Lecture Notes: Image Processing

Subhransu Maji and the TAs

February 26, 2025

Contents

1 Overview

We will cover ways to represent a digital image and to modify an image *after* it has already been digitized. The goal is to make the information easier to visualize. For example, we might want to reduce noise in the image, improve contrast, or remove motion blur from a photograph. The process is ill-posed and requires estimating missing data, so we cannot always be certain of the final result. Nevertheless, these techniques are widely used.

We will discuss simple methods for improving contrast by manipulating pixel brightness values. We will then introduce convolutions, a fundamental operator in image processing that serves as the basis for many image editing operations, such as denoising, smoothing, and sharpening.

Beyond editing, convolutions are essential for extracting image representations, including edges, corners, and scale-invariant features. They also form the basis of convolutional neural networks, which is the technology behind many modern computer vision applications.

2 Analog to Digital Conversion

Before an image is digitized, it exists as a continuous entity. It comprises a continuous range of wavelengths and extends over a two-dimensional space. At each point in this space, the image has a continuous range of power values, representing variations in intensity.

To simplify the representation, we often consider only a brightness image. This can be viewed as a two-dimensional function over a plane, where each point corresponds to a specific location, and the brightness varies continuously across the surface. The key challenge is to efficiently represent this continuous two-dimensional function in a digital format.

To digitize an image, we need to apply sampling strategies that convert the continuous representation into a discrete one. There are two primary aspects of sampling: *spatial sampling* and *brightness sampling*. Spatial sampling determines how the continuous image is divided into discrete pixels. Important considerations include:

- How many pixels should be used to represent the image?
- What arrangement of pixels provides the best representation of the original image?

Brightness sampling focuses on discretizing the range of brightness values. Key questions include:

- How many discrete brightness levels should be used?
- How should the spacing between brightness values be determined?

For video, an additional consideration is *time sampling*. Since video consists of a sequence of images over time, we must determine:

- How frequently frames should be captured (frame rate)?
- How does the sampling rate affect motion perception and temporal resolution?

Efficient sampling strategies are crucial for ensuring that the digitized image or video retains sufficient detail while minimizing storage and computational costs.

3 Contrast Enhancement by Brightness Mapping

The contrast of an image can often be enhanced by adjusting the distribution of brightness values.

Contrast stretching Contrast stretching (Figure 1) scales and shifts the brightness values to occupy the full range of values. If the range of brightness values are from 0 to 255, then an intensity value of x is mapped to x' as:

$$x' = \text{round}\left(\frac{x - \min(I)}{\max(I) - \min(I)} \times 255\right) \tag{1}$$

where $\min(I), \max(I)$ are the minimum and maximum values in the image I.

Logarithmic mapping In some cases, the intensity values in an image may be compressed in a way that makes it difficult to distinguish details in darker regions. Logarithmic mapping is a technique that enhances contrast by compressing high-intensity values while expanding lower-intensity values. The transformation is given by:

$$x' = c\log(1 + x) \tag{2}$$

where c is a scaling constant that ensures the output intensity values remain within the desired range. This transformation is particularly useful when dealing with images that have a high dynamic range, as it makes dimmer details more visible without overexposing bright areas.

