Discrete Fourier Transform

Jean-Baptiste Joseph **Fourier**
1768~1830

Wonyong Sung

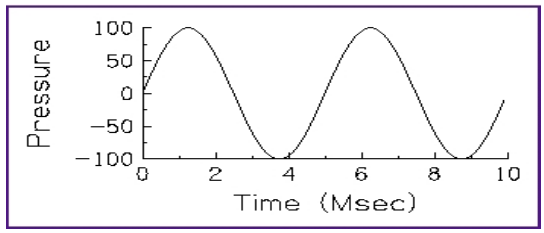
Fourier Transform

- Property of transforms:
  - They convert a function from one domain to another with no loss of information
- Fourier Transform: converts a function from the time (or spatial) domain to the frequency domain

$$F(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-i\omega t}\, dt$$

Time Domain and Frequency Domain

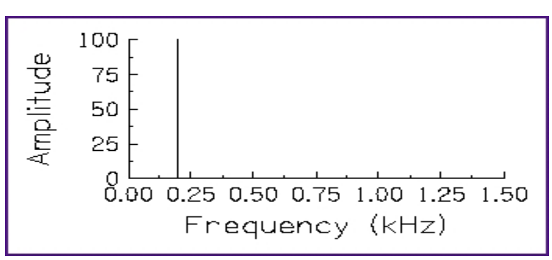
- Time Domain:
  - Tells us how properties (air pressure in a sound function, for example) change over time:



    - Amplitude = 100
    - Frequency = number of cycles in one second = 200 Hz

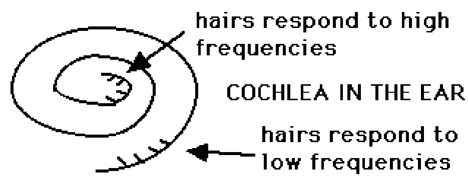
Time Domain and Frequency Domain

- Frequency domain:
  - Tells us how properties (amplitudes) change over frequencies:



Time Domain and Frequency Domain

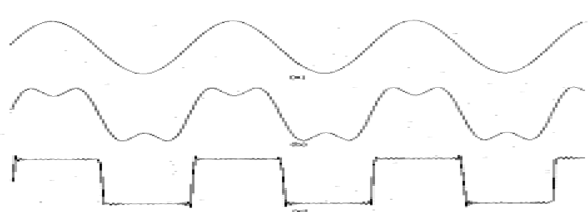
- Example:
  - Human ears do not hear wave-like oscillations, but constant tone



- Often it is easier to work in the frequency domain

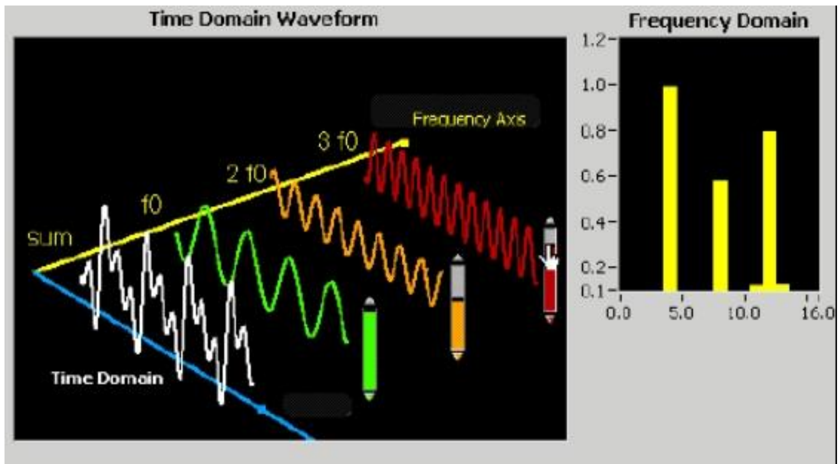
Time Domain and Frequency Domain

- In 1807, Jean Baptiste Joseph Fourier showed that any periodic signal could be represented by a series of sinusoidal functions



