Fourier Transforms

Fourier transform for 1D images

- A 1D image with N pixels is a vector of size N
- Every basis has N pixels
- There must be N basis elements
- n-th element of k-th basis in standard basis
  - $E_k(n) = \begin{cases} 1 & if\ k = n \\ 0 & otherwise \end{cases}$
- n-th element of k-th basis in Fourier basis
  - $B_k(n) = e^{\frac{i2\pi kn}{N}}$

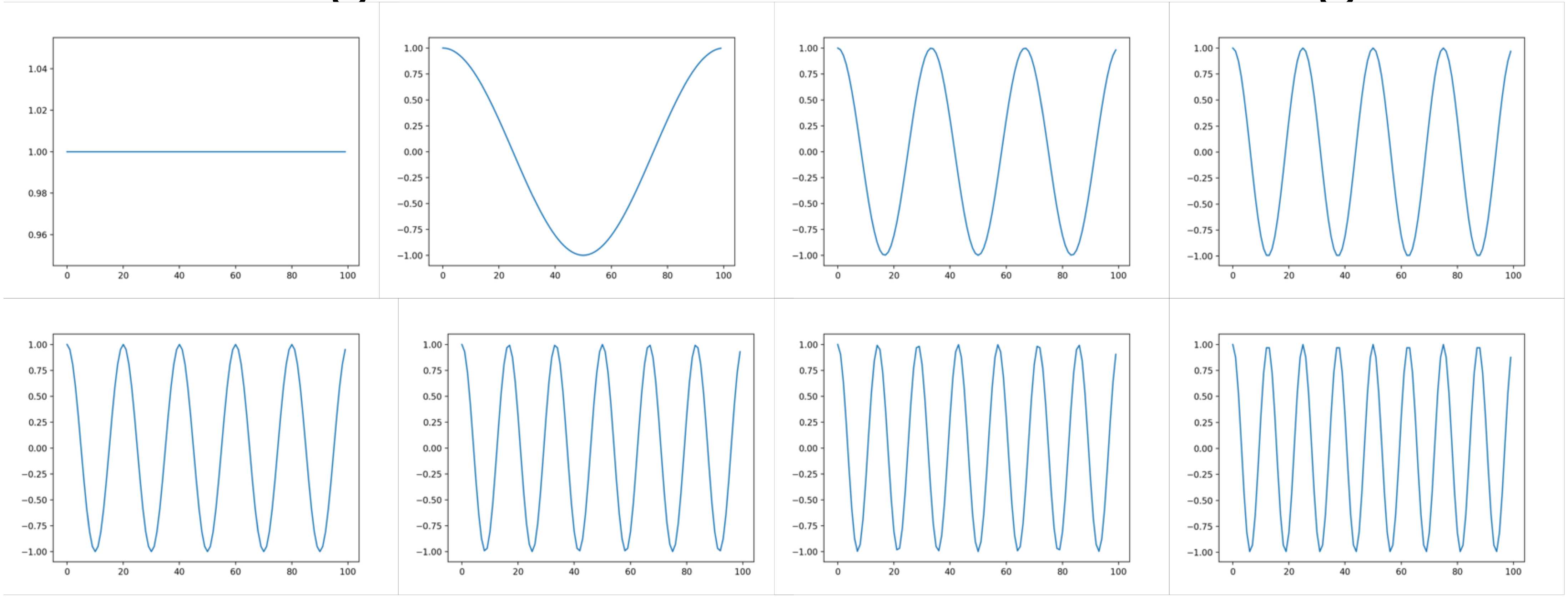
Fourier transform for 1D images

- Converting from standard basis to Fourier basis = Fourier transform
  - $X(k) = \sum_n x(n)\, e^{-\frac{i2\pi kn}{N}}$
  - Note that can be written as a matrix multiplication with X and x as vectors
    - $X = Bx$
- Convert from Fourier basis to standard basis
  - $x(n) = \sum_k X(k) e^{\frac{i2\pi kn}{N}}$

Fourier transform

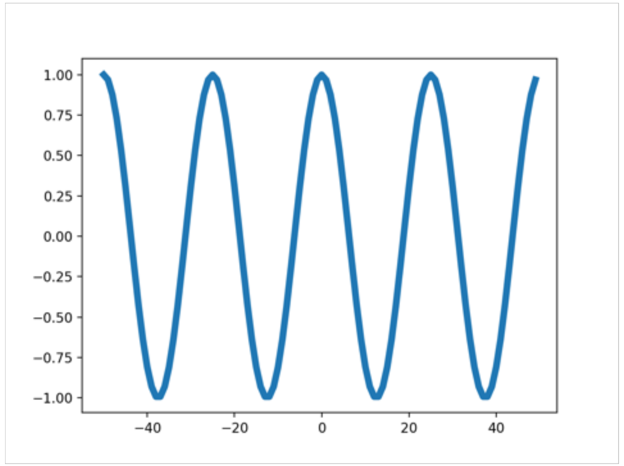
- Problem: basis is complex, but signal is real?
- Combine a pair of conjugate basis elements
- $B_{N-k}(\mathrm{n}) = e^{\frac{i2\pi(N-k)n}{N}} = e^{i2\pi n - \frac{i2\pi kn}{N}} = e^{-\frac{i2\pi kn}{N}} = B_{-k}(n)$
- Consider $B_{-N/2}$ to $B_{N/2}$ as basis elements
- Real signals will have same coefficients for B_k and B_{-k}

Visualizing the Fourier basis for 1D images

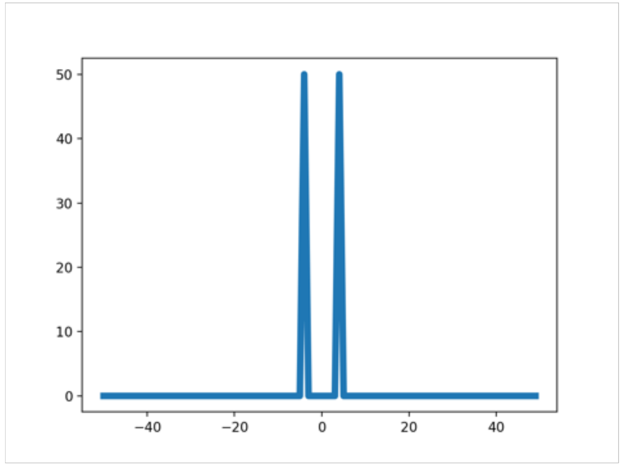


Signal

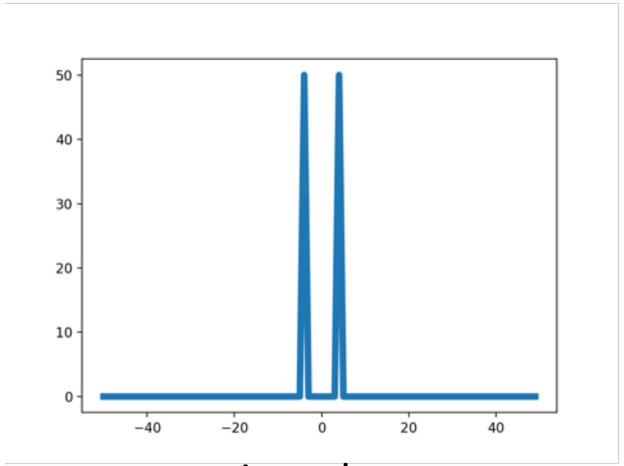
Fourier transform



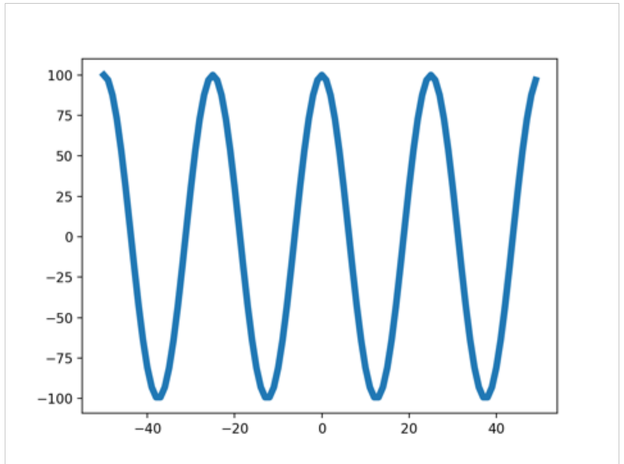
cos



impulse



impulse



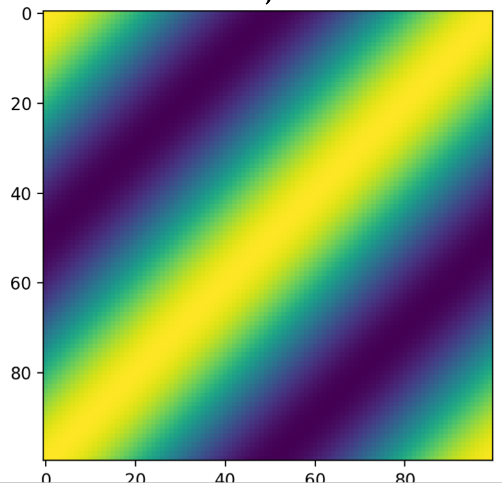
cos

Fourier transform for images

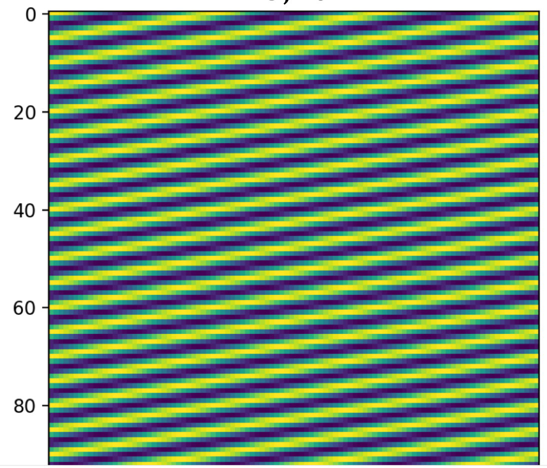
- Images are 2D arrays
- Fourier basis for 1D array indexed by frequence
- Fourier basis elements are indexed by 2 spatial frequencies
- (i,j)th Fourier basis for N x N image
  - Has period N/i along x
  - Has period N/j along y
- $B_{k,l}(x, y) = e^{\frac{2\pi ikx}{N} + \frac{2\pi ily}{N}}$
  $= \cos\left(\frac{2\pi kx}{N} + \frac{2\pi ly}{N}\right) + i\sin\left(\frac{2\pi kx}{N} + \frac{2\pi ly}{N}\right)$

