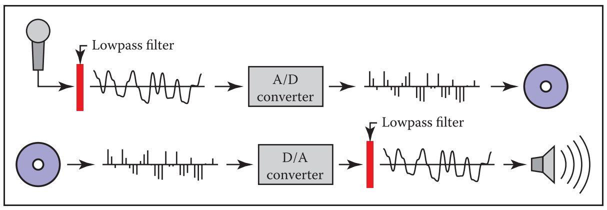
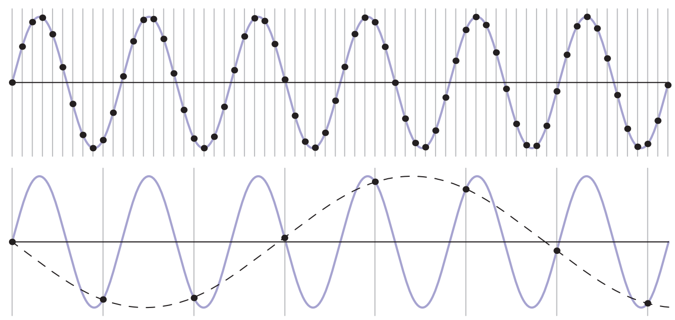
Motivation: Digital Audio

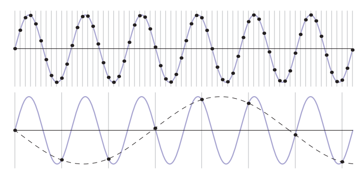
- Acquisition of images takes a continuous object and converts this signal to something digital
- Two types of artifacts:
  - **Undersampling** artifacts: on acquisition side
  - **Reconstruction** artifacts: when the samples are interpreted



Undersampling Artifacts



Shannon-Nyquist Theorem
(not needed for the exam)

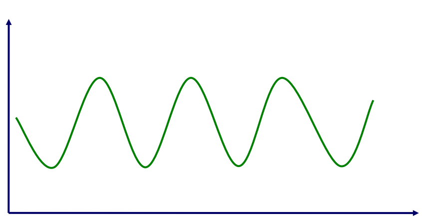


- The sampling frequency must be **double** the highest frequency of the content.
- If there are any higher frequencies in the data, or the sampling rate is too low, **aliasing**, happens
  - Named this because the discrete signal “pretends” to be something lower frequency

S-N Theorem Illustrated

How many samples are enough to avoid aliasing?

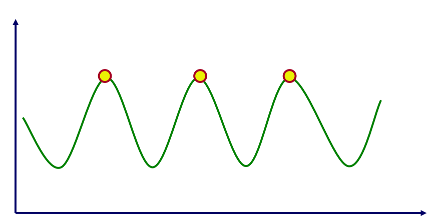
- How many samples are required to represent a given signal without loss of information?
- What signals can be reconstructed without loss for a given sampling rate?



S-N Theorem Illustrated

How many samples are enough to avoid aliasing?

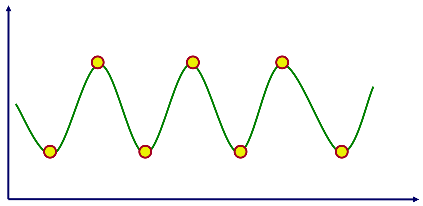
- How many samples are required to represent a given signal without loss of information?
- What signals can be reconstructed without loss for a given sampling rate?



S-N Theorem Illustrated

How many samples are enough to avoid aliasing?

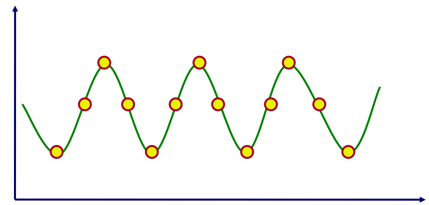
- How many samples are required to represent a given signal without loss of information?
- What signals can be reconstructed without loss for a given sampling rate?

