# Linear Filtering

Computer Vision (CSCI 4220U)

**Faisal Z. Qureshi**
http://vclab.science.ontariotechu.ca
OntarioTech UNIVERSITY

# Readings

- Szeliski 2nd Edition, Section 3.2
- Checkout linear filtering notes for Computational Photography course
- Checkout course notes at http://csundergrad.science.uoit.ca/courses/cv-notes/notebooks/03-linear-filtering.html

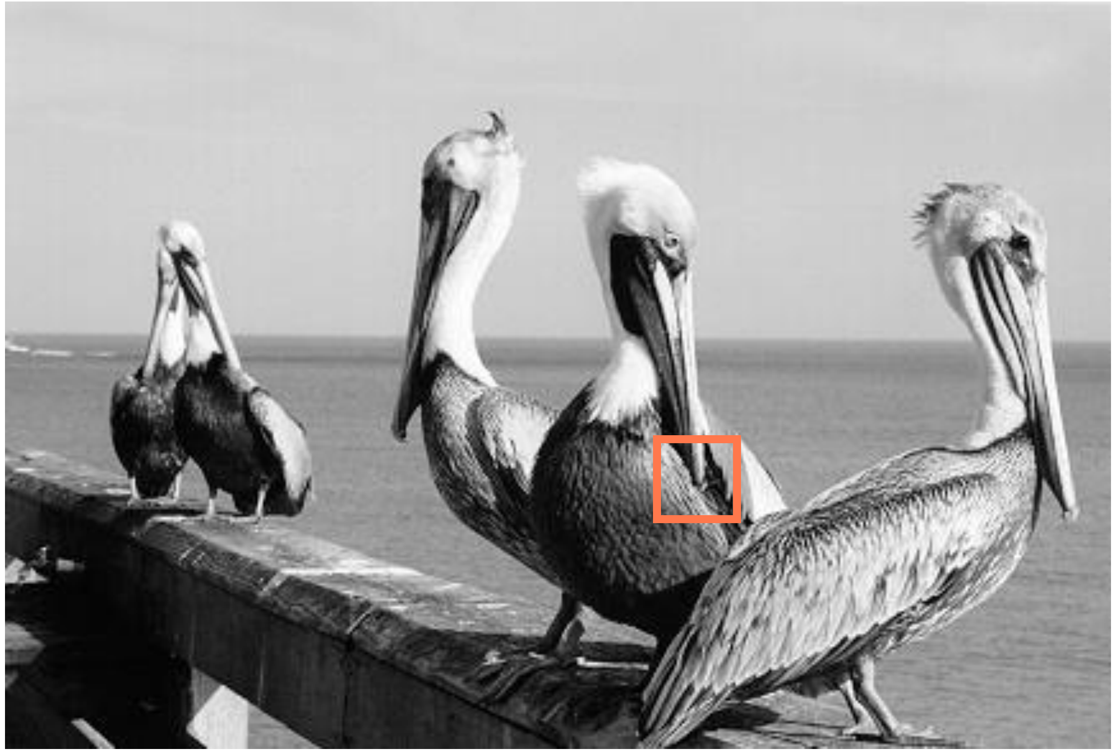
# An Example of Spatial Filtering

$f(x, y)$

$g(x, y)$

5 x 5 neighbourhood

# An Example of Spatial Filtering



$f(x, y)$

$g(x, y)$

5 x 5 neighbourhood

# Linear Filtering in 1D

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **1** | **2** | **4** | **1** | **0** | **9** | **1** | **3** | **5** | **2** |

**Signal** $(f)$

| | | |
|---|---|---|
| **1** | **0** | **-1** |

**Filter** $(h)$

Task: Apply filter $(h)$ to the signal $(f)$

# Linear Filtering in 1D

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 4 | 1 | 0 | 9 | 1 | 3 | 5 | 2 |

| | | |
|---|---|---|
| 1 | 0 | -1 |

# Linear Filtering in 1D

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **1** | **2** | **4** | **1** | **0** | **9** | **1** | **3** | **5** | **2** |

| | | |
|---|---|---|
| **1** | **0** | **-1** |

$(1)(1)+(2)(0)+(4)(-1)=-3$

# Linear Filtering in 1D

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 4 | 1 | 0 | 9 | 1 | 3 | 5 | 2 |

| | | |
|---|---|---|
| 1 | 0 | -1 |

$(1)(1)+(2)(0)+(4)(-1)=-3$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | -3 | | | | | | | | |

# Linear Filtering in 1D

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 4 | 1 | 0 | 9 | 1 | 3 | 5 | 2 |

| | | |
|---|---|---|
| 1 | 0 | -1 |

$$(1)(1)+(2)(0)+(4)(-1)=-3$$

Dot-product

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | -3 | | | | | | | | |

# Linear Filtering in 1D

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 4 | 1 | 0 | 9 | 1 | 3 | 5 | 2 |

| | | |
|---|---|---|
| 1 | 0 | -1 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | -3 | ? | | | | | | | |

# Linear Filtering in 1D

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **1** | **2** | **4** | **1** | **0** | **9** | **1** | **3** | **5** | **2** |

| | | |
|---|---|---|
| **1** | **0** | **-1** |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **-3** | **1** | | | | | | | |

# Linear Filtering in 1D

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **1** | **2** | **4** | **1** | **0** | **9** | **1** | **3** | **5** | **2** |

| | | |
|---|---|---|
| **1** | **0** | **-1** |

↓

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **-3** | **1** | **?** | | | | | | |

# Linear Filtering in 1D

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 4 | 1 | 0 | 9 | 1 | 3 | 5 | 2 |

| | | |
|---|---|---|
| 1 | 0 | -1 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | -3 | 1 | 4 | | | | | | |

# Linear Filtering in 1D

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **1** | **2** | **4** | **1** | **0** | **9** | **1** | **3** | **5** | **2** |

| | | |
|---|---|---|
| **1** | **0** | **-1** |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **-3** | **1** | **4** | **?** | | | | | |

# Linear Filtering in 1D

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 4 | 1 | 0 | 9 | 1 | 3 | 5 | 2 |

| | | |
|---|---|---|
| 1 | 0 | -1 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | -3 | 1 | 4 | -8 | | | | | |

# Linear Filtering in 1D

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **1** | **2** | **4** | **1** | **0** | **9** | **1** | **3** | **5** | **2** |

| | | |
|---|---|---|
| **1** | **0** | **-1** |

↓

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **-3** | **1** | **4** | **-8** | **?** | | | | |

# Linear Filtering in 1D

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **1** | **2** | **4** | **1** | **0** | **9** | **1** | **3** | **5** | **2** |

| | | |
|---|---|---|
| **1** | **0** | **-1** |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **-3** | **1** | **4** | **-8** | **-1** | | | | |

# Linear Filtering in 1D

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **1** | **2** | **4** | **1** | **0** | **9** | **1** | **3** | **5** | **2** |

| | | |
|---|---|---|
| **1** | **0** | **-1** |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **-3** | **1** | **4** | **-8** | **-1** | **?** | | | |

# Linear Filtering in 1D

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **1** | **2** | **4** | **1** | **0** | **9** | **1** | **3** | **5** | **2** |

| | | |
|---|---|---|
| **1** | **0** | **-1** |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **-3** | **1** | **4** | **-8** | **-1** | **6** | | | |

# Linear Filtering in 1D

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **1** | **2** | **4** | **1** | **0** | **9** | **1** | **3** | **5** | **2** |

| | | |
|---|---|---|
| **1** | **0** | **-1** |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **-3** | **1** | **4** | **-8** | **-1** | **6** | **?** | | |

# Linear Filtering in 1D

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 4 | 1 | 0 | 9 | 1 | 3 | 5 | 2 |

| | | |
|---|---|---|
| 1 | 0 | -1 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | -3 | 1 | 4 | -8 | -1 | 6 | -4 | | |

# Linear Filtering in 1D

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 4 | 1 | 0 | 9 | 1 | 3 | 5 | 2 |

| | | |
|---|---|---|
| 1 | 0 | -1 |

**Result**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | -3 | 1 | 4 | -8 | -1 | 6 | -4 | ? | |

# Linear Filtering in 1D

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **1** | **2** | **4** | **1** | **0** | **9** | **1** | **3** | **5** | **2** |

| | | |
|---|---|---|
| **1** | **0** | **-1** |

**Result**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **-3** | **1** | **4** | **-8** | **-1** | **6** | **-4** | **1** | |

# Linear Filtering in 1D

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 4 | 1 | 0 | 9 | 1 | 3 | 5 | 2 |

**Signal $(f)$**

| | | |
|---|---|---|
| 1 | 0 | -1 |

**Filter $(h)$**

Task: Apply filter $(h)$ to the signal $(f)$

# Linear Filtering in 1D

## Task: Apply filter $(h)$ to the signal $(f)$

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 4 | 1 | 0 | 9 | 1 | 3 | 5 | 2 | **Signal** $(f)$ |
| | | | | | | | 1 | 0 | -1 | **Filter** $(h)$ |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | -3 | 1 | 4 | -8 | -1 | 6 | -4 | 1 | | **Result** $CC(f, h)$ |

# Half-Width

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **1** | **2** | **4** | **1** | **0** | **9** | **1** | **3** | **5** | **2** |

| | | |
|---|---|---|
| **1** | **0** | **-1** |

half-width = 1

Filter width = 2 x (half width) + 1

= 3

**Result**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **-3** | **1** | **4** | **-8** | **-1** | **6** | **-4** | **1** | |

# Cross-correlation: $CC(i)$

$f =$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 3 | 4 | 1 | 10 | 3 | 0 | 1 |

$h =$

| | | |
|---|---|---|
| 1 | 0 | -1 |

$$CC(i) = \sum_{k \in [-w,w]} \mathbf{f}(i+k)\mathbf{h}(k)$$

Half-width $w$

# Convolution $f * h$

$f =$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **1** | **3** | **4** | **1** | **10** | **3** | **0** | **1** |

$h =$

| | | |
|---|---|---|
| **1** | **0** | **-1** |

$$(\mathbf{f} * \mathbf{k})_i = \sum_{k \in [-w, w]} \mathbf{f}(i-k)\mathbf{h}(k)$$

# Convolution $f * h$

$f =$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 3 | 4 | 1 | 10 | 3 | 0 | 1 |

$h =$

| | | |
|---|---|---|
| 1 | 0 | -1 |

$$(\mathbf{f} * \mathbf{k})_i = \sum_{k \in [-w,w]} \mathbf{f}(i-k)\mathbf{h}(k)$$

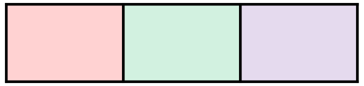
Filter is flipped

# Linear Filtering in 1D

Signal

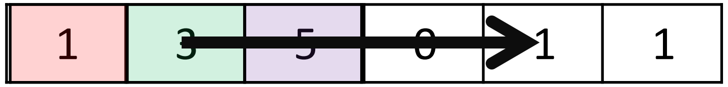
| | | | | | |
|---|---|---|---|---|---|
| 1 | 3 | 5 | 0 | 1 | 1 |

Kernel/Filter



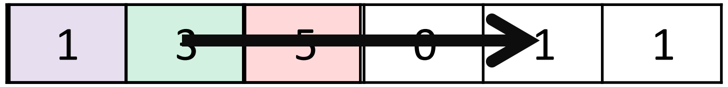
## Cross-correlation

$$CC(i) = \sum_{k \in [-w,w]} \mathbf{f}(i+k)\mathbf{h}(k)$$




## Convolution

$$(\mathbf{f} * \mathbf{k})_i = \sum_{k \in [-w,w]} \mathbf{f}(i-k)\mathbf{h}(k)$$



# Filter half-width

$f =$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **1** | **3** | **4** | **1** | **10** | **3** | **0** | **1** |

$h =$

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | |

What is the half-width of this filter $h$?

