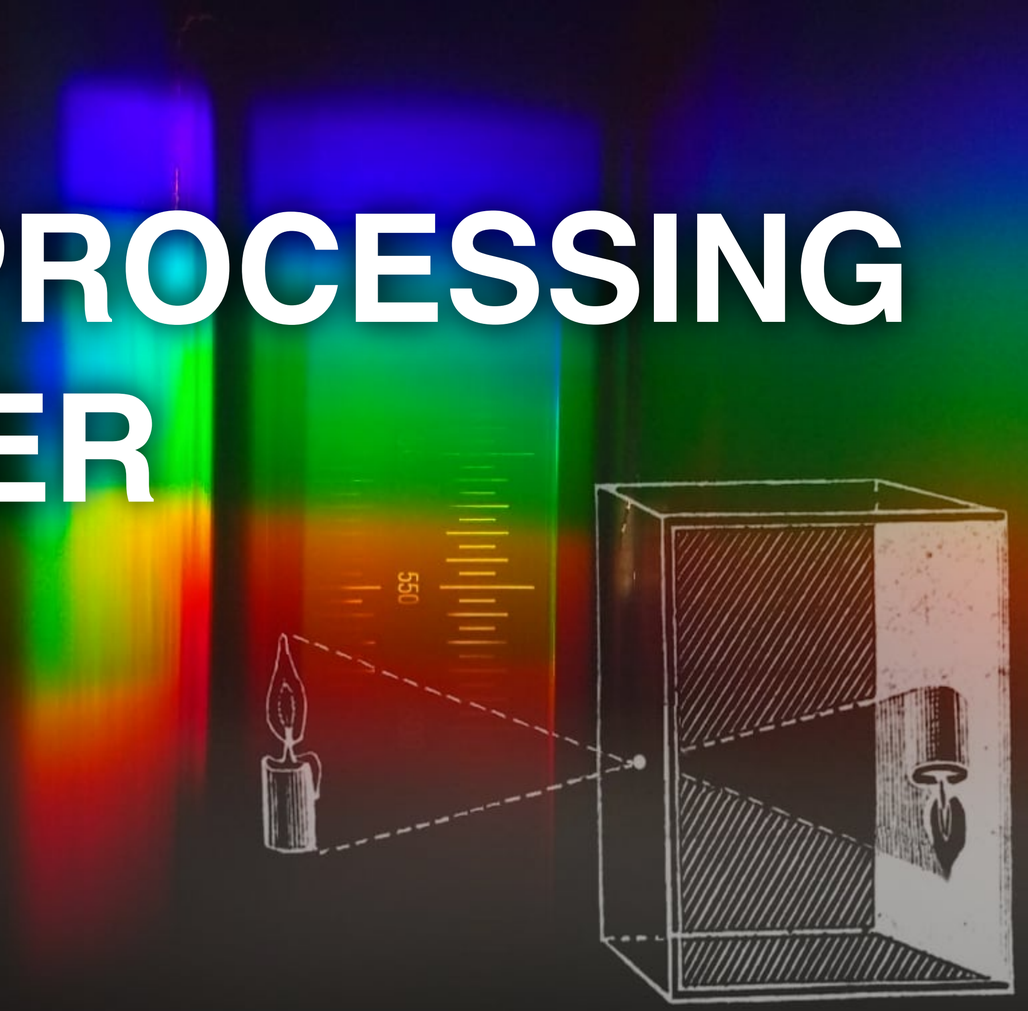
ADVANCES IN COMPUTER VISION

MODULE 1: Geometry

LINEAR IMAGE PROCESSING AND THE FOURIER TRANSFORM



Prof. Vincent Sitzmann

Prof. Frédo Durand

Stopped slide 140 (just before Laplacian pyramids)
Timing with p-set is challenging

Fredo Notes

- I kind of disagree that an image maps arbitrary rays to a color. That's the plenoptic funct
- When discussing magnitude of FT, we implicitly use a complex-valued FT
- Added different perspective on convolution, less local : sum of shifted versions of the en
- Laplacian pyramid is not technically a basis (overcomplete)
- Helicopter aliasing: https://www.instagram.com/reels/DUGgFVLkZtu/
    - Kid book Moire, strobe light, etc., Edgerton
    - Bring my light
- Positional encoding
- Vincent says I should move stuff from next lecture to this one
- Tangent kernels and eigen stuff?
- Momentum, Hessian
- U-Net vs Laplacian/Gaussian
- Discussion of eigen / singular vectors and when you can get them
- In linear shift-invariant physical systems you get sine waves in both space and time.
- Honestly I am unconvinced representation theory is that useful in vision.
- Forgot to say Fourier is orthogonal (not guaranteed)
- Discuss that NN are often modeled as linear for analysis/theory purposes
    - Layers are indeed linear + one simple non-linearity.

Feedback from Vincent from first year (2025):
make 1D and 2D clearer
Motivate laplacian pyramid better
Maybe spend less time on Fourier transform, people already know it…?

- Clarify that matrix multiplication is generally *not commutative*
- *A matrix is circulant iff it commutes with a shift*

Is this claim true?



Jean Baptiste Joseph Fourier
(1768-1830)

The Fourier series claim (1807):

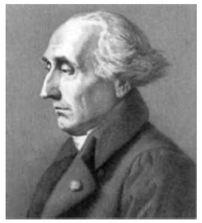
> *'<u>Any</u> univariate function can be rewritten as a weighted sum of sines and cosines of different frequencies.'*

Well, almost.

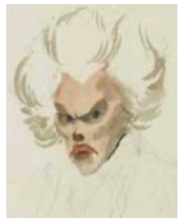
- The theorem requires additional conditions.
- Close enough to be named after him.
- Very surprising result at the time.



Malus



Lagrange



Legendre



Laplace

The committee examining his paper had expressed skepticism, in part due to not so rigorous proofs

Motivation for representing and manipulating images as vectors

- **Images as vectors in high-dimensional space**
  - Many phenomena are linear (e.g. optical blur) *Science perspective*
  - Many useful operations are linear (brighten, blur, sharpen, convolution) *Engineering perspective*
  - Linear is easy/cheap
  - Benefit from power of linear algebra (e.g. for inverse problems)
- **Change of basis, Fourier transform and its cousins**
  - We'll focus on eigenvector perspective and why it's so important
  - Make some things simpler/clearer
  - Match human perception
  - Used everywhere



Image nano banana

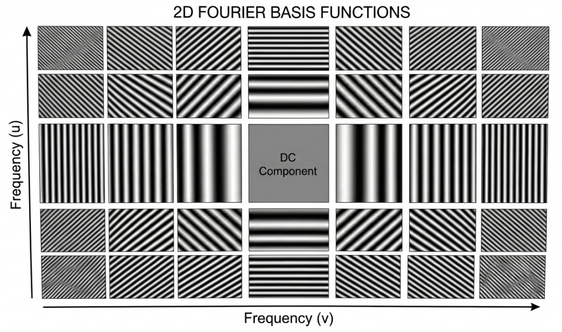


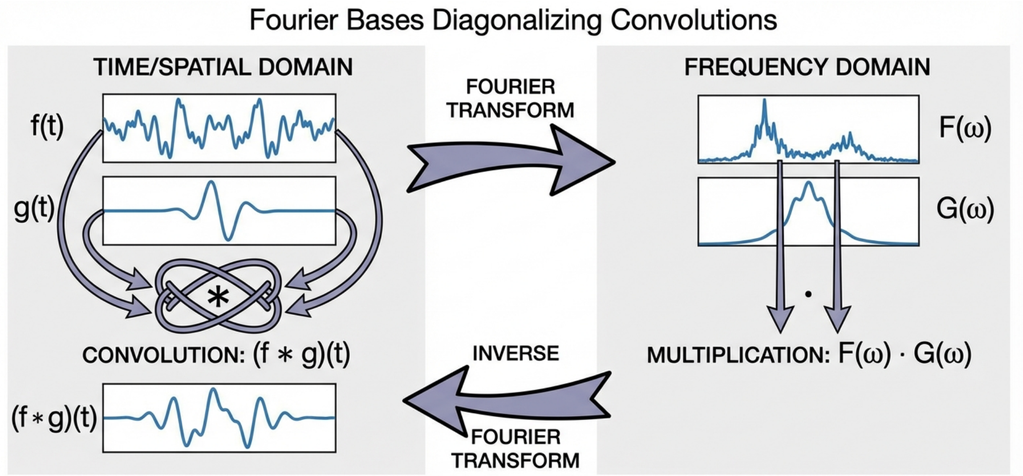
Image nano banana

Spoiler: Why Fourier ?

- **Diagonalizes shift / convolution**
- Aliasing
- Shift invariance in images / Translation
- Human visual perception

- Positional encoding

$$\begin{bmatrix} a & b & c & d & 0 & 0 \\ 0 & a & b & c & d & 0 \\ 0 & 0 & a & b & c & d \\ 0 & 0 & 0 & a & b & c \\ 0 & 0 & 0 & 0 & a & b \\ 0 & 0 & 0 & 0 & 0 & a \end{bmatrix} \Rightarrow \begin{bmatrix} \lambda_0 & & & & & \\ & \lambda_1 & & & & \\ & & \lambda_2 & & & \\ & & & \lambda_3 & & \\ & & & & \lambda_4 & \\ & & & & & \lambda_5 \end{bmatrix}$$



Fourier Basis functions (complex exponentials $e^{i\omega t}$) are eigenvectors of the convolution operator, making convolution in time equivalent to simple multiplication in frequency.

Image nano banana… which used mit.edu as a reference ;-)

When we use linearity

- **Things are actually linear**
  - e.g. light
  - Sometimes need to reformulate to expose linearity
    - (e.g. homogenous coordinate, log transforms)
- **Simplify things to make linear**
  - Change the scenario
  - Use Taylor expansion
- **Limit yourself to linear tools because it’s easier and cheaper**
- Note: the same apply to quadratic loss/energy for optimization (because derivative is linear)

High and Lows of this set of lectures

- Math is amazing, illuminates so much, and provides solutions that are more efficient
- The real world is complicated and idealized math usually doesn’t apply
- The insights still matter

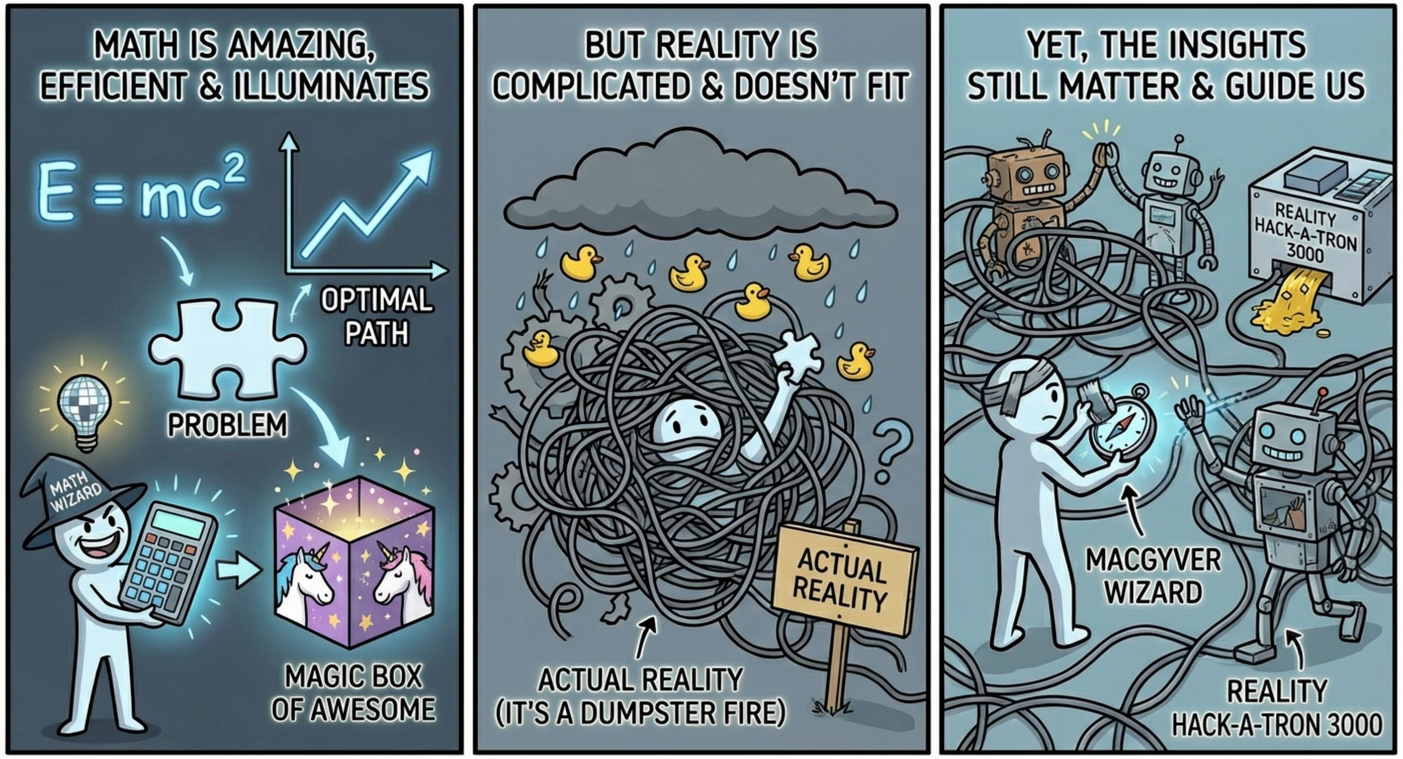
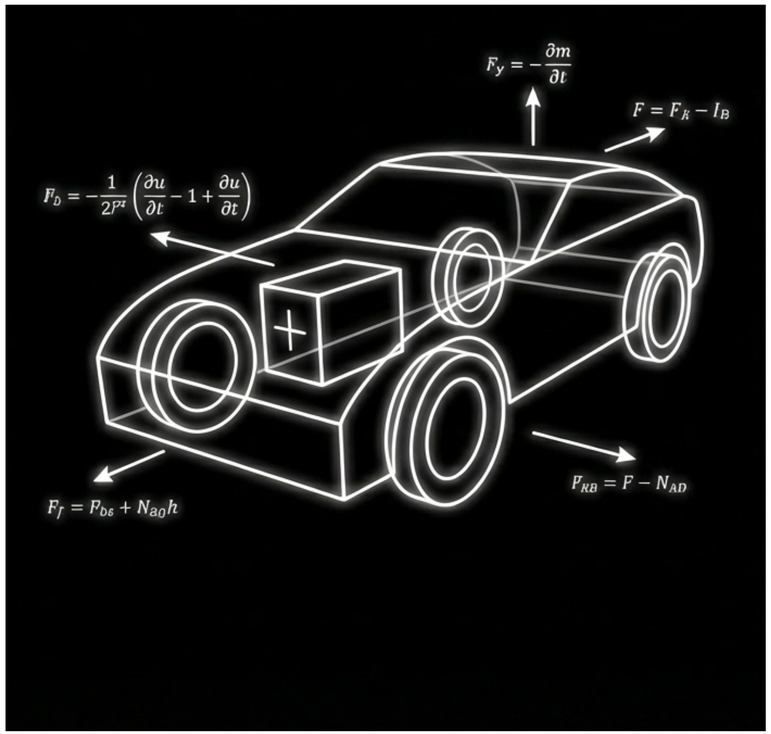


Image by nano banana

Theory tension

- Simple enough to write theorem and derivations
- Close enough to reality





Linear Image Processing: A Fundamental Toolbox for Image Editing & Understanding



What is this arrow?

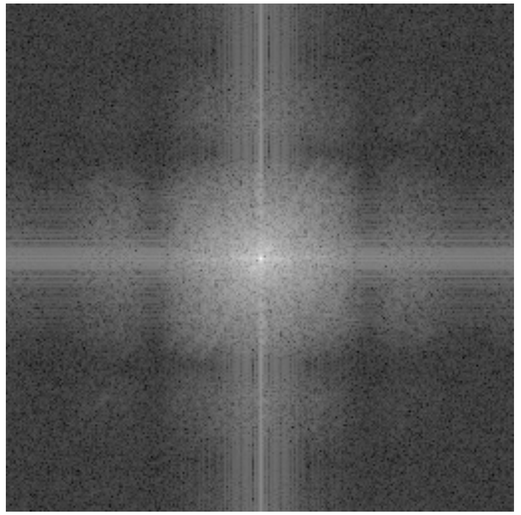
Edge Detection



Blurring



Fourier Analysis



Scale Spaces









Images as Functions

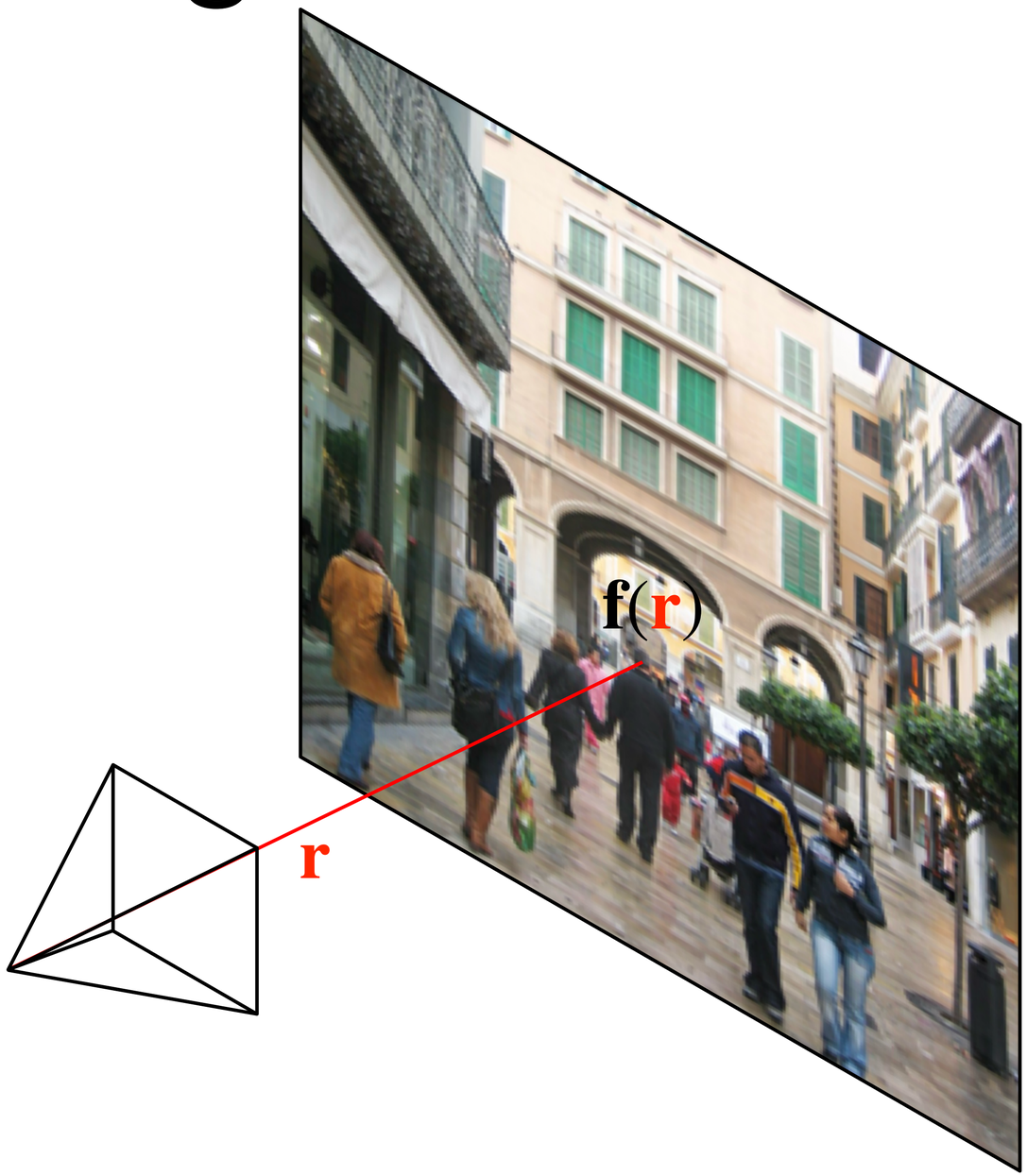


Image is a function f that
maps rays (direction + origin) to color
where the origin is the viewpoint

$$\mathbf{f} : \mathscr{R} \rightarrow \mathbb{R}^3$$

$$\mathbf{f}(\mathbf{r}) = \mathbf{c}$$

Images as Functions

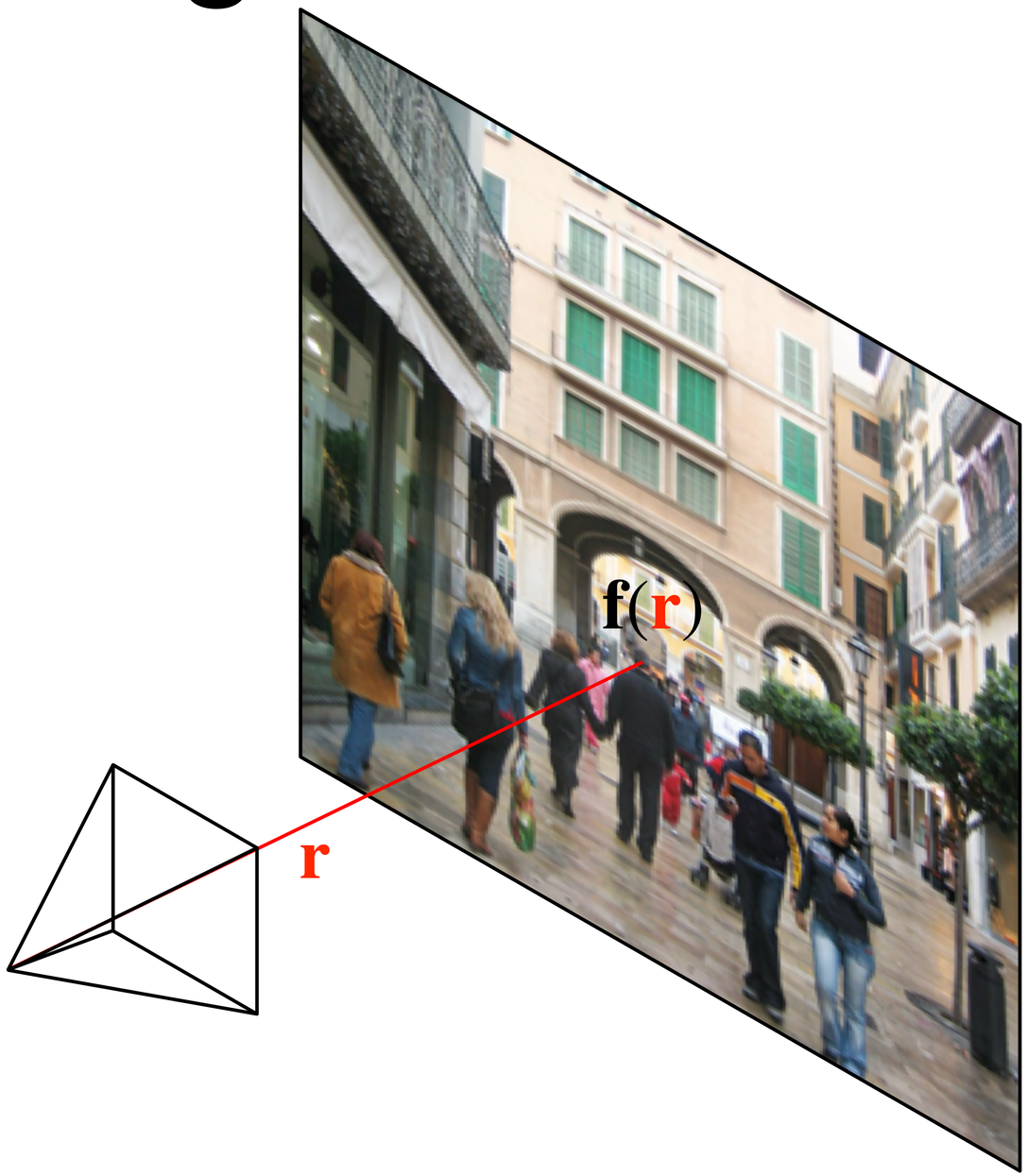


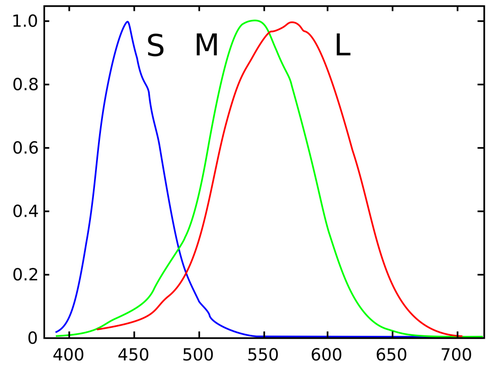
Image is a function f that
maps rays (direction + origin) to color
where the origin is the viewpoint

$$\mathbf{f} : \mathscr{R} \rightarrow \mathbb{R}^3$$

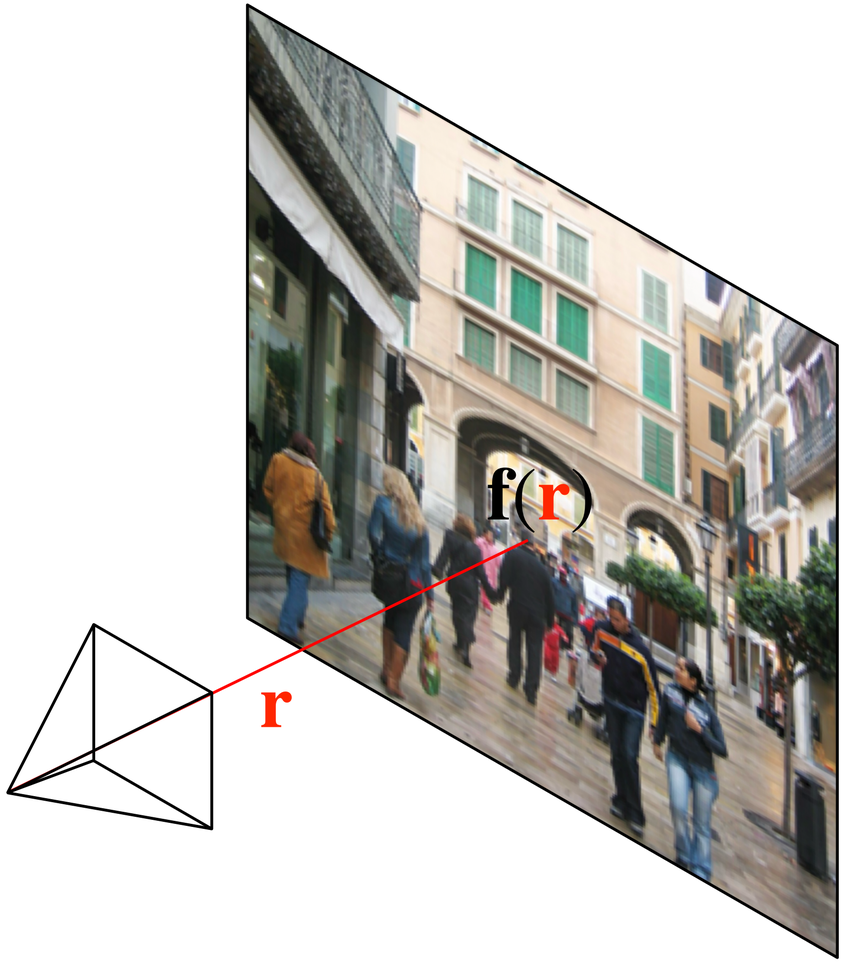
$$\mathbf{f}(\mathbf{r}) = \mathbf{c}$$

In this class we take it for granted that color can be described by a triplet R, G, B

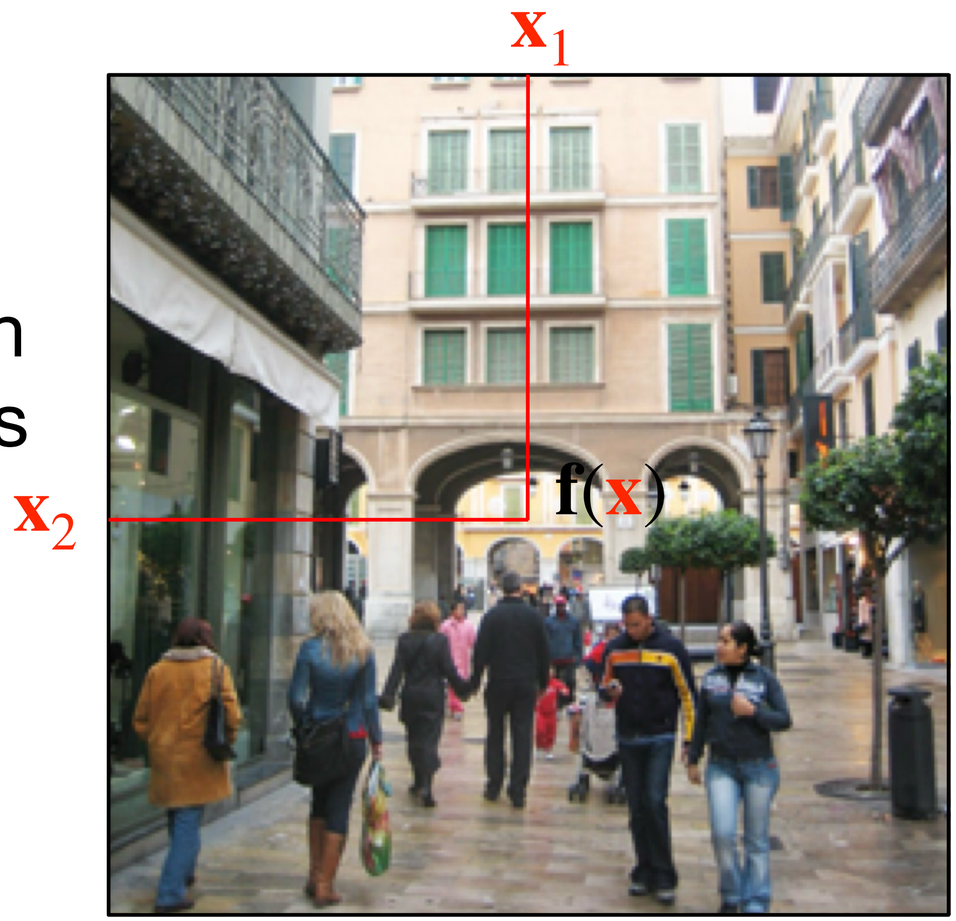
Each color channel is the linear projection of a full spectrum onto the spectral response of human cones or digital sensors
More linear algebra!



Images as Functions



Identify rays with pixel coordinates



$\mathbf{f}:\mathbb{R}^2 \to \mathbb{R}^3$ $\mathbf{f}(\mathbf{x})=\mathbf{c}$

Images as Functions

Image is a function f that
maps 2D pixel coordinates to color

$$\mathbf{f} : [0,1] \times [0,1] \rightarrow \mathbb{R}^3$$

$$\mathbf{f}(\mathbf{x}) = \mathbf{c}$$

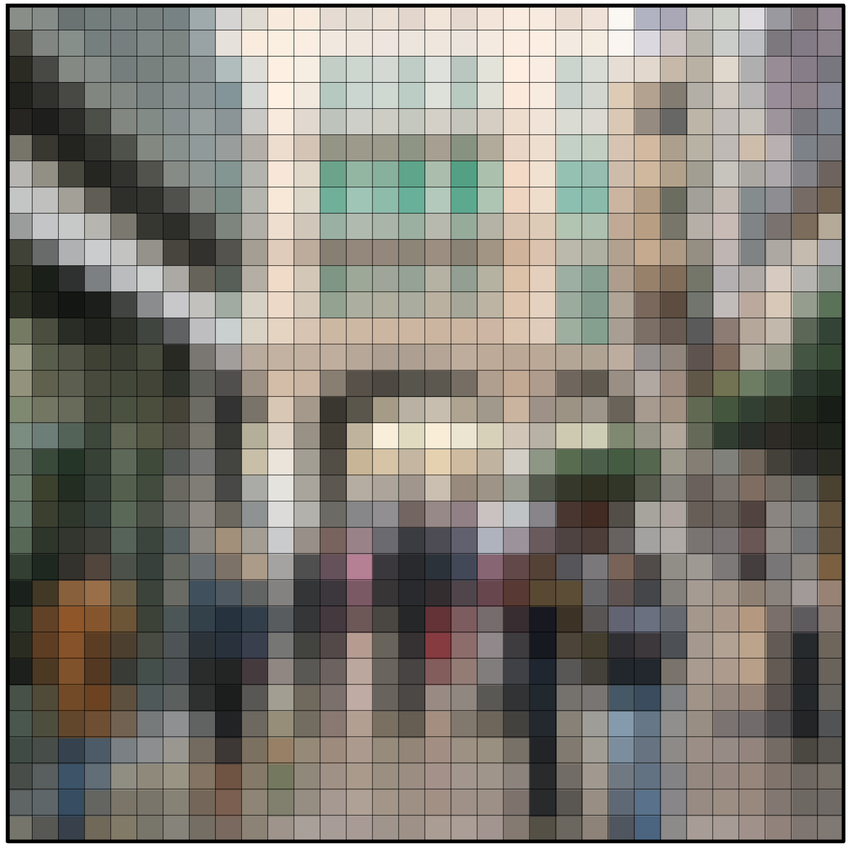


$$\mathbf{f}:\mathbb{R}^2\rightarrow\mathbb{R}^3 \qquad \mathbf{f}(\mathbf{x})=\mathbf{c}$$

Images as Functions



Discrete Sampling



$\mathbf{f}:\mathbb{R}^2 \to \mathbb{R}^3$ $\mathbf{f}(\mathbf{x})=\mathbf{c}$

Although a lot of what we will study is still valid for continuous images

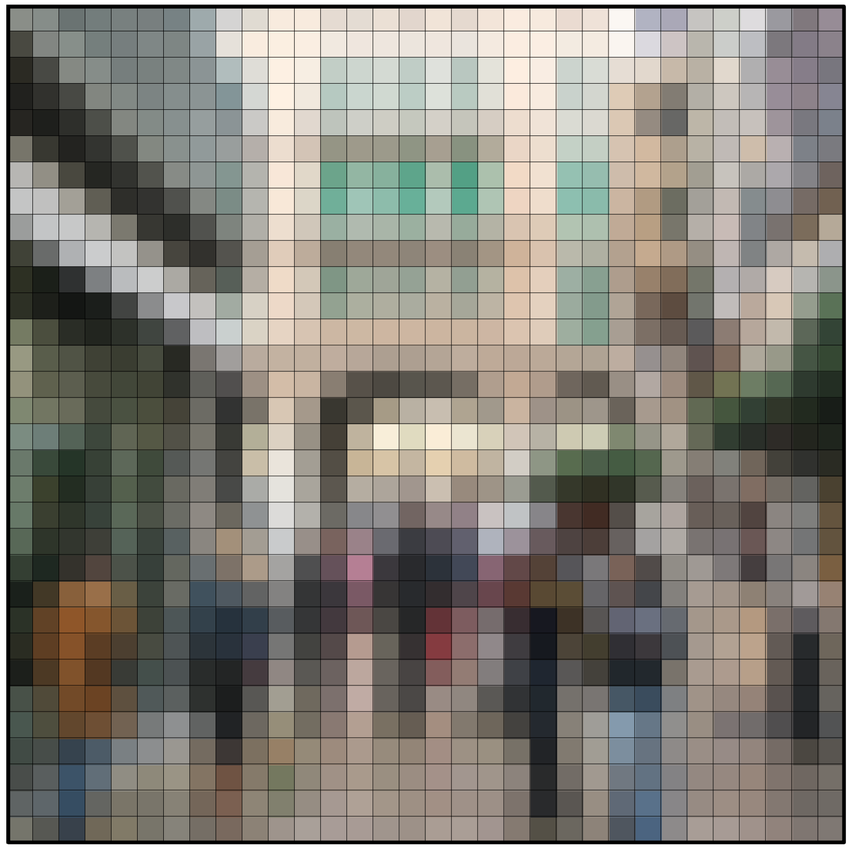
Images as Vectors

Images can be represented as vectors of shape

$$\mathbf{f} \in \mathbb{R}^{H \times W \times ch}$$

Sometimes literally,
sometimes conceptually

Note that the image is, nevertheless, a *function* over *pixel coordinates.*



"Flatten"



Why images as linear algebra vectors?

- Basic algebraic operations: scale (brighten/darken, not zoom!), add
- Linear operators such as (de)blurring
- Inverse problems
- Programming constructs (PyTorch, tensors)

- Gradient descent, etc.

- Conceptual vs data conversion

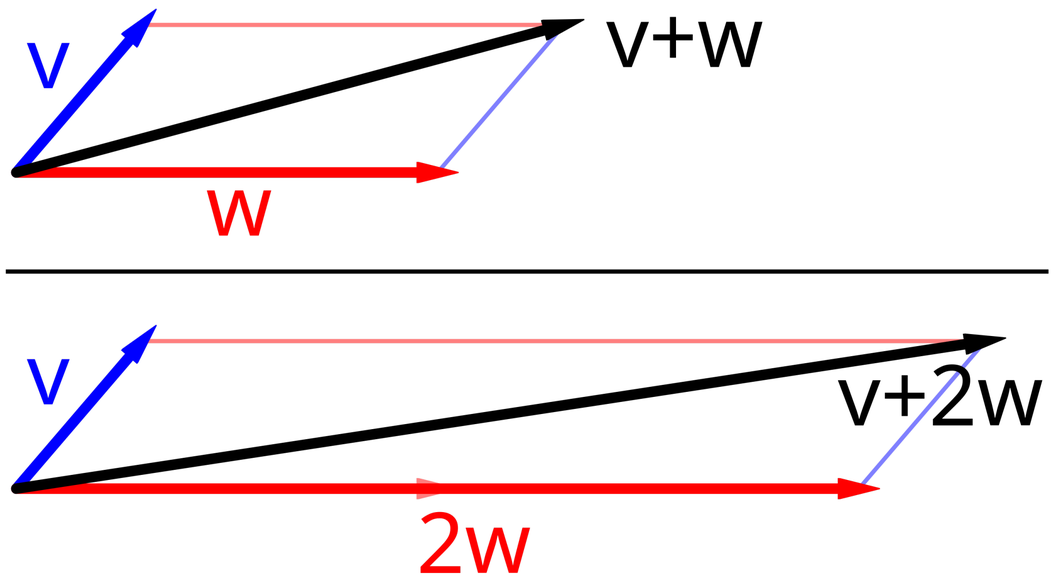
(Very quick) Math Recap

- Vector Spaces
- Inner Product Spaces
- Vector Space Bases
- Linear Maps
- Matrix Representations of Linear Maps
- Looking at Linear Maps in a different basis

Recap: Vector Spaces & Inner Product Spaces

A vector space V is a set such that its elements $\mathbf{v}, \mathbf{w} \in V$ can be summed and multiplied by scalars, and their output is still in V.

Inner Product Spaces are vector spaces where there also exists an inner product, denoted as $< \mathbf{v}, \mathbf{w} >$. Allow reasoning about lengths, distances, angles, …



By IkamusumeFan - Plot SVG using text editor., CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=41666856

Modeling the Space of Images as an Inner Product Space

- All of the following operations make sense for images, too:
  - Summing images
  - Scaling them by scalars
  - Compute an inner product
  - Compute their norm
- Hence, you can think of images as *vectors* that are elements of an *inner product space.*



Scaling an image

Different from scaling the 2D domain of images (zoom)



Note : Vector for image vs. coordinates

For example:

Scale image : brighten/darken



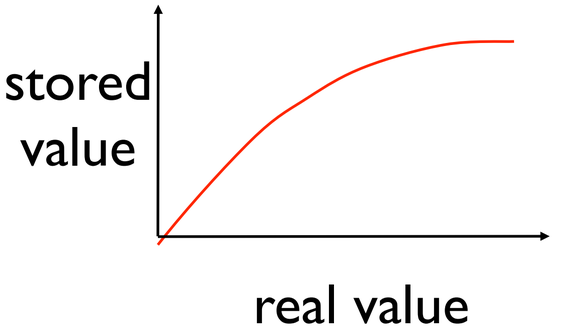
vs

scale coordinates : zoom in/out



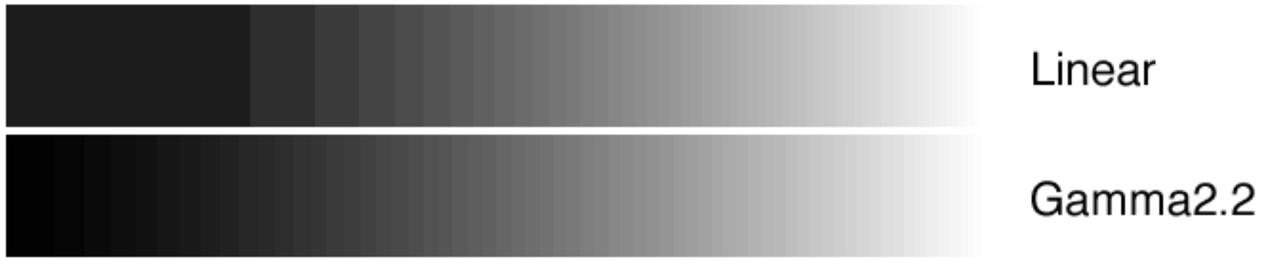
Are images well-modeled as a linear space?

- More complicated than it seems
- True for optical blur, shift
- But careful: **images are usually not encoded linearly**
  - Gamma encoding: encode $x^{1/\gamma}$
    - Better use of quantification levels for 8-bit storage
    - More uniform treatment of dark vs. bright values



Well, even that is an oversimplification ;-(see sRGB definition

- And in some cases working with log values makes more sense
- Good news: **most of the time you don't need to worry about it.**



Concretely, gamma encoding

- If the camera records an amount of light that is x at a given pixel
- What gets stored is encoded as $x^{1/\gamma}$
- When displaying an image, the OS or display or driver inverts the mapping

- Most processing/AI/vision is blissfully ignorant of this
- Important only if you care about physical linear processes (e.g. optical blur, camera shake blur, inverse physically-based light simulation)

Recap: Bases for Vector Spaces

A basis of a vector space V is a set $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\} \in V$ such that:

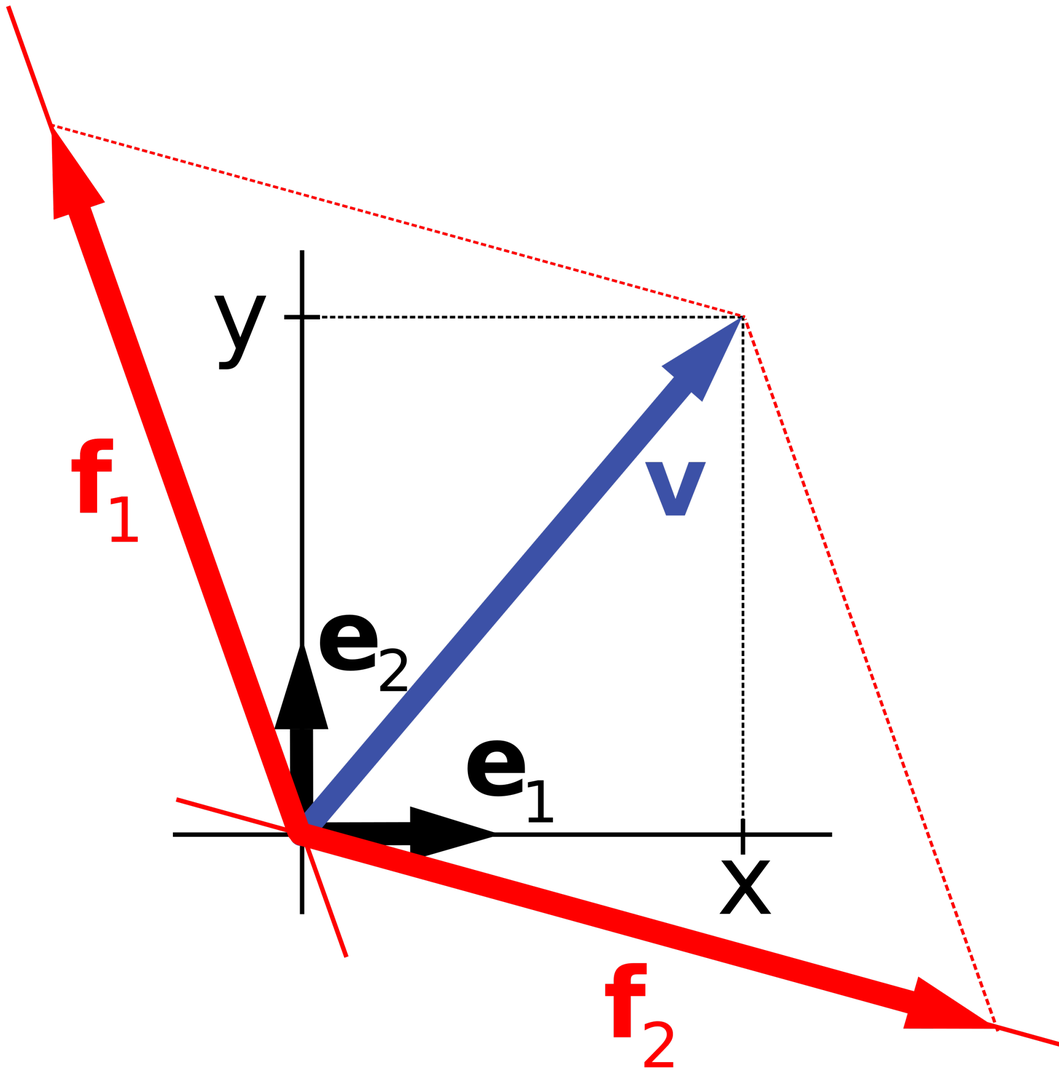
Any vector $\mathbf{v} \in V$ can be expressed as weighted sum of set elements:

$$\mathbf{v} = a_1\mathbf{v}_1 + \ldots + a_n\mathbf{v}_n$$

with scalar coefficients

$$\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{R}^n,$$

and the elements $\mathbf{v}_i$ are linearly independent, i.e., they cannot be expressed as linear sum of the other basis vectors.



By Jakob.scholbach - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=5779225

Recall: Linear Maps

Given vector spaces V and W, the map $L : V \to W$ is linear if for all $v, w \in V$ and $\alpha, \beta \in \mathbb{R}$:

$$L(\alpha v + \beta w) = \alpha L(v) + \beta L(w)$$

Matrix representations of linear maps

Given an n-dimensional vector space V with basis $\{v_1, \ldots, v_n\}$ and another W with basis $\{w_1, \ldots, w_n\}$, then $L : V \to W$ can be uniquely expressed as a matrix $\mathbf{L} \in \mathbb{R}^{m \times n}$:

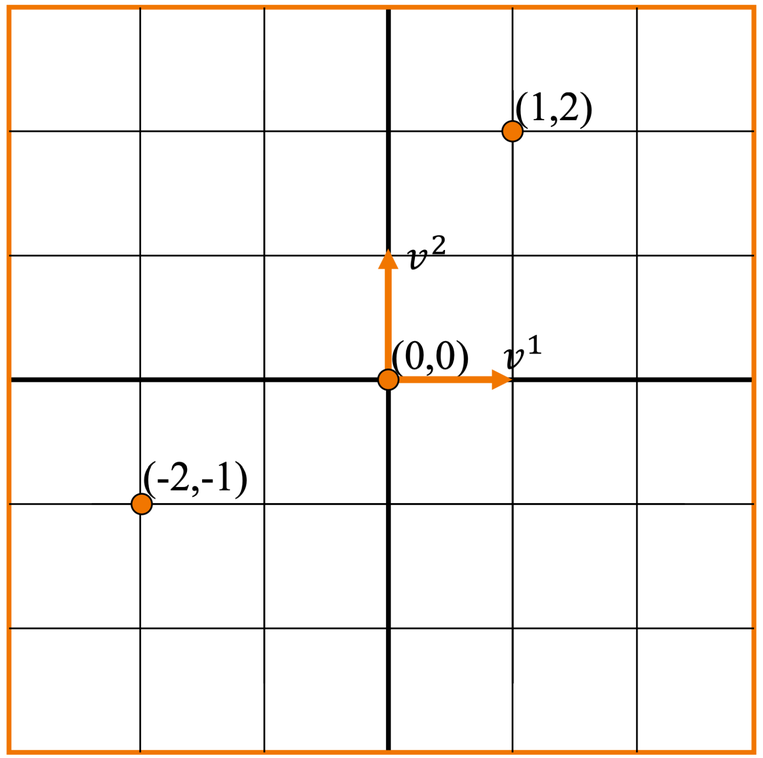
$$\mathbf{b} = \mathbf{La}$$

where

$$b_1 w_1 + \ldots + b_m w_n = L(a_1 v_1 + \ldots, + a_n v_n)$$

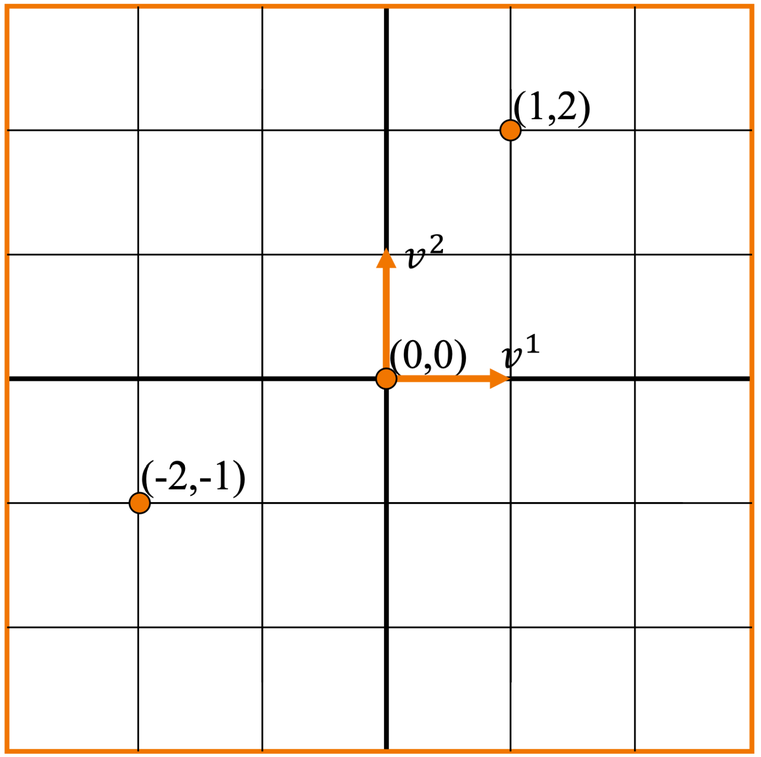
In words: any linear map can be written as a matrix that acts on the vector of basis coefficients.