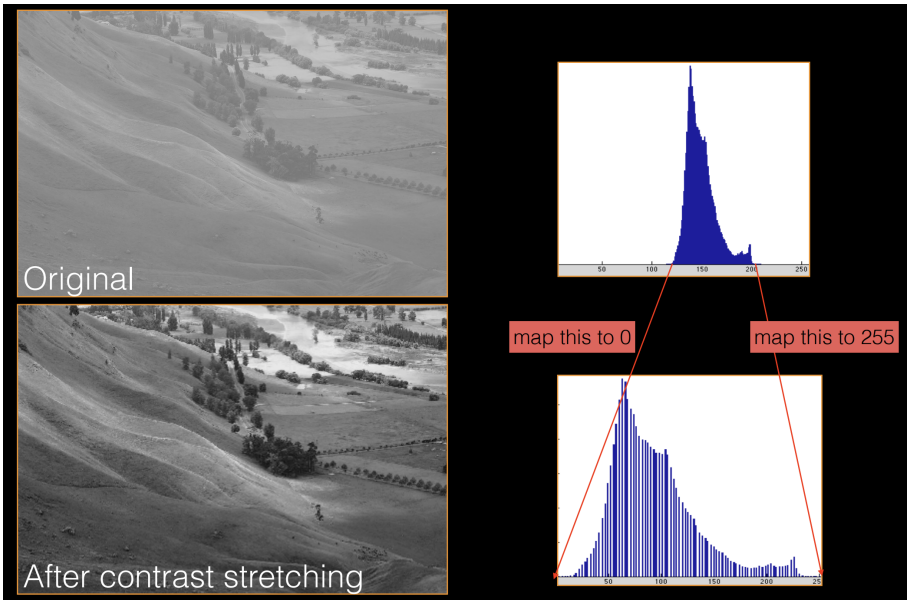


Figure 1: The effect of applying contrast-stretching to an image (left) and the change in distribution of brightness values (right). In this specific example, the brightness values of the original image centers around 150, with minimum and maximum brightness values around 120 and 200. Objects in the image appears to be indistinguishable. After applying contrast stretching, the image brightness values gets mapped (scaled) to [0,255] and centers (shifted) around 80; the shape of the distribution stays roughly the same. The new image has a higher contrast and the details in the image are more distinguishable than that of the original image.

Gamma correction Gamma correction is another non-linear transformation used to adjust brightness and contrast. It is particularly useful for compensating for the non-linear response of display devices. The gamma transformation is defined as:

$$x' = cx^{\gamma} \tag{3}$$

where c is a normalization constant (often chosen so that the maximum intensity remains at 255), and γ is the gamma value. When $\gamma < 1$ the transformation brightens the image by enhancing lower intensity values, while $\gamma > 1$ darkens the image by suppressing lower intensity values. Typical values for gamma range from 0.5 to 2.5, depending on the desired effect.

Histogram equalization Histogram equalization is a widely used technique for improving image contrast by redistributing intensity values so that they are more uniformly spread across the available range (Figure 2). The method works by computing the cumulative distribution function (CDF) of the intensity values and using it to map the original intensity values to a new set of values. The transformation is given by:

$$x' = \text{round}\left(\frac{CDF(x) - \min(CDF)}{\max(CDF) - \min(CDF)} \times 255\right) \tag{4}$$

where $CDF(x)$ is the cumulative sum of the histogram values up to intensity x. This mapping enhances the contrast in images where intensity values are clustered in a narrow range, making features more distinguishable. Histogram equalization is particularly effective for images with poor contrast due to uneven illumination.

4 Convolution and Filtering

Convolution is a fundamental operation in image processing, used for filtering, feature extraction, and various transformations. It involves applying a kernel (or filter) to an image to produce a transformed output.

