

Linear Filtering

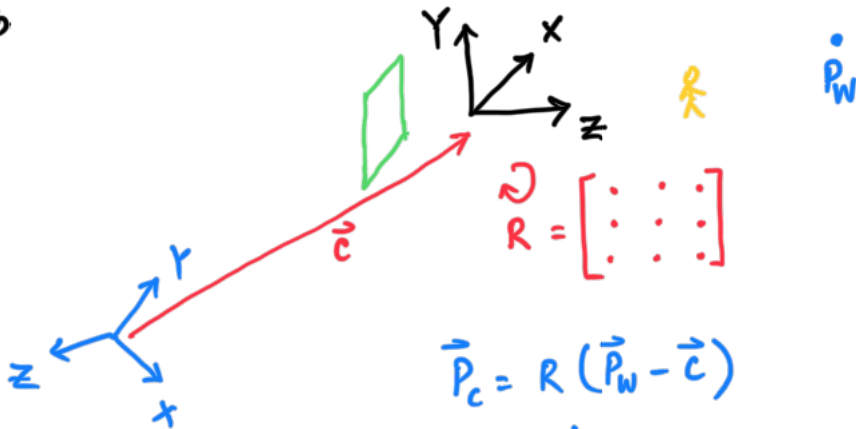
Computer Vision I (CSCI 3240U)

Faisal Z. Qureshi

<http://vclab.science.ontariotechu.ca>



Recap



$$\vec{P}_c = R (\vec{P}_w - \vec{c})$$



$$\begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix} = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} \vec{P}_w \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix} = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} \vec{p}_c \end{bmatrix}$$

$$\begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix} = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} \vec{p}_w \\ 1 \end{bmatrix}$$

Week 3 at a glance

► **Lecture 1: Convolution and correlation (Sec. 3.2)**

- Neighbourhood operators
- Cross-correlation and convolution
- Boundary handling
- Box and Gaussian filters

► **Lecture 2: Gaussians, separability, cost (Sec. 3.2.1)**

- Choosing σ and kernel size
- Separability and why it matters
- Complexity: $O(k^2)$ versus $O(k)$

Reading: Szeliski Sec. 3.2. Exercises w03a, w03b, w03c. **Lab 3** implements all of this by hand.

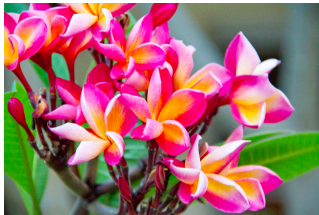
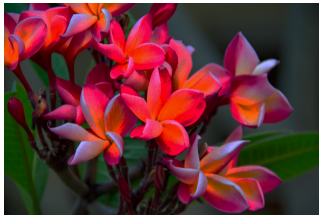
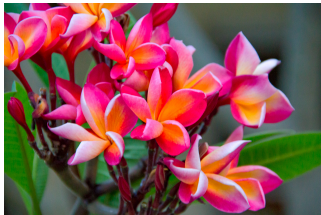
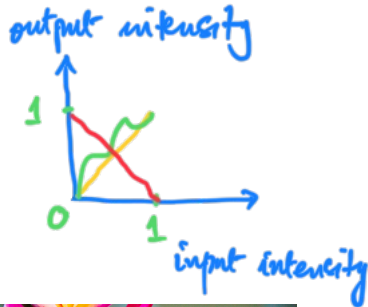
Lecture 1: Convolution and correlation

Point operators are not enough

A **point operator** maps each pixel independently:

$$I'[m, n] = g(I[m, n])$$

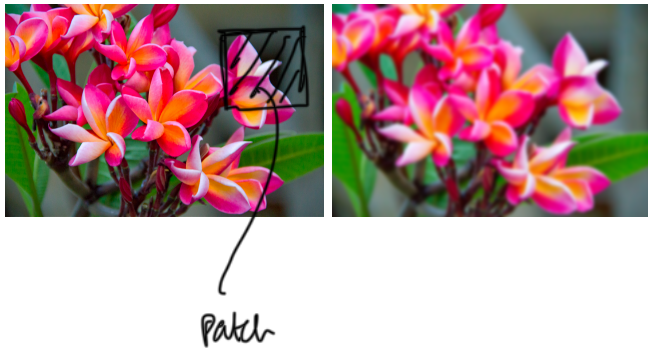
Brightness, contrast, gamma, thresholding.



Point operators are not enough

These cannot do anything that depends on *context*: they cannot blur, sharpen, find edges, or remove noise, because they never look at a neighbour.

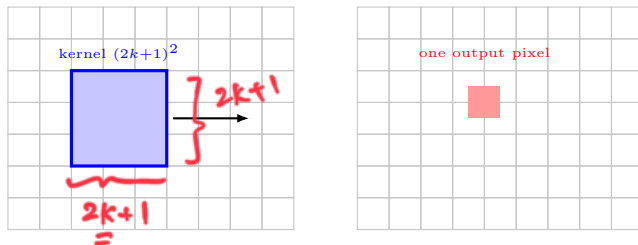
Neighbourhood operators compute each output from a window of inputs.



Cross-correlation

Slide a kernel $\mathbf{f}$ over the image and take a weighted sum:

$$I'[m, n] = \sum_{i=-k}^k \sum_{j=-k}^k \mathbf{f}[i, j] I[m + i, n + j]$$



- ▶ The kernel is $(2k + 1) \times (2k + 1)$; k is the **half-width**.
- ▶ Written $I' = \mathbf{f} \otimes I$.
- ▶ Each output pixel costs $(2k + 1)^2$ multiply-adds.

