Multimedia Software Systems
CS4551

Basics of Lossy Compression Algorithms

Lossy Compression

- Decompressed signal is not like original signal –data loss
- Objective: minimize the *distortion* for a given compression ratio
  - Ideally, we would optimize the system based on *perceptual distortion* (difficult to compute)
  - We'll need a few more concepts from statistics…

Distortion Measurements

- *y* : the original value of a sample (e.g I(*i*,*j*) the image pixel at (*i*,*j*))
- *y^ :* the sample value after compression/decompression (e.g K(*i*,*j*) the approximated image pixel at (*i*,*j*))
- 1. Measuring differences (errors)
  - *f(y-y^) :* error (or noise or distortion) where *f* is a distance function (e.g MSE that is **M**ean **S**quared **E**rror)

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

Distortion Measurements (2)

- 2. SNR (Signal to Noise Ratio) or PSNR (Peak SNR) in dB

$$\begin{aligned} PSNR &= 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\ &= 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \\ &= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE) \end{aligned}$$

- MAX_I - the maximum possible value of the sample. For example, when the pixels are represented using 8 bits per sample, this is 255.
- Typical values for the *PSNR* in lossy image and video compression are between 30 and 50 dB for 8bit depth image, where higher is better.
- In the absence of noise, the two images I and K are identical, and thus the **MSE** is zero. In this case the PSNR is undefined.

Original uncompressed image | Q=90, PSNR 45.53dB | Q=30, PSNR 36.81dB | Q=10, PSNR 31.45dB

Example luma PSNR values for a cjpeg compressed image at various quality levels.

https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio

Lossy Compression – Simple Examples

- Subsampling:
  - retain only samples on a *subsampling grid* (spatial/temporal).
  - See examples in previous lecture
  - Compression achieved by reducing the number of samples
  - Has fundamental limitations: see sampling theorem
- Quantization: quantize with fewer bits
  - Compression achieved by reducing the number of bits per sample
  - Different quantization methods: Scalar uniform/non-uniform quantizations, Vector quantization
  - As Quantization Interval size increases –compression increases (so does error!)
- Can we do better than simple subsampling and quantization?

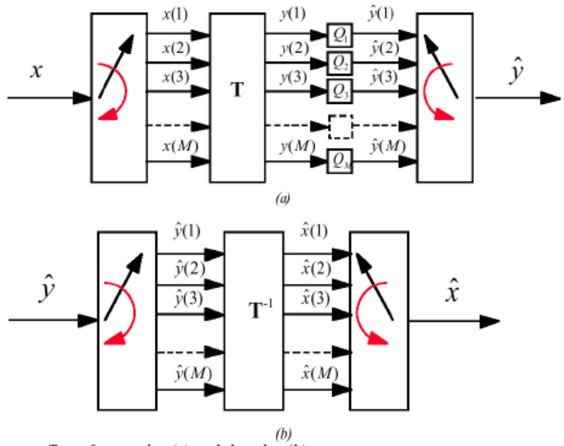
Transform Coding

- The rational behind the transform coding is that if **`Y`** is the result of a transform **`T`** of the input **`X`** in such a way that the components (a.k.a. channels) of **`Y`** are much less correlated, the **`Y`** can be coded more efficiently than **`X`**.
- Eg. Let's assume **`T:`** `RGB -> YCbCr` color space conversion
  - **`T`** transforms a RGB image into a YCbCr image.
  - Y(Luminance) component is little related to CbCr (Chrominance).
  - Luminance component can be compressed differently from color components.
  - Human visual system is more sensitive to luminance. So we can apply sub-sampling (a simple compression) to color components or use a bigger quantization interval (less number of bits for the quantization) for color components.
- In general, the transform **`T`** itself does not compress any data. The compression comes from the processing and quantization of the components of **`Y`**.

Transform Coding (2)

- In Transform Coding, a segment of information undergoes an *invertible* mathematical transformation.
- Segments of information can be samples (assume M samples) such as
  - 8x8 pixel block (*M*=64) of an image frame
  - A segment of speech
  - A chunk of data in any other format
- Transform Coding works as follows:
  - Let X = {x(1), x(2), ... ,x(M)} is a segment of information
  - Apply a suitable *invertible* transformation T (typically a matrix multiplication) to X
  - Y is the result of the transformation Y = T(X) = {y (1), y (2), ... ,y(M)}
  - Quantize, get Y^={ y^(1),...,y^(M)} and transmit Y^

Transform Coding (3)



Transform coder (a) and decoder (b).

