2D Fourier Transform

CS 355: Introduction to Graphics and Image Processing

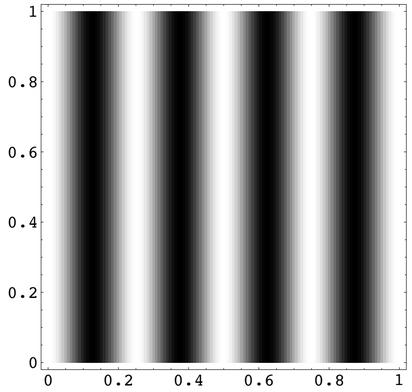
The 2D Discrete Fourier Transform

- Uses basis functions of two variables
  - u = frequency in the x direction
  - v = frequency in the y direction

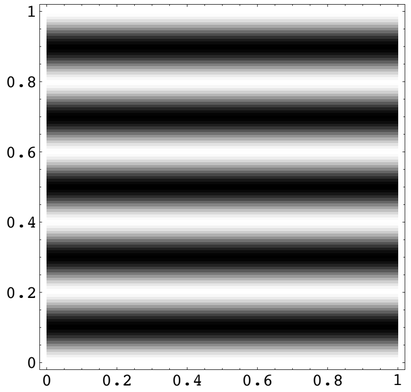
For an N x M image:

$$\cos(2\pi(ux/M + vy/N))$$

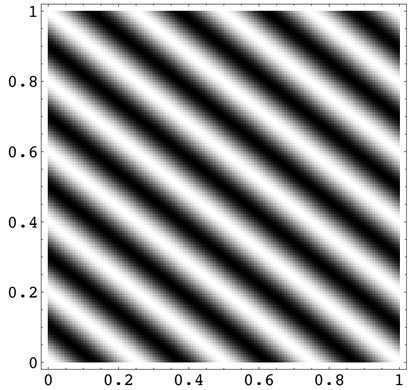
$$\sin(2\pi(ux/M + vy/N))$$



u = 4, v = 0



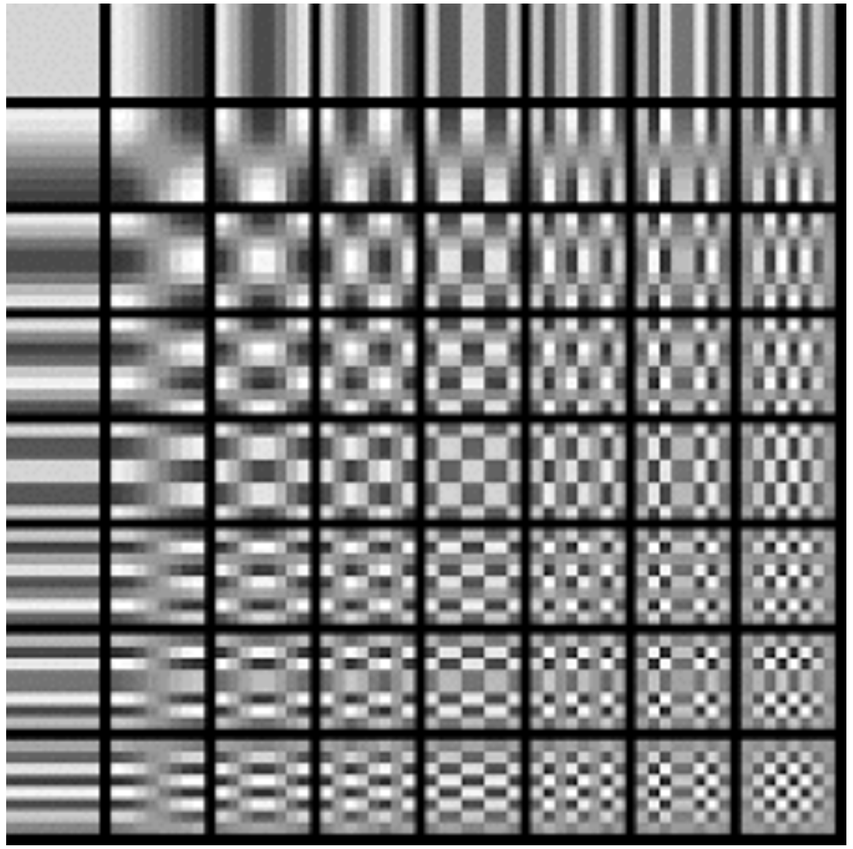
u = 0, v = 5



u = 4, v = 5

The 2D Discrete Fourier Transform

- Example: 64 basis images can be combined in a weighted sum to form **any** 8 x 8 image



* Technically, these are for the DCT, a close cousin to the DFT

Spatial Frequencies

- We call these different image components “spatial frequencies”
- Analogous to 1-D frequencies like audio
- Intuition:
  - low frequencies are slow, gradual changes
  - high frequencies are rapid changes (e.g., textures, edges)

The 2D Discrete Fourier Transform

$$
\begin{aligned}
F[u,v] &= \frac{1}{NM} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f[x,y] \ \cos(2\pi(ux/M + vy/N)) \ dx \ dy \\
&+ i \ \frac{1}{NM} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f[x,y] \ \sin(2\pi(ux/M + vy/N)) \ dx \ dy
\end{aligned}
$$