0	1	1	0
2	1	3	0
1	3	1	18
0	1	3	1

-1	0	1
-2	0	2
-1	0	1



1	0	-1
2	0	-2
1	0	-1



$$(2K+1=3) \Rightarrow K=1$$

	3		

$$\begin{aligned}
 & (0)(-1) + (1)(0) + (1)(1) \\
 & + (2)(-2) + (1)(0) + (3)(2) \\
 & + (1)(-1) + (3)(0) + (1)(1) \\
 & = 1 - 4 + 6 - 1 + 1 \\
 & = 3
 \end{aligned}$$

Convolution

Convolution flips the kernel:

$$I'[m, n] = \sum_i \sum_j \mathbf{f}[i, j] I[m - i, n - j]$$

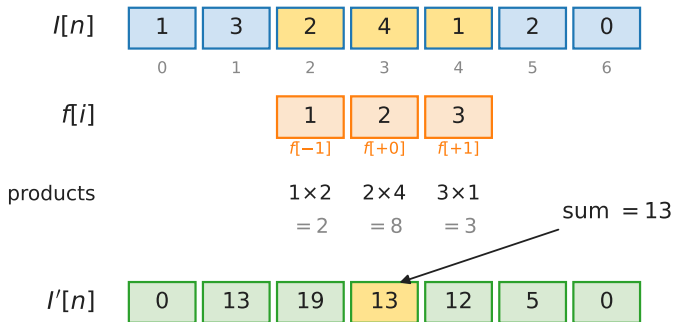
written $I' = \mathbf{f} * I$.

- ▶ For a **symmetric** kernel (box, Gaussian) the two are identical, which is why the distinction rarely bites.
- ▶ For an **asymmetric** kernel the two operations may yield different results.

Say which you implemented, and be consistent.

Cross-correlation in action, in 1D

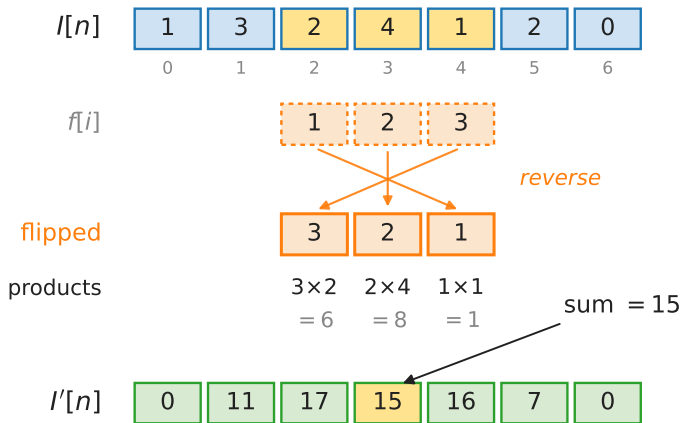
$$\text{cross-correlation: } I'[n] = \sum_i f[i] I[n+i]$$



At $n = 3$ the kernel window covers $I[2], I[3], I[4]$; the products $1 \cdot 2, 2 \cdot 4, 3 \cdot 1$ sum to 13, which lands in $I'[3]$. Slide the window right by one and repeat for every valid n .

Convolution in action, in 1D

$$\text{convolution: } I'[n] = \sum_i f[i] I[n-i]$$

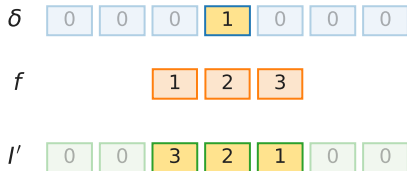


Same signal, same kernel as the previous slide, but the kernel is reversed before it slides.

The impulse reveals the flip

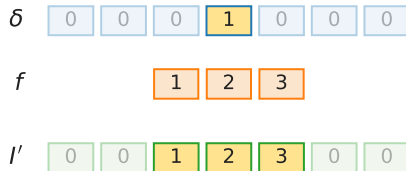
Same kernel, same impulse: the flip is now visible

cross-correlation



output at the impulse = kernel reversed

convolution



output at the impulse = kernel itself

Feed the same kernel a **unit impulse**. Correlation lays the kernel down *reversed* on the output; convolution lays it down *as is*. This is why convolution is called the filter's **impulse response**, and why derivations in signal processing use $*$ rather than $\otimes$.

Example: Convolve $\begin{pmatrix} 1 & 0 & -2 & 2 \\ 2 & -1 & 6 & 1 \\ 3 & 0 & 1 & 3 \end{pmatrix}$ with $\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$

$$\begin{aligned} & (1)(1) + (0)(2) + (-2)(1) \\ & + (2)(2) + (-1)(4) + (6)(2) \\ & + (3)(1) + (0)(2) + (1)(1) \end{aligned}$$

=

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

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

Example: Convolve $\begin{pmatrix} 1 & 0 & -2 & 2 \\ 2 & -1 & 6 & 1 \\ 3 & 0 & 1 & 3 \end{pmatrix}$ with $\begin{pmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \xrightarrow{H} \begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix}$

$\downarrow V$

$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix}$



$$[1 \quad 0 \quad 0 \quad 10 \quad 27 \quad 92]$$

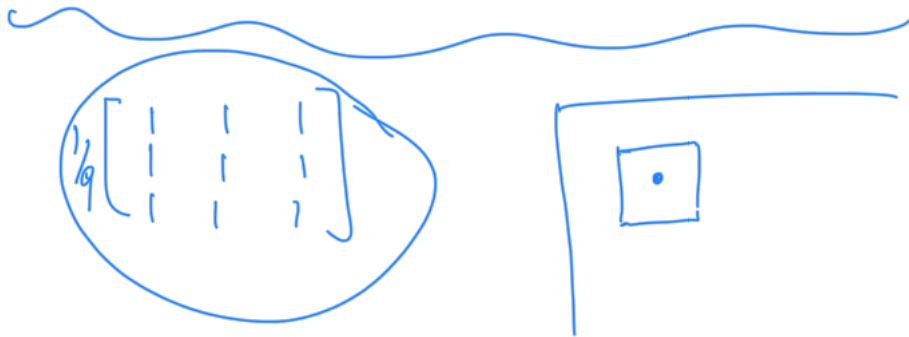
$$\begin{array}{c} -1 \\ \hline 0 \end{array} \begin{array}{c} \rightarrow \\ \downarrow \end{array}$$

$$[1 \quad 0 \quad -1]$$

↓ Flip

$$[-1 \quad 0 \quad 1]$$

Example: Convolve $\begin{pmatrix} 1 & 0 & -2 & 2 \\ 2 & -1 & 6 & 1 \\ 3 & 0 & 1 & 3 \end{pmatrix}$ with $\begin{pmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix}$



Why convolution and not correlation?