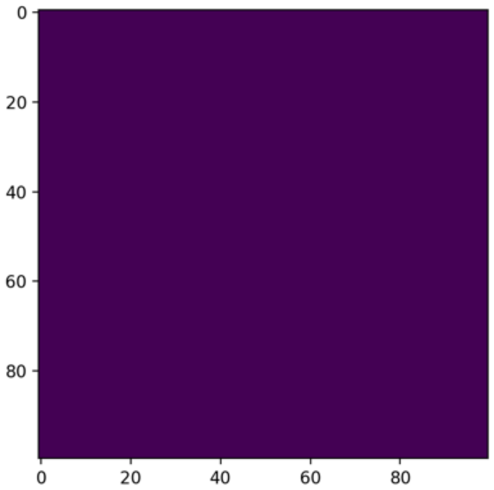
Visualizing the Fourier basis for images

$B_{1,1}$

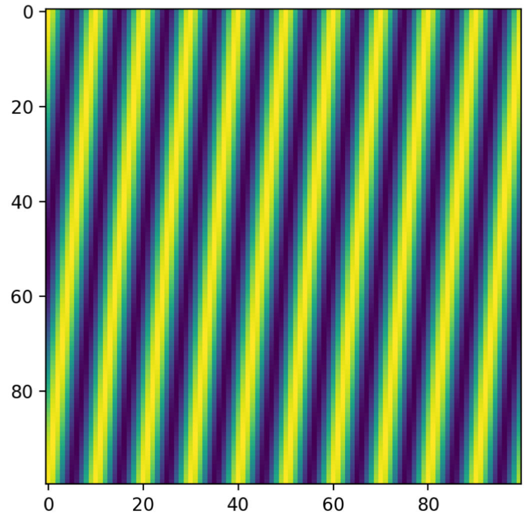


$B_{3,20}$





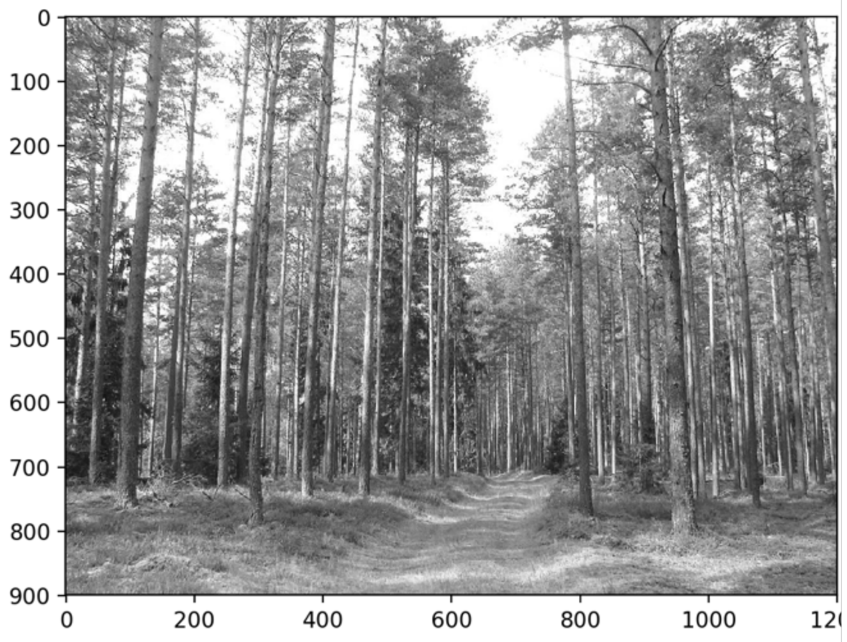
$B_{0,0}$

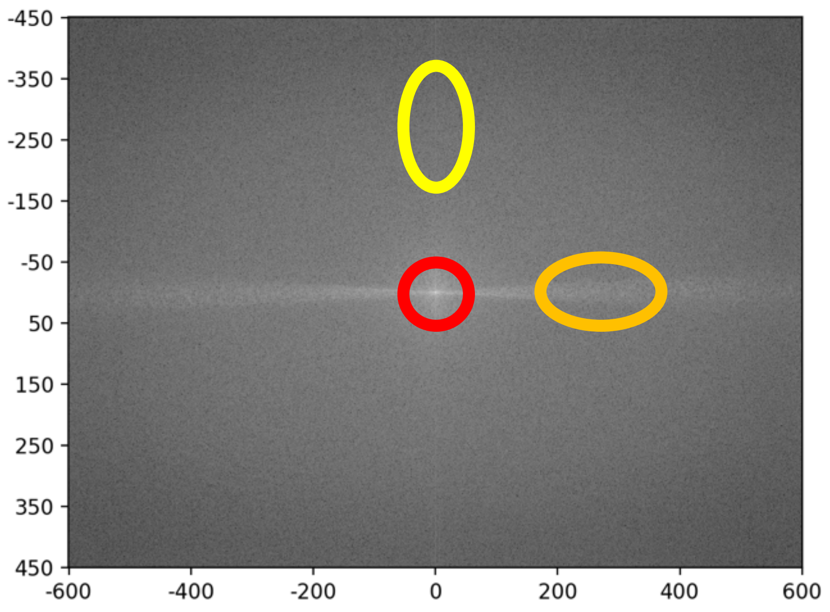


$B_{10,1}$

Visualizing the Fourier transform

- Given NxN image, there are NxN basis elements
- Fourier coefficients can be represented as an NxN image





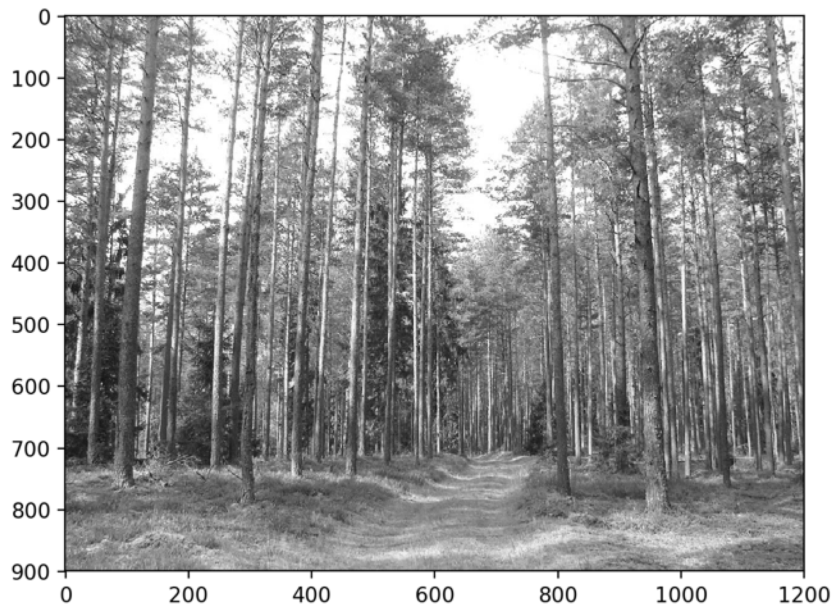
Converting to and from the Fourier basis

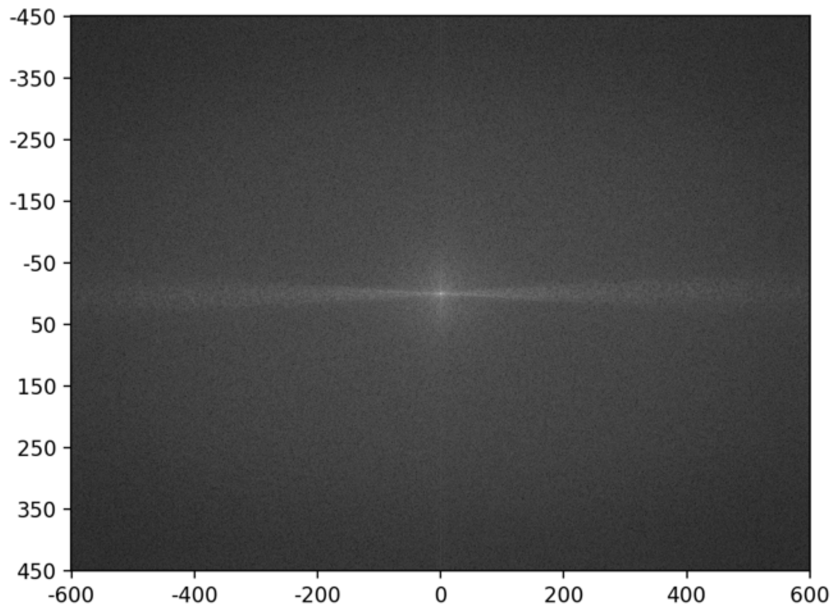
- Given an image f, Fourier coefficients F
- How do we get f from F?
  - $f = \sum_{k,l} F(k,l) B_{k,l}$
  - $f(m,n) = \sum_{k,l} F(k,l) e^{i\left(\frac{2\pi k m}{N} + \frac{2\pi l n}{N}\right)}$
  - "Inverse Fourier Transform"
- How do we get F from f?
  - $F(k,l) = \sum_{m,n} f(m,n) e^{-i\left(\frac{2\pi k m}{N} + \frac{2\pi i l n}{N}\right)}$
  - "Fourier Transform"

Why Fourier transforms?

