3D Computer Vision HW1

Lachin

November 2023

1

1.1

The plots for signals are are follows:

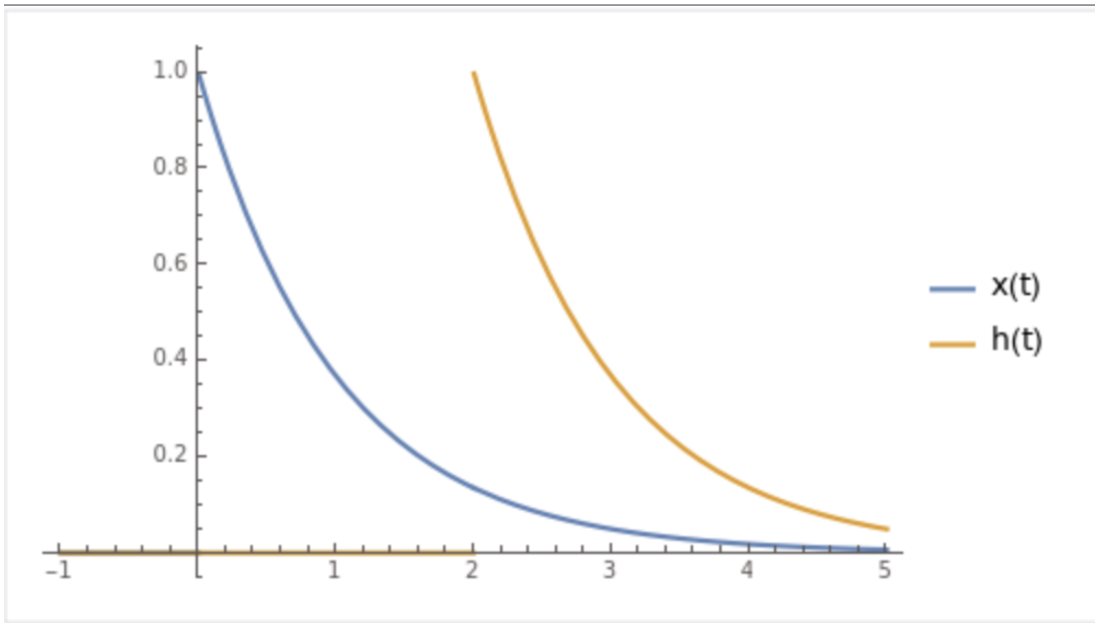


Figure 1: The plot of $x(t)$ and $h(t)$.

To find the explicit piecewise expression for the convolution of $x(t)$ and $h(t)$, we would typically break the problem into intervals based on the step functions:

1. For $t < 0$, both $x(t)$ and $h(t)$ are zero, so the convolution is zero.
2. For $0 \leq t < 2$, $h(t)$ is still zero, so the convolution is zero.
3. For $t \geq 2$, both $x(t)$ and $h(t)$ are non-zero, and the convolution integral needs to be evaluated from 2 to t.

The convolution for $t \geq 2$ is given by the integral:

$$\int_2^t e^{-\tau} e^{-(t-\tau-2)} d\tau$$

This integral simplifies to:

$$\int_2^t e^{-t} e^{2} d\tau = e^{2-t}(t-2)$$

Therefore, the piecewise function for the convolution $x(t) * h(t)$ is:

$$x(t) * h(t) = \begin{cases} 0 & \text{for } t < 2, \\ e^{2-t}(t-2) & \text{for } t \geq 2. \end{cases}$$