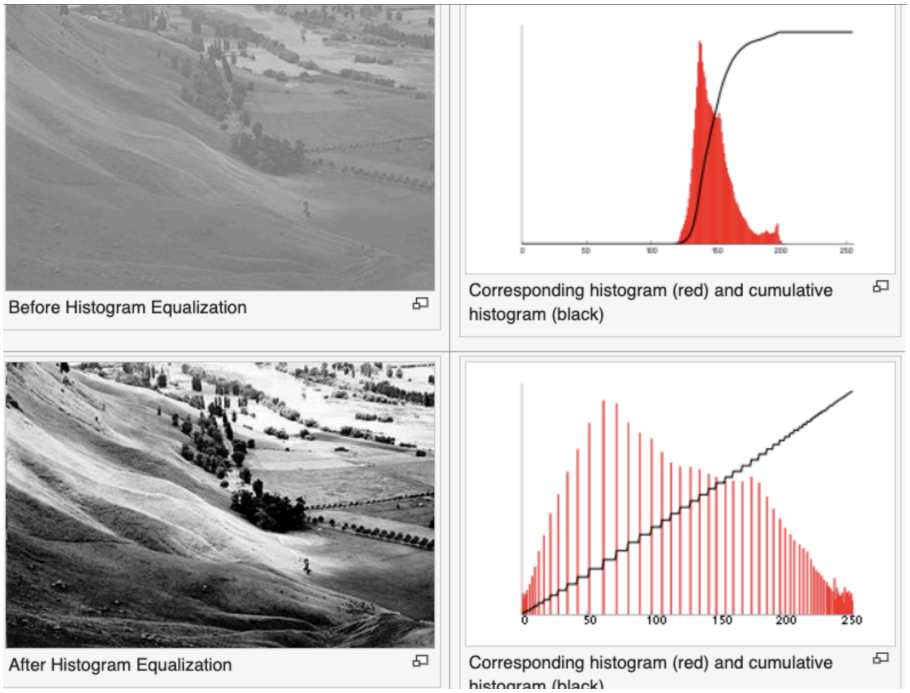


Figure 2: The effect of applying histogram equalization to the image and the change in distribution of brightness value. The graph in the top right shows the distribution of brightness value (red) and the cumulative distribution function (black) of the original image (top left). After applying histogram equalization to the image (bottom left), the distribution of brightness value (red) gets "smoothened" out and appears to be more uniform and the cumulative distribution function (black) increases uniformly as the brightness value increases.

4.1 Mathematical Definition

Given an image $f[i, j]$ and a kernel (or filter) $g[i, j]$, the convolution operation is defined as:

$$(f * g)[i, j] = \sum_{m=-k}^{k} \sum_{n=-k}^{k} f[i - m, j - n]g[m, n] \tag{5}$$

where k is the half-size of the kernel. This operation slides the flipped kernel over the image, computing a weighted sum of pixel values at each location. It is closely related to the cross-correlation (or filtering) operation defined as:

$$(f * g)[i, j] = \sum_{m=-k}^{k} \sum_{n=-k}^{k} f[i + m, j + n]g[m, n] \tag{6}$$

Unlike convolution, the filter is not flipped before multiplication.

4.2 Examples

Common convolution kernels include:

- **Identity kernel:** Leaves the image unchanged.

$$g = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

- **Box blur (averaging filter):** Smooths the image by averaging neighboring pixels.

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

- **Gaussian blur:** A weighted smoothing filter that reduces high-frequency noise.

$$g = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

- **Sharpening filter:** Enhances edges by emphasizing differences. More on sharpening later.

$$g = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

4.3 Output Size and Padding

When performing convolution or cross-correlation on an image, special handling is required for image boundaries. This is because part of the filter (kernel) may extend beyond the edges of the image, where pixel values are undefined.

Different implementations of convolution and correlation may produce outputs of varying sizes as show in Figure 3, depending on the selected **mode**:

- **'full'** – The output size is the sum of the sizes of f (image) and g (kernel).
- **'same'** – The output size is the same as f
- **'valid'** – The output size is the difference of the sizes of f and g, meaning only positions where the kernel fully overlaps the image are considered.

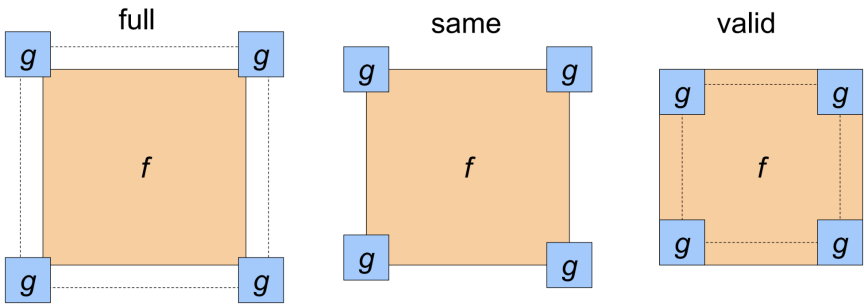


Figure 3: The output size of the image after convolving image f with filter(kernel) g using modes **'full'**, **'same'**, and **'valid'**. The solid lines shows the original image size whereas the dashed lines show output image size for the respective modes. Modes **'full'** and **'same'** requires padding the original image as the filter g does not fully overlap with the original image f.

To produce 'full' or 'same' outputs, padding is required. These padding schemes are commonly used:

- **Zero (or constant) padding** – Pads the image with zeros or a specified constant outside the boundaries.
- **Replicate padding** – Extends the boundary pixels outward by repeating their values.
- **Reflect padding** – Mirrors the image at the boundary, preserving continuity.
- **Wrap padding** – Treats the image as periodic, wrapping around the edges.

Refer to the lecture slides to see the impact of different padding schemes on image smoothing. For example, zero padding can introduce dark artifacts near the edges, which may be undesirable in some applications.

4.4 Coordinate System for Images

The most commonly used coordinate system follows a Cartesian grid with the origin at the top-left corner. The x-axis extends from left to right, while the y-axis extends from top to bottom.