Change-of-basis via matrix multiplication



Adapted from Misha Kazhdan “FFTs in Graphics and Vision”

Change-of-basis via matrix multiplication

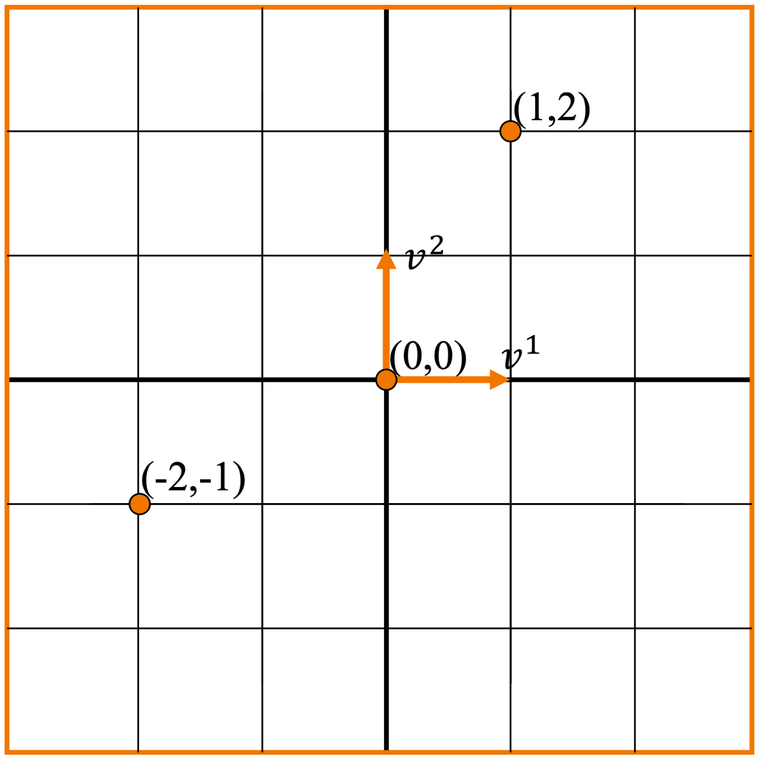


$$\mathbf{B}^{-1} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$$



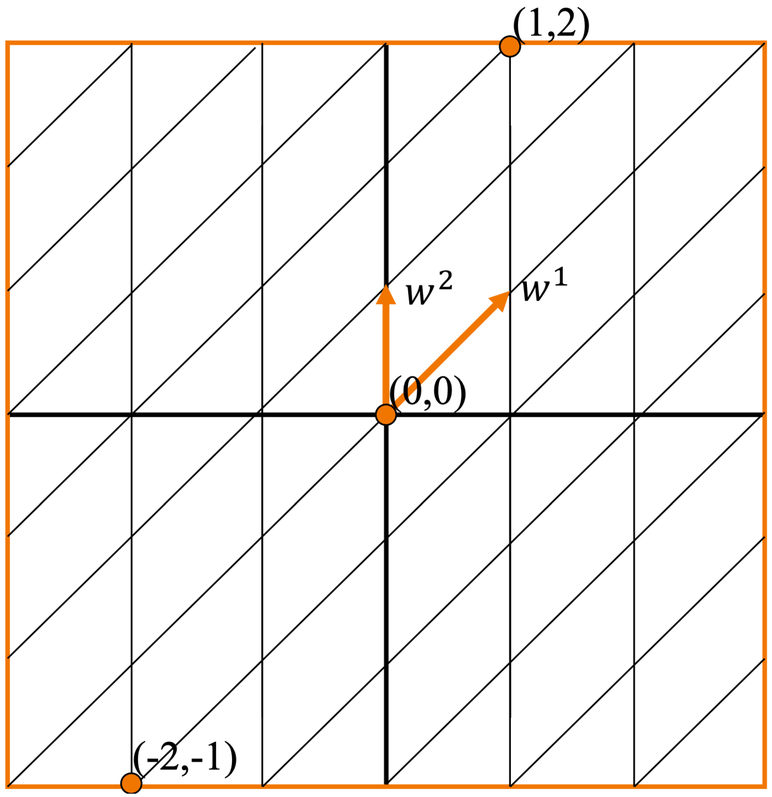
Adapted from Misha Kazhdan “FFTs in Graphics and Vision”

Change-of-basis via matrix multiplication



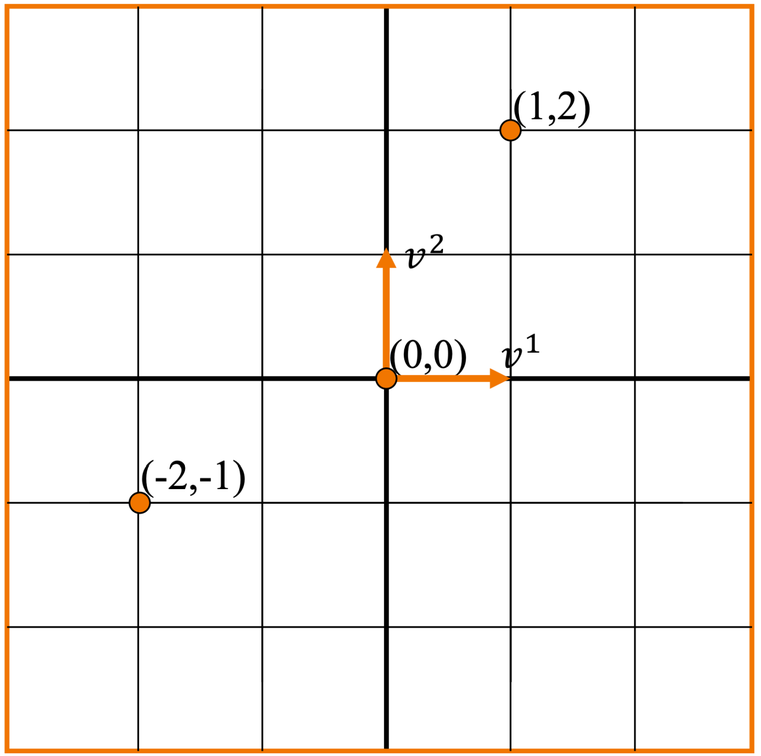
$$\mathbf{B}^{-1} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$



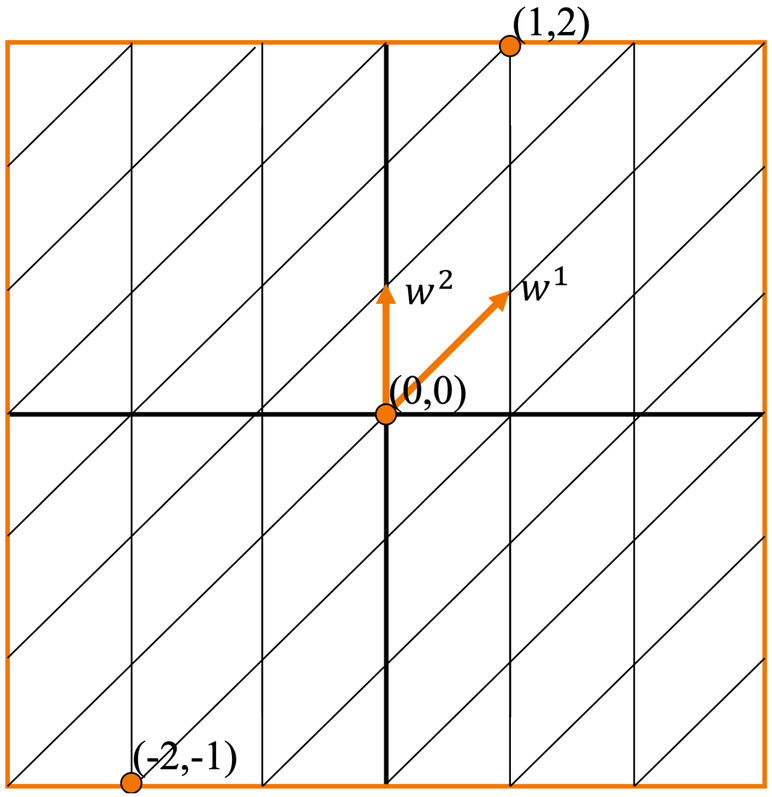
Adapted from Misha Kazhdan “FFTs in Graphics and Vision”

Change-of-basis via matrix multiplication



$$\mathbf{B}^{-1} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$



$$\mathbf{x}_v = \mathbf{B}\mathbf{x}_w$$

Adapted from Misha Kazhdan “FFTs in Graphics and Vision”

Change-of-basis for Linear Maps

We can express a linear operator L in different bases. Given…

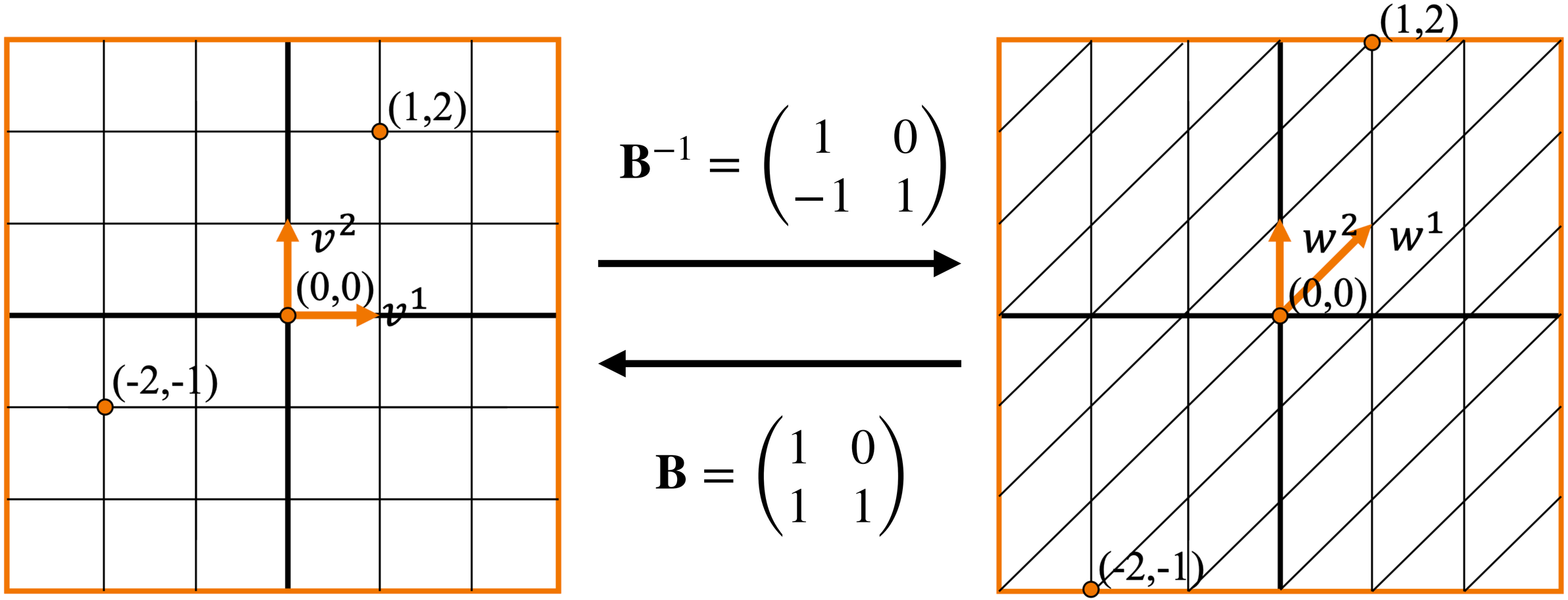
An n-dimensional vector space V,

Two bases for V, $\{v_1, \ldots, v_n\}$ and $\{w_1, \ldots, w_n\}$, and

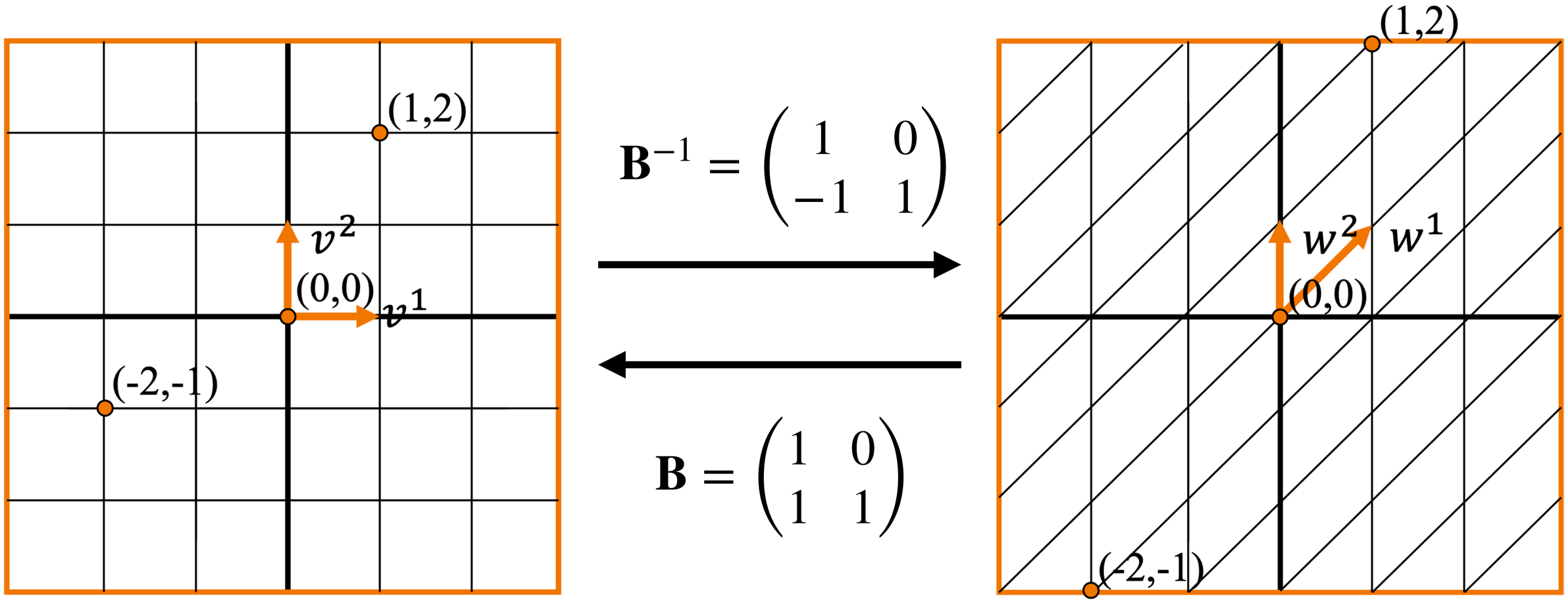
A linear operator L represented by matrix $\mathbf{L}$ in terms of the basis $\{v_1, \ldots, v_n\}$

Then the matrix representation for L in terms of the basis $\{w_1, \ldots, w_n\}$ is given by:

$$\mathbf{BLB}^{-1}$$



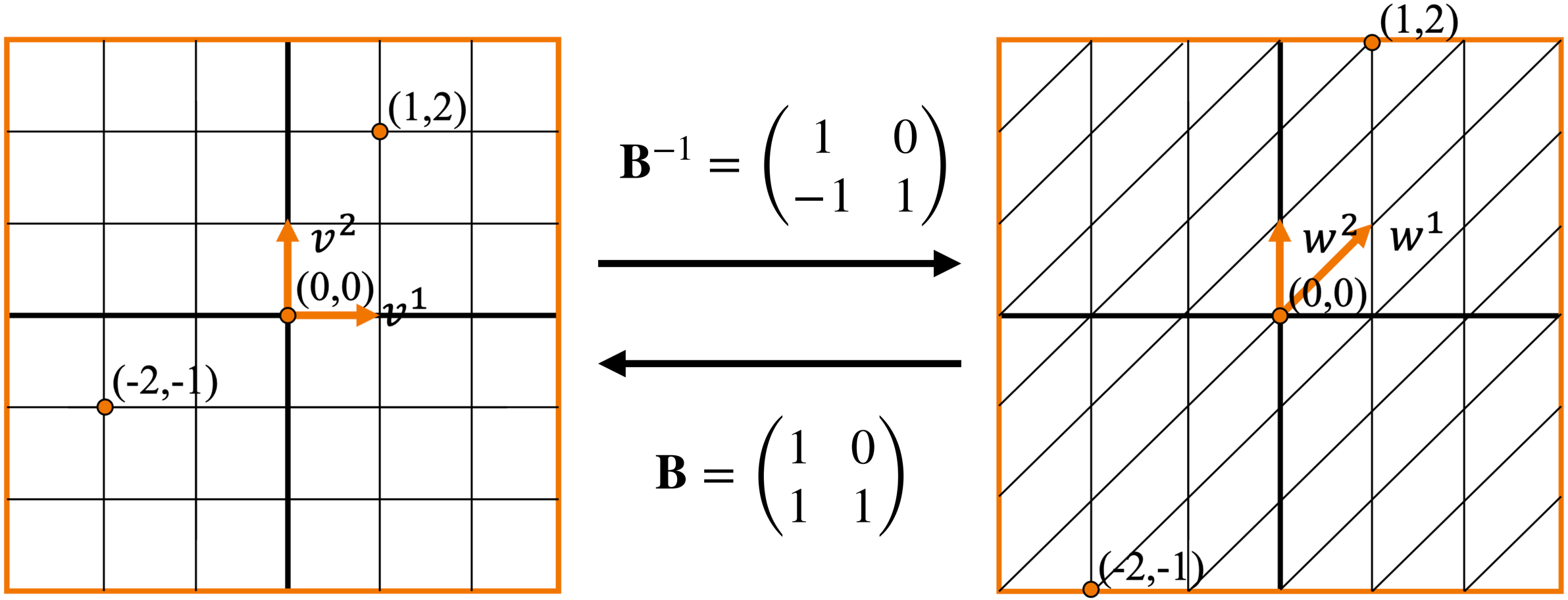
Adapted from Misha Kazhdan “FFTs in Graphics and Vision”



Consider the linear operator:
Expressed in v (left) basis

$$L = \begin{pmatrix} 2 & 0 \\ -1 & 1 \end{pmatrix}$$

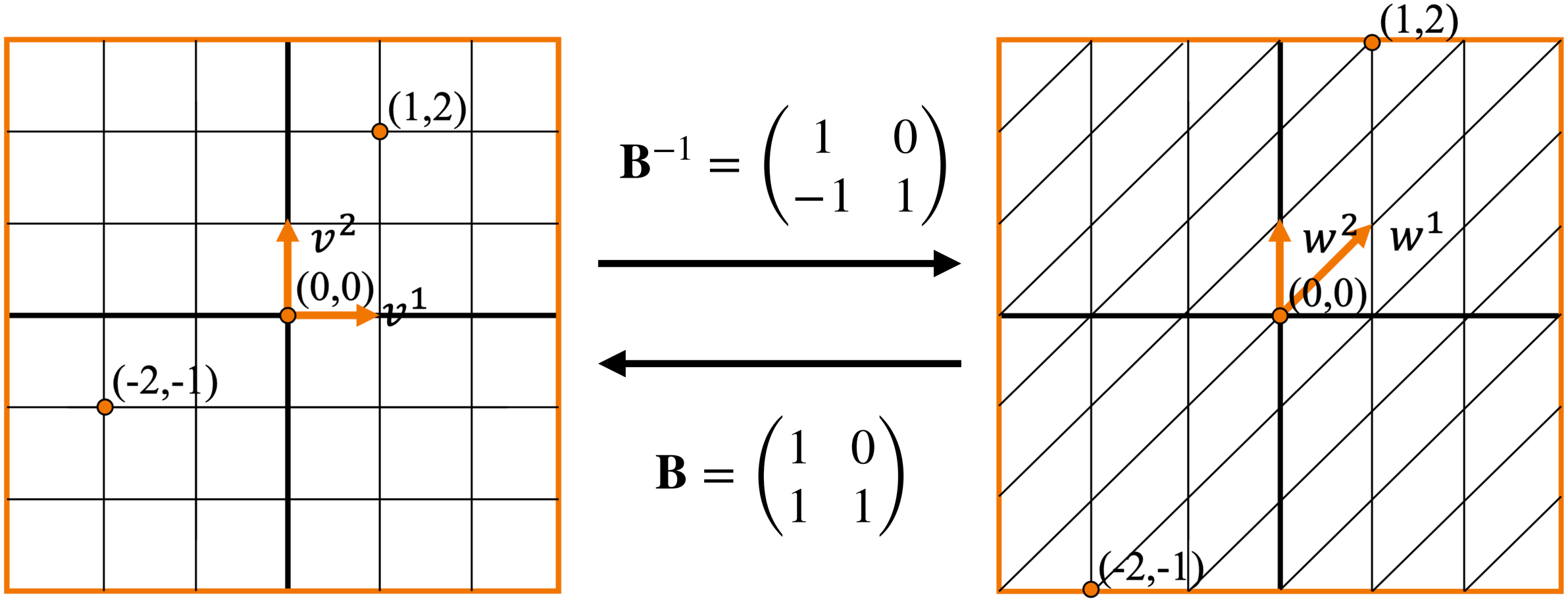
Adapted from Misha Kazhdan “FFTs in Graphics and Vision”



Transform to w (right) basis with B change of basis matrix

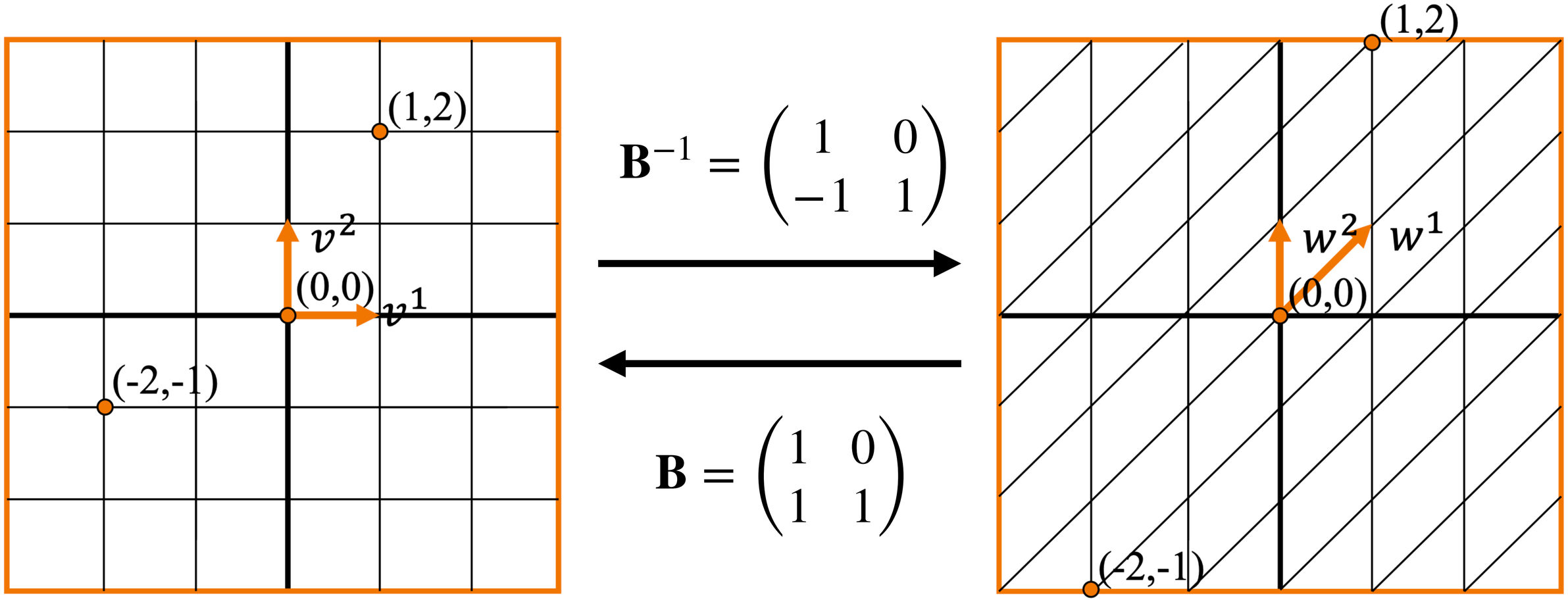
$$\mathbf{L} = \begin{pmatrix} 2 & 0 \\ -1 & 1 \end{pmatrix} \Rightarrow \mathbf{BLB}^{-1}$$

Adapted from Misha Kazhdan “FFTs in Graphics and Vision”



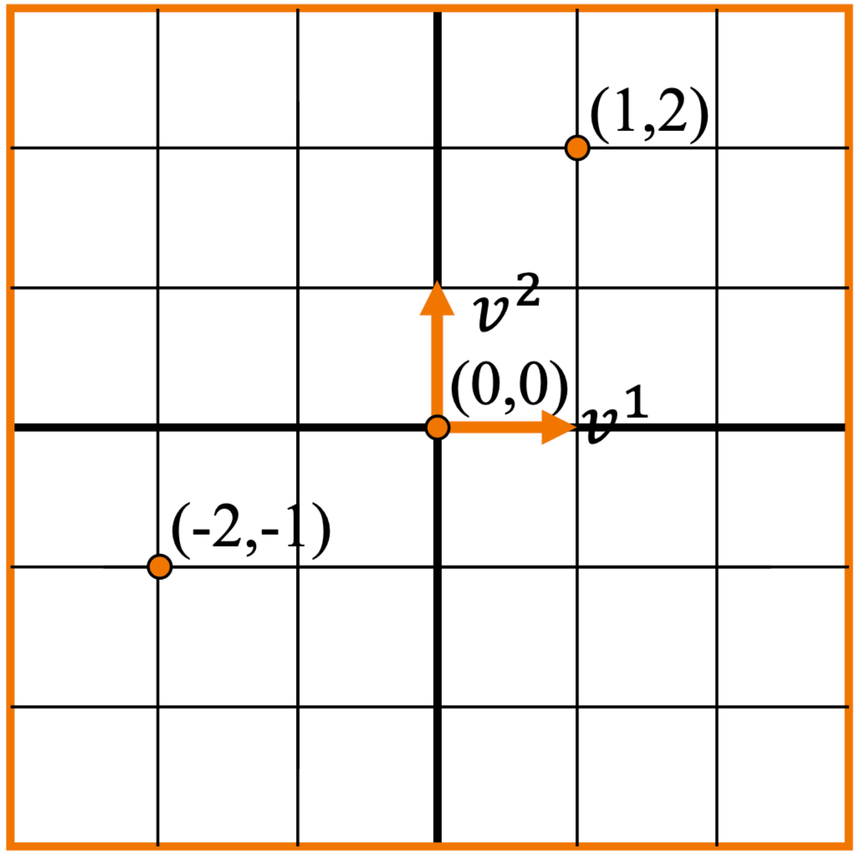
$$\mathbf{L} = \begin{pmatrix} 2 & 0 \\ -1 & 1 \end{pmatrix} \Rightarrow \mathbf{BLB}^{-1} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$$

Adapted from Misha Kazhdan “FFTs in Graphics and Vision”



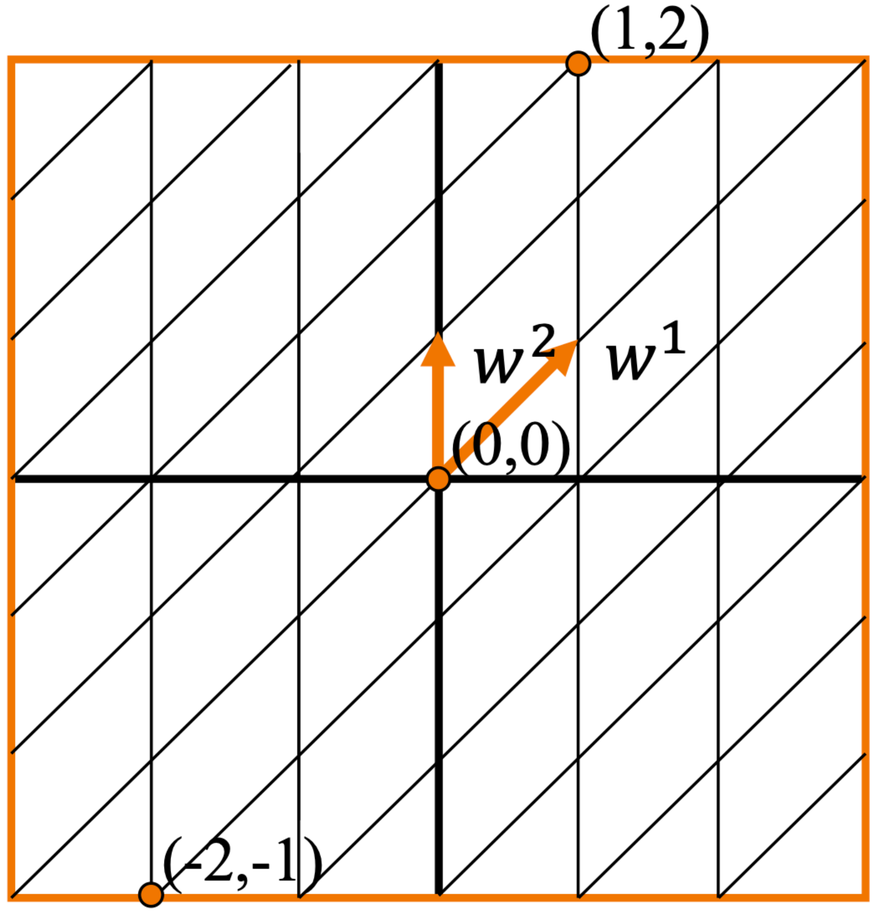
$$\mathbf{L} = \begin{pmatrix} 2 & 0 \\ -1 & 1 \end{pmatrix} \Rightarrow \mathbf{BLB}^{-1} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

Adapted from Misha Kazhdan “FFTs in Graphics and Vision”



$$\mathbf{B}^{-1} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$$

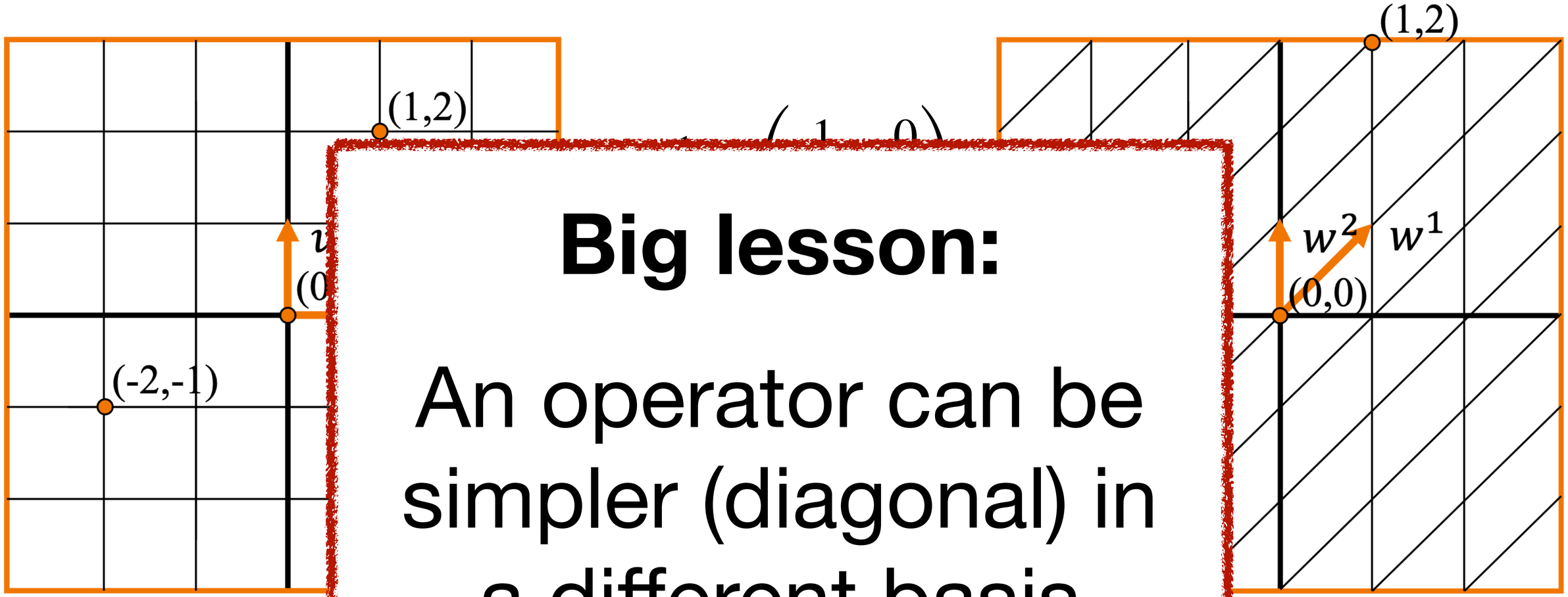
$$\mathbf{B} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$



Anisotropic scaling!!

$$\mathbf{L} = \begin{pmatrix} 2 & 0 \\ -1 & 1 \end{pmatrix} \Rightarrow \mathbf{BLB}^{-1} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} = \boxed{\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}}$$

Adapted from Misha Kazhdan “FFTs in Graphics and Vision”



Big lesson:

An operator can be simpler (diagonal) in a different basis

isotropic scaling!!

$$\mathbf{L} = \begin{pmatrix} 2 & 0 \\ -1 & 1 \end{pmatrix} \Rightarrow \mathbf{BLB}^{-1} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} = \boxed{\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}}$$

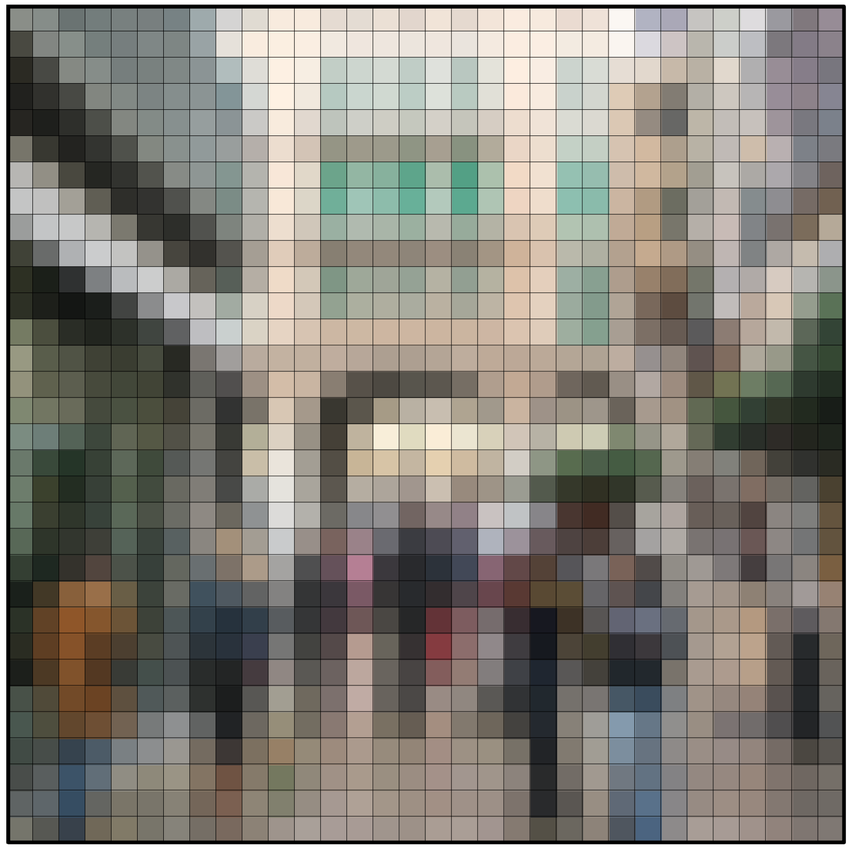
Adapted from Misha Kazhdan “FFTs in Graphics and Vision”

Back to Images as Vectors

Images can be represented as vectors of shape

$$\mathbf{f} \in \mathbb{R}^{H \times W \times ch}$$

Note that the image is, also, a *function* over pixel coordinates (just different perspective)



"Flatten"



Function Bases for Function Spaces

We can parameterize our image as a weighed sum of basis functions:

$$\mathbf{f} = \sum_k c_k \phi_k$$

We can *change function bases* via a simple multiplication with a matrix where the columns of that matrix are the basis functions ϕ_k:

$$\mathbf{f}' = \Phi \mathbf{f}$$

Function Bases for Function Spaces

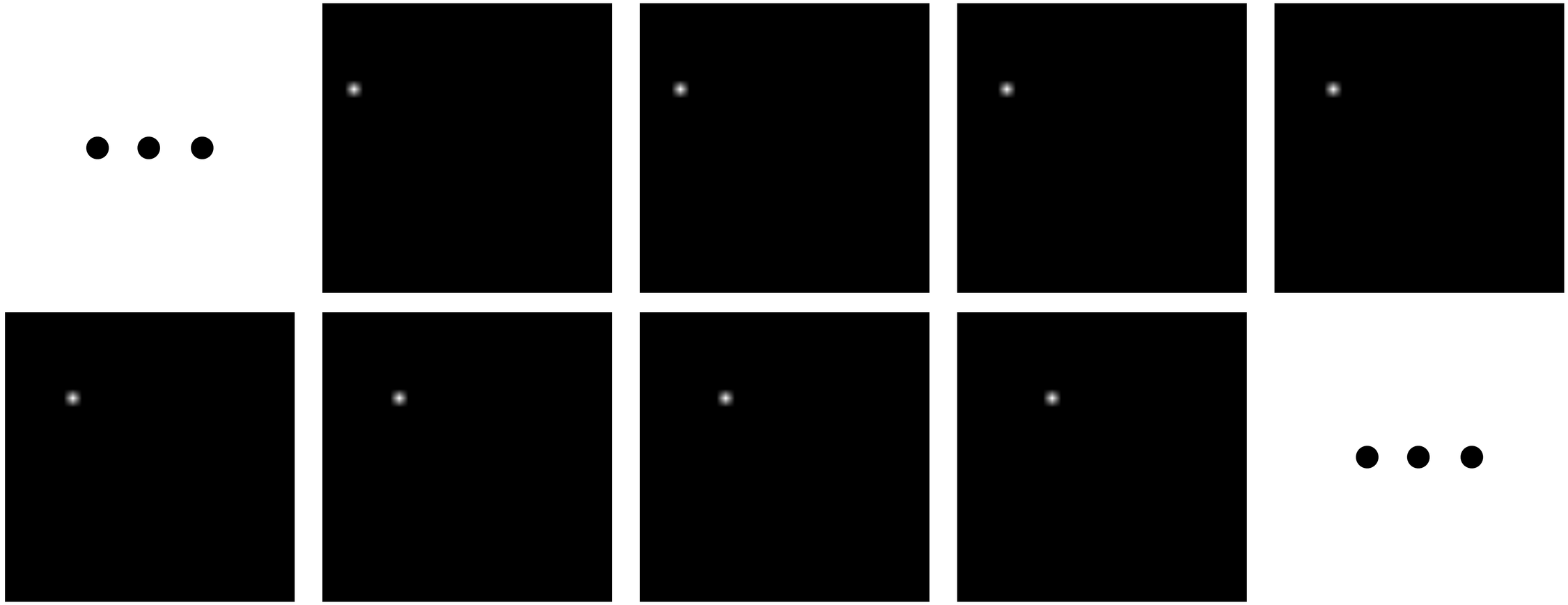
In numpy, PyTorch, etc., we usually store images in tensors

$$\mathbf{f} \in \mathbb{R}^{H \times W \times ch}$$

such that each element (coefficient)
is the intensity of that pixel for that color channel.

What's the basis?

Function Bases for Images - the pixel basis

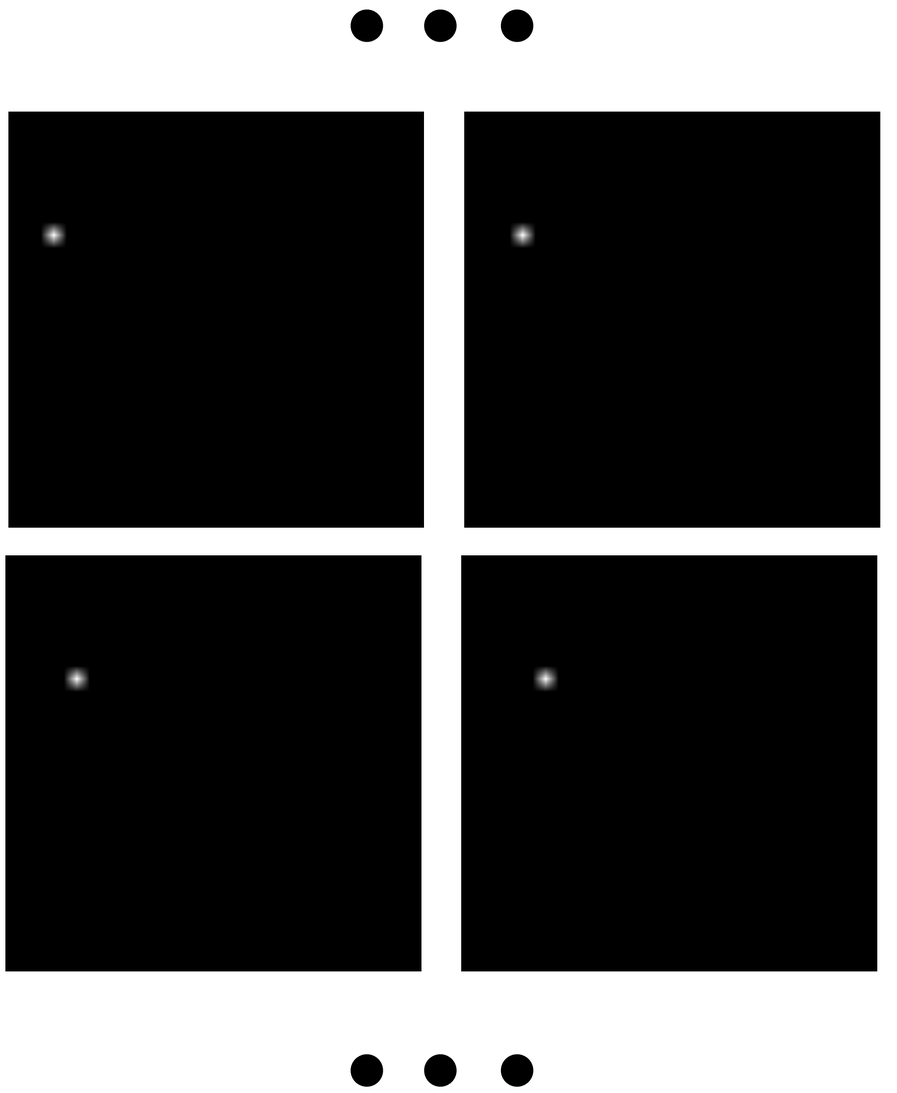


$$\phi_k[\mathbf{u}] = \begin{cases} 1, & \text{if } \mathbf{u}=\mathbf{u}_k, \\ 0, & \text{otherwise.} \end{cases}$$

Function Bases for Images - the pixel basis

This basis makes it easy to think about *locality.*

Each coefficient corresponds to a particular coordinate.



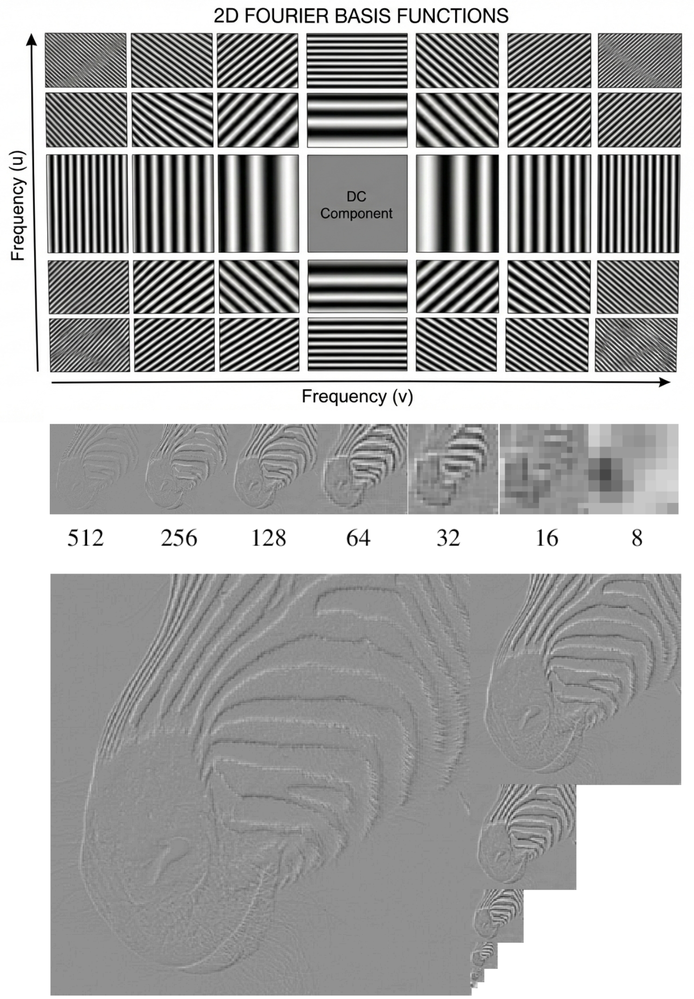
Motivation for other image bases/representations

Makes stuff simpler

Shift, blur, deblurring, sampling, aliasing

Make stuff less local, multiscale

Make stuff more related to human perception



First represent images with bases,
Later we will see overcomplete representation

(more than HxWxch coefficients)

Some vector/bases are special wrt some linear operators: Eigenvectors/Eigenbases

$$Lv = \lambda v$$

Some matrices (for example symmetric/Hermitian or circulant ones)
can be turned into (block) diagonal ones
using their eigenbasis

Why Eigenvectors/Eigenbases rock!

Many systems/operations are described by linear systems / matrices

- Blurry image : y = Ax
- Physical system, ODE : $\frac{dx}{dt} = Ax(t)$ or $\frac{d^2x}{dt^2} = Ax(t)$
- Markov chain : $p^{t+1} = Ap^t$

Matrices are complicated, often hard to understand behavior or solve problems (e.g. deblurring)

We can make it much easier if the matrix is diagonal (or block-diagonal)

- Better conceptual understanding
- Better numerical solutions
- Better algorithmic solutions

Why Eigenvectors/Eigenbases rock!

For inverse problems

- Blurry image : y = Ax

- **For iterated/integral operators**
  - Physical system, ODE : $\frac{dx}{dt} = Ax(t)$ $\frac{d^2x}{dt^2} = Ax(t)$
  - Markov chain : $p^{t+1} = Ap^t$

Why Eigenvectors/Eigenbases rock!

For inverse problems

- Blurry image : y = Ax

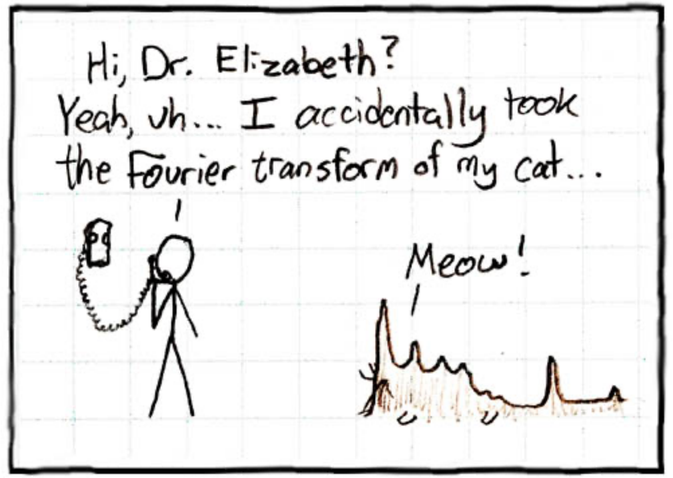
Multiplication on the diagonal are inverted by a division
Numerical conditioning depends on how small the eigenvalues are (worst case is 0)

- **For iterated/integral operators**
  - Physical system, ODE : $\frac{dx}{dt} = Ax(t)$ $\frac{d^2x}{dt^2} = Ax(t)$
  - Markov chain : $p^{t+1} = Ap^t$

Iterated diagonal multiplications become exponentials and/or rotations/sine waves
(Complex-valued) eigenvalues give us exponential base and rotation speed

Motivation for The Fourier Basis

The Fourier basis diagonalizes convolutions and shifts (translations)



https://m.xkcd.com/26/

Motivation for The Fourier Basis

The Fourier basis diagonalizes convolutions and shifts (translations)

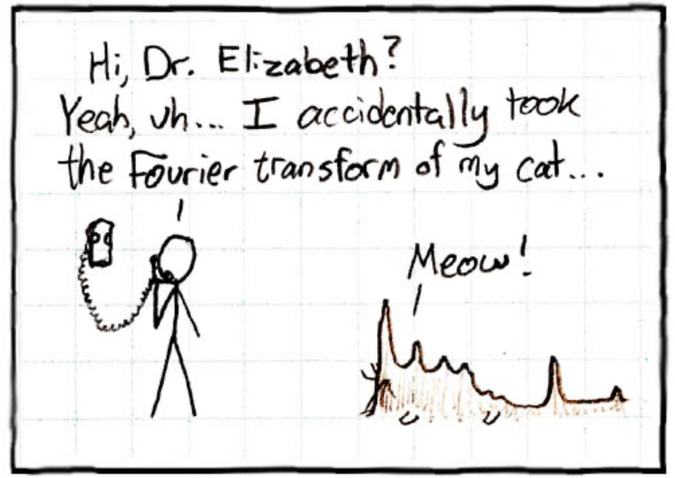
Makes lots of things simpler:

- Convolutions
- Inverse problems
- Sampling and aliasing

Naturally captures that the world/images are shift invariant

Not unrelatedly, it also relates to human perception

Will show up everywhere, even for text !
(e.g. positional encoding)



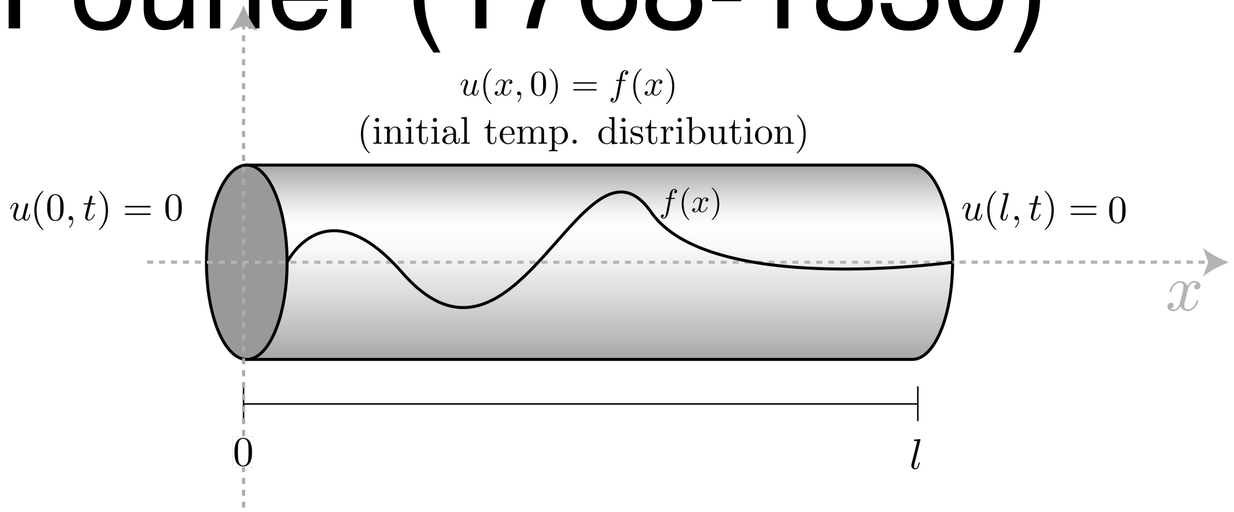
https://m.xkcd.com/26/

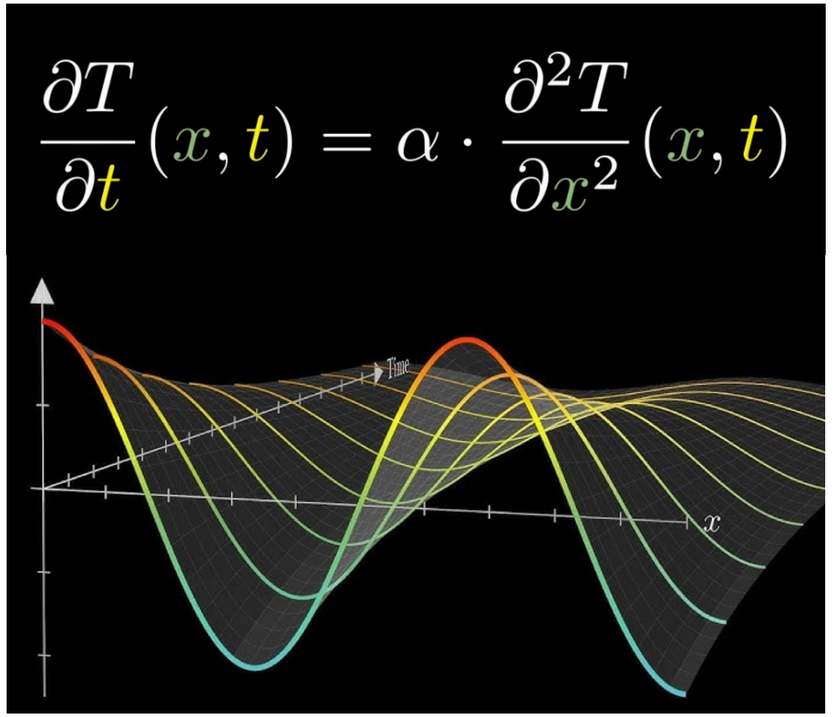
Jean-Baptiste Joseph Fourier (1768-1830)

Solution to heat diffusion

- Case of 1D rod with initial temperature distribution
- Simpler with sine waves
- Exponential decay







https://www.youtube.com/watch?v=ToIXSwZ1pJU

Fourier

Solution to heat diffusion

Simpler with sine waves

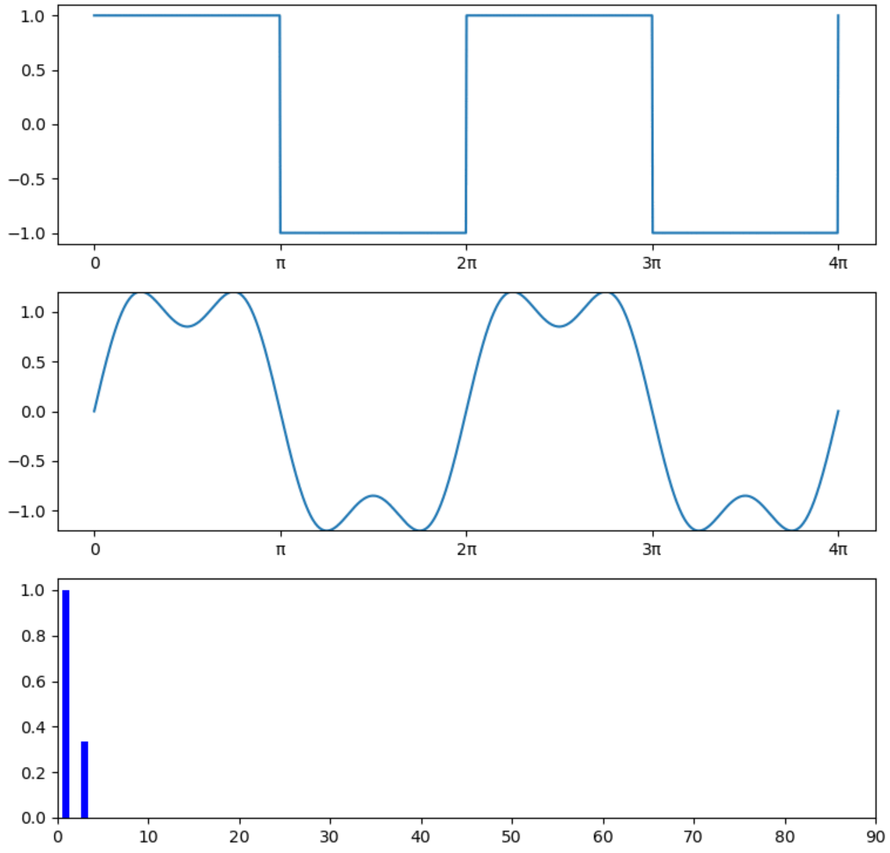
Any(**) univariate function can be expressed as a weighted sum of sinusoids of different frequencies (1807)



Jean-Baptiste Joseph Fourier (1768-1830)

Example: series for a square wave

$$\sum_{k=1,3,5,\ldots}^{\infty} \frac{1}{k}\sin(kt)$$



Slide credit: S. Lazebnik, C. Rupprecht

Fourier analysis

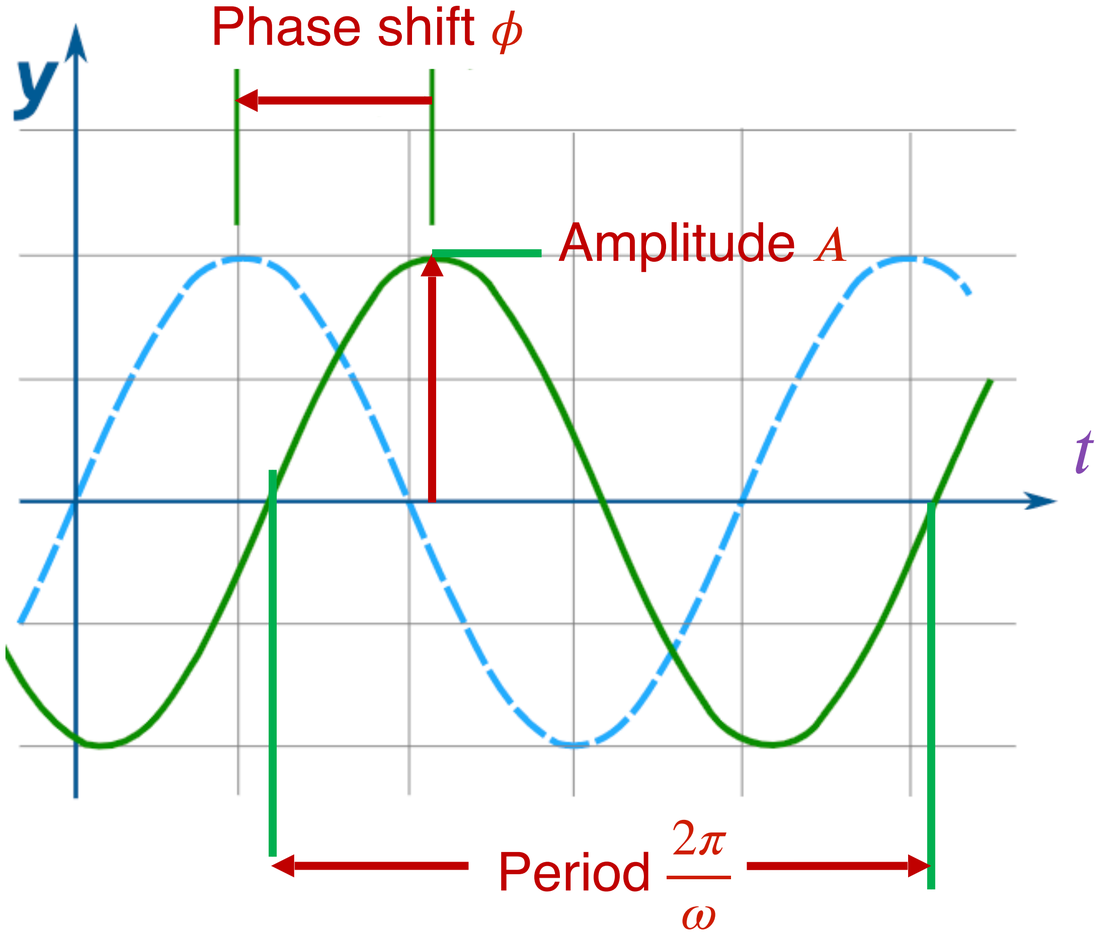
Our building block:

Frequency

$$A\sin(\omega t+\phi)$$

Amplitude

Phase



Add enough of these to get any signal you want!

Slide credit: S. Lazebnik

Are these sines a function basis…?

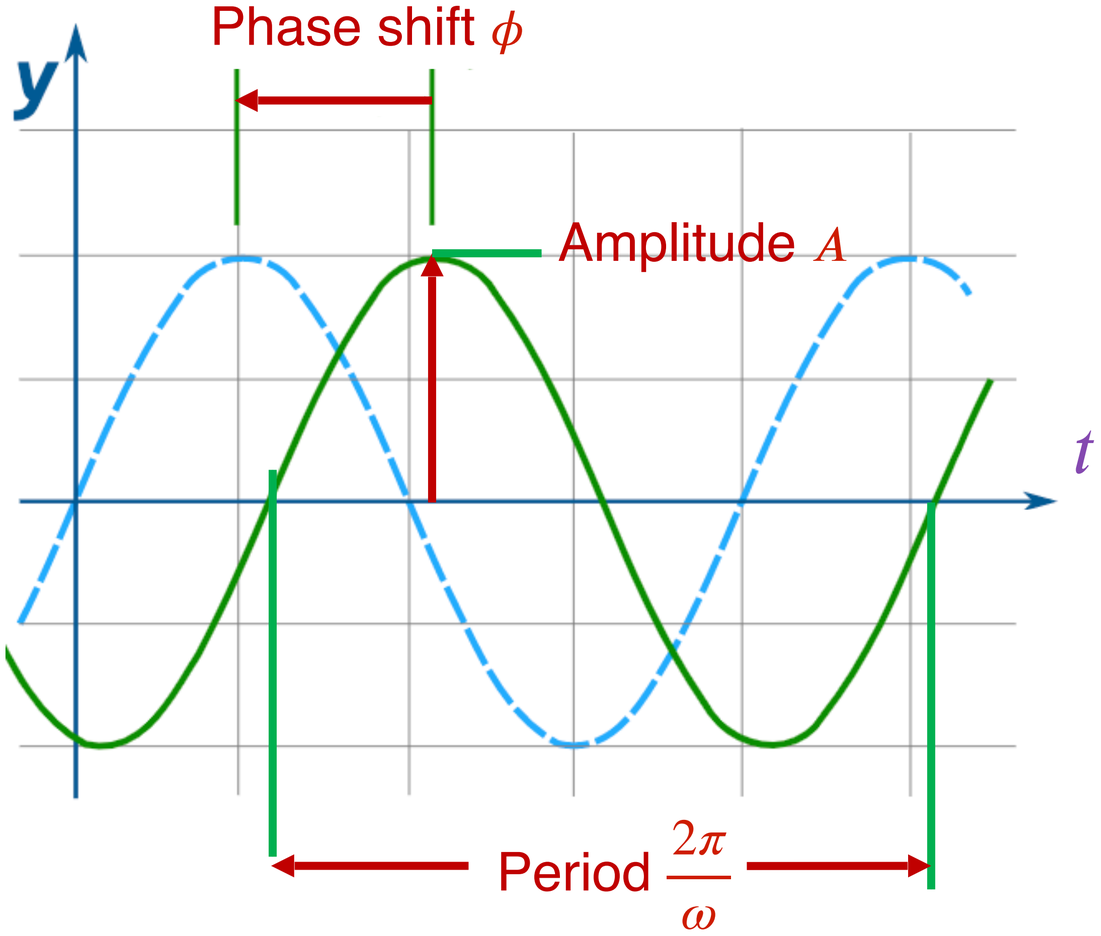
Our building block:

Frequency

$$A\sin(\omega t+\phi)$$

Amplitude

Phase



Add enough of these to get any signal you want!