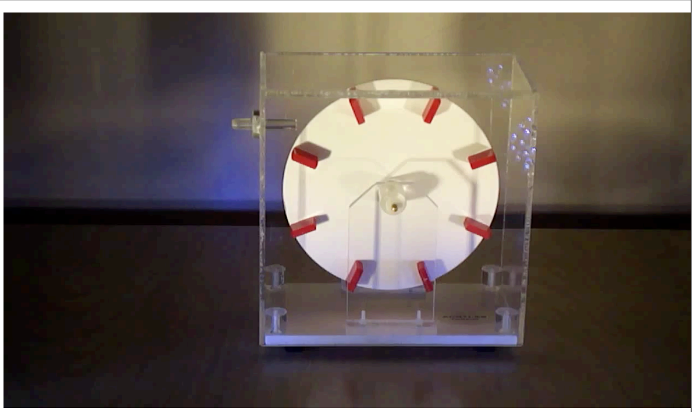


S-N Theorem Illustrated

How many samples are enough to avoid aliasing?

- How many samples are required to represent a given signal without loss of information?
- What signals can be reconstructed without loss for a given sampling rate?





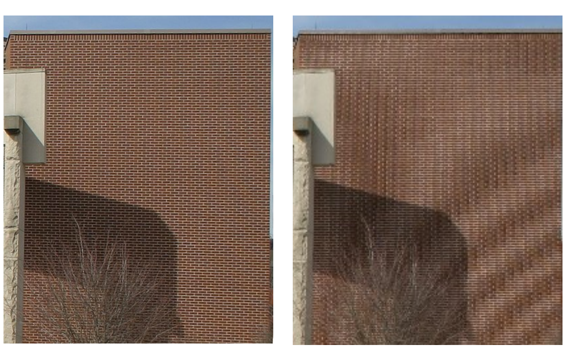
http://youtu.be/0k2lhYk6Lfs?rel=0

Aliasing in images

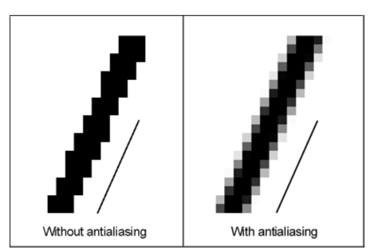
Two outcomes of under-sampling

1) Moire Pattern
2) Rasterization

Moire Patterns



Aliasing for edges



Each pixel is effected by nearby pixels
For example, even though the input image image is black/white,
We allow grey values for output pixels.

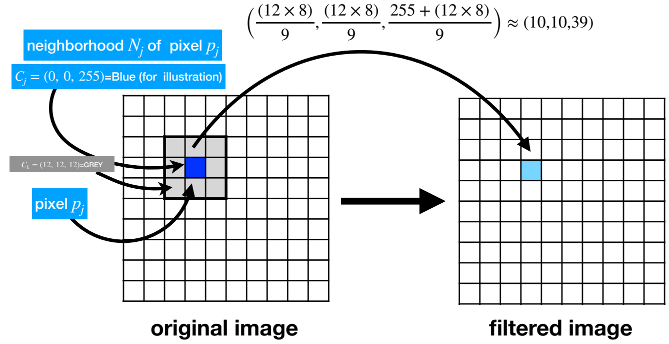
Convolution

Each pixel is effected by nearby pixels
For example, even though the image is black/white,
We allow grey values

Neighborhood Filtering (Schematic)

f(Nⱼ)=average color in this region (neighborhood)=

$$\left(\frac{(12\times 8)}{9}, \frac{(12\times 8)}{9}, \frac{255+(12\times 8)}{9}\right) \approx (10,10,39)$$

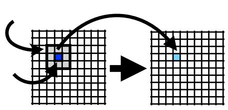


An Example: Mean Filtering

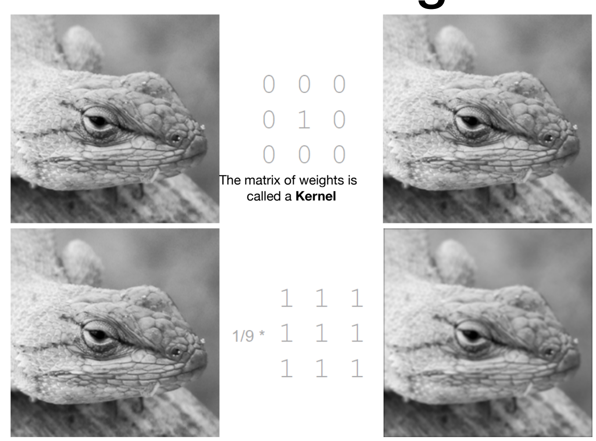
- Mean filters sum all of the pixels in a local neighborhood N_i and divide by the total number, computing the average pixel.
- Said another way, we replace each pixel as a linear combination of its neighbors (with equal weights!)
- To find the new color of a pixel *j,* we will look at N_j , defined as the (say) 3×3 neighborhood of the pixel p_j, and set
- Where the N_i is a square, we call these **box** filters
- Think about it as a weighted average:

$$f(N_i) = \sum_{\textbf{pixels } p_k \textbf{ in the region } N_j} w_k C_k = \sum_{\textbf{pixels } p_k \textbf{ in the region } N_j} \frac{1}{9} C_k$$

