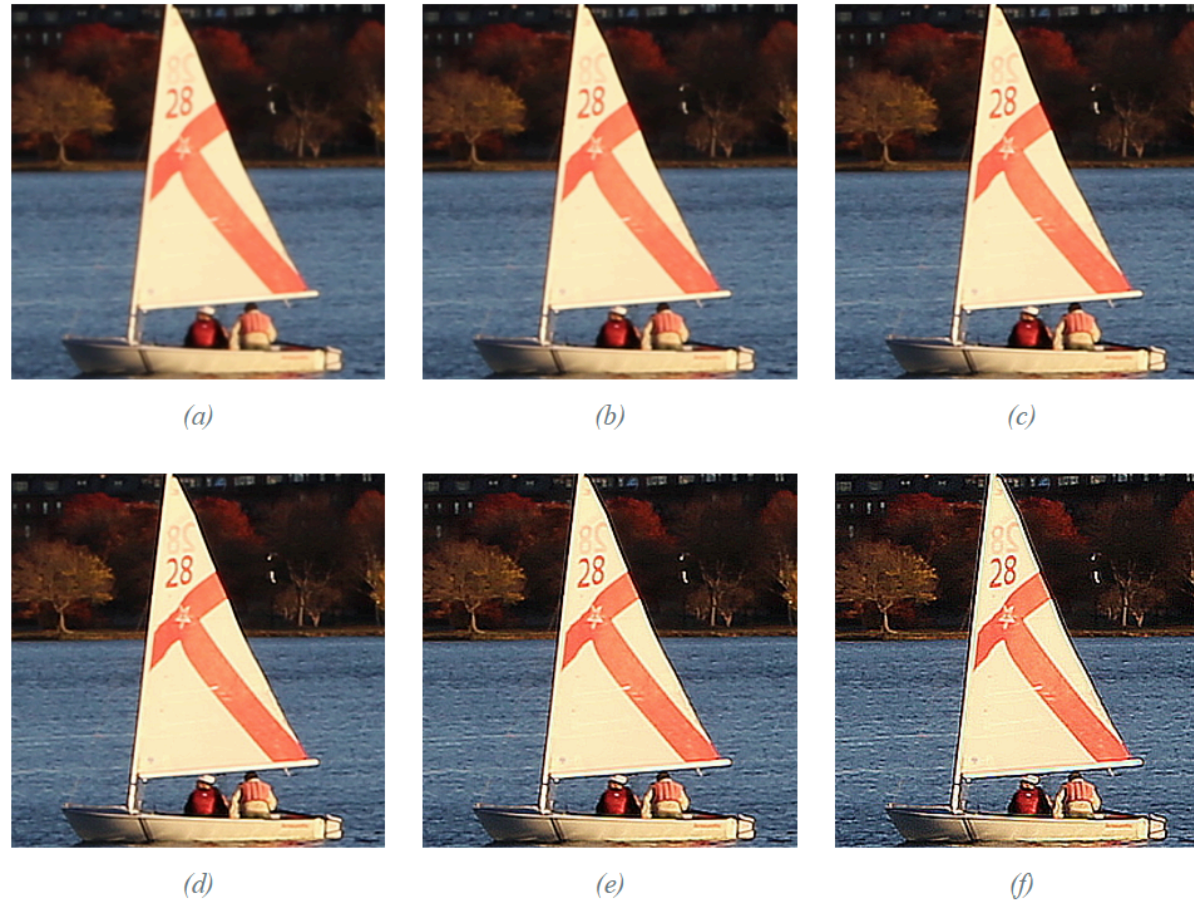


Figure 30: Repeated sharpening. Source: visionbook

Conclusions

- **Blur** filters (Box, Gaussian, binomial) are all low-pass
 - they remove noise and allow to highlight features at different scale.
 - Gaussian and binomial have smooth, monotonic frequency responses (no ringing), box does not
 - Gaussian filters have convenient properties: separable, self-Fourier-transform, variance-additive under convolution
- **Gradients**: almost all image structure (except the mean) lives in the gradients, exploited for editing/inpainting
 - to reduce noise, **Gaussian derivatives** combine smoothing and differentiation at a chosen scale σ
 - **LoG** give rotation-invariant edge detectors

- MIT Vision Book, Ch. 17 (Blur Filters), Ch. 18 (Image Derivatives): <https://visionbook.mit.edu/series.html>

- We now have scale-invariant filters and can extract several useful features
- We have rotation-invariance but only for some filters

We need to add (**Lecture 6**):

- Rotation-invariance: filter bank with multiple filters with different orientations
- Scale-space: image pyramids for multiscale representations
- discuss issues with sampling and aliasing