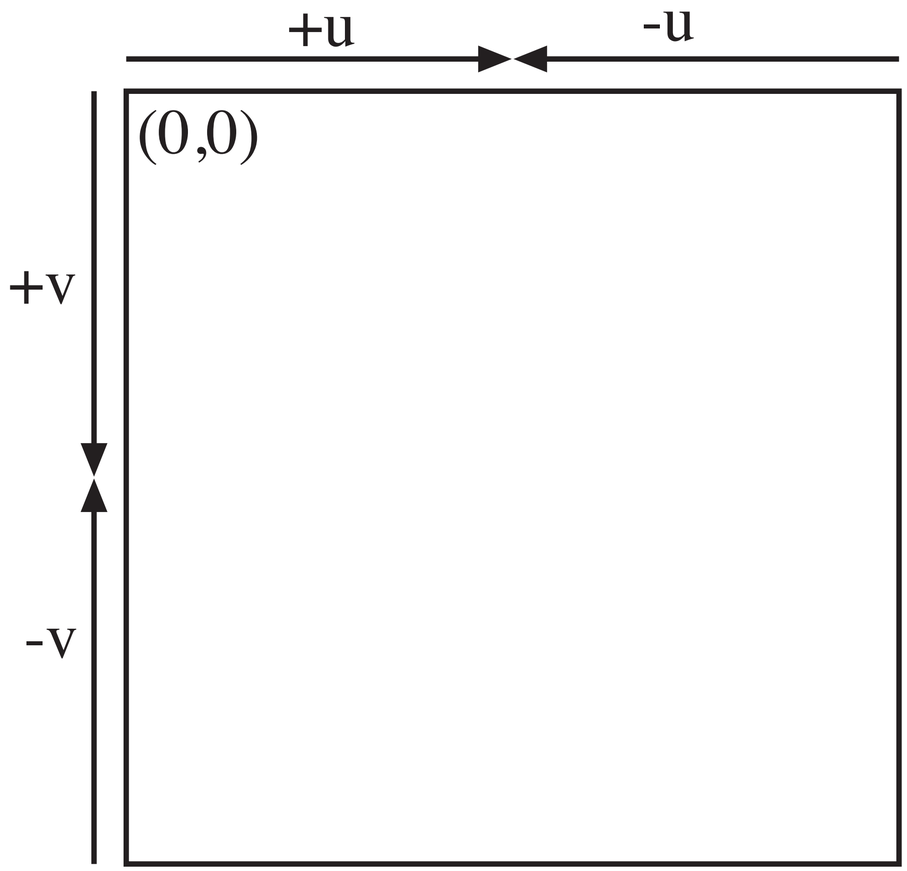
What's does the code look like?

What's the computational complexity?

The 2D Fast Fourier Transform

	DFT	FFT
1-D	$O(N^2)$	$O(N \log N)$
2-D	$O(N^4)$	$O(N^2 \log N)$

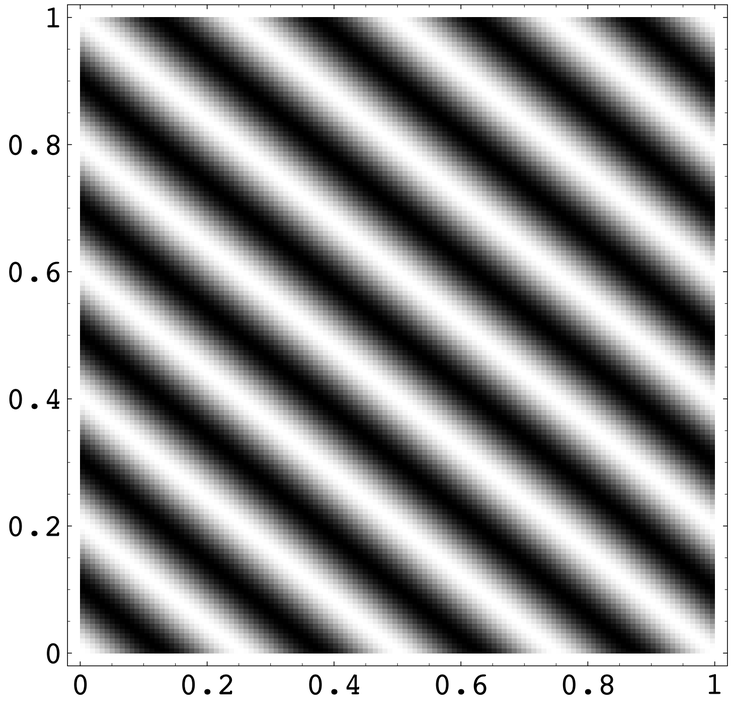
2D DFT Storage



(like 1-D, it's often useful to shift it to better visualize)