In picture: the composition of the first two functions gives the bottom one

Time Domain and Frequency Domain



	Hertz	cpspch	Half-Step	MIDI Note
c 5	523.25	9.00	3	72
b 4	493.88	8.11	2	71
a# 4	466.16	8.10	1	70
a 4	440.0	8.09	0	69
g# 4	415.3	8.08	-1	68
g 4	391.1	8.07	-2	67
f# 4	369.99	8.06	-3	66
f 4	349.23	8.05	-4	65
e 4	329.63	8.04	-5	64
d# 4	311.13	8.03	-6	63
d 4	293.66	8.02	-7	62
c# 4	277.18	8.01	-8	61
c 4	261.63	8.00	-9	60

Fourier Transform

- Because of the property:

EULER's FORMULA

$$e^{i\theta} = \cos\theta + i\sin\theta$$

$$e^{i\omega t} = \cos\omega t + i\sin\omega t$$

$$\text{where } i = \sqrt{-1}$$

- Fourier Transform takes us to the frequency domain:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)\, e^{-i\omega t}\, dt$$

the Fourier transform; strength of frequency ω contained in f(t)

scale factor for the Fourier Transform $F(\omega)$; the original signal in the time domain; the "inverse Fourier transform".

sinusoidally varying "basis" function for the expansion

CTFT

$$\begin{aligned} \text{CTFT} : ContSignals &\rightarrow ContSignals. \\ \text{InvCTFT} : ContSignals &\leftarrow ContSignals \end{aligned}$$

$$t \rightarrow x(t) \qquad\qquad \omega \rightarrow X(\omega)$$

$$\boxed{\forall \omega, \quad X(\omega) = \int_{-\infty}^{\infty} x(t) e^{-i\omega t} dt}$$

$$\boxed{\forall t, \quad x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega) e^{i\omega t} d\omega}$$

Discrete-time Fourier Transform (DTFT)

$$\text{DTFT} : DiscSignals \rightarrow ContPeriodic_{2\pi}$$
$$\text{InvDTFT} : ContPeriodic_{2\pi} \leftarrow DiscSignals$$

$$n \rightarrow x(n) \qquad\qquad \omega \rightarrow X(\omega)$$

$$\boxed{\forall \omega \in Reals, \quad X(\omega) = \textstyle\sum_{-\infty}^{\infty} x(n) e^{-i\omega n}}$$

Sequence domain: discrete and unlimited length
Frequency domain: continuous

$$\boxed{\forall n \in Ints, \quad x(n) = \frac{1}{2\pi} \textstyle\int_0^{2\pi} X(\omega) e^{i\omega n} d\omega}$$

Discrete Fourier Transform

- In practice, we often deal with discrete functions (digital signals, for example)
- The input is of limited length (n)
- Discrete version of the Fourier Transform is much more useful in computer science:

$$f_j = \sum_{k=0}^{n-1} x_k e^{-\frac{2\pi i}{n} jk} \qquad j = 0, \dots, n-1$$

- $O(n^2)$ time complexity

Discrete Fourier Transform

In practice, we often deal with discrete functions
The input is of limited length (n)
Discrete version of the Fourier Transform is much more useful in computer science:

Let $x[n]$ be an N-point signal, and W_N be the N^{th} root of unity. The N-point discrete Fourier Transform of $x[n]$, denoted $X(k) = \mathrm{DFT}\{x[n]\}$, is defined as