Slide credit: S. Lazebnik

Problem: Every signal requires sines with different phases and phase is non linear :(

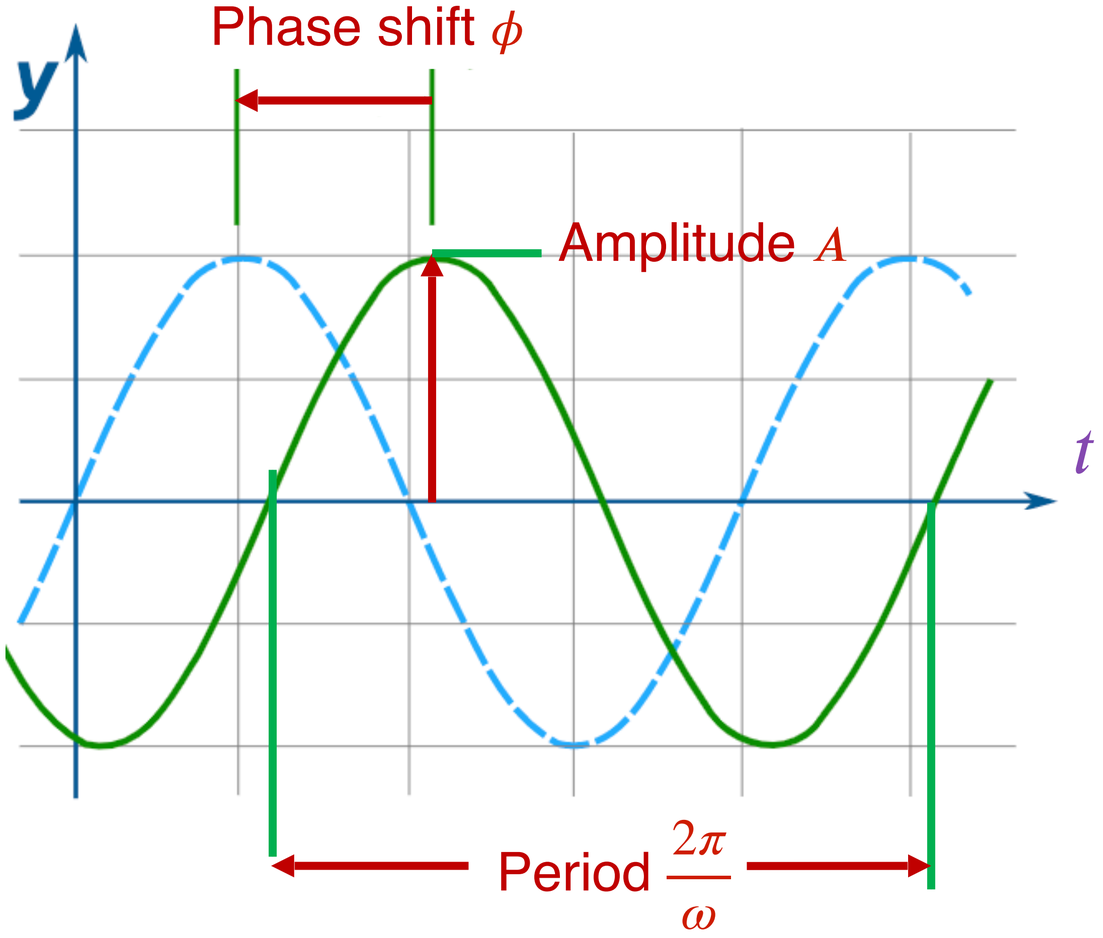
Our building block:

Frequency

$$A\sin(\omega t+\phi)$$

Amplitude

Phase

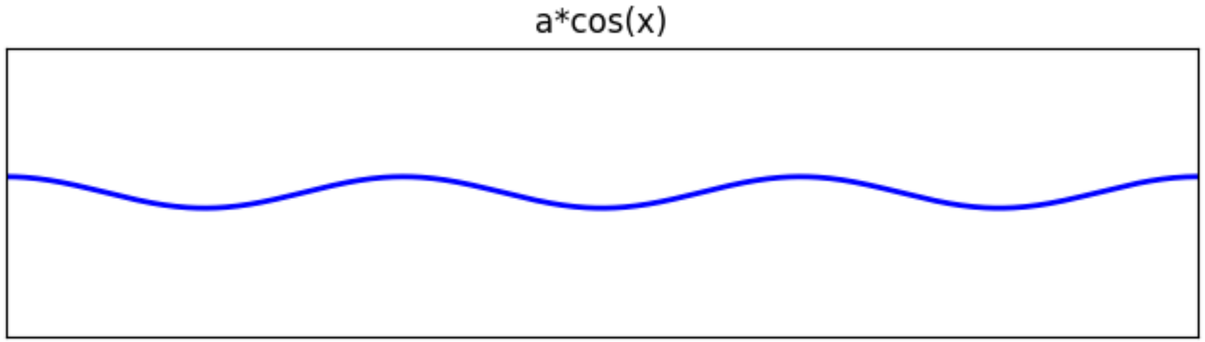


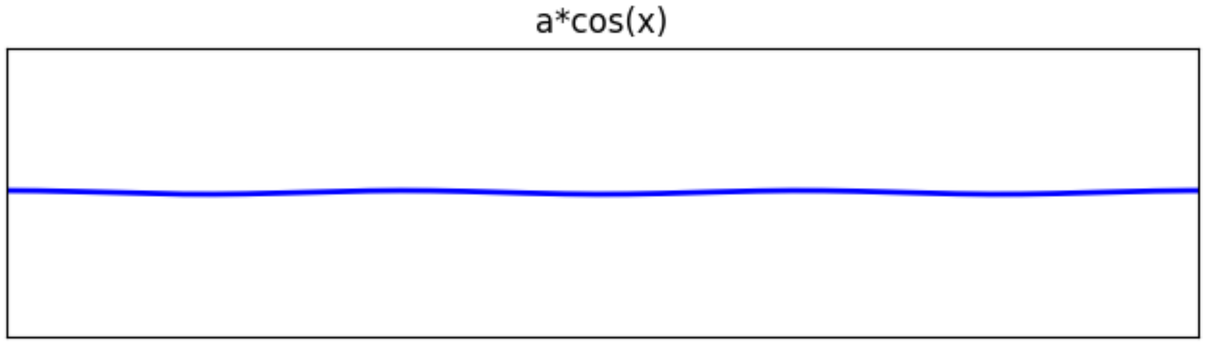
Add enough of these to get any signal you want!

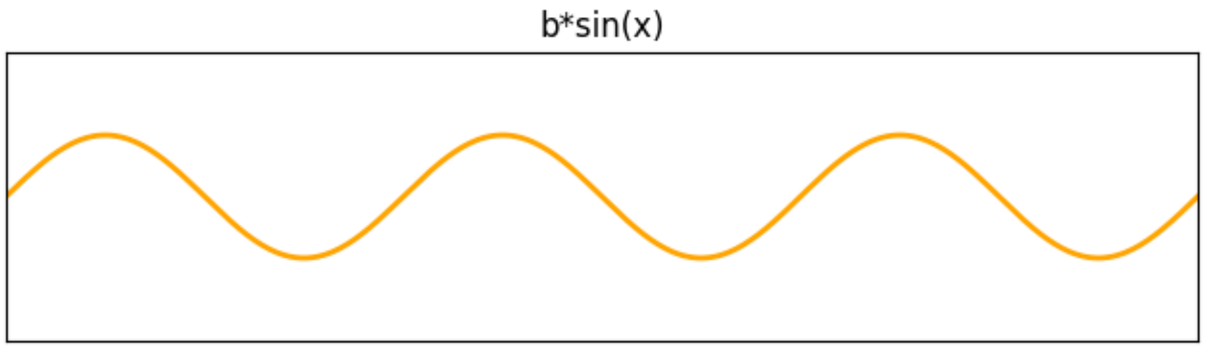
Slide credit: S. Lazebnik

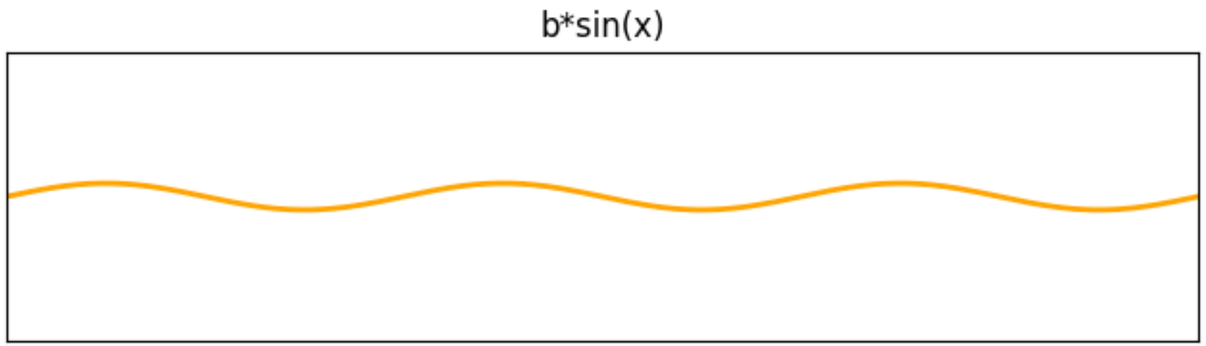
The Translation-Steerability of the Sine

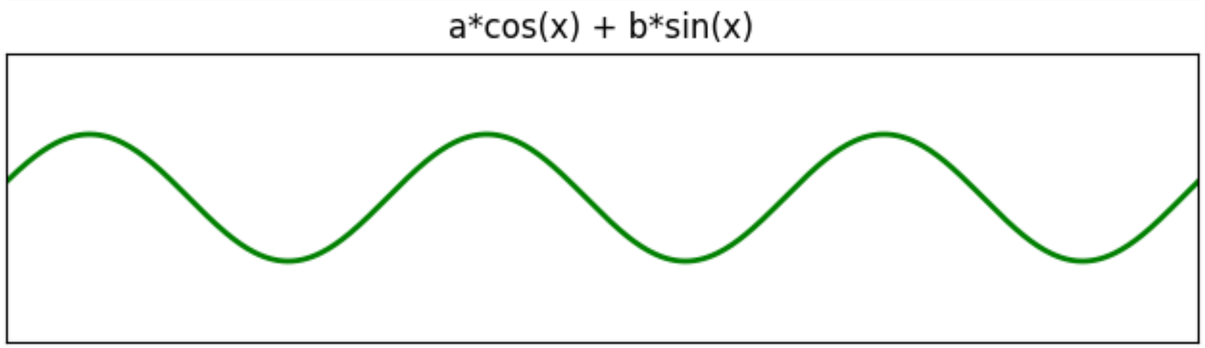
Consider different weighted sums of a sine and a cosine:

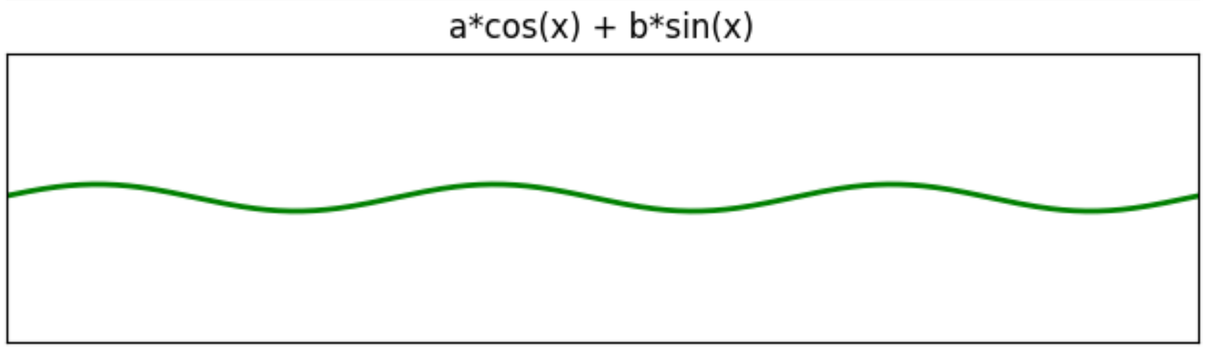






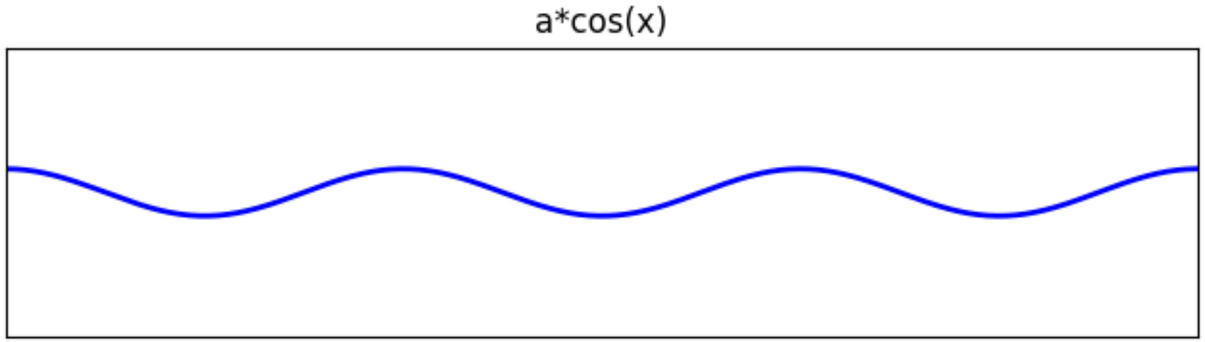


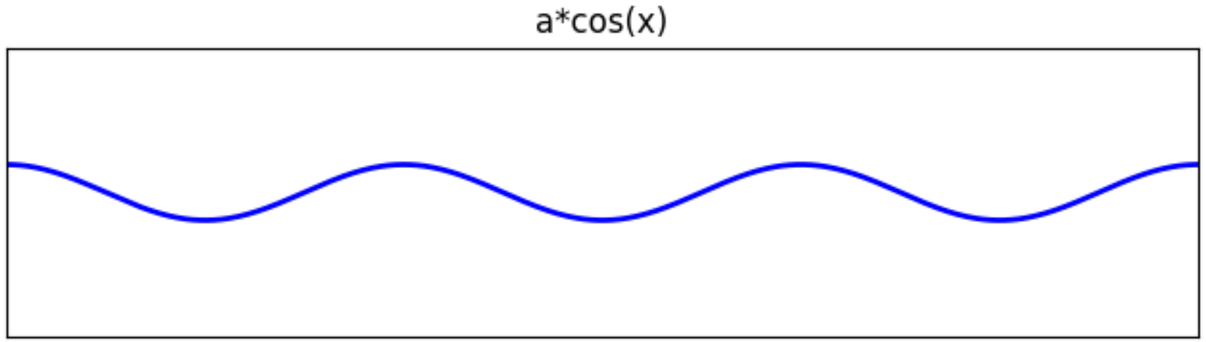


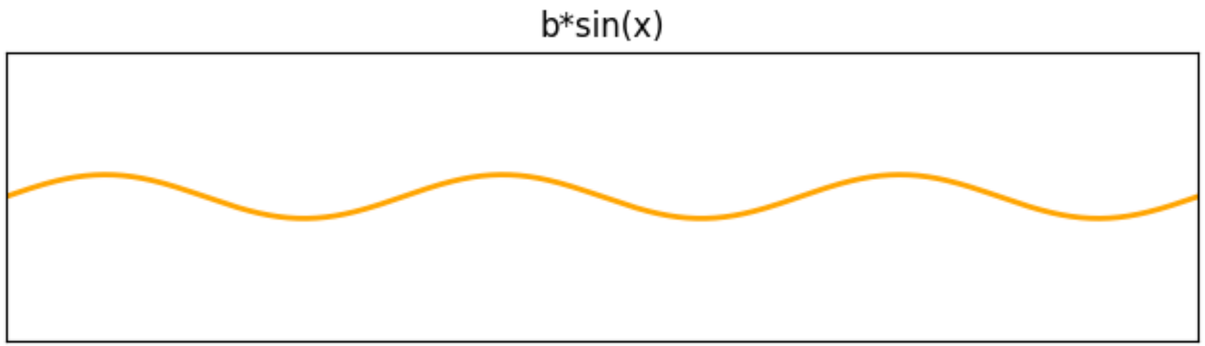


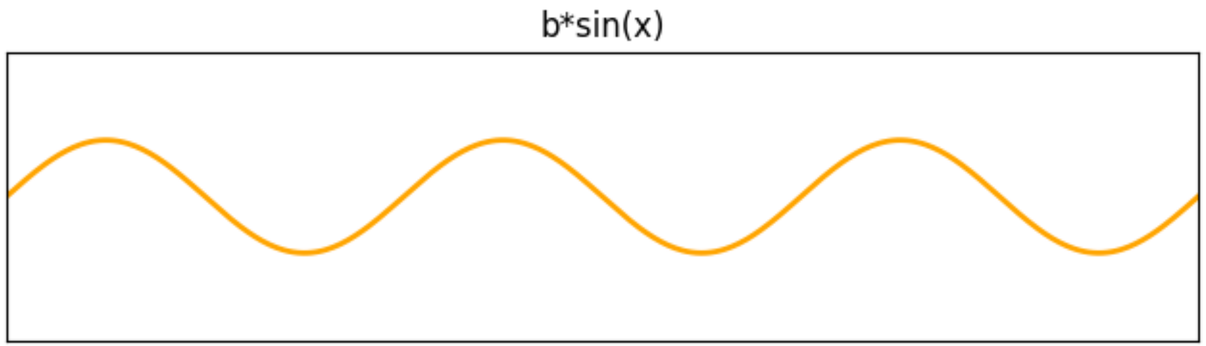
The Translation-Steerability of the Sine

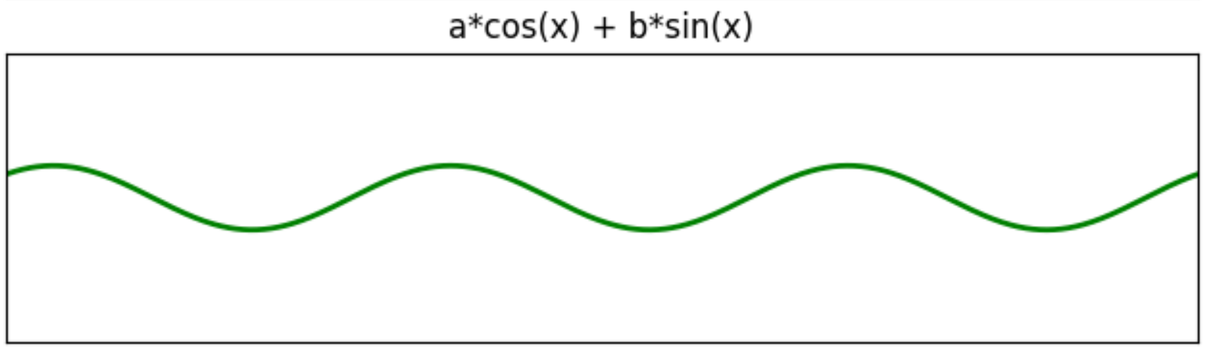
Consider different weighted sums of a sine and a cosine:

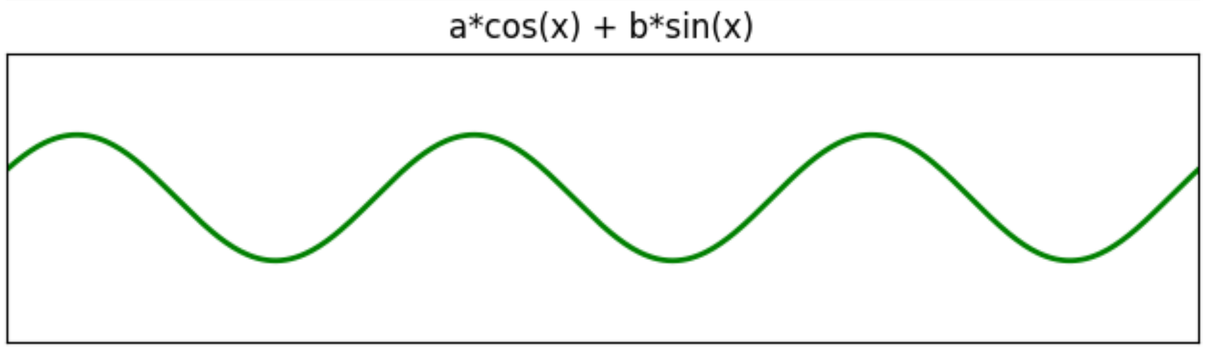


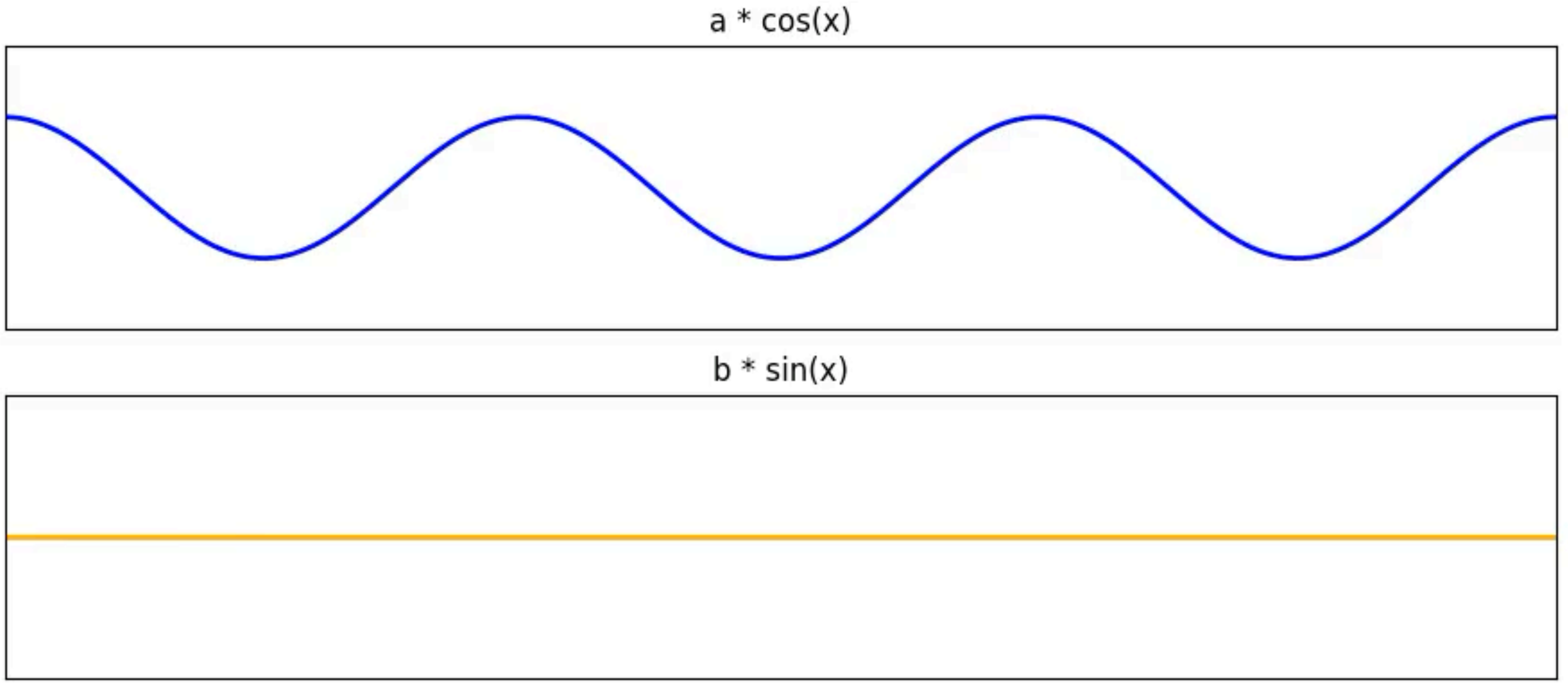




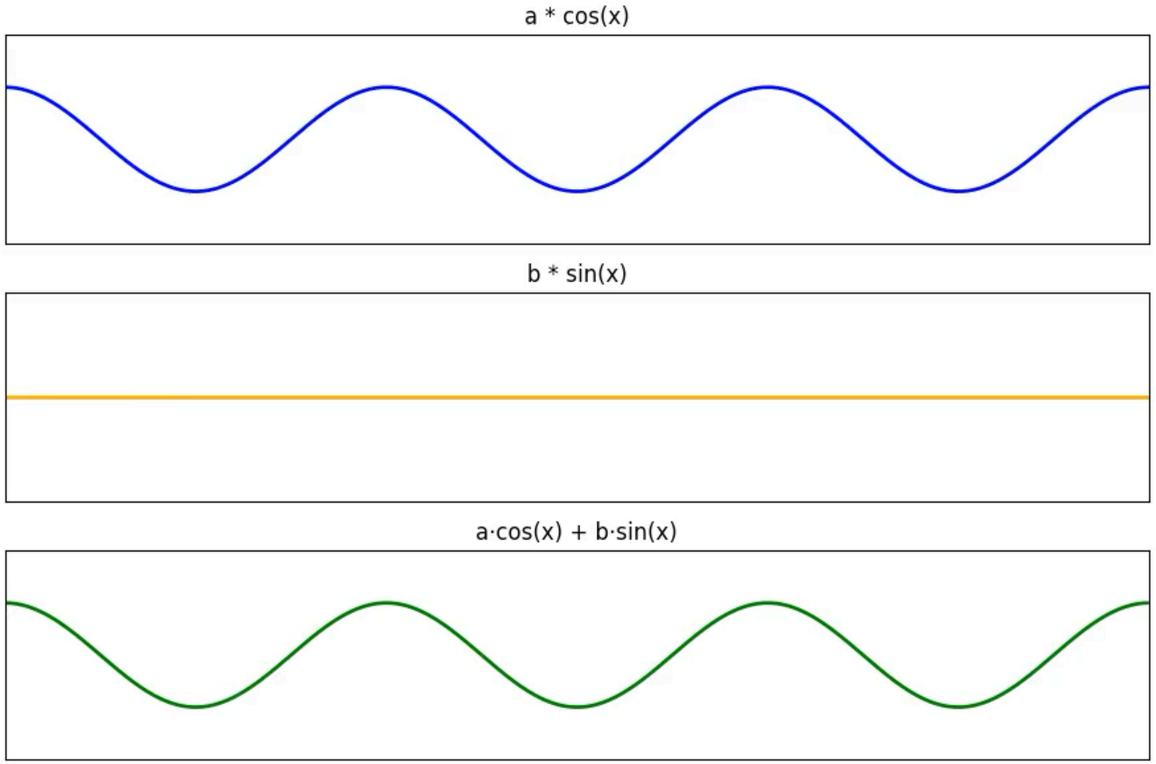


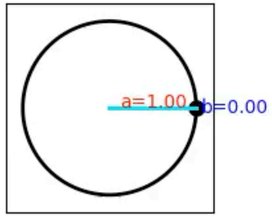





a * cos(x)
b * sin(x)

Using Sines and Cosines as a Function Basis



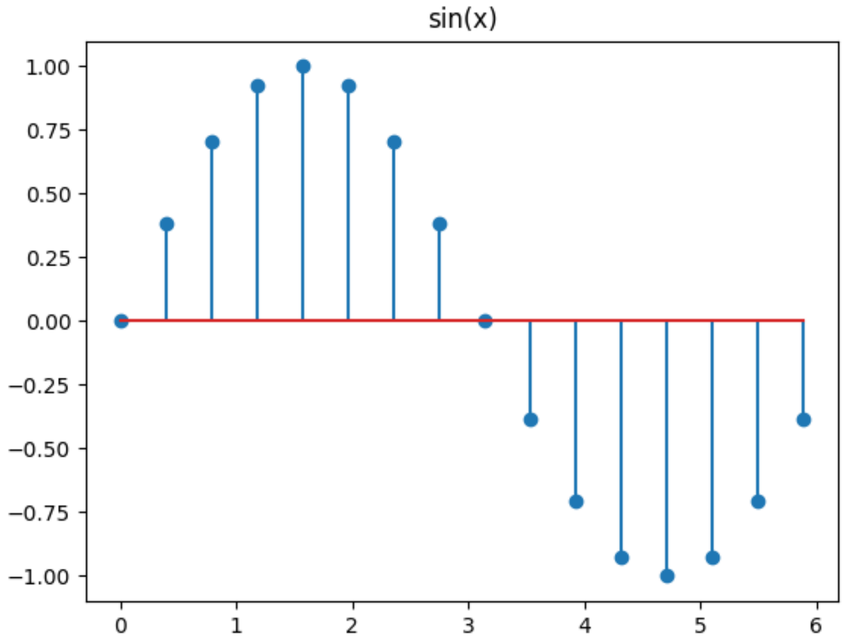


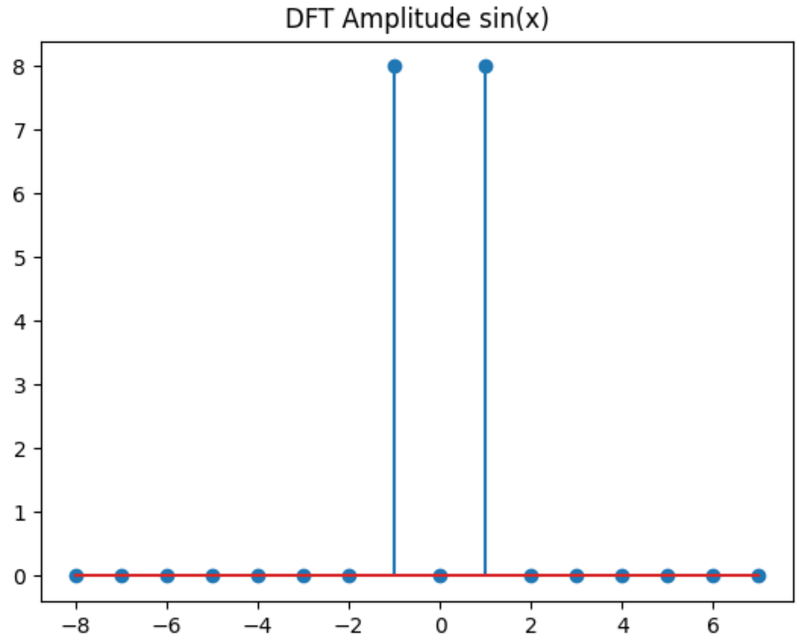
- Sine, cosine of a given frequency form a basis for the space of *all sines with that frequency,* allowing to arbitrarily vary amplitude and phase!
- Given a weighted sum of sines of arbitrary phase and frequency, we can form any** function.

The Fourier Basis: for every frequency, a pair of a sine and a cosine!

Discrete Fourier Transform

When we only have N discrete (evenly spaced) samples from a signal, we also only need a discrete set of N basis functions.





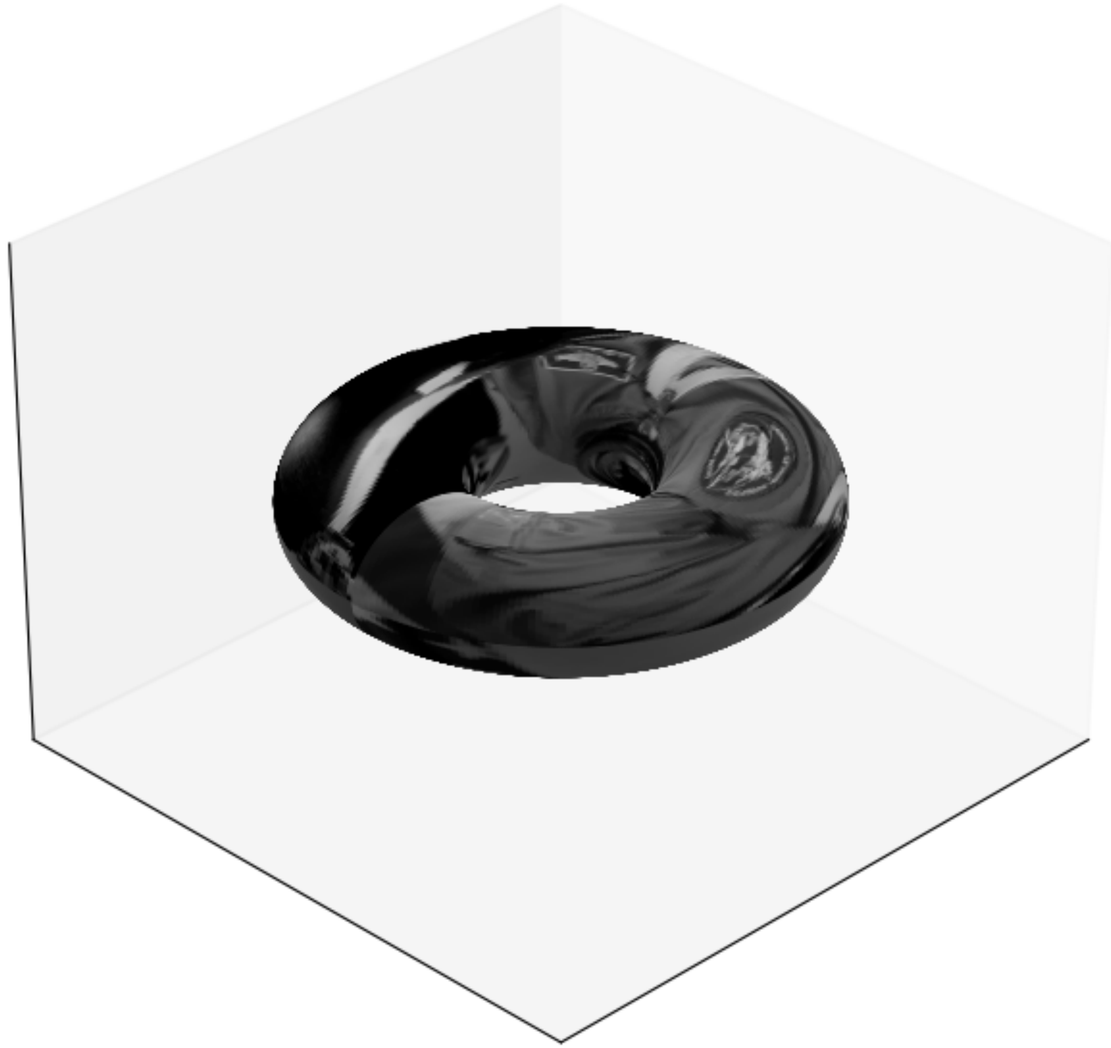
Slide credit: Christian Rupprecht, Oxford

This assumes Circular Boundary Conditions!



The image isn't a function on $\mathbb{R}^2$, it is a function defined on the surface of a topological 3D Torus!

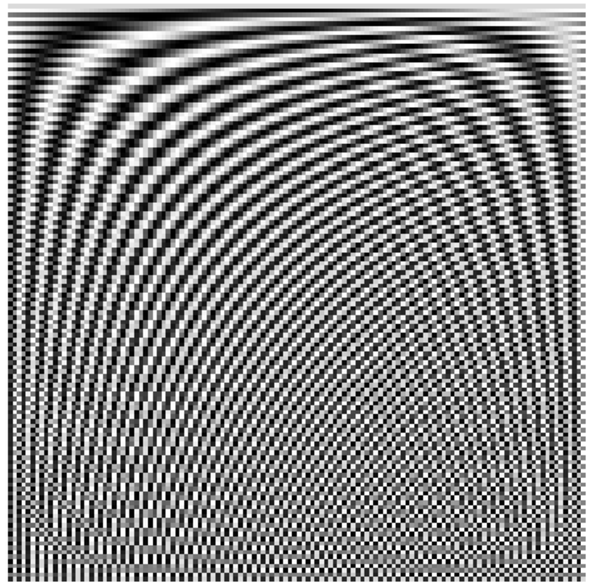




Discrete Fourier Transform

$$a_{2k} = \frac{2}{N}\sum_{n=0}^{N-1} f(n)\cos\left(\frac{2\pi k n}{N}\right), \qquad a_{2k+1} = \frac{2}{N}\sum_{n=0}^{N-1} f(n)\sin\left(\frac{2\pi k n}{N}\right),$$

- For each n we compute the dot-product between the discrete signal x and the cosine and sine.
- This is just a matrix-vector multiplication!

$\mathbf{F} =$ 

$\mathbf{f}$ 

Slide credit: Christian Rupprecht, Oxford

Wait - I thought the Fourier Transform was *complex?*

- The Fourier Transform is usually written as follows:

$$F(k) = \sum_{n=0}^{N-1} f(n) \exp\left(-i \frac{2\pi}{N} kn\right)$$

- What's the relationship?

The complex exponential

- Euler's Identity: $e^{i\pi} + 1 = 0$
- Euler's Formula: $e^{i\phi} = \cos\phi + i\sin\phi$
- Identity: $e^{i\pi} = \cos\pi + i\sin\pi = -1 + 0$

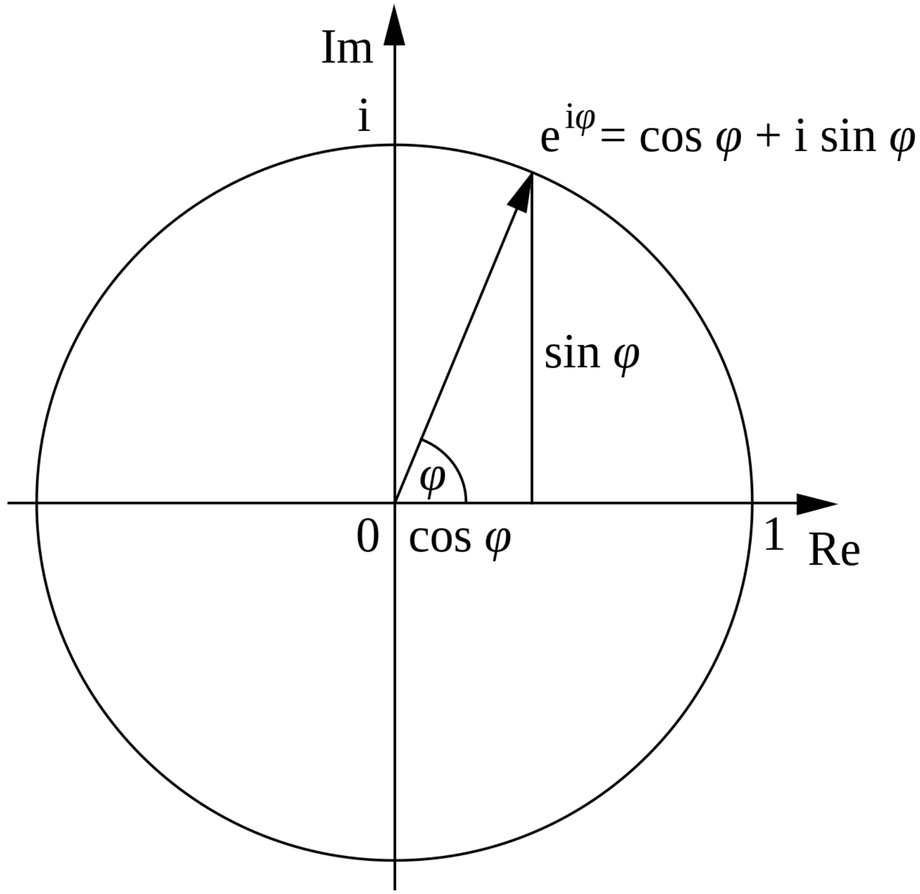


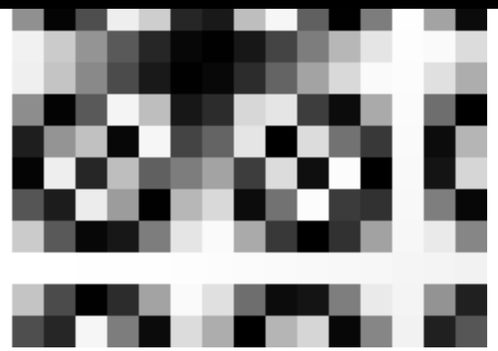
Image source

Wait - I thought the Fourier Transform was *complex?*

$$F(k) = \sum_{n=0}^{N-1} f(n) \exp\left(-i \frac{2\pi}{N} kn\right)$$

The complex-valued Fourier Transform has additional structure & symmetry over the real-valued version.
But the intuition is valuable and valid!

$$\mathbf{F} = \left(\quad + i \quad \right) \mathbf{f}$$





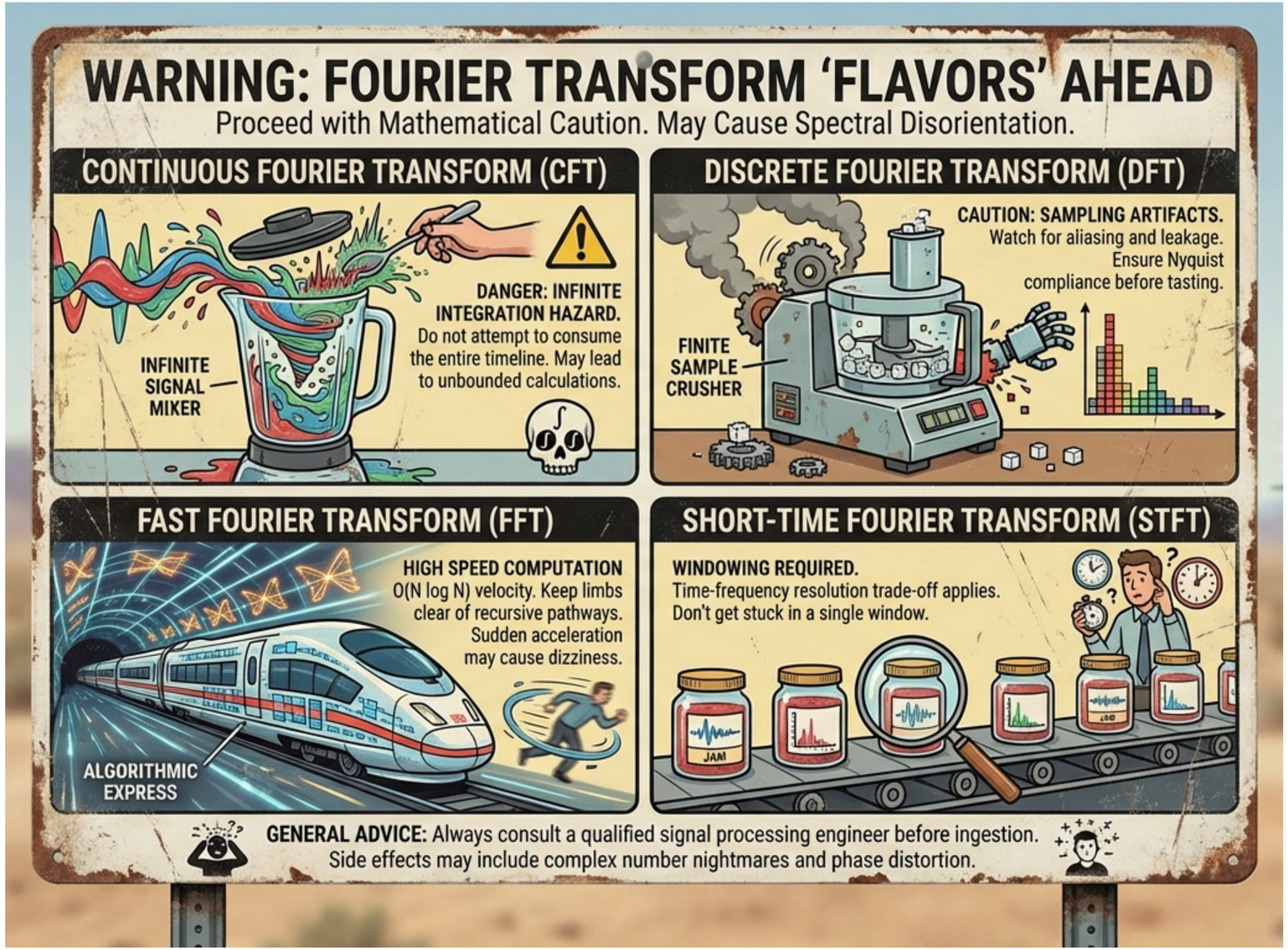


Slide credit: Christian Rupprecht, Oxford

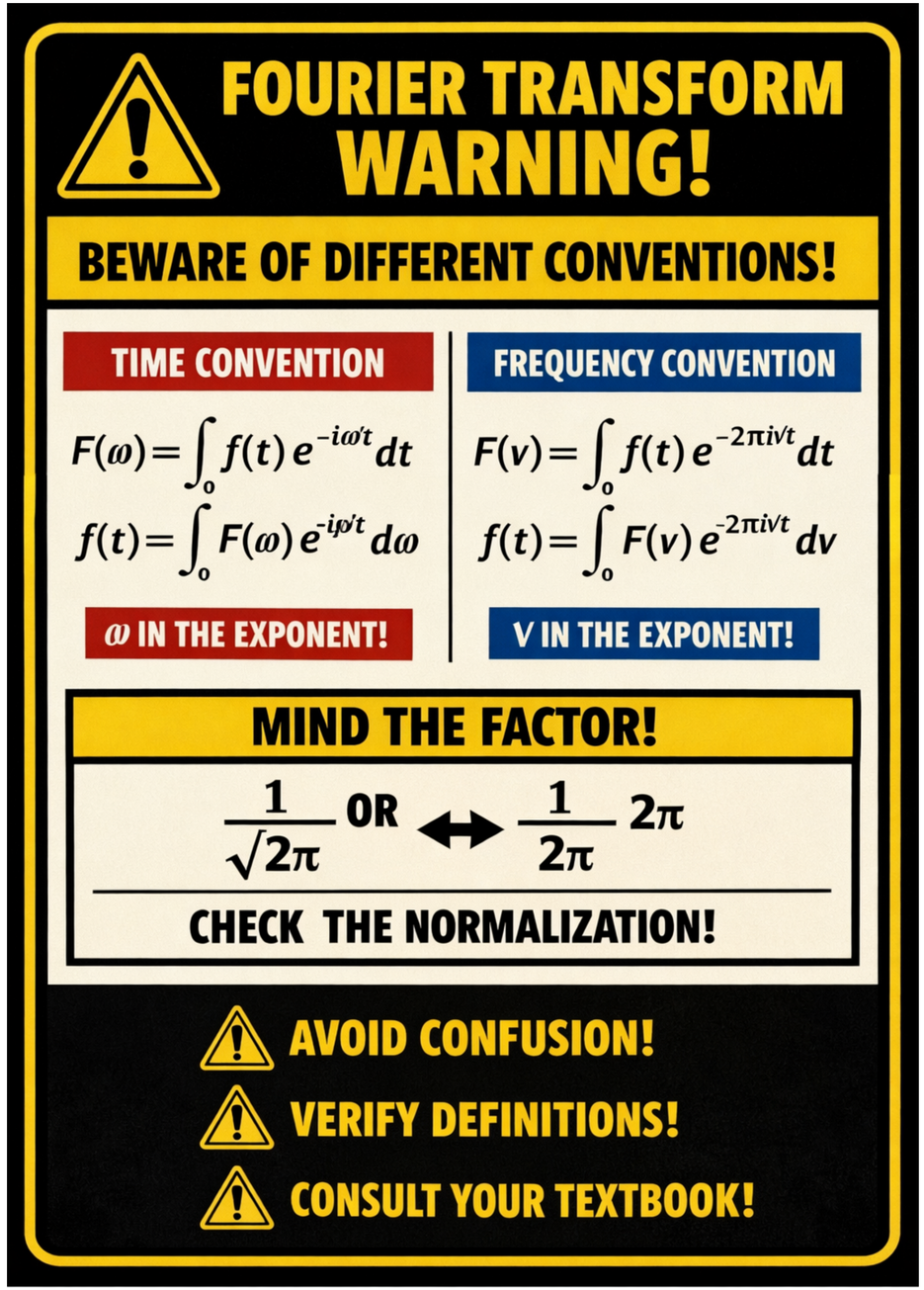
Warning

- Many flavors of Fourier+inverse transform with major/minor variations
  - Discrete vs continuous
  - Real vs complex
  - Circle/torus vs full line, boundary handling, windowing
  - Order of frequencies
  - Frequency units and normalization $e^{-i\omega x}$, $e^{-2\pi i f x}$ $1/\sqrt{2\pi}$
    - Sign (time reversal)
  - More fancy math (distributions like Dirac, etc.)

Warning

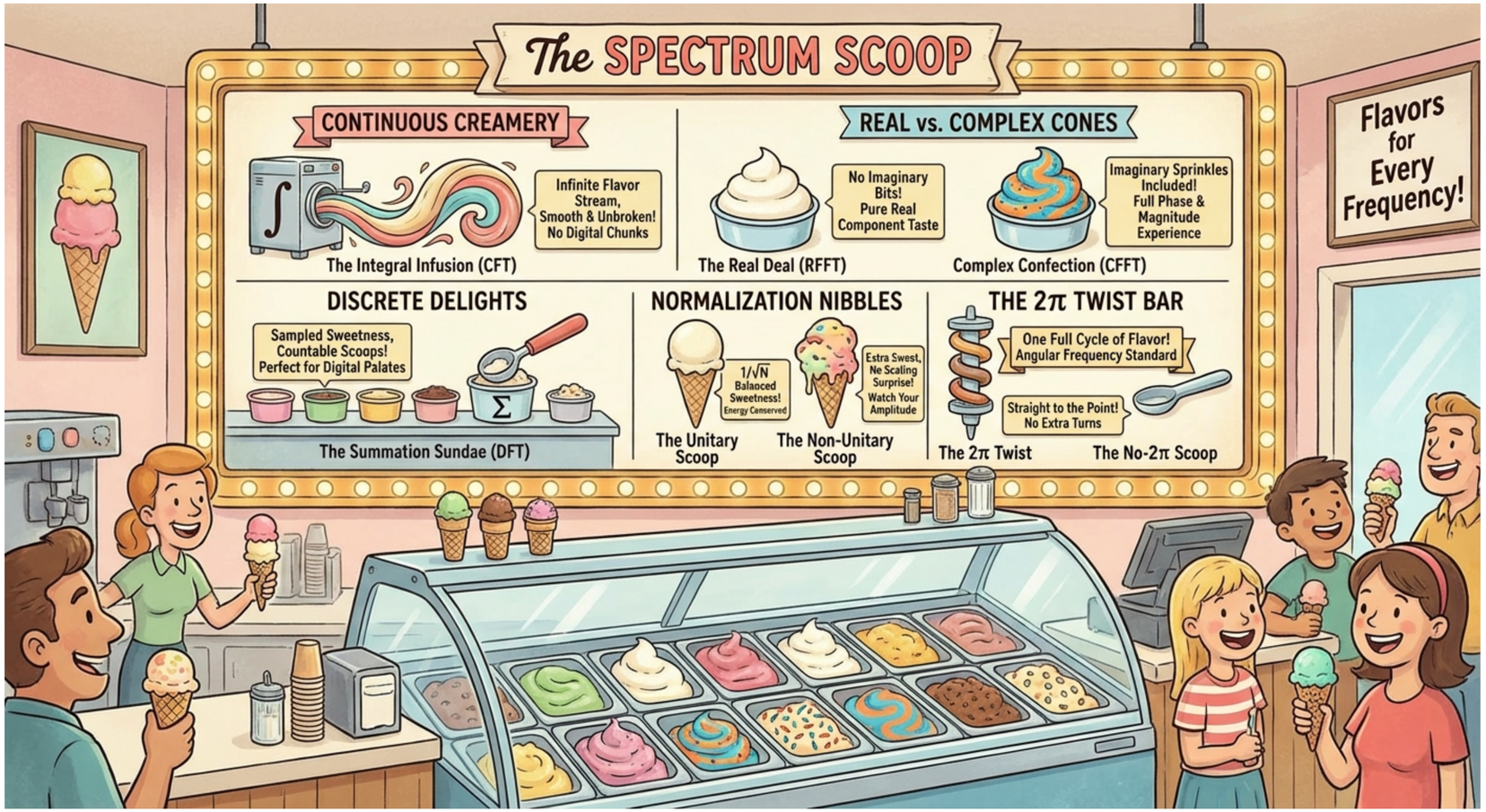


Nano Banana



ChatGPT

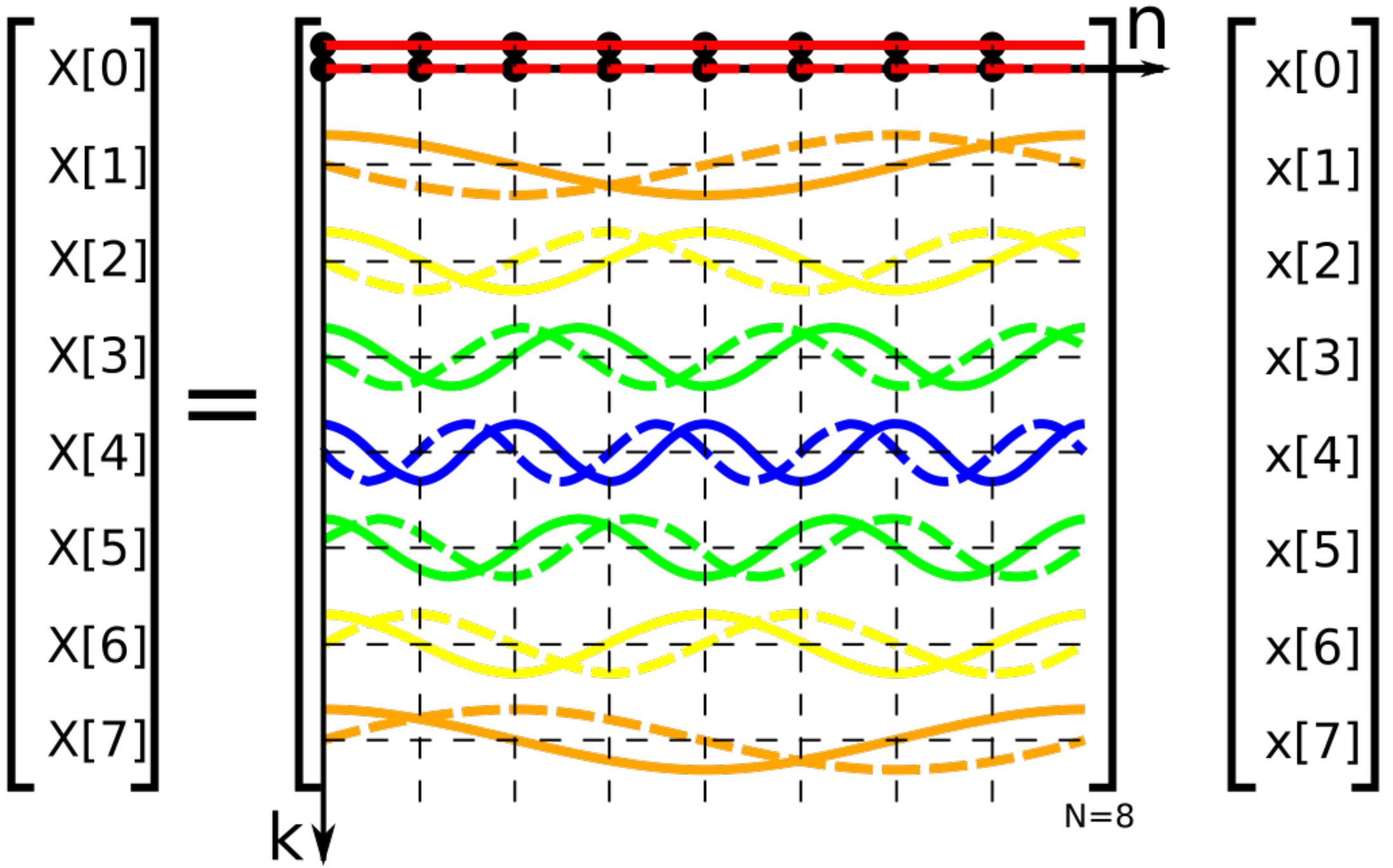
Warning



Nano Banana

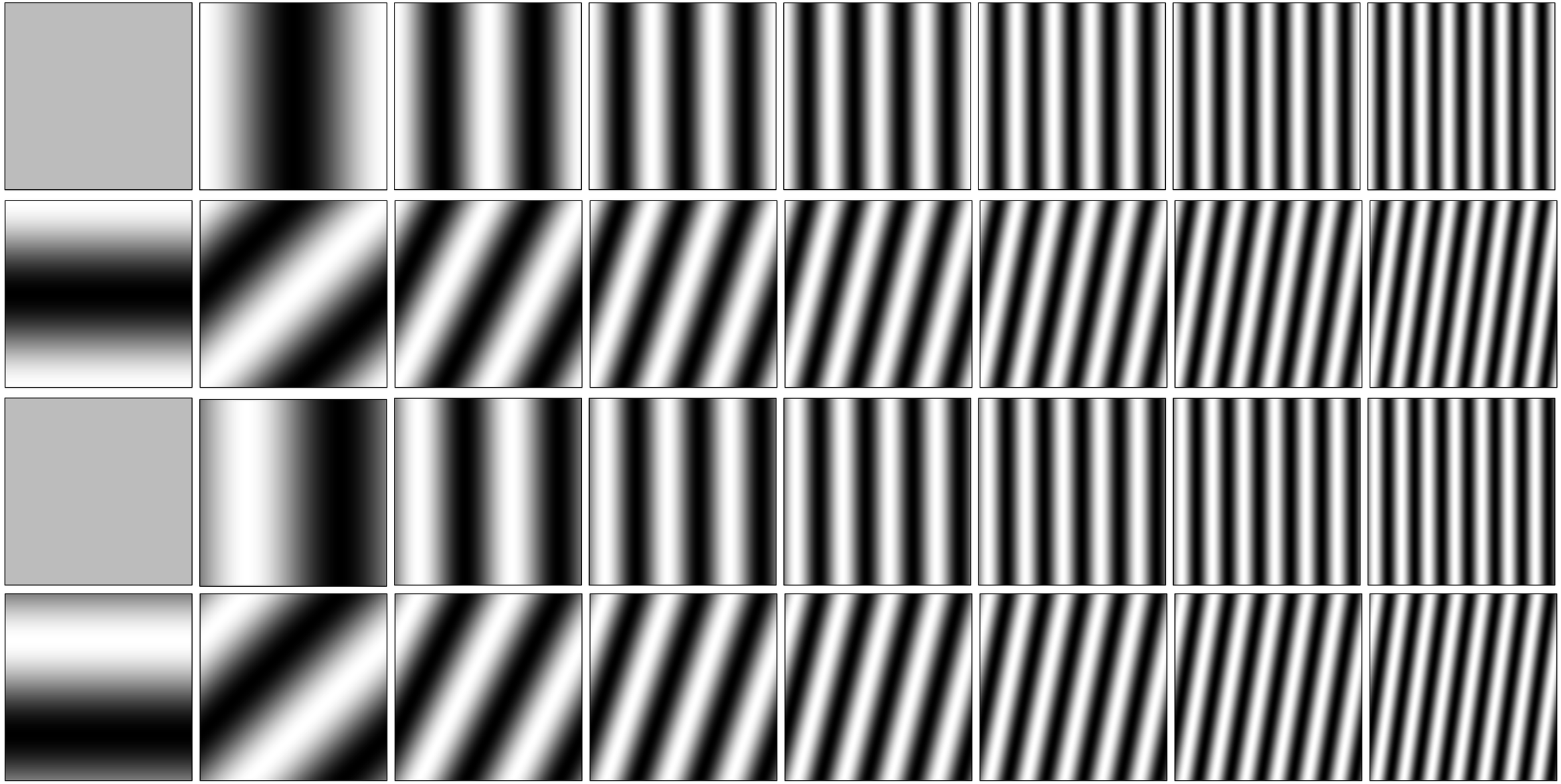


ChatGPT



Source: https://en.wikipedia.org/wiki/DFT_matrix

Same in 2D!