1.2

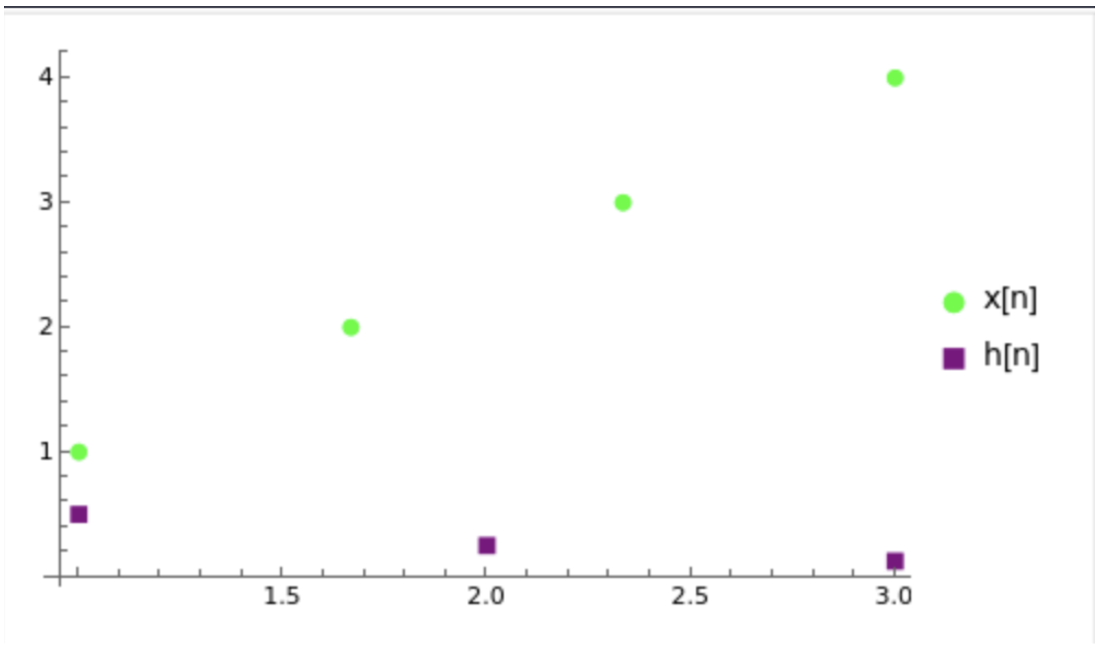


Figure 2: Plots of the discrete signals $x[n]$ and $h[n]$.

The convolution of the discrete signals $x[n] = [1, 2, 3, 4]$ and $h[n] = [0.5, 0.25, 0.125]$ is given by the sequence:

$$x[n] * h[n] = [0.5, 1.25, 2.125, 3, 1.375, 0.5]$$

This result represents the discrete convolution of the two sequences, where each term of the resulting sequence is the sum of products of the terms from $x[n]$ and $h[n]$, offset by the index of $h[n]$ and summed over the range of overlap.

1.3

Signal $X(f) = \text{sinc}(f)$:

The sinc function is defined as $\text{sinc}(f) = \frac{\sin(\pi f)}{\pi f}$. The inverse Fourier transform of the sinc function is well-known and corresponds to a rectangular pulse in the time domain.

$$x(t) = \frac{\sqrt{\pi/2}(\text{Sign}[1-t] + \text{Sign}[1+t])}{2}$$

This expression can be simplified to a rectangular function, which is typically represented as:

$$x(t) = \text{rect}\left(\frac{t}{2}\right) = \begin{cases} 1 & \text{if } |t| \leq \frac{1}{2}, \\ 0 & \text{otherwise.} \end{cases}$$

Signal $H(f)$:

The given $H(f)$ is a rectangular function in the frequency domain, which is 1 for $|f| \leq 1$ and 0 otherwise. The inverse Fourier transform of a rectangular function in the frequency domain is a sinc function in the time domain. The result obtained is:

$$h(t) = \frac{\sqrt{2/\pi}\sin(t)}{t}$$

This is the sinc function in the time domain, and it's important to note that this is not the normalized sinc function used in signal processing, which includes a factor of π. The sinc function in signal processing is defined as $\text{sinc}_{\text{sp}}(t) = \frac{\sin(\pi t)}{\pi t}$, but the result here is without the π factor because the rectangular function $H(f)$ is not normalized.

1.4

The convolution of the time-domain signals obtained from the inverse Fourier transforms of $X(f)$ and $H(f)$ is given by the integral:

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

where $x(t)$ is the inverse Fourier transform of $X(f) = \text{sinc}(f)$, resulting in a rectangular pulse, and $h(t)$ is the inverse Fourier transform of $H(f)$, which is a rectangular function in the frequency domain, resulting in a sinc function in the time domain. The convolution of these two functions is not trivial and does not simplify to a standard form easily. However, it is known that the convolution of a rectangular pulse with a sinc function results in a band-limited interpolation of the rectangular pulse, often referred to as a "sinc interpolation" of the pulse.

2

2.1

Gaussian Blur:

Gaussian blur is a smoothing technique that reduces image noise and detail by applying a Gaussian function. It is widely used due to its properties that closely resemble the way optical systems blur images.

- How it Works: A Gaussian kernel is applied to each pixel in the image. This kernel is a weighted average where the weights decrease with distance from the central pixel, according to the Gaussian distribution. The standard deviation of the Gaussian, denoted as σ, controls the extent of the blurring.