$$X(k) = \sum_{n=0}^{N-1} x[n]\, W_N^{kn}, \quad 0 \le k \le N-1$$

$$= \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi kn}{N}}$$

W_N^k

N^{th} Root of Unity

$$W_N = \exp\left(-\frac{j2\pi}{N}\right)$$

1) $W_N^{N/4} = j$

2) $W_N^{N/2} = -1$

3) $W_N^{3N/4} = j$

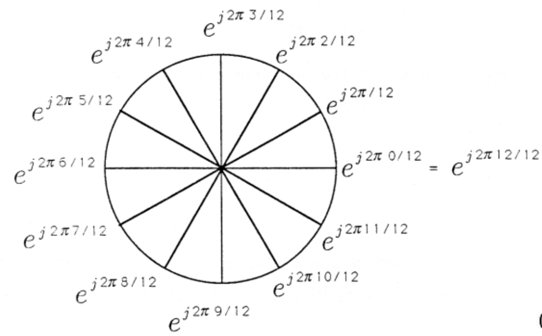
4) $W_N^{N} = 1$

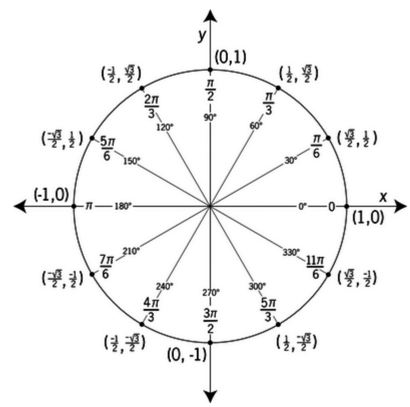
5) $W_N^{k+N} = W_N^k$

6) $W_N^{k+(N/2)} = -W_N^k$

7) $W_N^{2k} = W_{N/2}^k$

8) $W_N^{*} = W_N^{-1}$





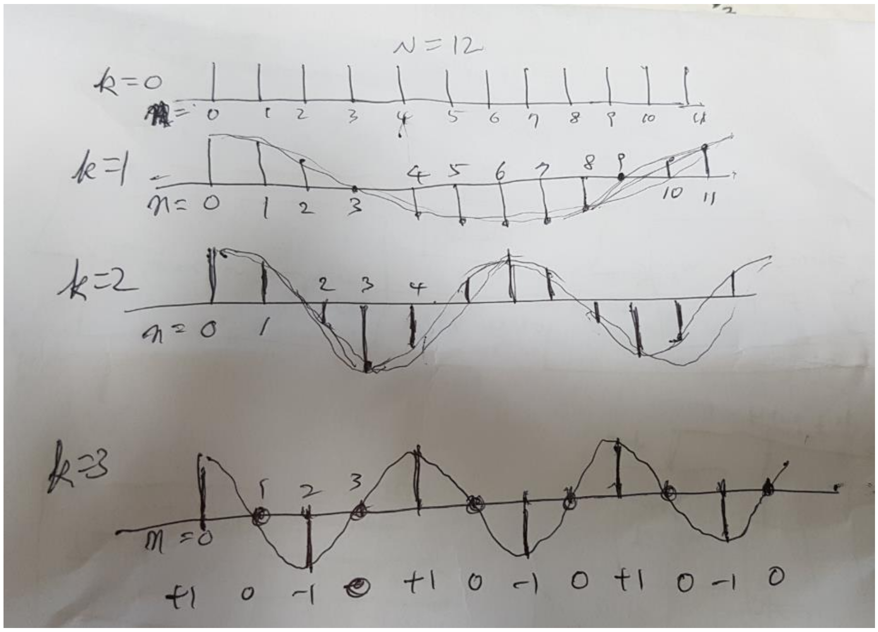
Matrix Formulation

$$
\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{bmatrix} =
\begin{bmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & W_N^1 & W_N^2 & \cdots & W_N^{N-1} \\
1 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & W_N^{N-1} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)(N-1)}
\end{bmatrix}
\begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \vdots \\ x[N-1] \end{bmatrix}
$$

$$\underline{X} = W\underline{x}$$

주기가 N인 discrete (cosine, sine) sequence (한 sample에 2π/N 씩 진전

주기가 N/2인 discrete (cosine, sine) sequence



Matrix Formulation

$$
\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{bmatrix} = \frac{1}{N}\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N^{-1} & W_N^{-2} & \cdots & W_N^{-(N-1)} \\ 1 & W_N^{-2} & W_N^{-4} & \cdots & W_N^{-2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{-(N-1)} & W_N^{-2(N-1)} & \cdots & W_N^{-(N-1)(N-1)} \end{bmatrix}\begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \vdots \\ x[N-1] \end{bmatrix}
$$

$$
\underline{x} = \frac{1}{N} W^* \underline{X}
$$

Inverse Discrete Fourier Transform

Let $X(k)$ be an N-point DFT sequence, and W_N be the N^{th} root of unity. The N-point inverse discrete Fourier Transform of $X(k)$, denoted $x[n]$ = IDFT{$X(k)$}, is defined as

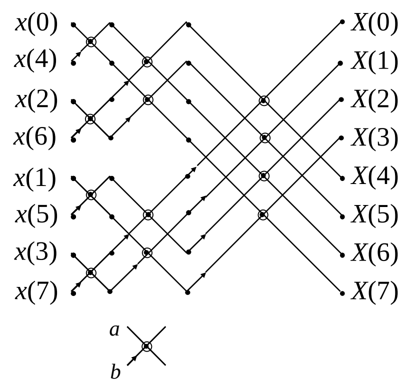
$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\, W_N^{-kn}, \quad 0 \le n \le N-1$$
$$= \frac{1}{N}\sum_{k=0}^{N-1} X(k)\, e^{j\frac{2\pi kn}{N}}$$

Faster DFT computation?