# Filter half-width

$f =$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **1** | **3** | **4** | **1** | **10** | **3** | **0** | **1** |

$h =$

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | |

Filter width = 2 x (half width) + 1

7 = 2 x (3) + 1

What is the half-width of this filter $h$? (Answer is 3)

Sometimes it is called a 7-tap filter

# (Recap) Linear Filtering in 1D

- Signal: $f$
- Kernel (sometimes called mask or filter): $h$
- Half-width of kernel: $w$

**Cross-correlation** $$CC(i) = \sum_{k \in [-w,w]} \mathbf{f}(i+k)\mathbf{h}(k)$$

**Convolution** $$(\mathbf{f} * \mathbf{k})_i = \sum_{k \in [-w,w]} \mathbf{f}(i-k)\mathbf{h}(k)$$

# Linear Filtering in 2D

| | | | | | |
|---|---|---|---|---|---|
| 1 | 3 | 5 | 0 | 1 | 1 |
| 2 | 1 | -1 | 3 | 0 | 4 |
| 0 | 1 | 2 | 2 | 3 | 4 |
| 1 | 3 | 4 | 9 | 10 | 3 |
| 3 | 7 | 8 | 2 | 1 | 4 |
| 1 | 5 | 6 | 4 | 2 | 1 |

Image

| | | |
|---|---|---|
| | | |
| | | |
| | | |

Kernel

# Linear Filtering in 2D

$CC$ (Image, Kernel) =

Image:

| | | | | | |
|---|---|---|---|---|---|
| 1 | 3 | 5 | 0 | 1 | 1 |
| 2 | 1 | -1 | 3 | 0 | 4 |
| 0 | 1 | 2 | 2 | 3 | 4 |
| 1 | 3 | 4 | 9 | 10 | 3 |
| 3 | 7 | 8 | 2 | 1 | 4 |
| 1 | 5 | 6 | 4 | 2 | 1 |

Kernel:

| | | |
|---|---|---|
| | | |
| | | |
| | | |

Result:

| | | | |
|---|---|---|---|
| ? | | | |
| | | | |
| | | | |
| | | | |

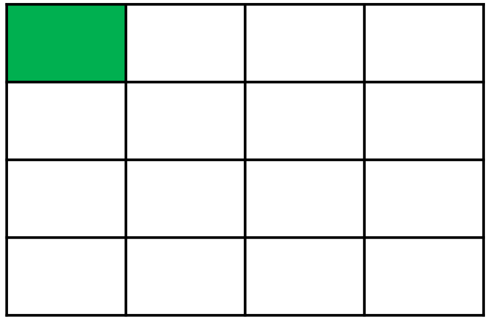
# Linear Filtering in 2D

$CC$ (Image, Kernel) =

| | | | | | |
|---|---|---|---|---|---|
| 1 | 3 | 5 | 0 | 1 | 1 |
| 2 | 1 | -1 | 3 | 0 | 4 |
| 0 | 1 | 2 | 2 | 3 | 4 |
| 1 | 3 | 4 | 9 | 10 | 3 |
| 3 | 7 | 8 | 2 | 1 | 4 |
| 1 | 5 | 6 | 4 | 2 | 1 |

Image

Kernel



# Linear Filtering in 2D

$CC$

| | | | | | |
|---|---|---|---|---|---|
| 1 | 3 | 5 | 0 | 1 | 1 |
| 2 | 1 | -1 | 3 | 0 | 4 |
| 0 | 1 | 2 | 2 | 3 | 4 |
| 1 | 3 | 4 | 9 | 10 | 3 |
| 3 | 7 | 8 | 2 | 1 | 4 |
| 1 | 5 | 6 | 4 | 2 | 1 |

Image

,

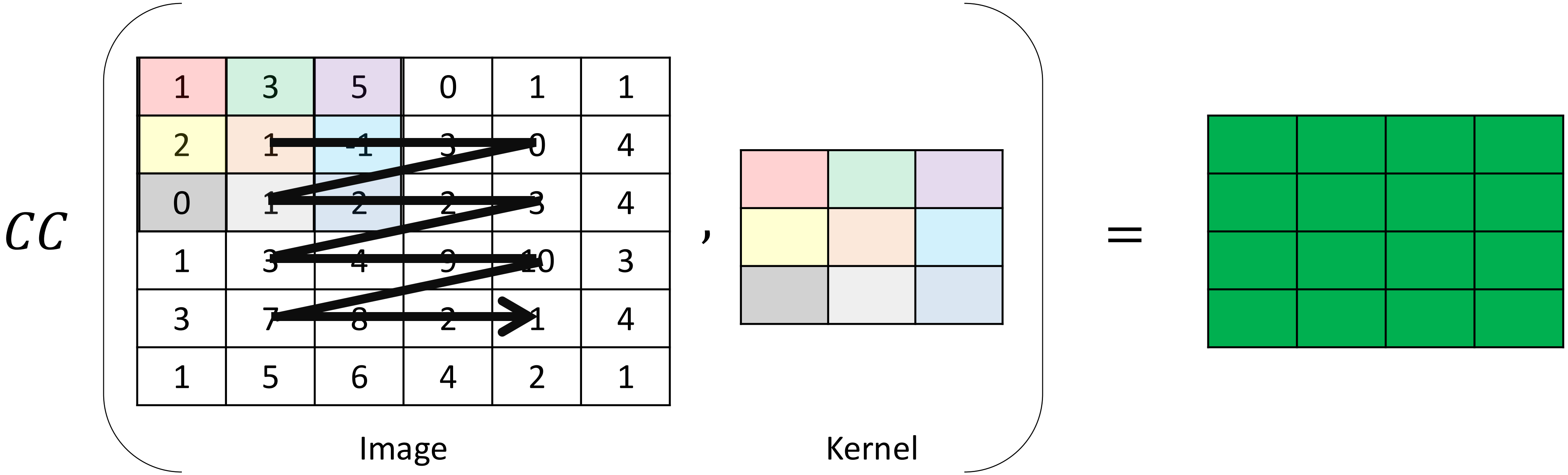
| | | |
|---|---|---|
| | | |
| | | |
| | | |

Kernel

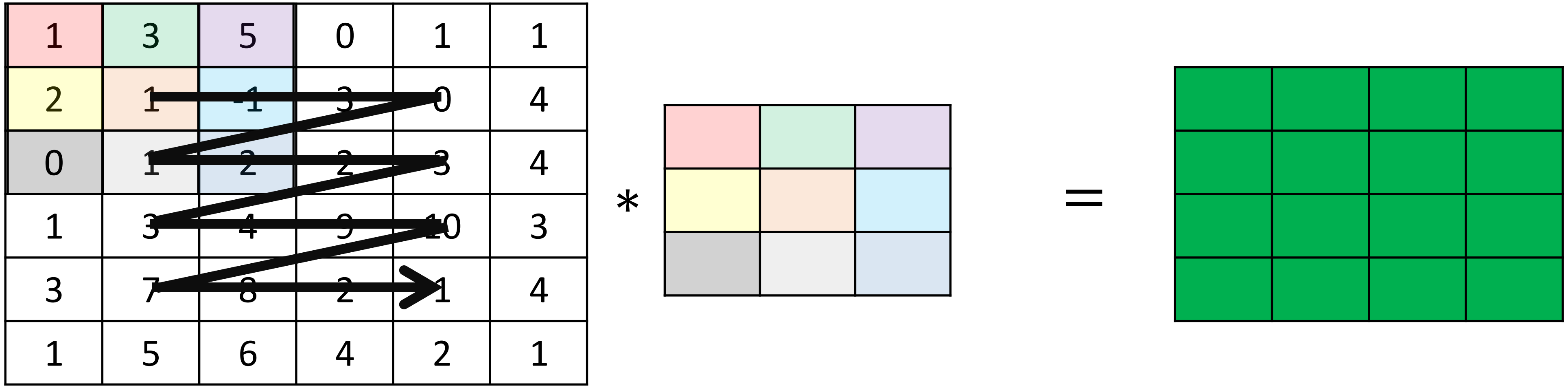
=

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

# Linear Filtering in 2D



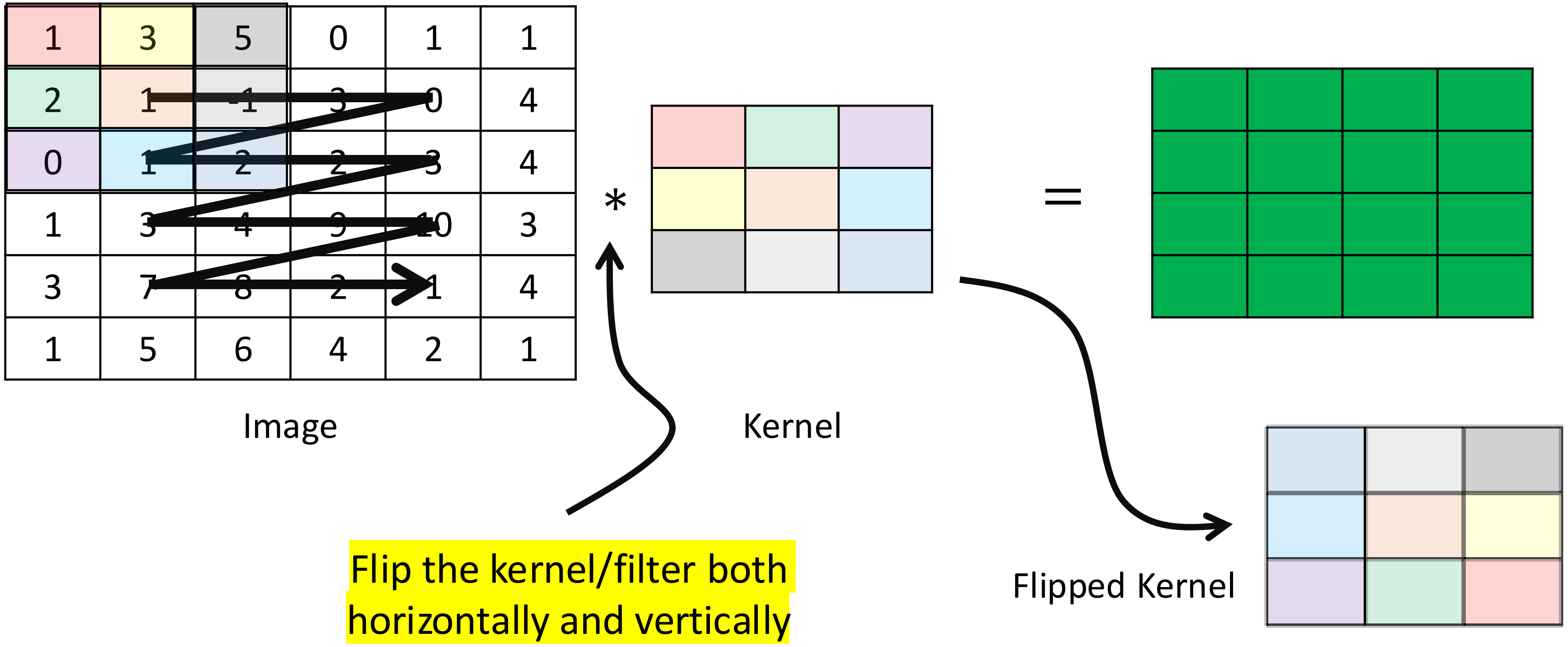
# Linear Filtering in 2D




Image

Kernel

# Linear Filtering in 2D



# (Example) Summing Filter

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 2 | 1 | 1 | 1 | 4 |
| 1 | 1 | 2 | 2 | 2 | 1 | 1 | 4 |
| 1 | 2 | 2 | 2 | 2 | 2 | 1 | 4 |
| 1 | 3 | 3 | 0 | 0 | 3 | 1 | 4 |
| 2 | 1 | 1 | 0 | 1 | 1 | 1 | 2 |
| 1 | 64 | 3 | 22 | 1 | 32 | 1 | 7 |
| 8 | 5 | 7 | 4 | 2 | 2 | 8 | 9 |
| 8 | 8 | 9 | 8 | 0 | 0 | 0 | 0 |

Image

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

Filter

# (Example) Summing Filter

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 2 | 1 | 1 | 1 | 4 |
| 1 | 1 | 2 | 2 | 2 | 1 | 1 | 4 |
| 1 | 2 | 2 | 2 | 2 | −2 | 1 | 4 |
| 1 | 3 | 3 | 0 | 0 | 3 | 1 | 4 |
| 2 | 1 | 1 | 0 | 1 | 1 | 1 | 2 |
| 1 | 64 | 3 | 22 | 1 | 32 | 1 | 7 |
| 8 | 5 | 7 | 4 | 2 | 2 | 8 | 9 |
| 8 | 8 | 9 | 8 | 0 | 0 | 0 | 0 |

11

Image

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

Filter

Summing kernel

# (Example) Averaging (Box) Filter

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 2 | 1 | 1 | 1 | 4 |
| 1 | 1 | 2 | 2 | 2 | 1 | 1 | 4 |
| 1 | 2 | 2 | 2 | 2 | $-2$ | 1 | 4 |
| 1 | 3 | 3 | 0 | 0 | 3 | 1 | 4 |
| 2 | 1 | 1 | 0 | 1 | 1 | 1 | 2 |
| 1 | 64 | 3 | 22 | 1 | 32 | 1 | 7 |
| 8 | 5 | 7 | 4 | 2 | 2 | 8 | 9 |
| 8 | 8 | 9 | 8 | 0 | 0 | 0 | 0 |