Transform Coding (4)

- Quantizers in different channels may have different numbers of quantization levels (different quantization interval size) => each channel ultimately might yield a different number of bits.
- Bit budget *B*: if we quantize *y^(1),...,y^(M)* using *B(1),...,B(M)* bits respectively, then

$$B = \sum_{i=1}^{i=M} B(i)$$

- Optimal bit allocation: allocate more bits to those channels that have the highest variance

Transform Groups

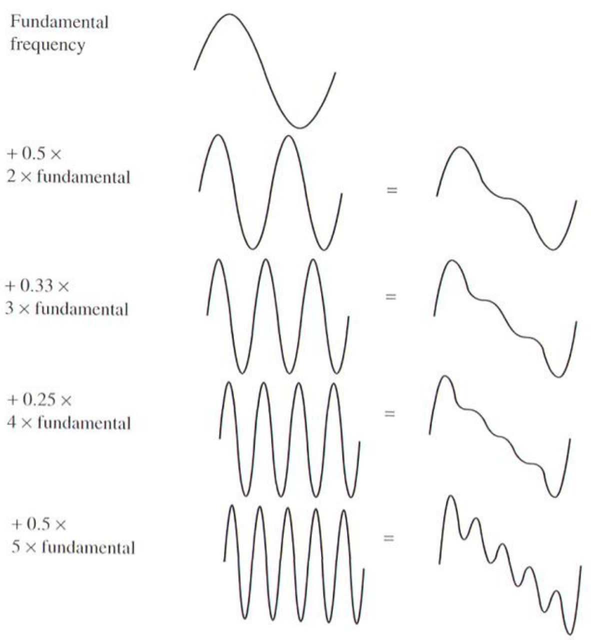
- Frequency transforms— Discrete Fourier transforms, Hadamard transforms, Lapped Orthogonal transforms, Discrete Cosine transforms
  – Involves converting the signal from the sample domain (e.g 1D time domain for audio and 2D spatial domain for images) to the frequency domain.
- Statistical transforms— Karhunen-Loève Transforms (a.k.a. *Eigenvector Transform)*
- Wavelet transforms— While similar to frequency transforms, these transforms work more efficiently because the input is transformed to a multi-resolution frequency representation.

Discrete Cosine Transform (DCT)

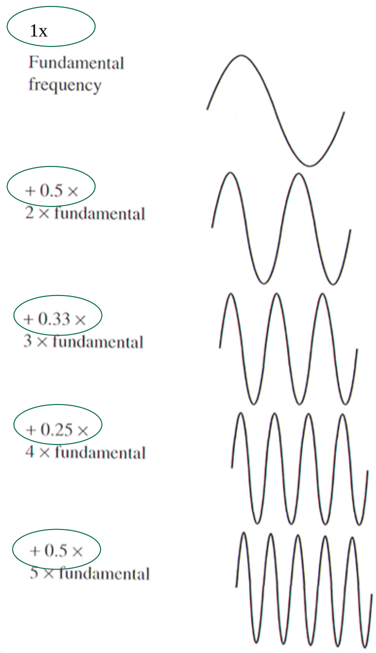
- The *DCT* is a widely used transform coding technique. It is able to perform de-correlation of the input signal in a data independent manner.
- It has the property that different channels represent the signal power along different (spatial or temporal) *frequencies* similarly to the (discrete) Fourier transform

Background Before DCT

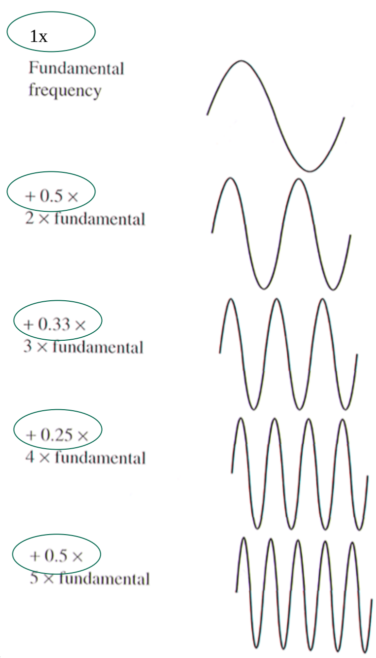
- DC (Direct Current) : An electrical signal with constant magnitude (eg. A battery carrying 1.5 volts DC)
- AC (Alternating Current) : an electrical signal that changes its magnitude periodically at a certain frequency. (eg. 110 volts AC, 60Hz)
- Most real signal is complex. However, any signal can be expressed as a sum of multiple sinusoidal waveform. (a.k.a. Fourier Analysis)