- Take advantage of the symmetry and periodicity of the complex exponential (let $W_N = e^{-j2\pi/N}$)
  - symmetry: $W_N^{k[N-n]} = W_N^{-kn} = (W_N^{kn})^*$
  - periodicity: $W_N^{kn} = W_N^{k[n+N]} = W_N^{[k+N]n}$
- Note that two length N/2 DFTs take less computation than one length N DFT: $2(N/2)^2 < N^2$
- Algorithms that exploit computational savings are collectively called *Fast Fourier Transforms*

8-point FFT

- 8-point Signal Flow Diagram



21

FFT times

- Time (1 multiplication per microsec)

	N	Direct DFT	FFT
2^6	64	.02 sec	.002 sec
2^9	512	1	.02 sec
2^{12}	4096	67	.2
2^{15}	32768	1 hr 11 mins	2
2^{18}	262144	3 days 4 hrs	19

22

DFT Computation Using MATLAB

fft(x) - Computes the N-point DFT of a vector x of length N

fft(x, M) - Computes the M-point DFT of a vector x of length N
If $N < M$, x is zero-padded at the end to make it into a vector of length M
If $N > M$, x is truncated to the first M samples

ifft(X) - Computes the N-point IDFT of a vector X of length N

ifft(X, M) - Computes the M-point IDFT of a vector X of length N
If $N < M$, X is zero-padded at the end to make it into a vector of length M
If $N > M$, X is truncated to the first M samples

DFT Interpretation

DFT sample $X(k)$ specifies the magnitude and phase angle of the k^{th} spectral component of $x[n]$.

$$\text{Magnitude spectrum} = \frac{1}{\text{N}} |X(k)|$$

$$\text{Phase spectrum} = \angle X(k)$$

The amount of power that $x[n]$ contains at a normalized frequency, f_k, can be determined from the **power density spectrum** defined as

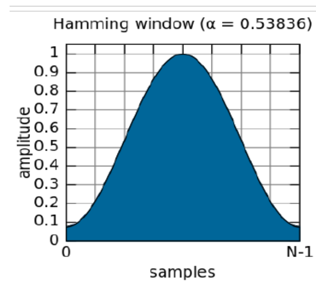
$$P_N(k) = \frac{|X(k)|^2}{N^2}, \quad 0 \le k \le N-1$$

Complex number 의 magnitude 는 real term 과 imaginary term을 각각 제곱하여 더 한후 square root를 씌운다

Windowing

- x[n]의 길이가 N보다 길 때, 이를 N으로 맞추어야 한다.
- Hamming window 가 유명

$$w_0(n) \stackrel{\text{def}}{=} w\left(n + \tfrac{N-1}{2}\right)$$
$$= 0.54 + 0.46\ \cos\left(\frac{2\pi n}{N-1}\right)$$