- ▶ **Commutative:** $f * g = g * f$
- ▶ **Associative:** $(f * g) * h = f * (g * h)$
- ▶ **Distributive over** $+$
- ▶ **Shift invariant**

Associativity is useful: blurring with σ_1 then σ_2 equals a *single* blur with $\sigma = \sqrt{\sigma_1^2 + \sigma_2^2}$.
Two passes become one.

Boundary handling

	?	?	?	
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

	0	0	0	
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

	?	?	?	
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

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

	?	?	?	
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

	?	?	?	
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

3000

4000

*

75 x 75



Ignore boundaries.

What is the size of the resulting image?

37

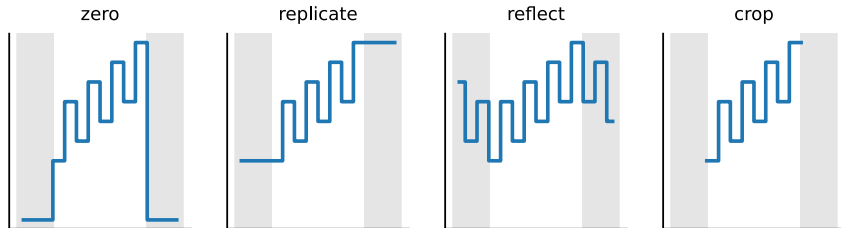
2 3 6 10 23 65

Boundary handling

The window falls off the edge. Four standard choices:

- ▶ **Crop:** output is smaller by k on each side.
- ▶ **Zero pad:** treat outside as 0. Darkens the border.
- ▶ **Replicate:** extend the edge pixel.
- ▶ **Reflect:** mirror about the boundary. Usually the best default.

Whatever you choose, state it and be consistent. A careless choice may produce a spurious “edge” all the way around the image (Lab 4).



RECAP

Convolution.

f

3	2	1	0	-1	2
---	---	---	---	----	---

-1	0	1
----	---	---

→

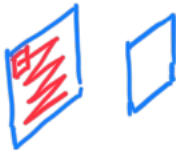
	-2	-2	-2	2	
--	----	----	----	---	--

h

1	0	-1
---	---	----

half-width = 1

Aside:



Impulse: 0 0 0 1 0 0 0

filter 1 2 3

cc: 0 0 3 2 1 0 0

* 0 0 1 2 3 0 0



F $\leadsto$ F

$$f: \quad \underline{f[0]} \quad \underline{f[1]} \cdots \underline{f[i-1]} \underline{f[i]} \underline{f[i+1]} \cdots \underline{f[N]}$$

$$h: \quad \underline{h[0] \quad h[1] \quad h[2]} \\ \underline{h[-1] \quad h[0] \quad h[1]}$$

$$cc_i: \quad h[-1] f[i-1] + h[0] f[i] + h[1] f[i+1]$$

$$\sum_{j=-1}^1 h[j] f[i+j] \quad \longrightarrow \quad \sum_{j=-k}^k h[j] f[i+j]$$

The box filter

All weights equal:

$$\mathbf{f} = \underbrace{\frac{1}{(2k+1)^2}}_{y_q} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$2k+1=3$$

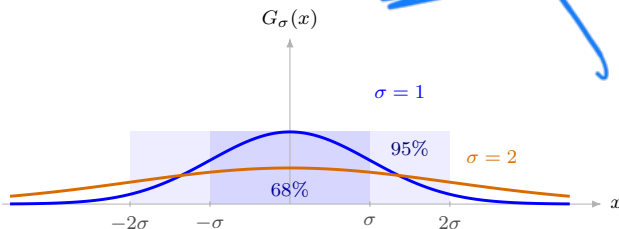
$$\Rightarrow k=1$$

- ▶ Simple, and fast via running sums.
- ▶ But: it weights a pixel k away exactly as much as the centre, which is not what “nearby” should mean.
- ▶ Its frequency response has ringing, so a blurred box-filtered image shows artefacts.

We can do better.

The Gaussian in one dimension

$$G_{\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

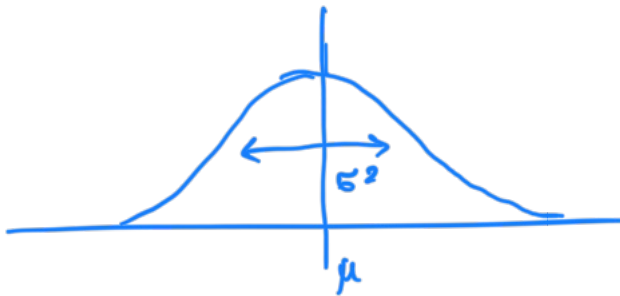


The exponent is a squared distance from the centre, measured in units of σ , so weight falls off smoothly and never quite reaches zero. About 68% of the mass lies within $\pm\sigma$ and 95% within $\pm 2\sigma$. The constant in front makes the curve integrate to 1: a filter that neither brightens nor darkens.

$$\text{Gaussian} \propto e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \left\{ \begin{array}{l} \mu=0 \\ e^{-x^2/2\sigma^2} \end{array} \right.$$

$\mu = \text{mean}$

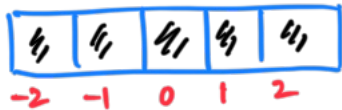
$\sigma^2 = \text{variance}$



5-tap Gaussian Filter.

$$-\frac{x^2}{2\sigma^2}$$

$$G \propto e$$



Select a value for σ^2

$$x_1, x_2, \dots, x_n$$

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i$$

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$$

$$\sigma = \sqrt{\sigma^2}$$

$$\left\{ \begin{array}{l} 2, 3, 5, 6 \\ \mu = \frac{2+3+5+6}{4} = 4 \\ \sigma^2 = \frac{(2-4)^2 + (3-4)^2 + (5-4)^2 + (6-4)^2}{4} \end{array} \right.$$