Temporal to Frequency



Frequency to Temporal



Discrete Cosine Transform (DCT)

- Let's start with 1D digital signal.
- Definition of 1D DCT

$$F(u) = \frac{C(u)}{2} \sum_{i=0}^{M-1} \cos \frac{(2i+1)u\pi}{2M} f(i)$$

where $u = 0,\ldots,M-1$

M = the number of samples in the signal

$$\mathrm{C(u)} = \begin{cases} \frac{\sqrt{2}}{2} & \text{if } u = 0 \\ 1 & \text{otherwise} \end{cases}$$

- 1D DCT transforms the signal *f(i)* in the **time** domain to *F(u)* in the **frequency** domain.

Discrete Cosine Transform (DCT)

$$F(u) = \frac{C(u)}{2} \sum_{i=0}^{M-1} \cos \frac{(2i+1)u\pi}{2M} f(i)$$

where $u = 0,\ldots,M-1$

M = the number of samples in the signal

$$\mathrm{C(u)} = \begin{cases} \frac{\sqrt{2}}{2} & \text{if } u = 0 \\ 1 & \text{otherwise} \end{cases}$$

Plot cos graphs with different frequencies at https://www.desmos.com/calculator/nqfu5lxaij

Inverse Discrete Cosine Transform (IDCT)

- Definition of 1D IDCT

$$f(i) = \sum_{u=0}^{M-1} \frac{C(u)}{2} \cos\frac{(2i+1)u\pi}{2M} F(u)$$

$$\textbf{where } u = 0,\ldots,M-1$$

$$M = \textbf{the number of samples in the signal}$$

$$\mathbf{C(u)} = \begin{Bmatrix} \frac{\sqrt{2}}{2} & \textbf{if u = 0} \\ 1 & \textbf{otherwise} \end{Bmatrix}$$

- 1D IDCT transforms back the coefficients *F(u)* in frequency domain into the original signal *f(i)* in the time domain.

Discrete Cosine Transform (DCT)

- The role of DCT is to use Cosine function to decompose the original signal into its one DC component and several AC components of the original signal.
- When *u = 0*, DCT yields the DC coefficient. When *u=1..M-1*, it yields the first, second, … (M-1)th AC coefficients.
- The role of IDCT is to reconstruct (recompose) the signal using DC and ACs.

Discrete Cosine Transform (DCT)

- Let's consider M=8 then

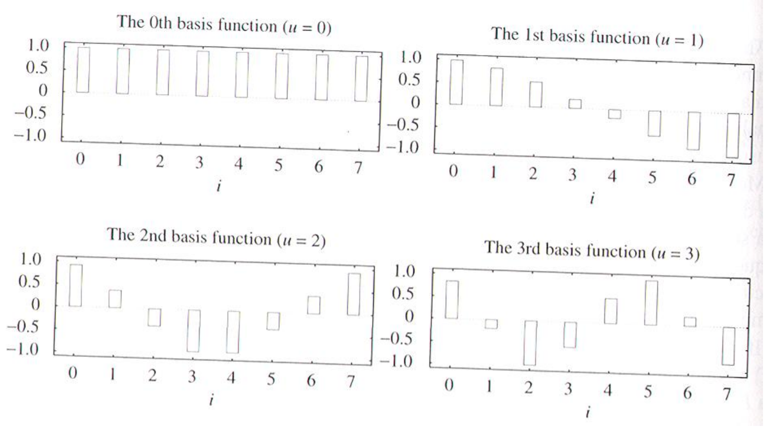
$$F(u) = \frac{C(u)}{2}\sum_{i=0}^{7}\cos\frac{(2i+1)u\pi}{16} f(i) : \textbf{forward transform}$$

$$f(i) = \sum_{u=0}^{7}\frac{C(u)}{2}\cos\frac{(2i+1)u\pi}{16} F(u) : \textbf{inverse transform}$$

