Image Processing and Analysis – Math Part: Convolution and Filtering

Ivar Farup

Based upon Chapter 4 of Broughton and Bryan's *Discrete Fourier Analysis and Wavelets* and Maciej Piętka's Lecture Notes from 2010

Overview

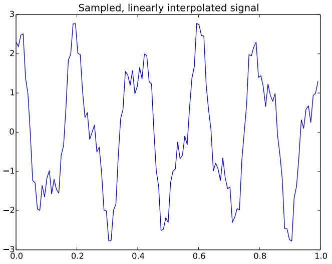
- One-dimensional convolution
- Convolution theorem and filtering
- 2D convolution – filtering images

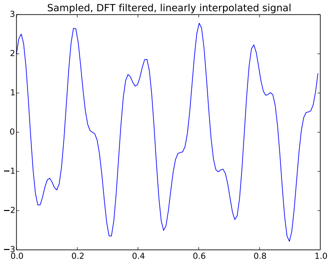
One-Dimensional Convolution
Signal Denoising Example

- Analog signal on $t \in [0,1]$:

$$x(t) = 2\cos(2\pi \cdot 5t) + 0.8\sin(2\pi \cdot 12t) + 0.3\cos(2\pi \cdot 47t)$$

- Sampled at $\Delta T = 1/128$
- Remove signal components above 40 Hz cutoff frequency





One-Dimensional Convolution

- Convolution is a different approach to filtering out certain frequencies of a signal or image
- Convolution is performed entirely in the time domain

One-Dimensional Convolution

Special Case: Low-Pass Filtering

- The goal: to remove (at least partially) high frequencies
  - Leave low frequencies unchanged
  - Smooth out any short-term flucutations
- Can be achieved with a moving average
  - Replace each sample x_k with the average value of itself and nearby samples
  - The output is a smoothed signal with much reduced high-frequency content
  - Low frequency waveforms almost unaffected

One-Dimensional Convolution

Example: Three-Point Weighted Moving Average

- Let $\mathbf{x} = (x_0, x_1, \ldots, x_{N-1})$ be the sampled signal
- Replace the sampled signal $\mathbf{x}$ with $\mathbf{w}$:

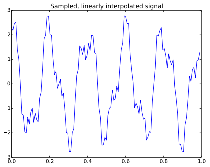
$$w_k = (3x_k + 2x_{k-1} + x_{k-2})/6$$

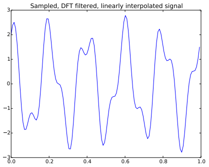
- Wrap around (extend periodically) the time series to solve the problem at $k = 0, 1$
- $x_k = x_{k \mod N}$ for any $k \in \mathbb{Z}$
- Thus $x_{-1} = x_{N-1}$ and $x_{-2} = x_{N-2}$
- The sum of any N consecutive components of $\mathbf{x}$ is the same

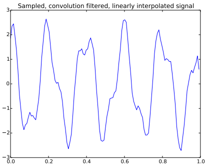
$$\sum_{k=m}^{m+N-1} x_k = \sum_{k=0}^{N-1} x_k$$

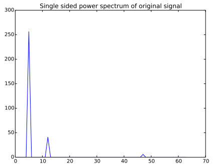
One-Dimensional Convolution

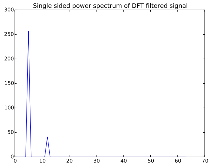
Example: Three-Point Weighted Moving Average

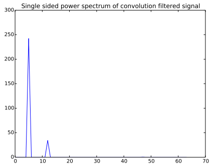












One-Dimensional Convolution

Example: Three-Point Weighted Moving Average

	Original	WMA	FFT
$k = 5$	256	247.55	256
$k = 12$	40.96	33.68	40.96
$k = 47$	5.76	0.43	0

Energy contribution $|X_k|^2/N$ for $k = 5, 12, 47$

WMA (Weighted Moving Average)

- can be computed much faster than DFT
- is less accurate

One-Dimensional Convolution

Example: Three-Point Weighted Moving Average

WMA is linear, $\mathbf{w} = \mathbf{M}\mathbf{x}$:

$$\begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_{N-1} \end{pmatrix} = \frac{1}{6} \begin{pmatrix} 3 & 0 & 0 & 0 & \cdots & 0 & 1 & 2 \\ 2 & 3 & 0 & 0 & \cdots & 0 & 0 & 1 \\ 1 & 2 & 3 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{N-1} \end{pmatrix}$$