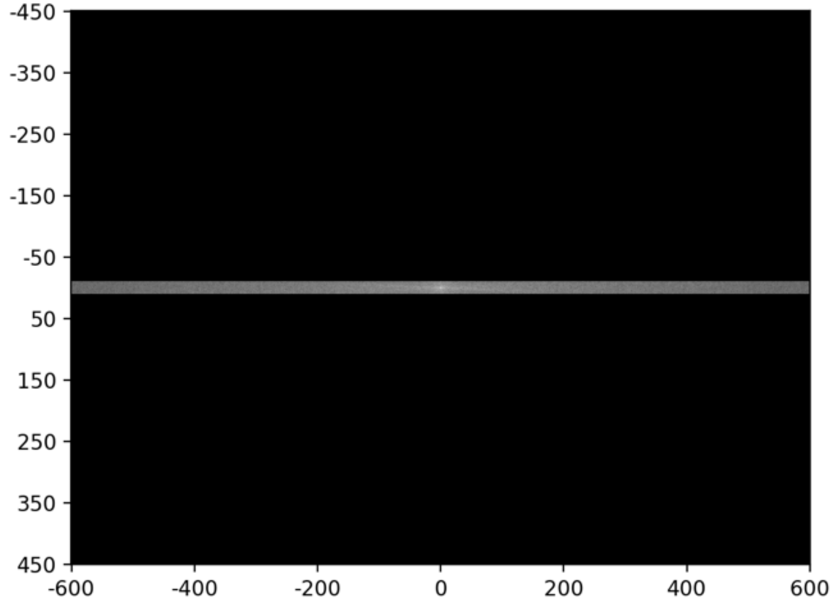
- Think of image in terms of low and high frequency information
- Low frequency: large scale structure, no details
- High frequency: fine structure

Why Fourier transforms?

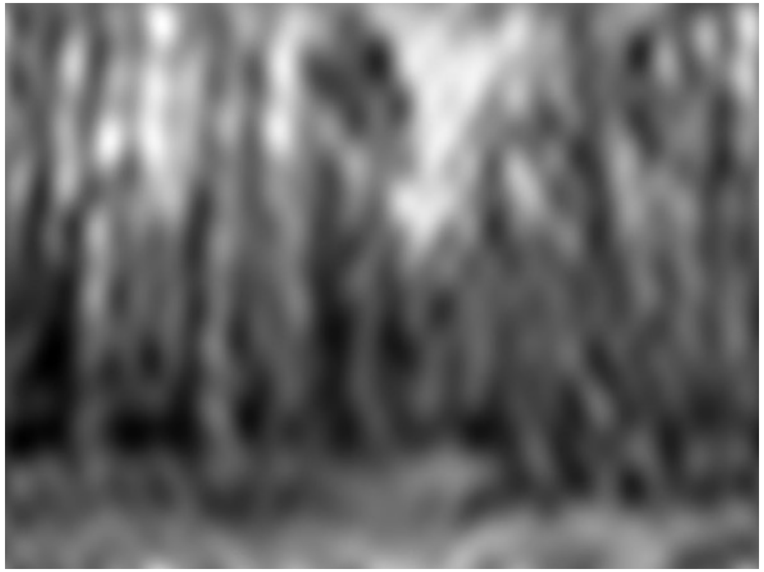


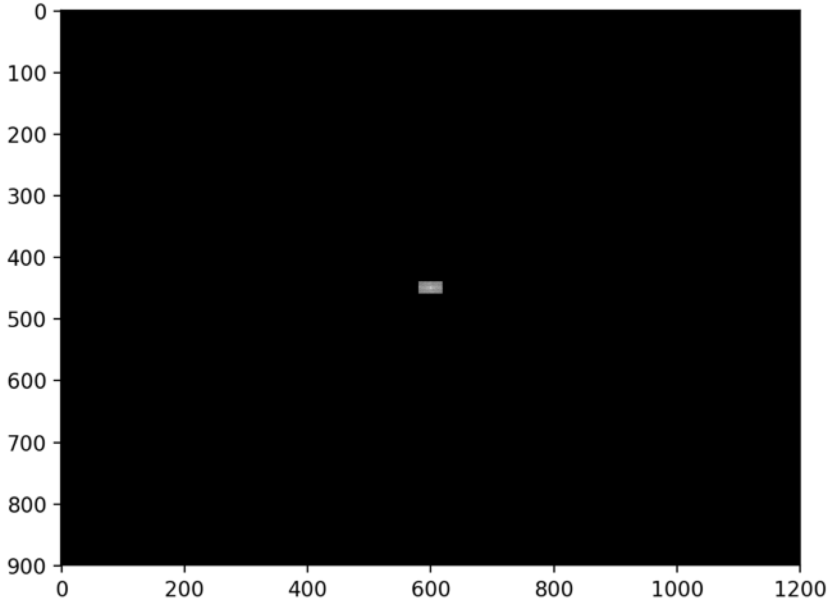


Why Fourier transforms?



Why Fourier transforms?





Removing high frequency components looks like blurring. Is there more to this relationship?

Dual domains

- Image: Spatial domain

- Fourier Transform: Frequency domain
  - Amplitudes are called spectrum

- For any transformations we do in spatial domain, there are corresponding transformations we can do in the frequency domain
- *And vice-versa*

Dual domains

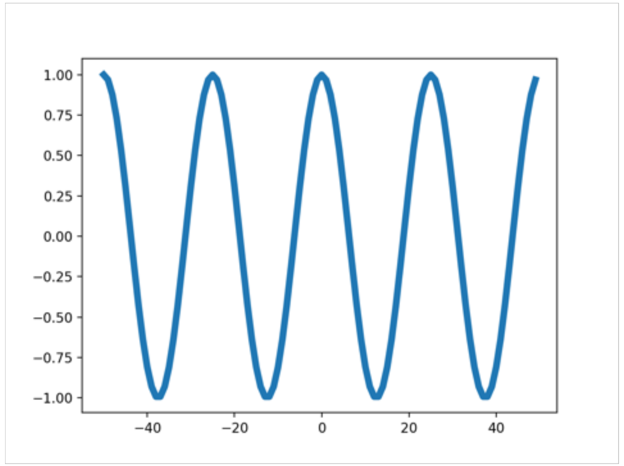
- *Convolution* in spatial domain = *Point-wise multiplication* in frequency domain

$$h = f * g$$
$$H = FG$$

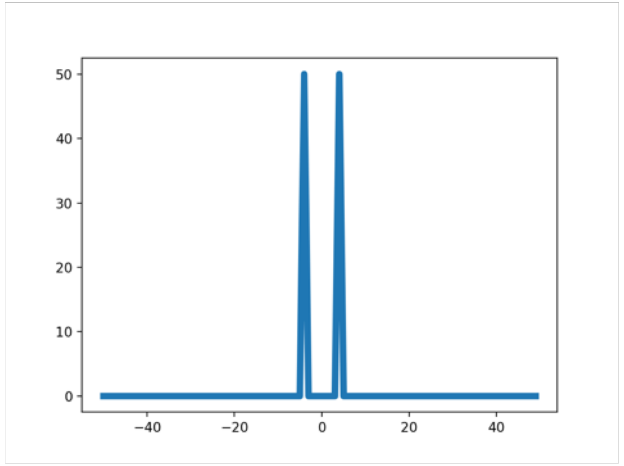
- *Convolution* in frequency domain = *Point-wise multiplication* in spatial domain