- The weights $w_1 \ldots w_k$ are convex combination. Meaning that they are all positive, and $w_1 + w_2 + \ldots w_k = 1$. For example, $w_1 = w_2 = w_3 = \frac{1}{3}$ **(convex combination)**
- Remember: The input matrix and the output matrix have the same size (in this case). This is **not** rescaling.
- Refer to the geogebra app https://www.geogebra.org/m/cetpvwaw
- The term filter is very common, but might be very confusing. We don't necessarily filter out anything.



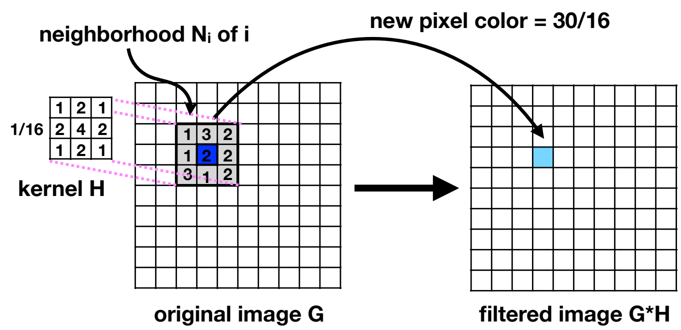
Box Filtering



Box Filtering

Convolution

- This process of adding up pixels multiplied by various weights is called **convolution.** We denote the result by **(confusion warning)** the symbol ***** **See example below.**



Kernels

- Convolution employs a rectangular grid of coefficients, (that is, weights) known as a **kernel**
- Kernels are like a neighborhood mask, they specify which elements of the image are in the neighborhood and their relative weights.
- A kernel is a set of weights that is applied to corresponding input samples that are summed to produce the output sample.
- For **smoothing** purposes, the sum of weights must be 1 (convex combination)

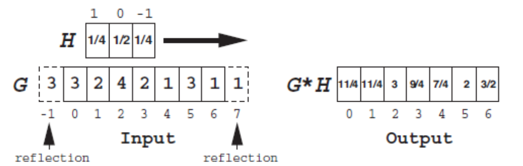
$$\frac{1}{9}\begin{pmatrix}1&1&1\\1&1&1\\1&1&1\end{pmatrix} \quad \frac{1}{13}\begin{pmatrix}1&1&1\\1&5&1\\1&1&1\end{pmatrix} \quad \frac{1}{37}\begin{pmatrix}1&1&1&1&1\\1&2&2&2&1\\1&2&5&2&1\\1&2&2&2&1\\1&1&1&1&1\end{pmatrix}$$

One-dimensional Convolution

- Can be expressed by the following equation, which takes a filter H and convolves it with G:

$$\hat{G}[i] = (G * H)[i] = \sum_{j=i-n}^{i+n} G[j]H[i-j],\ i \in [0, N-1]$$

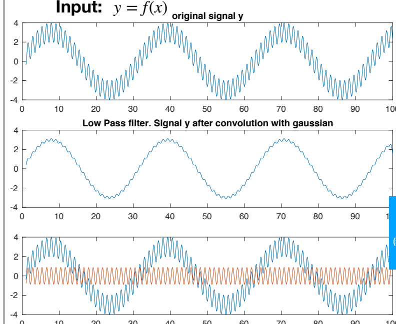
- Equivalent to sliding a window



Low-pass and high-pass filtering

The smoothing operation is always a low pass filter.
Only lower frequencies could pass.
It removes higher frequencies from the input.