When images are loaded as NumPy arrays, they follow a row-major order, meaning the first index refers to the row (y), and the second index refers to the column (x). Thus, a pixel at location (x, y) is accessed as `array`$[y, x]$, where $x \in [0, \text{width} - 1]$ and $y \in [0, \text{height} - 1]$.

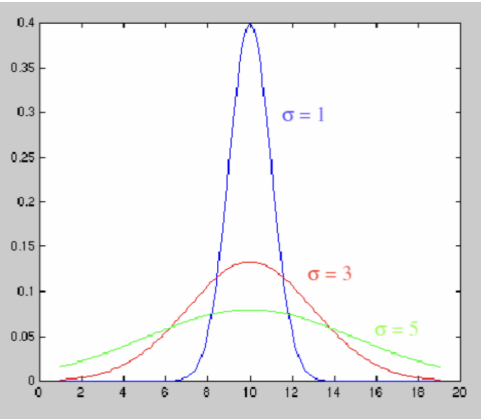
However, not all libraries follow this convention. For example, Matplotlib's `plot` function uses the bottom-left corner as the origin. Therefore, always check the coordinate system before using a function.

4.5 Implementing Convolution with Different Padding in Python

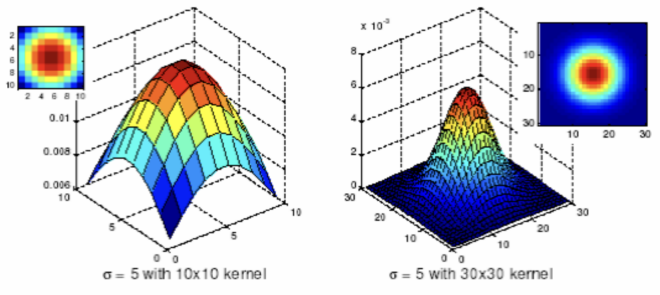
```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
from scipy.ndimage import convolve

# Load a grayscale image
image = cv2.imread("sample.jpg", cv2.IMREAD_GRAYSCALE)

# Define a simple 3x3 averaging kernel
kernel = np.array([[1, 1, 1],
                   [1, 1, 1],
                   [1, 1, 1]], dtype=np.float32) / 9  # Normalize kernel

# Apply convolution with different padding strategies
zero_padded = convolve(image, kernel, mode='constant', cval=0.0)
replicate_padded = convolve(image, kernel, mode='nearest')
reflect_padded = convolve(image, kernel, mode='reflect')
wrap_padded = convolve(image, kernel, mode='wrap')

# Display results
plt.figure(figsize=(10, 8))

plt.subplot(2, 3, 1)
plt.imshow(image, cmap='gray')
plt.title("Original Image")
plt.axis("off")

plt.subplot(2, 3, 2)
plt.imshow(zero_padded, cmap='gray')
plt.title("Zero Padding")
plt.axis("off")

plt.subplot(2, 3, 3)
plt.imshow(replicate_padded, cmap='gray')
plt.title("Replicate Padding")
plt.axis("off")

plt.subplot(2, 3, 4)
plt.imshow(reflect_padded, cmap='gray')
plt.title("Reflect Padding")
plt.axis("off")

plt.subplot(2, 3, 5)
plt.imshow(wrap_padded, cmap='gray')
plt.title("Wrap Padding")
plt.axis("off")

plt.show()
```

5 Denoising and Smoothing Filters

Denoising techniques aim to reduce unwanted noise in images while preserving important details. A common source of noise is the addition of independent noise at pixel. The noise distribution is often assumed to a zero-mean Guassian distribution which is a good approximation of noise resulting from a



(a) The effect of standard deviation σ



(b) Color maps of Gaussian kernels with different size

Figure 4: (a) The smaller the σ the more concentrated the Gaussian distribution. Gaussian value approaches 0 after three standard deviations, 3σ, away from the mean/center. (b) The Gaussian function has infinite support, but the values are discretized in finite kernels.

sum of many independent noisy observations. However, other forms are noise are possible, such as image compression artifacts, salt-and-pepper noise, etc.