Careful

- You can apply the 2D Fourier Transform onto the 1D vector representation of images
- But it's a different matrix from the 1D Fourier Transform.

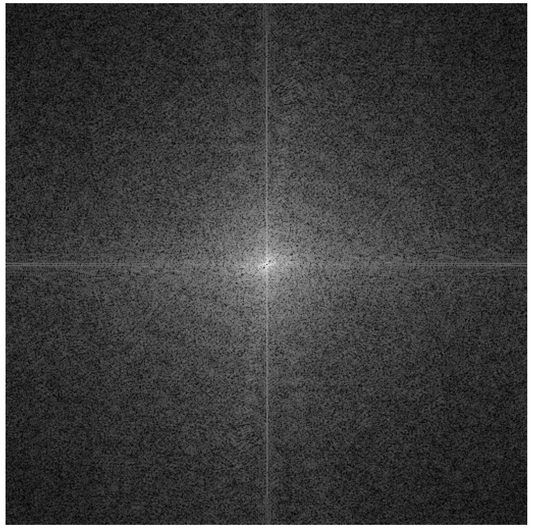
- Matrices for stuff that's fundamentally 2D (images) are more messy than those for 1D stuff
  - e.g. convolution is not a nice multidiagonal



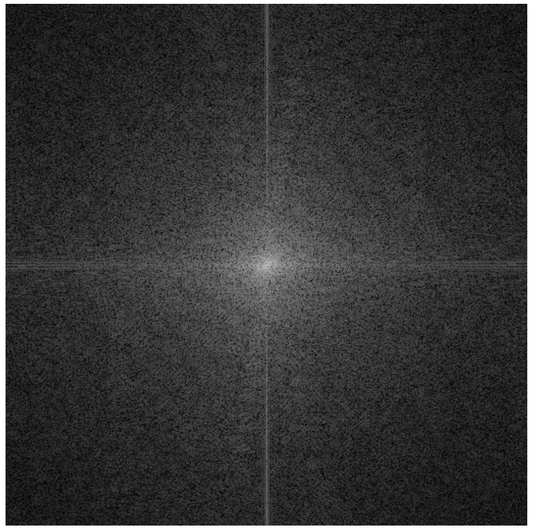
flatten

Fourier Matrix

Cosine Coeffs

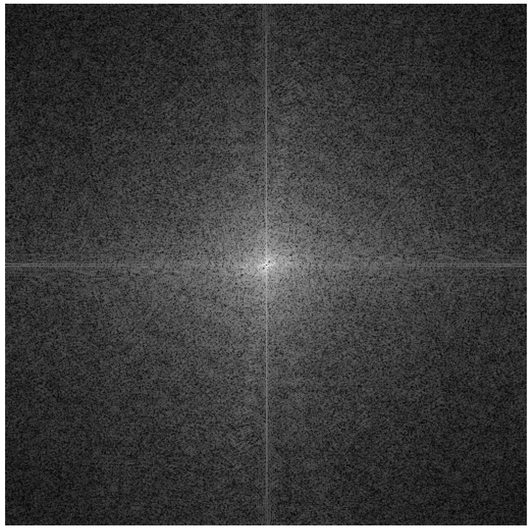


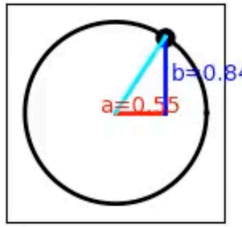
Sine Coeffs

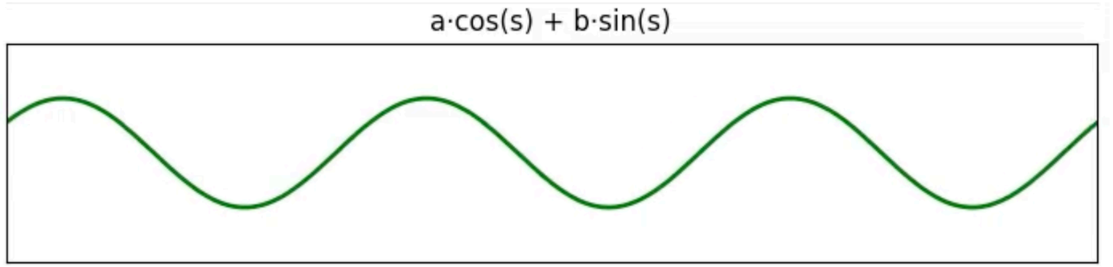


Sine & Cosine coeffs vs. Phase & Magnitude

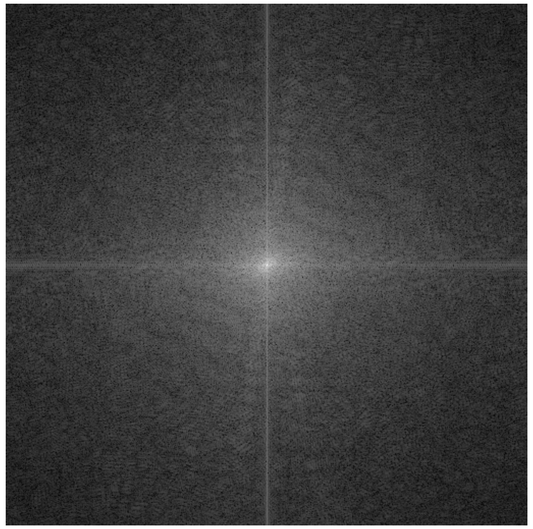
Cosine Coeffs



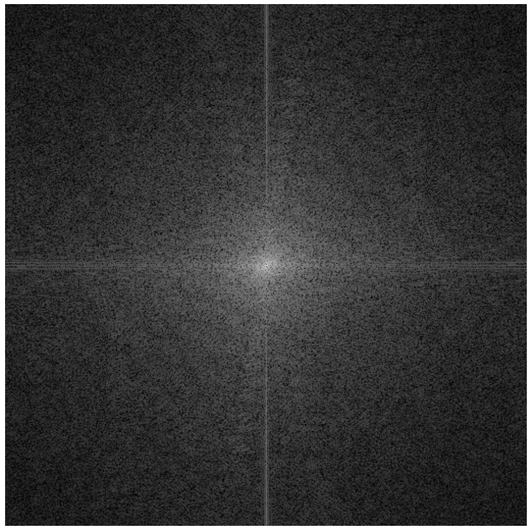


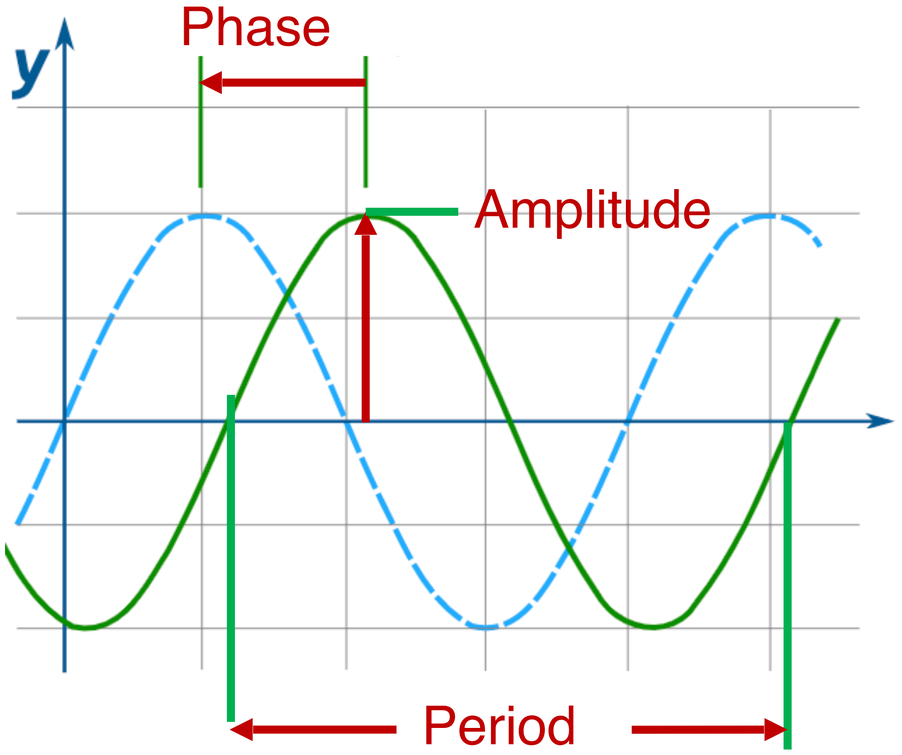


Magnitude

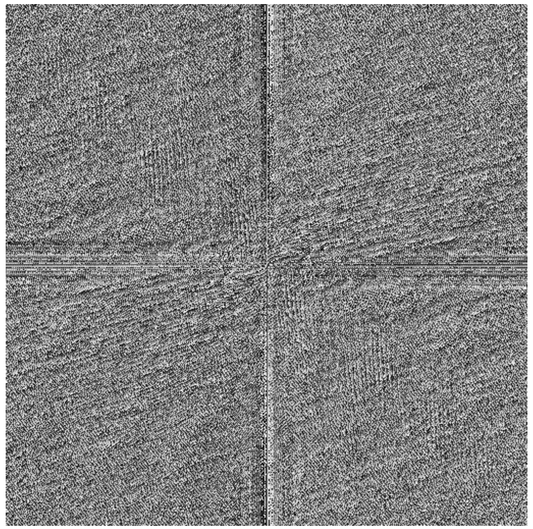


Sine Coeffs





Phase

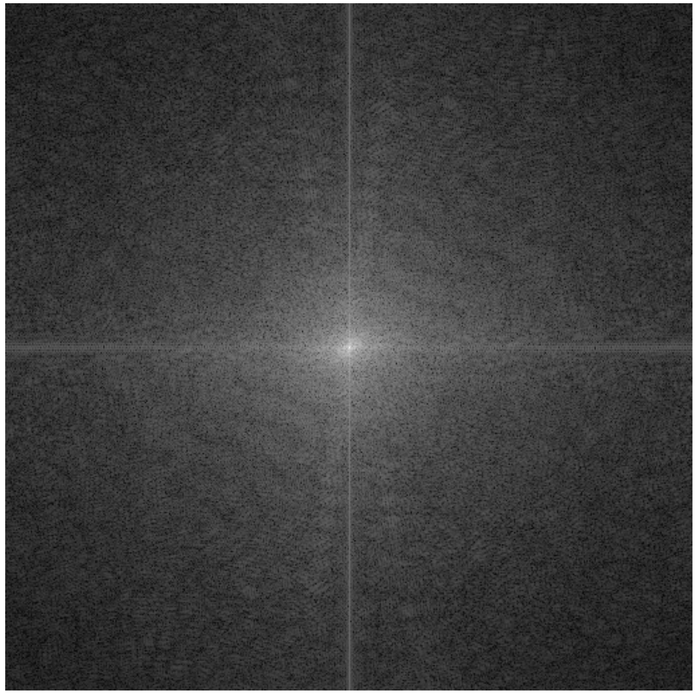


Real Images

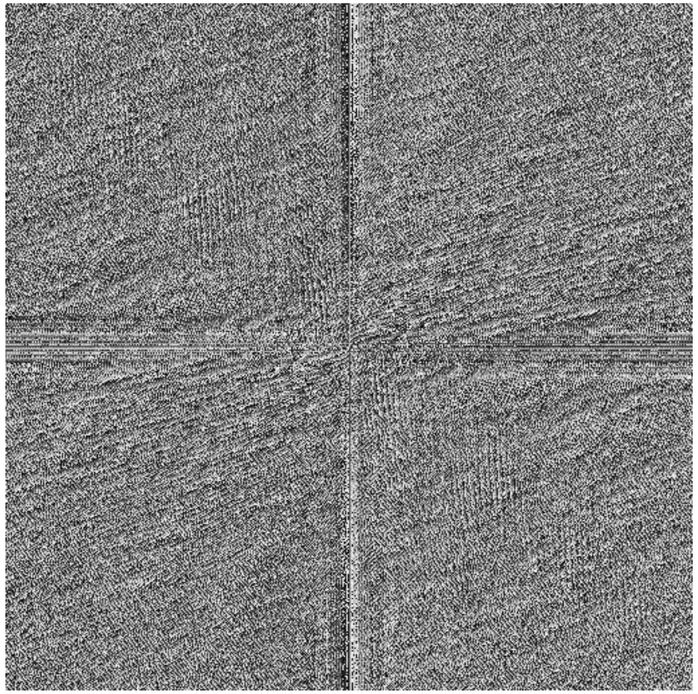
Image



Magnitude



Phase

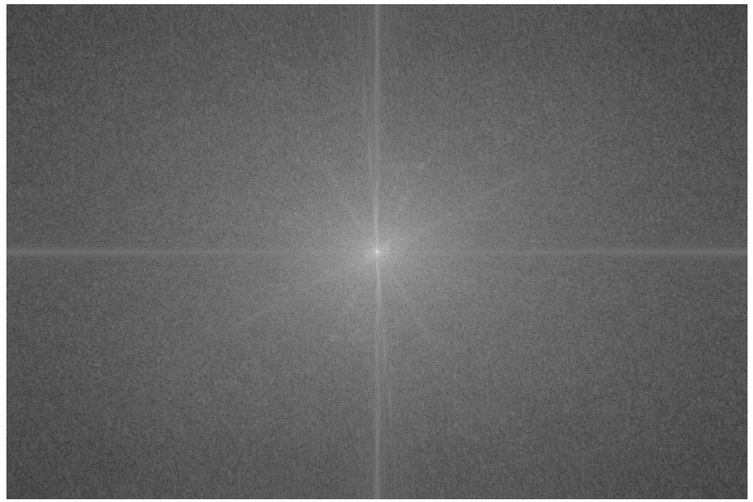


Real Images

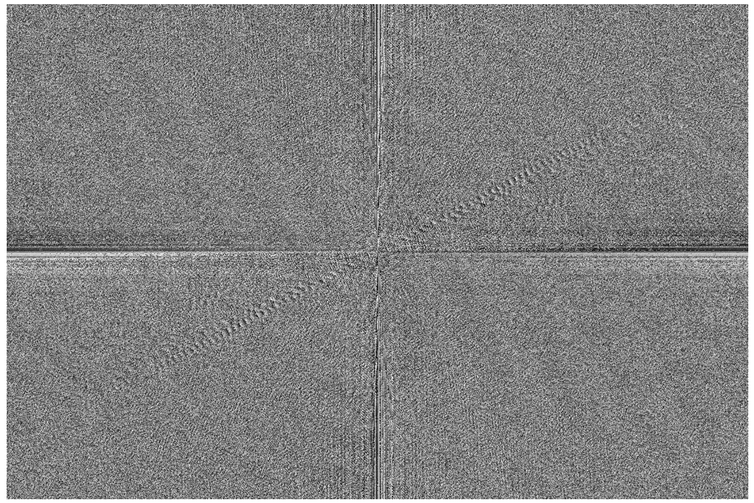
image



magnitude



phase



Slide credit: Christian Rupprecht, Oxford

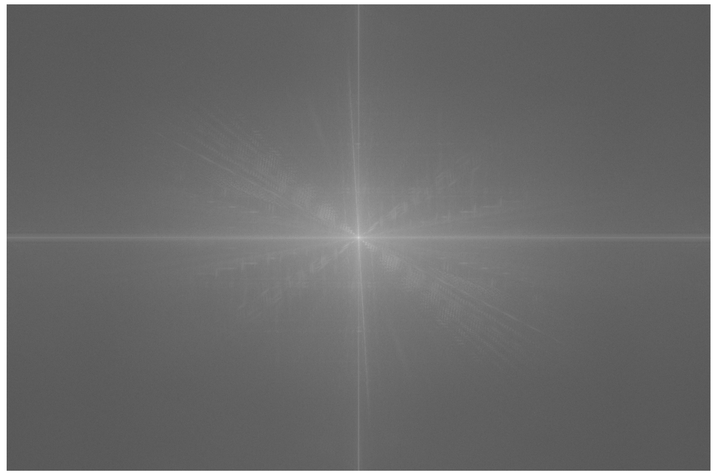
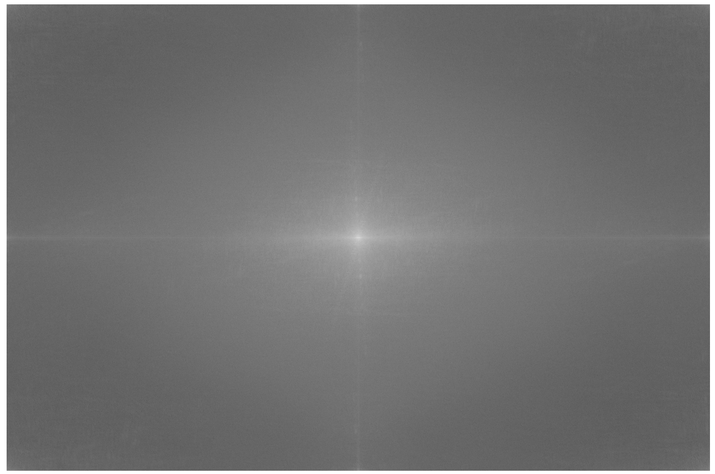
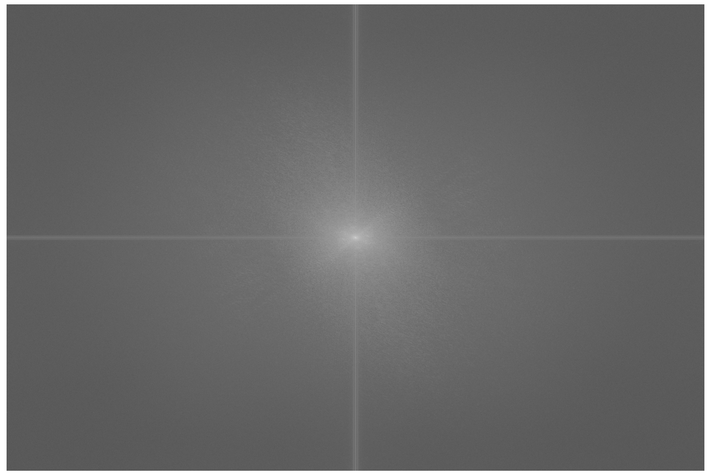
Examples

image






magnitude

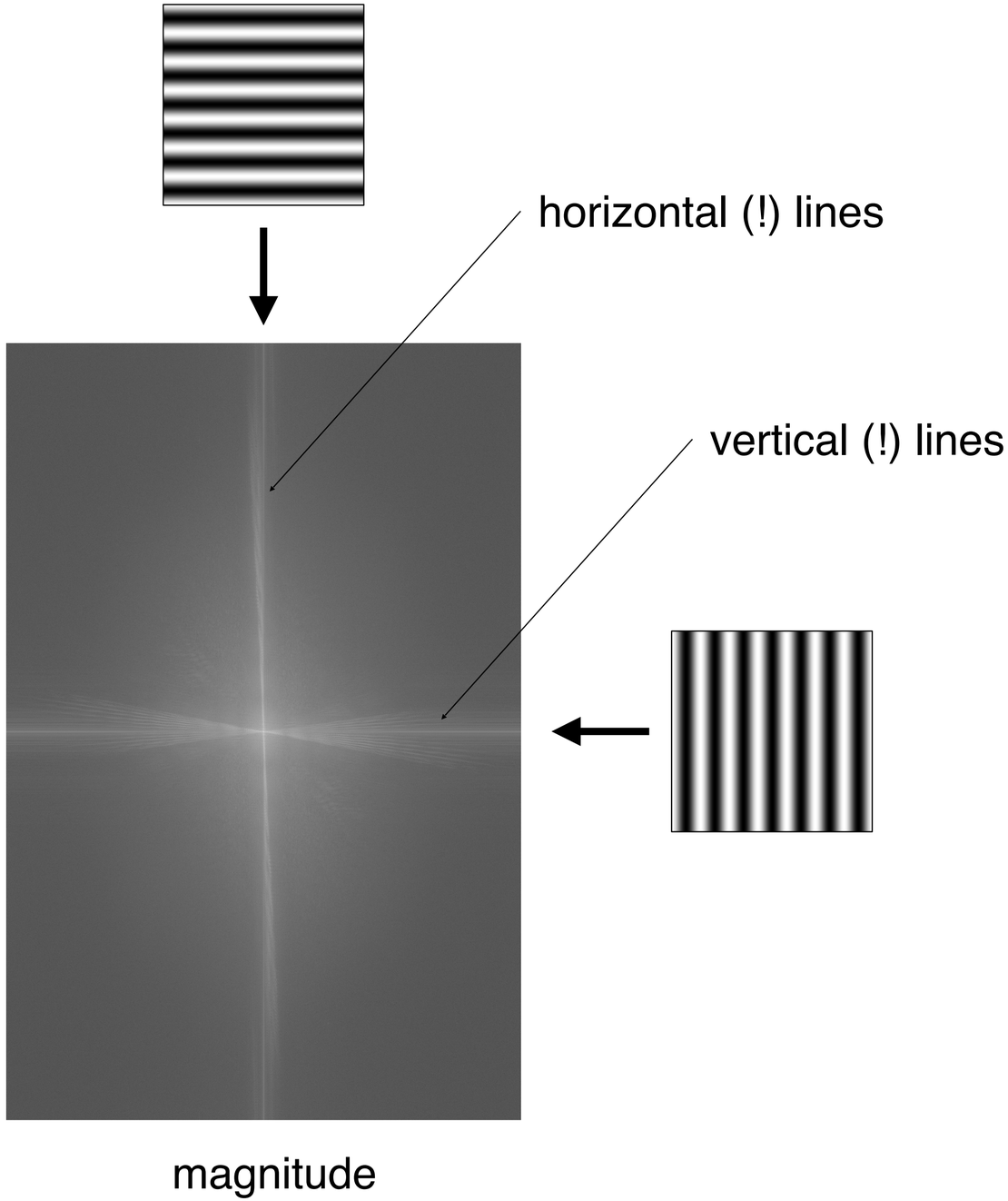





Slide credit: Christian Rupprecht, Oxford

Interpreting DFTs



image



magnitude

Slide credit: Christian Rupprecht, Oxford

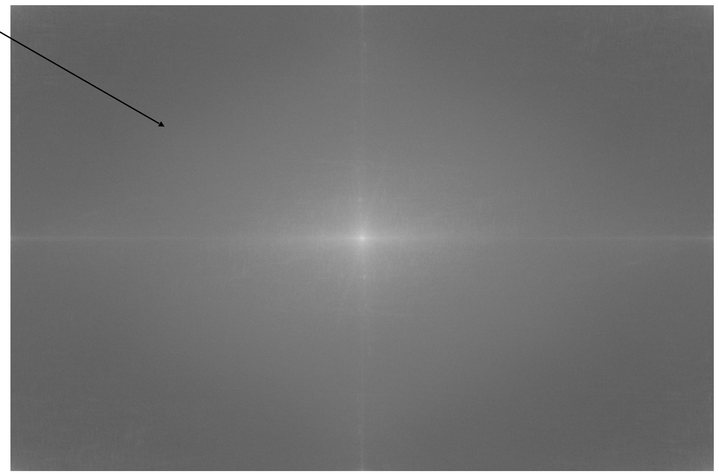
Interpreting DFTs

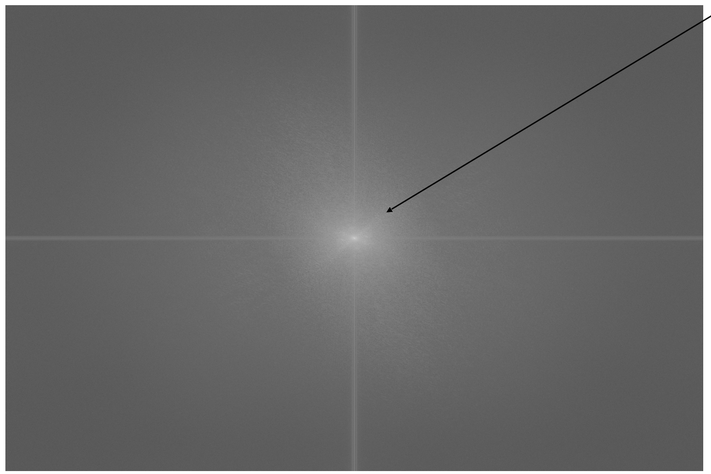




Away from the centre:
high-frequency details

Close to the centre:
low frequencies





Slide credit: Christian Rupprecht, Oxford

Warning

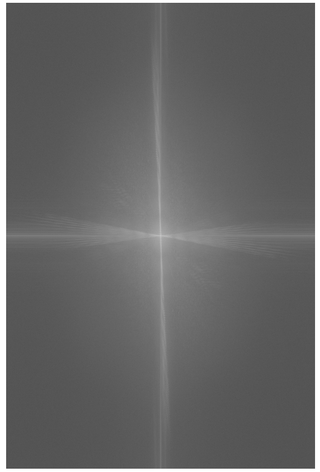
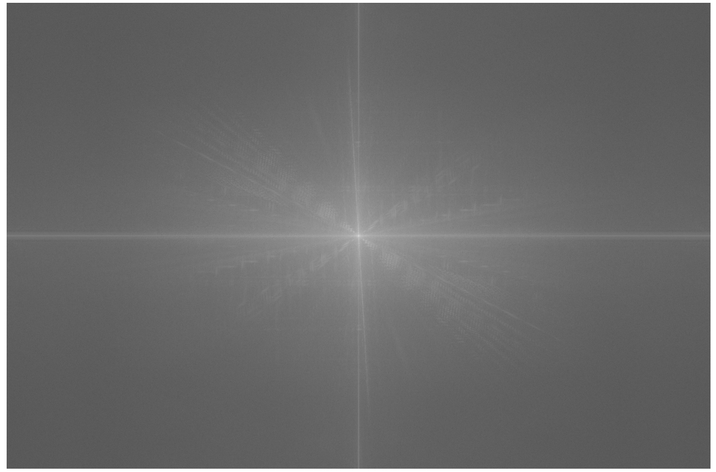
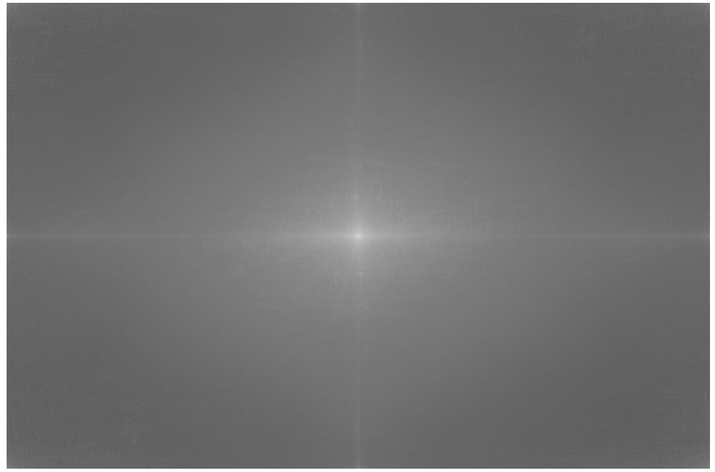
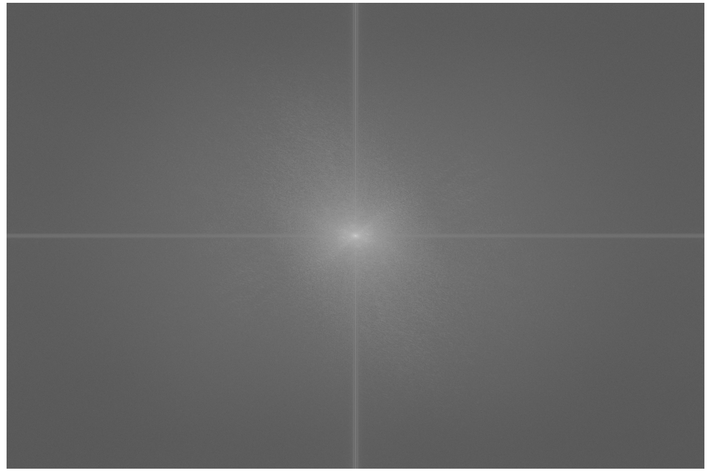
- The wrap-around assumption in Fourier transform causes a lot of vertical/horizontal power in the spectrum
  - For analysis purposes, you may need proper windowing

image






magnitude

Slide credit: Christian Rupprecht, Oxford

Why do we care about the basis of our image space?

The coefficients of an image for different bases are a simple form of *image representation,* making some information about the image easily accessible (locality, spatial frequency, …)

Some operations are simpler in a different basis, computationally more efficient, or easier to interpret.

Let’s look at some linear operators and discuss how they work in the pixel basis and the Fourier basis!

Linear Image Processing = Linear Maps Between Images



Linear **Operator**

Word for ”function” that emphasizes that it maps functions to functions



Intensity Scaling - Pixel Basis

Image $\mathbf{f} \in \mathbb{R}^{H \times W \times 1}$ in pixel basis (i.e., each entry is the intensity of one pixel!)

Then the intensity scaling operator is a scalar a:

$$\mathbf{f}' = a\mathbf{f}$$



Intensity Scaling - Fourier Basis

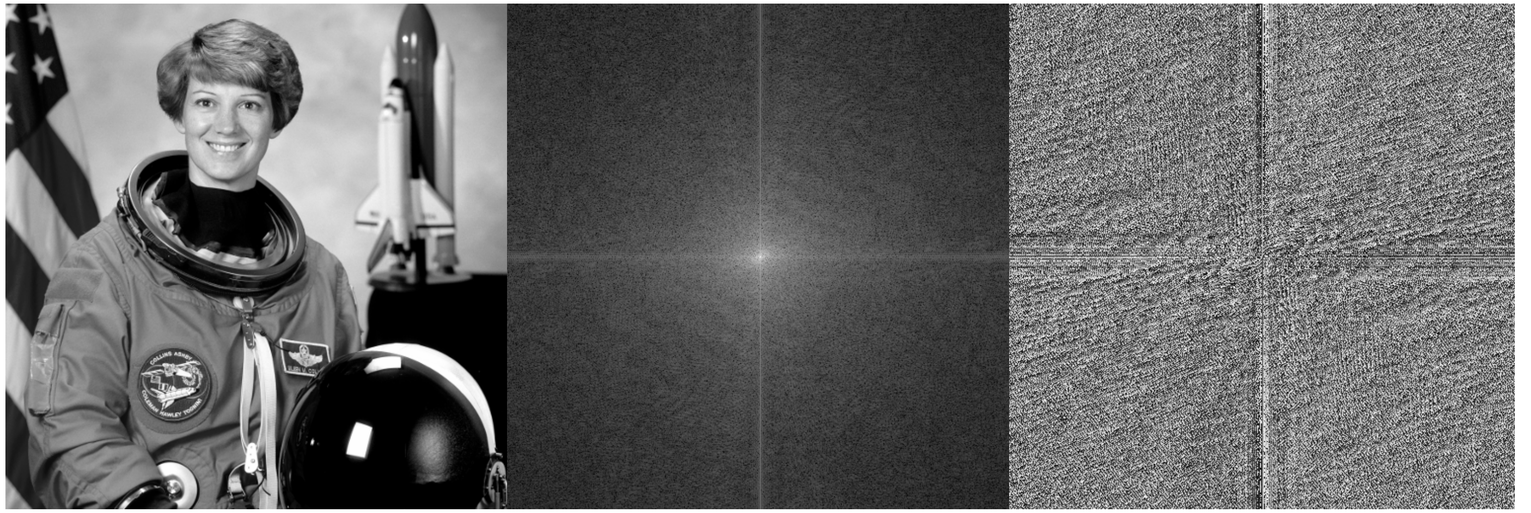
Fourier Transform of Image:

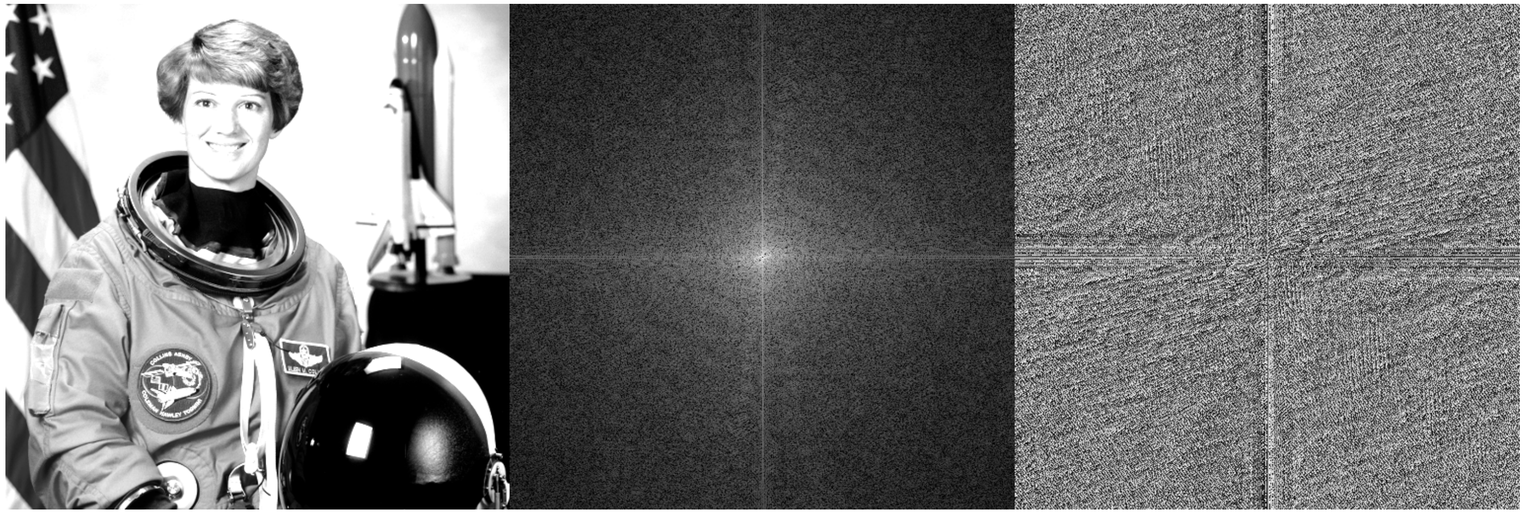
$$\mathcal{F}\{\mathbf{f}\}$$

Fourier Transform is linear (just a matrix multiply), hence:

$$\mathcal{F}\{a\mathbf{f}\} = a\mathcal{F}\{\mathbf{f}\}$$

Intensity scaling is a *diagonal matrix* in either basis.





Circular Shift - Pixel Basis

Image is multiplied with a *shift matrix* $\mathbf{T}$:

$$\mathbf{f}' = \mathbf{T}\mathbf{f}$$

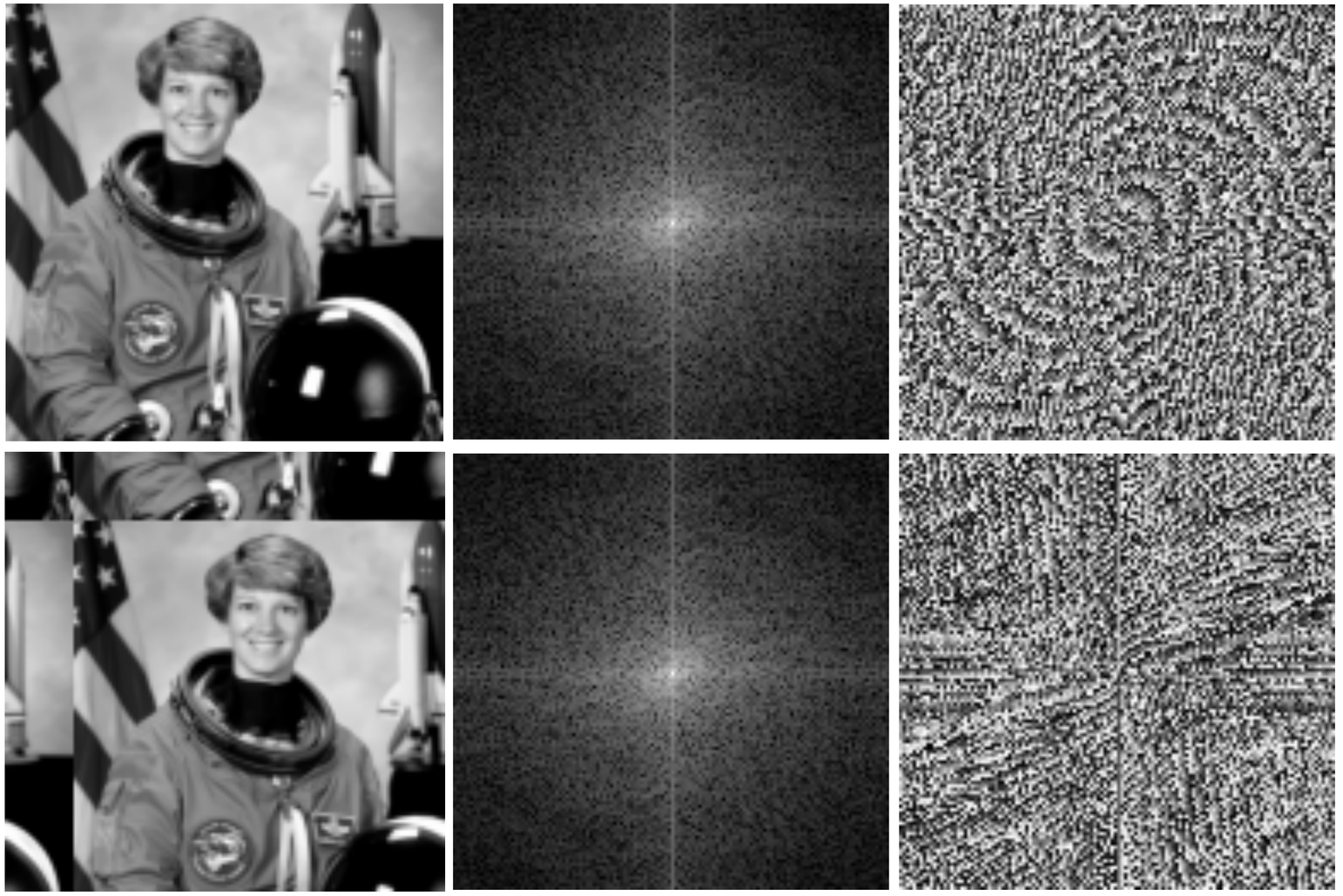
In 1D:

$$\mathbf{T} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

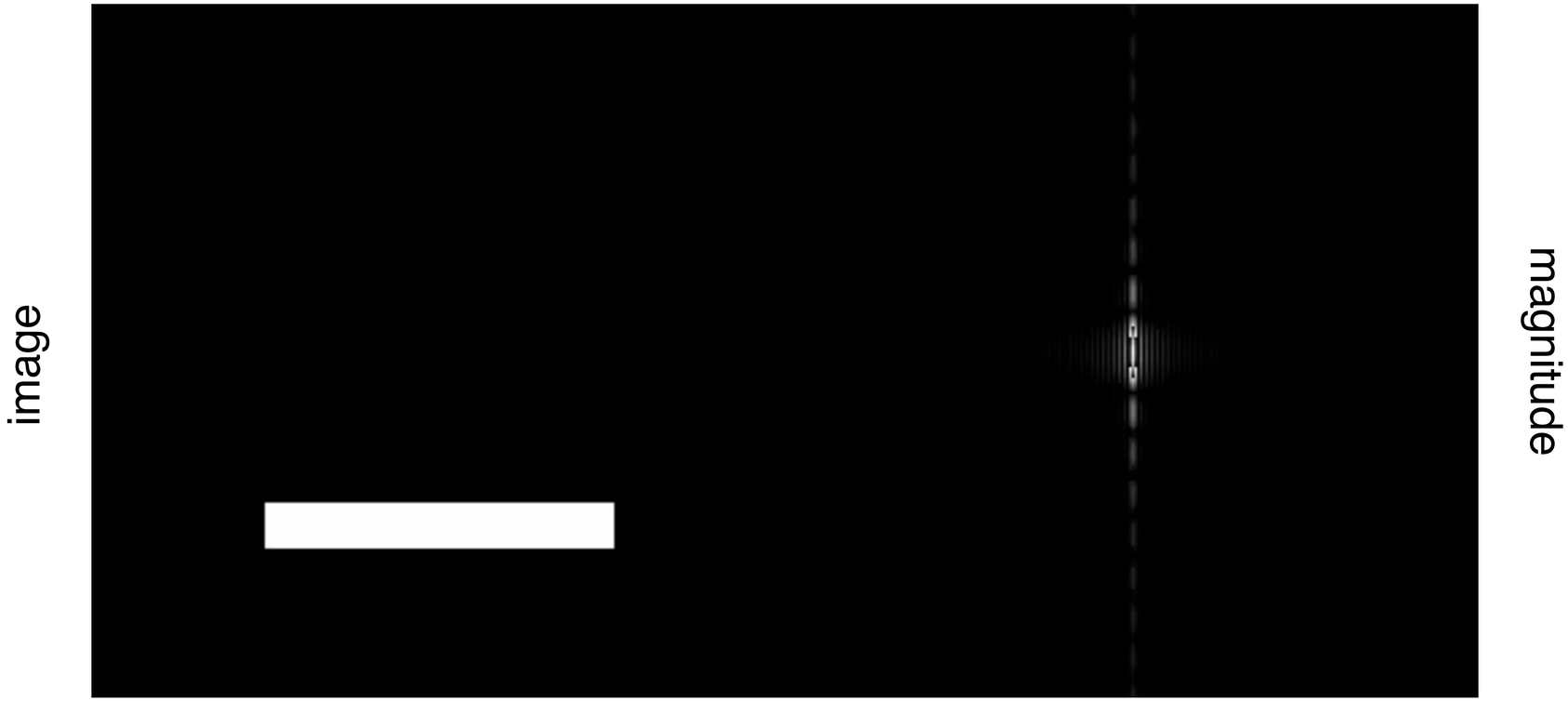


Circular Shift - Fourier Basis

Only the phase changes, magnitude stays constant!



Shifting an image with circular boundary conditions - animation



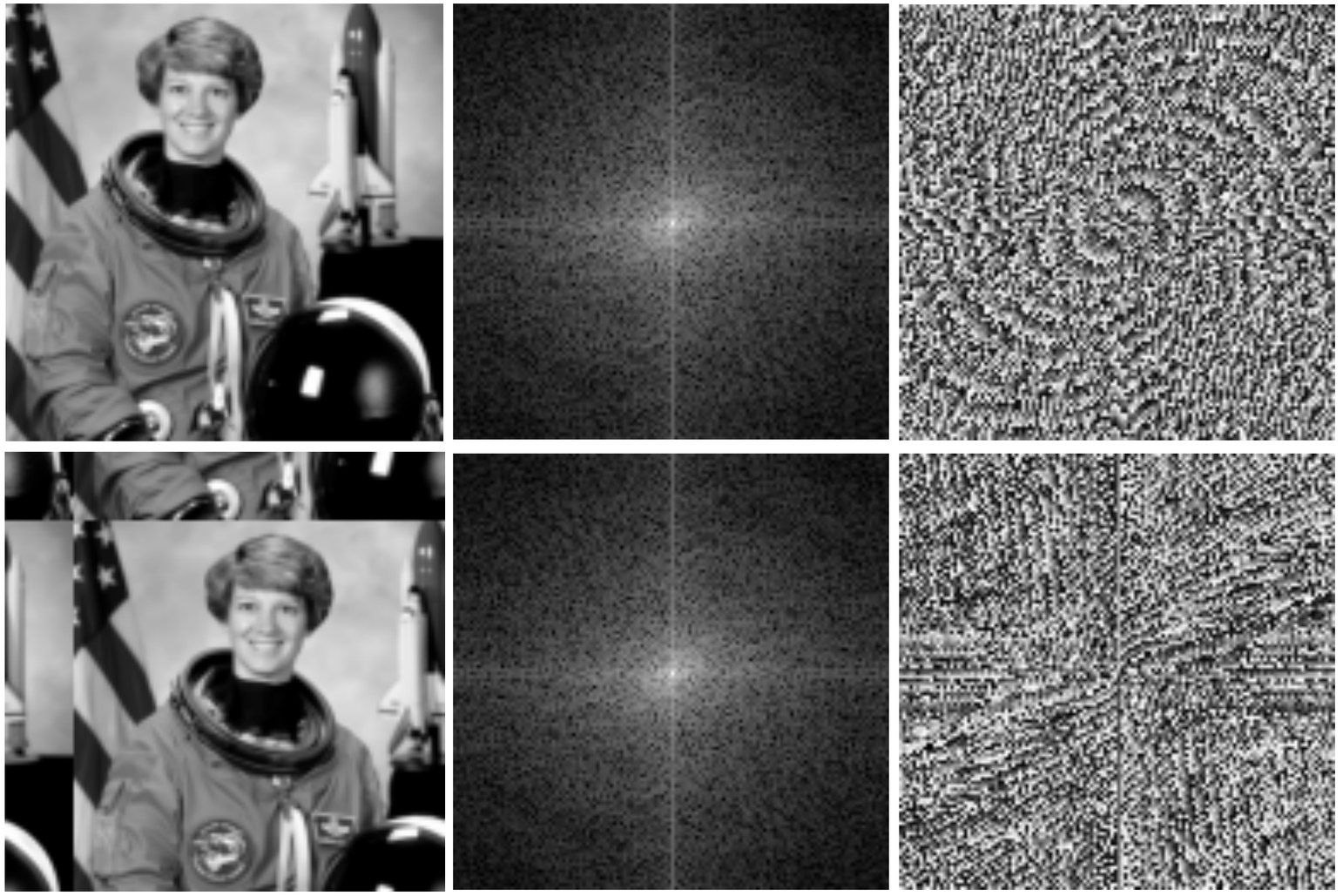
Why? How?

Circular Shift - Fourier Basis

Apply a change-of-basis to the shift matrix:

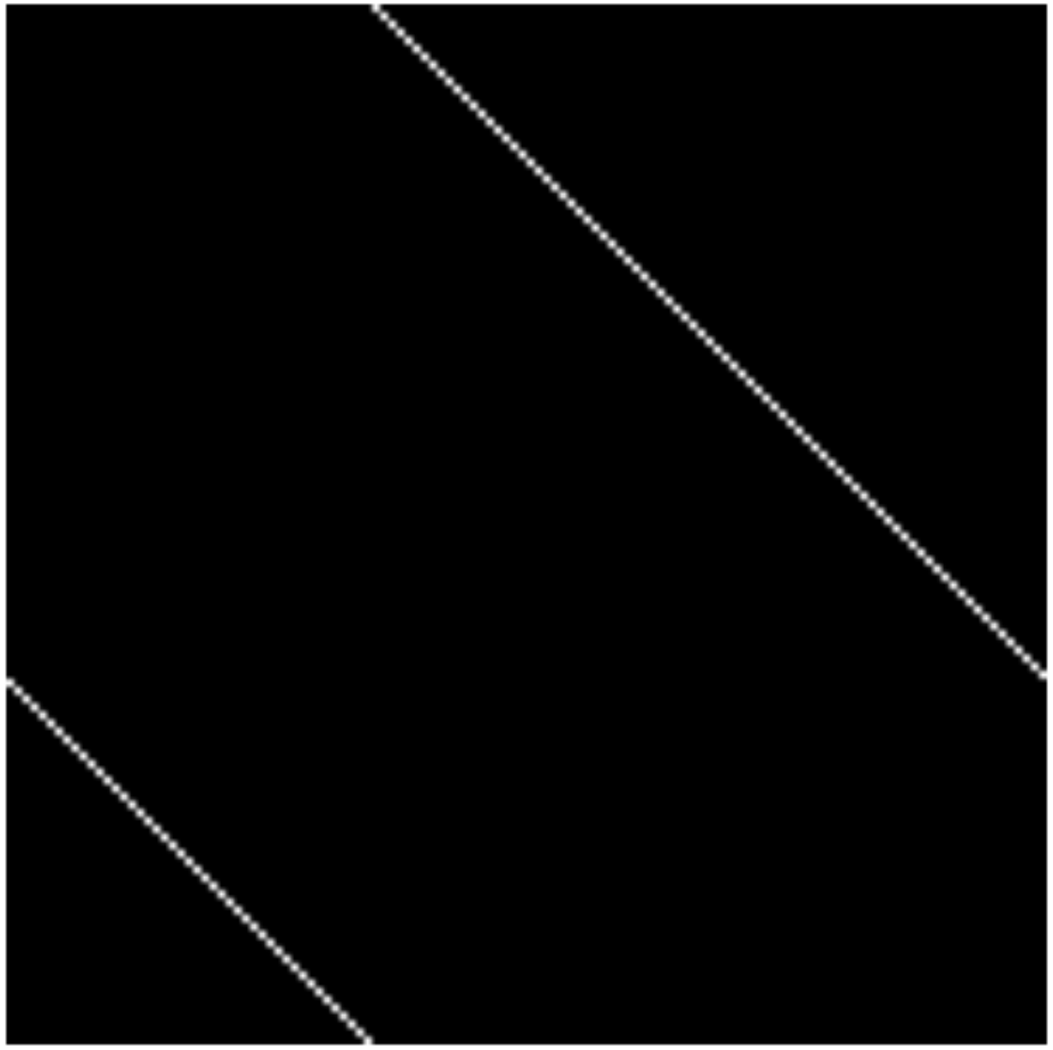
$$\mathcal{F}\mathbf{T}\mathcal{F}^{-1}$$

What happens?