- Mathematical Representation: The Gaussian function in two dimensions is given by:

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

Median Filter:

The median filter is a non-linear digital filtering technique, often used to remove noise from an image. It is particularly effective at preserving edges while removing noise.

- How it Works: A window slides over the image, and for each window position, the median of the pixel values within the window is calculated. The central pixel is then replaced with this median value.

- Advantages: The median filter does not create new pixel values, making it better for preserving the details of the image while removing noise, especially 'salt and pepper' noise.

2.2

Salt-and-pepper noise is characterized by random white and black pixels. Two common noise removal methods are:

Median Filter: This filter replaces each pixel with the median value of its neighbors, effectively preserving edges while removing noise.

- Pros: Preserves edges well; maintains original pixel values.
- Cons: May remove fine details; higher computational cost.

Morphological Filters: Morphological operations like opening and closing can remove noise based on shape, enhancing image structure. Dilation and erosion are the two basic morphological operators, where dilation selects the brightest value in the neighborhood of the structuring element and erosion selects the darkest value in a neighborhood.

- Pros: Customizable to noise shape; can improve object structure.
- Cons: Potential to alter object shapes; requires careful structuring element selection.

Comparison: Both methods are effective, but the median filter is often preferred for its simplicity. It is better at preserving details, especially edges, but is more computationally intensive. Morphological filters offer flexibility and are ideal for noise with known characteristics but require more expertise to use effectively.

2.3

Gaussian noise is a form of statistical noise with a normal distribution, affecting images and electronic signals. It introduces random variations in brightness or color that can degrade data quality.

To counteract Gaussian noise, a Gaussian kernel is utilized. This image filter averages pixel values, weighting them according to a Gaussian distribution, thereby smoothing the image and diminishing noise.

Process Overview:

1. **Kernel Construction:** A Gaussian kernel is formed, centered on the pixel in question, with the surrounding values decreasing according to the Gaussian function.
2. **Convolution:** This kernel is convolved with the image, recalculating each pixel as a weighted average of its neighbors, effectively blurring the image to reduce noise.
3. **Balancing Act:** The standard deviation (σ) of the Gaussian function is tuned to balance the blur—higher σ values increase blur and noise suppression but may also soften important image details.

Employing a Gaussian kernel helps smooth out the image, reducing noise while maintaining the integrity of the visual data, making it an effective approach for noise reduction.

3

3.1

In image processing, applying a kernel to an image is a standard operation. When the size of the output image needs to be the same as the input, we use *same* convolution. This requires padding the original image with zeros around the border before applying the kernel.

The Convolution Operation:

Given an image I and a kernel K, the *same* convolution operation at each pixel (x, y) in I is defined as:

$$I'(x,y) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} K(i,j) \cdot I(x+i, y+j)$$

where I' is the resulting image after applying the convolution.

In our case:

3.2

The kernel depicted is a standard edge-detection kernel, more specifically, a Laplacian filter. This kernel is crucial in image processing for enhancing the edges within an image.

The Laplacian filter is a second-order derivative filter that is used to find areas of rapid change (edges) in images. It operates by convolving the kernel with the image, which emphasizes regions of rapid intensity change.