Averaging local neighborhoods are a good way to reduce Gaussian noise, while the Median filter is more effective at reducing salt-and-pepper noise. These are defined as follows:

- Mean (box) filter: Averages pixel values in a local region to smooth the image.
- Gaussian filter: Applies a weighted average that gives more importance to central pixels.
- Median filter: Replaces each pixel with the median of the surrounding pixels. The median filter is a non-linear operator and *cannot* be implemented as convolutions or filtering.

5.1 Gaussian Filter

Mathematically, a Gaussian filter with standard deviation σ is defined as:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{7}$$

The parameter σ controls the size of the neighborhood used in the averaging and plays the same role as the width of the box in the box filter. To obtain an effective discrete approximation of the Gaussian filter, the kernel size should be chosen in proportion to σ. A common heuristic is to set the kernel's half-width to $3\sigma + 1$, as the Gaussian values beyond this range are negligible (Figure 4a).

5.2 Mathematical Properties

The Gaussian filter has several useful properties:

- It removes high-frequency components from the image, effectively smoothing it.
- Convolution of a Gaussian function with itself results in another Gaussian.
  - Applying multiple convolutions with a Gaussian of standard deviation σ produces the same result as a single convolution with a Gaussian of larger standard deviation.
  - Convolving an image twice with a Gaussian kernel of standard deviation σ is equivalent to convolving once with a Gaussian of standard deviation $\sqrt{2}\sigma$.
- **Separable kernel**: The 2D Gaussian function can be decomposed into the product of two 1D Gaussian functions as seen below:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} = \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}}\right) \tag{8}$$

– This property is called separability and enables more efficient computation. Since convolution is associative, applying a 2D Gaussian filter can be performed as two 1D convolutions. For Gaussian this corresponds to first applying a 1D Guassian along the rows (x-direction), followed by applying a 1D convolution along the columns (y-direction)

– This reduces the computational complexity from $O(N^2)$ to $O(2N)$ for a $N \times N$ kernel.

6 Contrast Enhancement and Sharpening Filter

Earlier we saw how to improve contrast by remapping the brightness values in an image. Now let's look at another approach that does this using convolutions with takes into account the local neighborhood of each pixel to modify its intensity.

6.1 Sharpening Filter

Blurring an image reduces the contributions of high-frequency components. We can observe this by computing the difference between an image I and a version smoothed using a Gaussian filter g_σ:

$$I_{\text{detail}} = I - I * g_\sigma. \tag{9}$$

What happens if we add the detail image back to I, scaled by a factor $\alpha > 0$?

$$I' = I + \alpha I_{\text{detail}}. \tag{10}$$

The result is an image I' where the effect of the details is enhanced. This operation is called **sharpening**. Since all these operations are linear, we can rewrite sharpening as a convolution of the image with a single filter, known as the **sharpening filter**, as follows:

$$\begin{aligned} I' &= I + \alpha I_{\text{detail}} \\ &= I + \alpha(I - I * g_\sigma) \\ &= I(1+\alpha) - \alpha I * g_\sigma \\ &= I(1+\alpha) * e - \alpha I * g_\sigma \quad // \text{ where } e \text{ is the impulse filter} \\ &= I * \underbrace{((1+\alpha) * e - \alpha g_\sigma)}_{\text{Sharpening filter}}. \end{aligned}$$

Thus, the sharpening filter can be expressed as the difference between scaled versions of the impulse filter and the Gaussian filter. The parameters α and σ control the strength of the sharpening effect.

This sharpening filter resembles a filter that is the difference between two Gaussian filters, as the impulse filter is the limiting case of the Gaussian filter as $\sigma \to 0$. The Difference of Gaussians filter also closely approximates a Laplacian filter (Figure 5), which is the second derivative of the Gaussian filter.

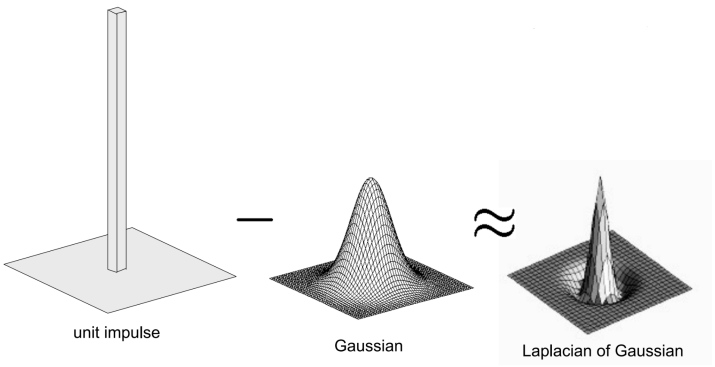


Figure 5: Sharpening filter as an approximation of Laplacian of Gaussian. The Laplacian of Gaussian has a Mexican hat shape, positive in the center, negative around it, and then fading to zero, which detects and responds strongly to regions with rapid intensity change, i.e. the details, to create the sharpening effect.