1.22

Image

$\frac{1}{9}$

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

Filter

9 by 9 averaging kernel

# Symmetric Kernels

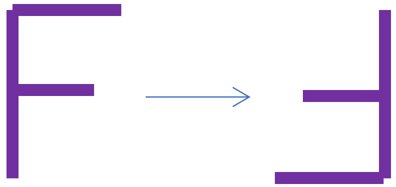
## Cross-Correlation

$$\mathrm{CC}(i,j) = \sum_{k\in[-w,w],l\in[-h,h]} \mathbf{f}(i+k, j+l)\mathbf{h}(k,l)$$

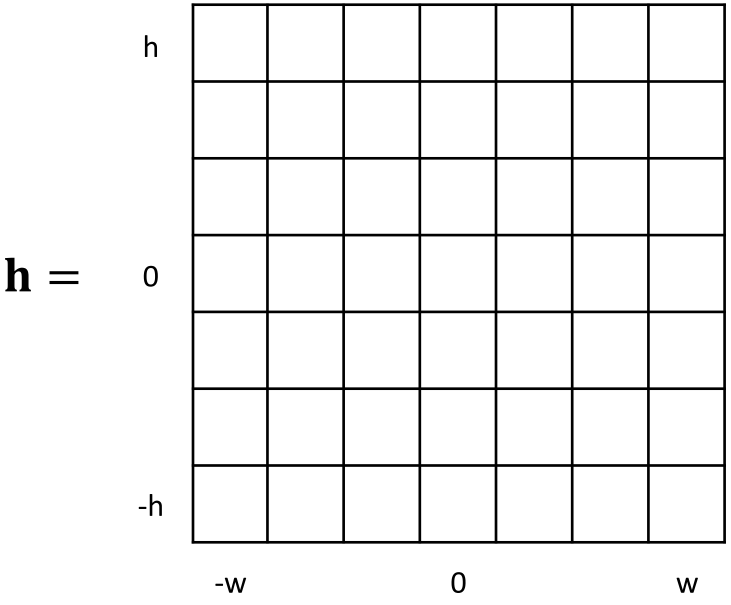
## Convolution

$$(f * h)_{ij} = \sum_{k\in[-w,w],l\in[-h,h]} \mathbf{f}(i-k, j-l)\mathbf{h}(k,l)$$

Filter $\mathbf{h}$ is flipped, both horizontally and vertically



Convolution and cross-correlation is the same for symmetric kernels



# Number of multiplications and additions

| | | | | | |
|---|---|---|---|---|---|
| 1 | 3 | 5 | 0 | 1 | 1 |
| 2 | 1 | -1 | 3 | 0 | 4 |
| 0 | 1 | 2 | 2 | 3 | 4 |
| 1 | 3 | 4 | 9 | 10 | 3 |
| 3 | 7 | 8 | 2 | 1 | 4 |
| 1 | 5 | 6 | 4 | 2 | 1 |

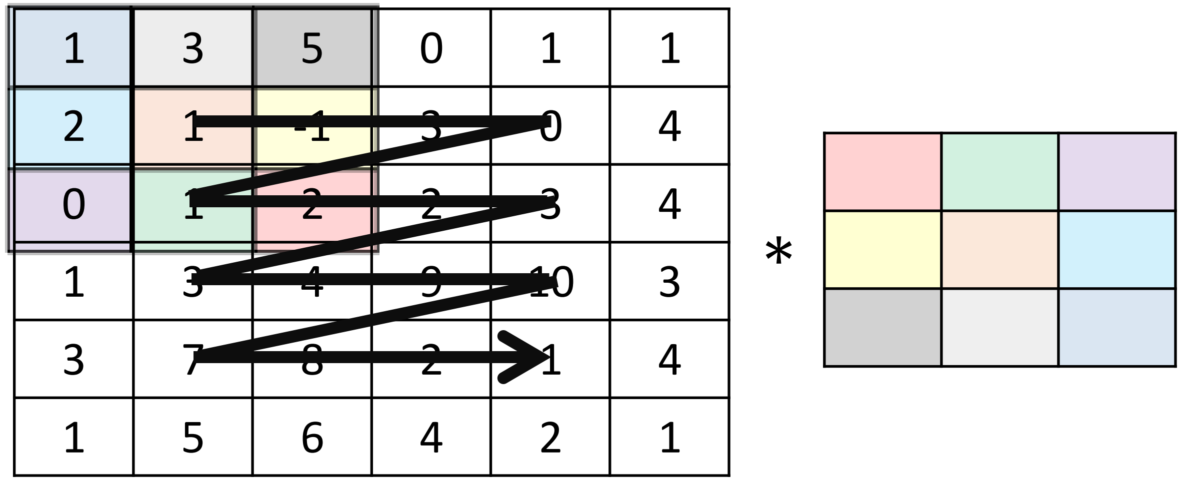
Image

*

| | | |
|---|---|---|
| | | |
| | | |
| | | |

Kernel

# Number of multiplications and additions




Image

Kernel

#locations = (4)(4)

#multiplications at each location = 9

#additions at each location = 8

#total = (9)(4)(4) multiplications

(8)(4)(4) additions

# (Aside) Multivariate Gaussian

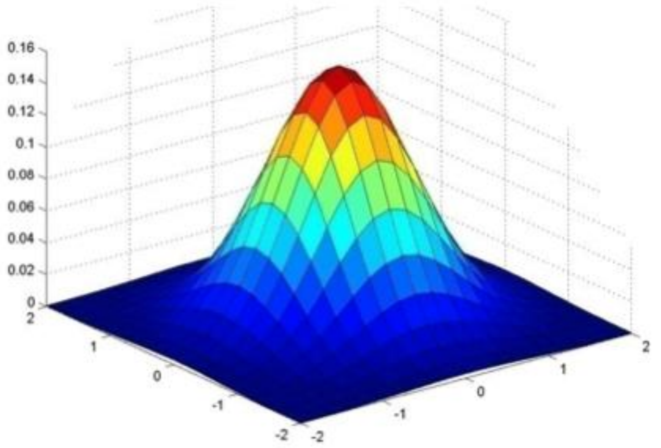
$$G(\boldsymbol{x};\boldsymbol{\mu},\Sigma) = \frac{1}{\sqrt{(2\pi)^k \det(\Sigma)}} exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

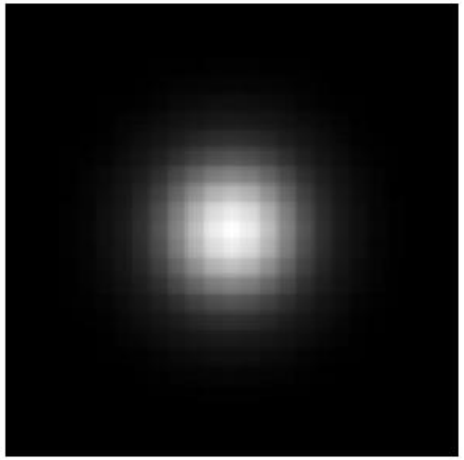
where

$\boldsymbol{x} \in R^k$

$\boldsymbol{\mu} \in R^k$

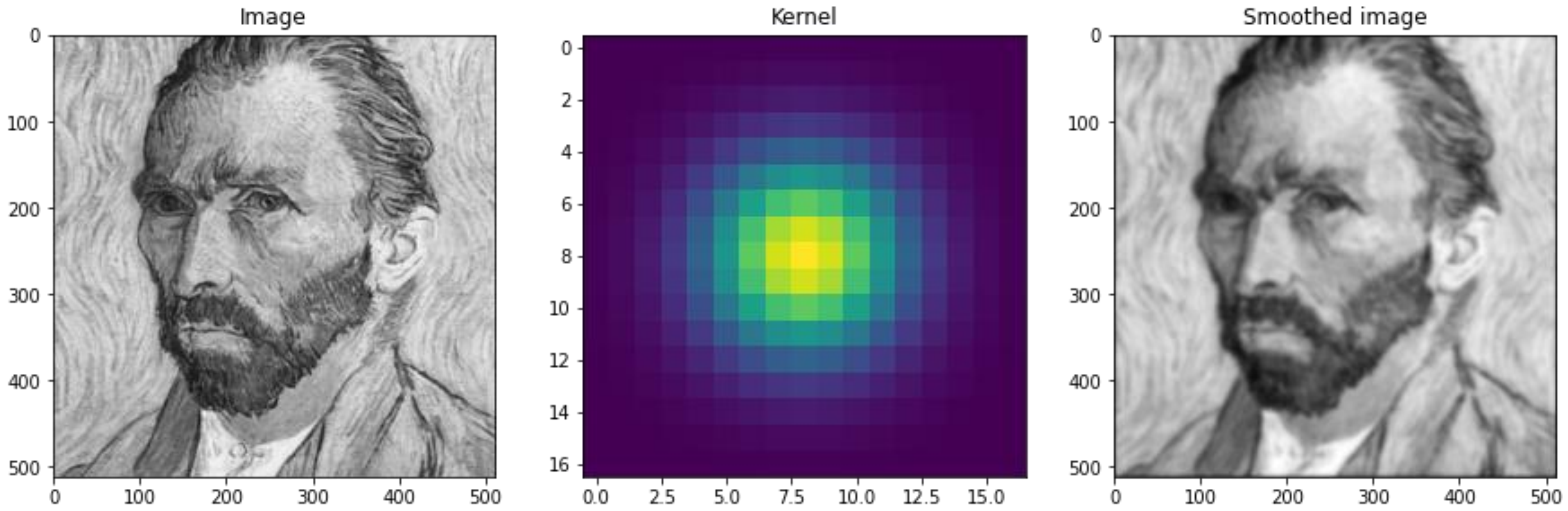
$\Sigma \in R^{k \times k}$



Gaussian in 2D

# Gaussian Blurring

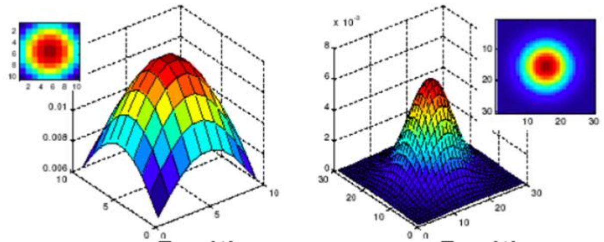


# Gaussian Blurring

- We often use the following approximation of a Gaussian function

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

- Gaussian functions have infinite support, but discrete Gaussian kernels are finite

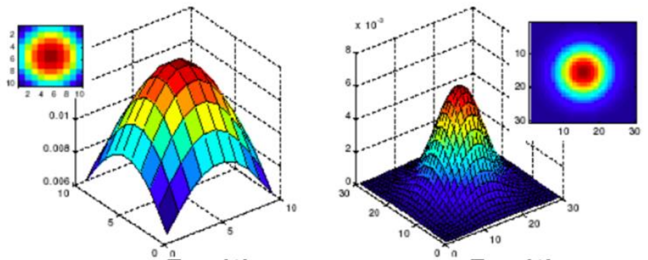


σ = 5 with 10 x 10 kernel

σ = 5 with 30 x 30 kernel

# Gaussian Blurring

- Variance controls how broad or peaky the filter is



σ = 5 with 10 x 10 kernel

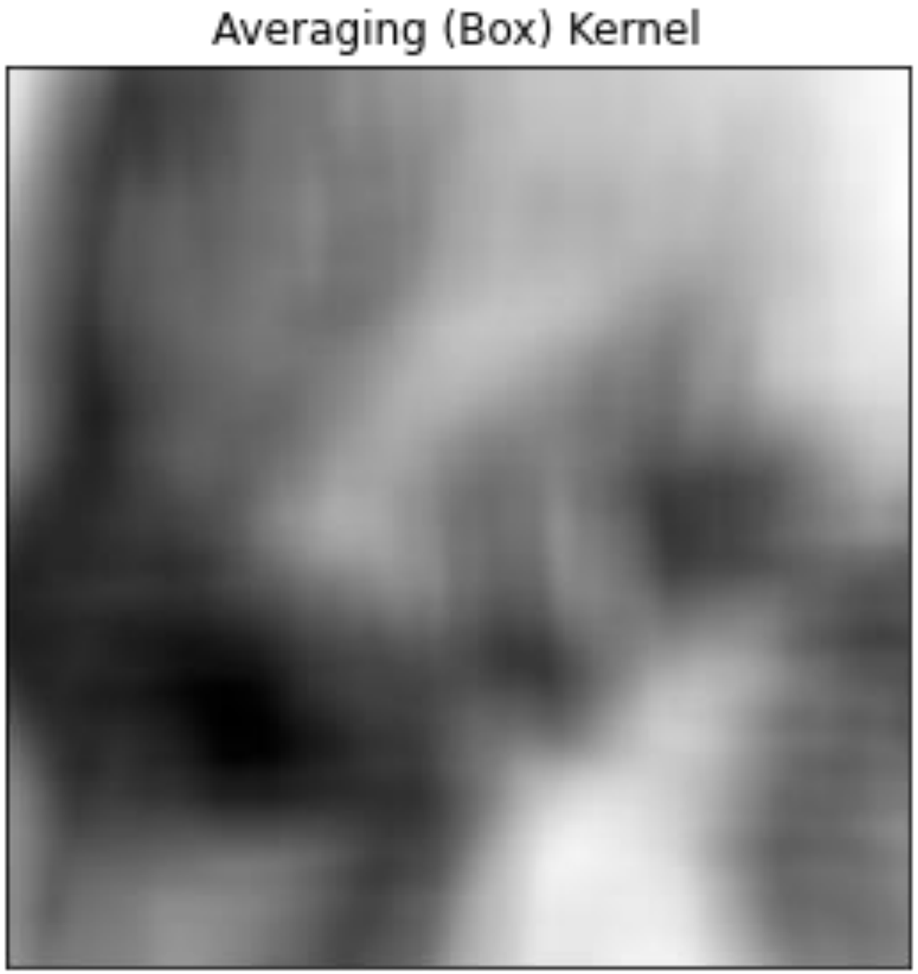
σ = 5 with 30 x 30 kernel

- Removes high-frequency components from the image
  - Blurs the image
  - Acts as a low-pass filter