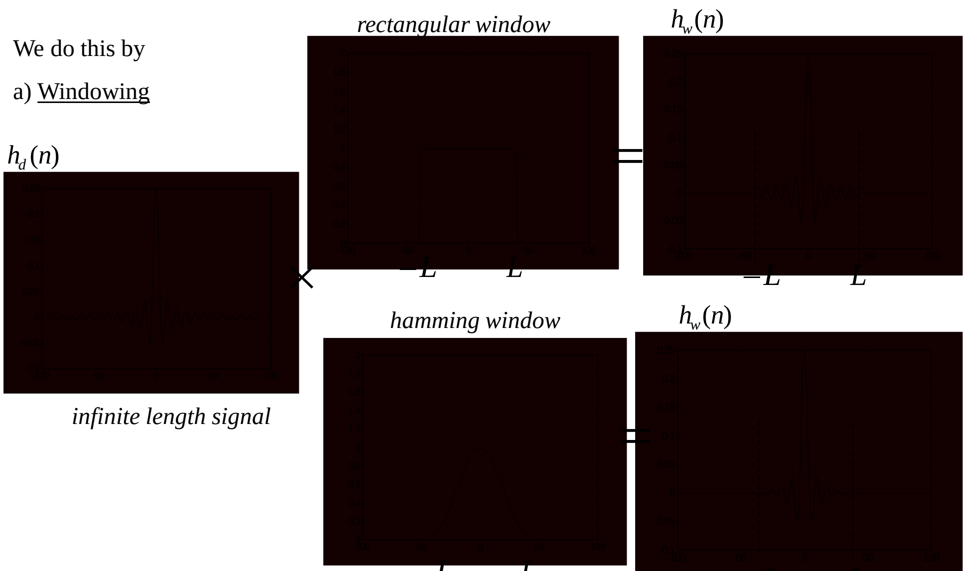


- Rectangular window의 경우 주파수 영역에서의 찌그러짐이 있다.

Problem: we want to determine a *causal Finite Impulse Response (FIR)* approximation of the ideal filter.

We do this by

a) Windowing

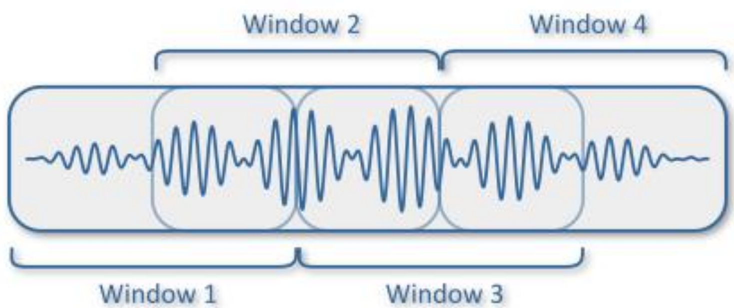


N = 2L + 1

finite impulse response

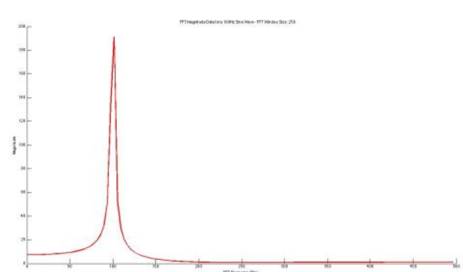
Real-time signal processing

- Sampling -> sliding window -> DFT for each window -> Plot



DFT의 bin number 와 실제 주파수

- 어떤 signal x(t)를 10KHz로 sample 하였다. 그리고 sampling 된 신호 x[0] ~ x[1023] 을 이용하여 1024 point FFT를 실행하였다.
- 아래 보이는 바와 같이 X[100]에 peak이 발생하였다.
- 몇 Hz에 해당?



DFT의 bin number 와 실제 주파수

- X[k = 0, 1, 2, … N-1]이 얻어지는데, 가상적으로 k = N이 sampling 주파수에 해당
- 즉 DFT domain에서 K=1이 변할 때 continuous time domain의 주파수는 fs/N 또는 rad/sec로 2π*fs/N 만큼씩 변한다.
- 앞의 문제에 대한 답은 100*10KHz/1024 ~=약 1KHz
- Sampling 하기 전에 신호가 lowpass filter fs/2를 거치며, k=N/2~ N-1은 k=0~N/2-1과 symmetric하다 (real input signal의 경우)

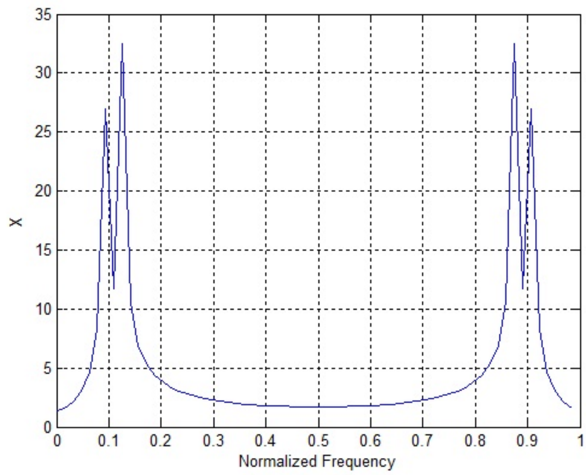
Matrix Formulation

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N^1 & W_N^2 & \cdots & W_N^{N-1} \\ 1 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \vdots \\ x[N-1] \end{bmatrix}$$