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \mu=0 \quad = \quad \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

$$x[n] = 2, -1, 0, 3, 1, 0$$

$$\frac{1}{5} [1 \quad 1 \quad 1 \quad 1 \quad 1]$$

$$y[n] = \frac{1}{5} \sum_{k=-2}^2 x[n-k]$$

$$k=2$$



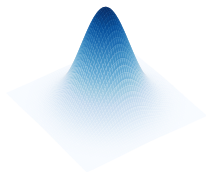
$$\begin{aligned}
 \downarrow \\
 y[0] &= \frac{1}{5} (x[0-(-2)] + x[0-(-1)] + x[0-0] + x[0-1] + x[0-2]) \\
 &= \frac{1}{5} (x[2] + x[1] + x[0] + x[-1] + x[-2]) \\
 &= (0 + (-1) + 2 + 0 + 0) / 5 \\
 &= \frac{1}{5}
 \end{aligned}$$

The Gaussian in two dimensions

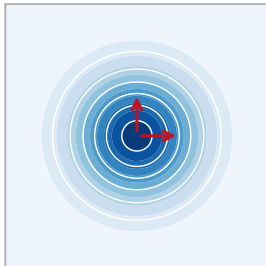
A pixel has two coordinates, so the exponent takes a covariance matrix Σ :

$$G(\mathbf{x}) = \frac{1}{2\pi\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}\mathbf{x}^\top \Sigma^{-1} \mathbf{x}\right), \quad \mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}$$

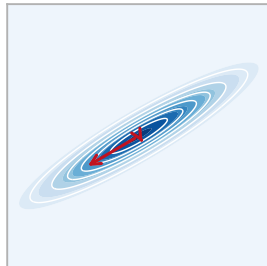
$\Sigma = \sigma^2 \mathbf{I}$: the surface



circular contours, $\sigma_x = \sigma_y$



elliptical: unequal, correlated



Σ sets the shape of the contours, and its eigenvectors, drawn in red, are their axes. Blurring an image uses the **isotropic** case $\Sigma = \sigma^2 \mathbf{I}$: circular contours, no direction preferred, which is what “nearby” should mean.

$$\vec{v} = \begin{bmatrix} 6 \\ 7 \\ 5 \end{bmatrix}$$

$$\vec{v}^T \vec{v} = [6 \ 7 \ 5] \begin{bmatrix} 6 \\ 7 \\ 5 \end{bmatrix}$$

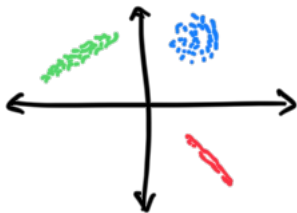
$$= 36 + 49 + 25$$

$$= \underline{\hspace{2cm}}$$



$$\begin{matrix} \mu_x, \mu_y \\ \sigma_x^2, \sigma_y^2 \end{matrix} \leftarrow \begin{matrix} \sigma_x, \sigma_y \\ \underline{\underline{\sigma_x \sigma_y}} \end{matrix}$$

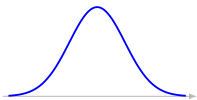
$$\begin{bmatrix} \sigma_x^2 & \sigma_x \sigma_y \\ \sigma_x \sigma_y & \sigma_y^2 \end{bmatrix} = \Sigma$$



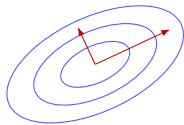
The Gaussian in k dimensions

One formula covers every case. For $\mathbf{x}, \boldsymbol{\mu} \in \mathbb{R}^k$ and $\boldsymbol{\Sigma}$ a $k \times k$ covariance matrix:

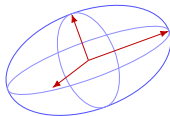
$$G(\mathbf{x}) = \frac{1}{(2\pi)^{k/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)$$



$k = 1$: a curve



$k = 2$: an ellipse



$k = 3$: an ellipsoid



$k > 3$: algebra only

Put $k = 1$ and $k = 2$ and the last two slides come back. The exponent is a squared distance in units of $\boldsymbol{\Sigma}$, so the level sets are ellipsoids centred at $\boldsymbol{\mu}$, with axes along the eigenvectors of $\boldsymbol{\Sigma}$. **Where you meet this again:** colour is $k = 3$, a feature vector is $k = 128$, and the uncertainty of an estimate is its $\boldsymbol{\Sigma}$.

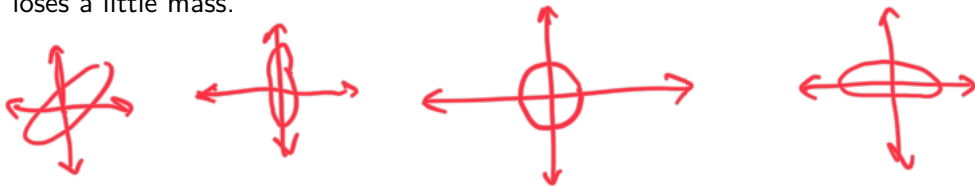
When $\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}$ the whole thing factorises into k copies of the 1D Gaussian, one per axis. That is separability, and Lecture 2 turns it into a k -fold saving.

The Gaussian filter

$$G_{\sigma}(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

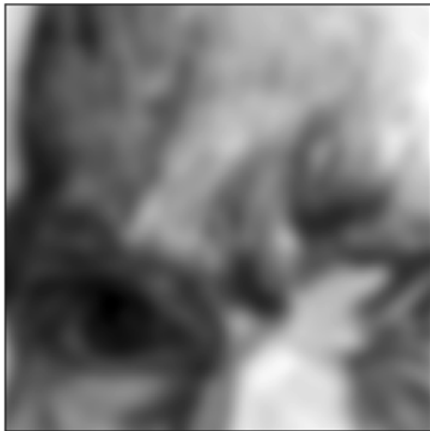
- ▶ Weight falls smoothly with distance; the kernel is rotationally symmetric.
- ▶ σ sets the **amount** of smoothing; the kernel size only sets how much of the Gaussian you keep.
- ▶ **Normalise** so the weights sum to 1, or the image gets brighter or darker.