Signal

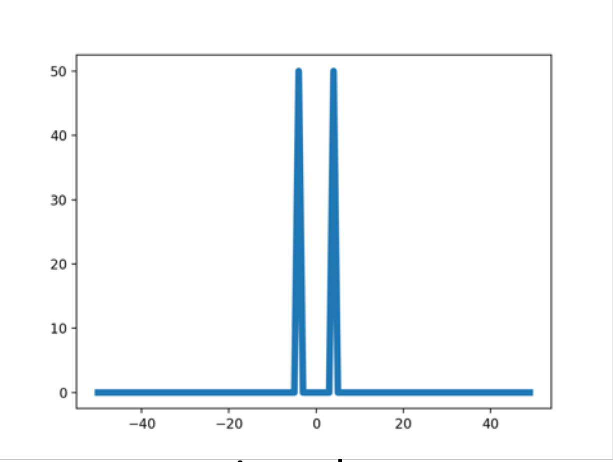
Fourier transform



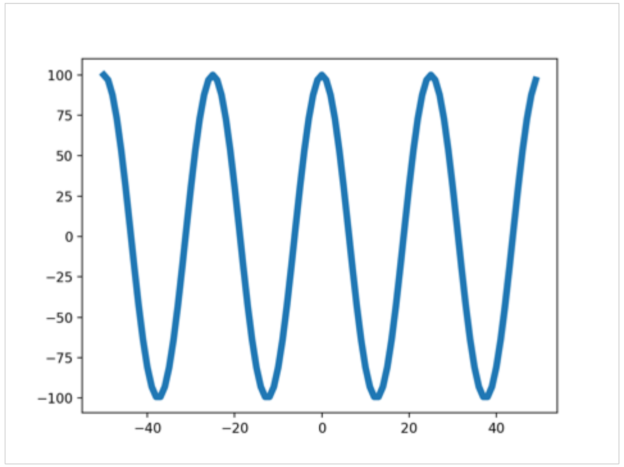
cos



impulse



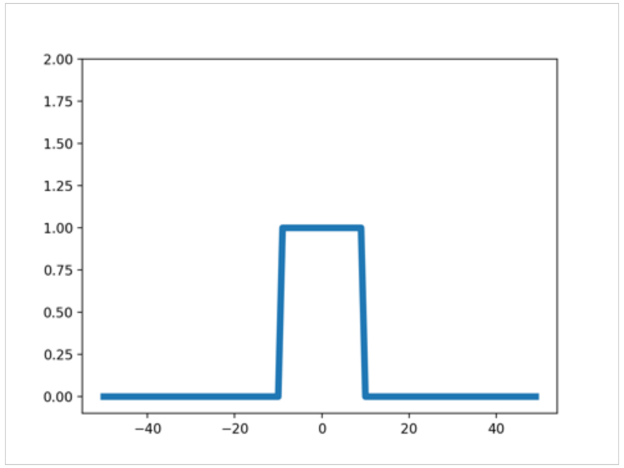
impulse



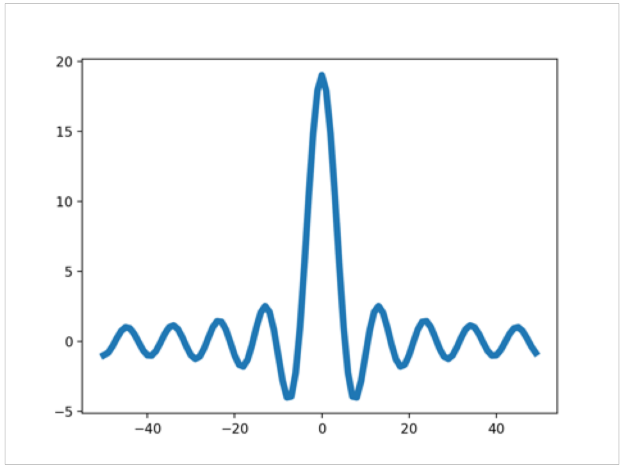
cos

Signal

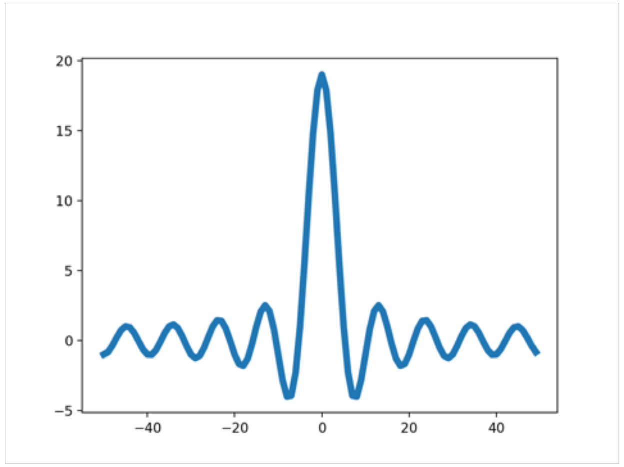
Fourier transform



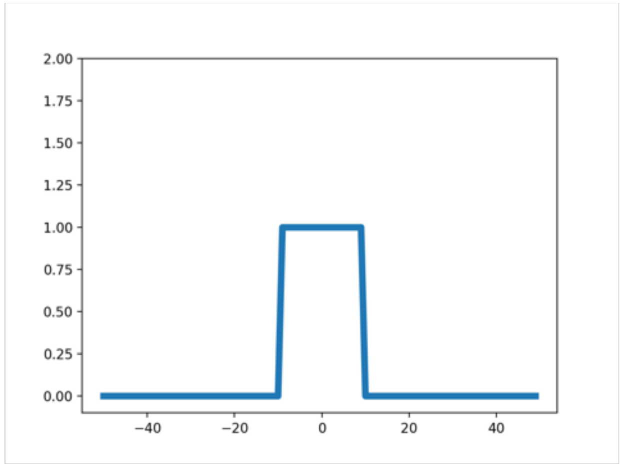
box



“sinc” = sin(x)/x



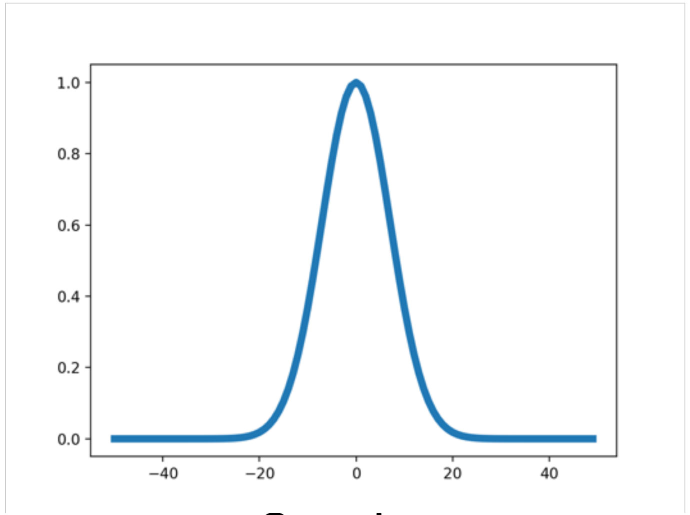
“sinc” = sin(x)/x



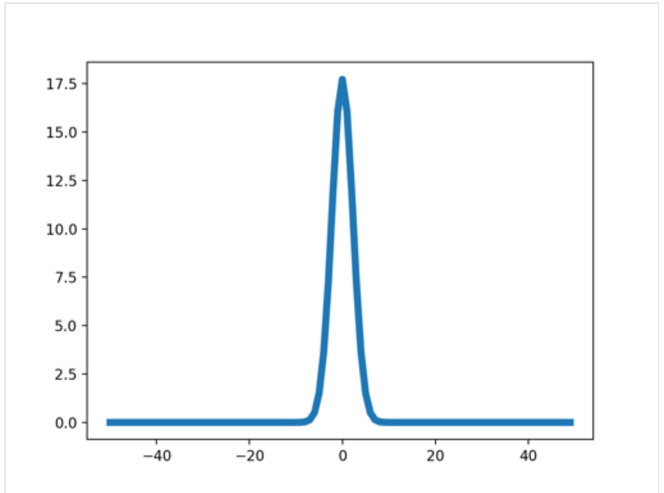
box

Signal

Fourier transform



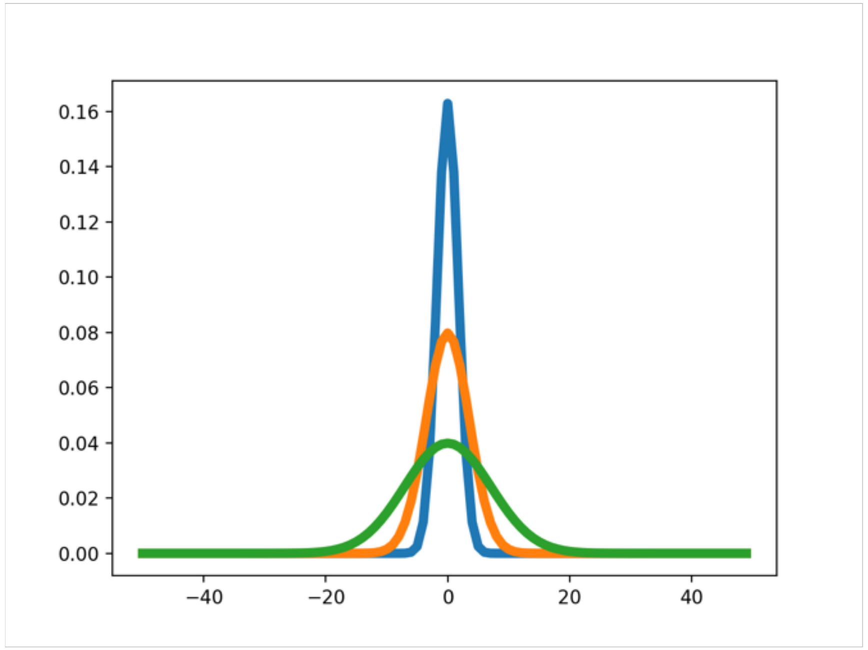
Gaussian



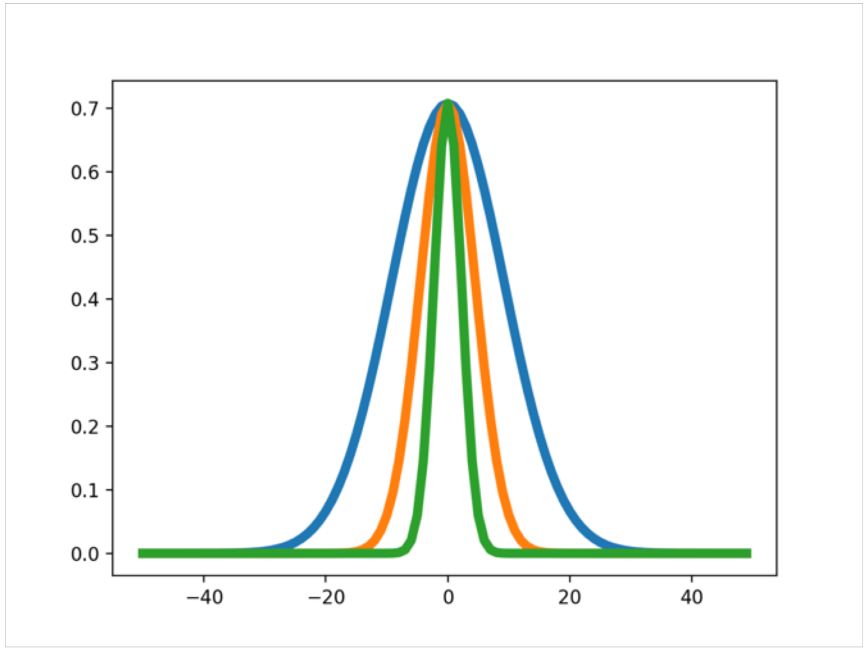
Gaussian

Signal

Fourier transform