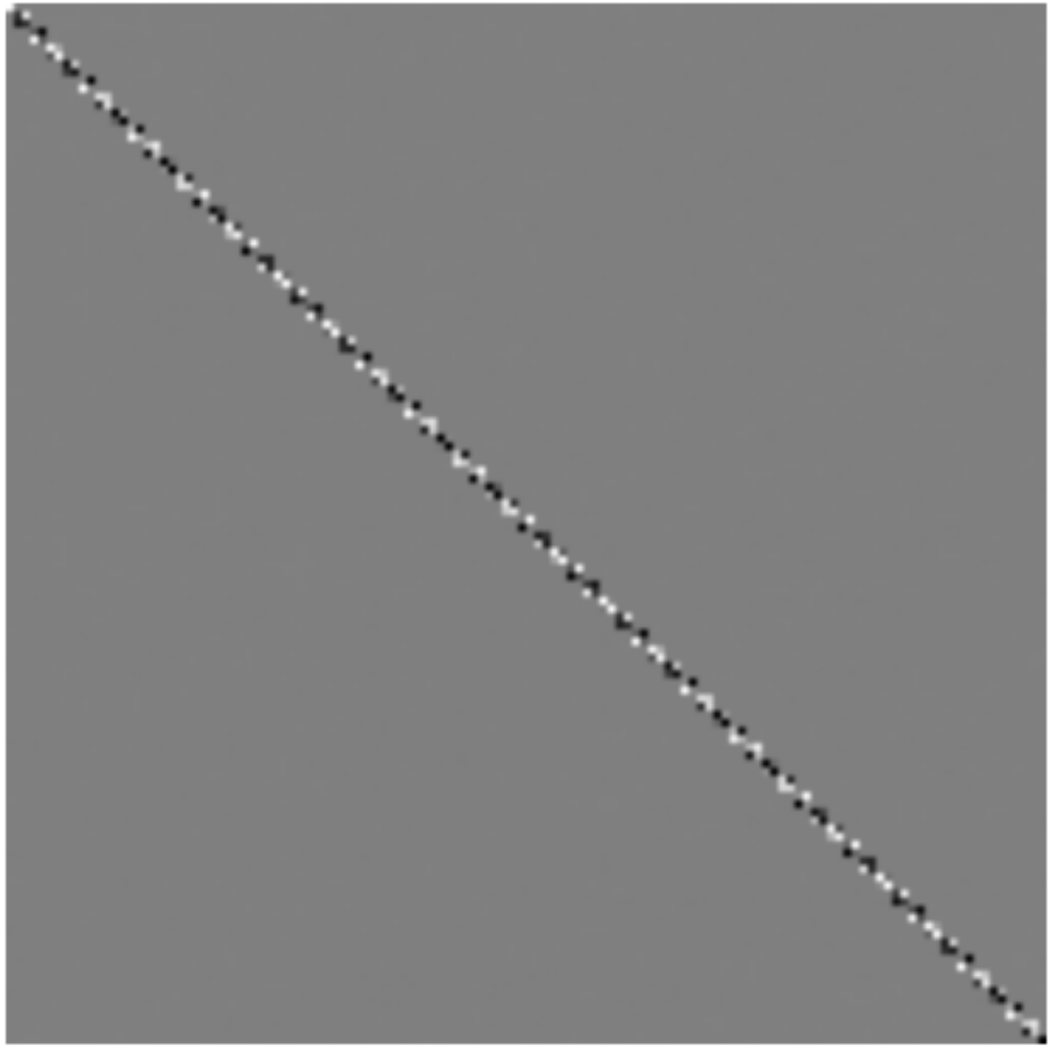


Circular Shift (in 1D) - Fourier Basis

$\mathbf{T}$

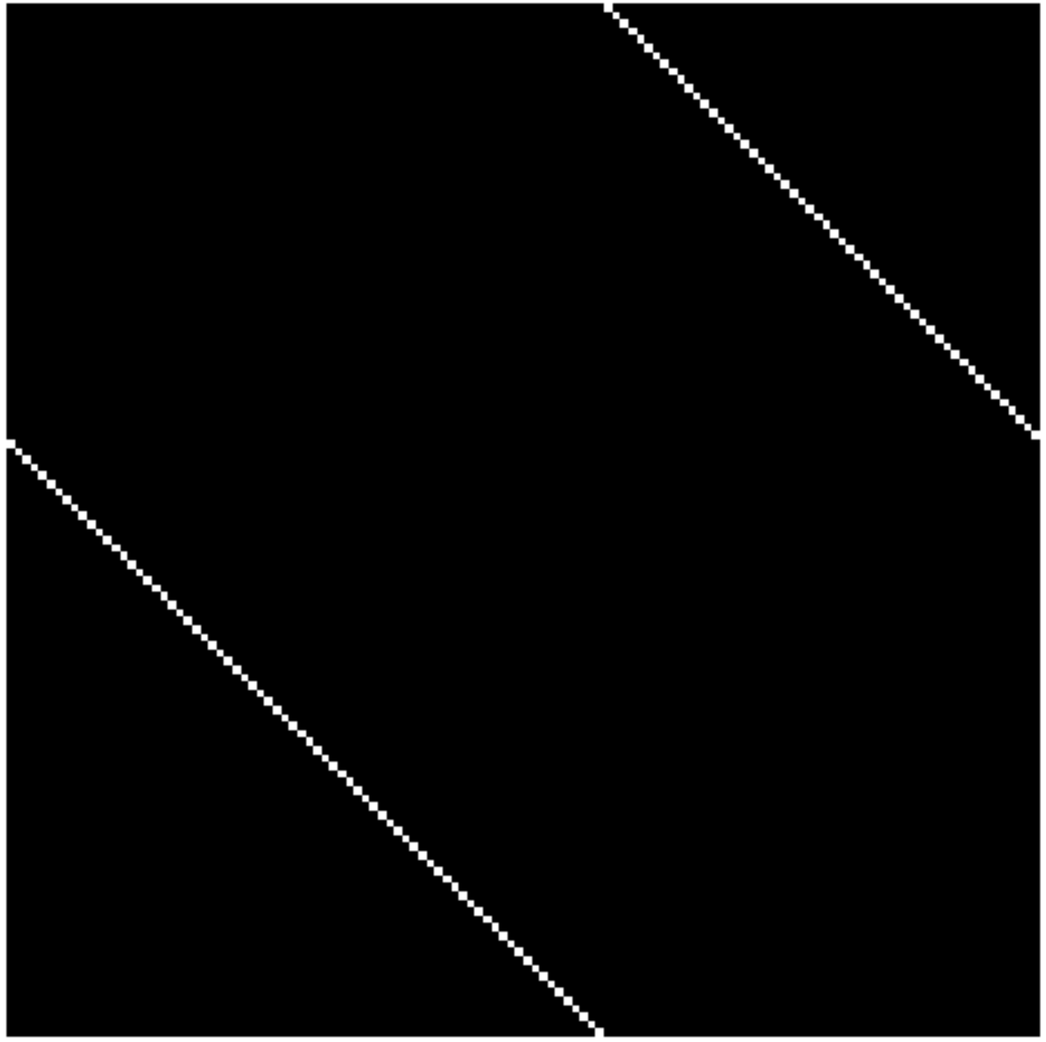


$\mathcal{F}\mathbf{T}\mathcal{F}^{-1}$

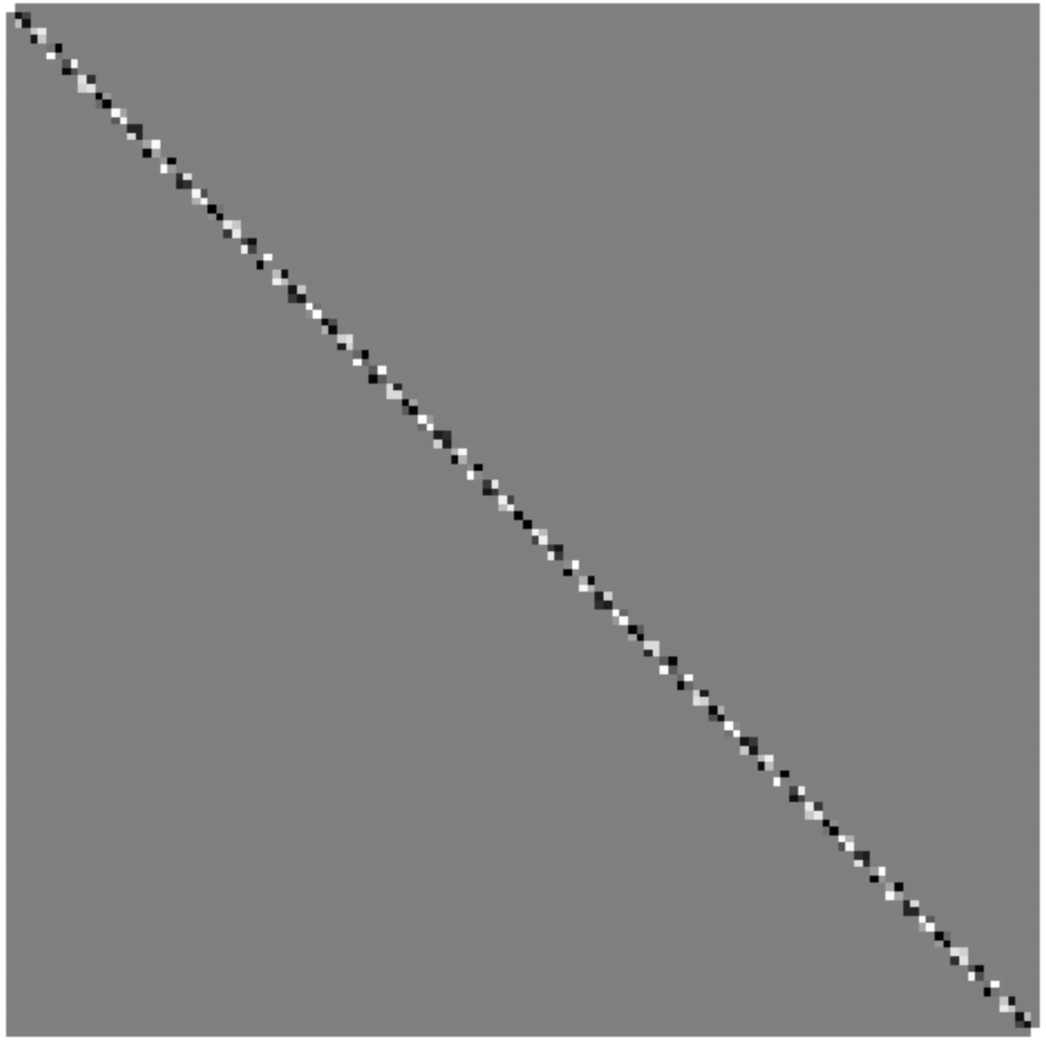


Circular Shift (in 1D) - Fourier Basis

$\mathbf{T}$

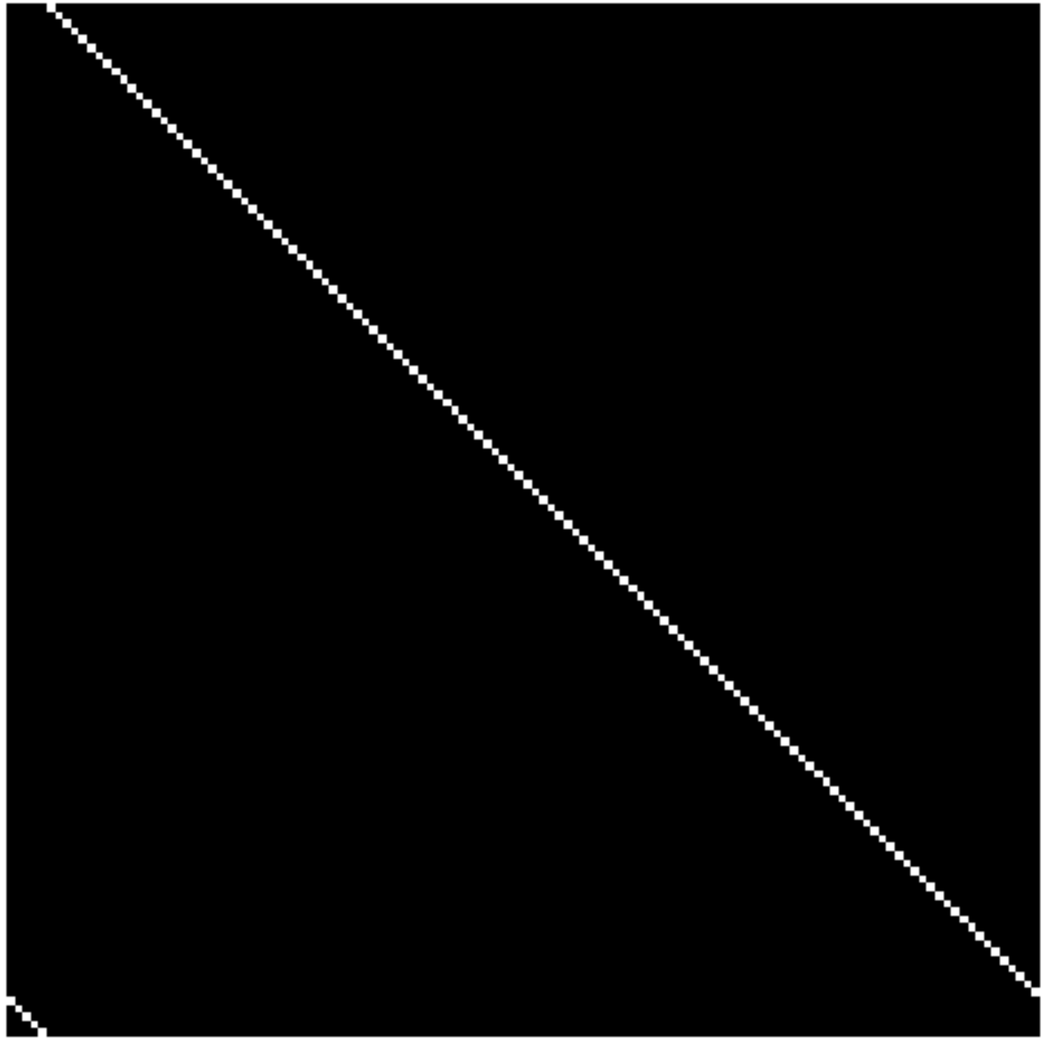


$\mathcal{F}\mathbf{T}\mathcal{F}^{-1}$

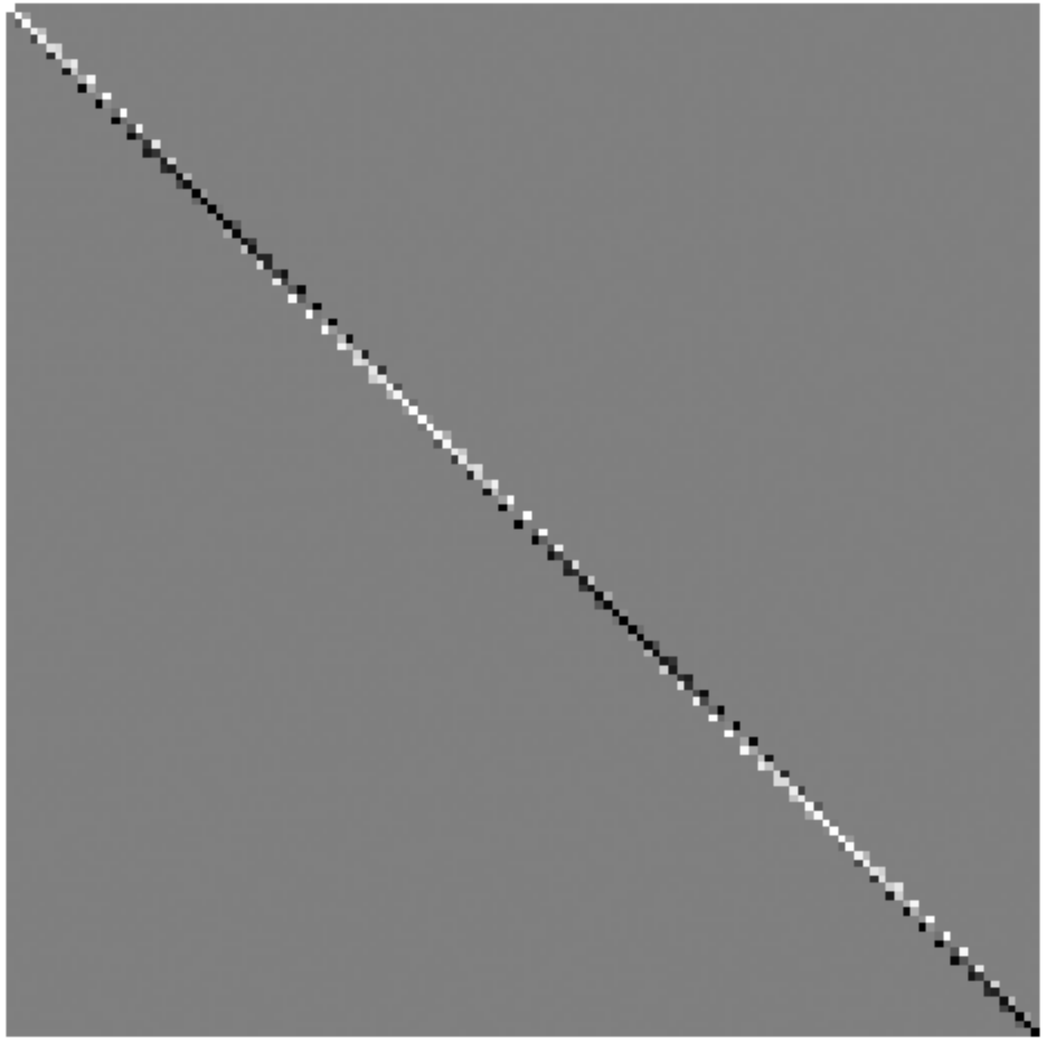


Circular Shift (in 1D) - Fourier Basis

$\mathbf{T}$

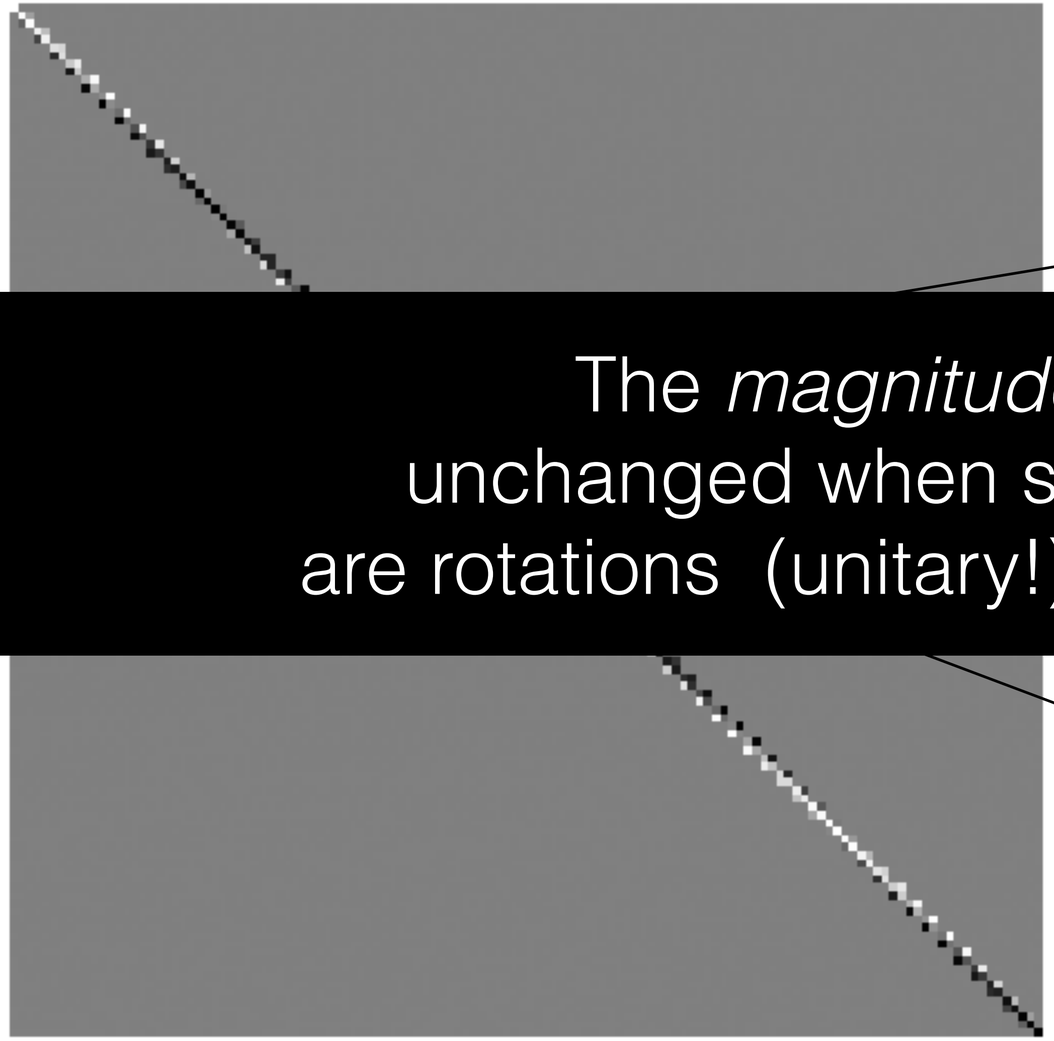


$\mathcal{F}\mathbf{T}\mathcal{F}^{-1}$



Circular Shift (in 1D) - Fourier Basis

$$\mathcal{F}\mathbf{T}\mathcal{F}^{-1}$$



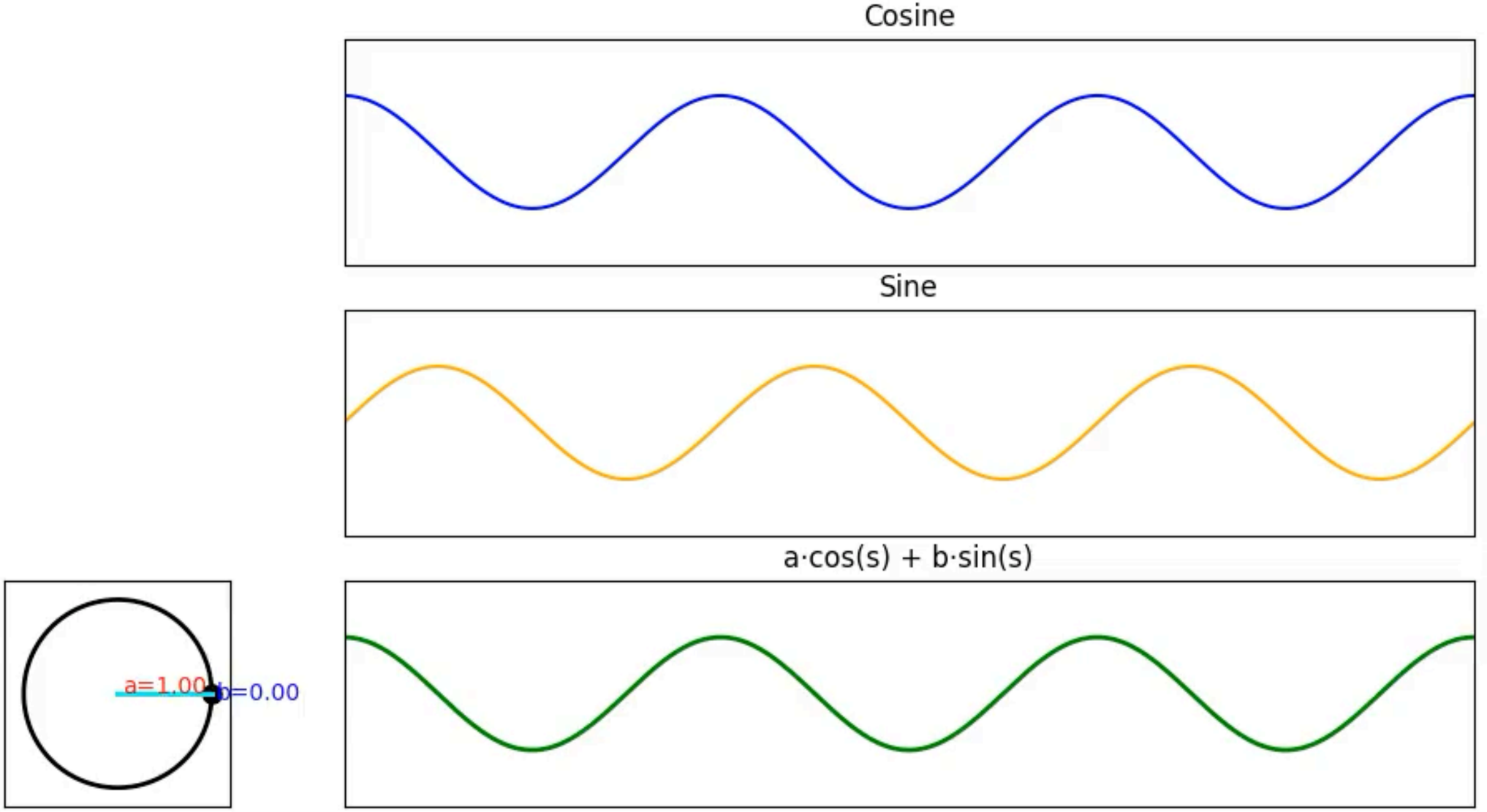
Rotation matrices!

$$\begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 & 0 & 0 \\ & & & & & \\ 0 & 0 & 0 & 0 & \sin\theta_3 & \cos\theta_3 \end{bmatrix}$$

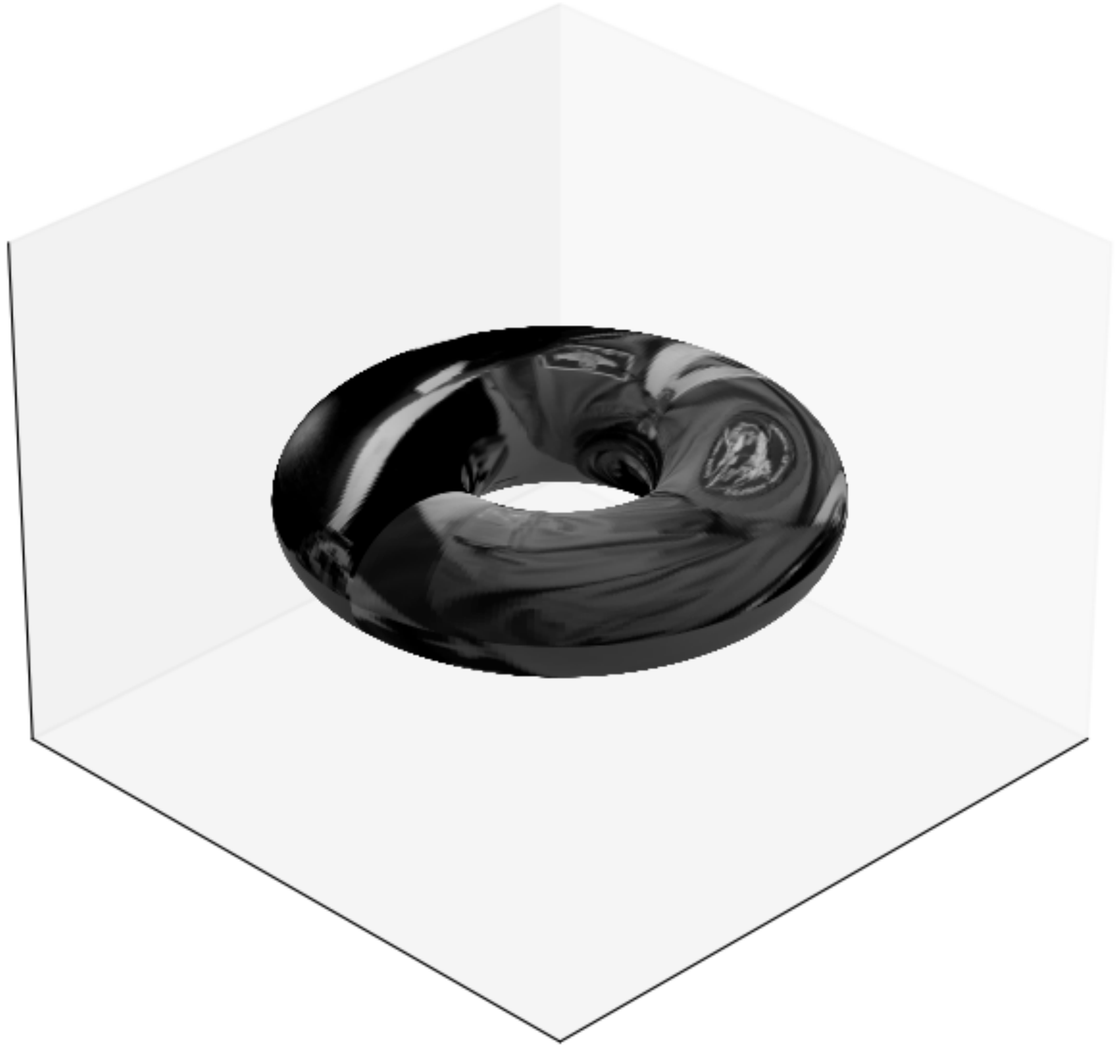
The *magnitude of an image* stays
unchanged when shifted b/c these matrices
are rotations (unitary!) in high dimensional space

Why? How?

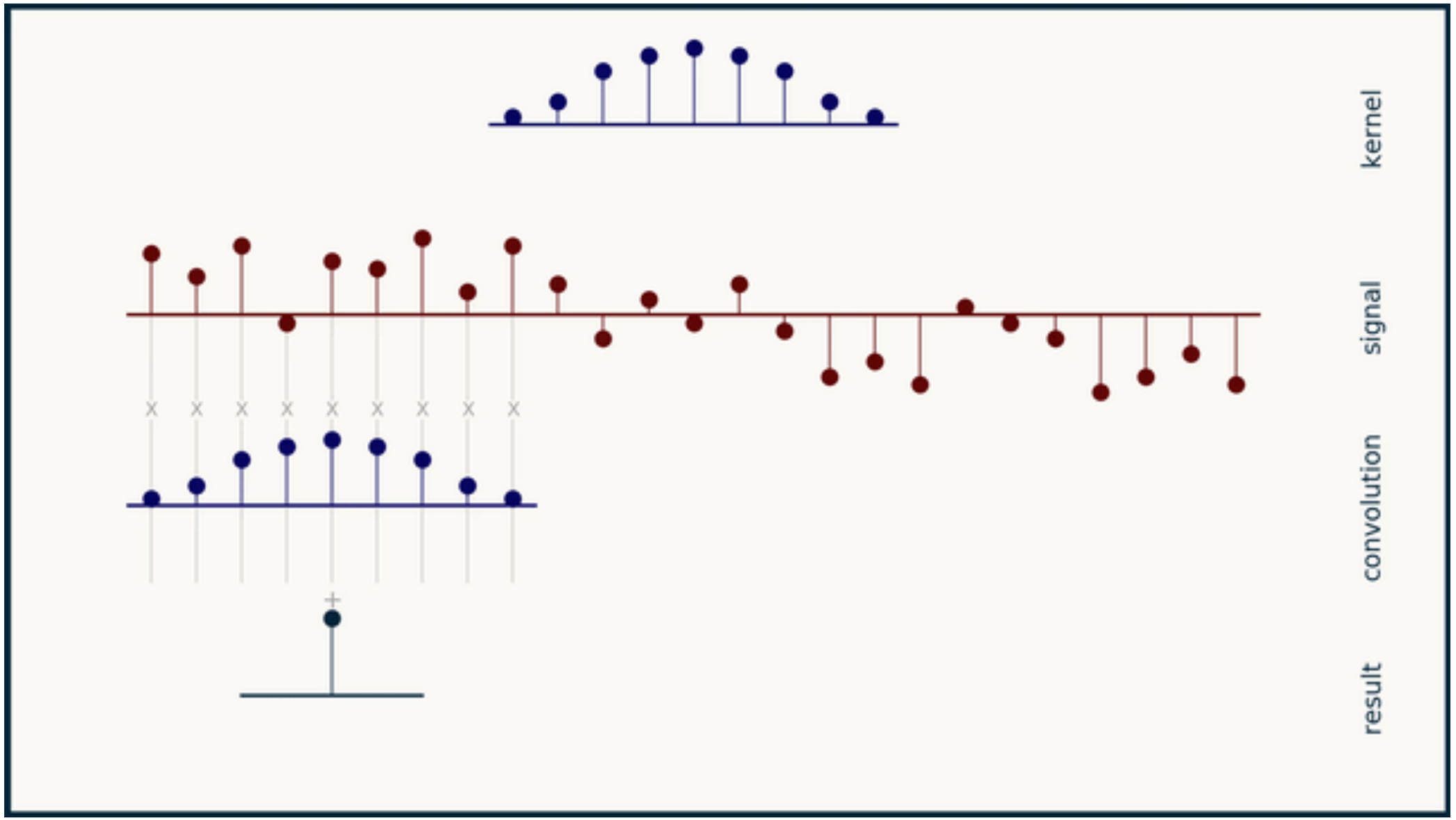
Each 2x2 Rotation Matrix re-scales the coeffs of the sine & cosine, shifting the resulting sine!



Each of the sines is “rotated” around the torus!

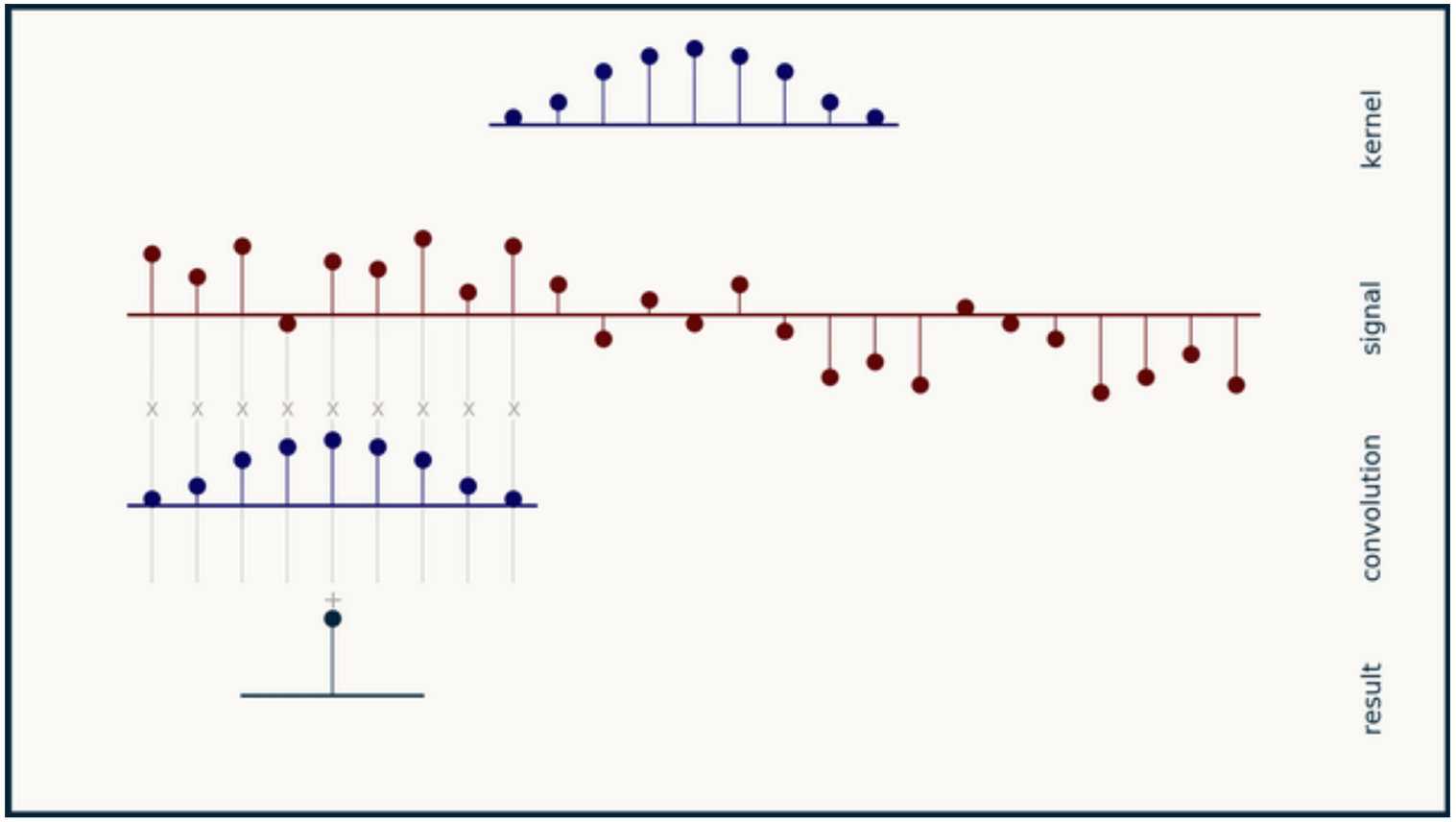


Recall: 1D Discrete Convolution



Source: Brandon Rohrer,
https://e2eml.school/convolution_one_d.html

Recall: 1D Discrete Convolution



Source: Brandon Rohrer, https://e2eml.school/convolution_one_d.html

$$k[x]$$

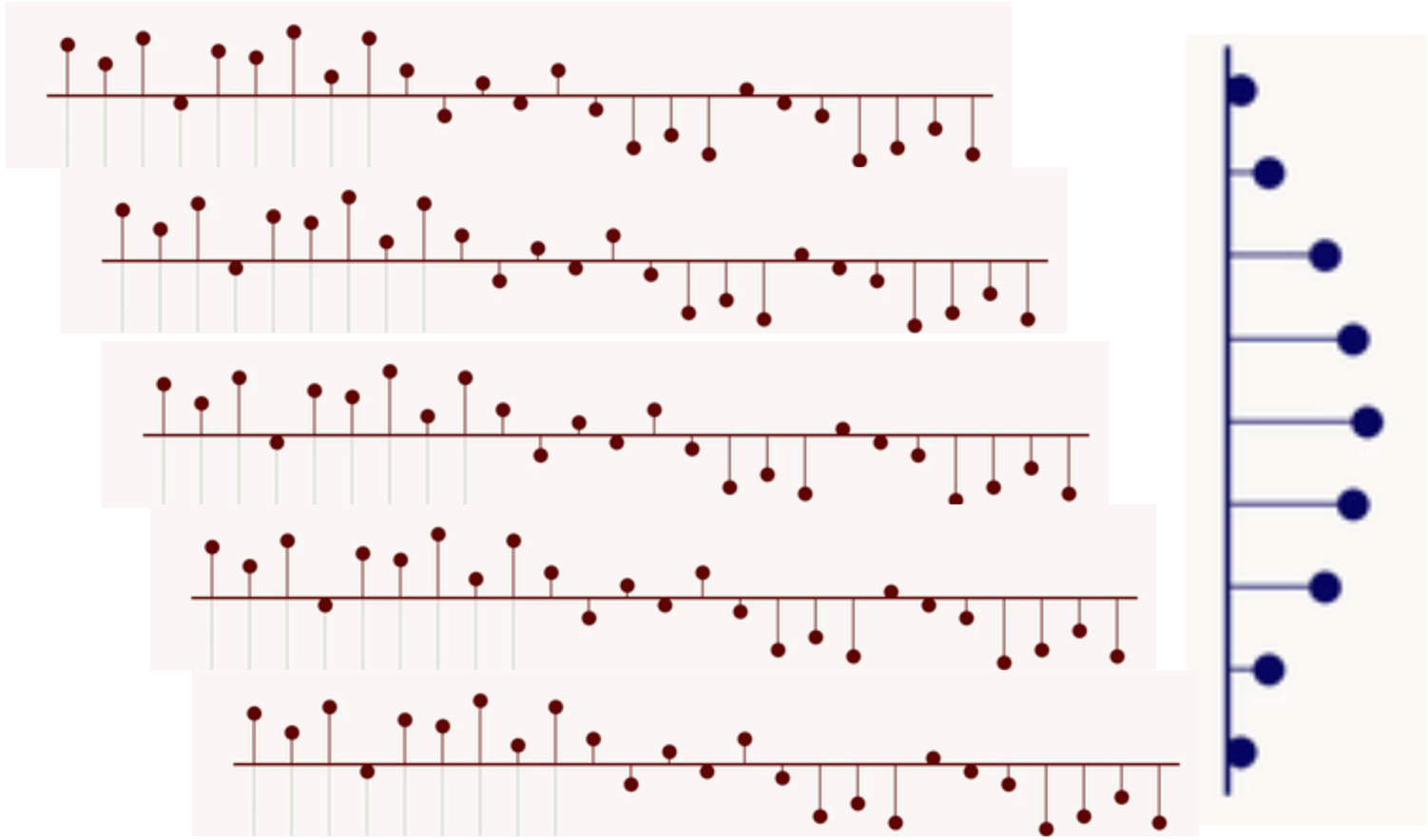
$$f[x]$$

$$(\mathbf{T}_x k)[x]$$

$$< \mathbf{T}_x k, f > [x]$$

At each coordinate x, shift kernel to that x
and compute a dot product between k and f.

Alternative perspective: 1D Discrete Convolution

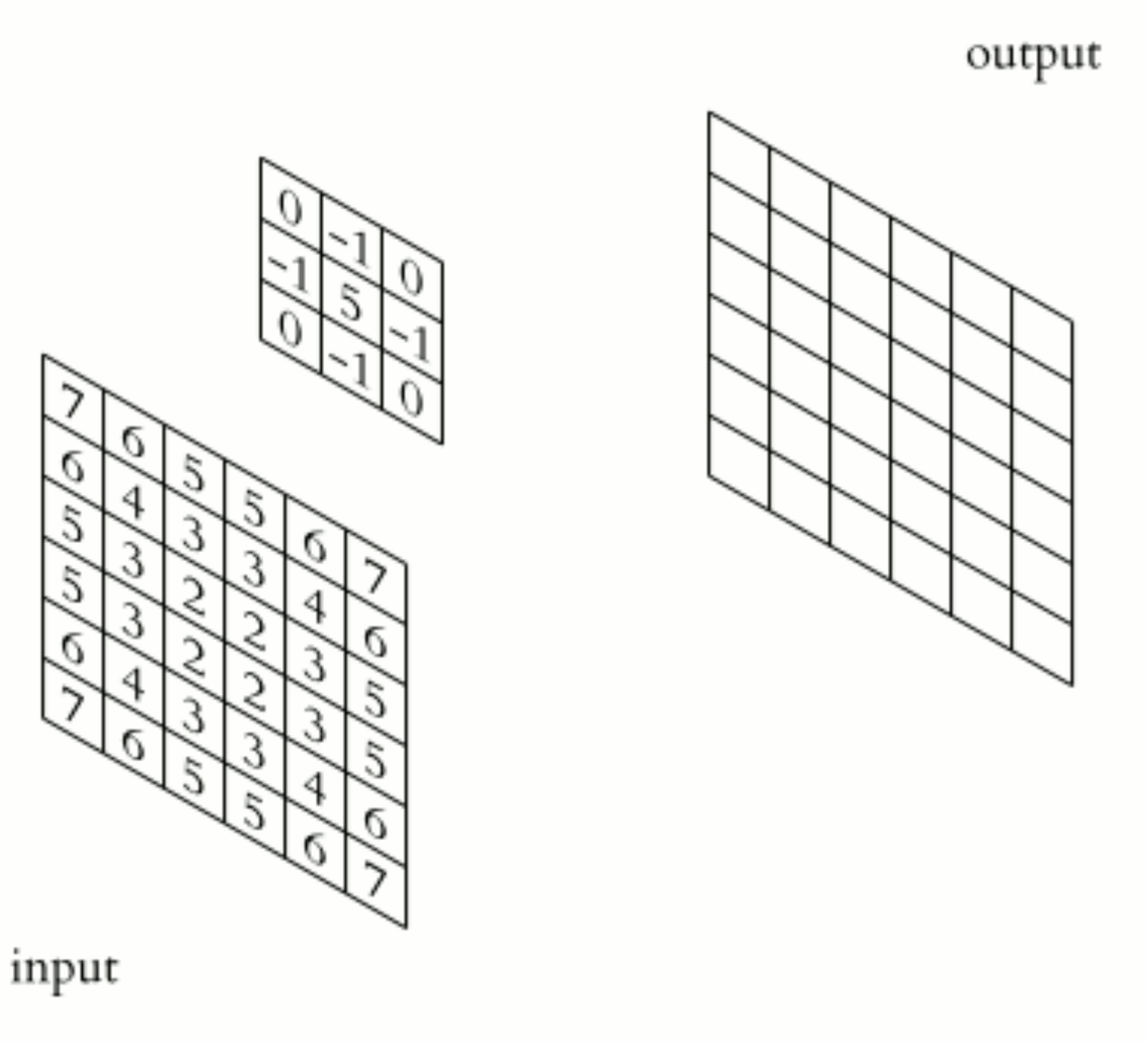


Source: Brandon Rohrer, https://e2eml.school/convolution_one_d.html

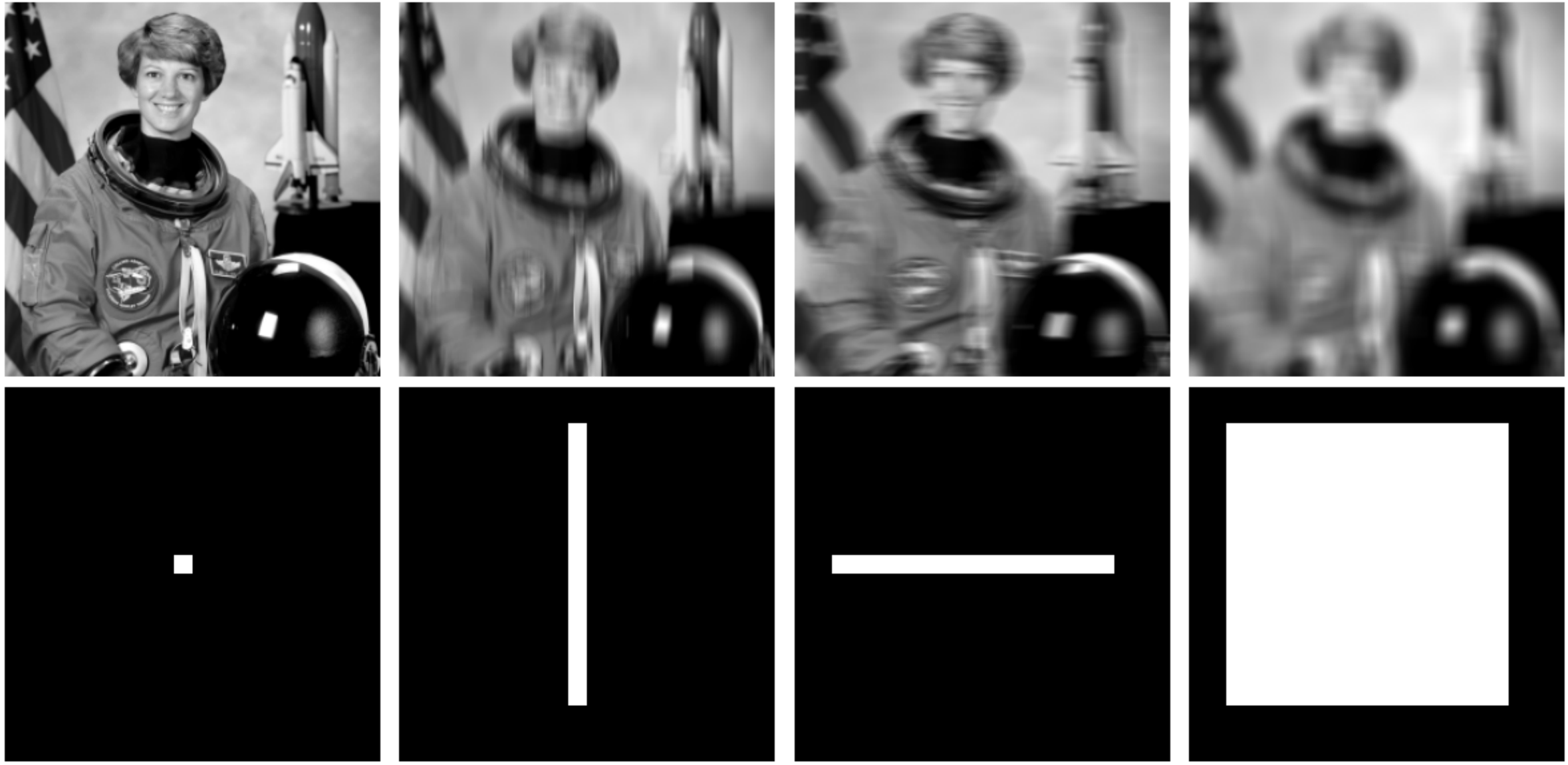
$$\sum_{shift\ x} k[x]T_x f$$

The convolution is a weighted sum of shifted versions of the image

Recall: 2D Discrete Convolution



By Michael Plotke - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=24288958

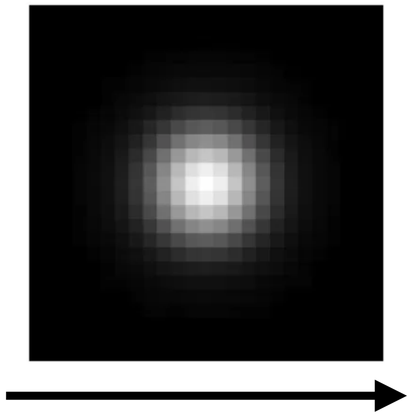


Identity | Vert. Motion Blur | Hor. Motion Blur | Averaging / “Blur Kernel”

Blur Kernels

Blur kernels are special convolutions that locally average over neighborhoods.

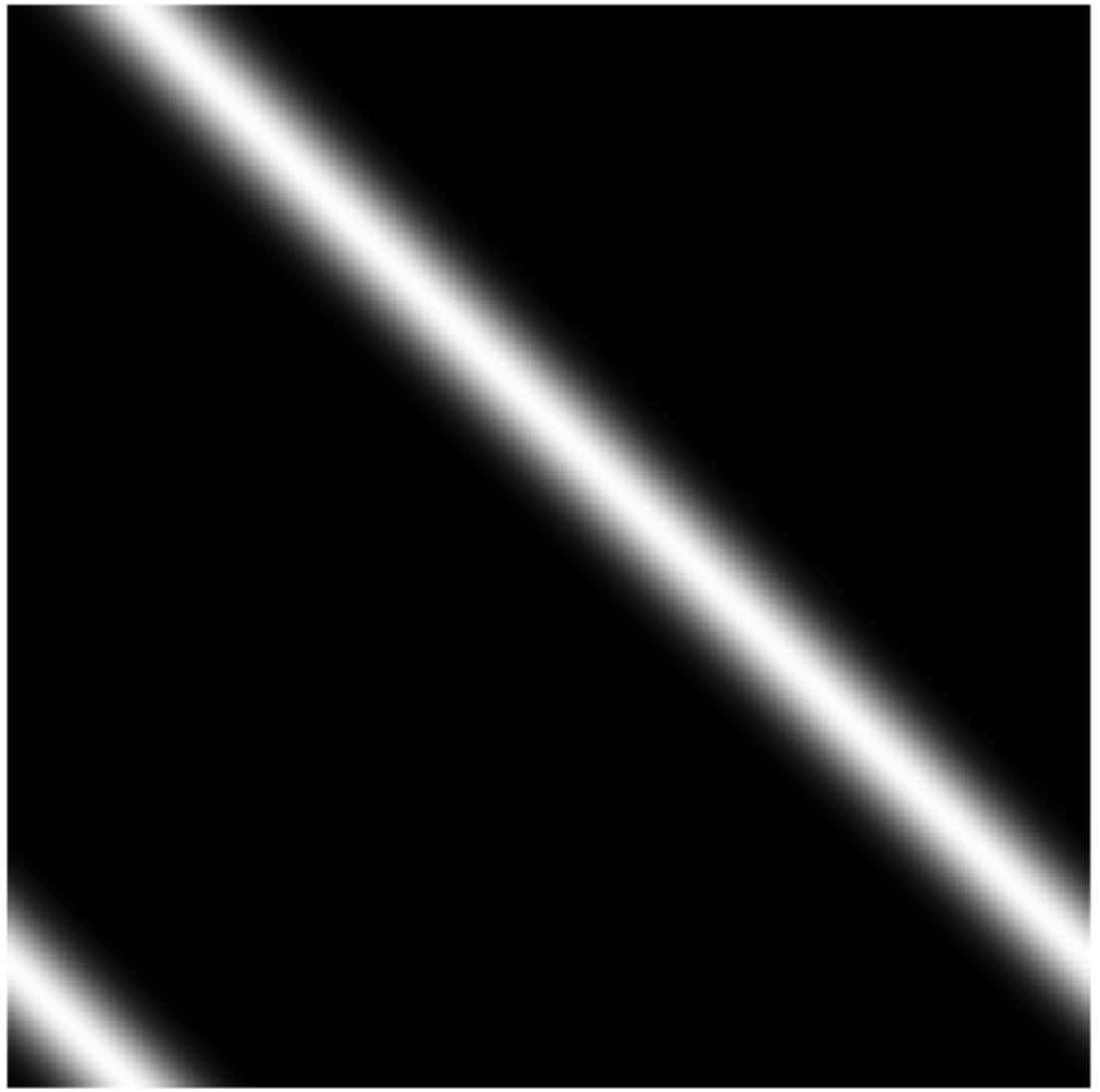






Convolution as a single matrix-vector product

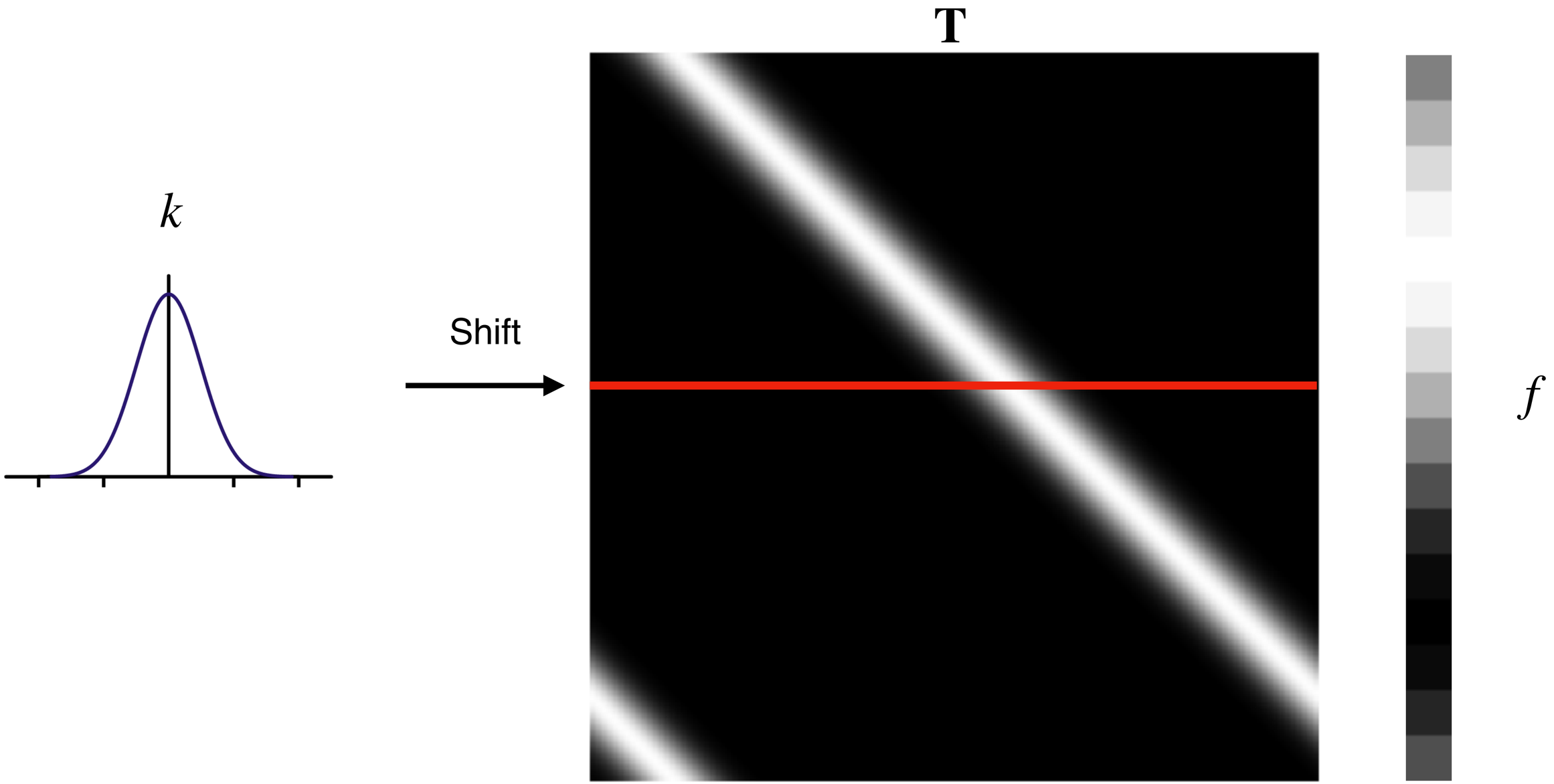
$\mathbf{T}$





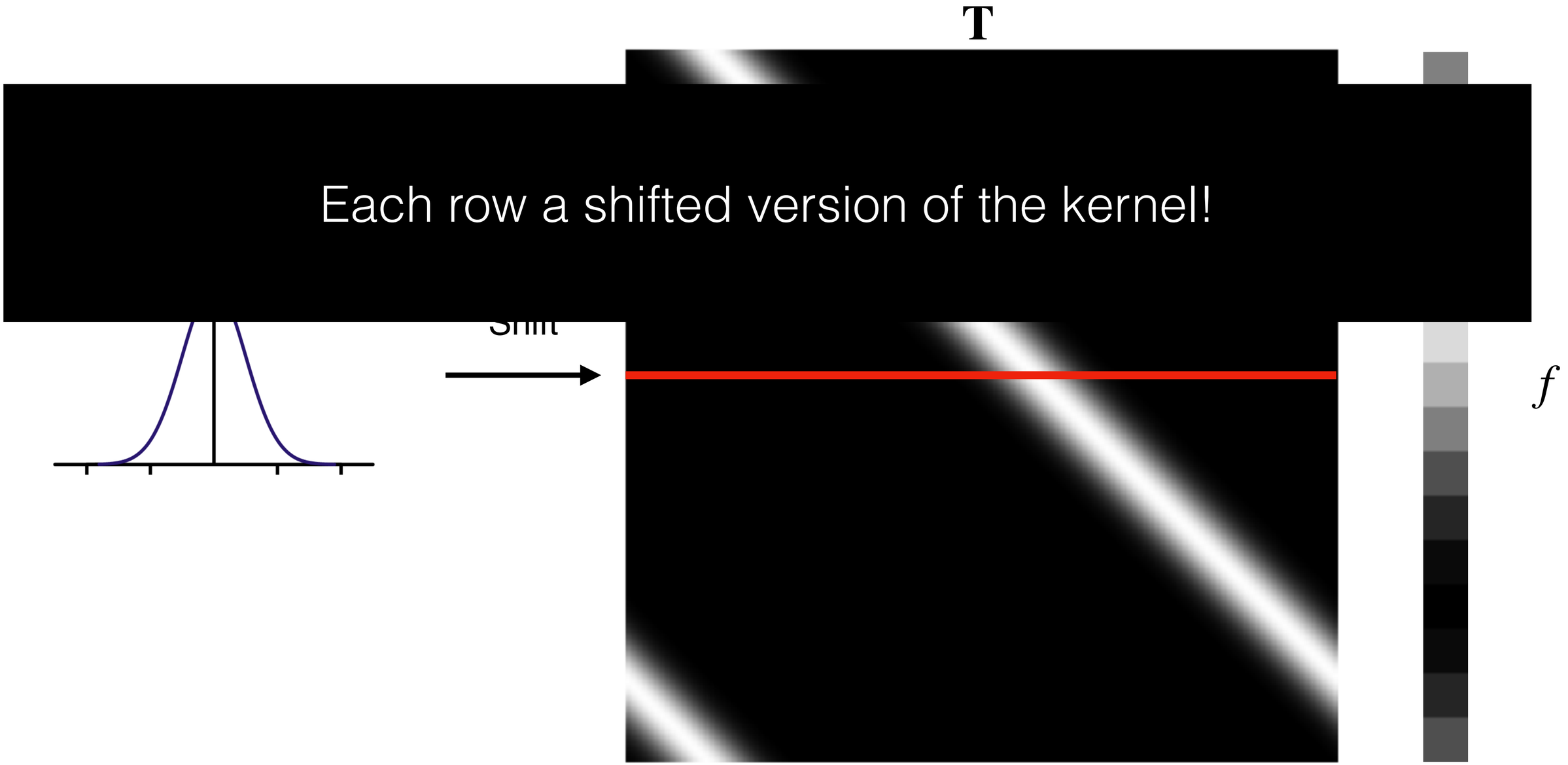
f

Convolution as a single matrix-vector product



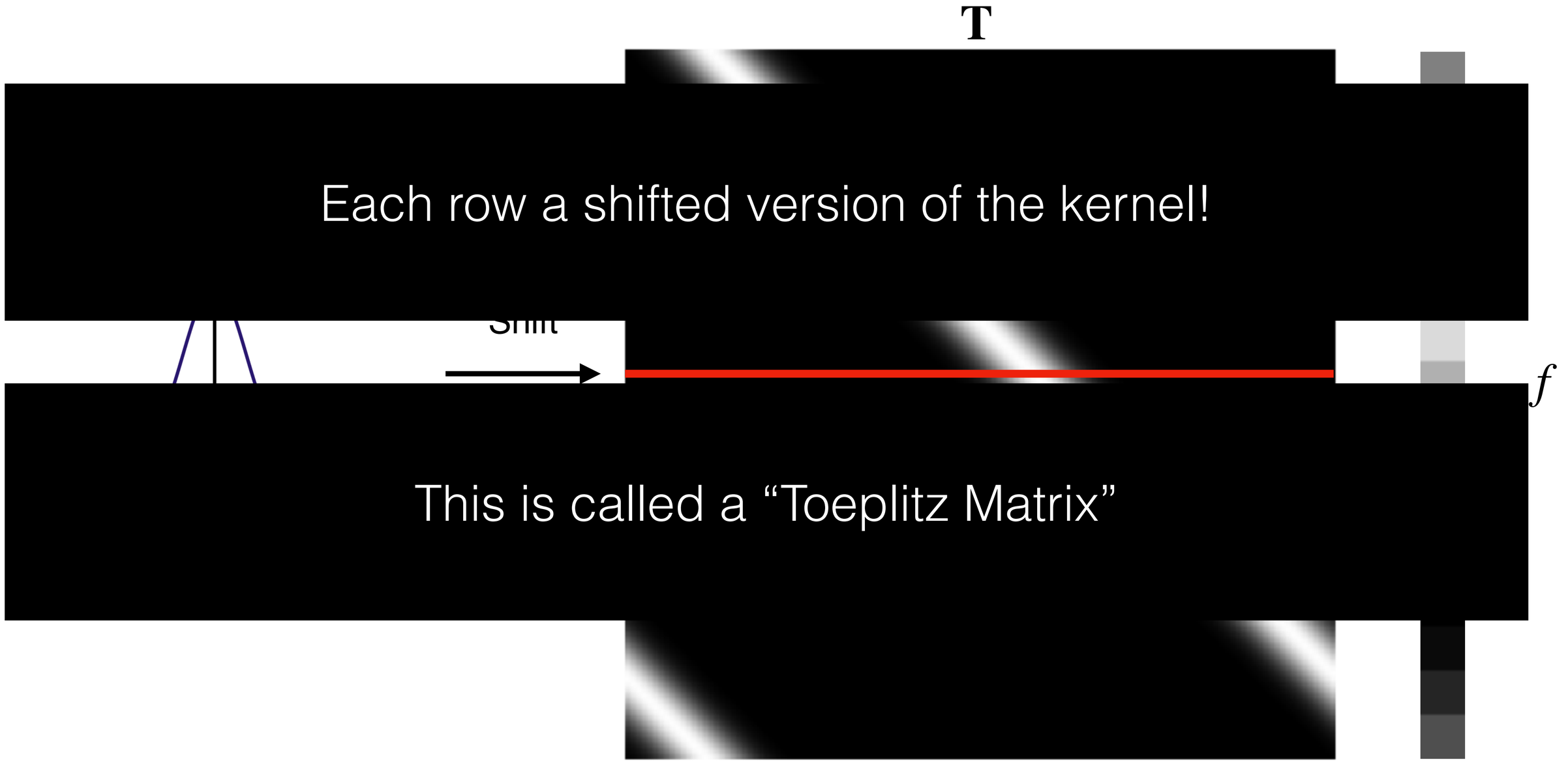
By Vierge Marie - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=3902337

Convolution as a single matrix-vector product



By Vierge Marie - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=3902337

Convolution as a single matrix-vector product



By Vierge Marie - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=3902337

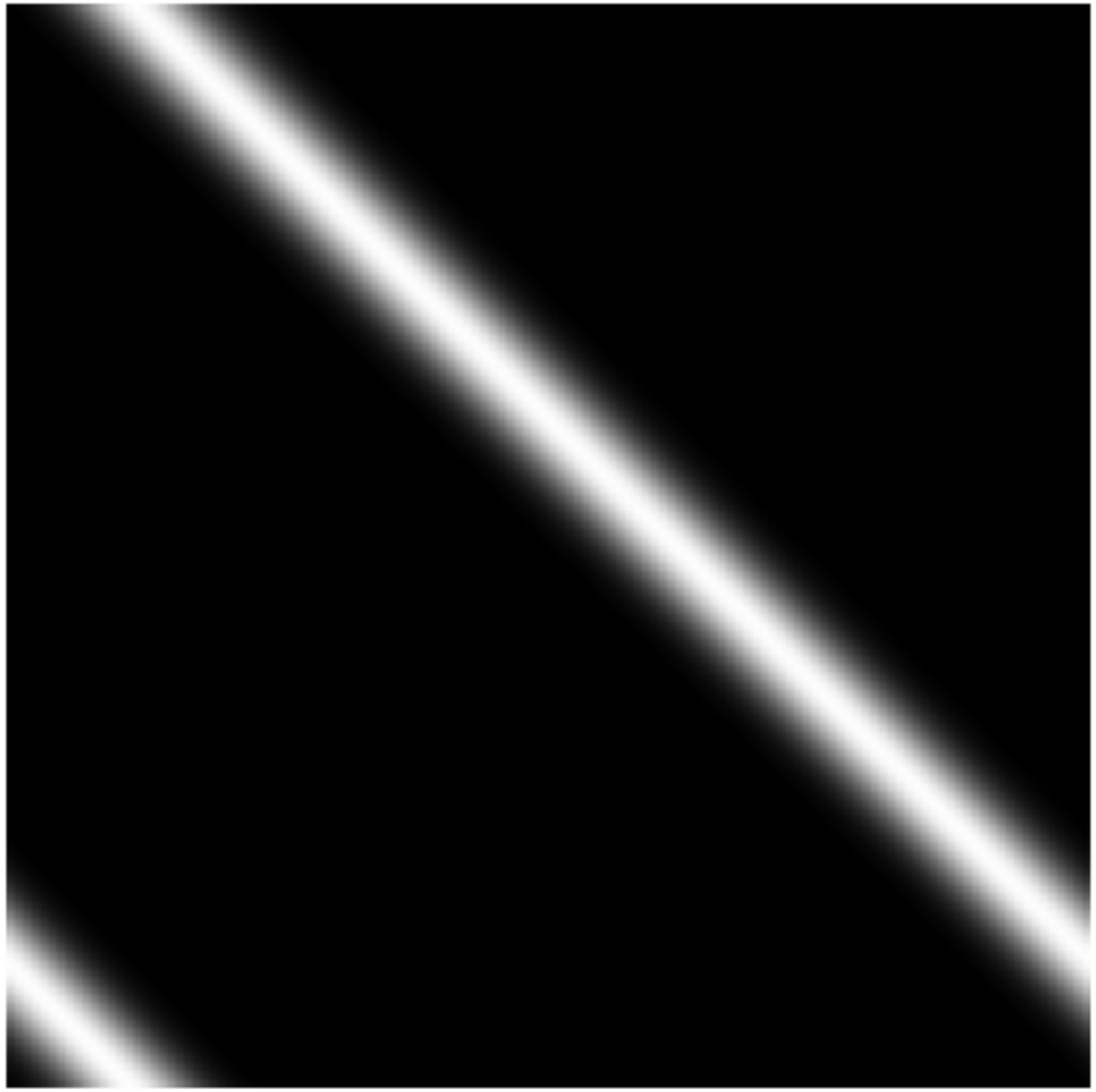
Convolution as a single matrix-vector product

Apply the Fourier change-of-basis to the convolution matrix:

$$\mathcal{F}\mathbf{T}\mathcal{F}^{-1}$$

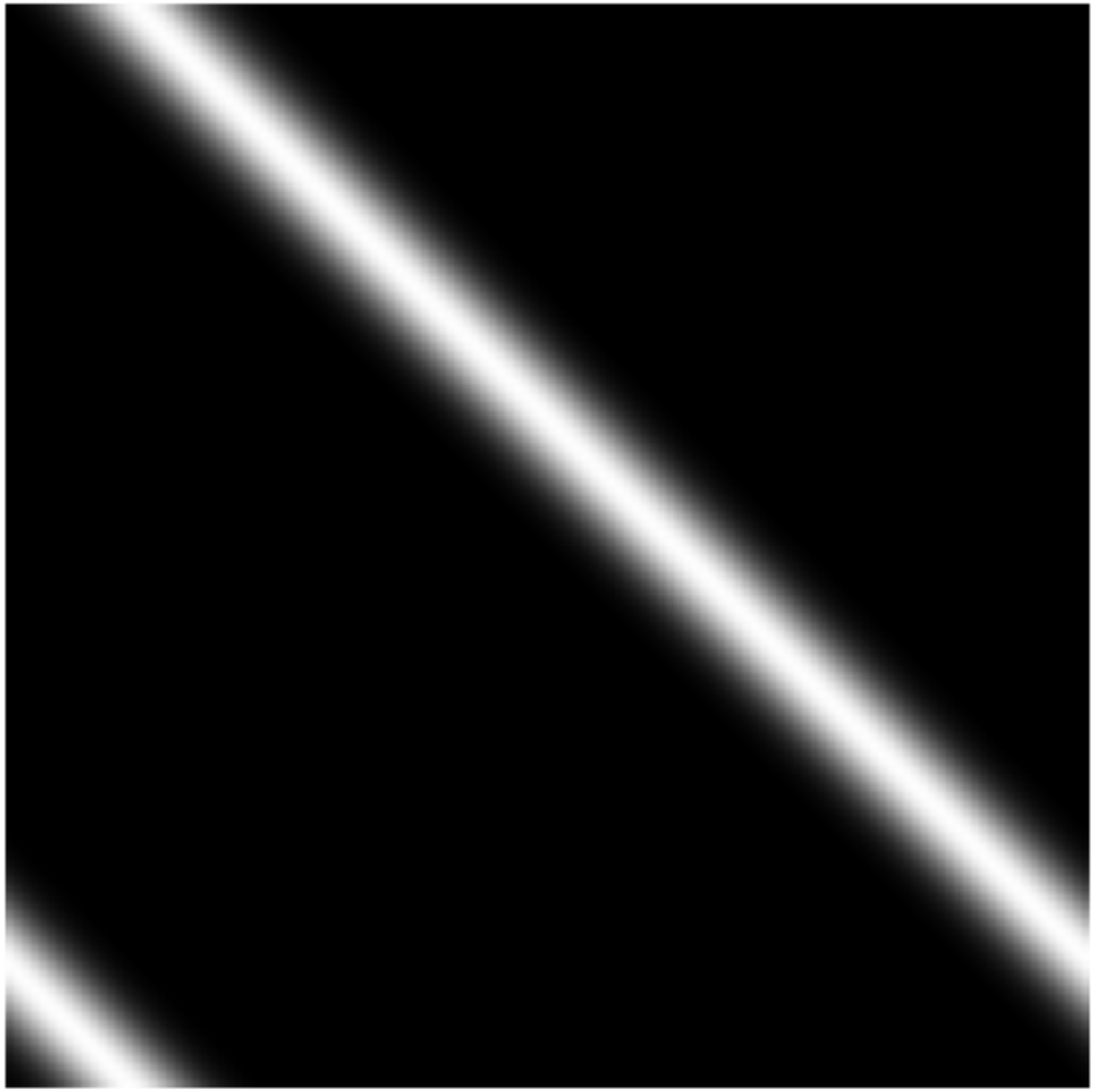
What happens?