The discrete kernel is G_{σ} sampled on the grid and then renormalised, since truncation loses a little mass.

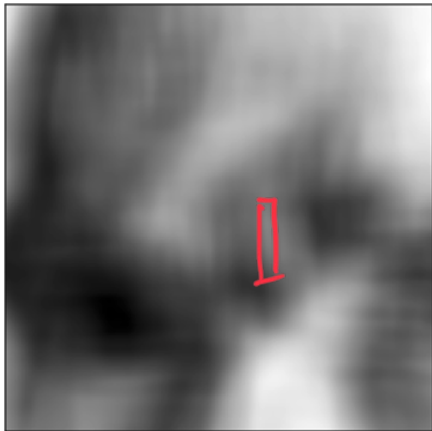


Blurring: Gaussian vs. Box filters

Gaussian Kernel



Averaging (Box) Kernel



Choosing the kernel size

The Gaussian never reaches zero, so we truncate. Truncating too early leaves a step at the window edge, which behaves like a box filter and rings.

Rule of thumb:

$$\text{half-width } k \geq 3\sigma \quad \text{i.e.} \quad \text{size} \geq 6\sigma + 1$$

which keeps about 99.7% of the mass.

An 11×11 kernel with $\sigma = 5$ is badly mismatched. It truncates at about 1σ . This is Task 4 of Lab 3, and the plot of kernel values makes it obvious.

The same kernels, as pixels

box 5×5 : every weight 0.04

0.040	0.040	0.040	0.040	0.040
0.040	0.040	0.040	0.040	0.040
0.040	0.040	0.040	0.040	0.040
0.040	0.040	0.040	0.040	0.040
0.040	0.040	0.040	0.040	0.040

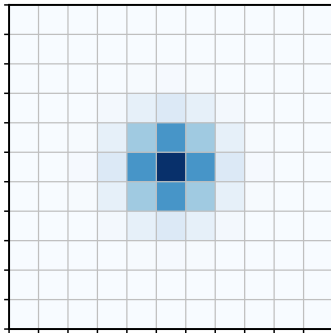
Gaussian 5×5 , $\sigma = 1$

0.003	0.013	0.022	0.013	0.003
0.013	0.060	0.098	0.060	0.013
0.022	0.098	0.162	0.098	0.022
0.013	0.060	0.098	0.060	0.013
0.003	0.013	0.022	0.013	0.003

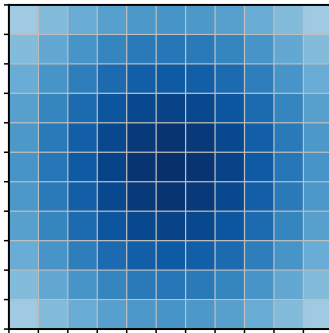
Both sum to 1 and both are drawn on the same colour scale. The box spreads 0.04 everywhere, so a pixel two steps away counts as much as the centre. The Gaussian puts 0.162 at the centre and 0.003 in the corners, a ratio of about 55:1: weight follows distance, which is what “nearby” should mean.

Truncation, as pixels

$\sigma = 1$: rim is 0.00 of the centre,
window keeps 100% of the mass



$\sigma = 5$: rim is 0.61 of the centre,
window keeps 53% of the mass



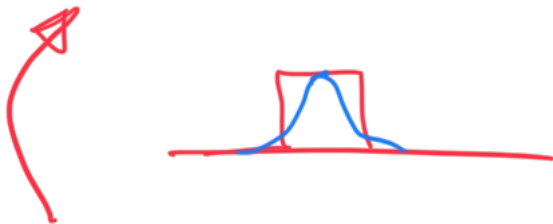
The same 11×11 window at two widths. At $\sigma = 1$ the weights have died away long before the rim, so the window holds essentially all the mass. At $\sigma = 5$ the rim still carries 0.61 of the centre and the window keeps only 53% of the mass; renormalising that stump gives you something close to a box, with the ringing that follows.

This is the 11×11 , $\sigma = 5$ mismatch from two slides back, and Task 4 of Lab 3.

Gaussian, an approximation

We often use the following approximation of a Gaussian filter:

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



Lecture 1 recap

- ▶ Point operators cannot blur, sharpen or detect edges; neighbourhood operators can.
- ▶ Correlation slides a kernel; convolution flips it first. Identical for symmetric kernels.
- ▶ Convolution is commutative, associative and shift invariant.
- ▶ Boundary handling is a decision you must state.
- ▶ Gaussian beats box; size the kernel to σ , and normalise.

Lecture 2: Separability and cost

The cost problem

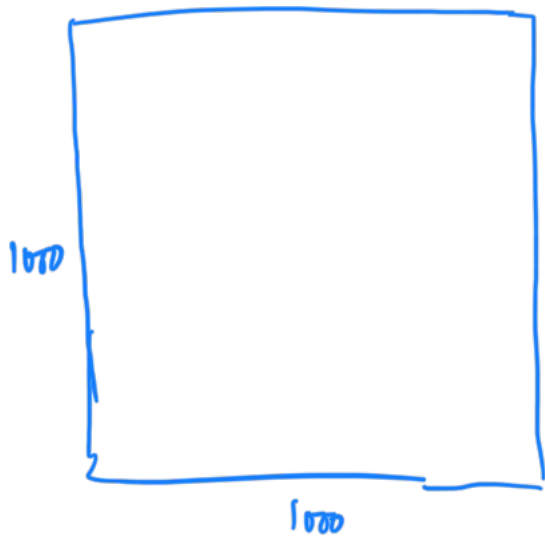
A $(2k + 1) \times (2k + 1)$ kernel on an $H \times W$ image costs

$$H \cdot W \cdot (2k + 1)^2$$

multiply-adds.