7 Hybrid Images

Hybrid images [Oliva and Torralba, SIGGRAPH 2006] are a single image that combines details from two different images in such a way that:

- At a close distance, the image appears as one object.
- From a far distance, the image appears as another object.

This effect is achieved by blending the low-frequency components of one image with the high-frequency components of another image. Hybrid images take advantage of how the human visual system processes high and low frequencies differently.

7.1 Mathematical Derivation

Mathematically this can be computed as:

1. *Compute the Low-Frequency Image.* The low-frequency components of an image I_1 are obtained by convolving it with a Gaussian filter g_σ, which acts as a low-pass filter:

$$I_{\text{low}} = I_1 * g_\sigma \tag{11}$$

   where:

   - g_σ is a Gaussian filter with standard deviation σ, which removes high-frequency components.
   - I_{low} retains only smooth, large-scale structures.

2. *Compute the High-Frequency Image.* The high-frequency components of an image I_2 are obtained by subtracting its low-frequency version from the original:

$$I_{\text{high}} = I_2 - (I_2 * g_\sigma) \tag{12}$$

   where:

   - $I_2 * g_\sigma$ is the low-frequency version of I_2, computed similarly to Step 1.
   - The subtraction removes smooth details, leaving only edges and textures.

3. *Combine Low- and High-Frequency Images.* The final hybrid image is formed by adding the low-frequency and high-frequency images:

$$I_{\text{hybrid}} = I_{\text{low}} + I_{\text{high}} \tag{13}$$

Since the human visual system perceives high frequencies better up close and low frequencies better at a distance, the image appears different depending on the viewing distance as shown in figure 6.

7.2 Implementation Details

- The choice of σ (Gaussian blur size) affects the blending quality.
- The images should be **aligned properly** to avoid ghosting artifacts.
- The high-pass component should be **contrast-adjusted** to avoid over-brightening or darkening.

8 Edge Detection and Derivative Filters

The goal of edge detection is to identify sudden changes (discontinuities) in an image. Edges encode most of the semantic and shape information of an image, making them a more compact representation than raw pixels. Ideally, an edge-detected image should resemble an artist's line drawing, although an artist also incorporates object-level knowledge.

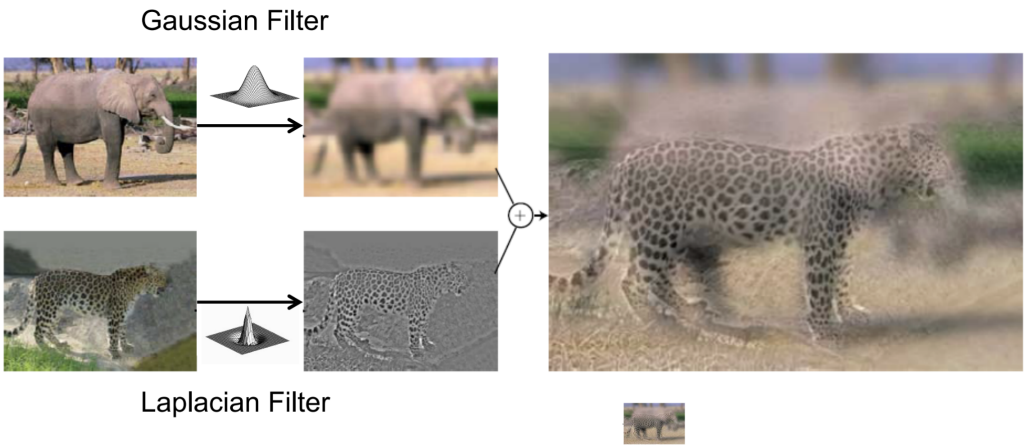


Figure 6: A hybrid image of an elephant and a leopard. Combining a Gaussian filtered image of an elephant and a Laplacian filtered image of a leopard makes the hybrid image to appear as an elephant from a far and a leopard from close.