$$\textit{where } i = 0,\ldots,7 \textbf{ and } u = 0,\ldots,7$$

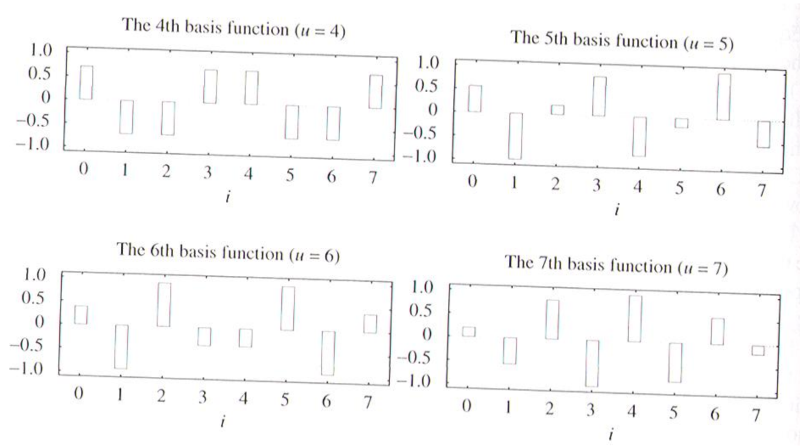
1D DCT Basis Functions

- **Let's consider M=8, then we have following basis cosine functions.**



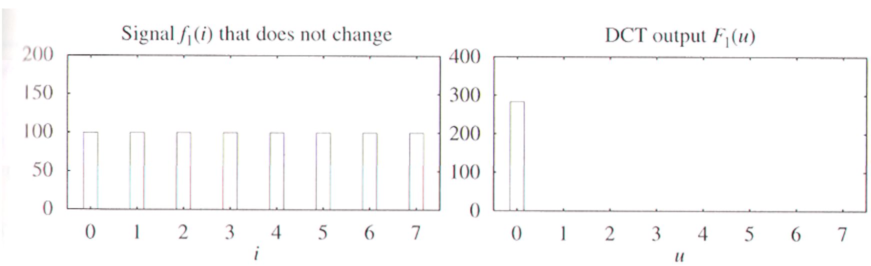
1D DCT Basis Functions

- **Let's consider M=8, then we have following basis cosine functions.**

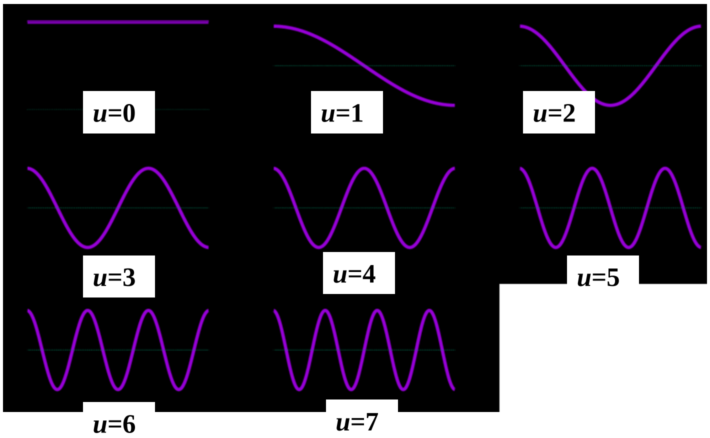


Discrete Cosine Transform (DCT) - Example

- If the original signal $f_1(i)$ = 100, i = 0,…,7 (M=8), then $F_1(0) \approx 283$, $F_1(1) = F_1(2) = \ldots = F_1(7) = 0$

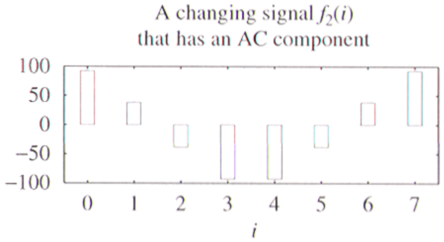


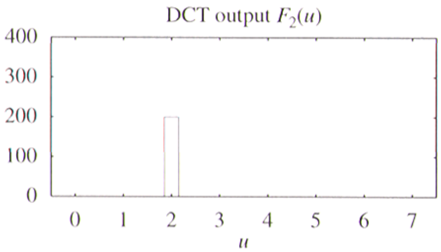
1D DCT Basis Functions : DCT0 ~ DCT 7



Discrete Cosine Transform (DCT) - Example

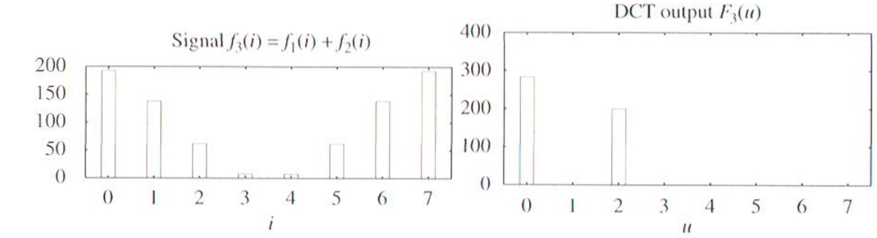
- If the original signal $f_2(i)$ has the same frequency and phase as the second cosine basis function and its amplitude is 100, then $F_2(0)=0, F_2(1)=0, F_2(2)=200, F_2(3)=\ldots= F_2(7)=0$