$\mathbf{T}$

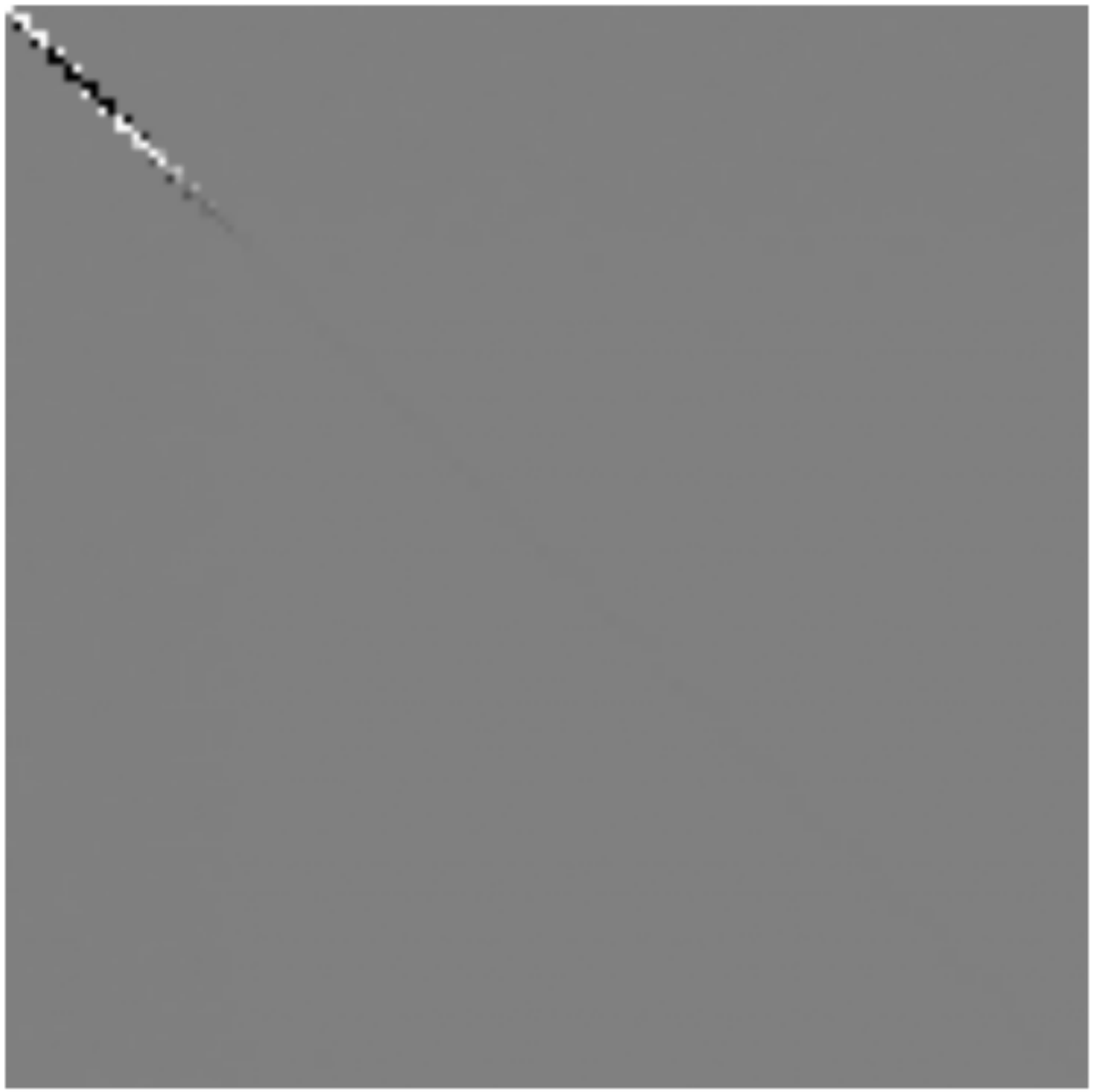


Blur Kernel in the Fourier Domain

$\mathbf{T}$

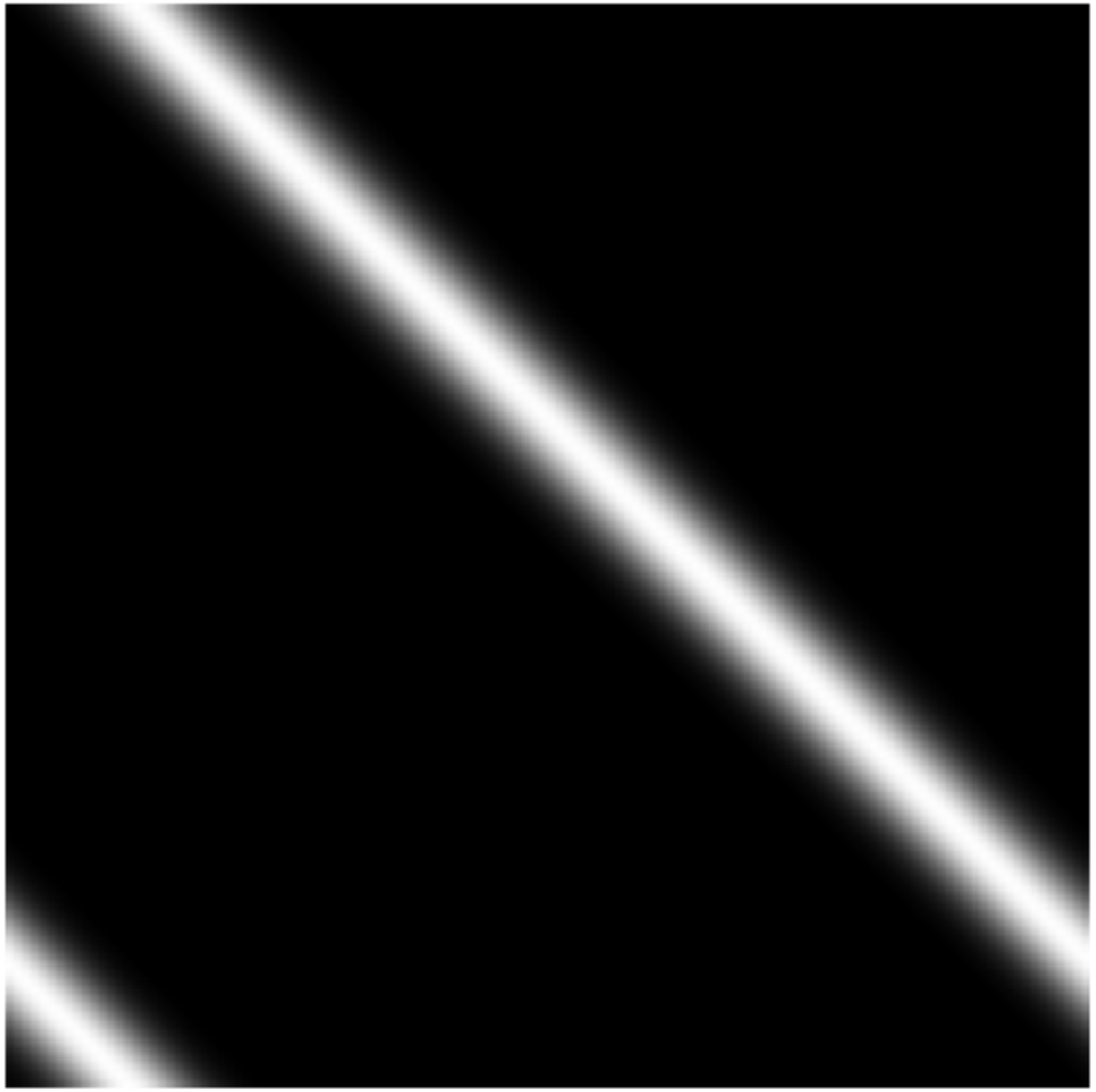


$\mathcal{F}\mathbf{T}\mathcal{F}^{-1}$

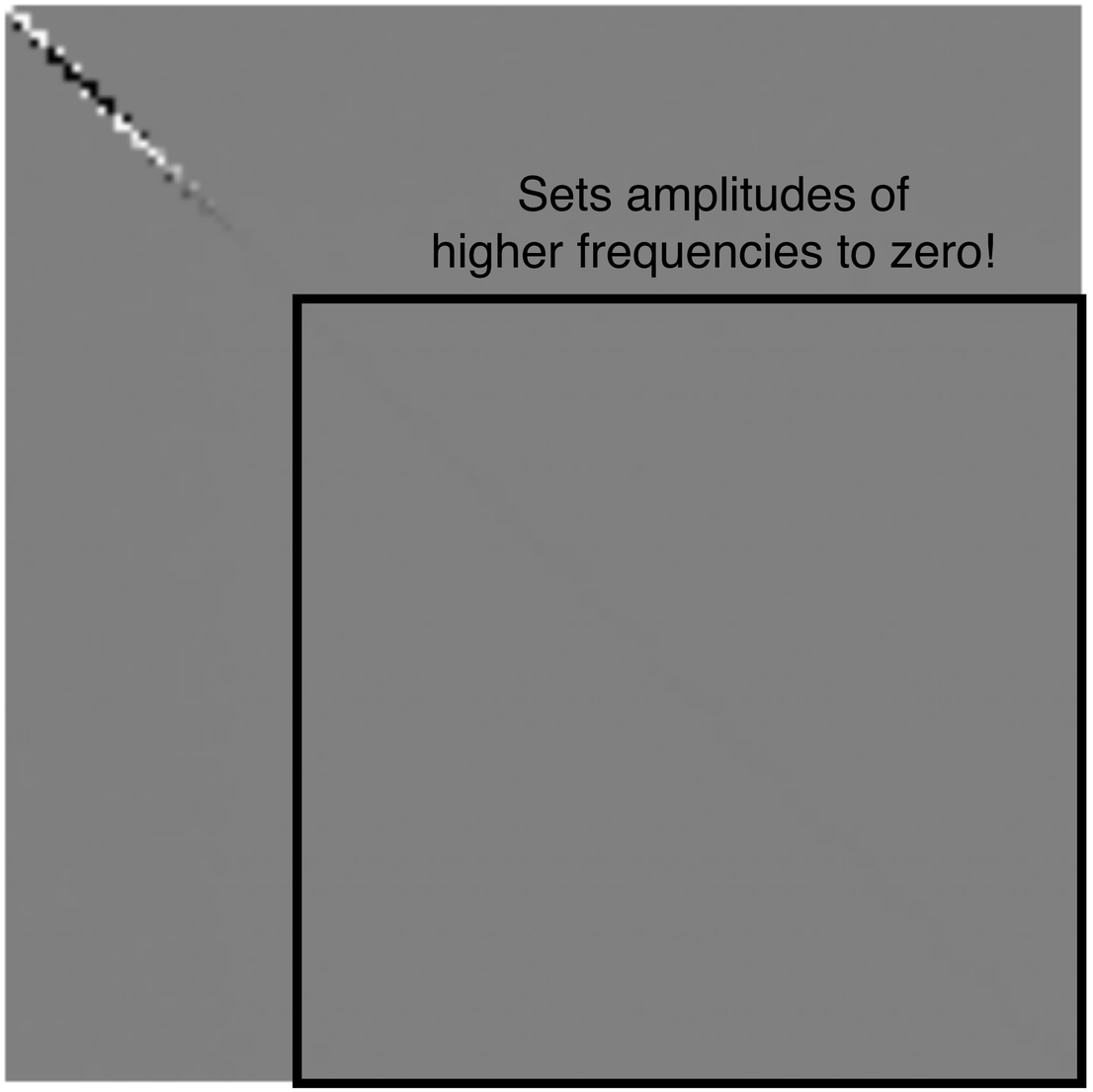


Blur Kernel in the Fourier Domain

$\mathbf{T}$



$\mathcal{F}\mathbf{T}\mathcal{F}^{-1}$

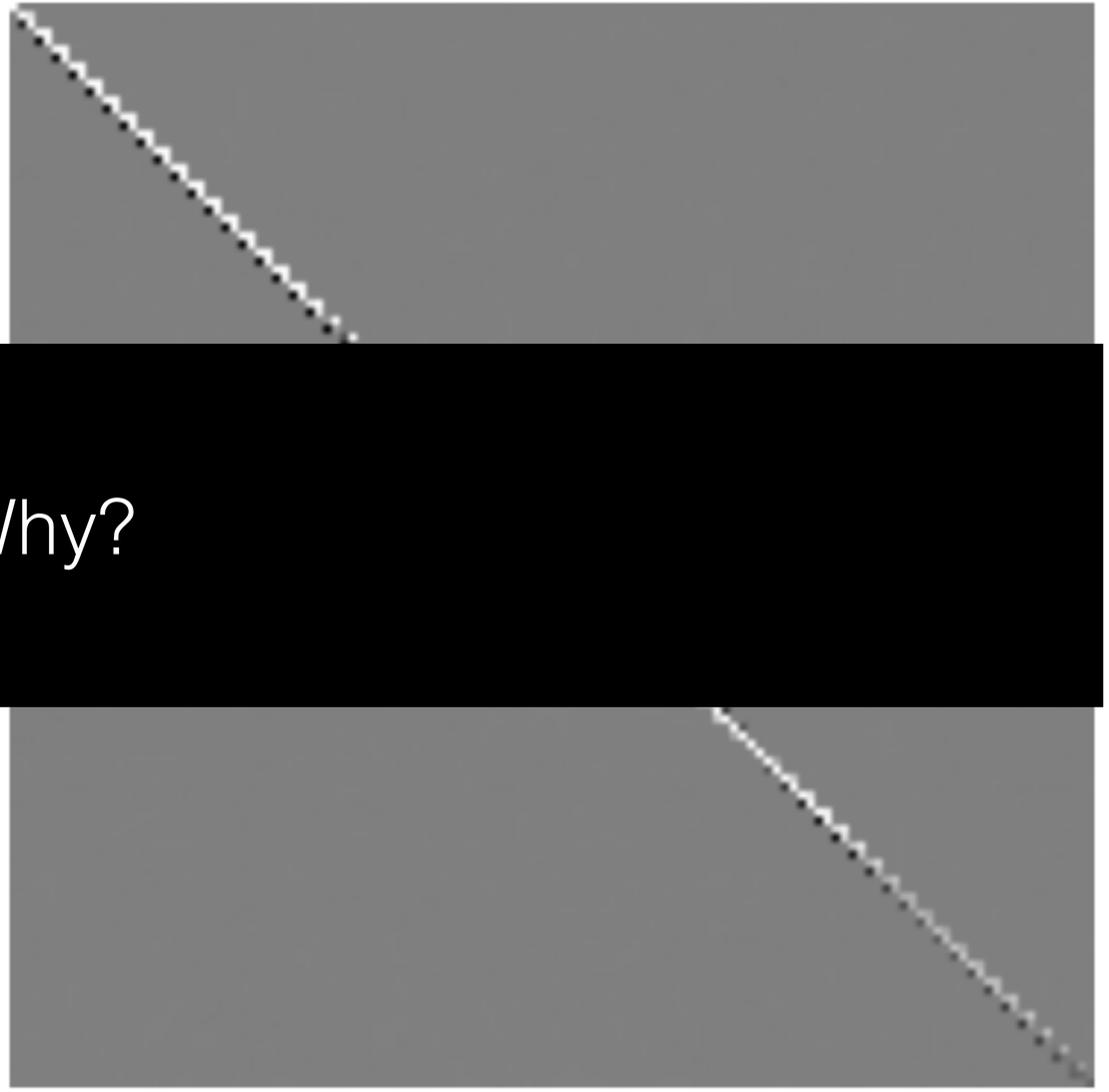


Every Convolution becomes (block) diagonal in Fourier Basis!

$\mathbf{T}$



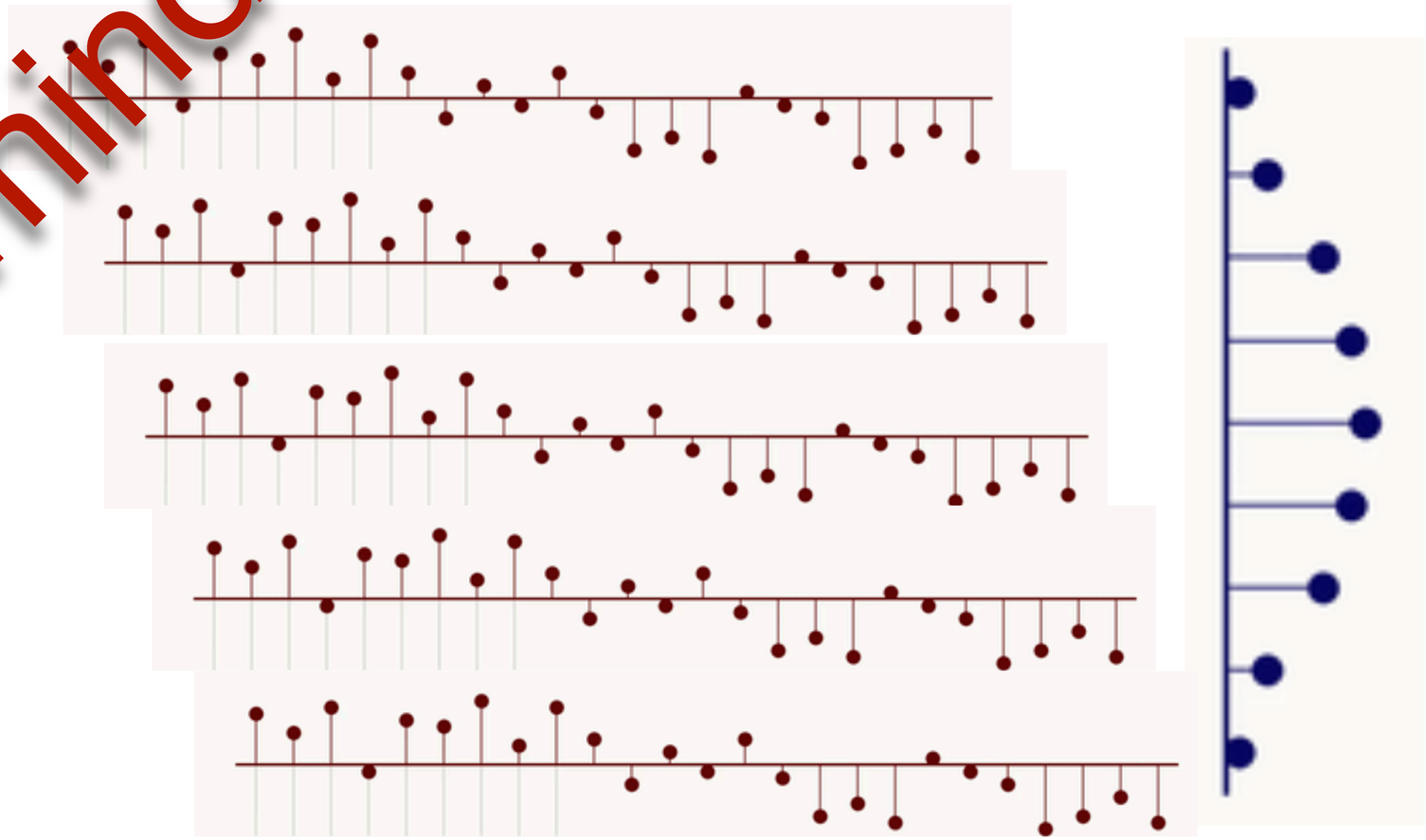
$\mathcal{F}\mathbf{T}\mathcal{F}^{-1}$



How? Why?

Alternative perspective: 1D Discrete Convolution

Reminder

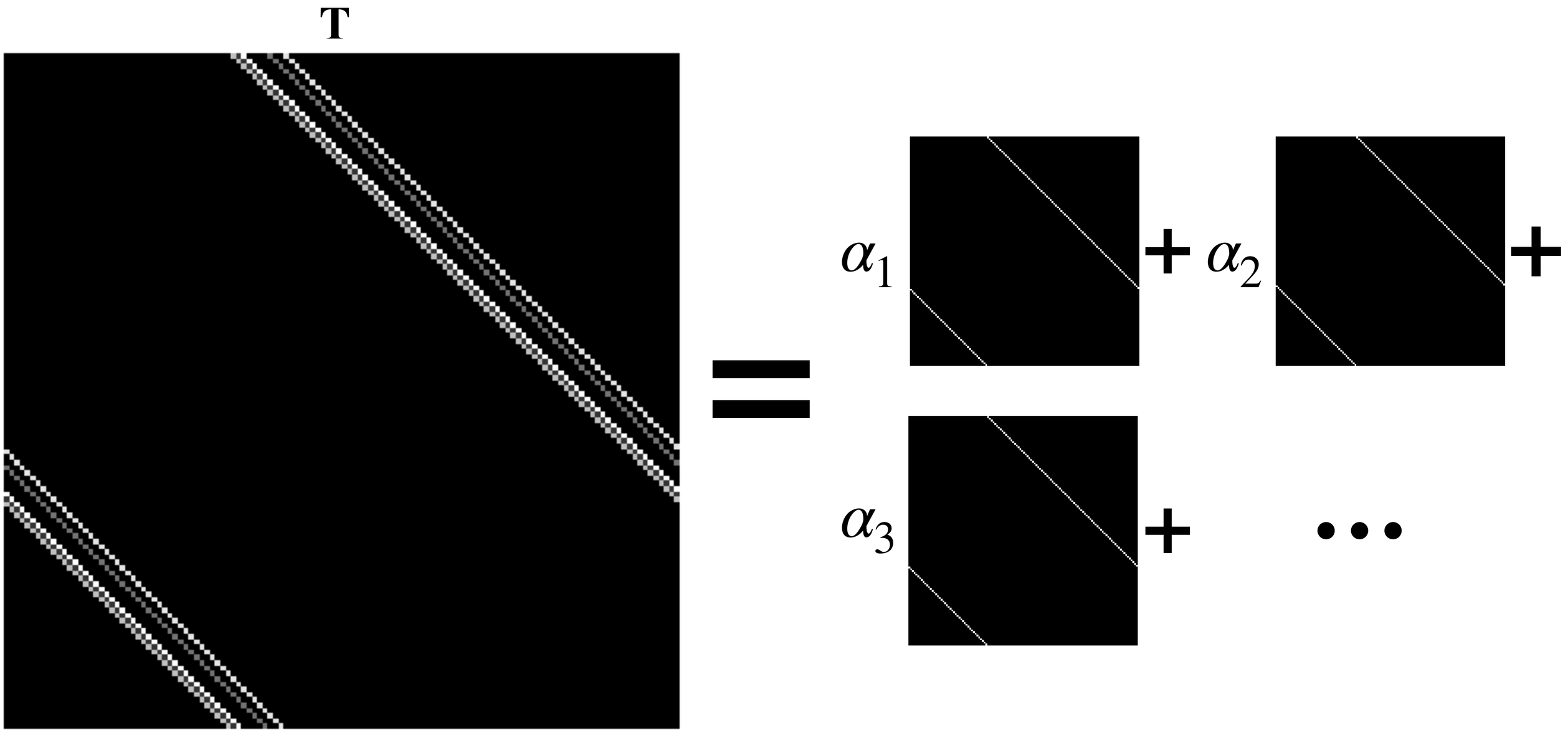


Source: Brandon Rohrer, https://e2eml.school/convolution_one_d.html

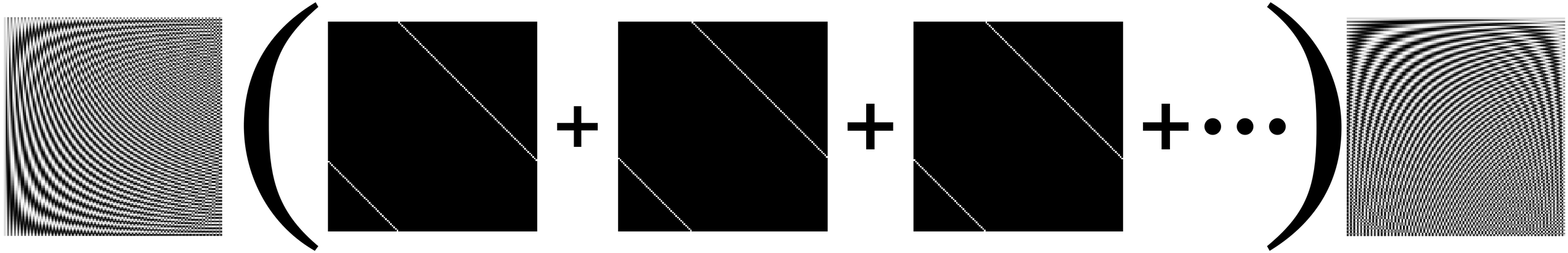
$$\sum_{shift\ x} k[x]T_x f$$

The convolution is a weighted sum of shifted versions of the image

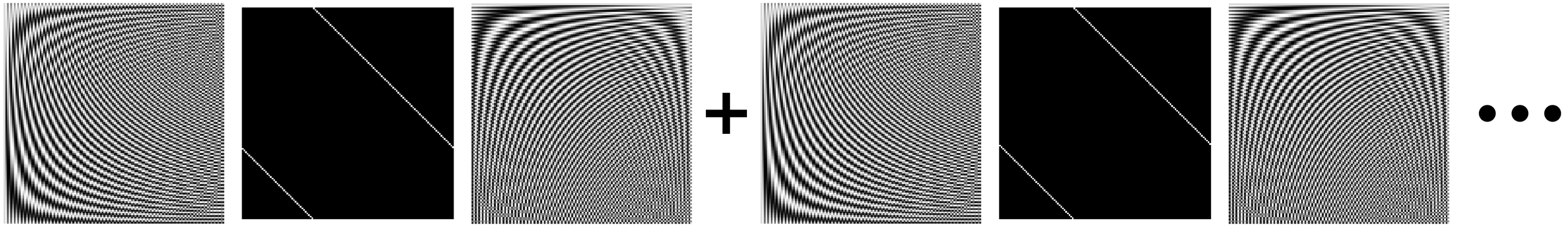
Convolution Matrices are weighted sums of circulant shift matrices!



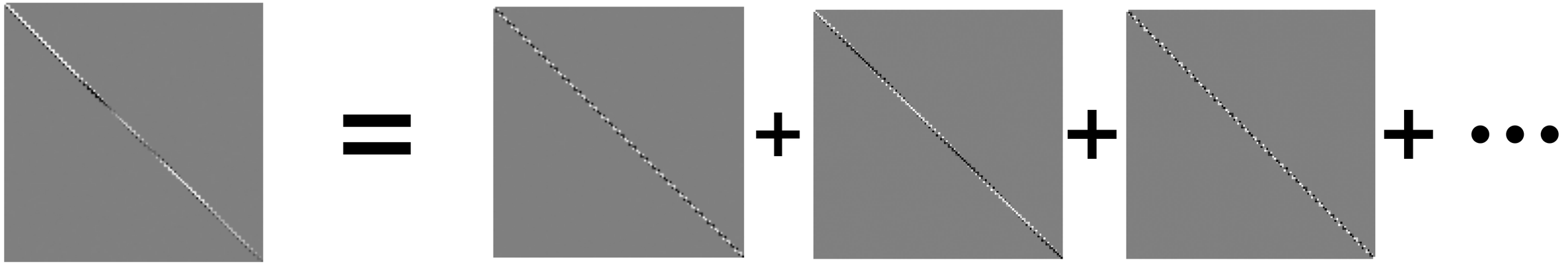
Each one of them is diagonalized by the FT matrix,
hence their sum is, too



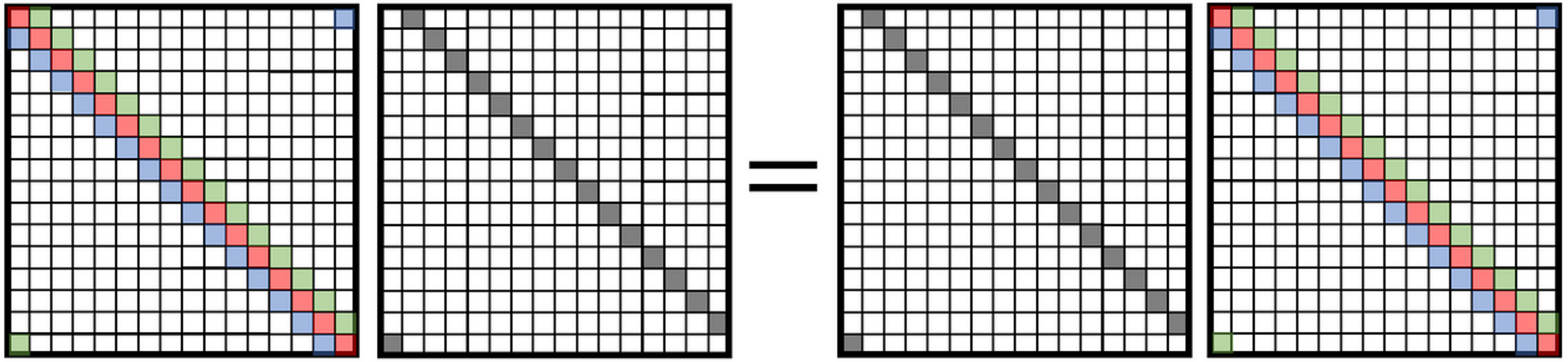
Each one of them is diagonalized by the FT matrix,
hence their sum is, too



Each one of them is diagonalized by the FT matrix,
hence their sum is, too



Important property: Convolutions commute with shifts!



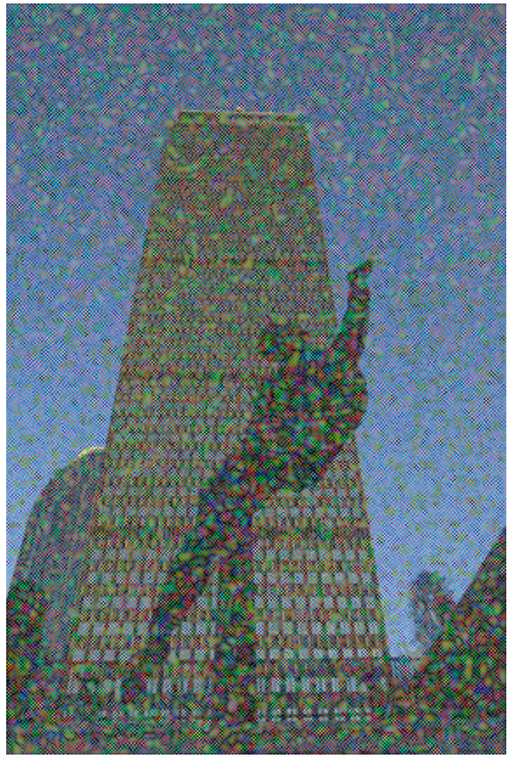
Source: Michael Bronstein - Deriving Convolutions from First Principles - https://medium.com/towards-data-science/deriving-convolution-from-first-principles-4ff124888028

Fourier & deblurring

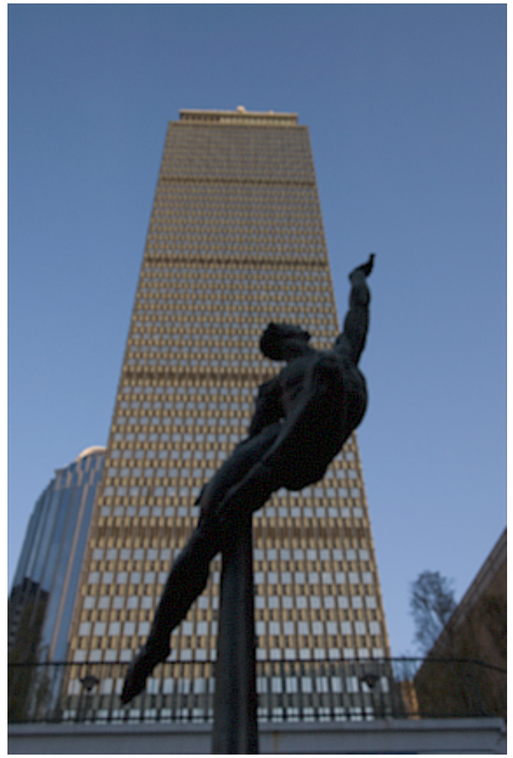
- Convolution is diagonal in Fourier
- Inverse (deblurring) is just a division by eigenvalue
  - The smaller the eigenvalue, the worse the conditioning



Blurry input

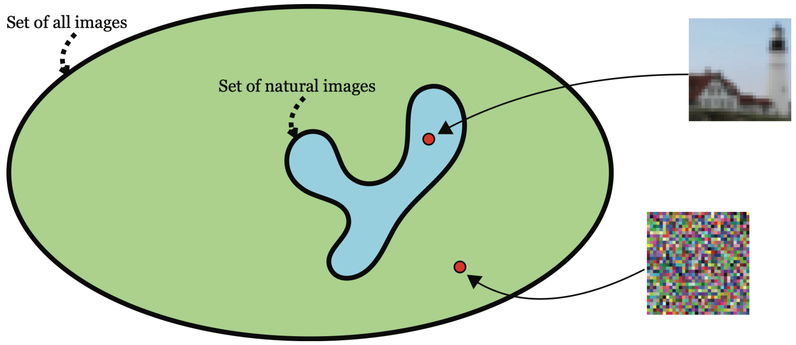


Result without regularization



Result with regularization

PCA and Fourier

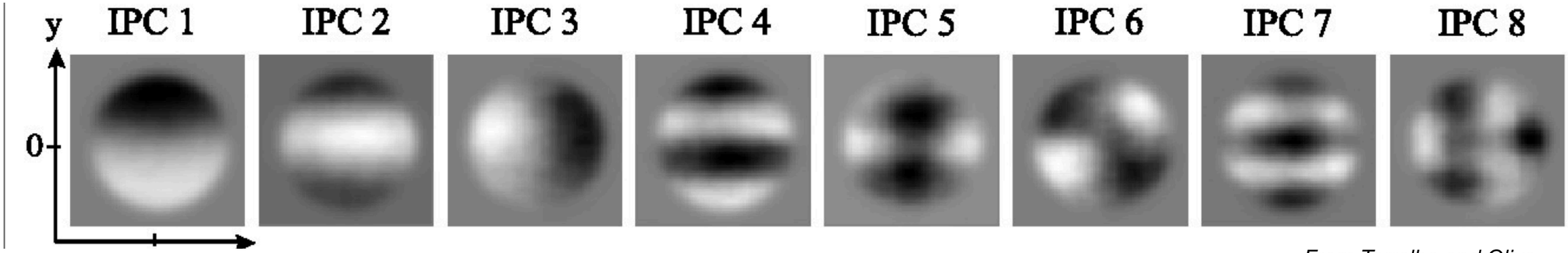


From Foundations of Computer Vision

- Applying PCA to a stationary signal (~shift invariant) essentially** yields the Fourier basis**
- e.g. if you do PCA on a large set of “natural”** images, they admit the Fourier bases as principal components

Natural means images of the world, whether it’s nature or man-made, as opposed to random signals

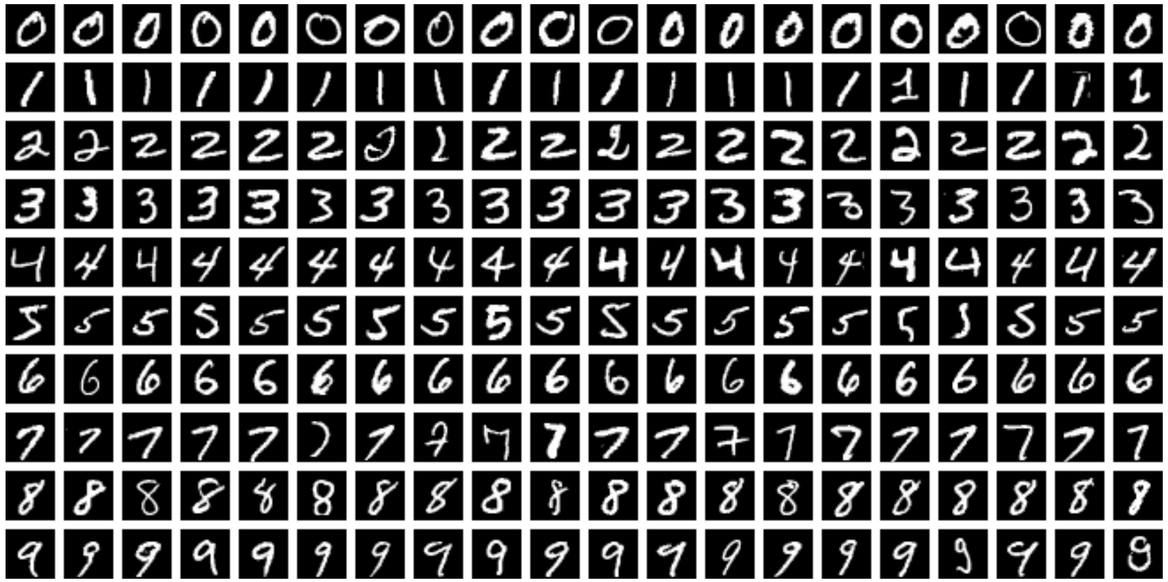
- Don’t be surprised when Fourier-like filters “emerge”



From Torralba and Oliva

Note: some datasets are not shift-invariant

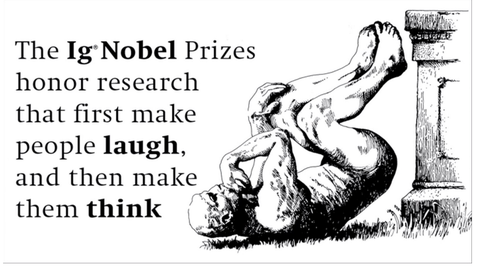
- What you learn from applying linear tools to them may not generalize to shift-invariant datasets





A bit of a tangent but…

- https://improbable.com/2020/06/05/fmri-brain-research-the-dead-salmon-has-lots-of-company/
- https://www.scientificamerican.com/blog/scicurious-brain/ignobel-prize-in-neuroscience-the-dead-salmon-study/
- With the extreme dimensionality of functional neuroimaging data comes extreme risk for false positives. Across the 130,000 voxels in a typical fMRI volume the probability of a false positive is almost certain. Correction for multiple comparisons should be completed with these datasets, but is often ignored by investigators. To illustrate the magnitude of the problem we carried out a real experiment that demonstrates the danger of not correcting for chance properly.
- Winner of 2012 Ig Nobel prize



- Lessons for AI interpretability research: https://arxiv.org/abs/2512.18792

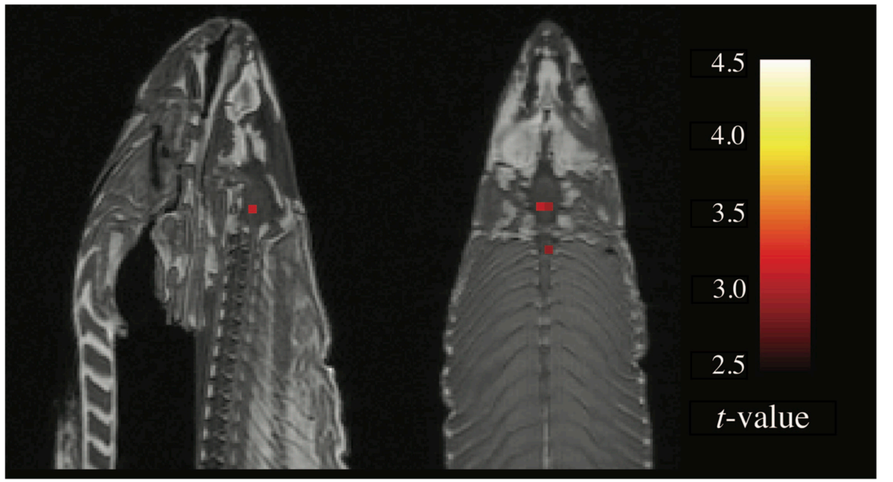


Fig. 1. Sagittal and axial images of significant brain voxels in the task > rest contrast. The parameters for this comparis were $t(131) > 3.15$, $p(\text{uncorrected}) < 0.001$, 3 voxel extent threshold. Two clusters were observed in the salmon centr nervous system. One cluster was observed in the medial brain cavity and another was observed in the upper spinal column.

Journal of Serendipitous and Unexpected Results

Neural Correlates of Interspecies Perspective Taking in the Post-Mortem Atlantic Salmon: An Argument For Proper Multiple Comparisons Correction

Craig M. Bennett [1*], Abigail A. Baird [2], Michael B. Miller [1] and George L. Wolford [3]

[1]Department of Psychology, University of California at Santa Barbara, Santa Barbara, CA 93106
[2]Department of Psychology, Blodgett Hall, Vassar College, Poughkeepsie, NY 12604
[3]Department of Psychological and Brain Sciences, Moore Hall, Dartmouth College, Hanover, NH 03755

THE DEAD SALMONS OF AI INTERPRETABILITY



Maxime Méloux[◇], **Giada Dirupo**[♣], **François Portet**[◇], **Maxime Peyrard**[◇]

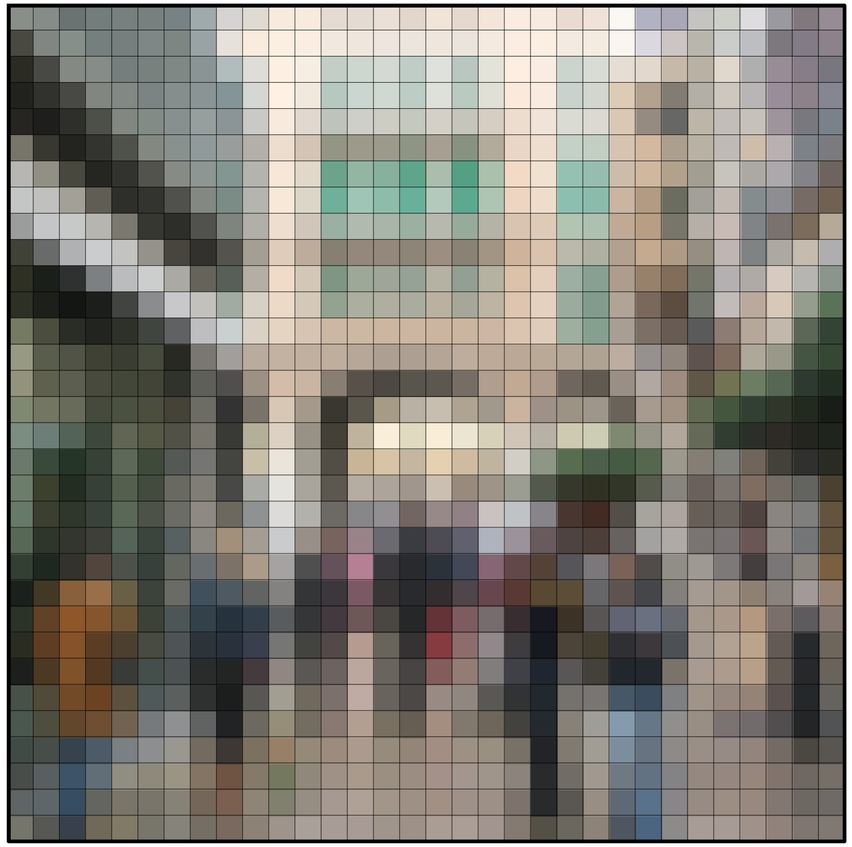
[◇]Université Grenoble Alpes, CNRS, Grenoble INP, LIG
[♣]Icahn School of Medicine at Mount Sinai

{melouxm, peyrardm}@univ-grenoble-alpes.fr

Sampling and Reconstruction



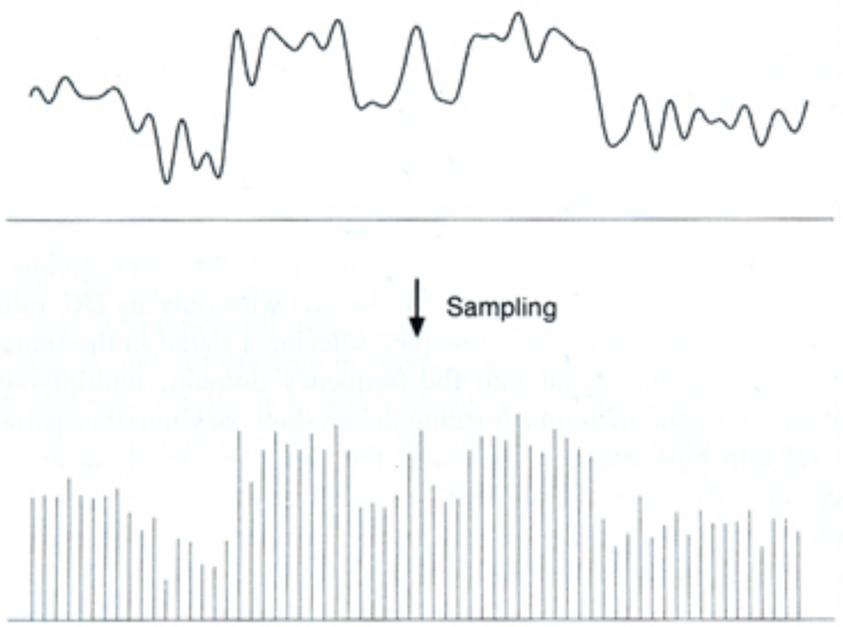
Discrete Sampling



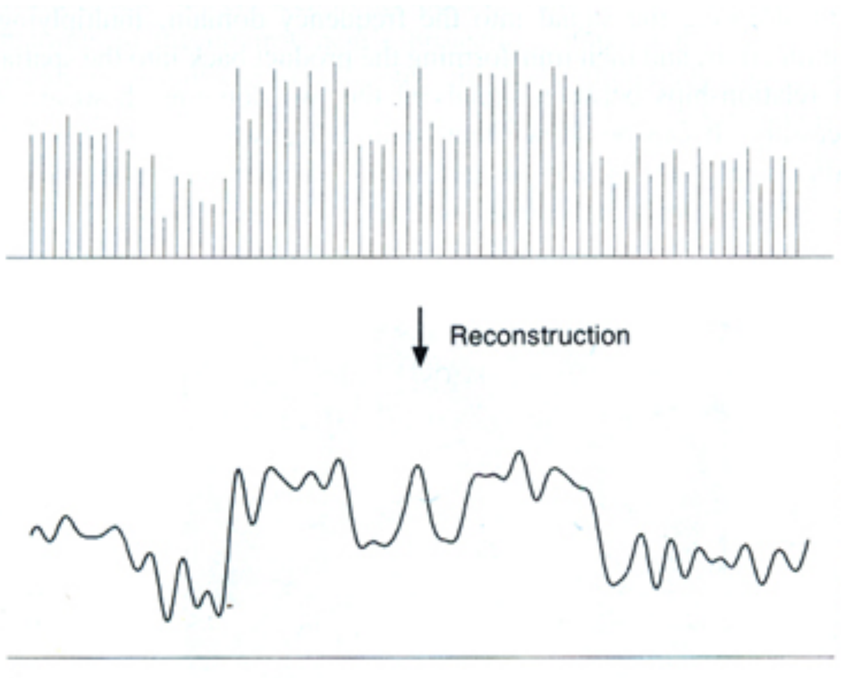
$\mathbf{f}:\mathbb{R}^2 \to \mathbb{R}^3$ $\mathbf{f}(\mathbf{x})=\mathbf{c}$

Sampling and reconstruction

Sampling: recording the function's values at a discrete set of locations



Reconstruction: converting a sampled representation back into a continuous function by "guessing" what happens between the samples

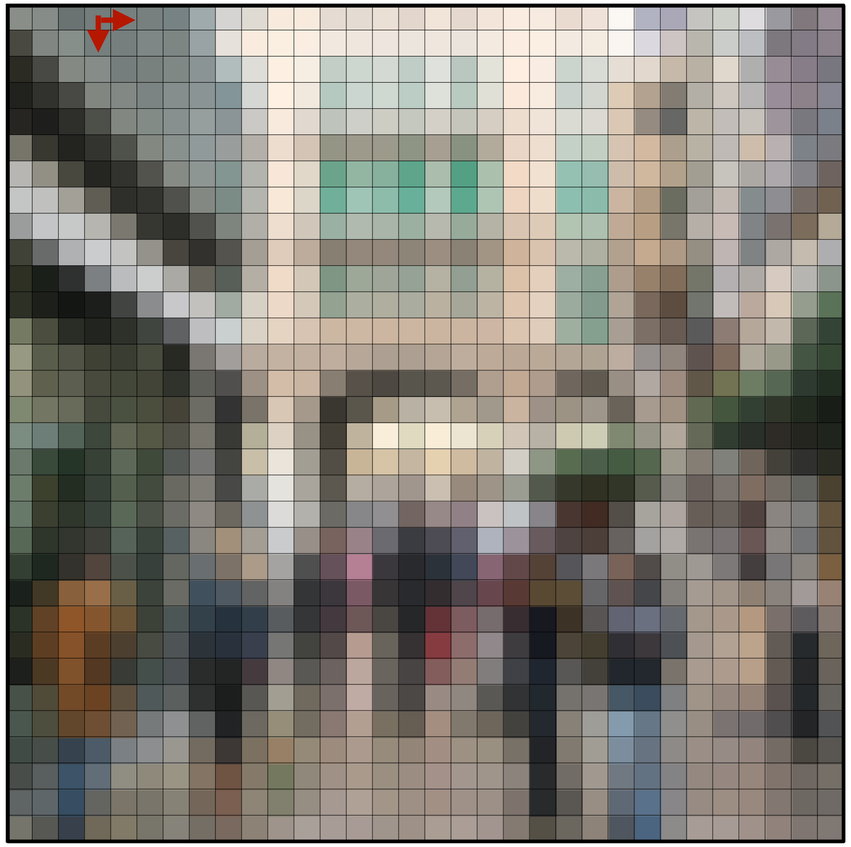


Source: S. Marschner (via A. Efros)

Why Fourier is good to model sampling

Sampling grids involve a lot of shifts (quotient space if you want to be fancy)

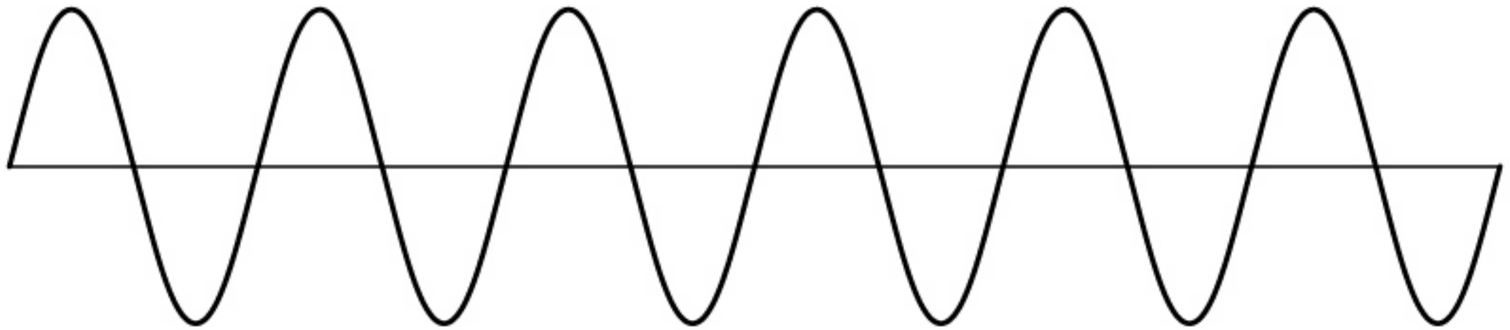




Reconstruction is usually a convolution if discrete samples by a reconstruction kernel (bilinear, bicubic, etc.)

