Fast Fourier Transform

The fast Fourier transform (FFT) algorithm computes the discrete Fourier transform of a sequence of N points $F_k = \sum_n f_n e^{-2\pi ink/N}$ in O(nlogn) operations. In 1965, Cooley and Tukey exploited the underlying symmetry of F_k by rewriting it as the sum of two discrete Fourier transforms formed from the even and odd numbered points of N:

$$\mathbf{F_k = F_k^e + W_k F_k^o}$$

where $W = e^{2\pi i/N}$. F_k^e and F_k^o are periodic in k with length N/2. They reapplied the above rule to F_k^e and F_k^o independently as well as to all the successive discrete transforms until the sequence was subdivided down to transforms of length 1. As the Fourier transform of a single number is the number itself, this divide and conquer approach scales as O(nlogn) as long as N is a power of 2. If N is not a power of 2, the input sequence can be padded with zeros up to the next power of 2.

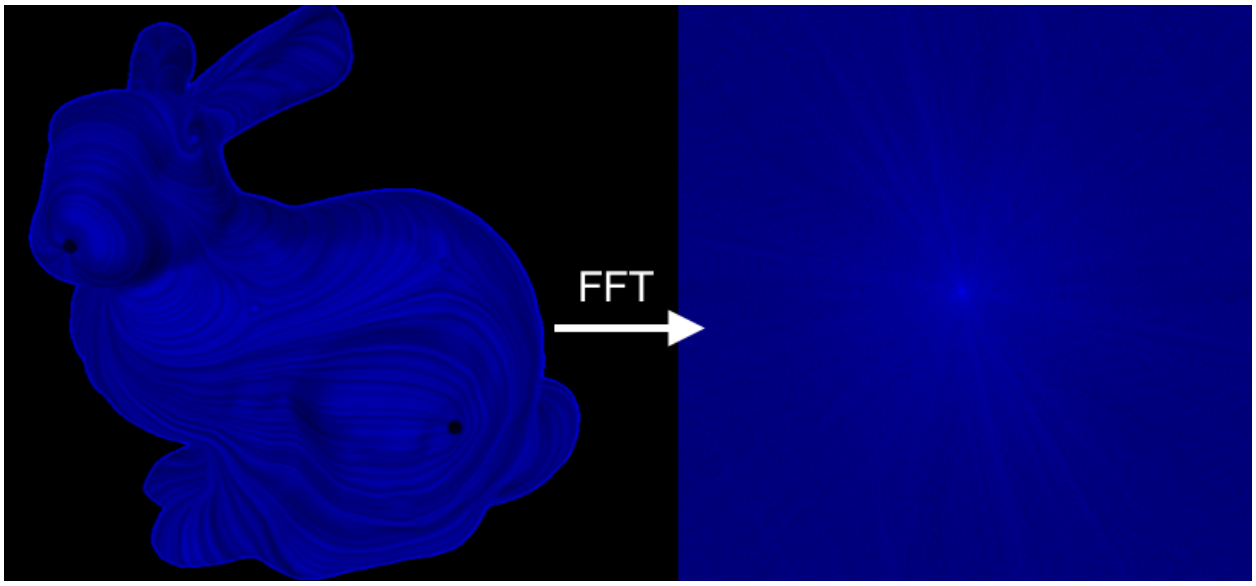

Figure 1: *Left:* Image of a bunny. Right: Its frequency spectrum

The FFT algorithm can be extended to higher dimensions. For example, the 2D FFT for a function f(x, y) requires doing a 1D FFT in x for each value of y followed by doing a 1D FFT in y for each value of k_x, i.e.,

$$\mathbf{F(k_x, y) = \sum_n f(x, y)\, e^{-2\pi inkx/N}}$$

and then

$$\mathbf{F(k_x, k_y) = \sum_n f(k_x, y)\, e^{-2\pi inky/N}}$$

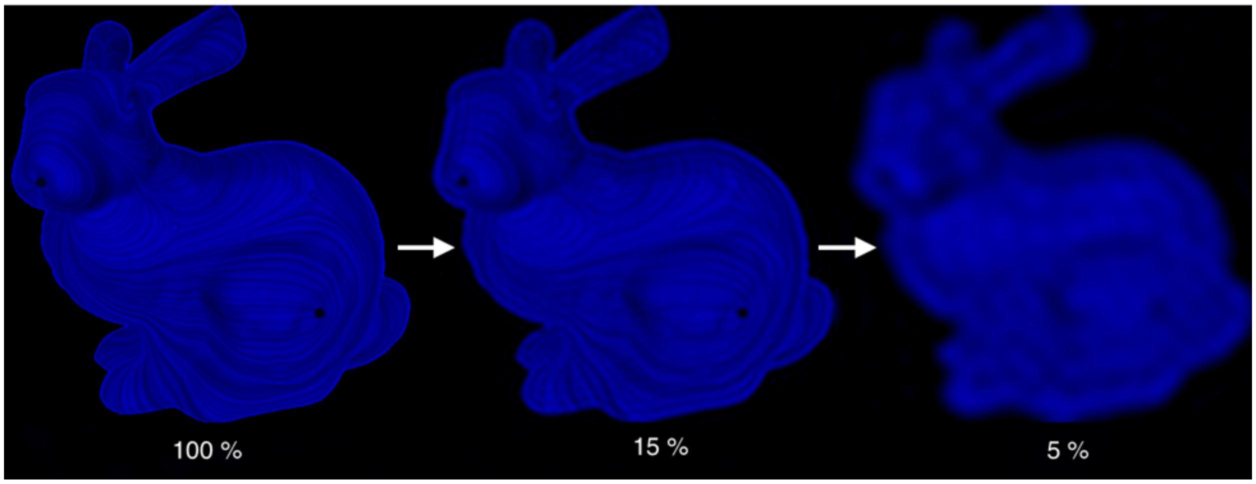

Figure 2: Image of the bunny reconstructed with successively fewer frequencies