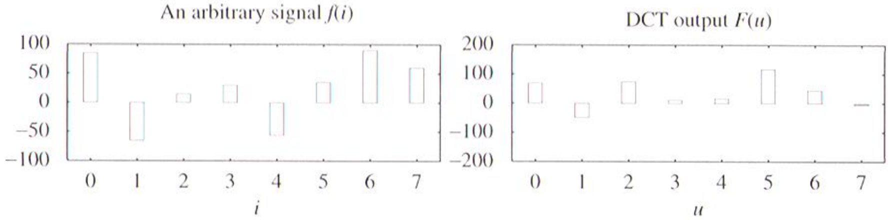
Discrete Cosine Transform (DCT) - Example

- If the original signal $f_3(i)$ is the sum of the previous two signals ($f_3(i) = f_1(i)+ f_2(i)$), then $F_3(0)=283$, $F_3(1)=0$, $F_3(2)=200$, $F_3(3)=\ldots= F_3(7)=0$



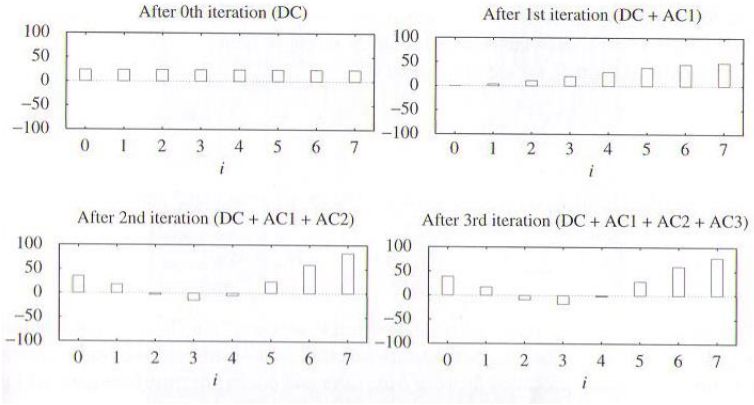
Discrete Cosine Transform (DCT) - Example

- If the original signal $f(i)$ has $f(0)=85$, $f(1)=-65$, $f(2)=15$, $f(3)=30$,$f(4)=-56$, $f(5)=35$, $f(6)=90$,$f(7)=60$, then $F(0)=69$,$F(1)=-49$, $F(2)=74$, $F(3)=11$,$F(4)=16$, $F(5)=117$,$F(6)=44$, $F(7)=-5$



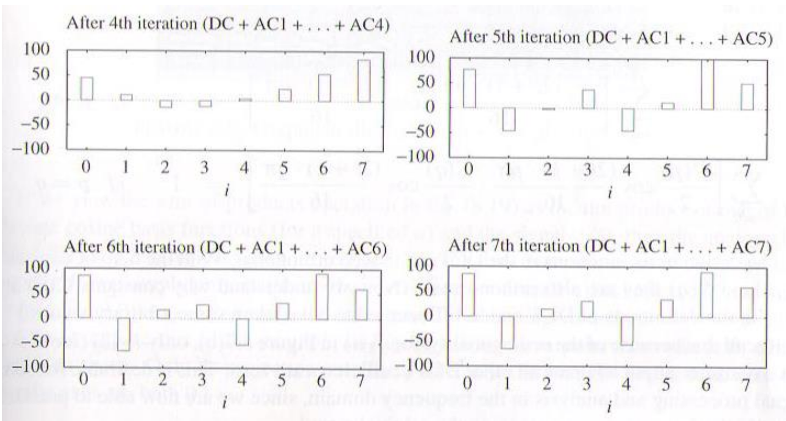
IDCT Example

- F(u): 69, -49, 74, 11, 16, 117, 44, -5 where u=0,…,7



IDCT Example

- F(u): 69, -49, 74, 11, 16, 117, 44, -5 where u=0,…,7



Orthonormal Basis Functions

Functions $B_p(i)$ and $B_q(i)$ are orthonormal if

$$\sum_i [B_p(i) \cdot B_q(i)] = 0,\ p \neq q$$

$$\sum_i [B_p(i) \cdot B_q(i)] = 1,\ p = q$$

$$\sum_{i=0}^{7} \left[\cos\frac{(2i+1)p\pi}{16} \cdot \cos\frac{(2i+1)q\pi}{16} \right] = 0,\ p \neq q$$

$$\sum_{i=0}^{7} \left[\frac{C(p)}{2} \cos\frac{(2i+1)p\pi}{16} \cdot \frac{C(q)}{2} \cos\frac{(2i+1)q\pi}{16} \right] = 1,\ p = q$$