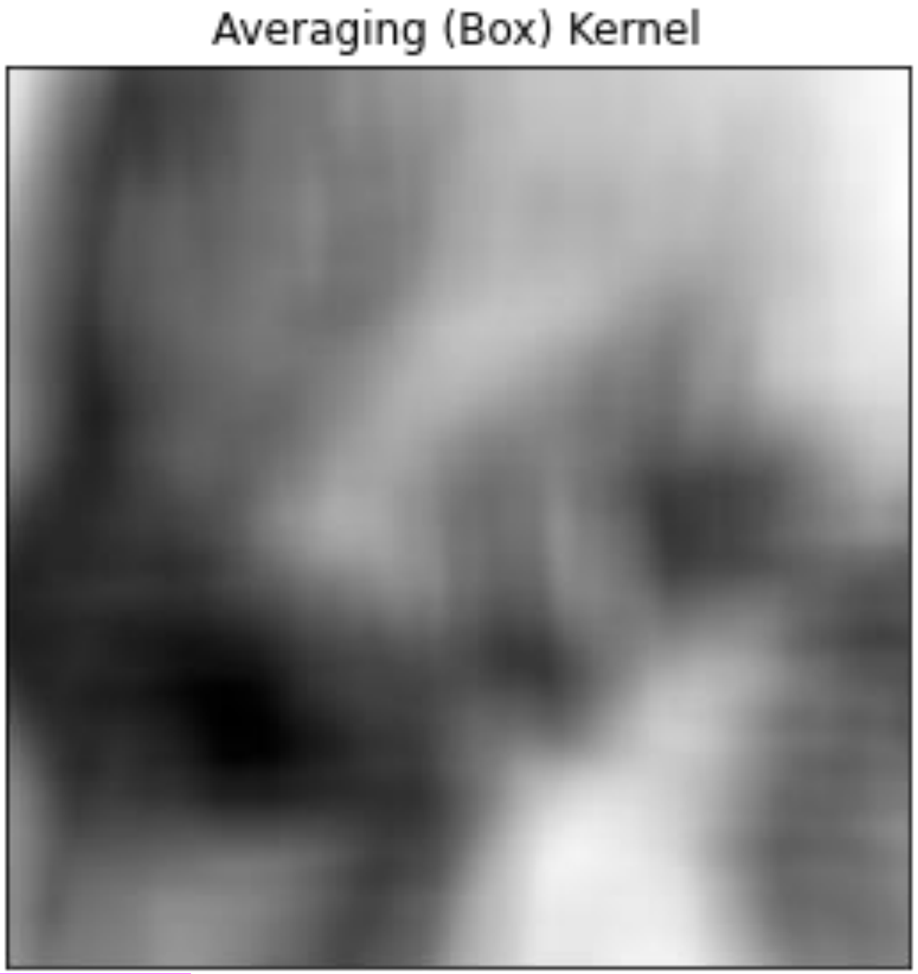
# Gaussian Blurring

- Convolving twice with Gaussian kernel of width $\sigma^2$ is the same as convolving once with kernel of width $\sigma\sqrt{2}$
- Applying a Gaussian filter with variance $\sigma_1^2$, followed by applying a Gaussian filter with variance $\sigma_2^2$ is the same as applying once with Gaussian filter with variance $\sqrt{\sigma_1^2 + \sigma_2^2}$
- All values are positive
- Values sum to 1?
  - Why is this relevant?
- This size of the filter, plus its variance, determines the extent of smoothing

# Gaussian Blurring vs. Average (Box) Filtering

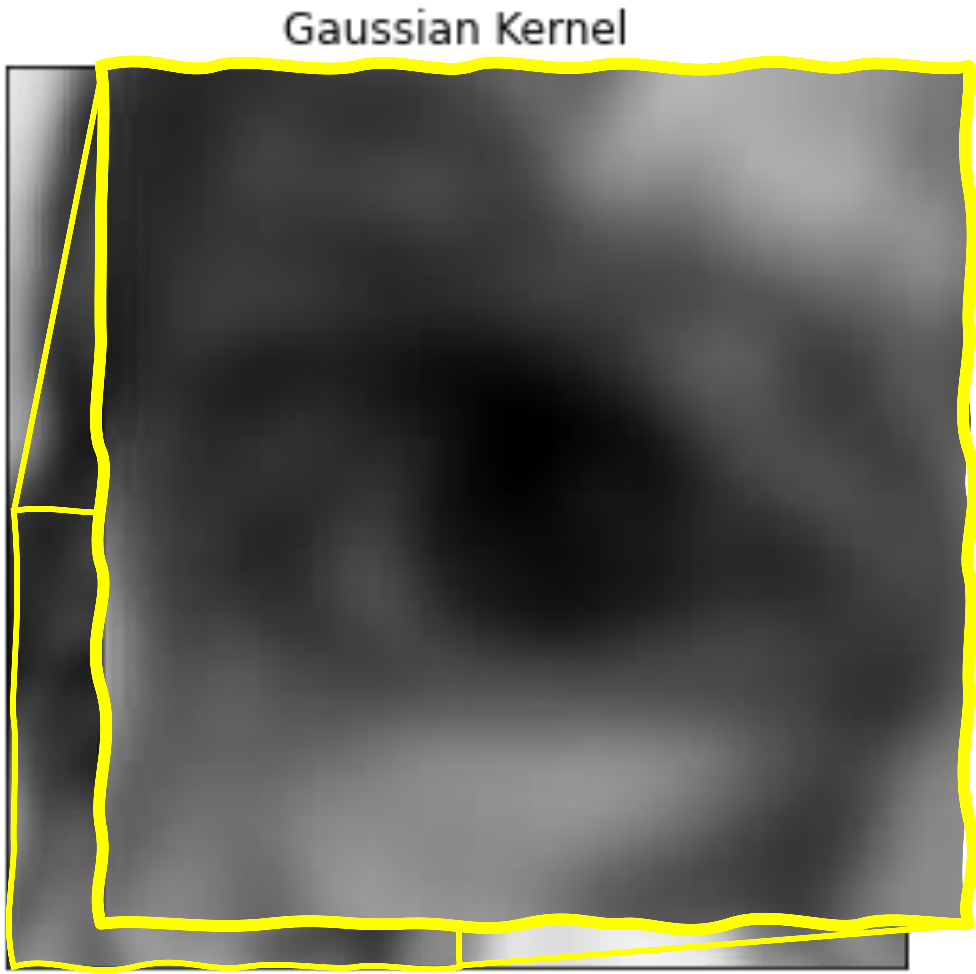



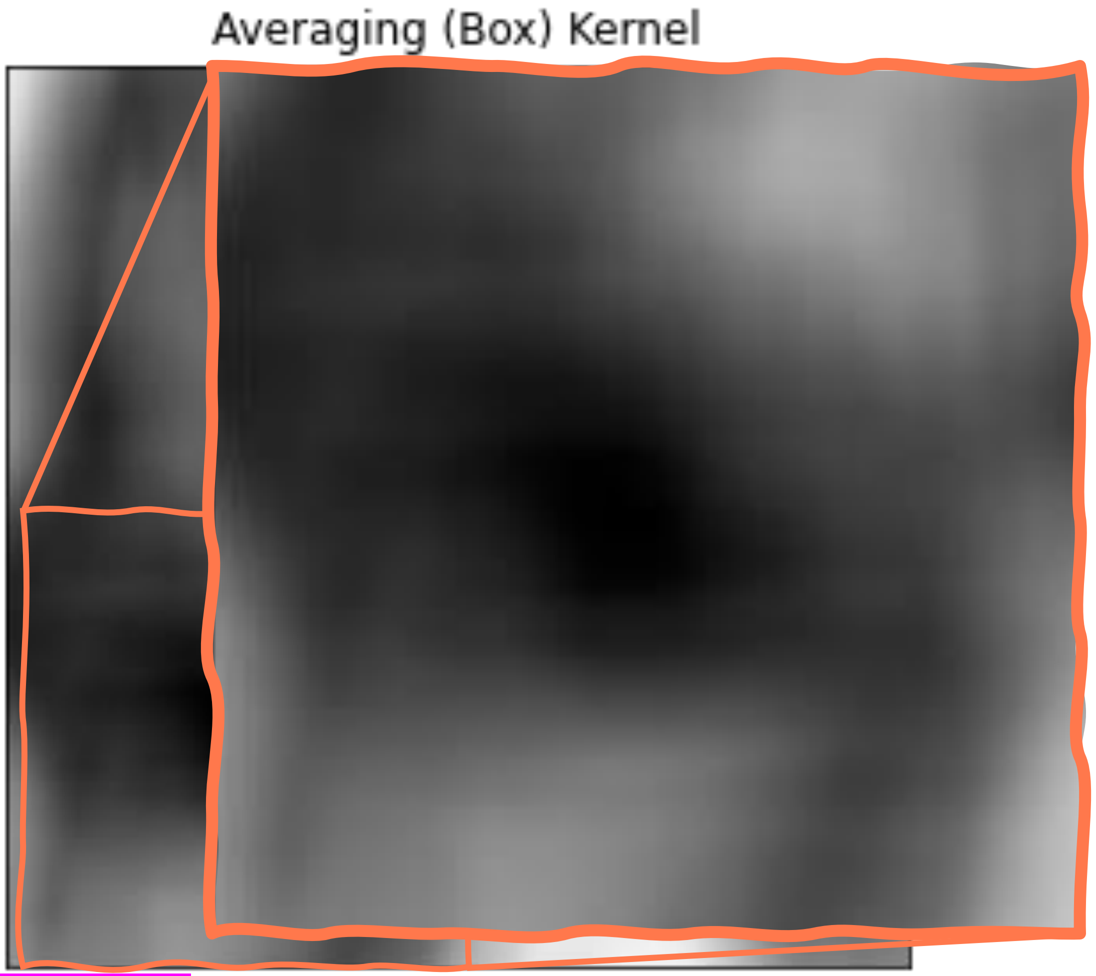
# Gaussian Blurring vs. Average (Box) Filtering


Which one is better?

# Gaussian Blurring vs. Average (Box) Filtering




Which one is better?