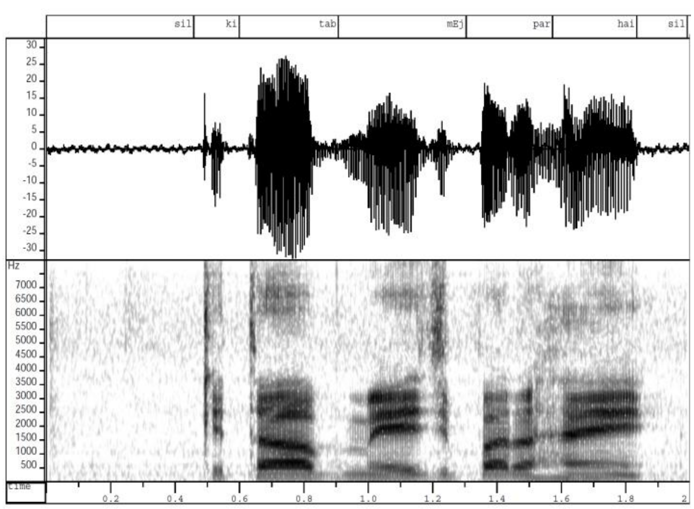
$$\underline{X} = W\underline{x}$$

주기가 N인 discrete (cosine, sine) sequence (한 sample에 2π/N 씩 진전

주기가 N/2인 discrete (cosine, sine) sequence

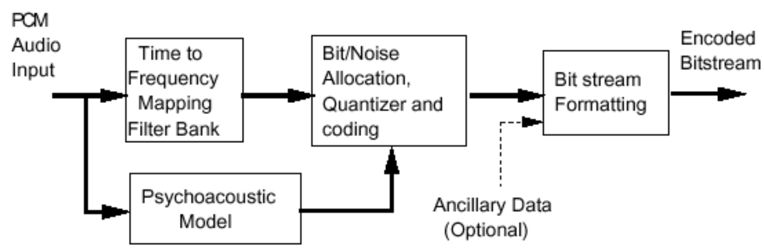


Spectrogram
sampl된 입력신호-> sliding window -> FFT



MP3 audio encoding

Sampling 된 audio 신호를 주파수 domain으로 바꾸어서 (filter bank – FFT 이용) 각 주파수별로 귀의 민감도와 masking 특성을 따진다. 사람의 귀는 높은 주파수에서 세밀하게 못 듣고 또 높은 세기의 주파수 성분 옆은 잘 안들린다 (masking 효과). 이를 이용하여 잘 못 듣는 부분은 엉성하게 양자화한다. 16bit 48KHz (768Kbp) -> 128Kbps, 64Kbps 로 줄인다.



MPEG/Audio Encoder

Applications

- In image processing:
  - Instead of time domain: *spatial domain* (normal image space)
  - *frequency domain:* space in which each image value at image position F represents the amount that the intensity values in image I vary over a specific distance related to F

Extending DFT to 2D (cont'd)

- Special case: f(x,y) is N x N.

- Forward DFT $$F(u,v)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(\frac{ux+vy}{N})},$$

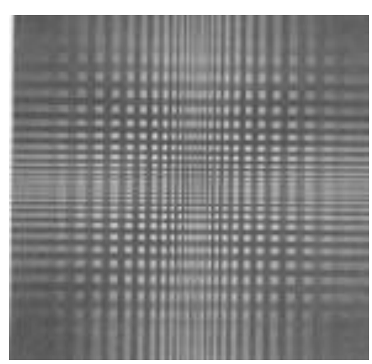
  u,v = 0,1,2, …, N-1

- Inverse DFT $$f(x,y)=\frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(\frac{ux+vy}{N})},$$

  x,y = 0,1,2, …, N-1

Applications: Frequency Domain In Images

- *Spatial frequency* of an image refers to the rate at which the pixel intensities change
- In picture on right:
  - High frequences:
    - Near center
  - Low frequences:
    - Corners



Other Applications of the DFT

- Signal analysis
- Sound filtering
- Data compression
- Partial differential equations
- Multiplication of large integers