For $\sigma = 8$ we need $k \approx 24$, so $(49)^2 = 2401$ multiplications **per pixel**. On a 1242×375 image that is over a billion operations for one blur.

Quadratic growth in k is the thing to avoid.



$$\# (1\,000\,000)(11^2)$$

Outer-product:

$$\vec{v} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \longrightarrow$$

inner product:

$$\vec{v}^T \vec{v} = [1 \quad 2 \quad 1] \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = [6]$$

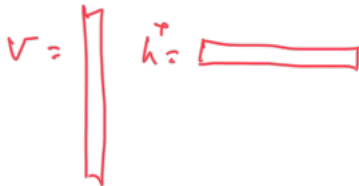
$$\vec{v} \vec{v}^T = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} [1 \quad 2 \quad 1]$$

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

Separability

A 2D kernel is **separable** if it factors into an outer product of two 1D kernels:

$$\underline{\mathbf{f} = \mathbf{v} \mathbf{h}^\top}$$



Hand-drawn diagram illustrating the separability of a 2D kernel. It shows a vertical vector labeled $\mathbf{v}$ and a horizontal vector labeled $\mathbf{h}^\top$, representing the outer product $\mathbf{f} = \mathbf{v} \mathbf{h}^\top$.

Then, by associativity,

$$\mathbf{f} * I = \mathbf{v} * (\mathbf{h} * I)$$

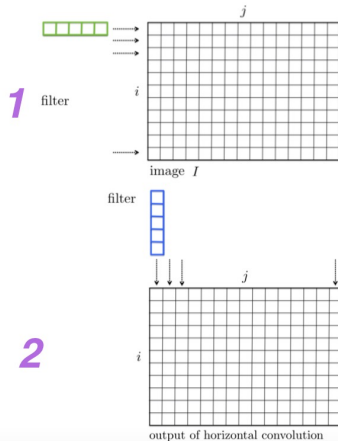
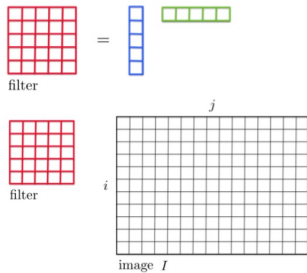
Filter every **row** with $\mathbf{h}$, then every **column** with $\mathbf{v}$.

cost: $2(2k + 1)$ per pixel, not $(2k + 1)^2$

$O(k)$ instead of $O(k^2)$. At $k = 24$ that is 98 versus 2401, a factor of about 25.

(Two 1D passes, so $2(2k + 1) = 98$, not 49.)

Separability



Example: Convolve $\begin{pmatrix} 1 & 0 & -2 & 2 \\ 2 & -1 & 6 & 1 \\ 3 & 0 & 1 & 3 \end{pmatrix}$ with $\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} [1 \ 2 \ 1]$

$$\begin{aligned} & (1)(1) + (0)(2) + (-2)(1) \\ & + (2)(2) + (-1)(4) + (6)(2) \\ & + (3)(1) + (0)(2) + (1)(1) \end{aligned}$$

$$= -1 + 12 + 4$$

$$= \textcircled{15}$$

$$\left. \begin{aligned} & (1)(1) + (2)(2) + (3)(1) \\ & = 8 \end{aligned} \right\} \left. \begin{aligned} & (1)(0) + (2)(-1) + (0)(1) \\ & = -2 \end{aligned} \right\}$$

$$\begin{aligned} & (-2)(1) + (6)(2) + (1)(1) \\ & = 11 \end{aligned}$$

$$8 \quad -2 \quad 11$$

$$\begin{aligned} & (1)(8) + (2)(-2) + (1)(11) \\ & = \textcircled{15} \end{aligned}$$

$$57 \times 49 = \cancel{513} \cancel{67} \quad (2793)$$


Is my kernel separable?

$\mathbf{f} = \mathbf{v}\mathbf{h}^\top$ is an outer product, so

$$\boxed{\mathbf{f} \text{ is separable} \iff \text{rank}(\mathbf{f}) = 1}$$

$$\begin{bmatrix} 3 & 2 & 1 \\ 12 & 8 & 4 \\ 30 & 20 & 10 \end{bmatrix}$$

Two ways to check:

1. **By hand:** every row is a scalar multiple of every other row.
2. **By SVD:** $\mathbf{f} = U\Sigma V^\top$ has exactly **one** non-zero singular value. Then $\mathbf{v} = \sigma_1 \mathbf{u}_1$ and $\mathbf{h}^\top = \mathbf{v}_1^\top$.

The split of the constant between $\mathbf{v}$ and $\mathbf{h}$ is arbitrary, and signs may both flip. Do not be alarmed when NumPy returns negative kernels; multiply them back out and check.

The Gaussian is separable

$$G_{\sigma}(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} = \underbrace{\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}}_{\text{1D in } x} \cdot \underbrace{\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}}}_{\text{1D in } y}$$

The exponent separates because $e^{a+b} = e^a e^b$. This is not a coincidence of the Gaussian's shape; it *is* the algebra.

The box filter is separable too, and can go further: see the next slide.

The box filter, worked out with a prefix sum

Precompute the running total $S[i] = \sum_{j \leq i} I[j]$, one pass over the row. Every window sum is then a subtraction:

$$\sum_{j=i-k}^{i+k} I[j] = S[i+k] - S[i-k-1]$$

Self-study

window $i \pm 1$ at $i = 4$: $1 + 5 + 9 = 15$

I	3	1	4	1	5	9	2	6
-----	---	---	---	---	---	---	---	---

S	3	4	8	9	14	23	25	31
	0	1	2	3	4	5	6	7

$$S[5] - S[2] = 23 - 8 = 15, \text{ whatever the window width}$$

$O(1)$ per pixel, whatever k , against $O(k)$ separable and $O(k^2)$ naive.