Input: $y = f(x)$



We convolved the original signal f(x) a smoothing kernel H.
For example

$$g(x) = \frac{f(x-1) + f(x) + f(x+1)}{3}$$

The output of the smoothing operation
$g(x) = f(x) * H$
The higher frequencies are less noticeable:
we need to move a lot (in x) to notice a large different in y

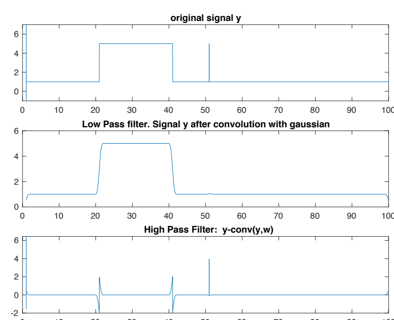
New idea: High-pass filter.
$h(x) = f(x) - g(x)$
Only high frequencies pass
(shown: Original signal (blue) and the result of the high pass filter (red))

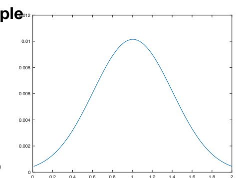
We remove (subtract) from the signal all lower frequencies

Twitter - could move very fast, but only small distances

Woofer - moves slowly but cold cover large distances

Low pass and hight pass filters - another example

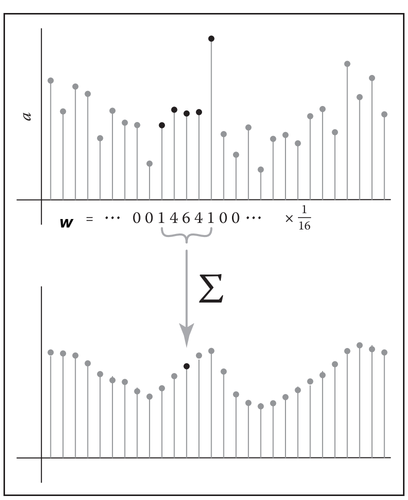




We convolved the original signal y with this gaussian

Convolution is a Moving, Weighted Average

- Getting used to the new notation: $b[i] = \frac{1}{3}(\ a[i-1] + a[i] + a[i+1]\ ) \quad \forall i$
- is similar to writing $b = a \star w$, where $b[i] = (a \star w)[i] = \sum_{j=1}^{3} a[i-j+2] \cdot w[j]$ and w[1]=w[2]=w[3]=1/3
- Commonly $(a \star w)[i] = \sum_{j=i-r}^{j=i+r} a[j]w[i-j]$
- For example, w[-1]=w[0]=w[1]=1/3
- Note that we did not define exactly what are the first and last values



2-Dimensional Version

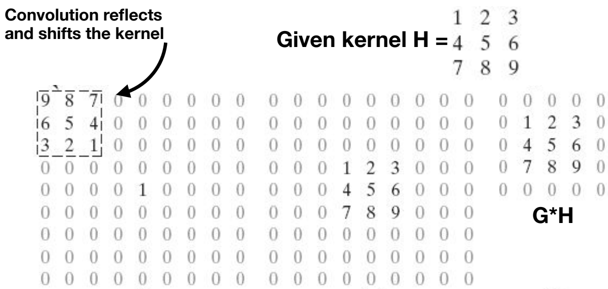
- Given an image a and a kernel b with $(2r+1)^2$ values, the convolution of a with b is given below as a*b:

$$(a \star b)[i,j] = \sum_{i'=i-r}^{i+r} \sum_{j'=j-r}^{j+r} a[i',j']\, b[i-i', j-j']$$

- The (i-i') and (j-j') terms can be understood as reflections of the kernel about the central vertical and horizontal axes.
- The kernel weights are multiplied by the corresponding image samples and then summed together.

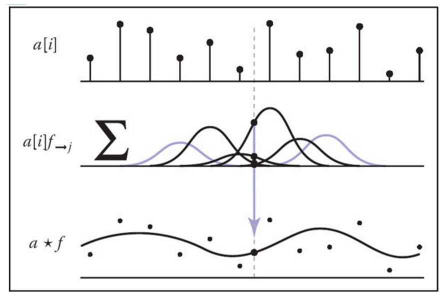
A Note on Indexing

- Convolution **reflects** the filter to preserve orientation.
- **Correlation** does **not** have this reflection.
  - But we often use them interchangeably since most kernels are symmetric!!