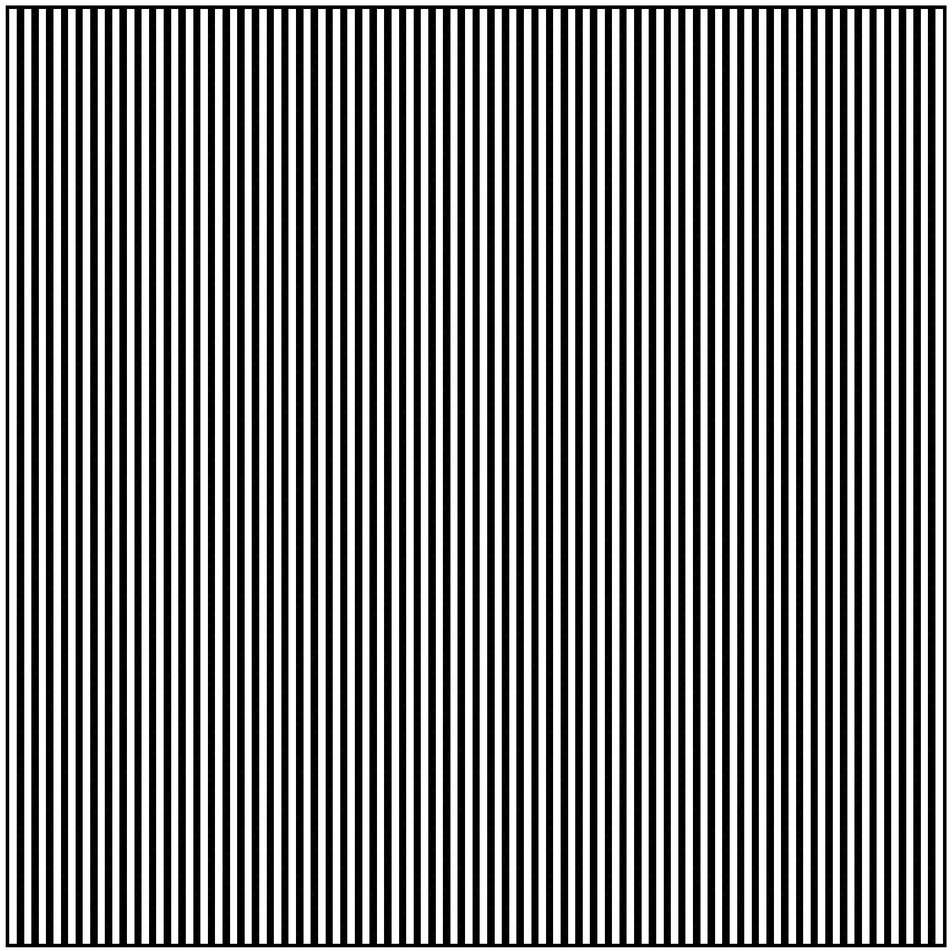
Interpreting the 2D DFT

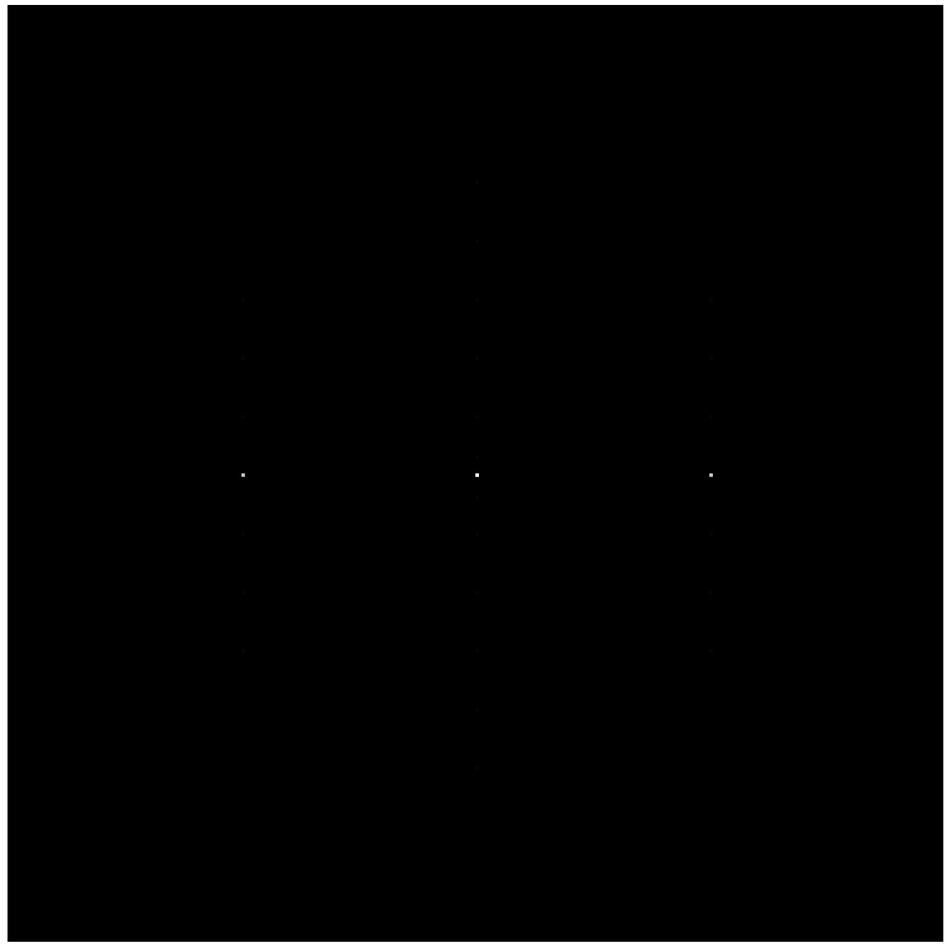
- Frequencies in x and y directions
  - u = frequency in the x direction
  - v = frequency in the y direction
- A sinusoidal pattern in a single direction
- Useful property: rotating the image rotates its Fourier Transform