One-Dimensional Convolution

Definition of Convolution

- Let $\mathbf{x}$ and $\mathbf{y}$ be vectors in $\mathbb{C}^N$ indexed from 0 to $N-1$
- Assume that $\mathbf{x}$ and $\mathbf{y}$ are extended periodically, so $x_k = x_{k \mod N}$ for $k \in \mathbb{Z}$
- The (circular discrete) convolution of $\mathbf{x}$ and $\mathbf{y}$, denoted $\mathbf{x} * \mathbf{y}$ is the vector $\mathbf{w}$ with components

$$w_r = \sum_{k=0}^{N-1} x_k y_{r-k}$$

for $0 \leq r \leq N-1$

One-Dimensional Convolution

Matrix Formulation

- If $\mathbf{w} = \mathbf{x} * \mathbf{y}$, then $\mathbf{w} = \mathbf{M_y}\mathbf{x}$:

$$\begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_{N-1} \end{pmatrix} = \begin{pmatrix} y_0 & y_{N-1} & y_{N-2} & \cdots & y_1 \\ y_1 & y_0 & y_{N-1} & \cdots & y_2 \\ y_2 & y_1 & y_0 & \cdots & y_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{N-1} & y_{N-2} & y_{N-3} & \cdots & y_0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{pmatrix}$$

- The 3-point moving average from the previous example can be represented as

$$\mathbf{w} = \mathbf{x} * \mathbf{f}$$

with $\mathbf{f} = (3, 2, 1, 0, \cdots, 0)/6$

One-Dimensional Convolution

Convolution Properties

- Linearity: $\mathbf{x} * (a\mathbf{y} + b\mathbf{w}) = a(\mathbf{x} * \mathbf{y}) + b(\mathbf{x} * \mathbf{w})$
- Commutativity: $\mathbf{x} * \mathbf{y} = \mathbf{y} * \mathbf{x}$
- Associativity: $\mathbf{x} * (\mathbf{y} * \mathbf{w}) = (\mathbf{x} * \mathbf{y}) * \mathbf{w}$
- Periodicity: if $\mathbf{x}$ and $\mathbf{y}$ are extended periodically and $\mathbf{w} = \mathbf{x} * \mathbf{y}$, then w_r is defined for all $r \in \mathbb{Z}$ and satisfies

$$w_r = w_{r \mod N}$$

- Matrix representation: $\mathbf{w} = \mathbf{x} * \mathbf{y} = \mathbf{M}_{\mathbf{y}}\mathbf{x}$, where $\mathbf{M}_{\mathbf{y}}$ is called the *circulant matrix* for $\mathbf{y}$

Convolution Theorem and Filtering

- Filtering is a linear operation performed on input data $\mathbf{x} \in \mathbb{C}^N$
- Let $\mathbf{h} \in \mathbb{C}^N$ be a filter vector
- $\mathbf{h}$ has arbitrarily chosen components
- The operation $\mathbf{w} = \mathbf{x} * \mathbf{h}$ is called (linear) filtering with $\mathbf{w}$ being the output of the filter

$$\underbrace{\mathbf{x}}_{\text{Input}} \rightarrow \underbrace{\mathbf{h}}_{\text{Filter}} \rightarrow \underbrace{\mathbf{w} = \mathbf{x} * \mathbf{h}}_{\text{Output}}$$

- Filters can be designed to systematically alter the frequency content of the input signal

Convolution Theorem and Filtering

Remarks

- The time complexity of computing a convolution directly is quadratic, $\mathcal{O}(N^2)$ unless the filter vector has a special form, e.g.,
$$\mathbf{h} = \frac{1}{6}(3, 2, 1, 0, \ldots, 0)$$
in which case the running time is linear, $\mathcal{O}(N)$
- The convolution theorem reduces the computation to finding the DFTs of $\mathbf{x}$ and $\mathbf{h}$
- DFT has quasi-linear complexity, $\mathcal{O}(N \log N)$