# Separability

- An n-dimensional filter that can be expressed as an outer-product of n 1-dimensional filters is called a separable filter

# Outer-Product and Inner-Product

$a = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and $b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

# Outer-Product and Inner-Product

$a = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and $b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

Inner-Product = $a^T b = (1)(1) + (2)(0) = 1$

# Outer-Product and Inner-Product

$a = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and $b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

Inner-Product = $a^T b = (1)(1) + (2)(0) = 1$

Outer-Product = $ab^T = \begin{bmatrix} 1 & 0 \\ 2 & 0 \end{bmatrix}$

# Outer-Product and Inner-Product

$a = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$ and $b = \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix}$

Exercise

# Separability

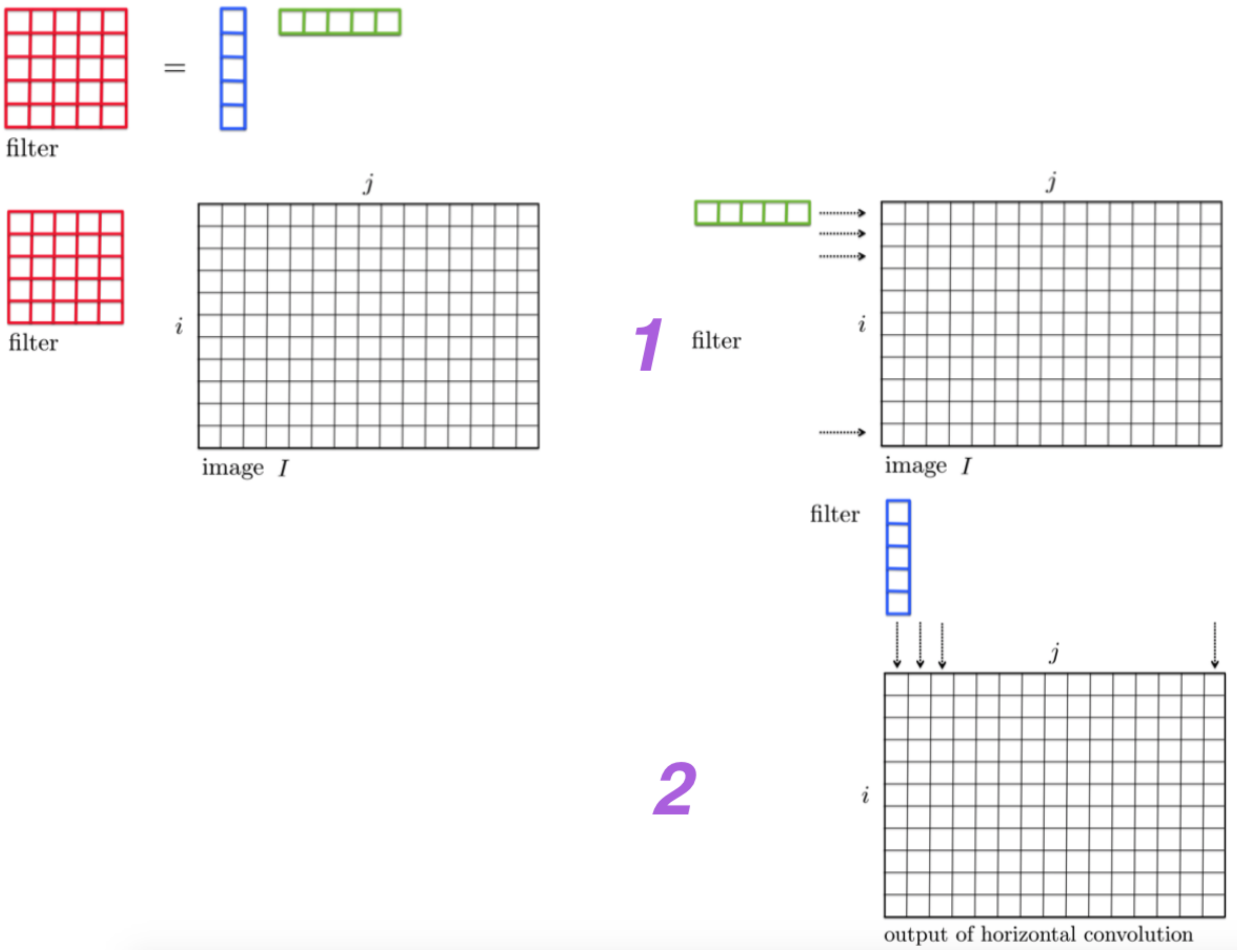
- An n-dimensional filter that can be expressed as an outer-product of n 1-dimensional filters is called a separable filter

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

# Convolution with Separable Filters in 2D

- [Step 1] Perform row-wise convolution with horizontal filter
- [Step 2] Perform column-wise convolution with the results obtained in step 1 with vertical filter

# Convolution with Separable Filters in 2D



# Convolution with Separable Filters in 2 d

Signal $\begin{bmatrix} 1 & 0 & -2 & 2 \\ 2 & -1 & 6 & 1 \\ 3 & 0 & 1 & 3 \end{bmatrix}$

Filter/Kernel $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$

# Convolution with Separable Filters in 2 d

Signal $\begin{bmatrix} 1 & 0 & -2 & 2 \\ 2 & -1 & 6 & 1 \\ 3 & 0 & 1 & 3 \end{bmatrix}$ Filter/Kernel $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$

Say we want to compute the response at location $(1,1)$, highlighted above.

# Convolution with Separable Filters in 2 d

Signal $\begin{bmatrix} 1 & 0 & -2 & 2 \\ 2 & -1 & 6 & 1 \\ 3 & 0 & 1 & 3 \end{bmatrix}$ Filter/Kernel $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$

Say we want to compute the response at location $(1,1)$, highlighted above.

**Using 2D convolution (without exploiting separability)**

$$(1)(1) + (0)(2) + (-2)(1) + (2)(2) + (-1)(4) + (6)(2) + (3)(1) + (0)(2) + (1)(1)$$
$$= 1 + 0 - 2 + 4 - 4 + 12 + 3 + 0 + 1$$
$$= 15$$

A

# Convolution with Separable Filters in 2 d

Signal $\begin{bmatrix} 1 & 0 & -2 & 2 \\ 2 & -1 & 6 & 1 \\ 3 & 0 & 1 & 3 \end{bmatrix}$ Filter/Kernel $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$

Say we want to compute the response at location $(1,1)$, highlighted above.

**Using 2D convolution (exploiting separability)**

Step 1: use horizontal filter

$$\begin{bmatrix} (1)(1) + (0)(2) + (-2)(1) \\ (2)(1) + (-1)(2) + (6)(1) \\ (3)(1) + (0)(2) + (1)(1) \end{bmatrix}$$

$$= \begin{bmatrix} -1 \\ 6 \\ 4 \end{bmatrix}$$

# Convolution with Separable Filters in 2 d

Signal $\begin{bmatrix} 1 & 0 & -2 & 2 \\ 2 & -1 & 6 & 1 \\ 3 & 0 & 1 & 3 \end{bmatrix}$ Filter/Kernel $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$

Say we want to compute the response at location $(1,1)$, highlighted above.

**Using 2D convolution (exploiting separability)**

Step 1: use horizontal filter to

$$\begin{bmatrix} (1)(1) + (0)(2) + (-2)(1) \\ (2)(1) + (-1)(2) + (6)(1) \\ (3)(1) + (0)(2) + (1)(1) \end{bmatrix}$$

$$= \begin{bmatrix} -1 \\ 6 \\ 4 \end{bmatrix}$$

Step 2: use vertical filter

$$(-1)(1) + (6)(2) + (4)(1)$$

$$= 15$$

B

Check that this is the same value as in A

# Computational Considerations

- For non-separable filters

$$O(w_k \times h_k \times w \times h)$$

- For separable filters

$$O(w_k \times w \times h) + O(w_h \times w \times h)$$

Signal $\begin{bmatrix} 1 & 0 & -2 & 2 \\ 2 & -1 & 6 & 1 \\ 3 & 0 & 1 & 3 \end{bmatrix}$

Filter/Kernel $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$

# Computational Considerations

- For non-separable filters

$$O(w_k \times h_k \times w \times h)$$

- For separable filters

$$O(w_k \times w \times h) + O(w_h \times w \times h)$$

Signal $\begin{bmatrix} 1 & 0 & -2 & 2 \\ 2 & -1 & 6 & 1 \\ 3 & 0 & 1 & 3 \end{bmatrix}$

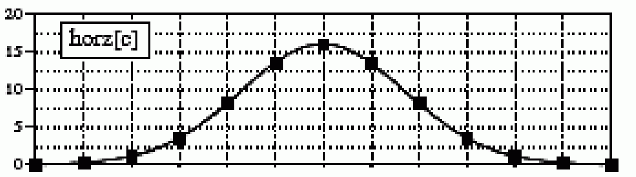
Filter/Kernel $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$

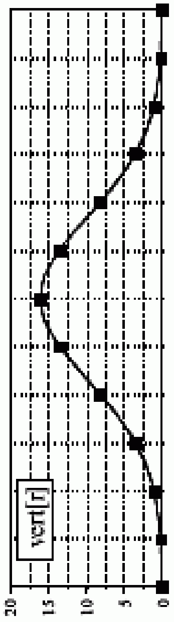
Where possible exploit separability to speed up convolutions

# Gaussian filter is separable

$$
\begin{aligned}
G_\sigma(x,y) &= \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \\
&= \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{x^2}{2\sigma^2}\right)\cdot\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{y^2}{2\sigma^2}\right) \\
&= g_\sigma(x)\cdot g_\sigma(y)
\end{aligned}
$$

FIGURE 24-7
Separation of the Gaussian. The Gaussian is the only PSF that is circularly symmetric *and* separable. This makes it a common filter kernel in image processing.




| | 0.04 | 0.25 | 1.11 | 3.56 | 8.20 | 13.5 | 16.0 | 13.5 | 8.20 | 3.56 | 1.11 | 0.25 | 0.04 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.04 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0.25 | 0 | 0 | 0 | 1 | 2 | 3 | 4 | 3 | 2 | 1 | 0 | 0 | 0 |
| 1.11 | 0 | 0 | 1 | 4 | 9 | 15 | 18 | 15 | 9 | 4 | 1 | 0 | 0 |
| 3.56 | 0 | 1 | 4 | 13 | 29 | 48 | 57 | 48 | 29 | 13 | 4 | 1 | 0 |
| 8.20 | 0 | 2 | 9 | 29 | 67 | 111 | 131 | 111 | 67 | 29 | 9 | 2 | 0 |
| 13.5 | 1 | 3 | 15 | 48 | 111 | 183 | 216 | 183 | 111 | 48 | 15 | 3 | 1 |
| 16.0 | 1 | 4 | 18 | 57 | 131 | 216 | 255 | 216 | 131 | 57 | 18 | 4 | 1 |
| 13.5 | 1 | 3 | 15 | 48 | 111 | 183 | 216 | 183 | 111 | 48 | 15 | 3 | 1 |
| 8.20 | 0 | 2 | 9 | 29 | 67 | 111 | 131 | 111 | 67 | 29 | 9 | 2 | 0 |
| 3.56 | 0 | 1 | 4 | 13 | 29 | 48 | 57 | 48 | 29 | 13 | 4 | 1 | 0 |
| 1.11 | 0 | 0 | 1 | 4 | 9 | 15 | 18 | 15 | 9 | 4 | 1 | 0 | 0 |
| 0.25 | 0 | 0 | 0 | 1 | 2 | 3 | 4 | 3 | 2 | 1 | 0 | 0 | 0 |
| 0.04 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

The Scientist and Engineer's Guide to Digital Signal Processing
By Steven W. Smith, Ph.D.

# Constructing a 2D Gaussian Filter

- How would you construct a 5x5 Gaussian filter given a the following 1x5 Gaussian filter?

$$\frac{1}{16}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

- Hint: employ the fact that the Gaussian filter is separable.