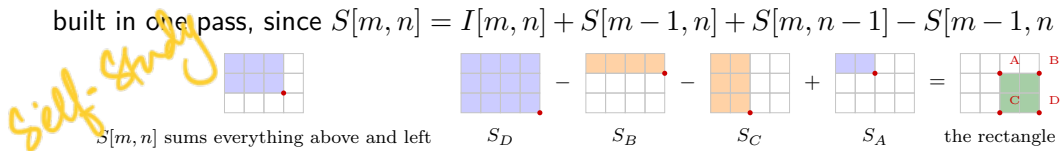
In 2D this is the integral image: any rectangle costs four lookups. Box kernels only; a Gaussian does not get this.

The same trick in 2D: the integral image

Store at every pixel the sum of everything above and to the left:

$$S[m, n] = \sum_{i \leq m} \sum_{j \leq n} I[i, j]$$

built in one pass, since $S[m, n] = I[m, n] + S[m - 1, n] + S[m, n - 1] - S[m - 1, n - 1]$.



Any rectangle is then four lookups, $S_D - S_B - S_C + S_A$. Subtracting the strip above and the strip to the left removes their shared corner twice, so S_A goes back in. One add per pixel to build it, and after that the cost of a window sum no longer depends on its size.

Also called a **summed-area table**. It is what makes sliding-window detectors cheap. Watch the numbers grow: a 12-megapixel uint8 image sums to about 3.1 billion, which overflows a signed 32-bit integer. Accumulate in int64 or float64.

In-class exercise: w03b

Show that

$$\begin{pmatrix} 1 & 3 & 1 \\ 3 & 9 & 3 \\ 1 & 3 & 1 \end{pmatrix}$$

is separable, and write down its 1D components.

In-class exercise: w03b

Show that

$$\begin{pmatrix} 1 & 3 & 1 \\ 3 & 9 & 3 \\ 1 & 3 & 1 \end{pmatrix}$$

is separable, and write down its 1D components.

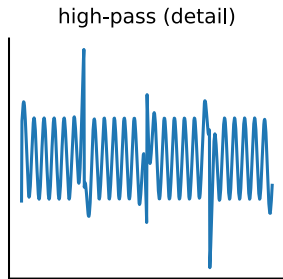
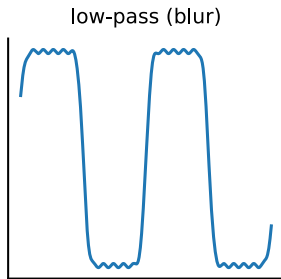
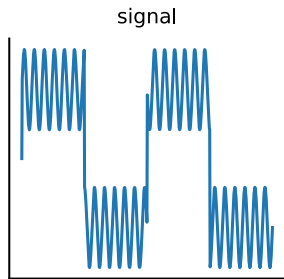
Answer sketch: rows are $1 \times (1, 3, 1)$, $3 \times (1, 3, 1)$, $1 \times (1, 3, 1)$, all multiples of one vector, so rank 1:

$$\begin{pmatrix} 1 & 3 & 1 \\ 3 & 9 & 3 \\ 1 & 3 & 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 3 & 1 \end{pmatrix}$$

Filtering and frequency

A useful way to think about what a filter does: it scales each spatial frequency.

- ▶ **Low-pass** (box, Gaussian): keeps slow variation, removes fine detail. Blurring.
- ▶ **High-pass** (derivatives, Week 4): removes slow variation, keeps edges and texture.



Why your implementation will be slow

In Lab 3 you will time three implementations. Expect:

Implementation	Growth in k
Your 2D loops	$O(k^2)$, and interpreted
Your separable version	$O(k)$
OpenCV GaussianBlur	nearly flat over this range

Separability alone will **not** close the gap to OpenCV. The rest is compiled code, SIMD, cache-friendly memory access, multithreading, and fixed-point arithmetic.

Practical notes for Lab 3

- ▶ **Convert to float before filtering.** On `uint8`, any value leaving $[0, 255]$ wraps around and you get salt-and-pepper artefacts.
- ▶ **Normalise the kernel.** Show what happens if you do not; it is worth seeing once.
- ▶ **Time the median of several runs.** A single measurement on a laptop is mostly noise.
- ▶ **Record results in a pandas dataframe.**

Week 3 recap

- ▶ Correlation slides, convolution flips; identical for symmetric kernels.
- ▶ Convolution is associative, which gives both the σ composition rule and separability.
- ▶ Gaussian: normalise, and size the kernel to at least 3σ .
- ▶ Separable $\iff$ rank 1, checkable by SVD.
- ▶ $O(k^2) \rightarrow O(k)$, and box filters can reach $O(1)$.

This week's work

- ▶ **Reading:** Szeliski Sec. 3.2.
- ▶ **In-class exercises:** w03a (linear filtering, FIR), w03b (separability), w03c (filtering and power).
- ▶ **Lab 3:** write `apply_filter_to_patch` and `apply_filter_to_image` yourself, build Gaussians, exploit separability, and measure the cost.

Next week: we differentiate the image instead of smoothing it, and start looking for edges.

Singular Value Decomposition

$$\begin{array}{ccccc} M & = & U & \Sigma & V^T \\ \begin{array}{|c|c|c|c|c|} \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline \end{array} & = & \begin{array}{|c|c|c|c|c|} \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline \end{array} & \begin{array}{|c|c|c|c|c|} \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline \end{array} & \begin{array}{|c|c|c|c|c|} \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline \end{array} \\ M \in R^{5 \times 5} & & U \in R^{5 \times 5} & \Sigma \in R^{5 \times 5} & V^T \in R^{5 \times 5} \end{array}$$

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

Singular Value Decomposition

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

$M \in \mathbb{R}^{3 \times 7}$ $U \in \mathbb{R}^{3 \times 3}$ $\Sigma \in \mathbb{R}^{3 \times 7}$ $V^T \in \mathbb{R}^{7 \times 7}$

The diagram illustrates the Singular Value Decomposition (SVD) of a matrix M . The matrix M is shown as a 3×7 grid of light gray squares, labeled $M \in \mathbb{R}^{3 \times 7}$. It is equal to the product of three matrices: U (a 3×3 grid of blue squares, labeled $U \in \mathbb{R}^{3 \times 3}$), Σ (a 3×7 grid with a diagonal of dark gray squares and white elsewhere, labeled $\Sigma \in \mathbb{R}^{3 \times 7}$), and V^T (a 7×7 grid of yellow and orange squares, labeled $V^T \in \mathbb{R}^{7 \times 7}$).