Convolution Can Also Convert from Discrete to Continuous

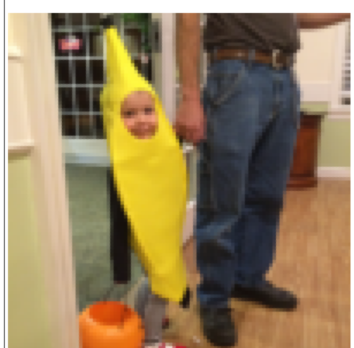
- Discrete signal a
- Continuous filter f
- Output a*f defined on positions x as opposed to discrete pixels i



$$(a \star f)(x) = \sum_{i=\lceil x-r \rceil}^{\lfloor x+r \rfloor} a[i] f(x-i)$$

B g

Filtering helps to reconstruct the signal better when rescaling



Inverse Rescaling

Reconstructed w/ Discrete-to-Continuous

Types of Filters: Smoothing

Smoothing Spatial Filters

- Any weighted filter with positive values will smooth in some way, examples:

$\frac{1}{9} \times$

1	1	1
1	1	1
1	1	1

$\frac{1}{16} \times$

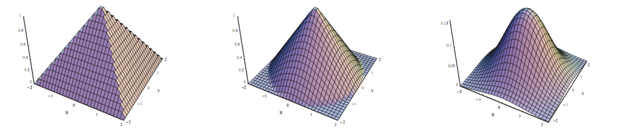
1	2	1
2	4	2
1	2	1

- Normally, we use integers in the filter, and then divide by the sum (computationally more efficient)
- These are also called **blurring** or **low-pass** filters

Smoothing Kernels

$f(x,y) = -\alpha \cdot \max(|x|,|y|)$

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



(a) Pyramid. (b) Cone. (c) Gaussian.

$f(x,y) = -\alpha \cdot \sqrt{x^2+y^2}$

(a) Pyramid.

1	2	3	2	1
2	4	6	4	2
3	6	9	6	3
2	4	6	4	2
1	2	3	2	1

(b) Cone.

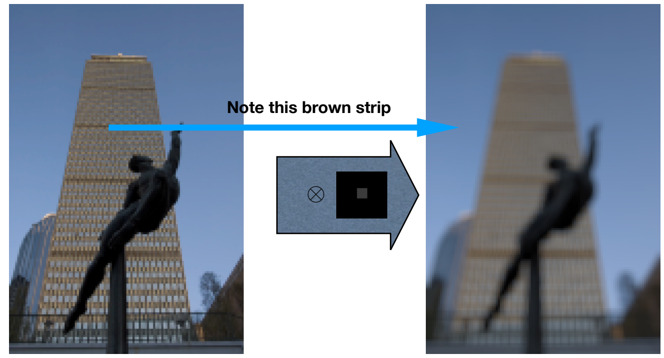
0	0	1	0	0
0	2	2	2	0
1	2	5	2	1
0	2	2	2	0
0	0	1	0	0

(c) Gaussian.

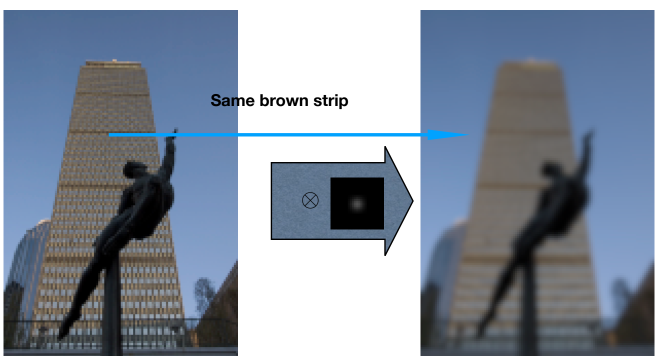
1	4	7	4	1
4	16	28	16	4
7	28	49	28	7
4	16	28	16	4
1	4	7	4	1

Table 6.1. Discretized kernels.

Box Filter



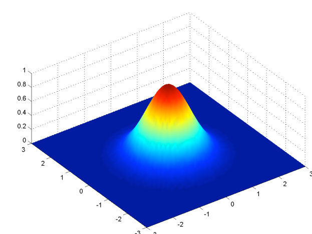
Gaussian Filter



Gaussians

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

- Gaussian kernel is parameterized on the standard deviation σ
- Large σ's reduce the center peak and spread the information across a larger area
- Smaller σ's create a thinner and taller peak
- Gaussians are smooth everywhere.
- Gaussians have infinite **support**
  - >0 everywhere
- But often truncate to 2σ or 3σ
- Volume =1 (sum of weights =1)