$$K = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

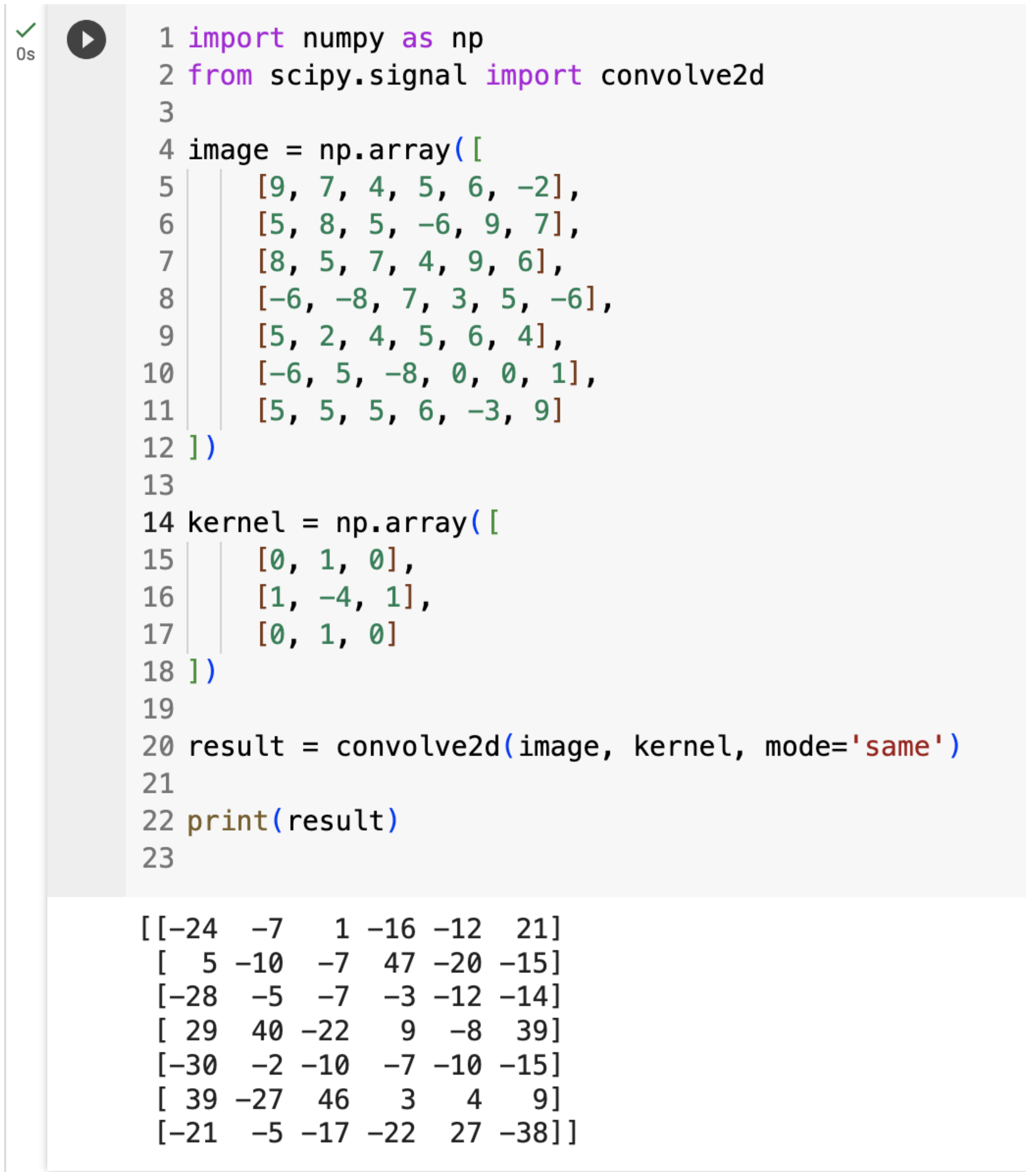
The result of applying this kernel is an image that has pronounced edges, which is useful in various image processing applications such as feature extraction, object detection, and pattern recognition.

3.3

The Gaussian blur kernel is widely used for noise reduction in images. It applies a weighted average to the pixels, with the weights following a Gaussian distribution.

The Gaussian kernel has values determined by the Gaussian function which are used for averaging pixel values. A 3x3 example is given by:

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


0s

```
import numpy as np
from scipy.signal import convolve2d

image = np.array([
    [9, 7, 4, 5, 6, -2],
    [5, 8, 5, -6, 9, 7],
    [8, 5, 7, 4, 9, 6],
    [-6, -8, 7, 3, 5, -6],
    [5, 2, 4, 5, 6, 4],
    [-6, 5, -8, 0, 0, 1],
    [5, 5, 5, 6, -3, 9]
])

kernel = np.array([
    [0, 1, 0],
    [1, -4, 1],
    [0, 1, 0]
])

result = convolve2d(image, kernel, mode='same')

print(result)

```

```
[[-24  -7   1 -16 -12  21]
 [  5 -10  -7  47 -20 -15]
 [-28  -5  -7  -3 -12 -14]
 [ 29  40 -22   9  -8  39]
 [-30  -2 -10  -7 -10 -15]
 [ 39 -27  46   3   4   9]
 [-21  -5 -17 -22  27 -38]]
```

Application Example:
Consider a noisy pixel with its surrounding pixels:

$$\begin{bmatrix} 100 & 200 & 100 \\ 200 & 0 & 150 \\ 120 & 230 & 130 \end{bmatrix}$$

Applying the Gaussian kernel to the central pixel:

$$\text{New pixel value} = \frac{1}{16}(1 \cdot 100 + 2 \cdot 200 + 1 \cdot 100 + 2 \cdot 200 + 4 \cdot 0 + 2 \cdot 150 + 1 \cdot 120 + 2 \cdot 230 + 1 \cdot 130) \approx 116$$

Hence, the Gaussian blur effectively reduces noise by smoothing out variations in pixel intensities while preserving the edge integrity of the image.