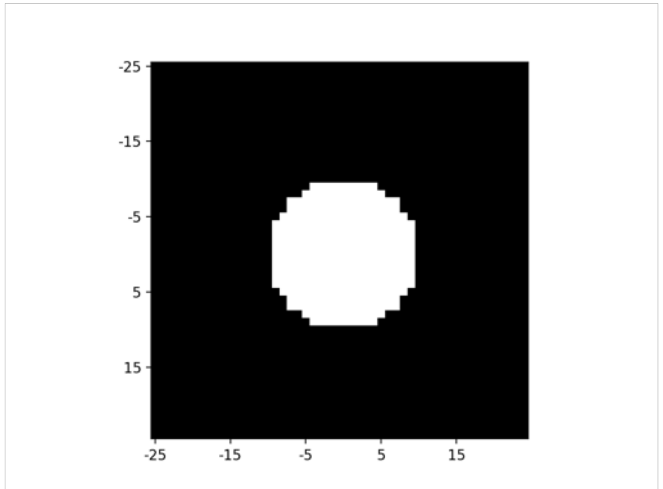
Gaussian

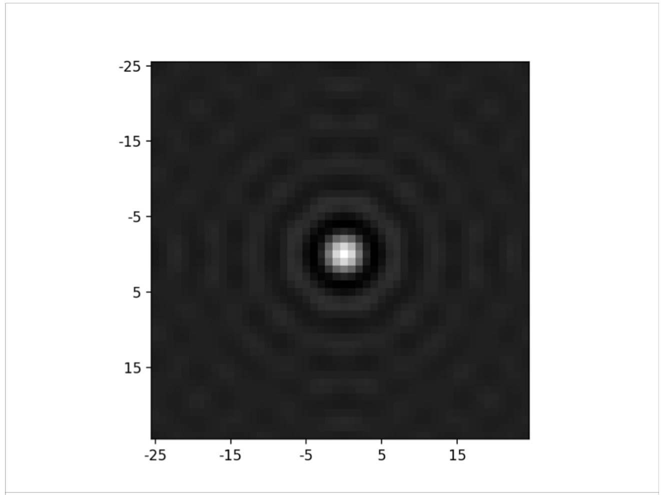


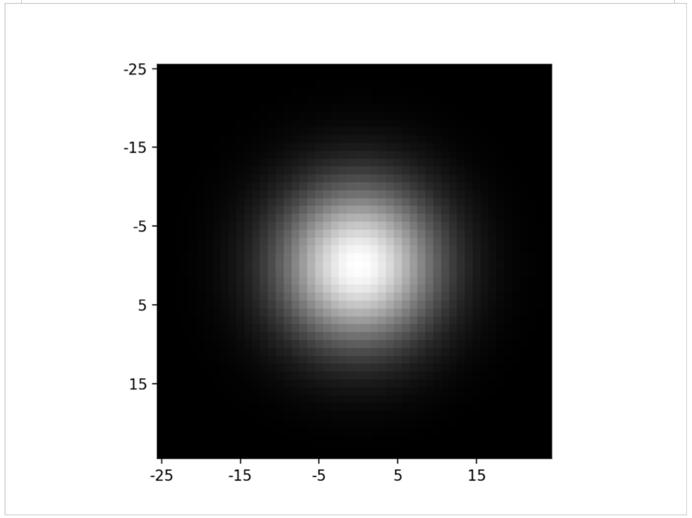
Gaussian

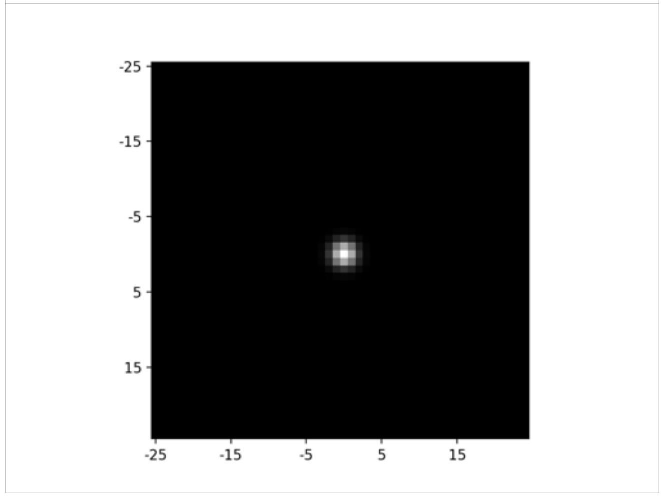
Image

Fourier transform

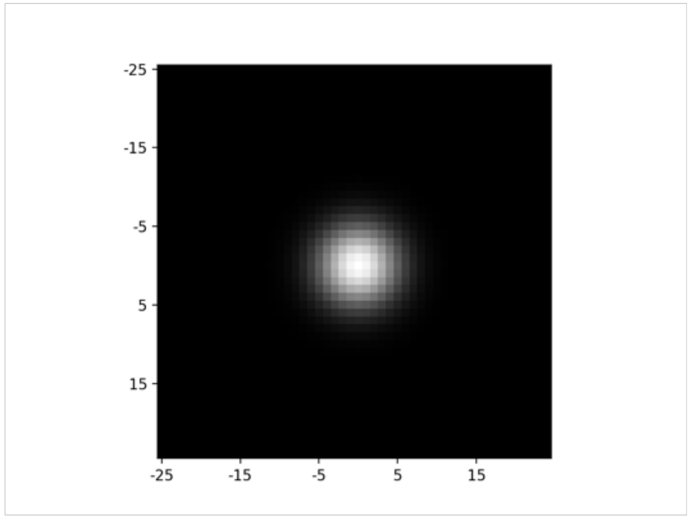


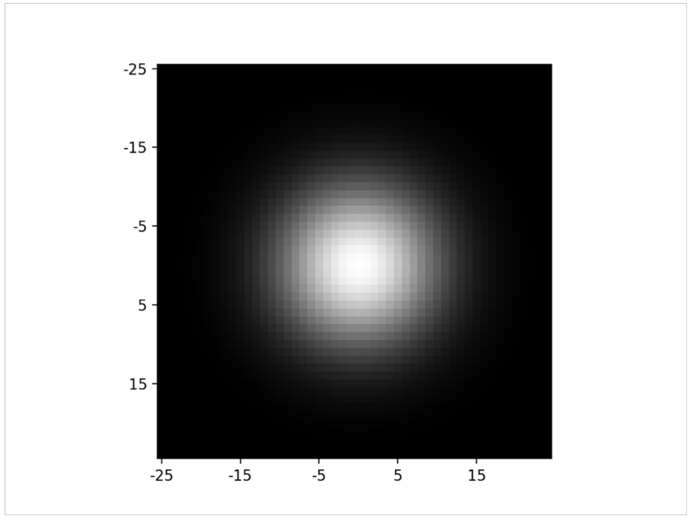




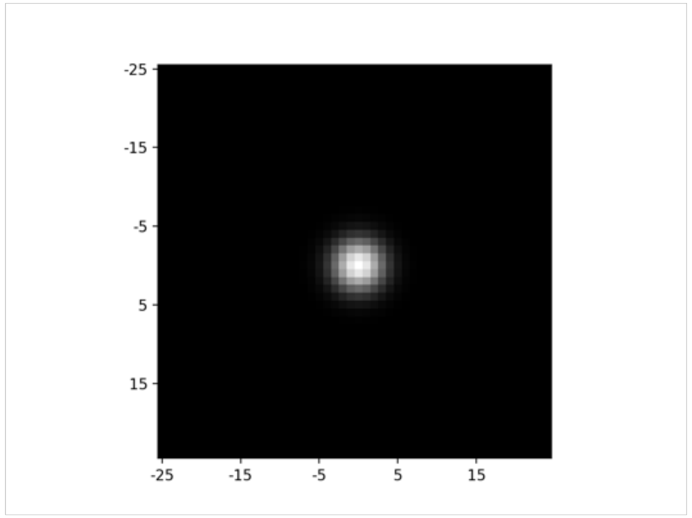


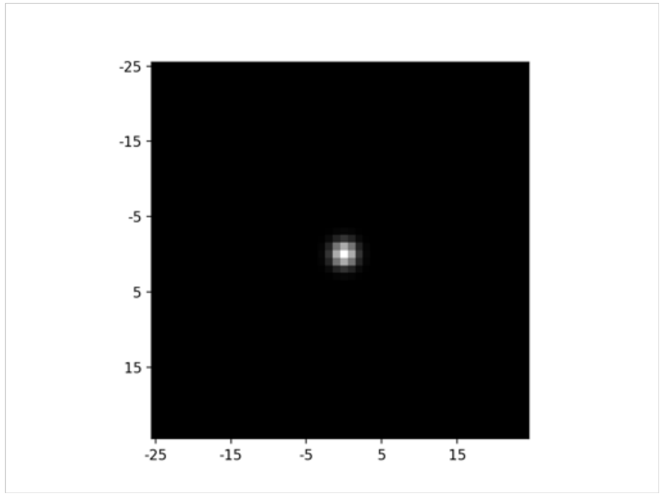
Image





Fourier transform





Detour: Time complexity of convolution

- Image is w x h
- Filter is k x k
- Every entry takes $O(k^2)$ operations
- Number of output entries:
  - (w+k-1)(h+k-1) for full
  - wh for same
- Total time complexity:
  - $O(whk^2)$