- Let $\mathbf{x}, \mathbf{h} \in \mathbb{C}^N$
- Let $\mathbf{w} = \mathbf{x} * \mathbf{h}$
- Let $\mathbf{X} = DFT(\mathbf{x})$, $\mathbf{H} = DFT(\mathbf{h})$, $\mathbf{W} = DFT(\mathbf{w})$
- Then,
$$W_k = X_k H_k$$
- The DFT of the convolution is the product of the DFTs
- Simplifies the computation of the convolution

Convolution Theorem and Filtering

The Convolution Theorem – Proof

$$\begin{aligned}
W_k &= \sum_{m=0}^{N-1} e^{-2\pi ikm/N} w_m \\
&= \sum_{m=0}^{N-1} \left(e^{-2\pi ikm/N} \sum_{r=0}^{N-1} x_r h_{m-r} \right) \\
\text{(introduce } n = m - r\text{)} \quad &= \left(\sum_{r=0}^{N-1} e^{-2\pi ikr/N} x_r \right) \left(\sum_{n=-r}^{N-1-r} e^{-2\pi ikn/N} h_n \right) \\
&= \left(\sum_{r=0}^{N-1} e^{-2\pi ikr/N} x_r \right) \left(\sum_{n=0}^{N-1} e^{-2\pi ikn/N} h_n \right) \\
&= X_k H_k
\end{aligned}$$

Convolution Theorem and Filtering

Comments

- Filtering is a linear operation on input data
- Completely defined by the effect on basis vectors
- Consider discrete basic waveforms $\mathbf{E}_k$ as the basis for $\mathbb{C}^N$
- Convolve $\mathbf{E}_k$ and $\mathbf{h}$ in the frequency domain
- $DFT(\mathbf{E}_k)$ is the vector $\mathbf{X}$ with a single nonzero component $X_k = N$, while $X_l = 0$ for $l \neq k$
- The DFT of the filter output $\mathbf{w} = \mathbf{E}_k * \mathbf{h}$ also has one nonzero component, $W_k = NH_k$, and $W_l = 0$ for $l \neq k$
- The IDFT of $\mathbf{W}$ yields $\mathbf{w} = H_k \mathbf{E_k}$ with H_k being the k-th frequency component of $\mathbf{h}$

Convolution Theorem and Filtering

Comments

- We conclude that
$$\mathbf{E}_k * \mathbf{h} = H_k \mathbf{E}_k$$
- Recall that $\mathbf{E}_k$ is the sampled complex exponential $e^{2\pi ikt}$
- The basic waveform $\mathbf{E}_k$, when filtered with $\mathbf{h}$ is
  - Amplified in magnitude by $|H_k|$
  - Shifted in phase by $\alpha/2\pi$, where $\alpha = \arg(H_k)$
- In case of a general sampled signal $\mathbf{x}$, amplification and phase-shifting is done frequency by frequency (due to linearity)
- By choosing the magnitude and phase of each H_k, we can design any type of filter with a suitable frequency response (e.g., low-pass or high-pass)

Convolution Theorem and Filtering

Filter Design

- Decide on the desired frequency response
- Set H_k-s accordingly
- Compute $\mathbf{h} = IDFT(\mathbf{H})$
- Do the filtering $\mathbf{x} * \mathbf{h}$

Convolution Theorem and Filtering

Filter Design

- Disadvantage: most h_m's will be $\neq 0$
- $\mathcal{O}(N^2)$ algorithm in the time domain
- $FFT(\mathbf{x})$ and then multiplication $W_k = X_k H_k$ followed by $IFFT(\mathbf{W})$ might be faster
- Alternative: impose constraints on H_k so that only few of h_m-s are nonzero

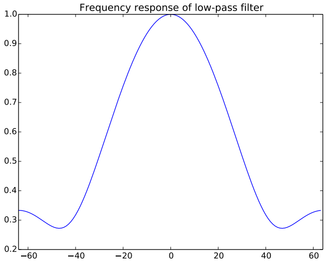
Convolution Theorem and Filtering

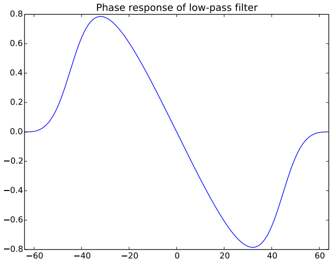
Filter Design

- Consider again the input vector $\mathbf{x} \in \mathbb{C}^{128}$ convolved with