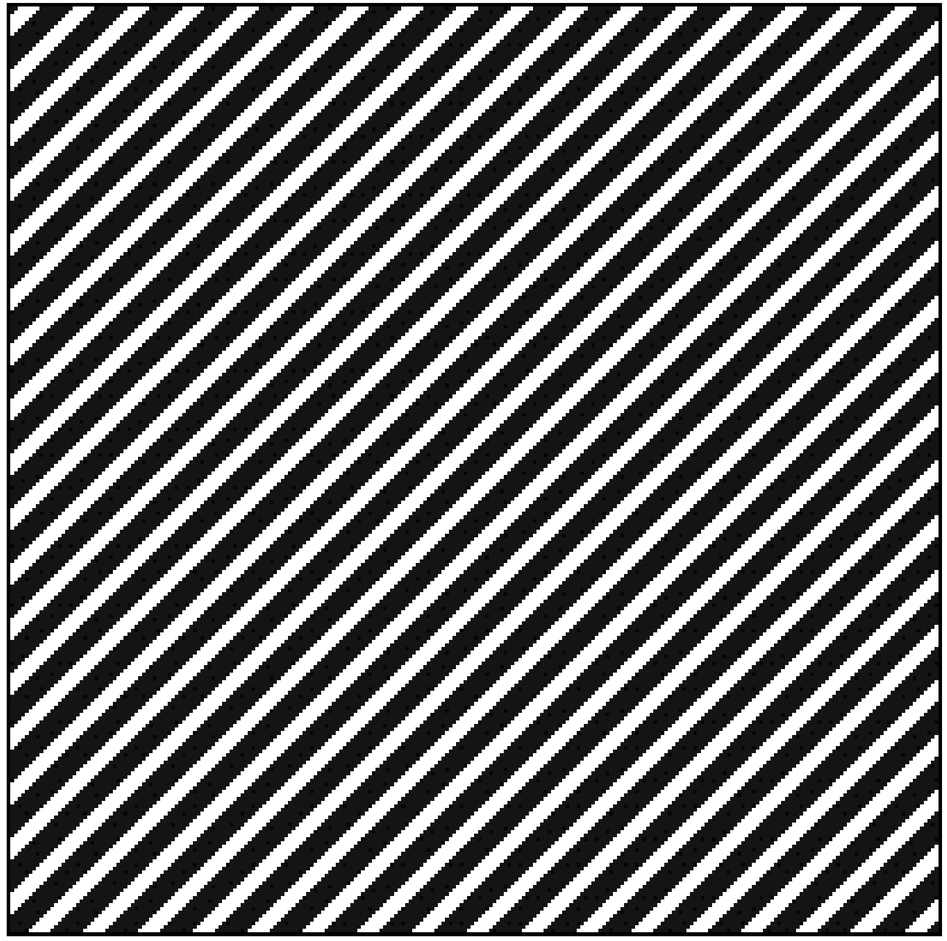
u = 4, v = 5

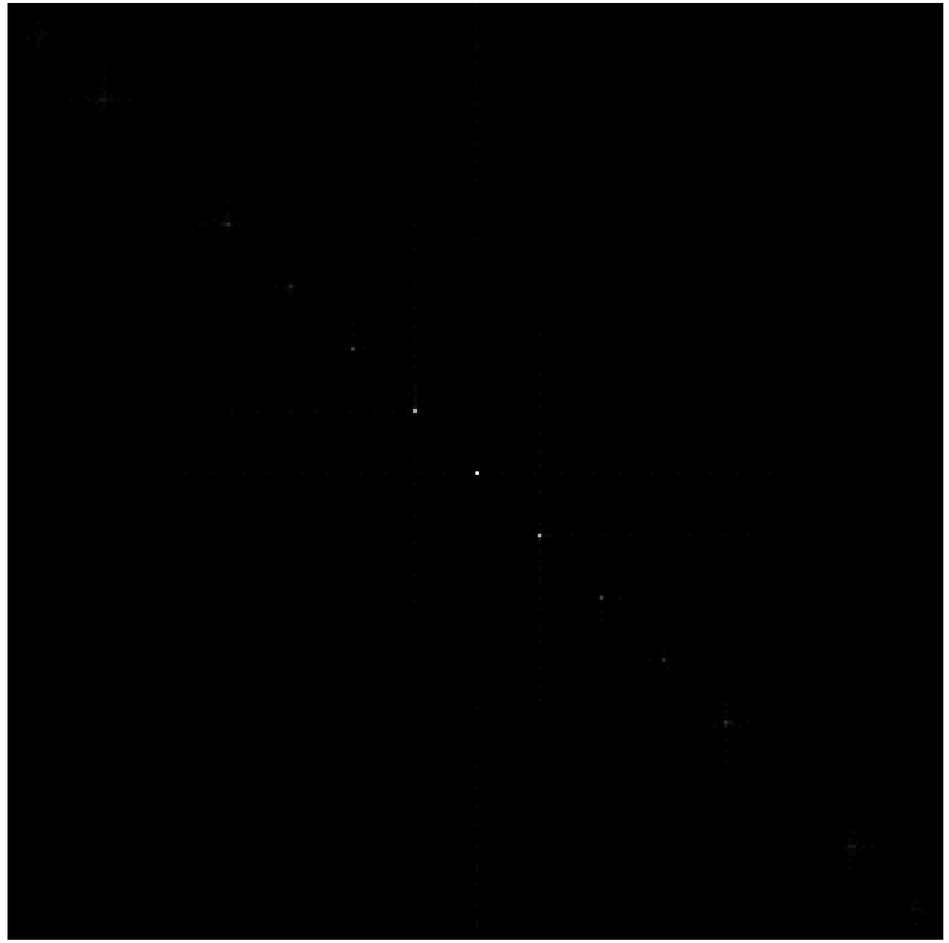
Examples





Examples

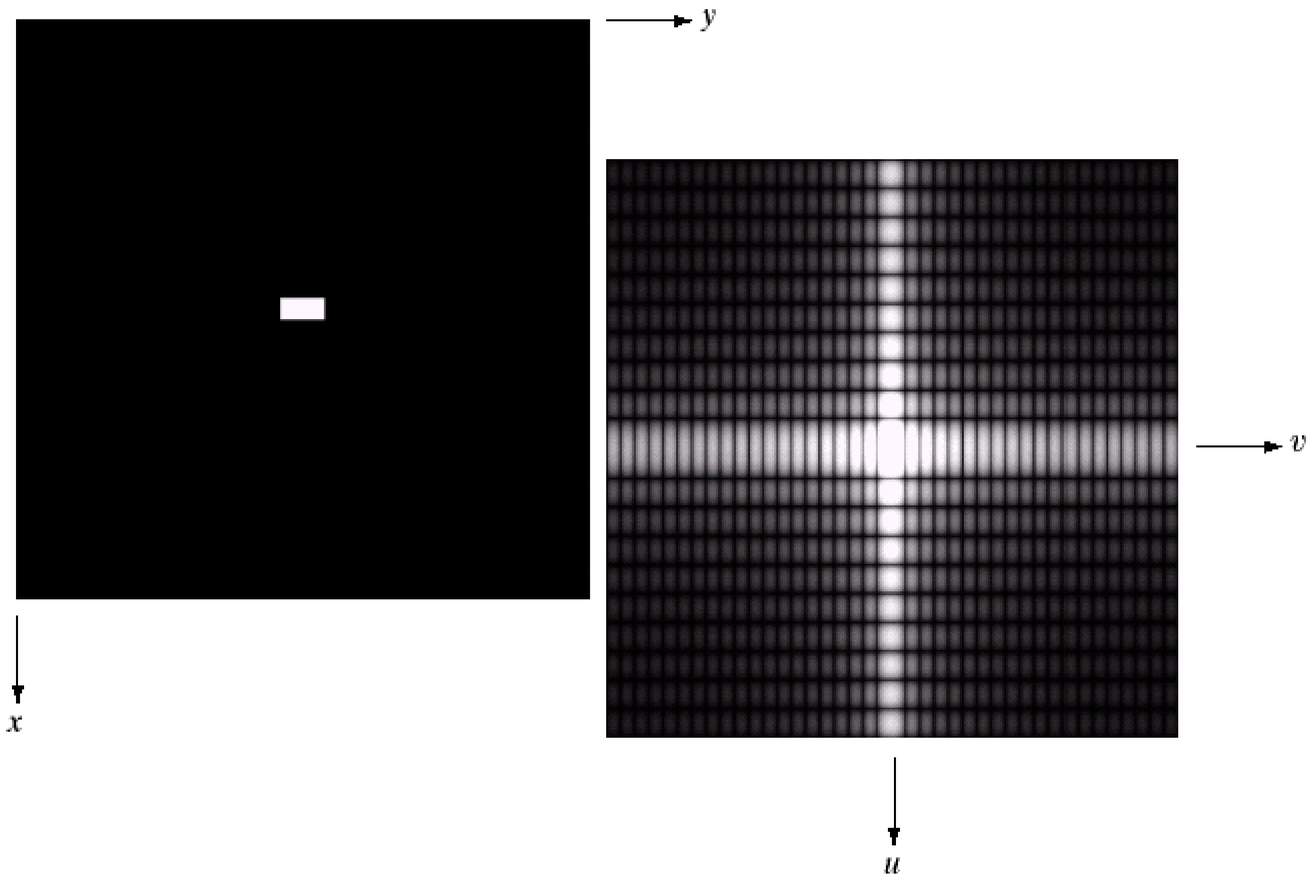




Examples

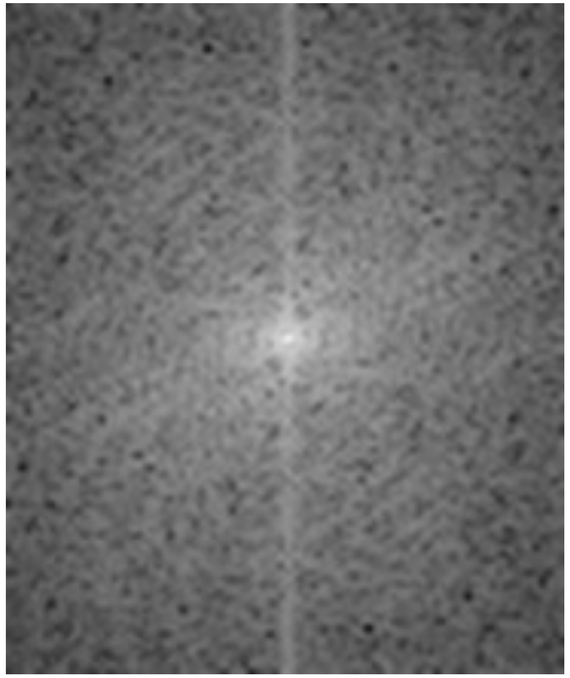
a b

FIGURE 4.3
(a) Image of a 20×40 white rectangle on a black background of size 512×512 pixels.