# Examples of Separable Filters

(a) box, $K = 5$: $\frac{1}{K^2}$

| | | | |
|---|---|---|---|
| 1 | 1 | ⋯ | 1 |
| 1 | 1 | ⋯ | 1 |
| ⋮ | ⋮ | 1 | ⋮ |
| 1 | 1 | ⋯ | 1 |

$\frac{1}{K}$

| | | | |
|---|---|---|---|
| 1 | 1 | ⋯ | 1 |

(b) bilinear: $\frac{1}{16}$

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

$\frac{1}{4}$

| | | |
|---|---|---|
| 1 | 2 | 1 |

(c) "Gaussian": $\frac{1}{256}$

| | | | | |
|---|---|---|---|---|
| 1 | 4 | 6 | 4 | 1 |
| 4 | 16 | 24 | 16 | 4 |
| 6 | 24 | 36 | 24 | 6 |
| 4 | 16 | 24 | 16 | 4 |
| 1 | 4 | 6 | 4 | 1 |

$\frac{1}{16}$

| | | | | |
|---|---|---|---|---|
| 1 | 4 | 6 | 4 | 1 |

(d) Sobel: $\frac{1}{8}$

| | | |
|---|---|---|
| −1 | 0 | 1 |
| −2 | 0 | 2 |
| −1 | 0 | 1 |

$\frac{1}{2}$

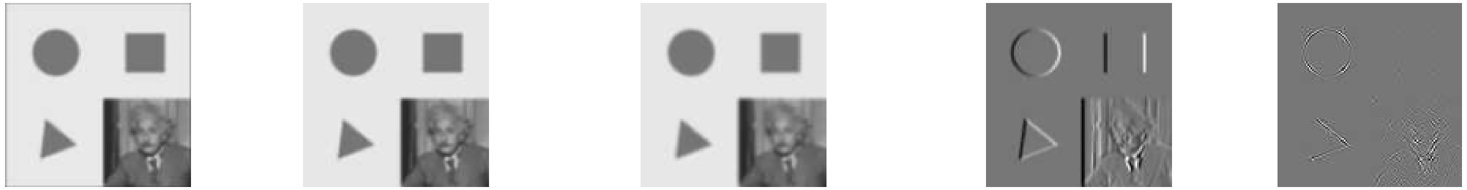
| | | |
|---|---|---|
| −1 | 0 | 1 |

(e) corner: $\frac{1}{4}$

| | | |
|---|---|---|
| 1 | −2 | 1 |
| −2 | 4 | −2 |
| 1 | −2 | 1 |

$\frac{1}{2}$

| | | |
|---|---|---|
| 1 | −2 | 1 |



(a) box, $K = 5$ (b) bilinear (c) "Gaussian" (d) Sobel (e) corner

**Figure 3.14** *Separable linear filters: For each image (a)–(e), we show the 2D filter kernel (top), the corresponding horizontal 1D kernel (middle), and the filtered image (bottom). The filtered Sobel and corner images are signed, scaled up by 2× and 4×, respectively, and added to a gray offset before display.*

# How to find if a 2D filter is separable?

- Use Singular Value Decomposition (SVD)
  - If only one singular value is non-zero then the 2D filter is separable

- [Step 1] Compute SVD and check if only one singular value is non-zero

$$\mathbf{F} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}} = \sum_{i} \sigma_{ii}\, \boldsymbol{u}_i \boldsymbol{v}_i^T$$

where $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_{ii})$

# Singular Value Decomposition (SVD)

Factor a matrix $M$ as follows: $M = \underbrace{U\Sigma V^T} = \sum_i \sigma_{ii}\boldsymbol{u}_i\boldsymbol{v}_i^T$

$$U = [\boldsymbol{u}_1 \quad \cdots \quad \boldsymbol{u}_m] \qquad \Sigma = \begin{bmatrix} \sigma_{11} & \\ & \ddots \end{bmatrix} \qquad V^T = \begin{bmatrix} \boldsymbol{v}_1^T \\ \vdots \\ \boldsymbol{v}_n^T \end{bmatrix}$$

$M \in R^{m \times n}$

$U \in R^{m \times m}$

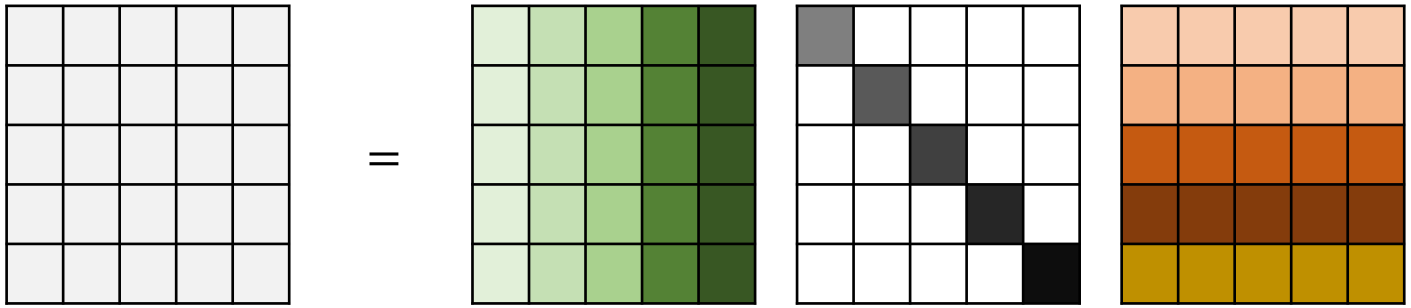
$\Sigma \in R^{m \times n}$ is a rectangular diagonal matrix. $\sigma_{ii}$ contains the singular values

$V^T \in R^{n \times n}$

# Singular value decomposition (Case 1)

**Matrix $M$ is square**

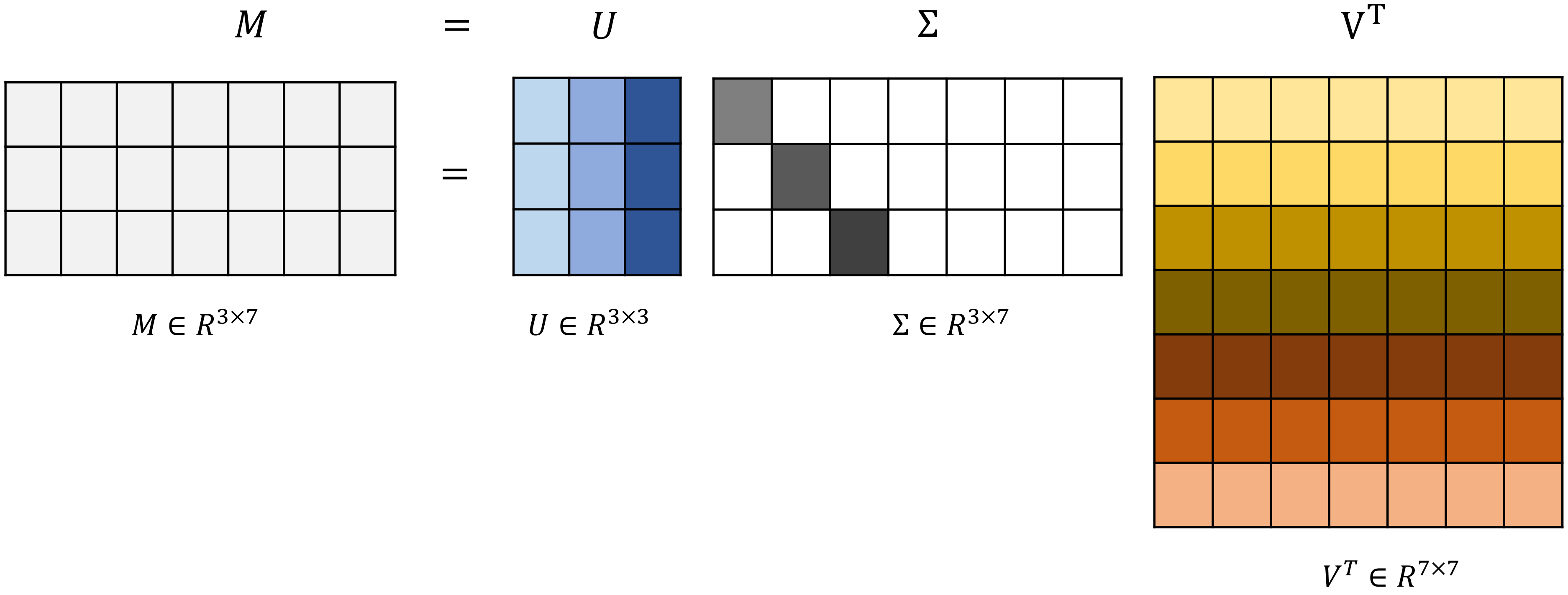
$$M \quad = \quad U \quad \Sigma \quad \mathrm{V^T}$$



$M \in R^{5\times5}$ $U \in R^{5\times5}$ $\Sigma \in R^{5\times5}$ $V^T \in R^{5\times5}$

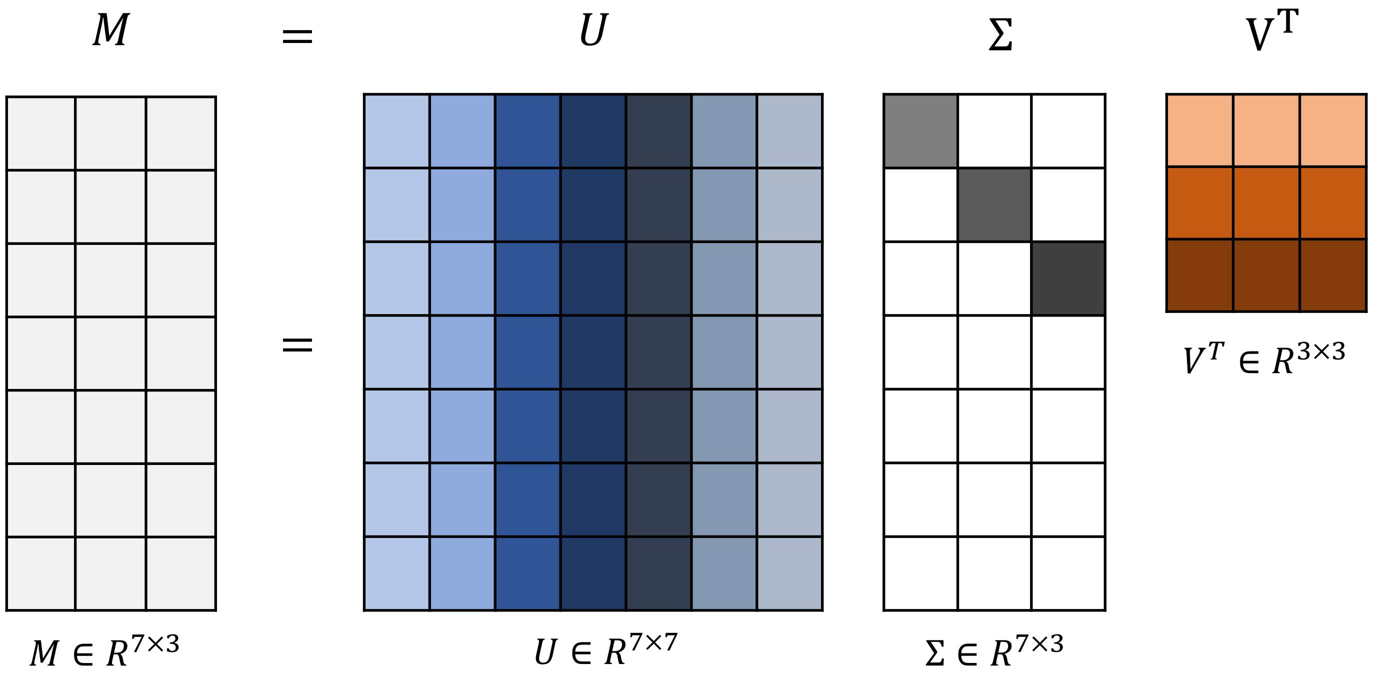
# Singular value decomposition (Case 2)

**Matrix $M$ is wide**



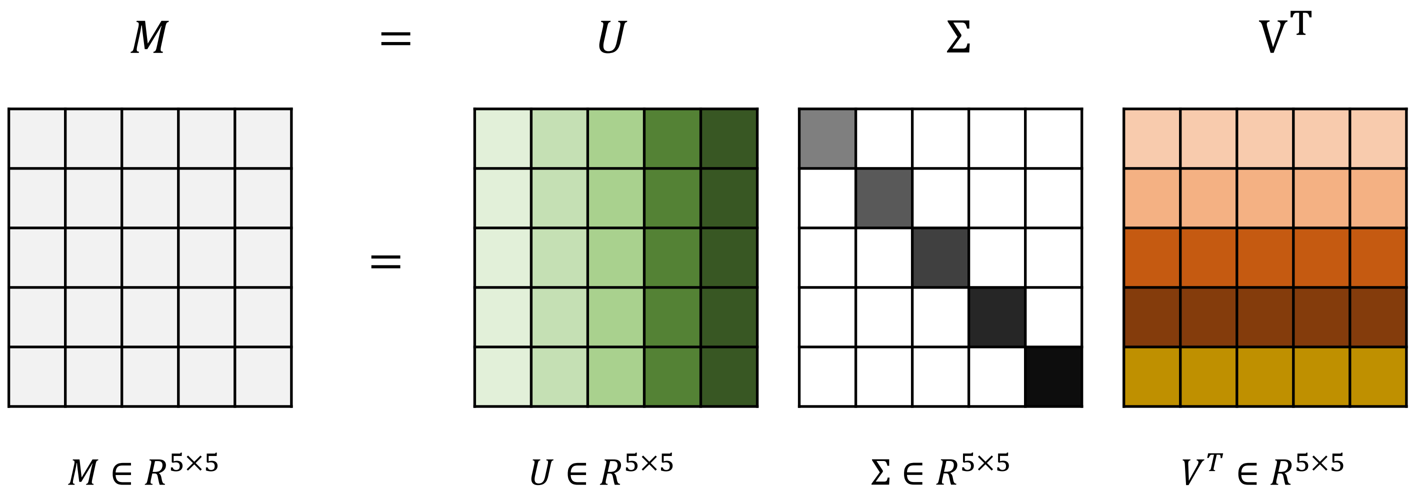
# Singular value decomposition (Case 3)