$$\boldsymbol{\ell} = \frac{1}{6}(3, 2, 1, 0, \ldots, 0)$$

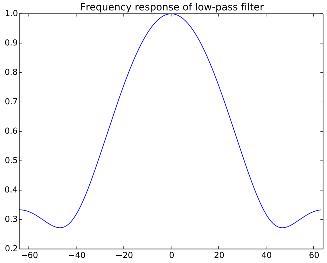
- Introduce $\mathbf{L} = DFT(\boldsymbol{\ell})$
  - $|L_k|$ is the amplification factor for frequency component k
  - $\arg(L_k)$ is the phase shift of that frequency component

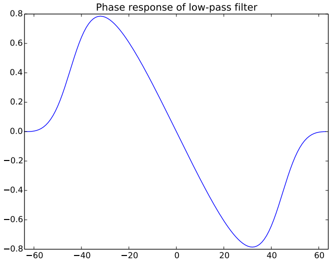




Convolution Theorem and Filtering

Filter Design





- Small frequencies almost unchanged in magnitute, $|L_k| \approx 1$ for small k
- Large frequencies diminished by a factor ≈ 0.3 (or 0.09 in energy)
- No sharp distinction between filtered out and passed frequencies
- Very low and very high frequencies almost unchanged in phase

Convolution Theorem and Filtering

Example: High-Pass Filter

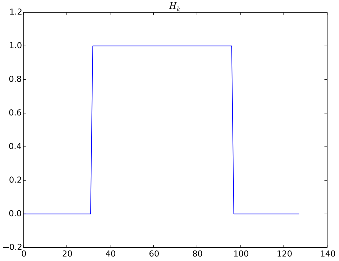
- Suppose we work with sampled signals $\mathbf{x} \in \mathbb{C}^{128}$
- A high-pass filter will block all waveforms with frequencies in the range 0 to 31
- Pass unchanged all waveforms with frequencies from 32 to the Nyquist frequency, 64
- If we consider k in the range $0 \leq k \leq 127$, then blocked frequencies are for $0 \leq k \leq 31$ and $97 \leq k \leq 127$
- Allowed frequencies are for $32 \leq k \leq 96$
- The filter in the frequency domain:

$$H_k = \begin{cases} 0 & \text{for } 0 \leq k \leq 31 \text{ and } 97 \leq k \leq 127 \\ 1 & \text{for } 32 \leq k \leq 96 \end{cases}$$

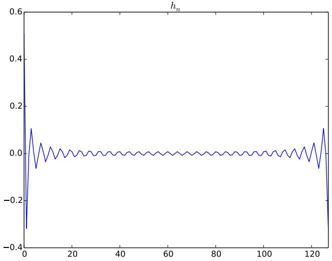
- Notice that there is no phase shift

Convolution Theorem and Filtering

Example: High-Pass Filter



Amplitude spectrum



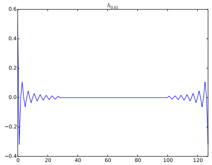
Time-domain filter

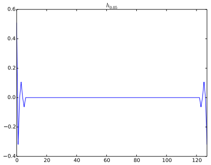
- $\mathbf{h} = IDFT(\mathbf{H})$ is real-valued (due to symmetry about $k = 64$)
- All h_m-s are $\neq 0$
- May use thresholding to eliminate som nonzero h_m values
- Thus introducing an error to both $|H_k|$ and $\arg(H_k)$

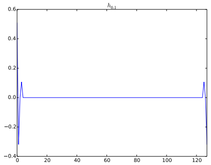
Convolution Theorem and Filtering

Example: High-Pass Filter

- Let $\mathbf{h}_\epsilon$ be the filter obtained from $\mathbf{h}$ by zeroing all its components satisfying $|h_m| < \epsilon$
- We use $\epsilon = 0.01, 0.05, 0.1$
- Corresponding filters have 29, 7, and 5 taps



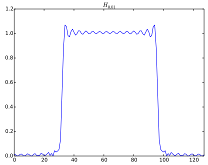


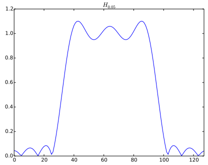


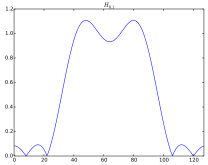
Convolution Theorem and Filtering

Example: High-Pass Filter

Corresponding DFTs:







- Increasing ϵ decreases the number of taps
- At the same time, the frequency response of the filter deviates more and more from the ideal high-pass filter

2D Convolution

Definition

- Generalisation of the one-dimensional convolution to images
- Let $\mathbf{A}, \mathbf{B} \in M_{m,n}(\mathbb{C})$ be sampled images with components $a_{r,s}, b_{r,s}$ for $0 \leq r \leq m-1$ and $0 \leq s \leq n-1$
- When necessary, treat $\mathbf{A}, \mathbf{B}$ as their periodic extension
- The (circular discrete) convolution of $\mathbf{A}$ and $\mathbf{B}$ denoted $\mathbf{A} * \mathbf{B}$ is the matrix $\mathbf{C} \in M_{m,n}(\mathbb{C})$ with entries

$$c_{p,q} = \sum_{r=0}^{m-1} \sum_{s=0}^{n-1} a_{r,s} b_{p-r,q-s}$$

2D Convolution
Properties

- Linearity
- Commutativity
- Associativity
- Periodicity (in the plane)

2D Convolution

The Convolution Theorem

- Let $\mathbf{A}, \mathbf{B} \in M_{m,n}(\mathbb{C})$
- Let $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ be the two-dimensional DFTs of $\mathbf{A}$ and $\mathbf{B}$ with components $\hat{a}_{k,l}$, $\hat{b}_{k,l}$
- Let $\hat{\mathbf{C}}$ be the DFT of $\mathbf{C} = \mathbf{A} * \mathbf{B}$
- Then, the components of $\hat{\mathbf{C}}$ are

$$\hat{c}_{k,l} = \hat{a}_{k,l}\hat{b}_{k,l}$$

2D Convolution

2D Filtering

- Is a linear operation on matrices/images
- Let $\mathbf{A} \in M_{m,n}(\mathbb{C})$ be the input image
- Let $\mathbf{D} \in M_{m,n}(\mathbb{C})$ be the filter image (often called mask)
- Filtering the image $\mathbf{A}$ is convolving it with $\mathbf{D}$
- $\mathbf{B} = \mathbf{A} * \mathbf{D}$ is the output of the filter

$$\underbrace{\mathbf{A}}_{\text{Input}} \rightarrow \underbrace{\mathbf{D}}_{\text{Mask}} \rightarrow \underbrace{\mathbf{B} = \mathbf{A} * \mathbf{D}}_{\text{Output}}$$

2D Convolution

Example: Noise Removal and Blurring

- One simple approach to image denoising is to remove high frequency waveforms by averaging each pixel with its nearest neighbours
- Analogous to moving average for 1D signals
- A suitable mask is

$$d_{r,s} = \begin{cases} 1/9 & \text{for } r, s = -1, 0, 1 \\ 0 & \text{otherwise} \end{cases}$$

- The entries of $\mathbf{A} * \mathbf{D}$ are

$$\begin{aligned}(\mathbf{A} * \mathbf{D})_{k,l} = (&a_{k-1,l-1} + a_{k-1,l} + a_{k-1,l+1} \\ &+ a_{k,l-1} + a_{k,l} + a_{k,l+1} \\ &+ a_{k+1,l-1} + a_{k+1,l} + a_{k+1,l+1})/9\end{aligned}$$

2D Convolution

Example: Noise Removal and Blurring

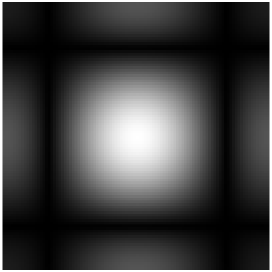
- Remember that $\mathbf{D}$ is extended periodically
- Explicit form for $\mathbf{D}$:

$$\mathbf{D} = \frac{1}{9}\begin{pmatrix} 1 & 1 & 0 & 0 & \cdots & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & \cdots & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix}$$

2D Convolution

Example: Noise Removal and Blurring

DFT of the mask **D**:

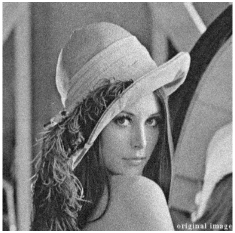


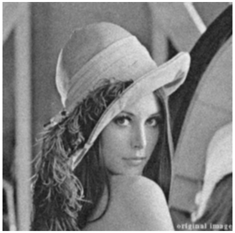
2D Convolution

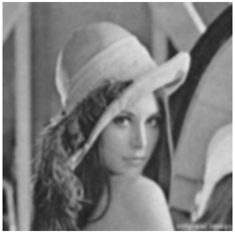
Example: Noise Removal and Blurring

Original

Noise added







Denoised $A * D$ Denoised $A * D * D * D * D * D * D$

2D Convolution

Example: Noise Removal and Blurring

- The noise is reduced by applying the averaging mask
- At the same time, the image is ‘blurred’
- High frequency contribution needed to synthesize sharp edges
- Edges and contrasted features get ‘smeared out’

2D Convolution

Example: Edge Detection

- Let $\mathbf{A} \in M_{m,n}(\mathbb{C})$ be the rectangular image
- Horizontal edge in the image could be detected by observing an intensity step between neighbouring pixels $(r-1, s)$ and (r, s)
- A suitable mask has $h_{0,0} = 1$, $h_{1,0} = -1$ and all other $h_{r,s} = 0$, because
$$(\mathbf{A} * \mathbf{H})_{r,s} = a_{r,s} - a_{r-1,s}$$
- Similarly, a suitable mask $\mathbf{V}$ for vertical edges has $v_{0,0} = 1$, $v_{0,1} = -1$ and all other $v_{r,s} = 0$

2D Convolution
Example: Edge Detection

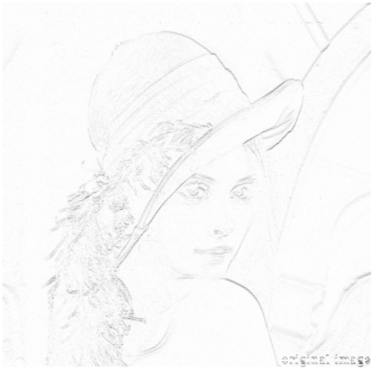
Explicit forms:

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}, \mathbf{V} = \begin{pmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

2D Convolution

Example: Edge Detection

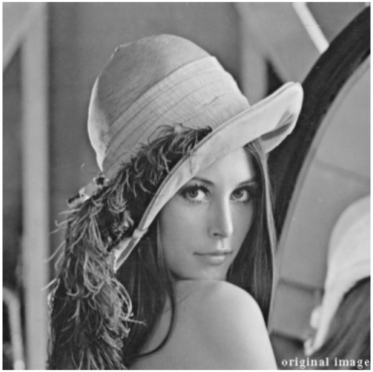
$\mathbf{A}$



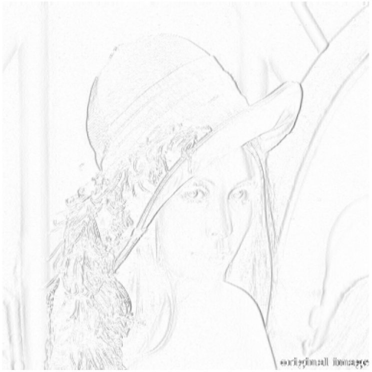
$|\mathbf{A} * \mathbf{H}|$

2D Convolution

Example: Edge Detection



$\mathbf{A}$

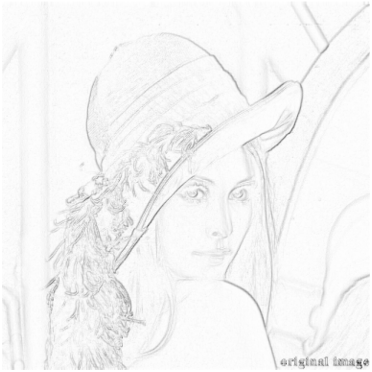


$|\mathbf{A} * \mathbf{V}|$

2D Convolution

Example: Edge Detection

$\mathbf{A}$



$\sqrt{|\mathbf{A} * \mathbf{H}|^2 + |\mathbf{A} * \mathbf{V}|^2}$

Overview

- One-dimensional convolution
- Convolution theorem and filtering
- 2D convolution – filtering images

Exercises

Convolution and Filtering

Do the following exercises in the text book: 4.2, 4.3, 4.9, 4.10, 4.15