(b) Centered Fourier spectrum shown after application of the log transformation given in Eq. (3.2-2). Compare with Fig. 4.2.



Another Example

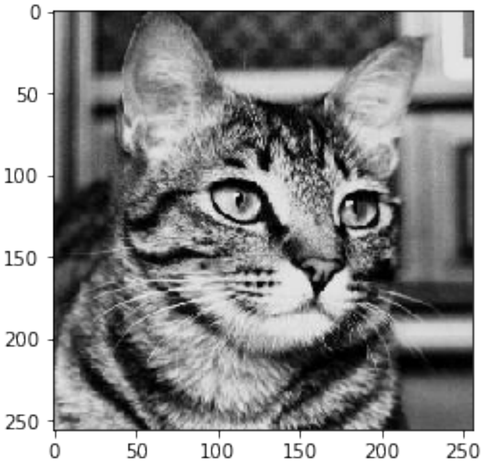
Jean Baptiste
Joseph Fourier

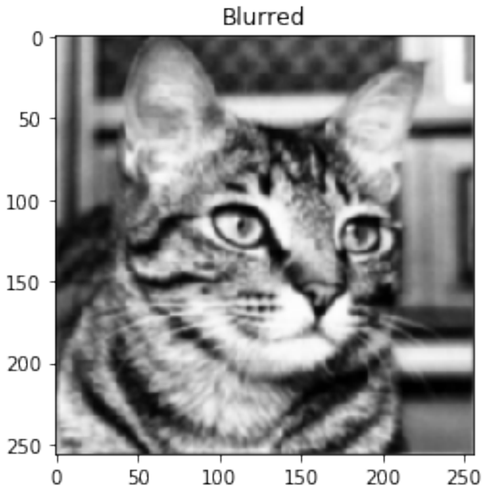


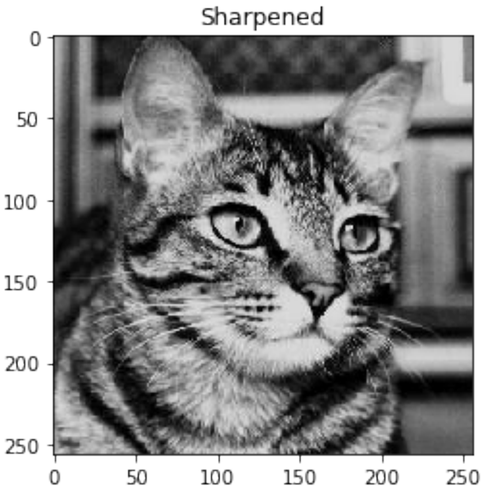
His Fourier Transform

Filtering Images in the Frequency Domain

- Just like we can filter 1-D signals, we can also filter 2-D images
  - Fourier transform the image
  - Multiply by the filter
  - Inverse Fourier Transform
- Low-pass filtering = blurring
- High-pass filtering = edge detection
- High-boost filtering = sharpening