http://en.wikipedia.org/wiki/Gaussian_function

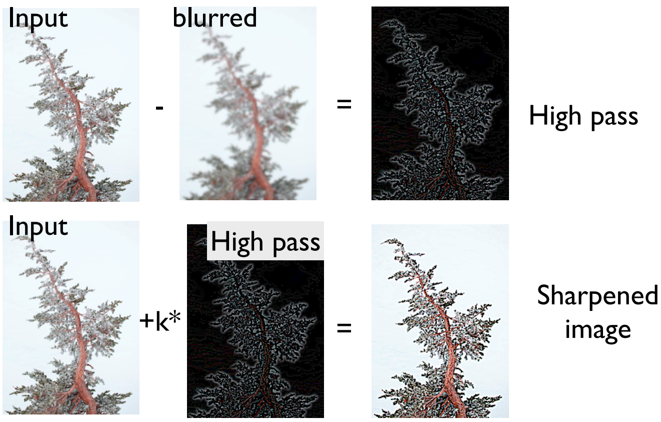
Smoothing Comparison

(a) Source image. (b) 17 × 17 Box. (c) 17 × 17 Gaussian.

Figure 6.10. Smoothing examples.

Types of Filters: Sharpening

Sharpening (Idea)



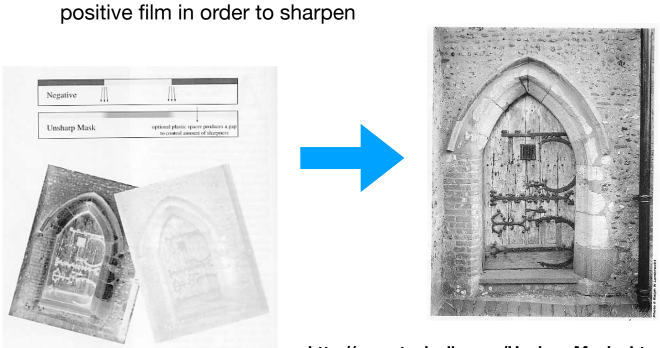
Another example

Original Image,   Imaged convolved

Left: difference (only boundaries are non-black)
Right   Imaged minus differences convolved

Unsharp Masks

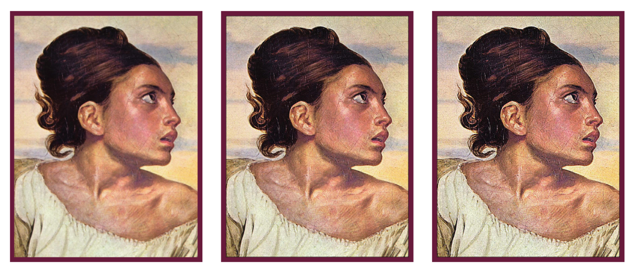
- Sharpening is often called "unsharp mask" because photographers used to sandwich a negative with a blurry positive film in order to sharpen



http://www.tech-diy.com/UnsharpMasks.htm

Edge Enhancement

- The parameter α controls how much of the source image is passed through to the sharpened image.



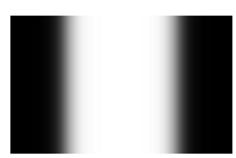
(a) Source image. (b) $\alpha = .5$. (c) $\alpha = 2.0$.

Figure 6.20. Image sharpening.

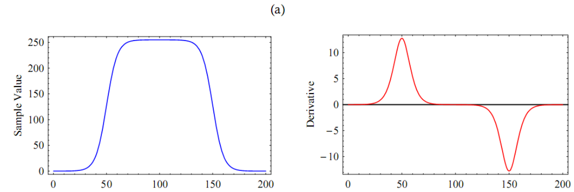
Defining Edges

- Sharpening uses negative weights to enhance regions where the image is changing rapidly
  - These rapid transitions between light and dark regions are called **edges**
- Smoothing reduces the strength of edges, sharpening strengthens them.
  - Also called **high-pass** filters
- Idea: smoothing filters are weighted averages, or integrals. Sharpening filters are weighted differences, or derivatives!

Edges



(a)



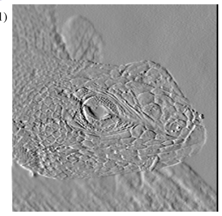
(b) (c)

Figure 6.11. (a) A grayscale image with two edges, (b) row profile, and (c) first derivative.