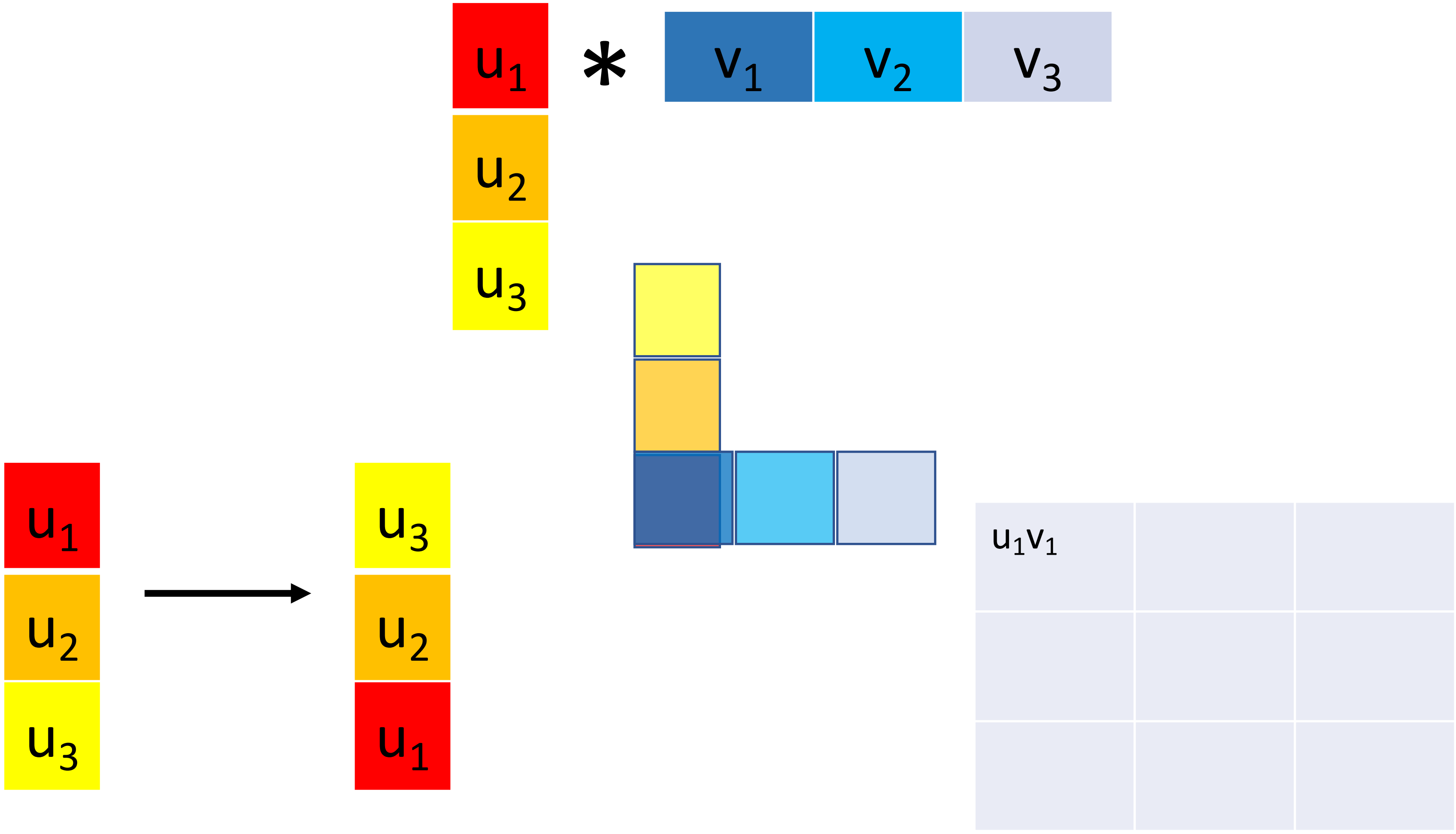
Optimization: separable filters

- basic alg. is $O(r^2)$: large filters get expensive fast!
- definition: $w(x,y)$ is *separable* if it can be written as:

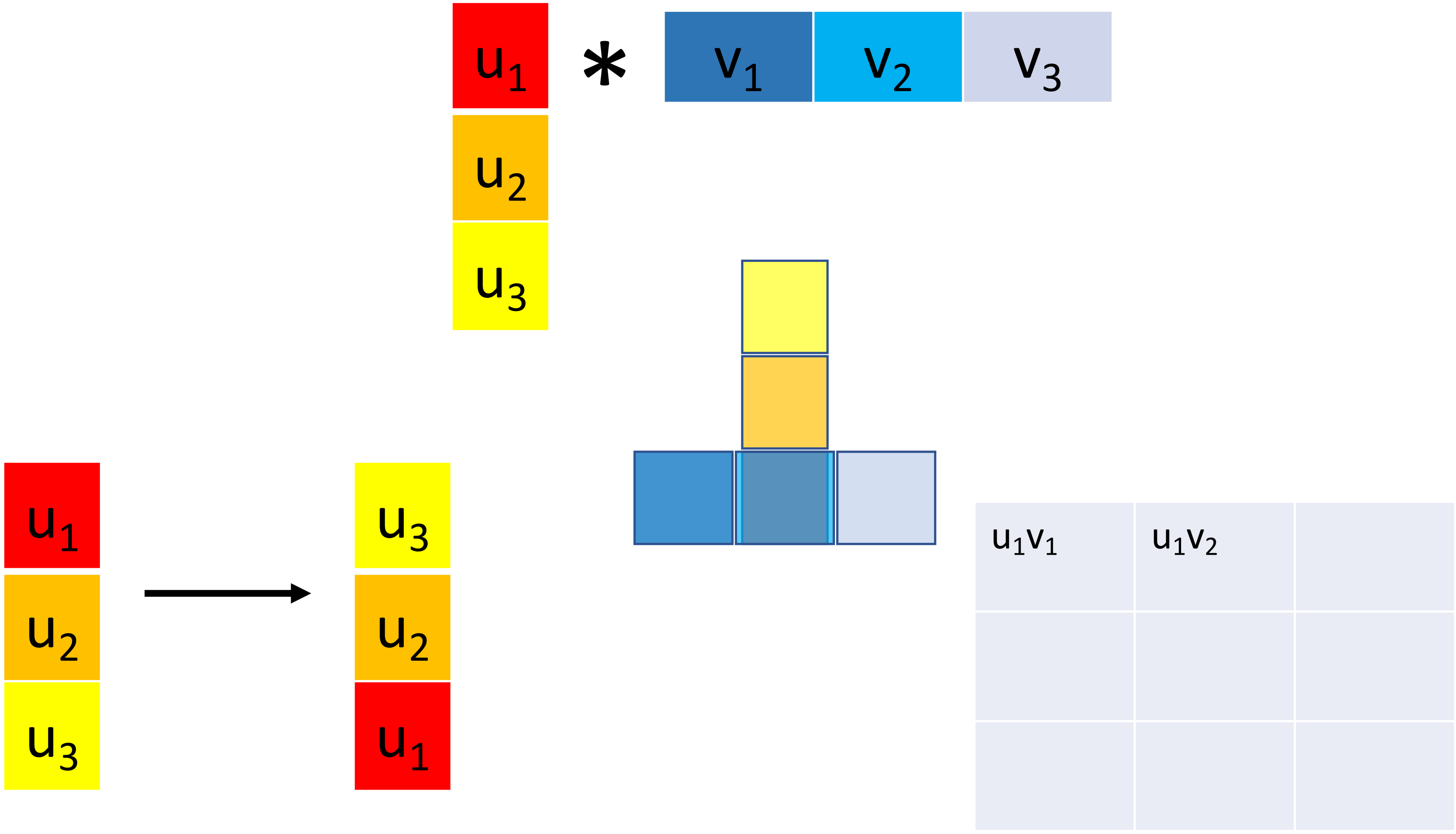
$$w(i, j) = u(i)v(j)$$

- Write u as a k x 1 filter, and v as a 1 x k filter
- Claim: $w = u * v$

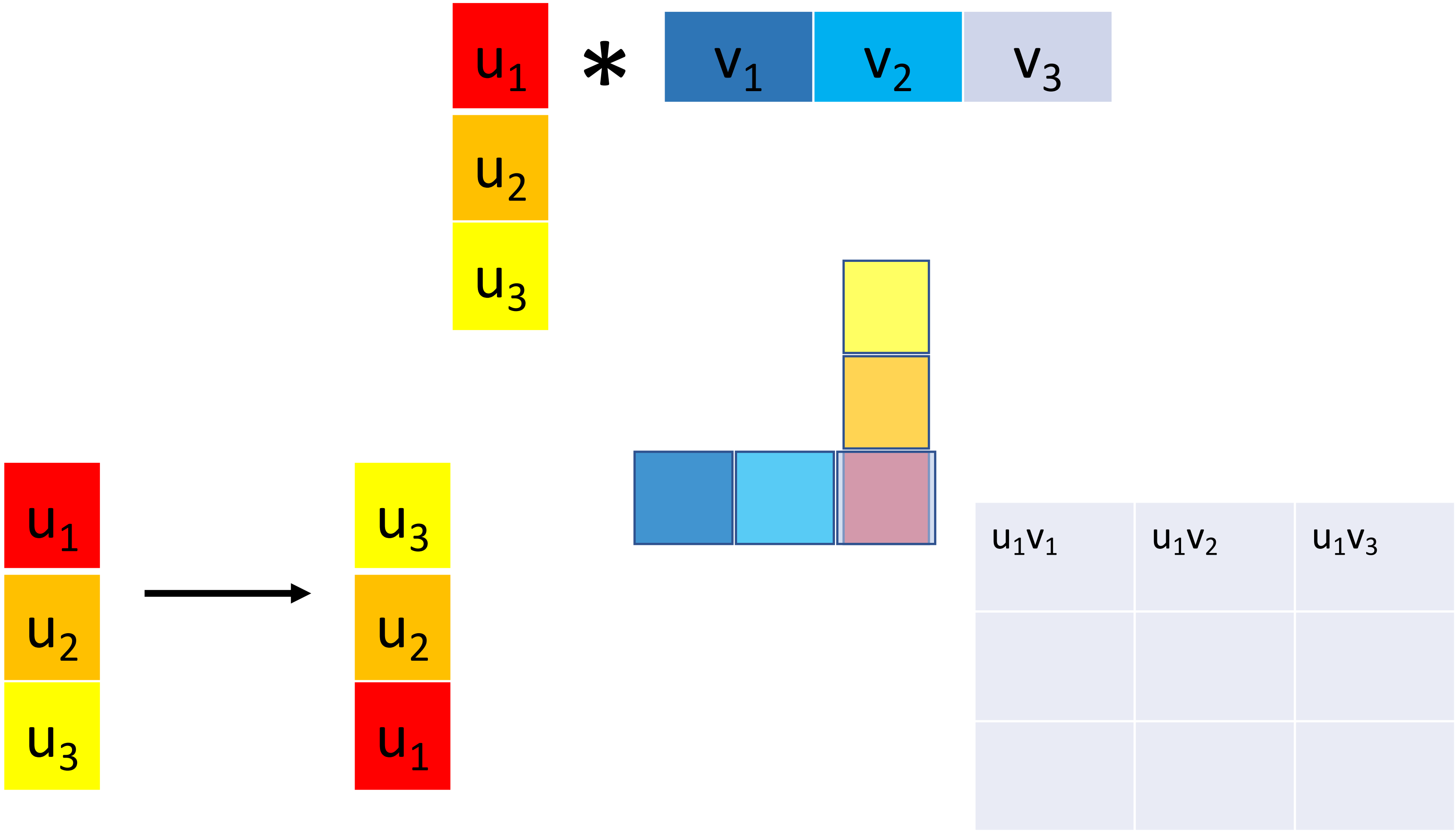
Separable filters



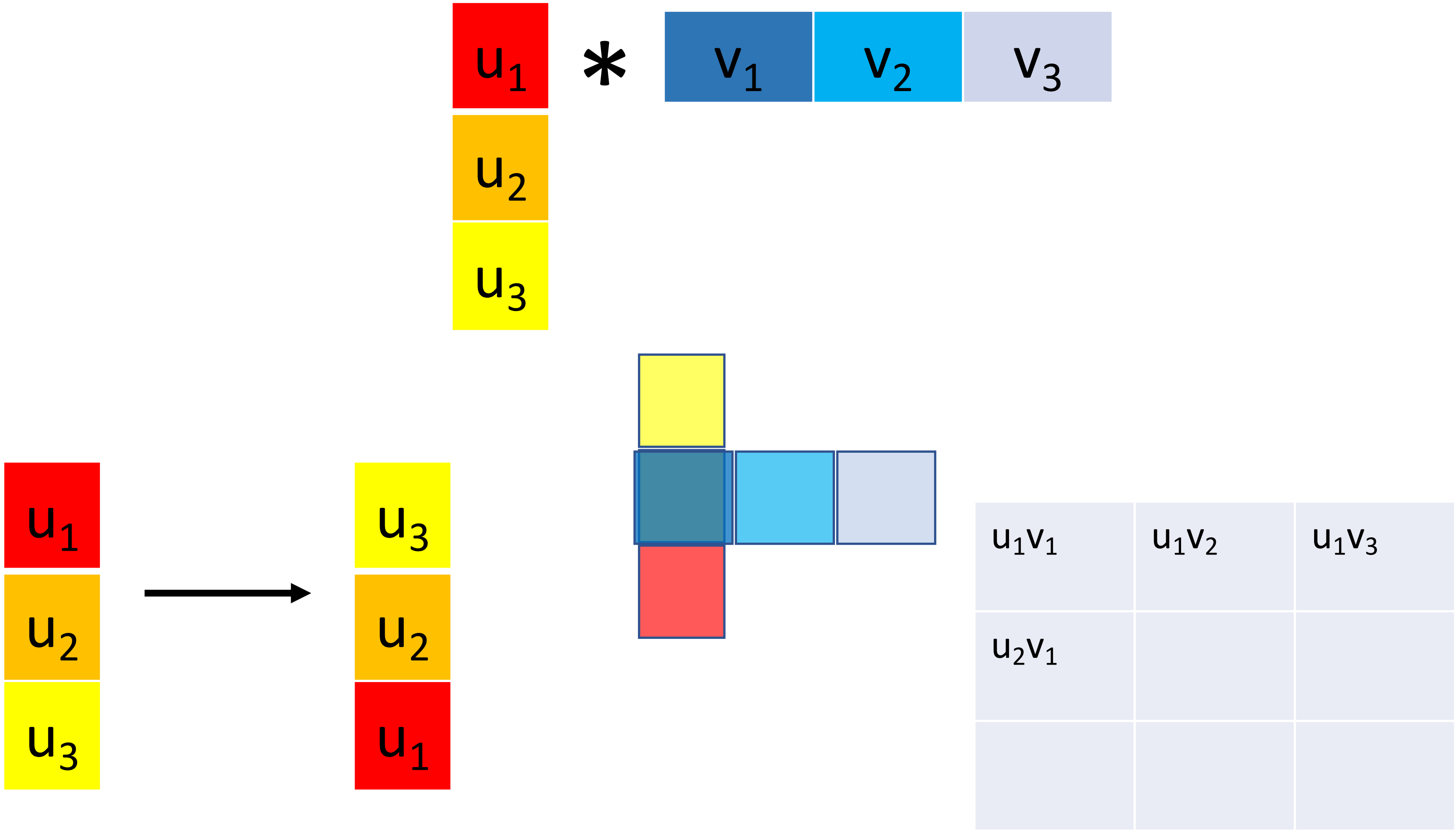
Separable filters



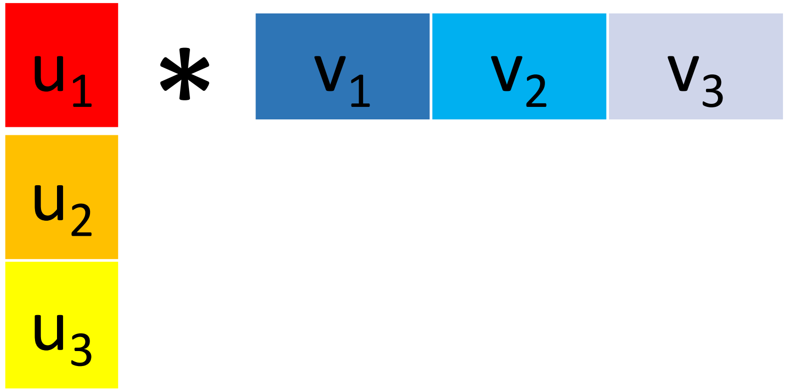
Separable filters

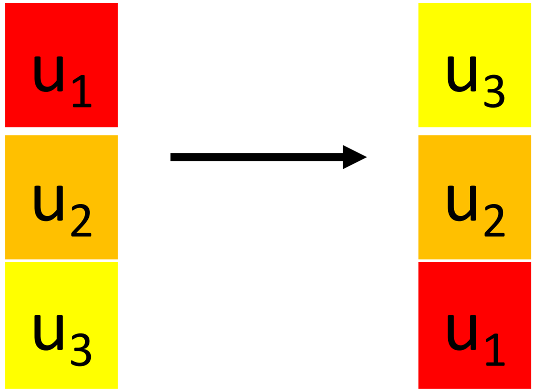


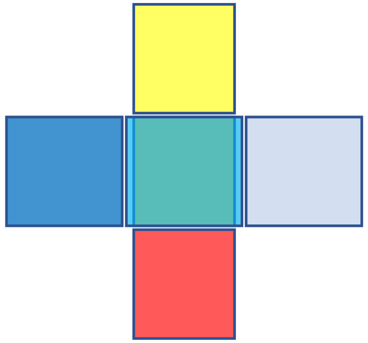
Separable filters



Separable filters

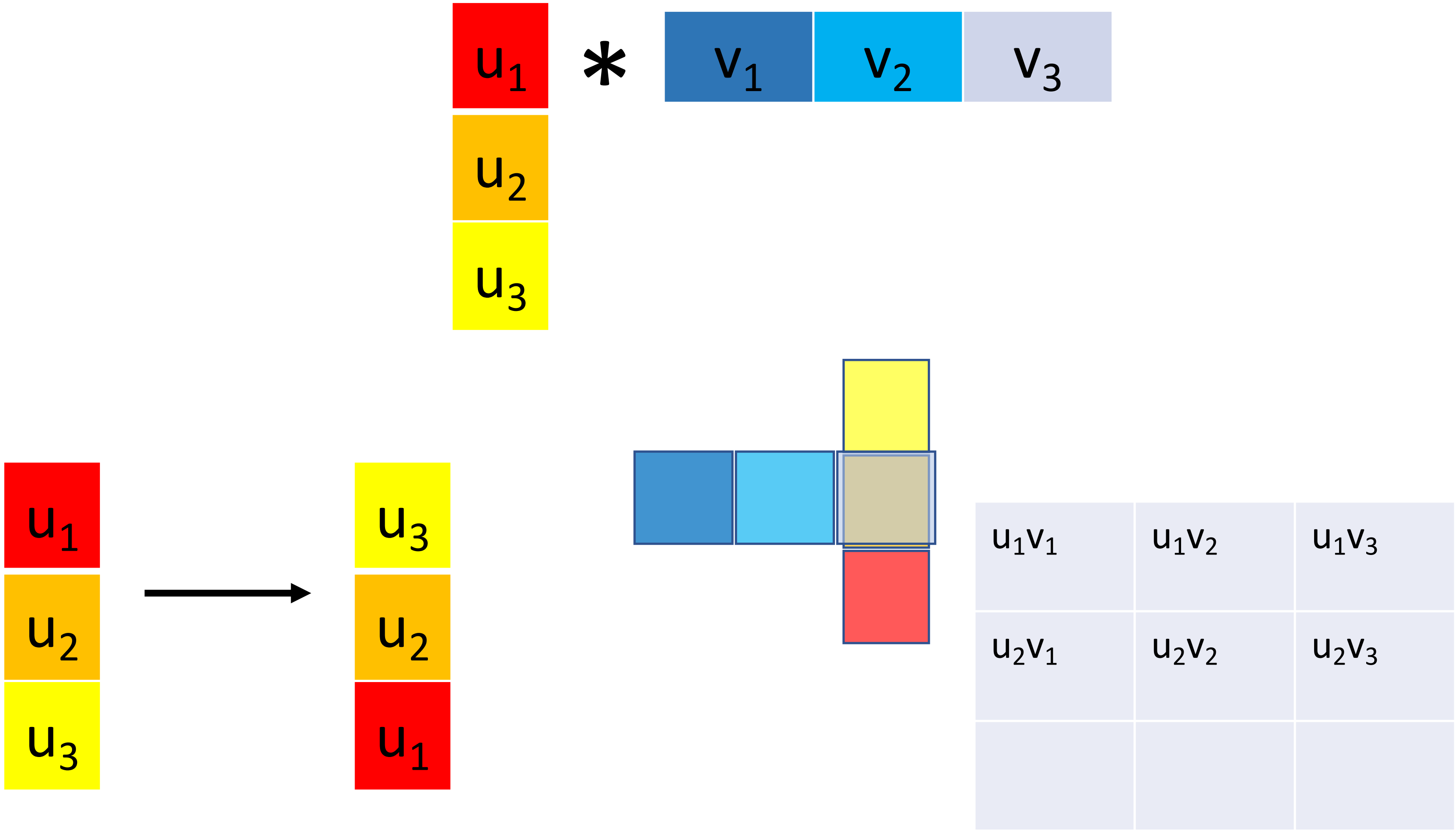




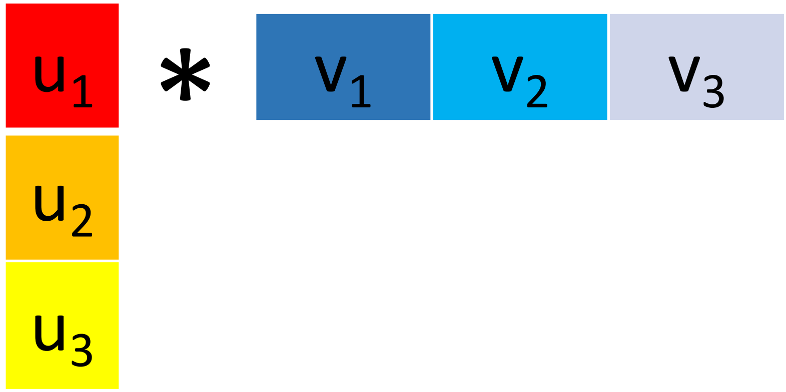


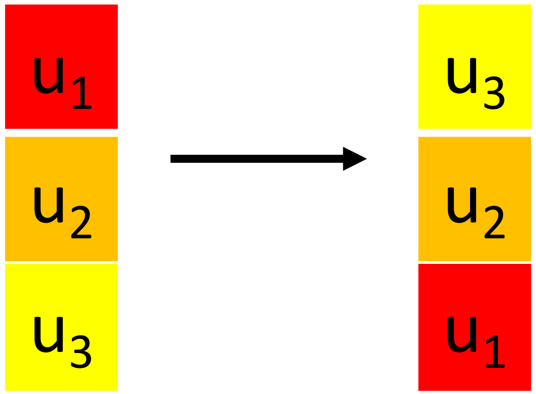
u_1v_1	u_1v_2	u_1v_3