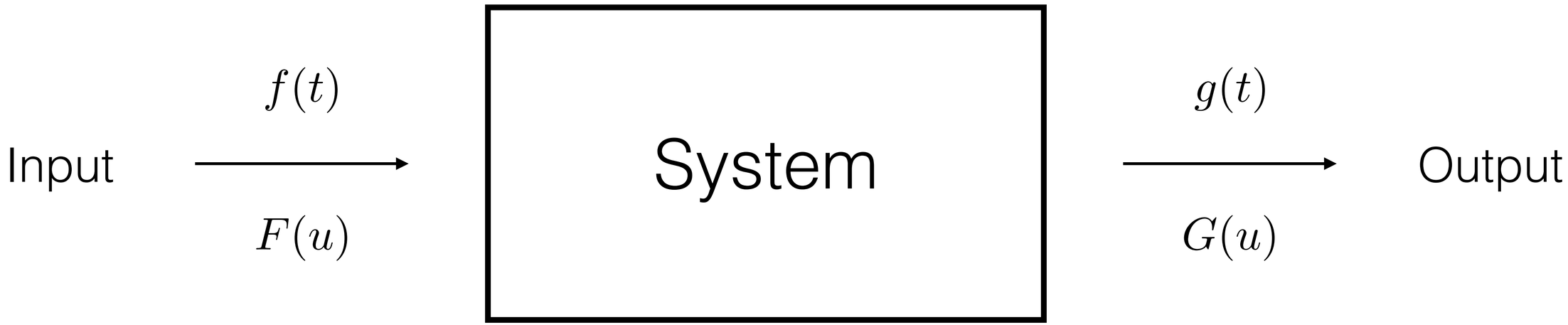


A really important detour…

Time Domain vs. Frequency Domain

Time Domain

What's the equivalent?



Frequency Domain

$$G(u) = F(u)\ H(u)$$

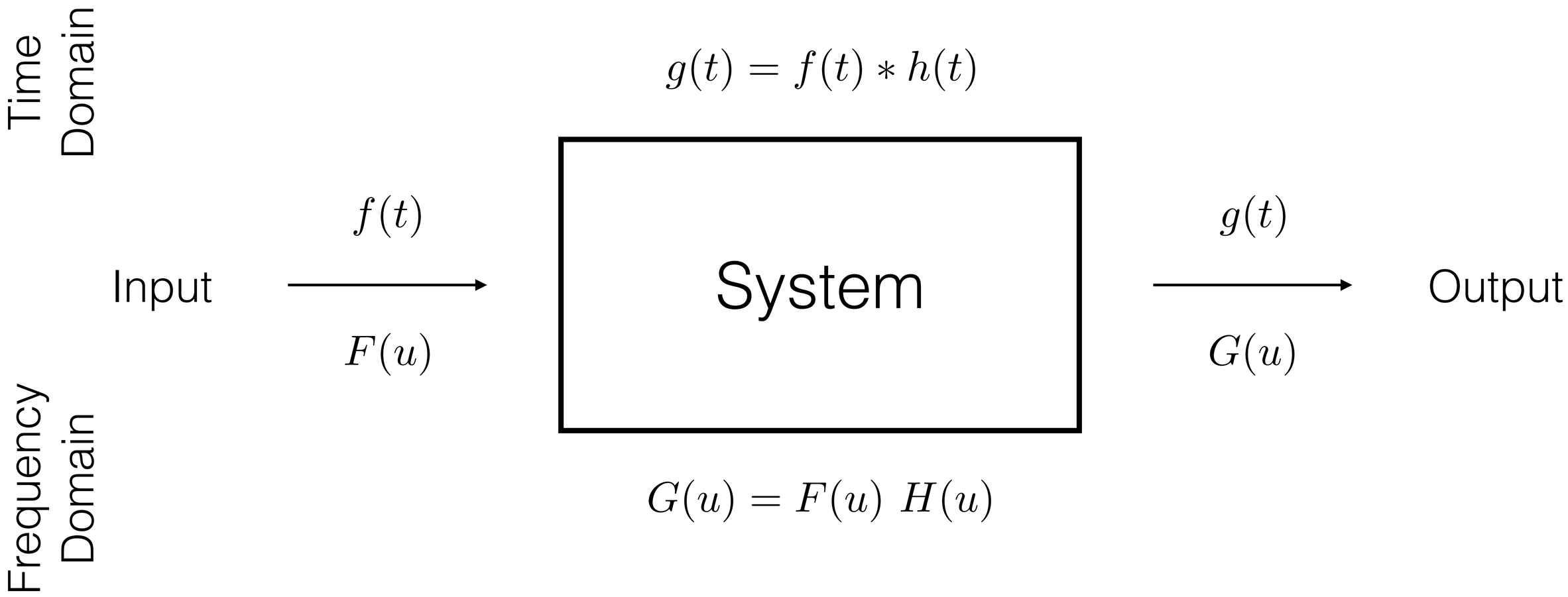
The Convolution Theorem

- **Convolution in one domain** is **multiplication in the other** and vice versa
  - Convolution in the time domain and multiplication in the frequency domain are equivalent
  - Multiplying two signals in the time domain convolves their FTs (useful for lots of things)
- Relationship between convolution kernels and transfer functions?