Sampling and reconstruction

Simple example: a sine wave



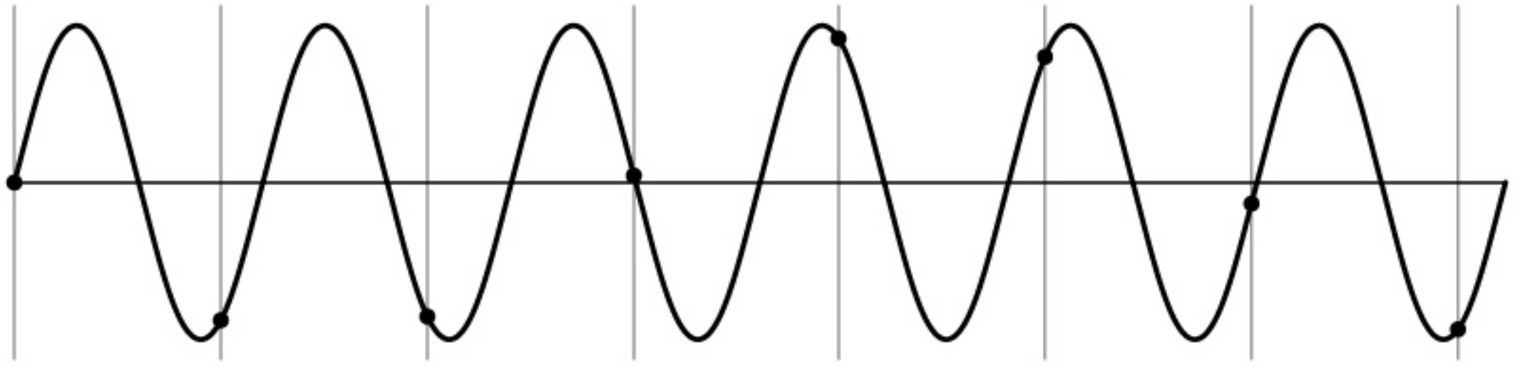
Source: S. Marschner (via A. Efros)

Sampling and reconstruction

Simple example: a sine wave

What if we “missed” things between the samples?

Unsurprising result: information is lost



Source: S. Marschner (via A. Efros)

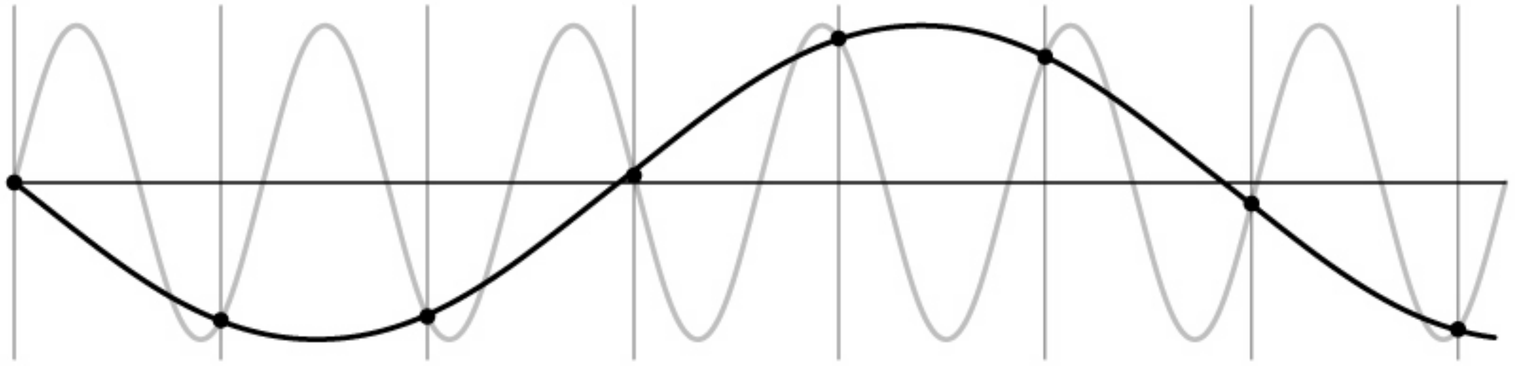
Sampling and reconstruction

Simple example: a sine wave

What if we “missed” things between the samples?

Unsurprising result: information is lost

Surprising result: indistinguishable from lower frequencies



Source: S. Marschner (via A. Efros)

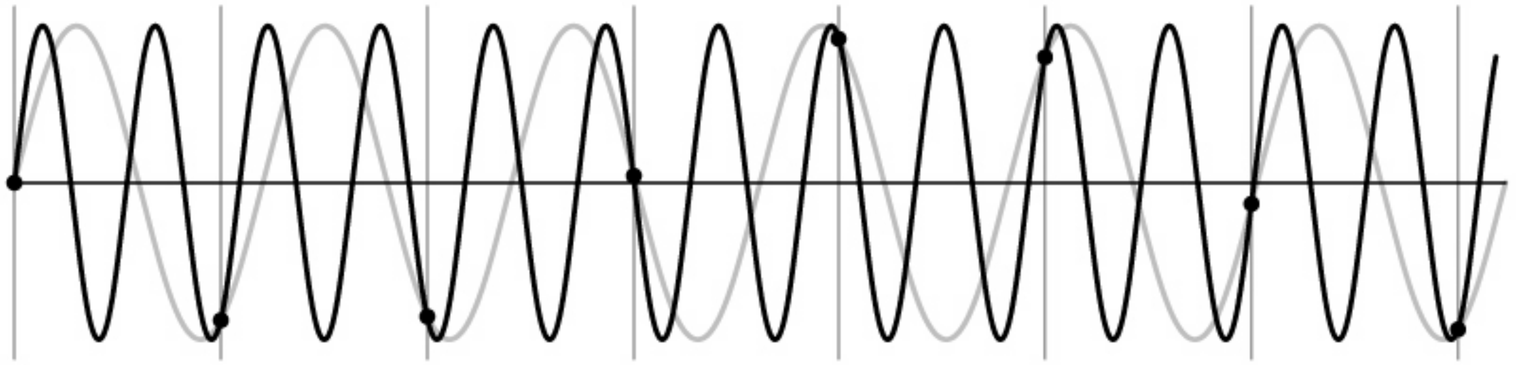
Sampling and reconstruction

Simple example: a sine wave

What if we “missed” things between the samples?

Unsurprising result: information is lost

Surprising result: indistinguishable from lower frequencies
(or even higher frequencies)



Source: S. Marschner (via A. Efros)

Sampling and reconstruction

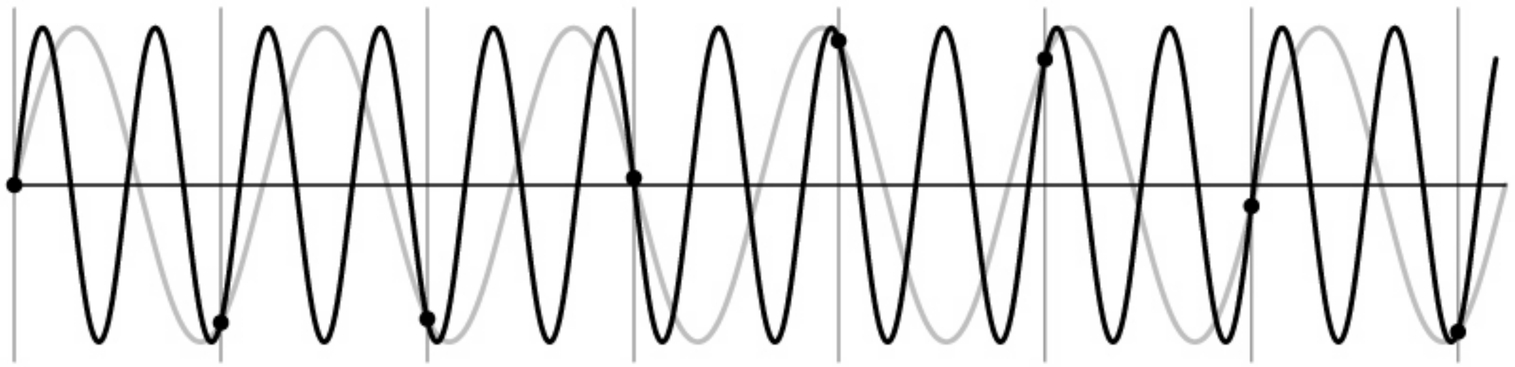
Simple example: a sine wave

What if we “missed” things between the samples?

Unsurprising result: information is lost

Surprising result: indistinguishable from lower frequencies (or even higher frequencies)

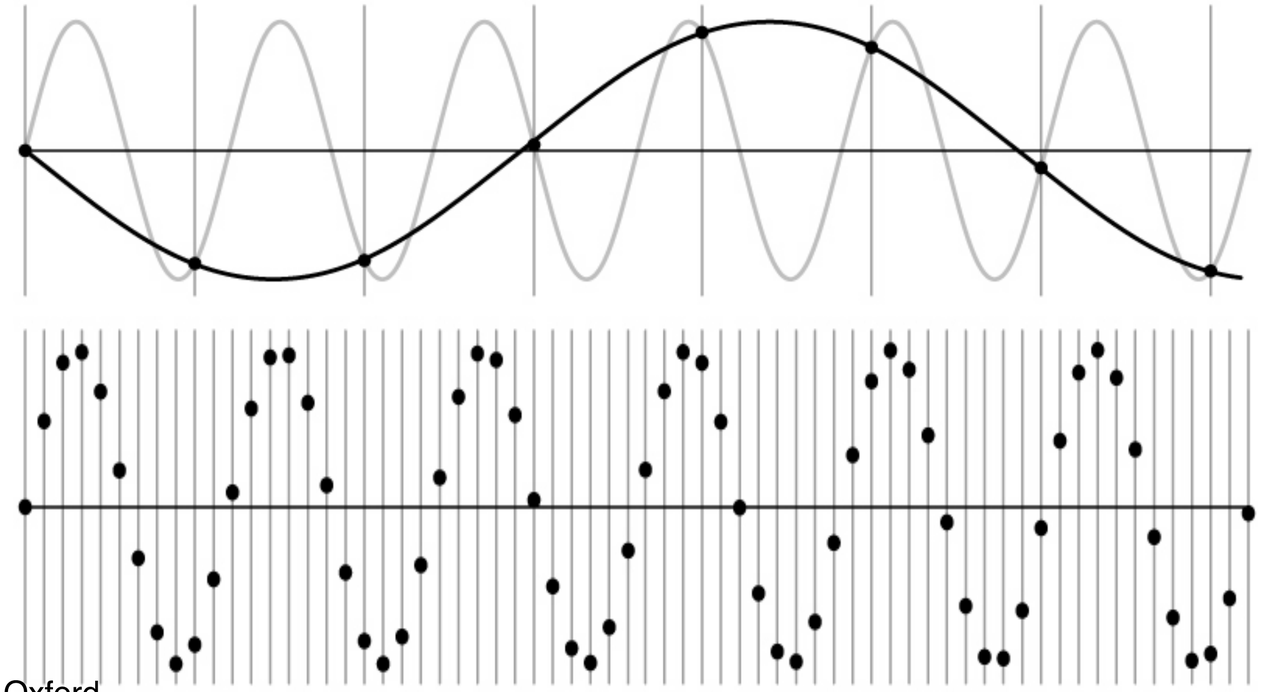
Aliasing: signal “traveling in disguise” as other frequencies



Source: S. Marschner (via A. Efros)

Nyquist-Shannon sampling theorem

When sampling a signal at discrete intervals, the sampling frequency must be at least *twice* the maximum frequency of the input signal to allow us to reconstruct the original perfectly from the sampled version



bad

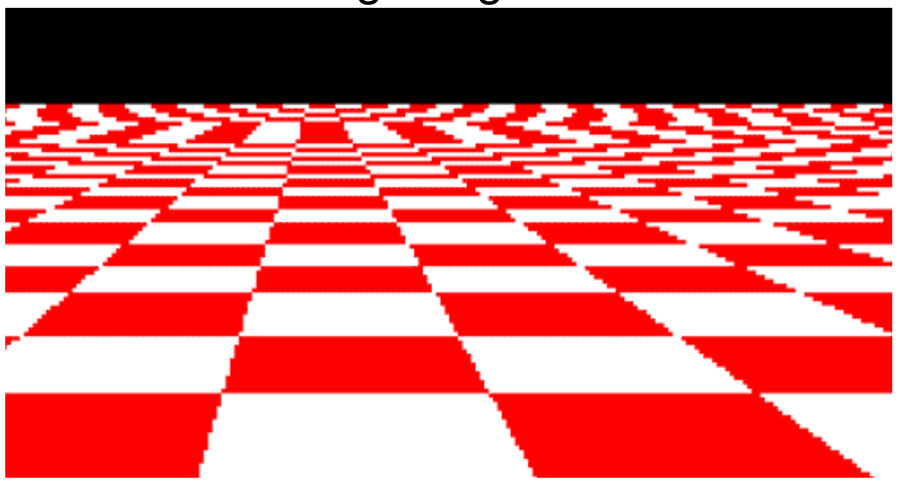
good

https://en.wikipedia.org/wiki/Nyquist-Shannon_sampling_theorem

Slide credit: Christian Rupprecht, Oxford

Aliasing “in the wild”

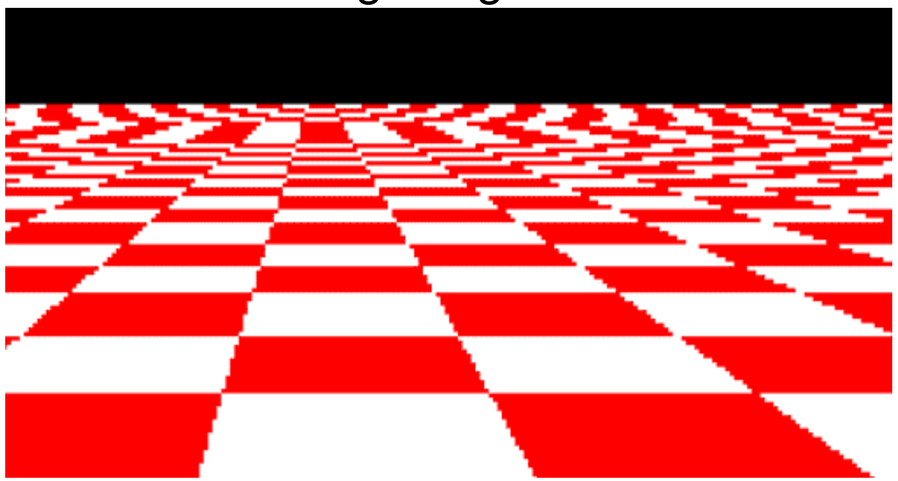
Disintegrating textures

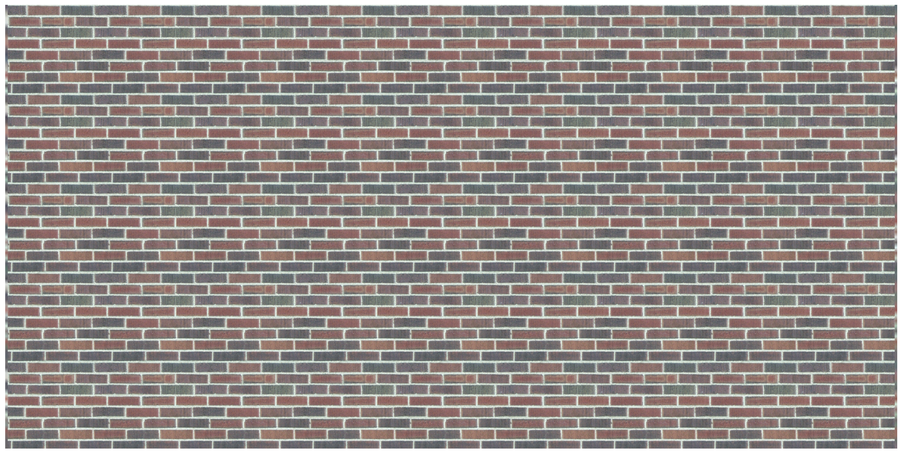


Slide credit: Christian Rupprecht, Oxford

Aliasing “in the wild”

Disintegrating textures



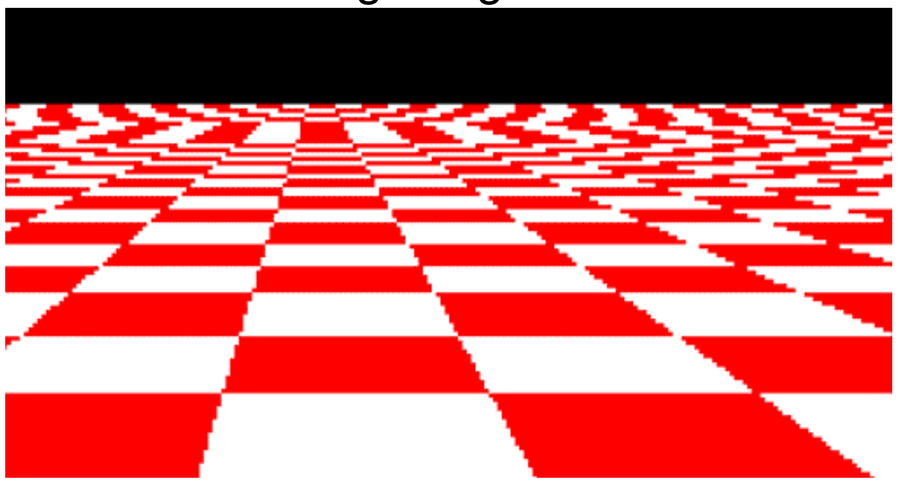


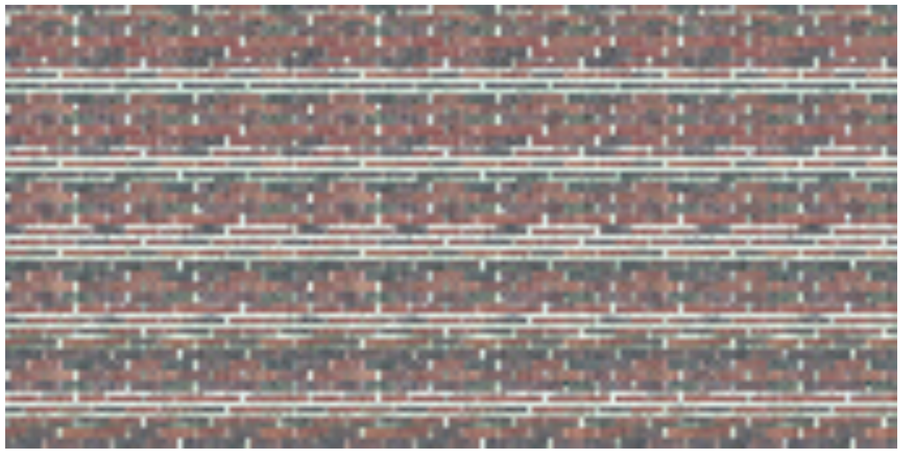
Slide credit: Christian Rupprecht, Oxford

Source

Aliasing “in the wild”

Disintegrating textures





Slide credit: Christian Rupprecht, Oxford

Source

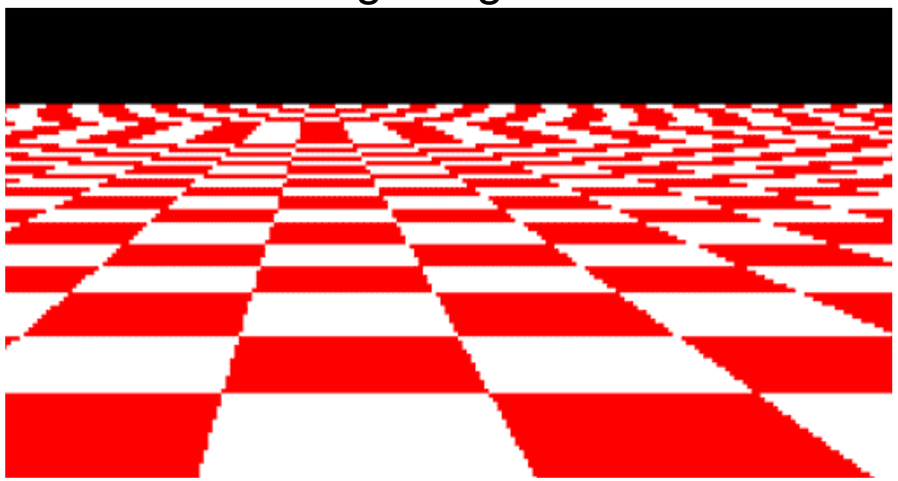
Moiré patterns, false color

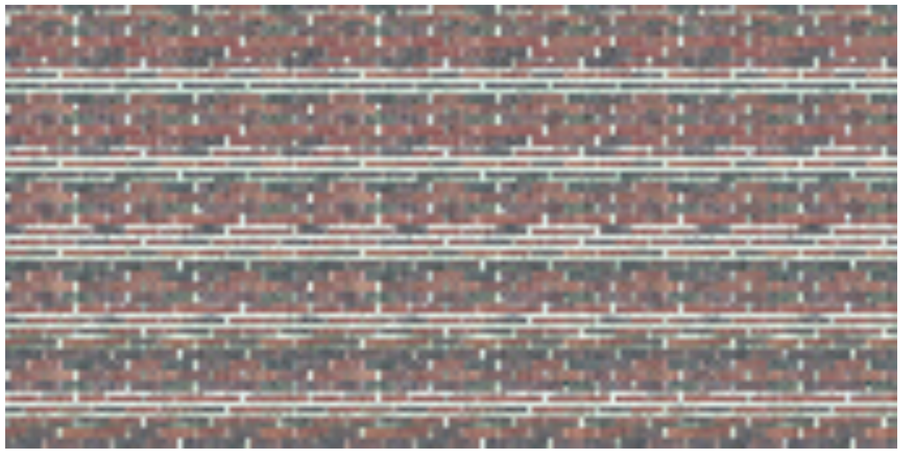


Source

Aliasing “in the wild”

Disintegrating textures





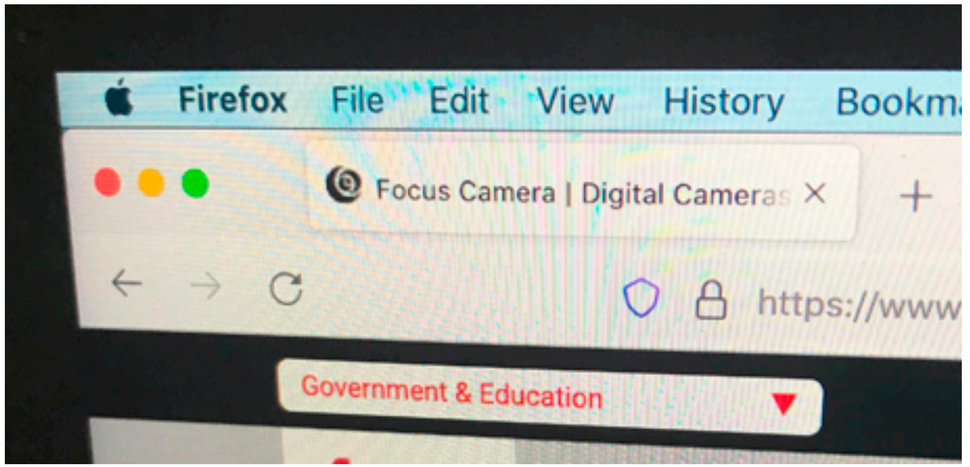
Slide credit: Christian Rupprecht, Oxford

Source

Moiré patterns, false color



Source



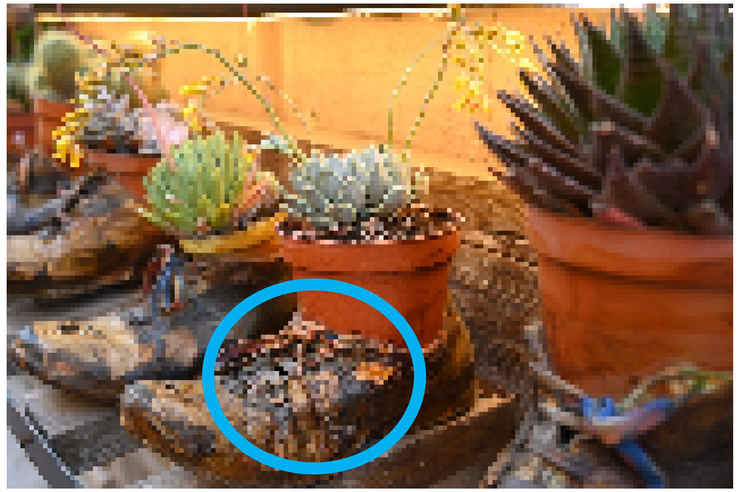
Source

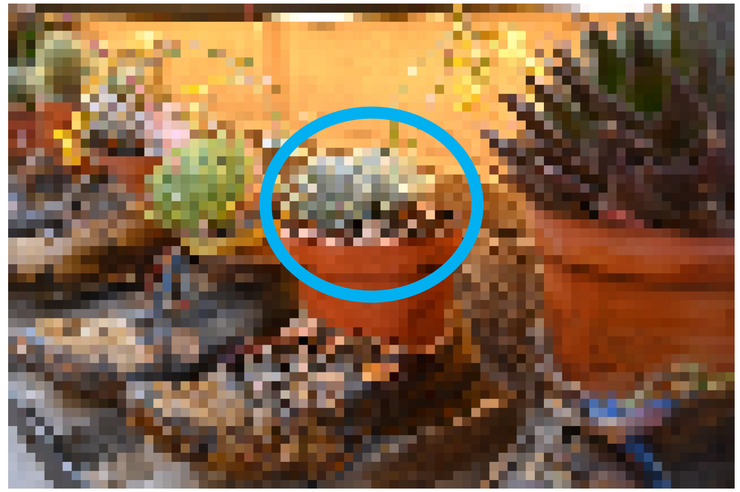
Subsampling Images

factor 4 | factor 8 | factor 16

naive





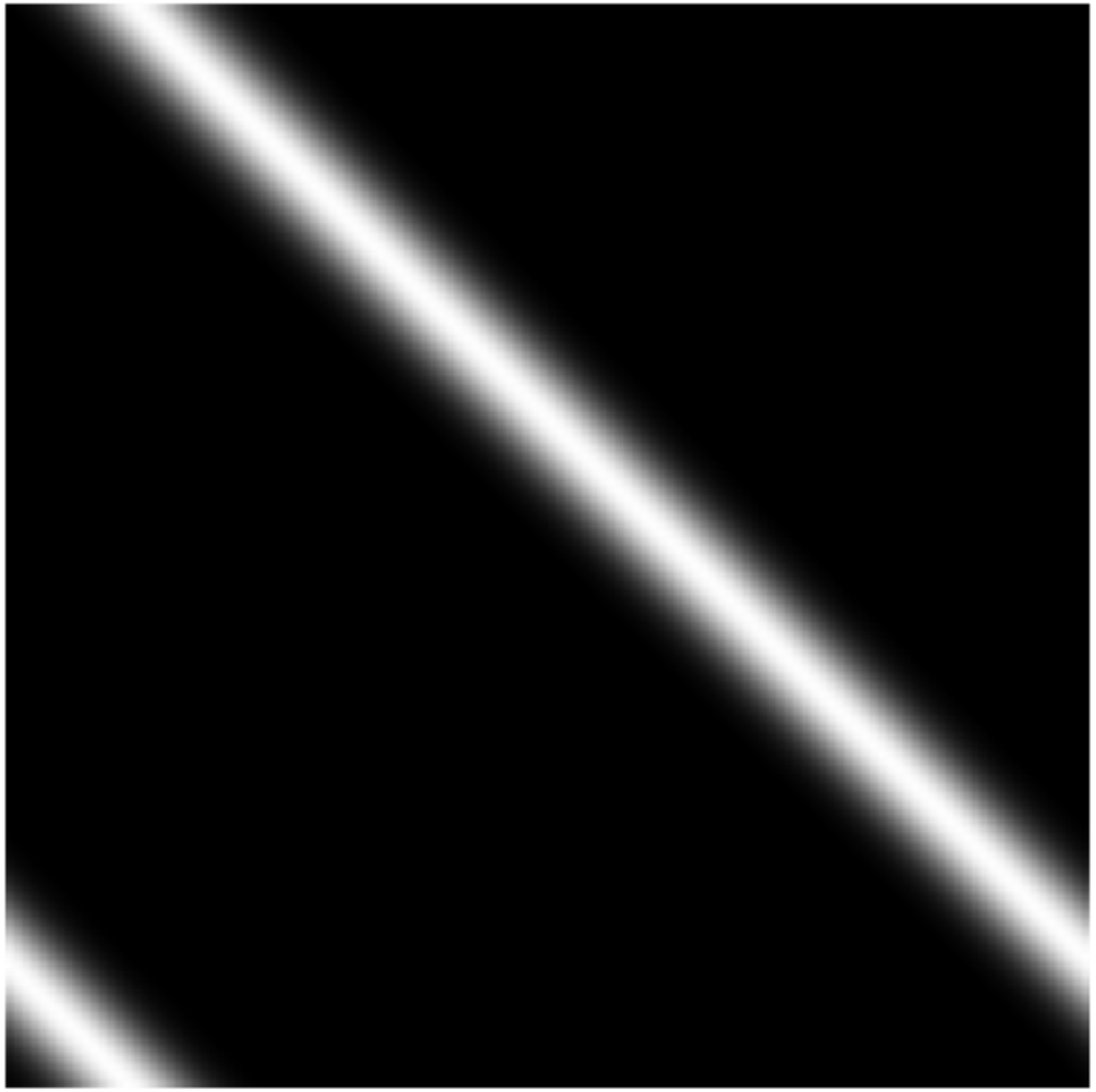


Slide credit: Christian Rupprecht, Oxford

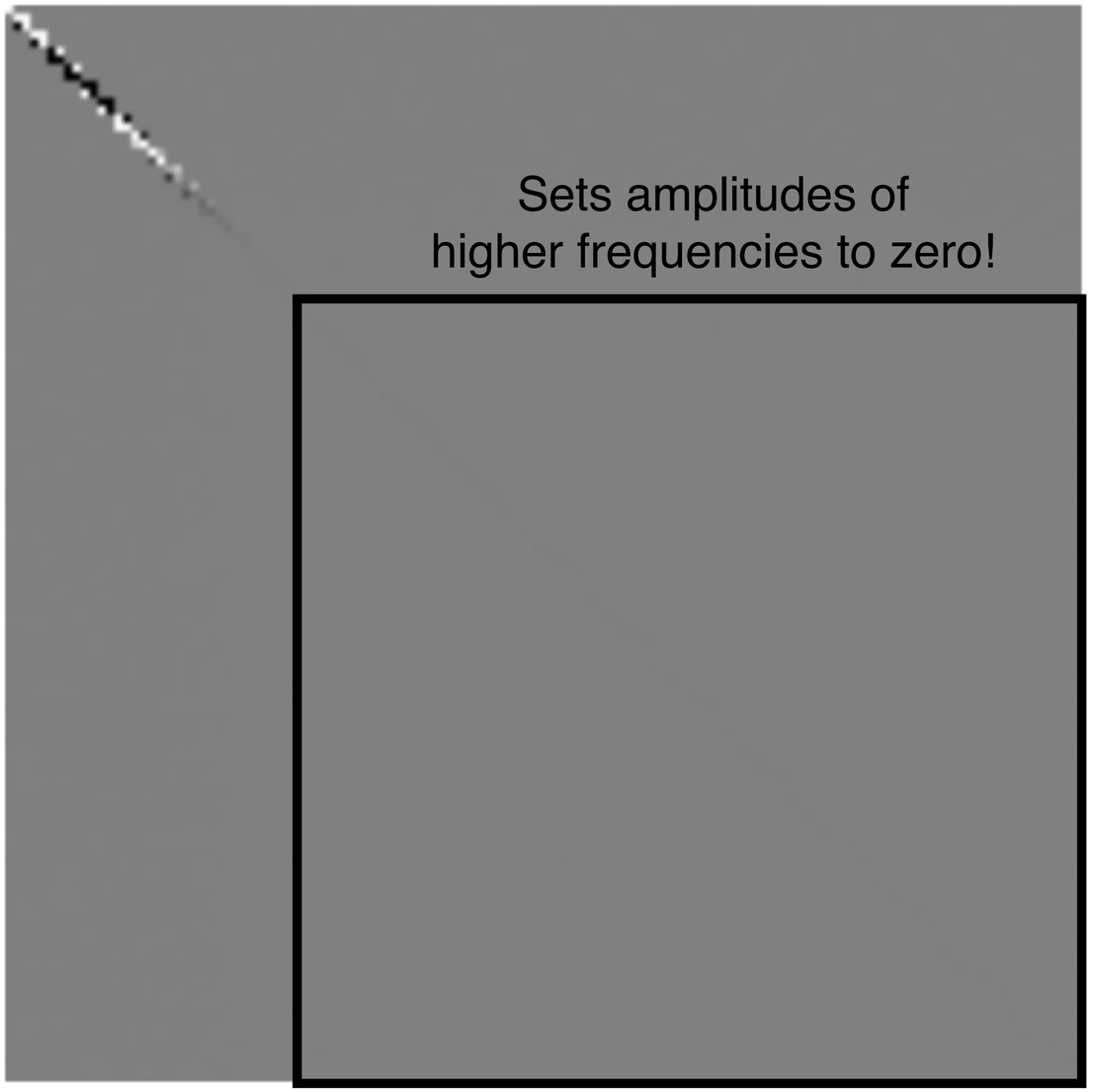
Demo time !

Solution: “Kill” high frequencies before downsampling!

$\mathbf{T}$



$\mathscr{F}\mathbf{T}\mathscr{F}^{-1}$

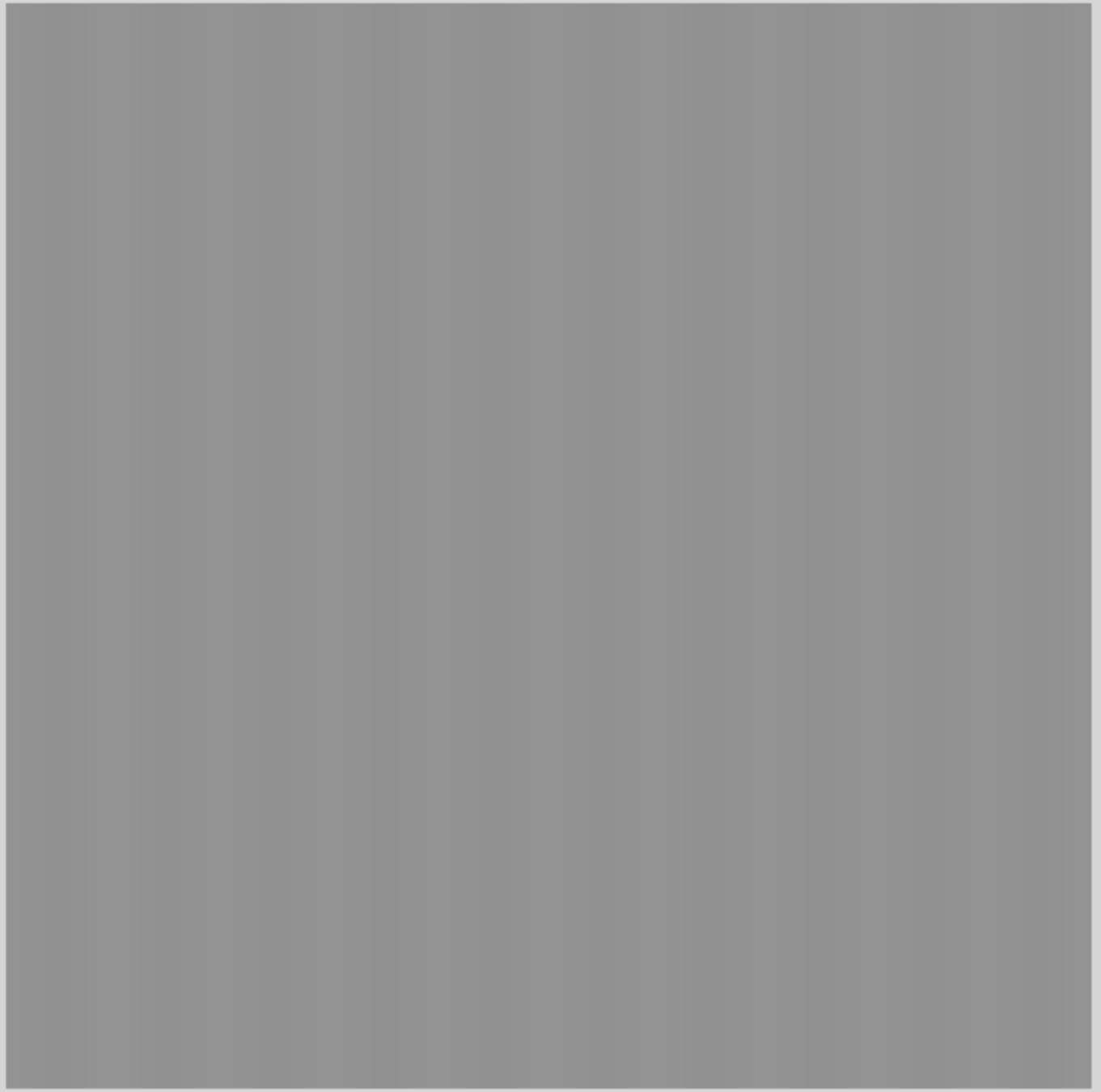


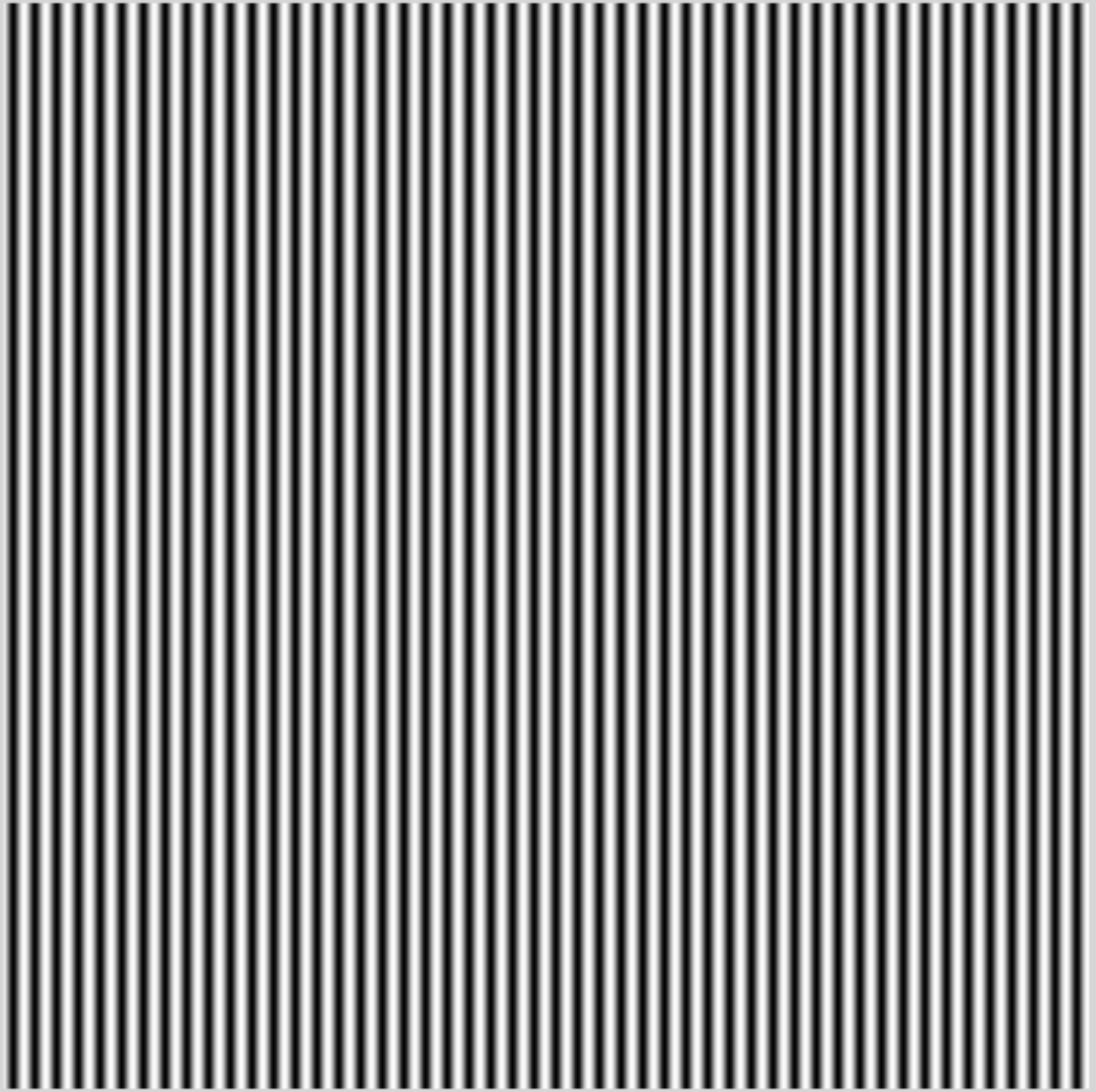
Subsampling Images



Slide credit: Christian Rupprecht, Oxford

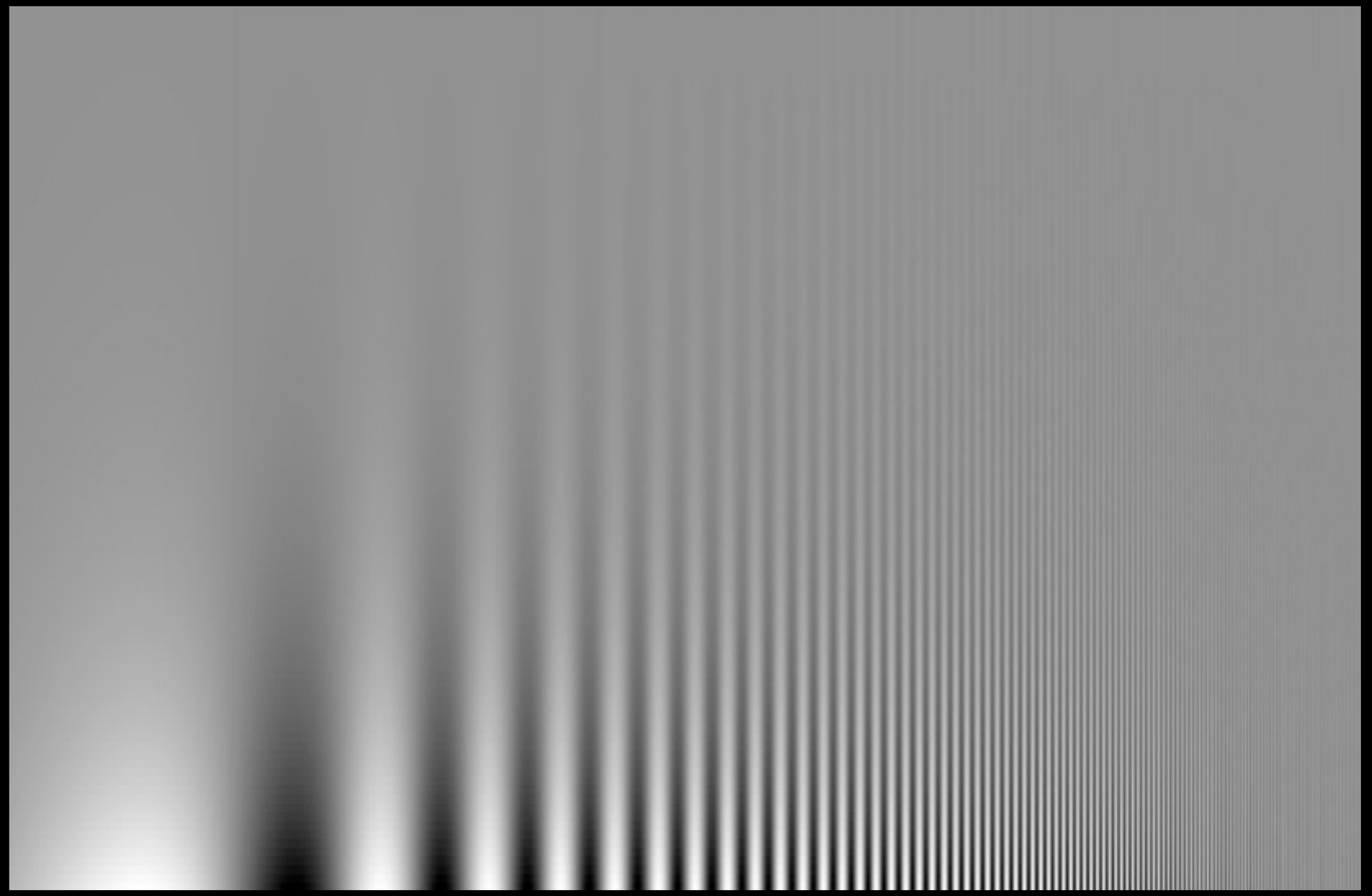
Human spatial perception & JPEG





Contrast Sensitivity Function

Campbell-Robson Chart



Contrast Sensitivity Function (CSF)

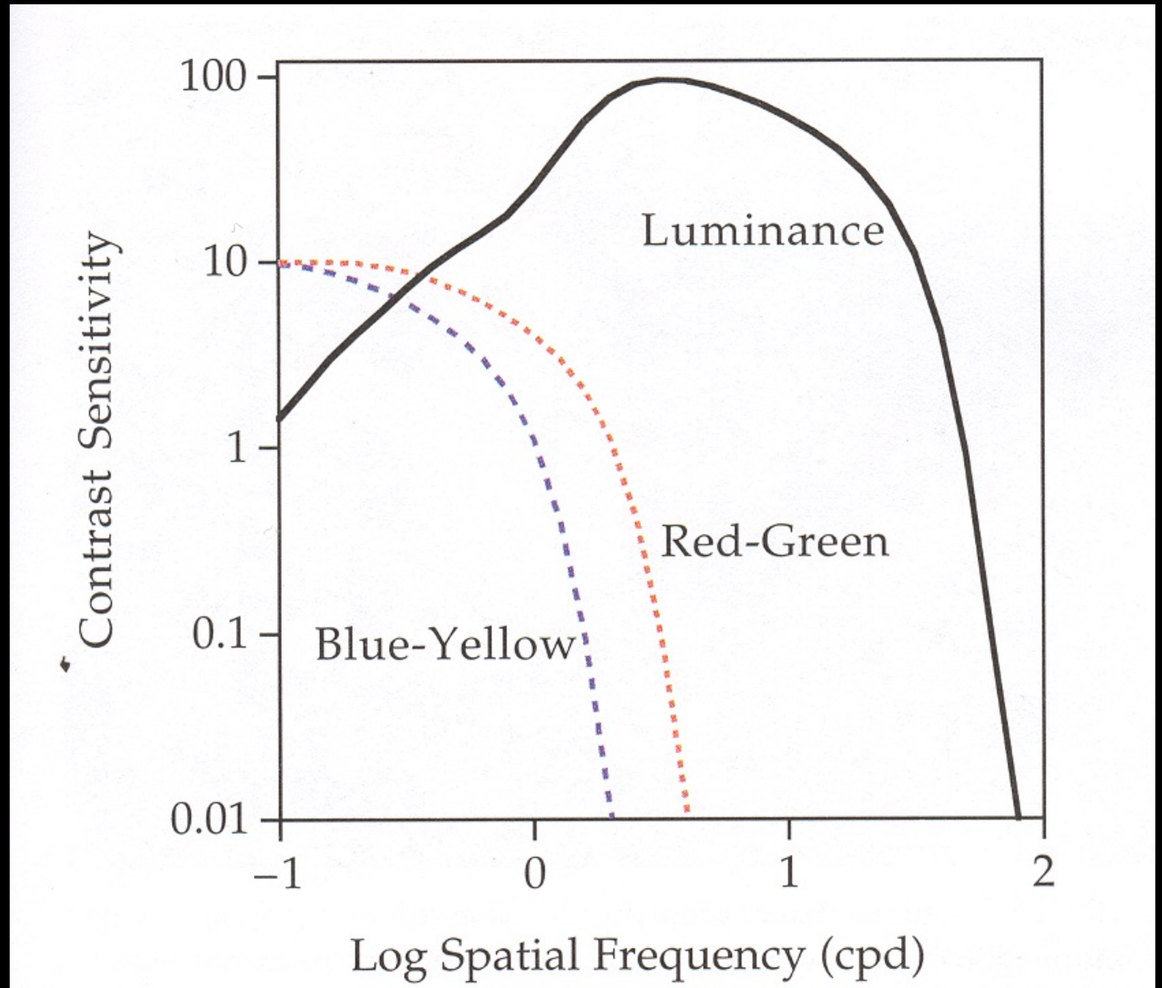
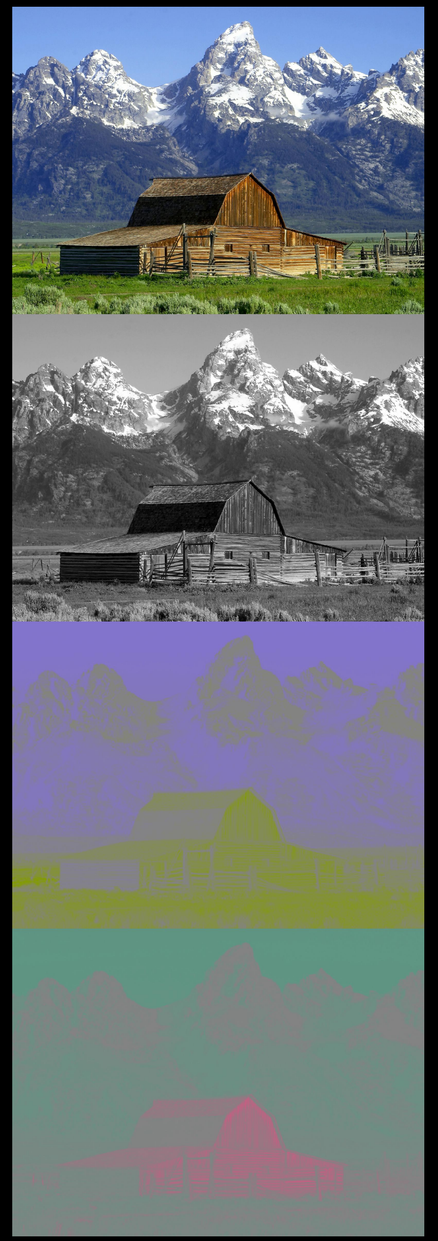


Figure 1-18. Spatial contrast sensitivity functions for luminance and chromatic contrast.

Opponents and image compression

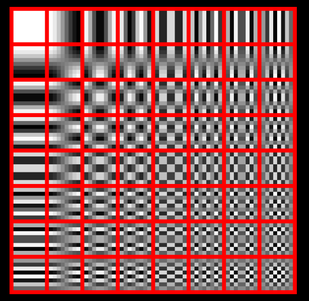
- JPG, MPG, television
- Color opponents instead of RGB
  - YCrCB, similar to YUV
- Compress color more than luminance
  - downsample by factor of two for jpeg
  - less bandwidth for TV
- Exploit contrast sensitivity function
  - Compress high frequencies more



JPEG Compression



- convert to YCbCr
  - half the resolution for Cr & Cb
- Perform Discrete Cosine Transform to work in frequency space
  - Local DCT, 8x8 pixel blocks



- Use CSF for quantization
  - more bits for frequencies with more sensitivity (medium)

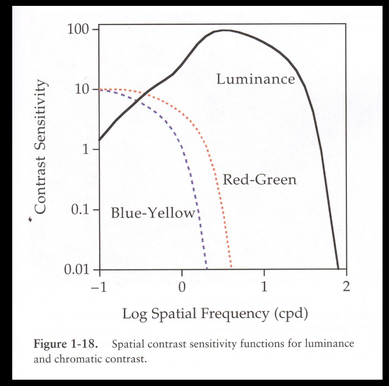
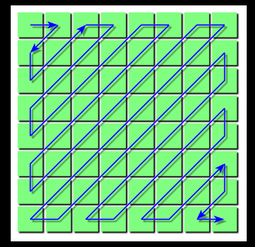


Figure 1-18. Spatial contrast sensitivity functions for luminance and chromatic contrast.

- Other usual coding tricks
  - entropy coding, smart order of blocks



Example

- 800 x 533 image



RGB Lightness a

Low quality JPEG (0 in Photoshop, 172 KB)



High quality JPEG (12 in Photoshop, 460 KB)



Recap and discussion

- Canonical basis
  - Simple
  - Emphasizes locality
- Fourier
  - Captures shift-invariance
  - Simplifies some operations (convolution, sampling)
  - Frequency (and kind of scale) separation
  - Related to human perception
  - BUT very annoying detail of boundary conditions
    - Big reason why it’s not directly used as much for practical implementations

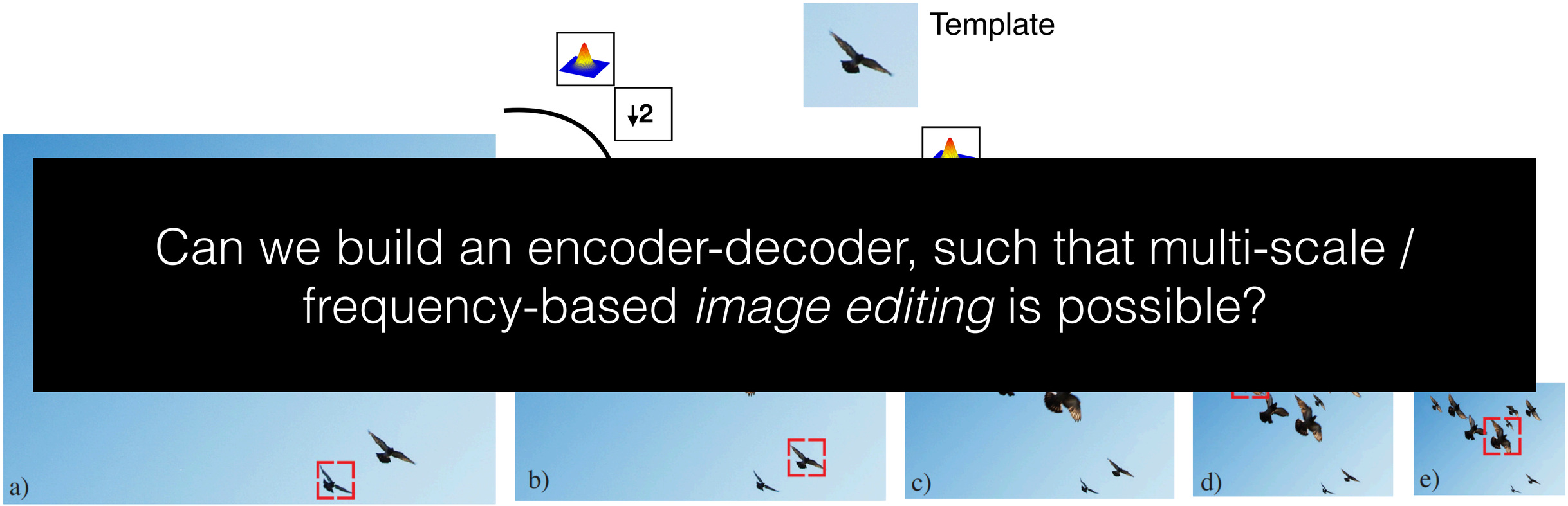
Beyond shifts: Detection across shift & scale





Template

The Gaussian Pyramid



Can we build an encoder-decoder, such that multi-scale / frequency-based *image editing* is possible?