**Matrix $M$ is tall**



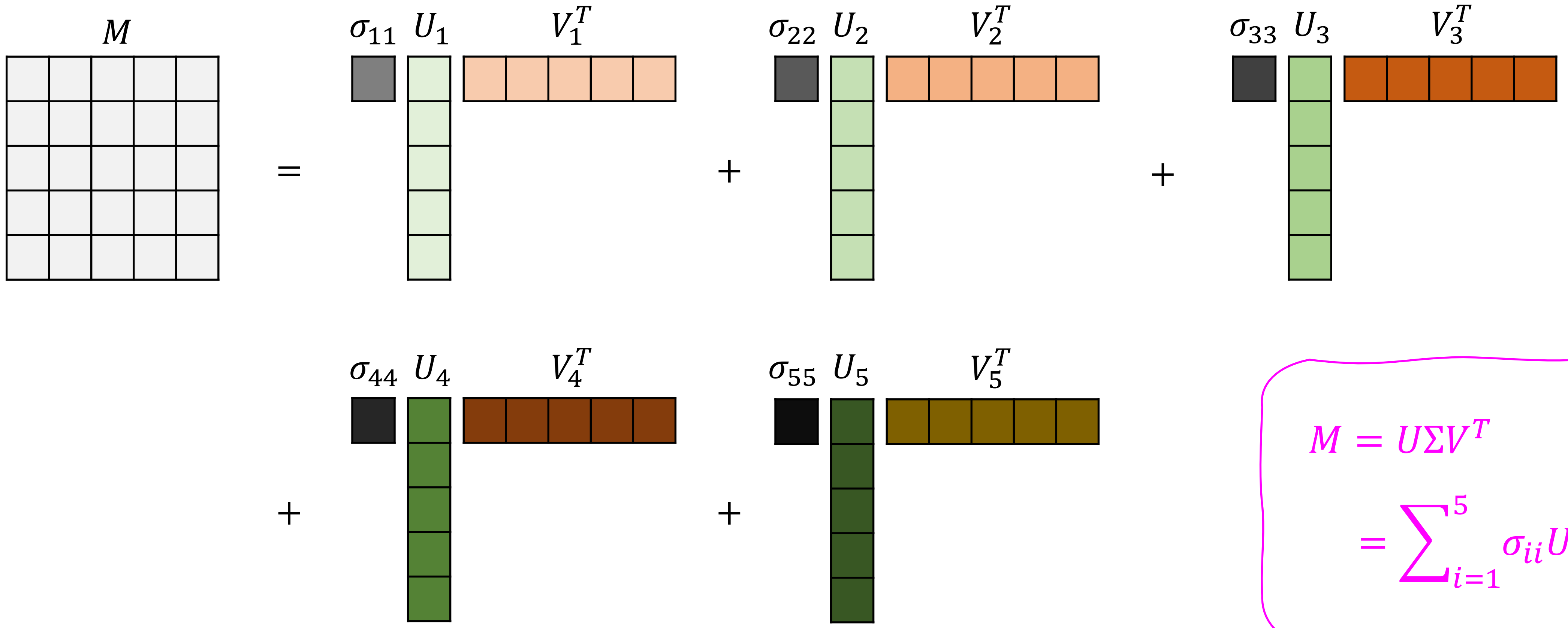
# Singular value decomposition

**Expressed as a sum of scaled outer-products between columns of $U$ and rows of $V^T$**



# Singular value decomposition

**Expressed as a sum of scaled outer-products between columns of $U$ and rows of $V^T$**



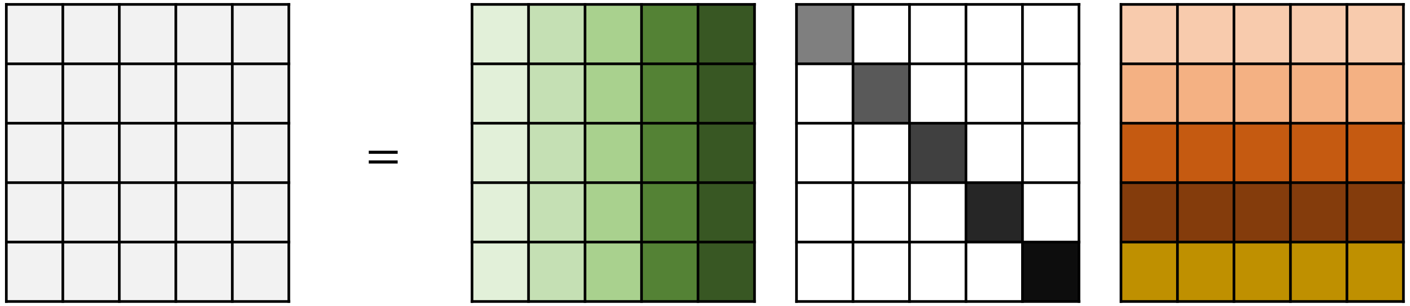

$$M = U\Sigma V^T$$

$$= \sum_{i=1}^{5} \sigma_{ii} U_i V_i^T$$

# Checking for Separability using SVD

**What if only $\sigma_{11}$ is non-zero?**

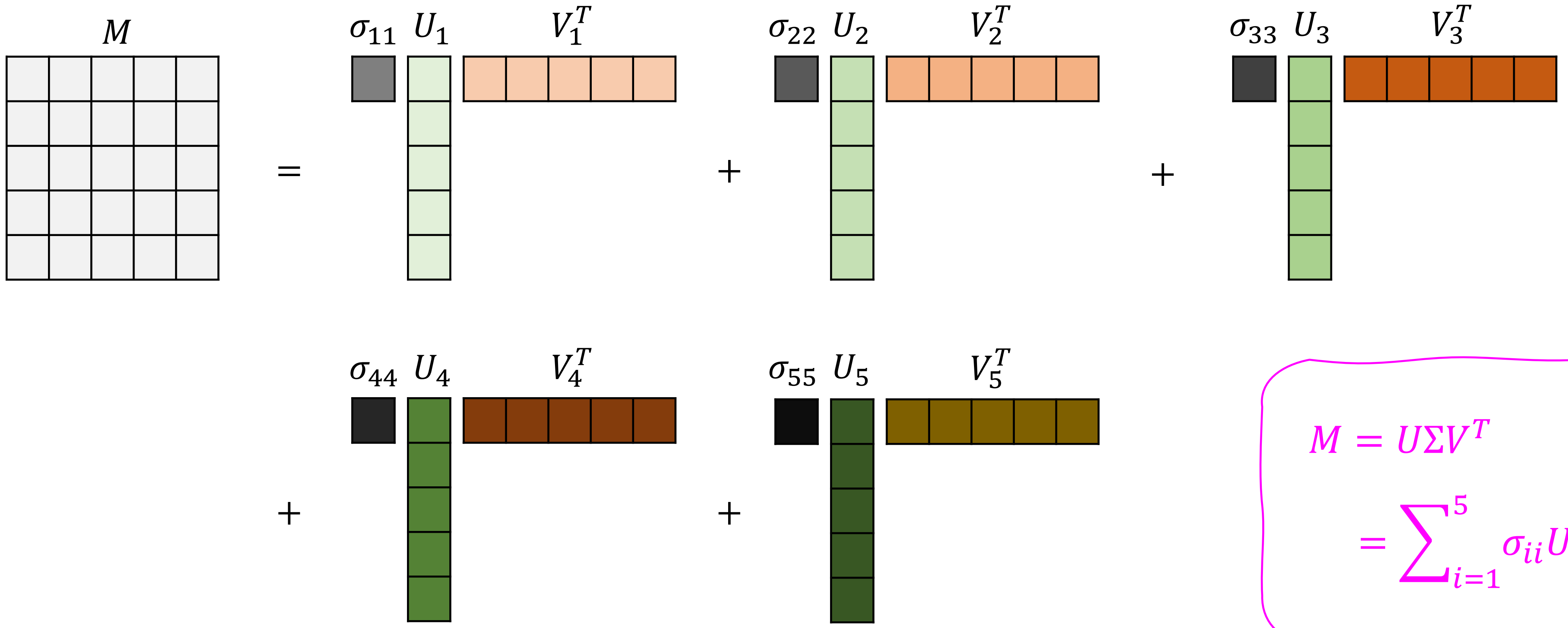
$M \quad = \quad U \quad \Sigma \quad V^{\mathrm{T}}$



$M \in R^{5\times5}$ $\quad U \in R^{5\times5}$ $\quad \Sigma \in R^{5\times5}$ $\quad V^T \in R^{5\times5}$

# Singular value decomposition

**What if only $\sigma_{11}$ is non-zero?**

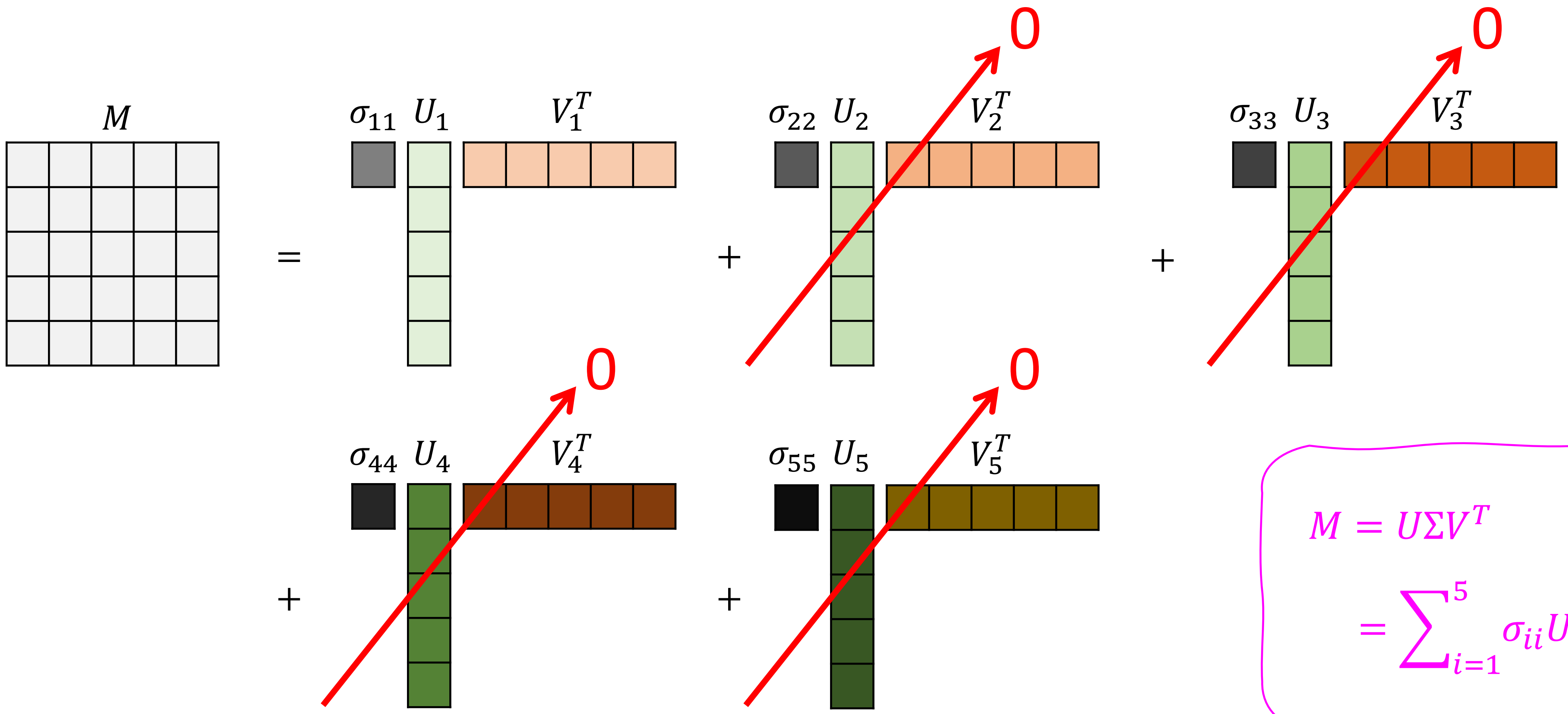



$$M = U\Sigma V^T$$

$$= \sum_{i=1}^{5} \sigma_{ii} U_i V_i^T$$

# Singular value decomposition

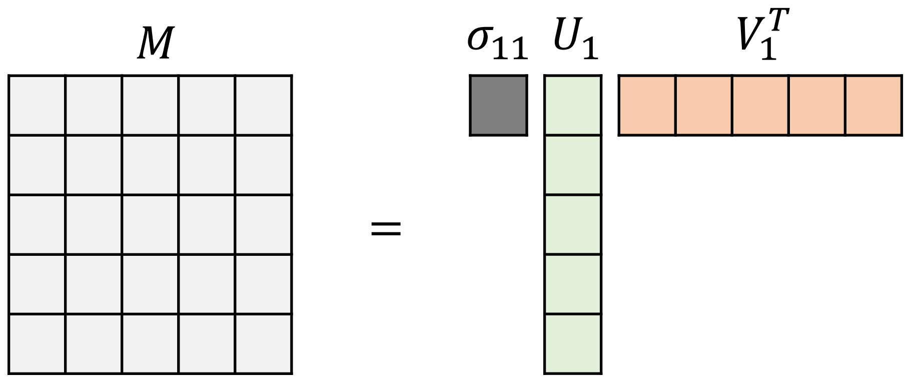
**What if only $\sigma_{11}$ is non-zero?**




$$M = U \Sigma V^T$$
$$= \sum_{i=1}^{5} \sigma_{ii} U_i V_i^T$$

# Singular value decomposition

**What if only $\sigma_{11}$ is non-zero?**




Outer-product of $\left(\sqrt{\sigma_{11}}\right)U_1$ and $\left(\sqrt{\sigma_{11}}\right)V_1^T$

$$M = U\Sigma V^T$$
$$= \sigma_{11} U_1 V_1^T$$

# How to find if a 2D filter is separable?

- Use Singular Value Decomposition (SVD)
  - If only one singular value is non-zero then the 2D filter is separable