Taking Derivatives with Convolution (just in case you studied calculus. Not required)

$$\frac{\partial}{\partial x} f(x,y) \approx \frac{1}{2} f(x+1) - f(x+1)$$

0	0	0
−1	0	1
0	0	0



0	−1	0
0	0	0
0	1	0

Gradients with Finite Differences

(just in case you studied calculus. Not required)

- These partial derivatives approximate the image gradient, ∇I.
- Gradients are the unique direction where the image is changing the most rapidly, like a slope in high dimensions
- We can separate them into components kernels G_x, G_y. $\nabla I = (G_x, G_y)$

$$\nabla I(x,y) = \begin{pmatrix} \delta I(x,y)/\delta x \\ \delta I(x,y)/\delta y \end{pmatrix}.$$

$$G_x = [1, 0, -1] \qquad G_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix};$$

$$\nabla I = \begin{pmatrix} \delta I/\delta x \\ \delta I/\delta y \end{pmatrix} \simeq \begin{pmatrix} I \otimes G_x \\ I \otimes G_y \end{pmatrix}.$$

Figure 6.12. Image gradient (partial).

(a) Source Image.

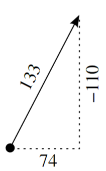
128	187	210	238	251
76	121	193	225	219
66	91	110	165	205
47	81	83	119	157
41	59	63	75	125

(b) $\delta I/\delta x$.

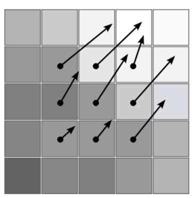
	117	104	26	
	44	74	95	
	36	38	74	

(c) $\delta I/\delta y$.

	−96	−100	−73	
	−40	−110	−106	
	−32	−47	−90	



(d) Center sample gradient.



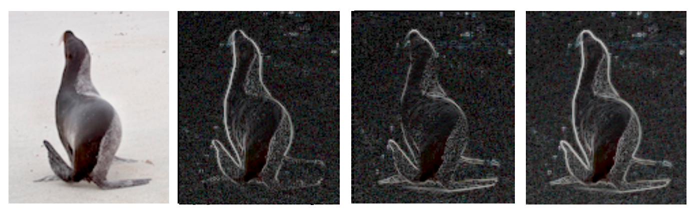
(e) Gradient.

	151	144	77	
	59	133	142	
	48	60	117	

(f) Magnitude of gradient.

Figure 6.14. Numeric example of an image gradient.

Gradients G_x, G_y



$|G_x|$ $|G_y|$ $|G|$

$$|G| = \sqrt{(G_x^2 + G_y^2)}$$

Second Derivatives (Sharpening, almost)

- Partial derivatives in x and y lead to two kernels:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

and, similarly, in the y-direction we have

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

Compare with Sharpening filter: unbalanced counts!

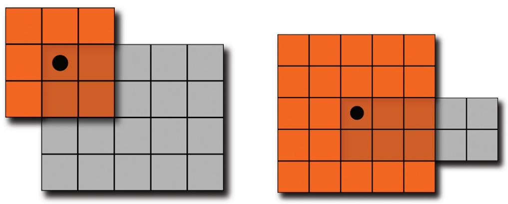
$$\begin{bmatrix} -\alpha & -\alpha & -\alpha \\ -\alpha & (9+8\alpha) & -\alpha \\ -\alpha & -\alpha & -\alpha \end{bmatrix}$$

1	1	1
1	-9	1
1	1	1

Boundaries

Handling Image Boundaries

- What should be done if the kernel falls off of the boundary of the source image as shown in the illustrations below?



(a) Kernel at $I(0,0)$. (b) Kernel larger than the source.

Figure 6.4. Illustration of the edge handling problem.

Handling Image Boundaries

- When pixels are near the edge of the image, neighborhoods become tricky to define
- Choices:

1. Shrink the output image (ignore pixels near the boundary)
2. Expanding the input image (padding to create values near the boundary which are “meaningful”)
3. Shrink the kernel (skip values that are outside the boundary, and reweigh accordingly)

Boundary Padding

- When one pads, they pretend the image is large and either produce a constant (e.g. zero), or use circular / reflected indexing to tile the image:

(a) (b) (c)

Figure 6.5. (a) Zero padding, (b) circular indexing, and (c) reflected indexing.