$$f(t) * g(t) = \mathcal{F}(f(t)\ \mathcal{F}(g(t))$$

$$f(t)\ g(t) = \mathcal{F}(f(t) * \mathcal{F}(g(t))$$

Time Domain vs. Frequency Domain

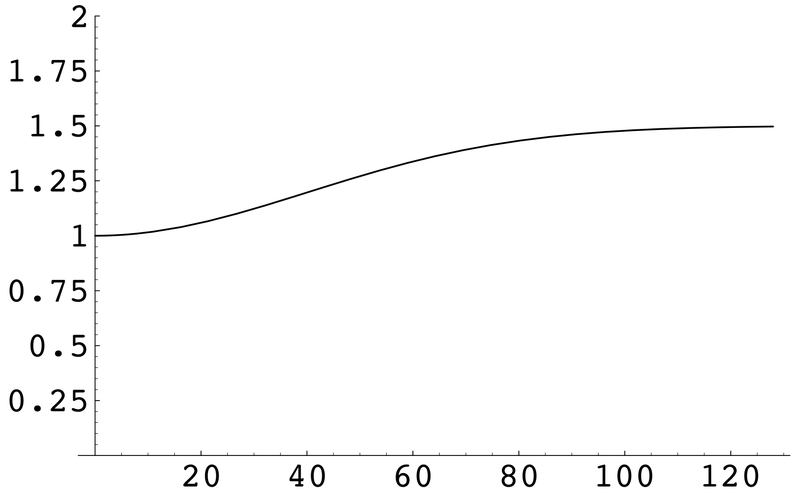


Example: High Boost Filters

- Can construct a high-boost filter by
  - Low-pass filtering the signal
  - Subtracting it from the original
  - Adding a fraction of that back to the original

$$H(u) = 1 + \alpha(1 - e^{-\frac{1}{2}u^2/{u_c}^2})$$

- Sound familiar?



The Need For Speed…

- Convolution of an N x N image with a K x K kernel:

$$O(N^2 K^2)$$

- FFT the signal, FFT the kernel, inverse FFT:

$$O(N^2 \log N)$$

- For large kernels, doing it in the frequency domain can be faster!

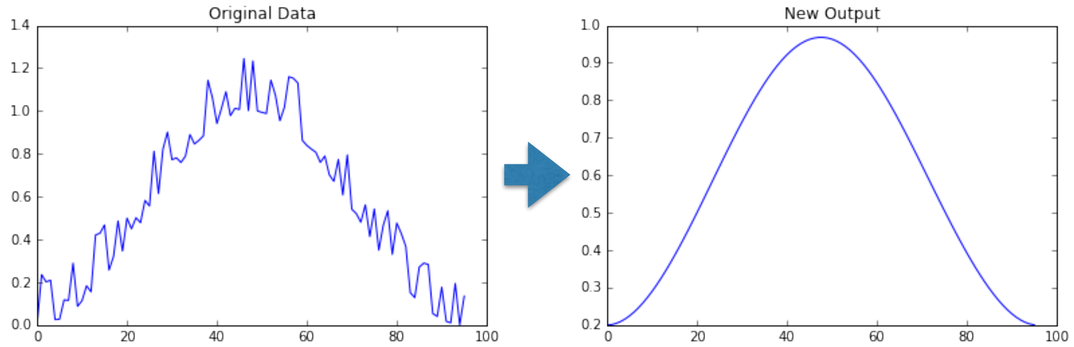
- General rule of thumb, the break-even point is 7 x 7 kernels

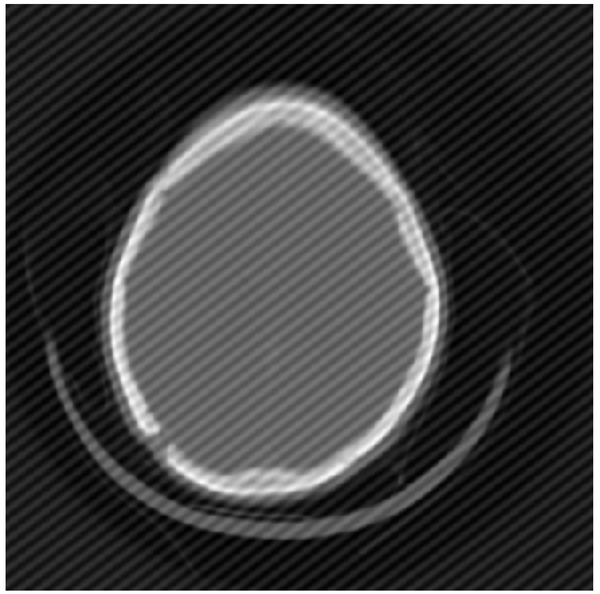
Design vs. Implementation

- We can design in either domain
  - Spatial filters in (masks or kernels) in the time/spatial domain
  - Frequency filters in the frequency domain
- And we can implement them in either domain
  - Convolution in the time/spatial domain
  - Multiplication in the frequency domain

Lab 10

- Filtering 1-D signals
  - Synthetic example (noisy function)
  - Audio example
- Filtering 2-D images
  - Smoothing/sharpening
  - Isolating and removing a single-frequency interference pattern





Coming up...

- Other uses of these ideas in Computer Science
- Applications in Computer Graphics (and CS 455)
- Applications in Computer Vision (and CS 450)