8.1 Mathematical Definition of Edges

An edge is a region where the image intensity function f changes rapidly (Figure 7. This can be detected using derivative filters which compute image gradients. The image gradient is defined as:

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

where:

- $G_x = \frac{\partial f}{\partial x}$ represents the horizontal gradient.
- $G_y = \frac{\partial f}{\partial y}$ represents the vertical gradient.

The gradient points in the direction of the most rapid intensity increase. The gradient direction is given by:

$$\theta = \arctan(G_y, G_x)$$

The gradient magnitude, which measures the strength of an edge, is given by:

$$G = \sqrt{G_x^2 + G_y^2} \tag{14}$$

8.2 Basic Derivative Filters

The simplest derivative filters are:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

These compute the centered pixel-wise differences along the horizontal and vertical directions.

8.3 Noise and Smoothing Before Edge Detection

Unlike smoothing filters, derivative filters are highly sensitive to image noise. To mitigate noise, it is beneficial to first smooth the image using a Gaussian filter before applying the derivative filter as shown in Figure 8.

Since convolution is associative, applying a Gaussian filter followed by a derivative filter is equivalent** to convolving the image with a single filter that is the convolution of the Gaussian filter with the derivative filter. These derivatives of Gaussian filters simultaneously smooth the image and compute derivatives

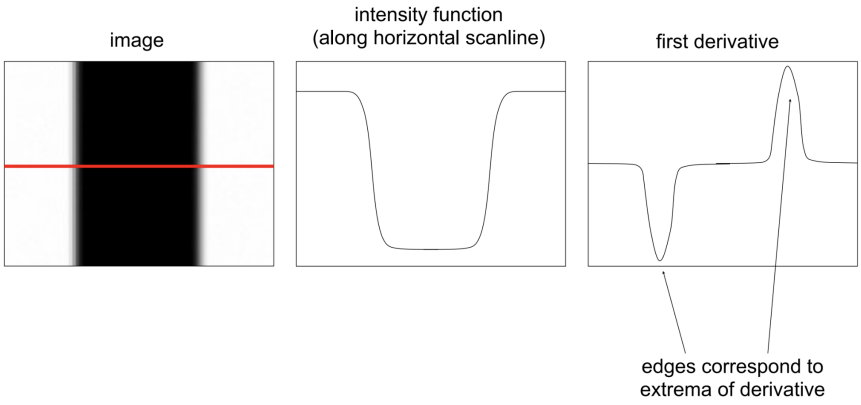


Figure 7: The intensity function f (middle) and its first derivative $\frac{\partial f}{\partial x}$ (right) along the red horizontal scanline in the image (left). The sudden decrease and increase in the intensity function locates the edges along on the horizontal scanline. Applying a horizontal derivative filter to the image gives us the horizontal gradient $\frac{\partial f}{\partial x}$ which indicates locations of rapid intensity changes in the horizontal direction.

(Figure 9). By varying σ (the Gaussian standard deviation), we can not only reduce the impact of noise, but also detect edges at different scales.

A commonly used edge detection filter is the Sobel operator, which approximates a derivative of Gaussian filter:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

9 Summary

Smoothing filters remove high-frequency components (acting as a low-pass filter).

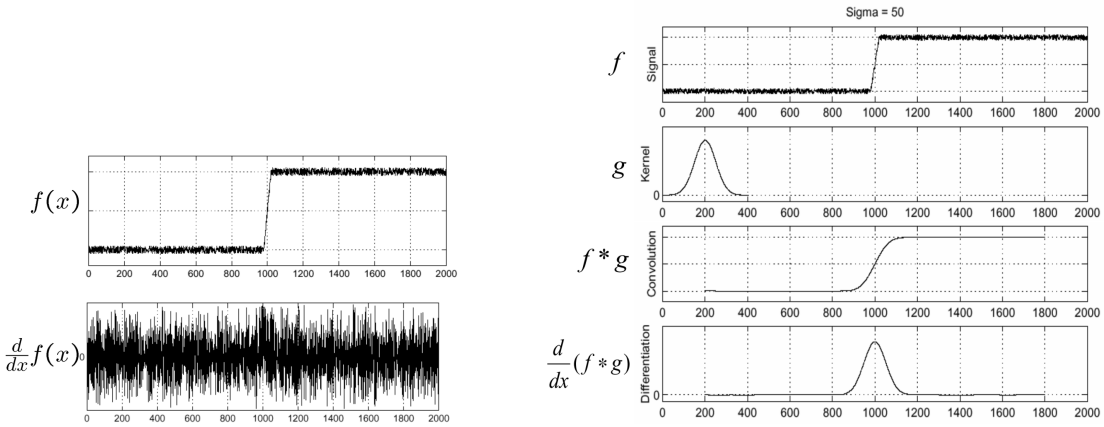
- Can the values of a smoothing filter be negative? *No, as it averages neighboring values.*
- What should the values sum to? *One, ensuring that constant regions remain unchanged.*