A simple “encoder”: take image and transform it into a “representation” where analysis is simple

Eigenvector approach is possible but challenging

“Mellin” transform

https://en.wikipedia.org/wiki/Mellin_transform

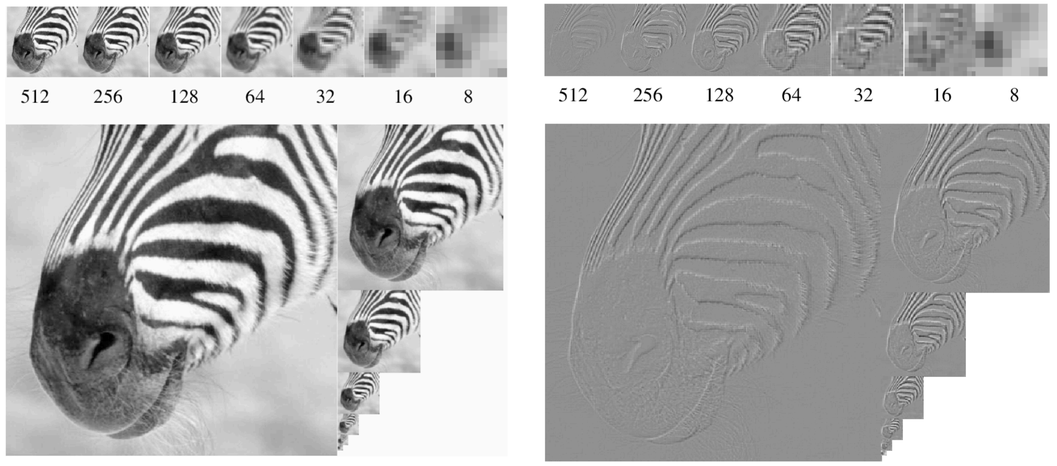
But requires assumption of infinite domain, boundary problems worse than Fourier

Pragmatic solution: something half-way between local and frequency

Warning: not even a basis, overcomplete (more coefficients than pixels)

Image pyramid idea

- Process images at different resolutions
- Operating independently at different scales enables many applications
  - Blending
  - Denoising
  - Tone mapping
  - Analysis
  - Compression
  - Optical flow (motion a
  - Key points
  - CNN & UNet



Original papers from the 80s

- Burt & Adelson ; Williams

532 IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-31, NO. 4, APRIL 1983

The Laplacian Pyramid as a Compact Image Code

PETER J. BURT, MEMBER, IEEE, AND EDWARD H. ADELSON

A Multiresolution Spline With Application to Image Mosaics

PETER J. BURT and EDWARD H. ADELSON
RCA David Sarnoff Research Center

Computer Graphics Volume 17, Number 3 July 1983

Pyramidal Parametrics

Lance Williams

Computer Graphics Laboratory
New York Institute of Technology
Old Westbury, New York

- Later and slightly different: wavelets

674 IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 11, NO. 7, JULY 1989

A Theory for Multiresolution Signal Decomposition: The Wavelet Representation

STEPHANE G. MALLAT

The Lifting Scheme:
A New Philosophy in Biorthogonal Wavelet Constructions

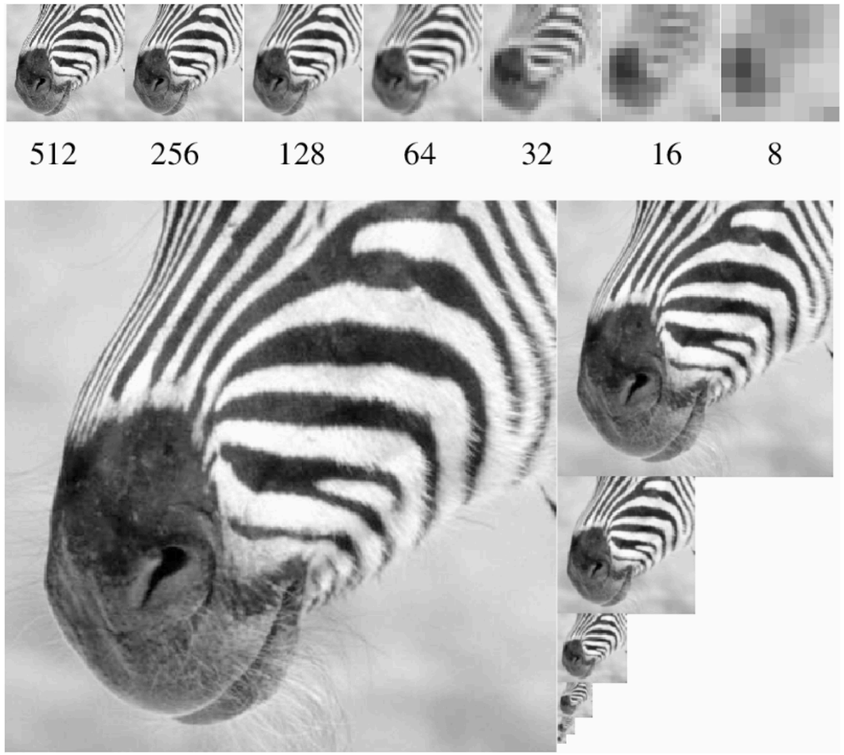
Wim Sweldens

Katholieke Universiteit Leuven Belgium, Department of Computer Science

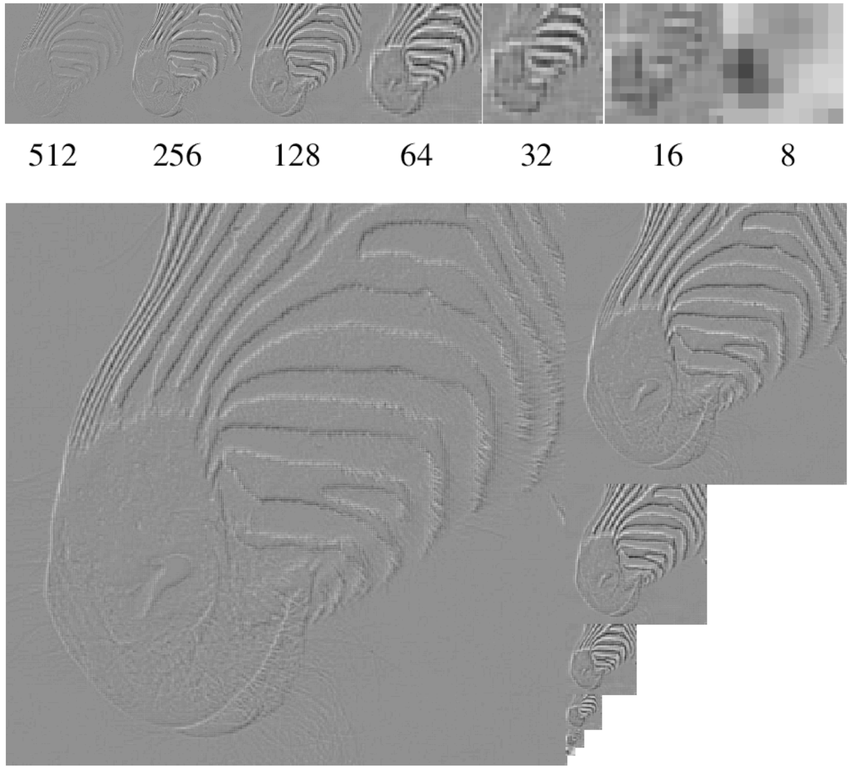
Wim.Sweldens@cs.kuleuven.ac.be

Two main families

Low pass
Each level is a lower resolution of the original

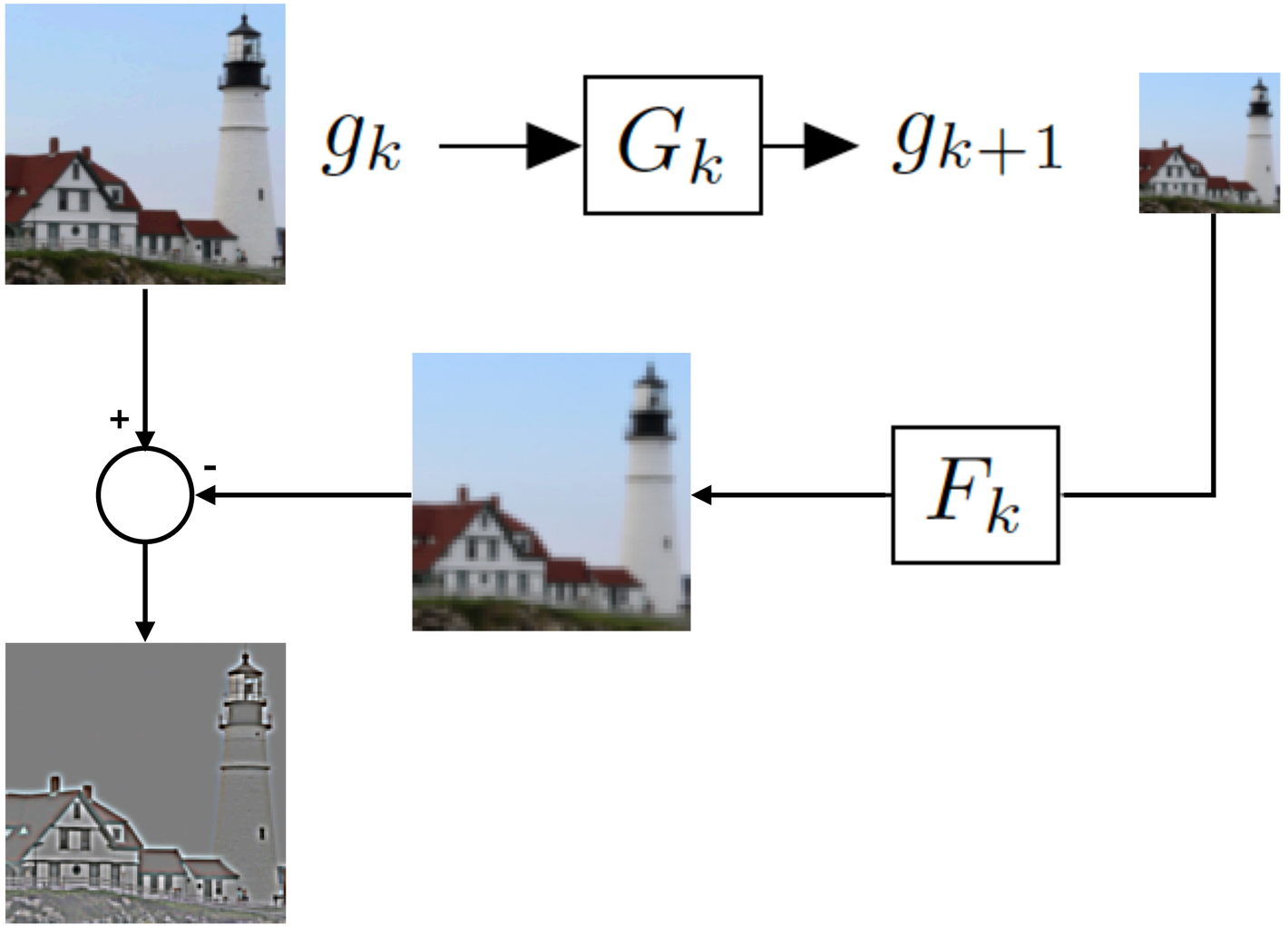


Band pass:
Only narrow frequencies per level

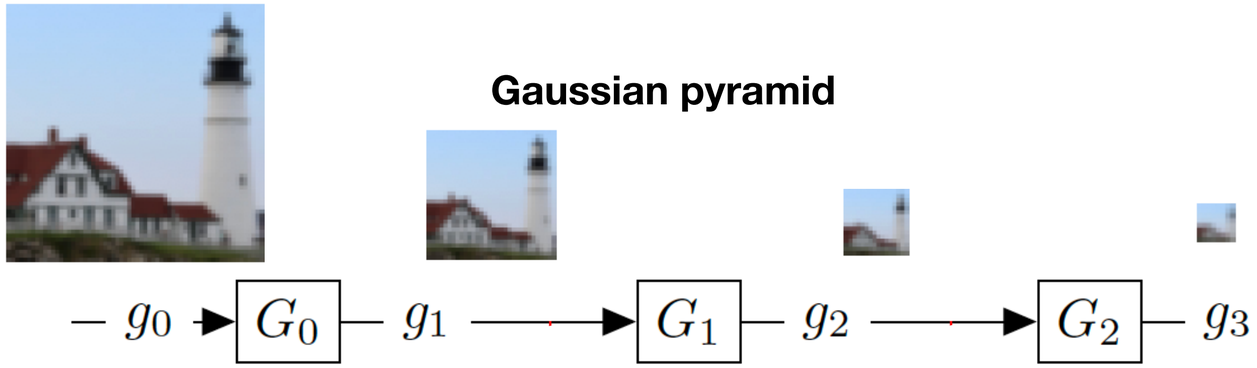


The Laplacian Pyramid

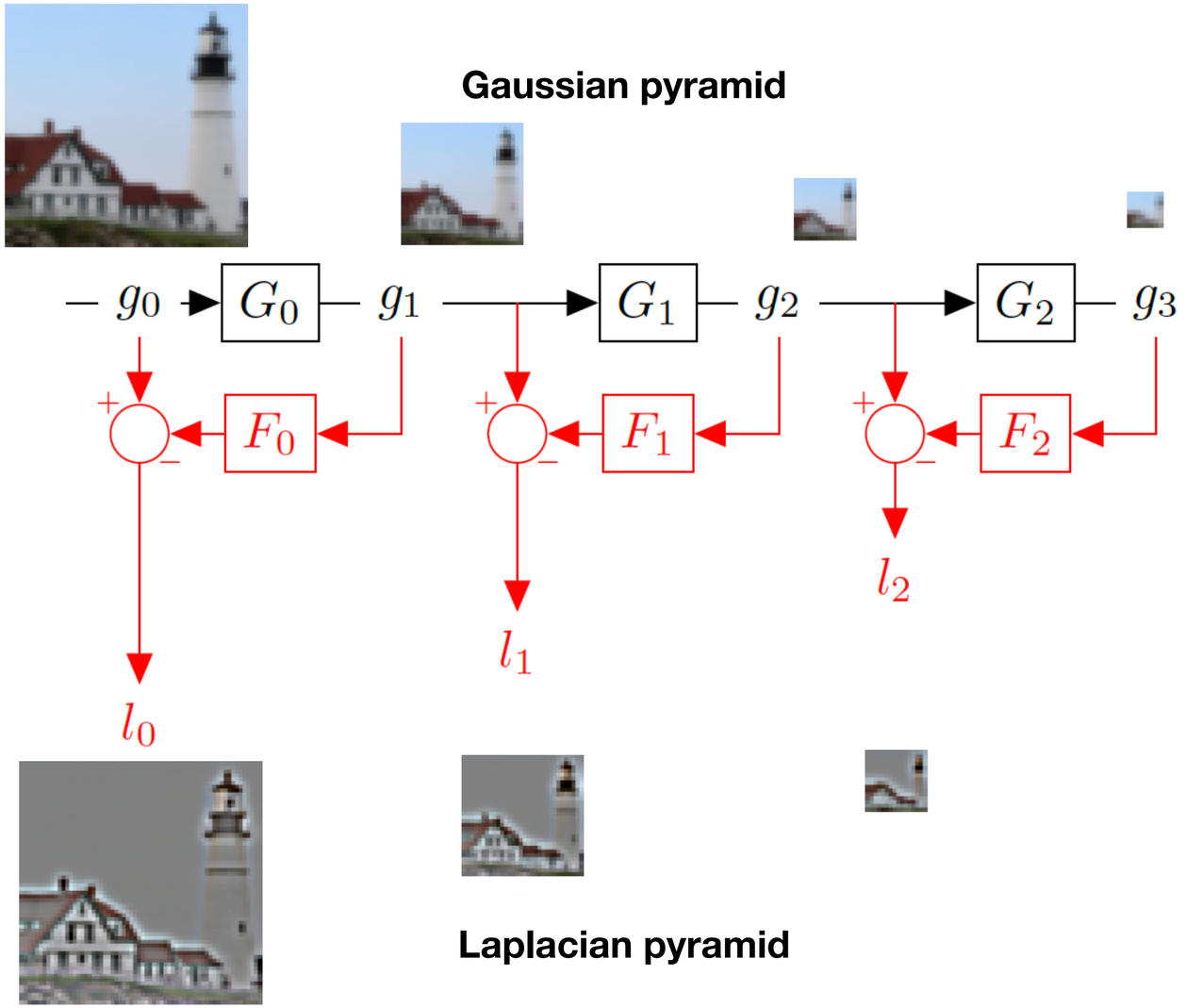
Compute the difference between upsampled Gaussian pyramid level k+1 and Gaussian pyramid level k.



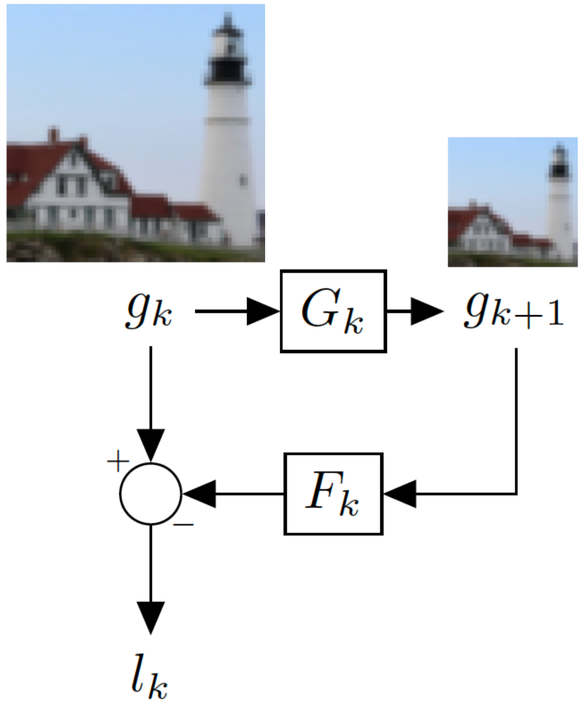
The Laplacian Pyramid



The Laplacian Pyramid



The Laplacian Pyramid operators in matrix form



Blurring and downsampling:

$$G_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \frac{1}{16} \begin{bmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 \end{bmatrix}$$

(Downsampling by 2)

(blur)

Upsampling and blurring:

$$F_0 =$$

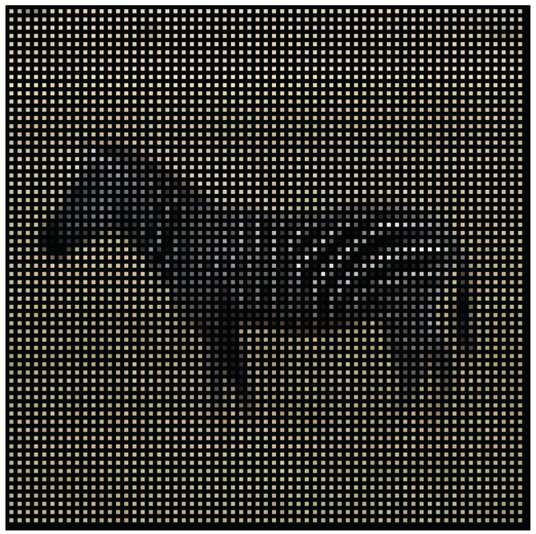
Upsampling

64x64



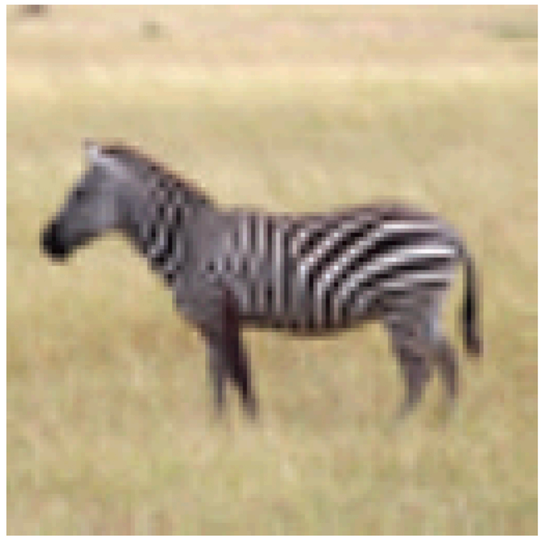
Insert zeros

128x128

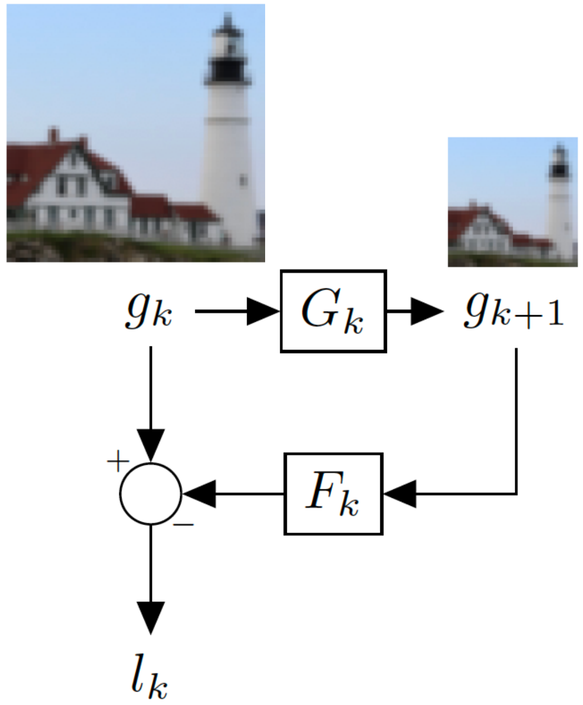


$\bigcirc \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0.5 & 1 & 0.5 \\ 0.25 & 0.5 & 0.25 \end{bmatrix} =$

128x128



The Laplacian Pyramid operators in matrix form



Blurring and downsampling:

$$G_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \frac{1}{16} \begin{bmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 \end{bmatrix}$$

(Downsampling by 2) (blur)

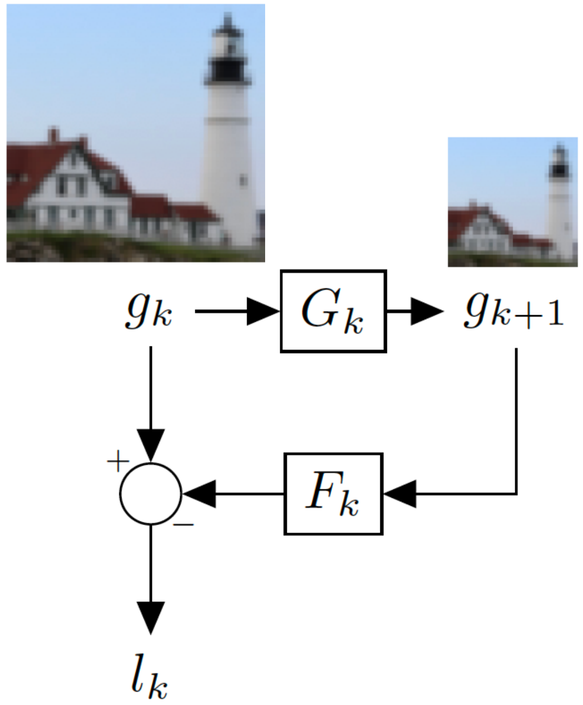
Upsampling and blurring:

$$F_0 = \frac{1}{8} \begin{bmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

(blur) (Upsampling by 2)

$$l_0 = (I_0 - F_0 G_0) g_0$$

The Laplacian Pyramid

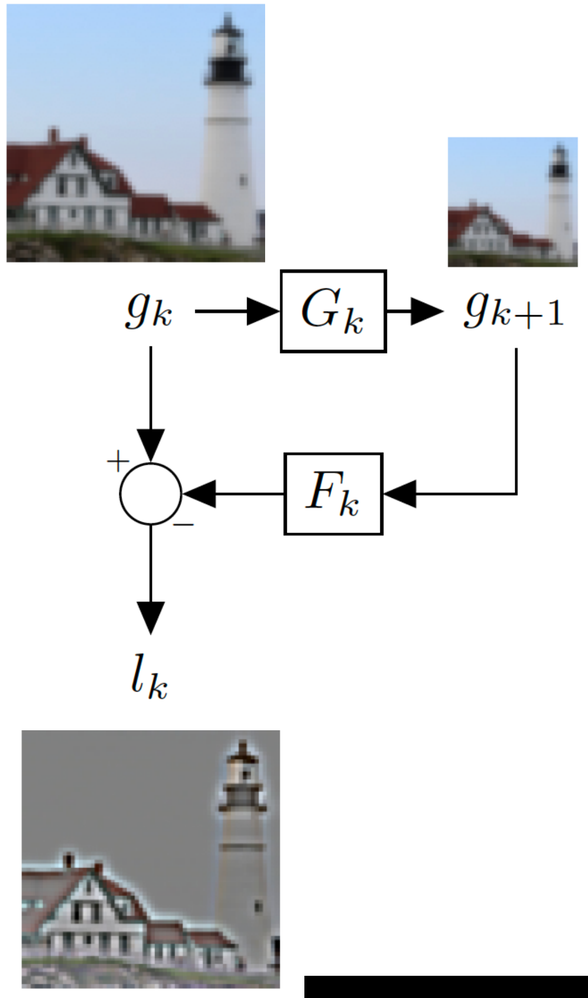




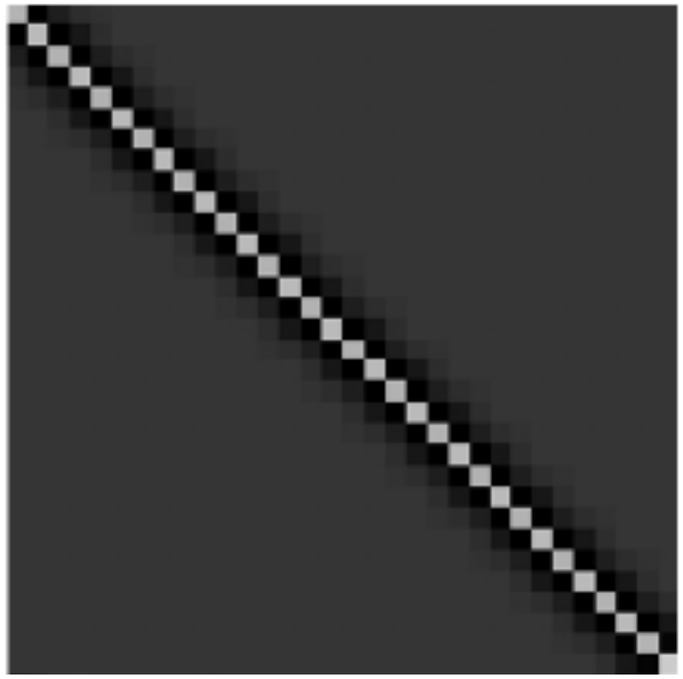
$$l_0 = (I_0 - F_0 G_0) g_0$$

$$= \frac{1}{256} \begin{bmatrix} 182 & -56 & -24 & -8 & -2 & 0 & 0 & 0 \\ -56 & 192 & -56 & -32 & -8 & 0 & 0 & 0 \\ -24 & -56 & 180 & -56 & -24 & -8 & -2 & 0 \\ -8 & -32 & -56 & 192 & -56 & -32 & -8 & 0 \\ -2 & -8 & -24 & -56 & 180 & -56 & -24 & -8 \\ 0 & 0 & -8 & -32 & -56 & 192 & -56 & -32 \\ 0 & 0 & -2 & -8 & -24 & -56 & 182 & -48 \\ 0 & 0 & 0 & 0 & -8 & -32 & -48 & 224 \end{bmatrix} x$$

The Laplacian Pyramid



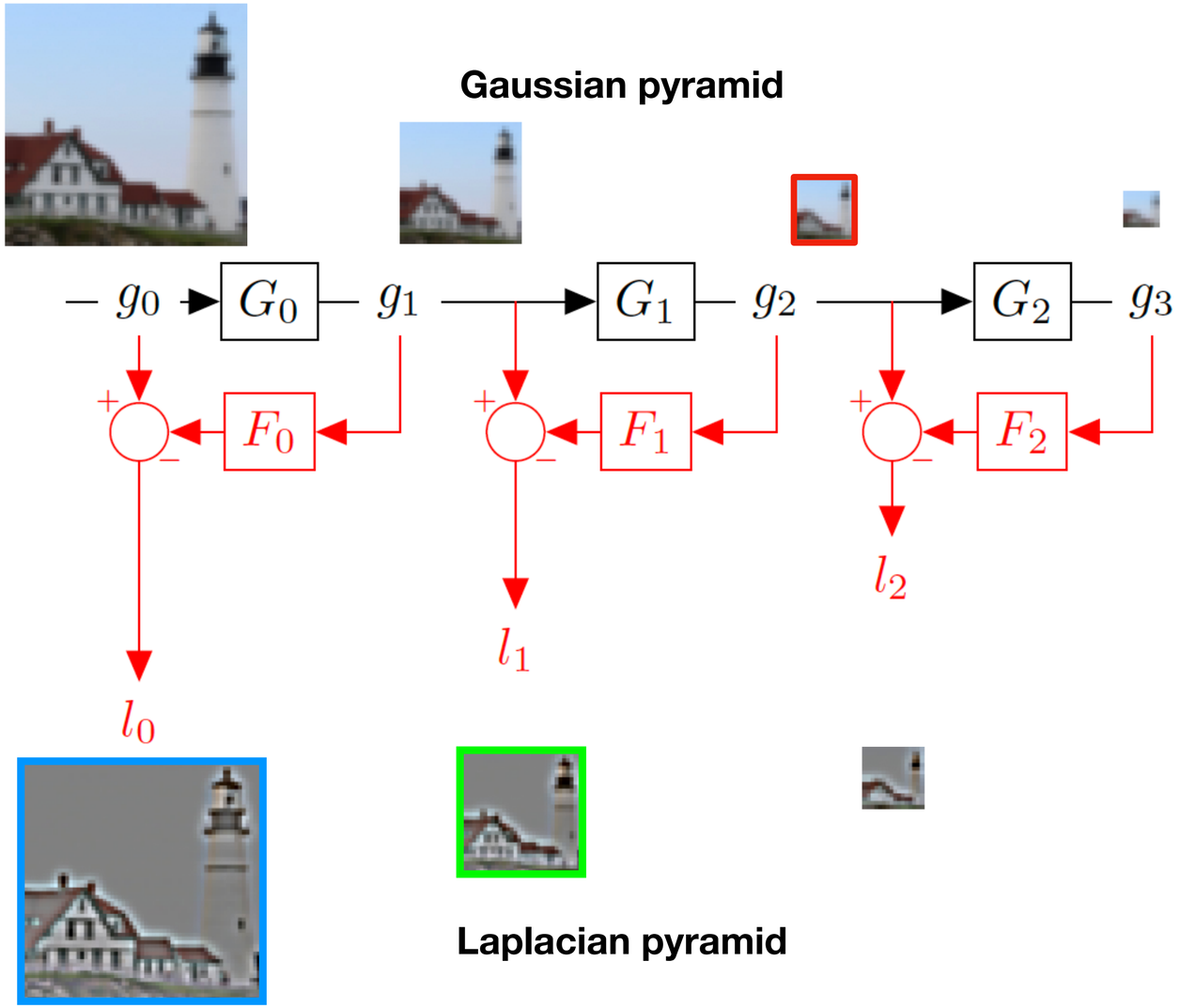
$$l_0 = (I_0 - F_0 G_0) g_0$$

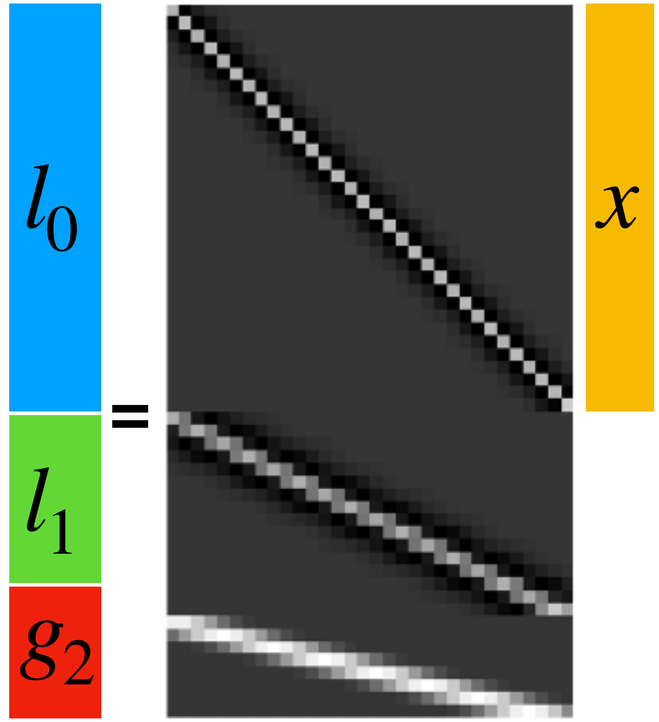


$$\begin{bmatrix}
182 & -56 & -24 & -8 & -2 & 0 & 0 & 0 \\
-56 & 192 & -56 & -32 & -8 & 0 & 0 & 0 \\
-24 & -56 & 180 & -56 & -24 & -8 & -2 & 0 \\
-8 & -32 & -56 & 192 & -56 & -32 & -8 & 0 \\
-2 & -8 & -24 & -56 & 180 & -56 & -24 & -8 \\
0 & 0 & -8 & -32 & -56 & 192 & -56 & -32 \\
0 & 0 & -2 & -8 & -24 & -56 & 182 & -48 \\
0 & 0 & 0 & 0 & -8 & -32 & -48 & 224
\end{bmatrix}$$

"The Laplace Operator": How much does the value of a function differ from the average of its neighbors?

The Laplacian Pyramid shown in matrix form



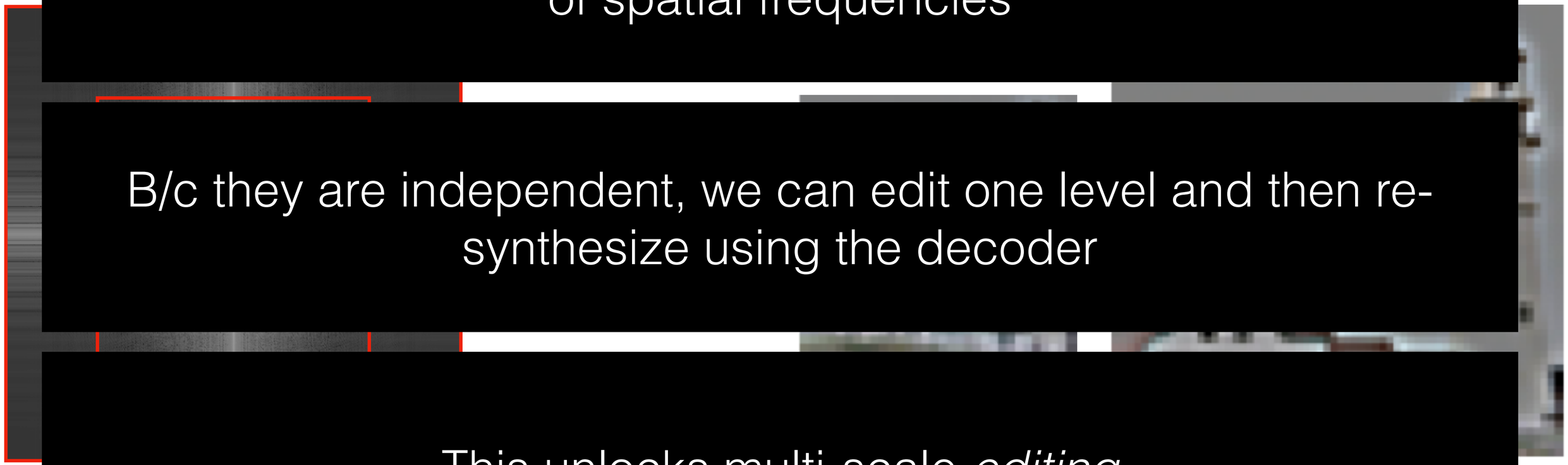


Laplacian Pyramid - Frequency Perspective

Essentially, each level of the laplacian pyramid is a particular band of spatial frequencies

B/c they are independent, we can edit one level and then re-synthesize using the decoder

This unlocks multi-scale *editing*



Inverting the Laplacian Pyramid









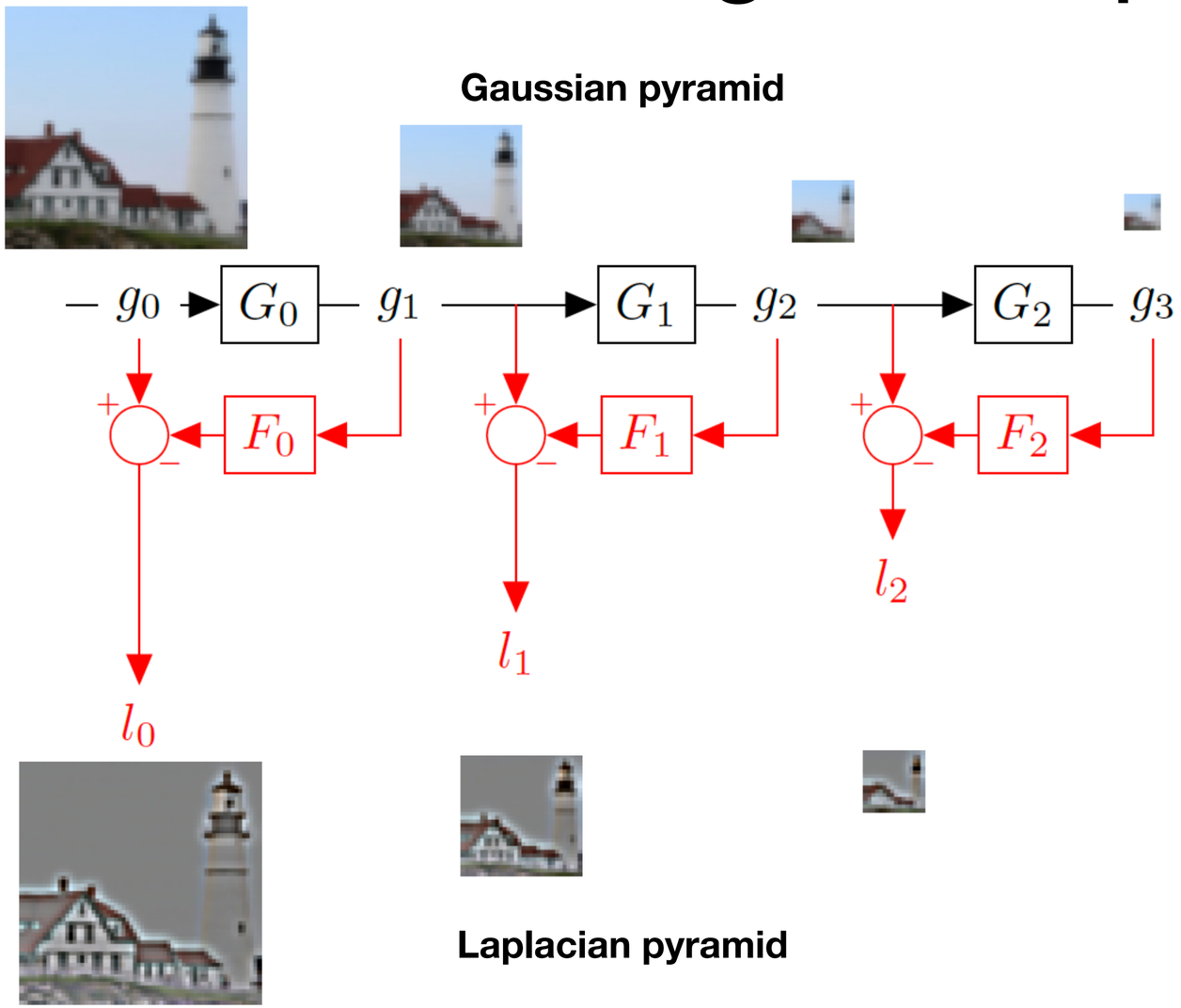
Gaussian residual

Laplacian pyramid

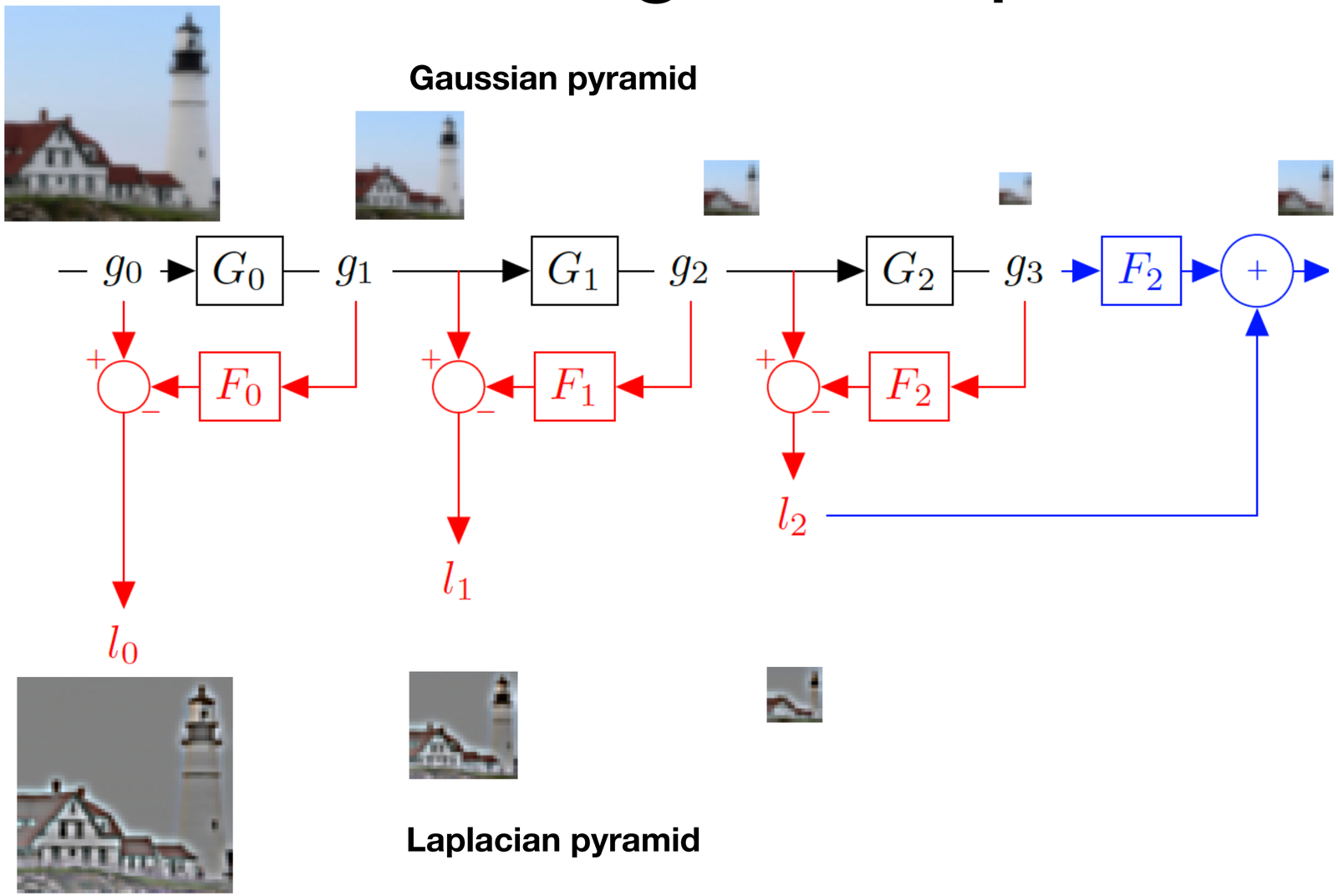


Can we invert the Laplacian Pyramid?

Inverting the Laplacian Pyramid



Inverting the Laplacian Pyramid



Inverting the Laplacian Pyramid

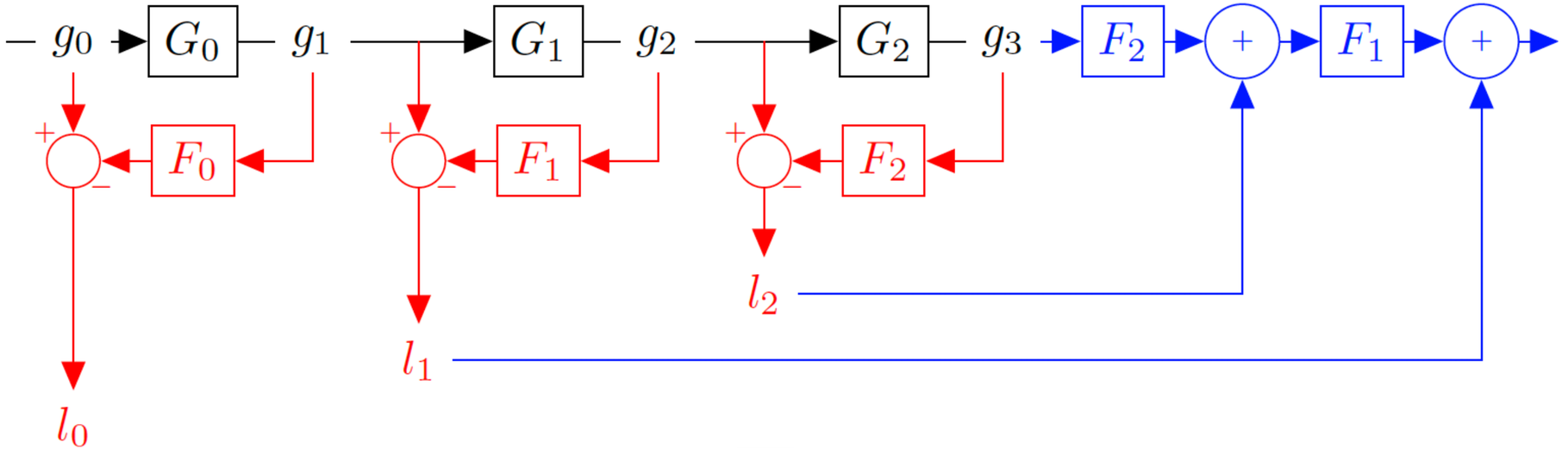


Gaussian pyramid







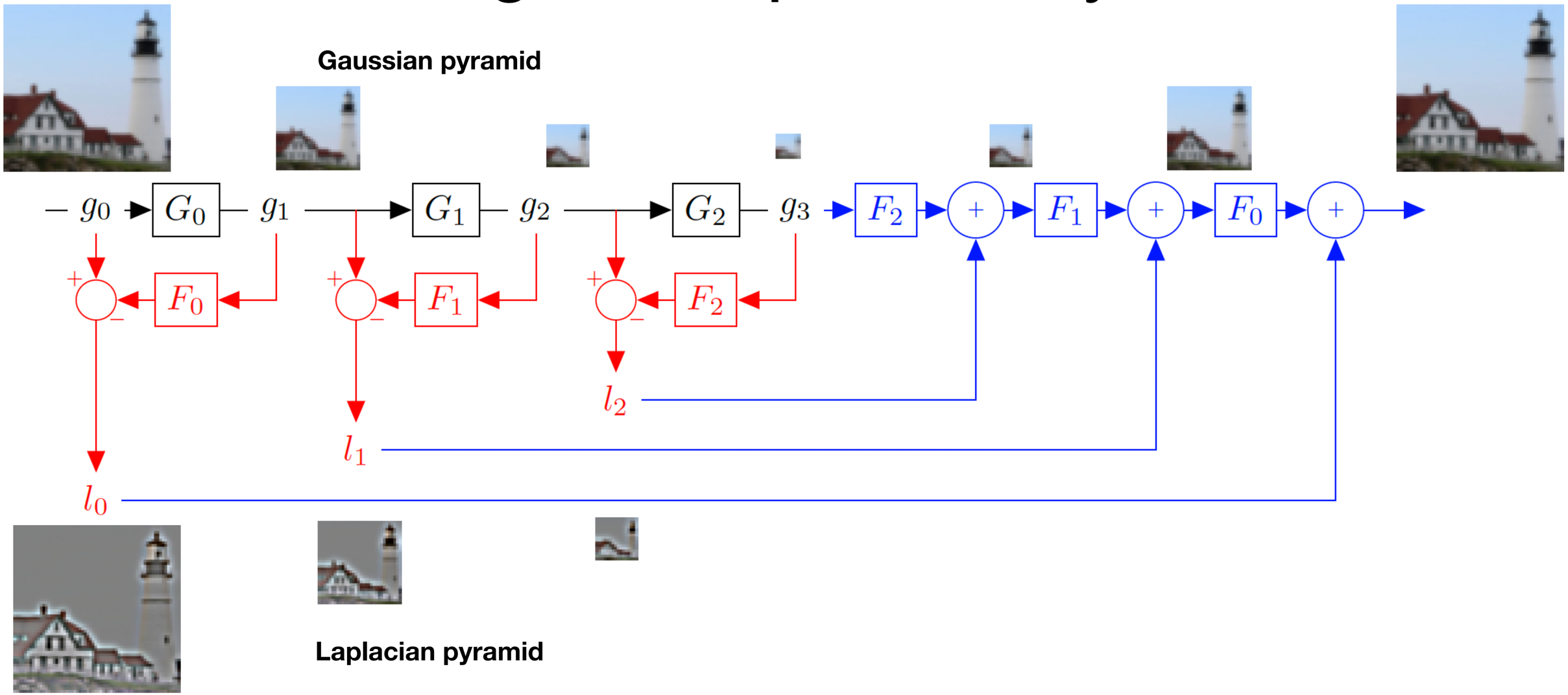




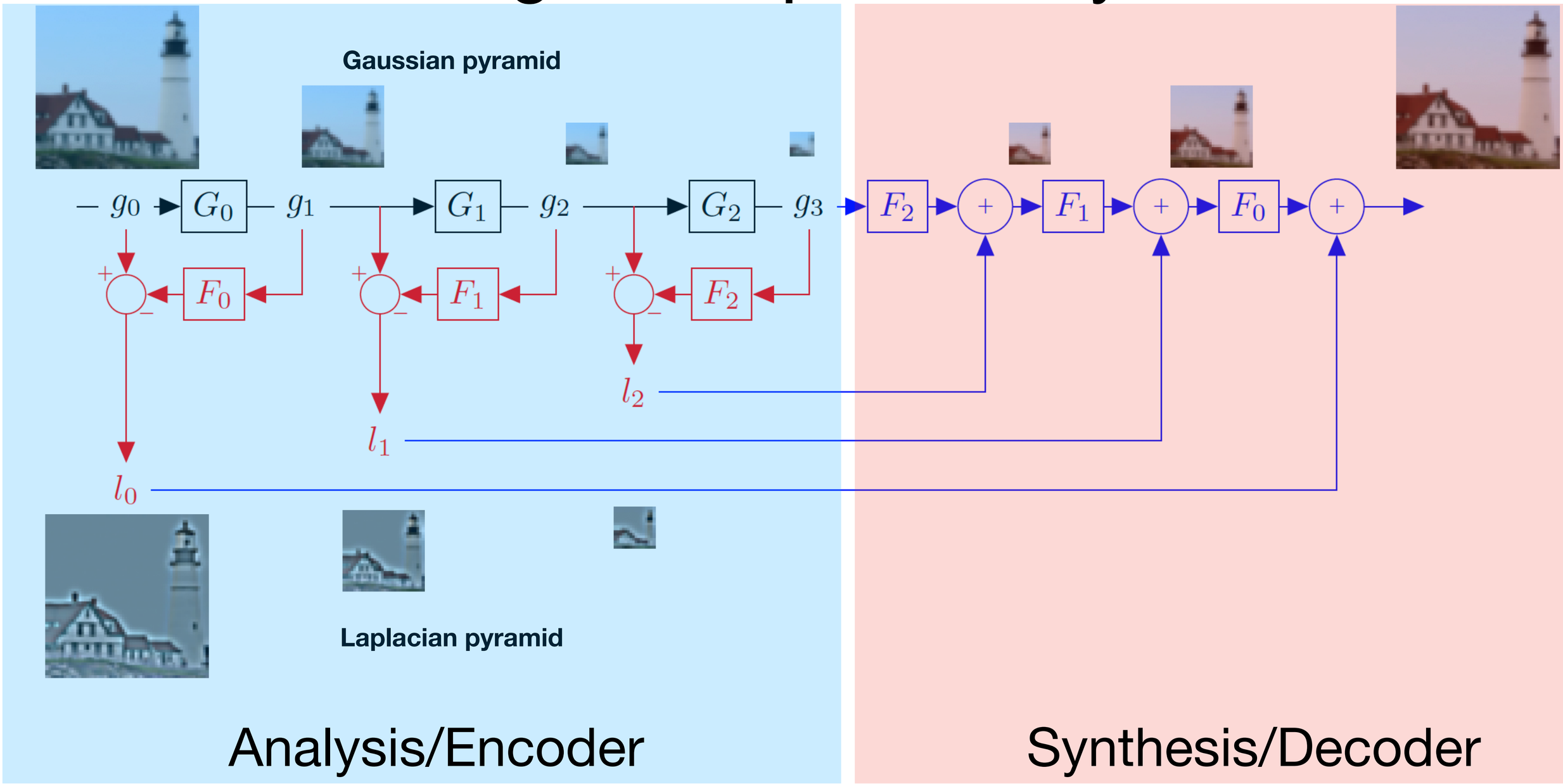


Laplacian pyramid

Inverting the Laplacian Pyramid



Inverting the Laplacian Pyramid



Application: Blending Apple & Orange



Apple image

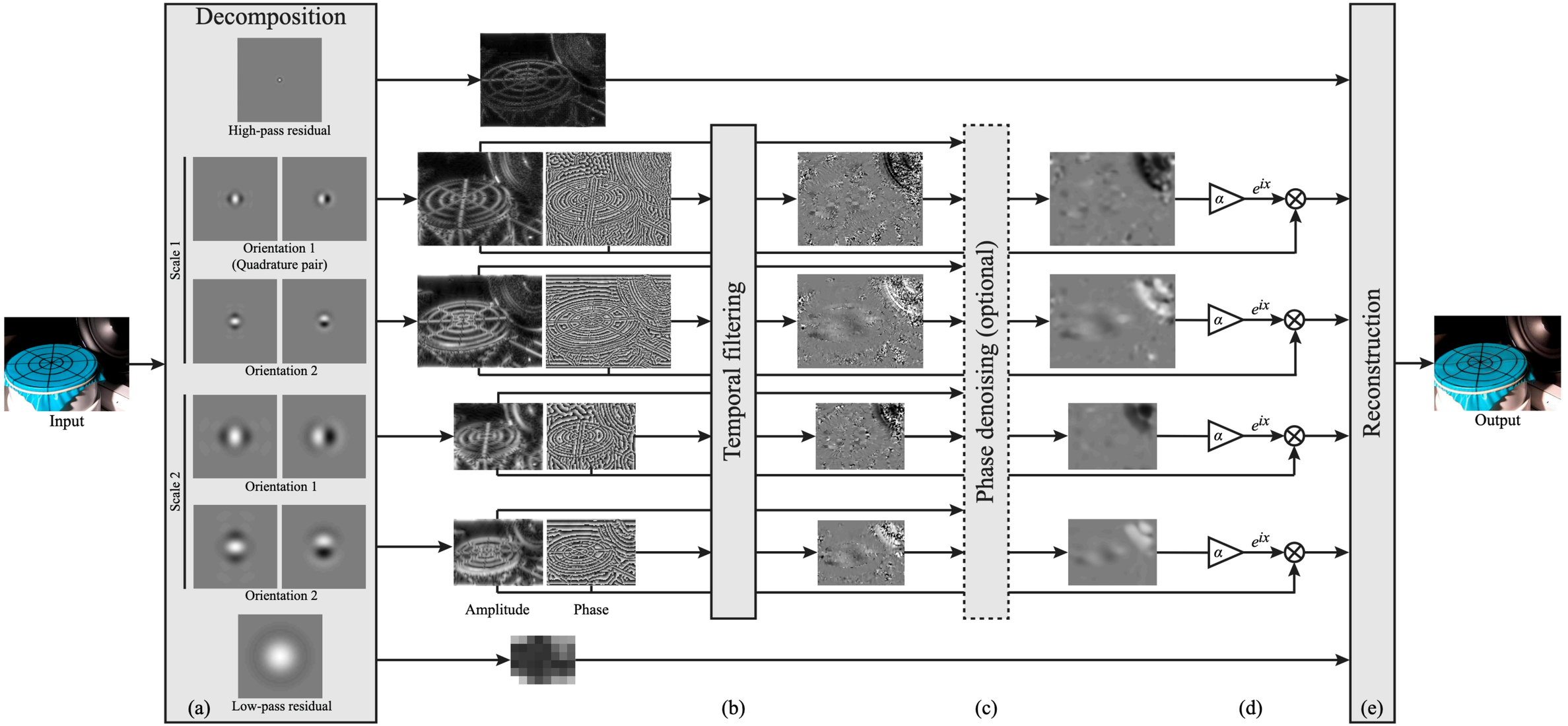


Orange Image



Unblended image

Application (and your homework): motion magnification



Recap

We mathematically characterized images as an inner product space

We discussed different choices of basis for this space and why they matter

We discussed different linear operators and how they look like in the spatial and pixel domain

Next lecture: Where does the Fourier Transform come from, and is there more where it came from?