- [Step 1] Compute SVD and check if only one singular value is non-zero

$$\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}} = \sum_i \sigma_i \, \boldsymbol{u}_i \boldsymbol{v}_i^T$$

where $\mathbf{\Sigma} = \mathrm{diag}(\sigma_i)$

- [Step 2] Vertical and horizontal filters are: $\sqrt{\sigma_1}\boldsymbol{u}_1$ and $\sqrt{\sigma_1}\boldsymbol{v}_1^T$

# Dealing with boundary values

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **1** | **2** | **4** | **1** | **0** | **9** | **1** | **3** | **5** | **2** |

| | | |
|---|---|---|
| **1** | **0** | **-1** |

half-width = 1

Filter width = 2 x (half width) + 1

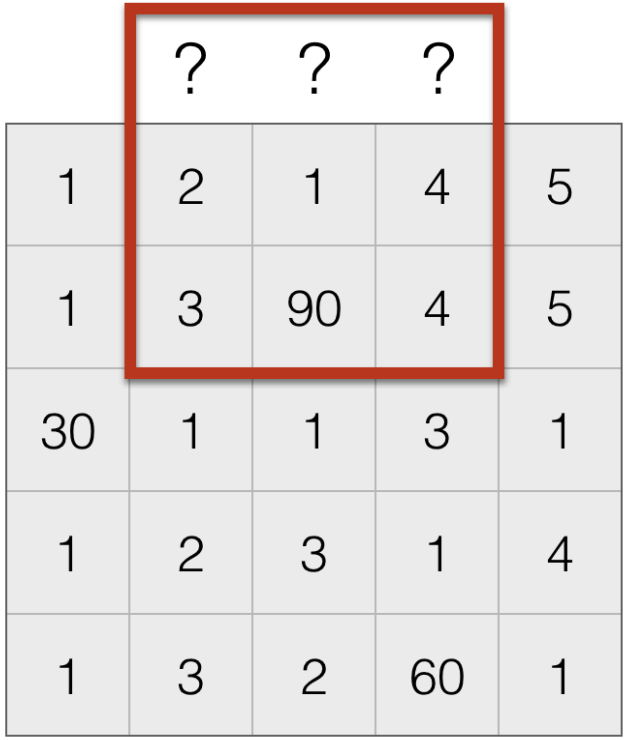
= 3

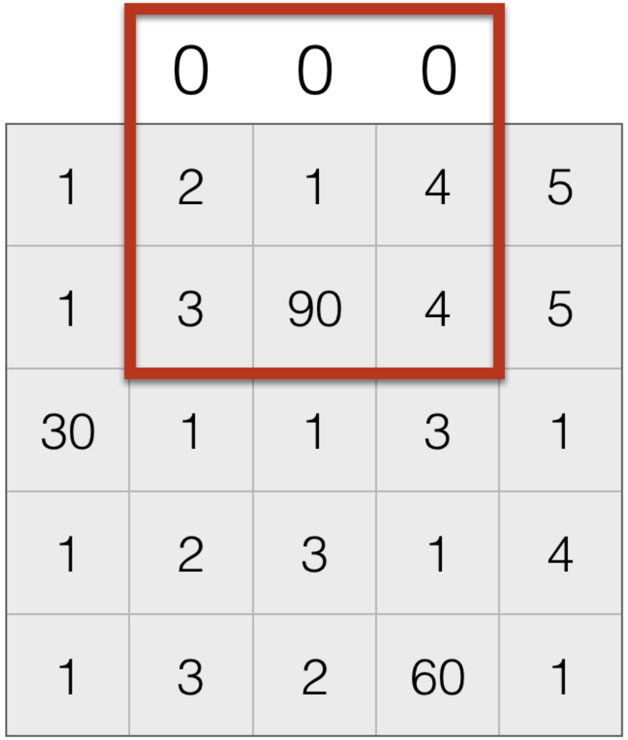
**Result**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **-3** | **1** | **4** | **-8** | **-1** | **6** | **-4** | **1** | |

What to do with these values?

# How to deal with missing (boundary values)




Set missing value to a particular value, say 0

# How to deal with missing (boundary values)

| | | | | |
|---|---|---|---|---|
| | ? | ? | ? | |
| 1 | 2 | 1 | 4 | 5 |
| 1 | 3 | 90 | 4 | 5 |
| 30 | 1 | 1 | 3 | 1 |
| 1 | 2 | 3 | 1 | 4 |
| 1 | 3 | 2 | 60 | 1 |

| | | | | |
|---|---|---|---|---|
| | 2 | 1 | 4 | |
| 1 | 2 | 1 | 4 | 5 |
| 1 | 3 | 90 | 4 | 5 |
| 30 | 1 | 1 | 3 | 1 |
| 1 | 2 | 3 | 1 | 4 |
| 1 | 3 | 2 | 60 | 1 |

Repeat boundary entries

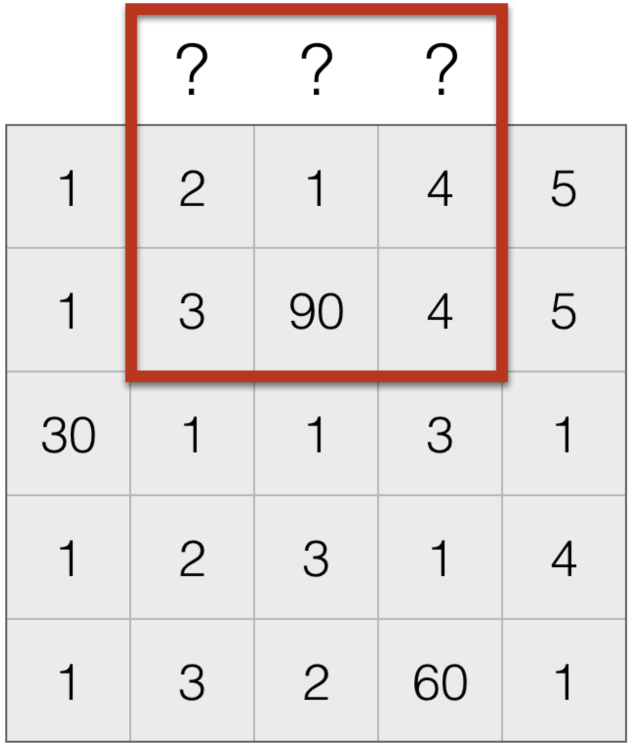
# How to deal with missing (boundary values)

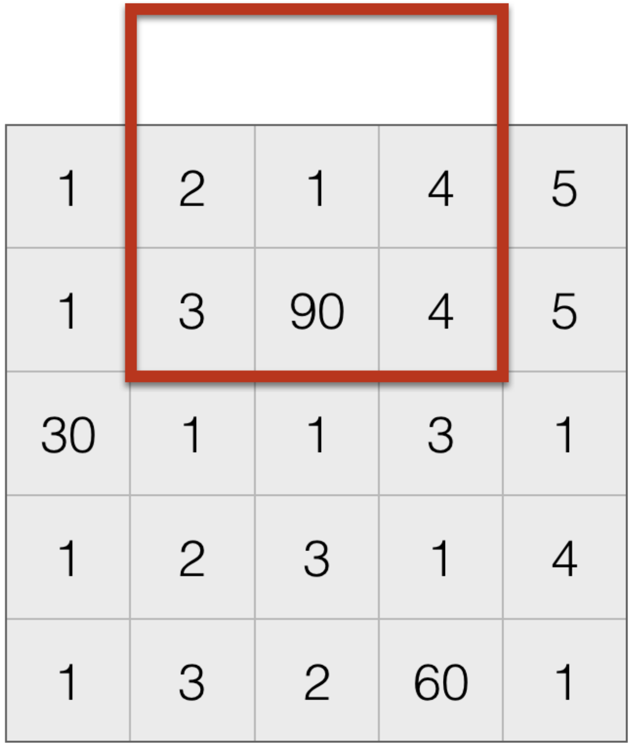
| | | | | |
|---|---|---|---|---|
| | ? | ? | ? | |
| 1 | 2 | 1 | 4 | 5 |
| 1 | 3 | 90 | 4 | 5 |
| 30 | 1 | 1 | 3 | 1 |
| 1 | 2 | 3 | 1 | 4 |
| 1 | 3 | 2 | 60 | 1 |

| | | | | |
|---|---|---|---|---|
| | 3 | 2 | 60 | |
| 1 | 2 | 1 | 4 | 5 |
| 1 | 3 | 90 | 4 | 5 |
| 30 | 1 | 1 | 3 | 1 |
| 1 | 2 | 3 | 1 | 4 |
| 1 | 3 | 2 | 60 | 1 |

Wrap around.  Useful to create an infinite domain.

# How to deal with missing (boundary values)

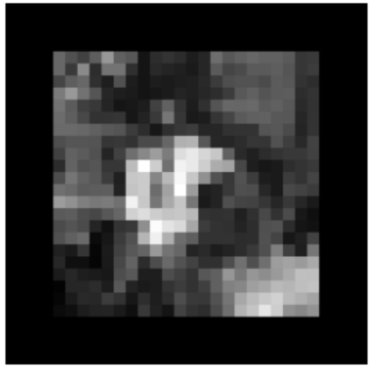


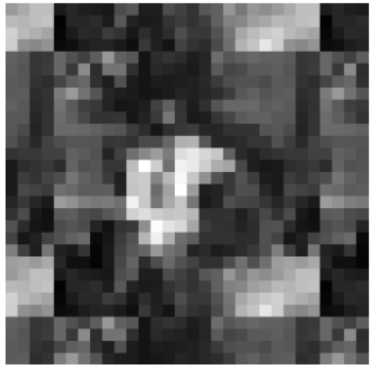

Do nothing.  Not a good choice, since the output size isn't the same as the input image, creating a host of engineering problems

# Padding

- Zero
- Constant
- Clamp or replicate
- Cyclic wrap
- Mirror

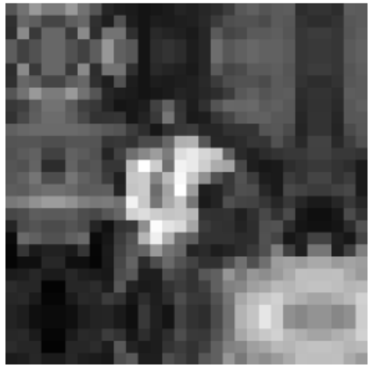


zero



wrap

clamp



mirror

# Linear Filtering Properties

- Linearity

$$\texttt{filter}(\alpha_1 f_1 + \alpha_2 f_2) = \alpha_1 \texttt{filter}(f_1) + \alpha_2 \texttt{filter}(f_2)$$

- Shift-invariance

$$\texttt{filter}(\texttt{shift}(f)) = \texttt{shift}(\texttt{filter}(f))$$

- Any linear, shift-invariant filter can be represented as a convolution.

# Properties of convolution

- Commulative: $a * b = b * a$
- Associative: $a * (b * c) = (a * b) * c$
- Distributes over addition: $a * (b + c) = a * b + a * c$
- Scalars factors out: $ka * b = a * kb = k(a * b)$
- Identity: $a * e = a$, where $e$ is unit impulse

# Linear filtering

- Remove, isolate, modify frequencies in the image
- Foundation based upon the convolution theorem

# Self-Study

- Band-pass and steerable filters
- Summed area images (integral images)

# Summary

- Linear filtering
- Cross-correlation
- Convolution
- Separable filters
- Gaussian filter