Derivative filters on the other hand detect edges:.

- Can the values of a derivative filter be negative? *Yes, since they compute differences.*
- What should the values sum to? *Zero, ensuring no response in constant regions.*
- Where do we get high absolute values? *At points of high contrast (edges).*

10 Optional readings

1. Binary edges in images can be computed by thresholding the gradient magnitude. The more advanced *Canny edge detector* performs a multi-stage edge detection process, which includes thinning by retaining local maxima and hysteresis thresholding, where two thresholds are used to filter weak edges.
2. Advanced techniques leverage large datasets of human-annotated object boundaries to develop robust edge detectors trained using machine learning. For example, the recent "Segment Anything" algorithm (`https://segment-anything.com`) is trained on a billions of annotated object masks and can detect semantic boundaries in general images across a wide range of domains. Take a look at the Berkeley Segmentation Dataset (BSDS) for some early attempts to model semantic boundary detection using machine learning.



(a) Edge detection with noisy image

(b) Denoising before edge detection

Figure 8: (a) Applying edge detection directly to a noisy image vs (b) Denoising the image before edge detection. (a) The intensity function f has a lot noise. These abrupt low intensity changes will cause large gradients in the first derivative $\frac{df}{dx}$ making it difficult to locate the peak that corresponds to the edge in the intensity function f. (b) Applying a Gaussian filter g on f outputs a denoised intensity function $f * g$, making it easy to locate the peaks that corresponds to edges from the first derivative $\frac{d}{dx}(f * g)$.

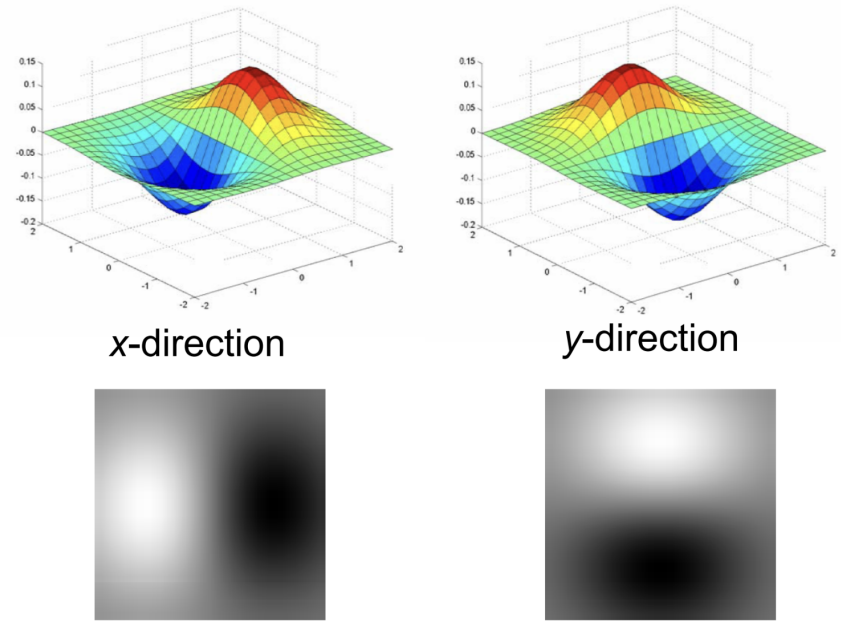


Figure 9: Color maps of Horizontal (left) and vertical (right) Gaussian derivative filters, which simultaneously smooth the image and compute derivatives. In constant (and noisy) regions where edges are not present, the concave and convex parts of the Gaussian derivative filter tends to cancel out or have response values close to 0. For regions where edges are present, the response values of the filter will increase in magnitude and reach the maximum when the edge separates the concave and convex of the Gaussian derivative filter