- DCT basis functions (with the help of C(u)) are orthonormal. With this property, the signal is not amplified during the transformation. We get the same signal back.

Summary – 1D Discrete Cosine Transform (DCT)

- The 0^{th} DCT coefficient *F(0)*
  - *F(0)* corresponds to the DC component of the signal *f(i)*.
  - *F(0)* is equal to the average magnitude of the signal.
- The other seven DCT coefficients
  - They reflect the various changes (AC) components at different frequencies.
  - If we denote F(1) as AC1, F(2) as AC2… F(7) as AC7, then AC1 completes half a cycle as a cosine function over [0..7] (low frequency);AC2 completes a full cycle; … ; and AC7 completes three and a half cycle (high frequency).

2D DCT Formula

- 2D Discrete Cosine Transform (DCT) and Inverse DCT

** DCT*

$$F(u,v) = \frac{C(u)C(v)}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} \cos\frac{(2i+1)u\pi}{16} \cos\frac{(2j+1)v\pi}{16} f(i,j)$$

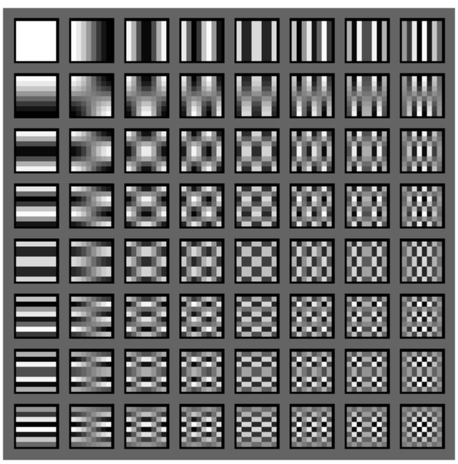
** IDCT*

$$f(i,j) = \sum_{i=0}^{7} \sum_{j=0}^{7} \frac{C(u)C(v)}{4} \cos\frac{(2i+1)u\pi}{16} \cos\frac{(2j+1)v\pi}{16} F(u,v)$$

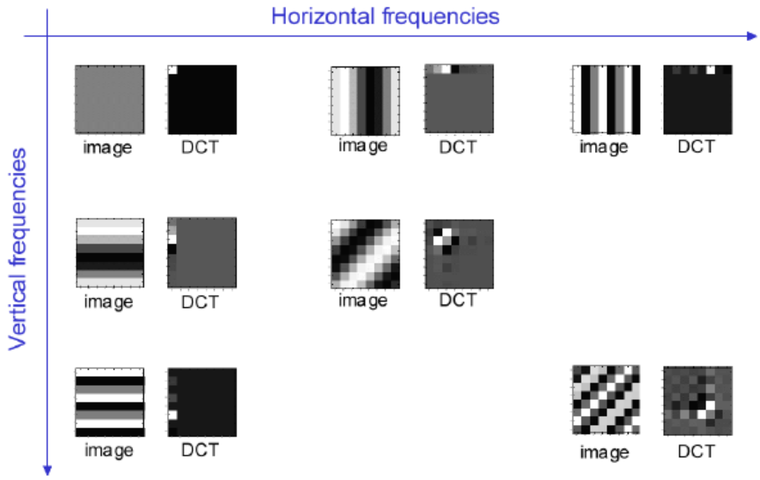
$where\ i,j,u,v = 0,1,...7$

$$C(x) = \begin{cases} \frac{\sqrt{2}}{2}\ if\ x = 0 \\ 1\ otherwise \end{cases}$$

8x8 2D DCT Basis Functions



Examples of 8x8 DCTs



DCT for Compression

- How can we use this transformation for compression?
  - Each component is little dependent to each other.
  - Human visual system is less sensitive to high frequency components.
  - So we apply different quantization interval size to each components. i.e. More bits for DC component and less bits for AC components.