Singular Value Decomposition

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

$M \in \mathbb{R}^{7 \times 3}$ $U \in \mathbb{R}^{7 \times 7}$ $\Sigma \in \mathbb{R}^{7 \times 3}$ $V^T \in \mathbb{R}^{3 \times 3}$

The diagram illustrates the Singular Value Decomposition (SVD) of a matrix M . The equation $M = U \Sigma V^T$ is shown. The matrices are represented by grids of colored squares:

- $M \in \mathbb{R}^{7 \times 3}$: A 7x3 grid of light gray squares.
- $U \in \mathbb{R}^{7 \times 7}$: A 7x7 grid of squares with a blue-to-gray gradient, darker in the center.
- $\Sigma \in \mathbb{R}^{7 \times 3}$: A 7x3 grid where the first three rows have a dark gray square on the diagonal and white elsewhere, and the last four rows are entirely white.
- $V^T \in \mathbb{R}^{3 \times 3}$: A 3x3 grid of squares with an orange-to-brown gradient, darker in the bottom-right corner.

Singular Value Decomposition

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

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

The diagram illustrates the Singular Value Decomposition (SVD) of a 5x5 matrix M . The equation $M = U \Sigma V^T$ is shown, where each matrix is represented by a 5x5 grid of colored squares. The matrix M is a uniform light gray grid. The matrix U is a green-to-dark-green gradient grid. The matrix Σ is a diagonal matrix with a grayscale gradient from top-left to bottom-right. The matrix V^T is an orange-to-brown-to-gold gradient grid.

Singular Value Decomposition

The diagram illustrates the Singular Value Decomposition (SVD) of a matrix M . On the left, matrix M is represented as a 5x5 grid of light gray squares. This matrix is equal to the sum of five rank-1 matrices, each consisting of a scalar value, a column vector, and a row vector.

The decomposition is shown as follows:

$$M = \sigma_{11} U_1 V_1^T + \sigma_{22} U_2 V_2^T + \sigma_{33} U_3 V_3^T + \sigma_{44} U_4 V_4^T + \sigma_{55} U_5 V_5^T$$

Each term in the sum is represented by a scalar, a column vector, and a row vector:

- σ_{11} (dark gray square), U_1 (light green column vector), V_1^T (light orange row vector)
- σ_{22} (dark gray square), U_2 (light green column vector), V_2^T (light orange row vector)
- σ_{33} (dark gray square), U_3 (light green column vector), V_3^T (dark orange row vector)
- σ_{44} (dark gray square), U_4 (dark green column vector), V_4^T (dark brown row vector)
- σ_{55} (dark gray square), U_5 (dark green column vector), V_5^T (dark brown row vector)

Singular Value Decomposition

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

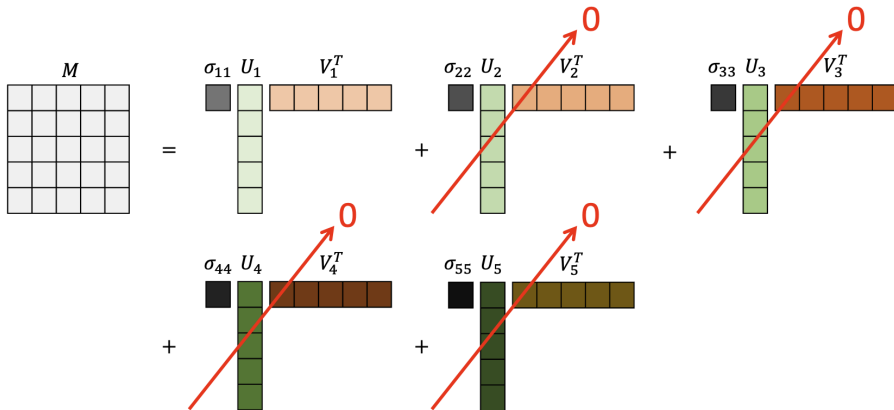
$M \in \mathbb{R}^{5 \times 5}$ $U \in \mathbb{R}^{5 \times 5}$ $\Sigma \in \mathbb{R}^{5 \times 5}$ $V^T \in \mathbb{R}^{5 \times 5}$

The diagram illustrates the Singular Value Decomposition (SVD) of a 5x5 matrix M . The equation $M = U \Sigma V^T$ is shown, with each matrix represented by a 5x5 grid of colored squares. The matrix M is a uniform light gray grid. The matrix U is a green grid with a horizontal gradient, becoming darker from left to right. The matrix Σ is a white grid with a dark gray diagonal, representing the singular values. The matrix V^T is an orange-to-brown grid with a vertical gradient, becoming darker from top to bottom.

Singular Value Decomposition

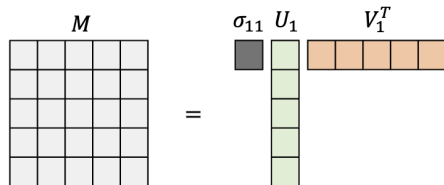
$$M = \sigma_{11} U_1 V_1^T + \sigma_{22} U_2 V_2^T + \sigma_{33} U_3 V_3^T + \sigma_{44} U_4 V_4^T + \sigma_{55} U_5 V_5^T$$

Singular Value Decomposition



Singular Value Decomposition

What if only σ_{11} is zero?



$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sigma_{11}U_1V_1^T$$

$\mathbf{M}$ is an outer-product of $\sqrt{\sigma_{11}}U_1$ and $\sqrt{\sigma_{11}}U_2$.

Copyright and License

©Faisal Z. Qureshi



This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.