u_2v_1	u_2v_2	

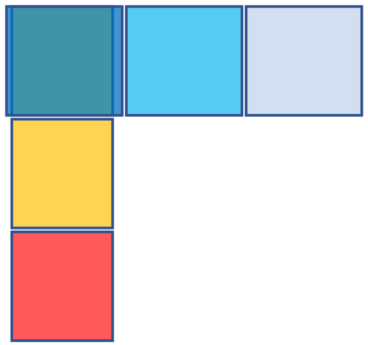
Separable filters



Separable filters

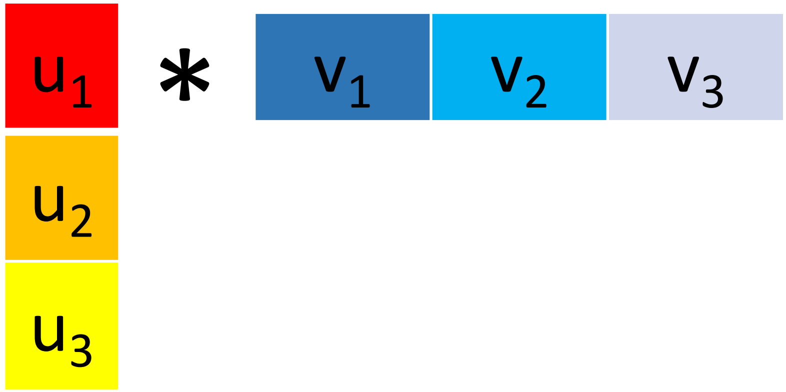


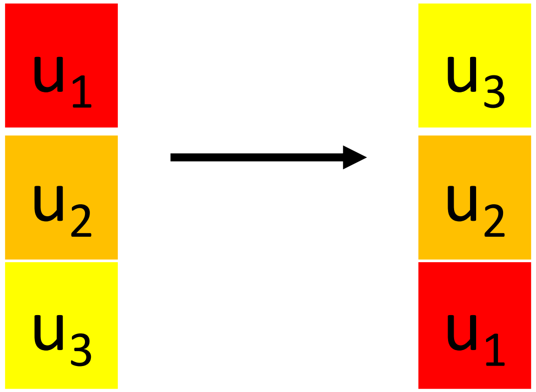


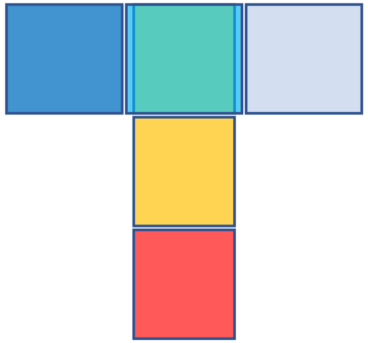


u_1v_1	u_1v_2	u_1v_3
u_2v_1	u_2v_2	u_2v_3
u_3v_1		

Separable filters

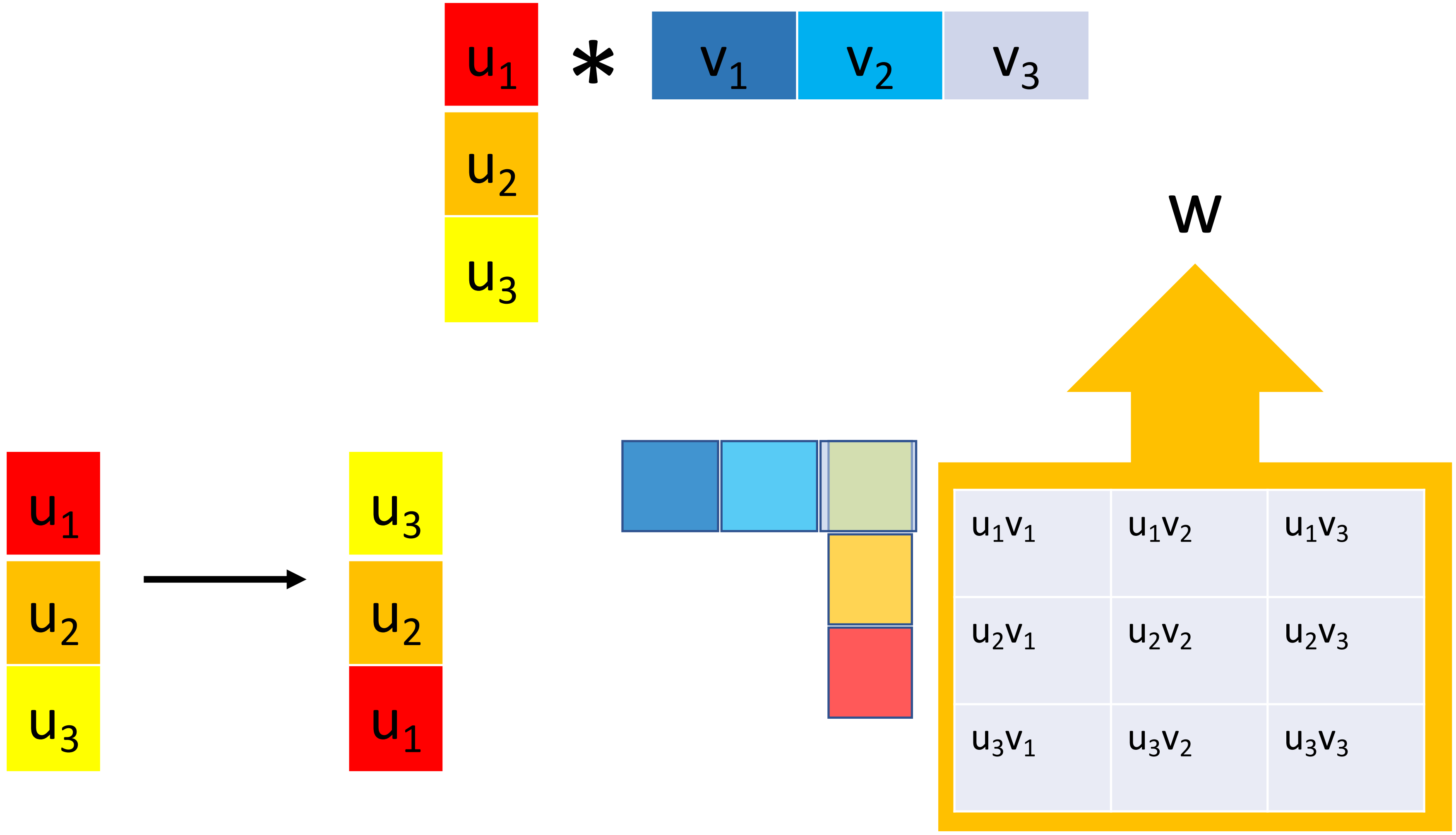






u_1v_1	u_1v_2	u_1v_3
u_2v_1	u_2v_2	u_2v_3
u_3v_1	u_3v_2	

Separable filters



Separable filters

$$
\begin{aligned}
w * f &= (u * v) * f \\
&= u * (v * f)
\end{aligned}
$$

- Time complexity of original : $O(whk^2)$
- Time complexity of separable version : $O(whk)$