# ENGR 76

# Information Science and Engineering

## Lecture 8: Frequency Domain Representation IV

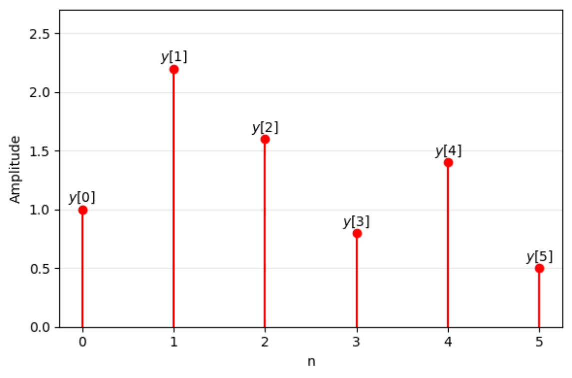
Siddharth Chandak

# Recap: DCT

# Discrete Time Signals

- Consider the array of length $L$

$$y[0], \ldots, y[L-1]$$




- Frequency Domain Representation?

# Discrete Cosine Transform

**Fact**

Given discrete time signal $y[0], \ldots, y[L-1]$, then $y[n]$ for $n = 0, \ldots, L-1$ can be represented as

$$y[n] = b_0 + \sum_{j=1}^{L-1} b_j \cos\left(\frac{\pi \cdot j \cdot (n+0.5)}{L}\right).$$

Moreover, the coefficients $b_0, \ldots, b_{L-1}$ are unique for given signal.

- We only require $L$ terms as we have only $L$ degrees of freedom for the signal

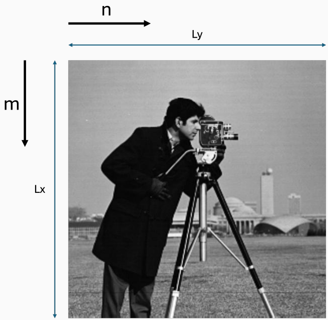
# Representations

- Time-domain representation: $y[0], y[1], \ldots, y[L-1]$
  - Signal at different time indices
- Frequency-domain representation: $b_0, b_1, \ldots, b_{L-1}$
  - Coefficients

# Recap: 2-D DCT

# 2-D Signals (Images)

- $X[m,n]$ for $m \in \{0,1,\ldots,L_x-1\}, n \in \{0,1,\ldots,L_y-1\}$
- Convention:




- $X[m,n]$ is the pixel at $m$-th row and $n$-th column

# 2-D Discrete Cosine Transform

- Any image $X$ can be represented as

$$X[m,n] = \sum_{i=0}^{L_x-1} \sum_{j=0}^{L_y-1} A[i,j]\phi_{i,j}[m,n],$$

where

$$\phi_{i,j}[m,n] = \cos\left(\frac{\pi \cdot i \cdot (m+0.5)}{L_x}\right)\cos\left(\frac{\pi \cdot j \cdot (n+0.5)}{L_y}\right).$$

# Representations

- $X[m,n]$ - $L_x \times L_y$ - image domain representation
  - Pixel intensities
- $A[i,j]$ - $L_x \times L_y$ - frequency domain representation
  - Coefficients
- Both have the same size - so why does it help with compression?

# Image Compression

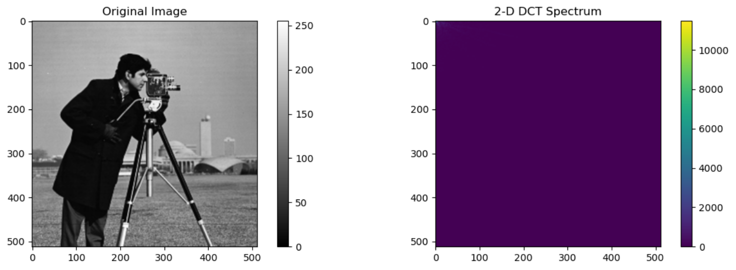
# Step I

- The first step is to apply 2D-DCT to get frequency domain representation
- Both have the same size $(L_x \times L_y)$- so why does it help with compression?

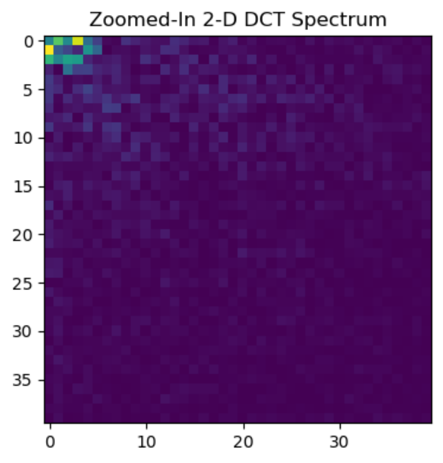
# Why is 2-D DCT helpful for image compression? — Reason I

- **Energy Compaction:**
  - For most natural images: DCT is sparse (concentrated in lower frequencies)
  - Most high-frequency coefficients have low magnitudes: can be discarded (i.e., low-pass filter on $A$)
  - Low Pass Filter on Image Visualization

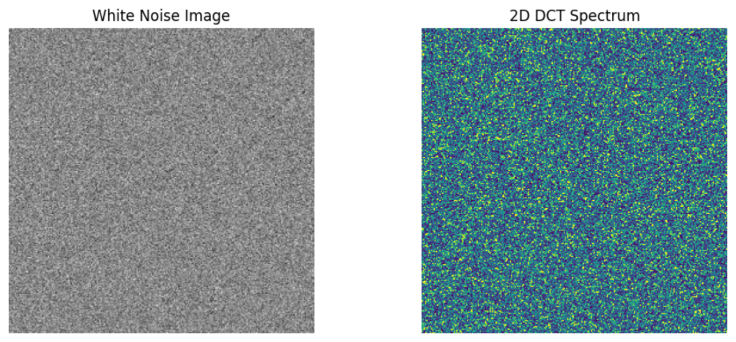
# Example


Original Image
0
100
200
300
400
500
0
100
200
300
400
500
250
200
150
100
50
0
2-D DCT Spectrum
0
100
200
300
400
500
0
100
200
300
400
500
10000
8000
6000
4000
2000
0

# Example

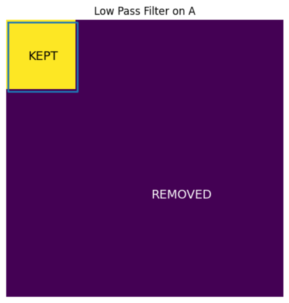


# Why Only Natural Images?



# Using energy compaction to our benefit

- Apply low-pass filter on $A$ to get $\tilde{A}$ using cutoff $k_c$



# Why is 2-D DCT helpful for image compression? — Reason II

- **Decorrelation:**
  - For most natural images, neighboring pixels are highly correlated
  - But in frequency-domain representation, coefficients are much less statistically dependent.
  - Huffman coding on individual symbols performs better

# Process

- Given image $X$
- Apply 2D-DCT to get frequency domain representation $A$
- Apply low-pass filter on $A$ to get $\tilde{A}$ using cutoff $k_c$

Now we want to apply Huffman coding...

# Applying Huffman Coding

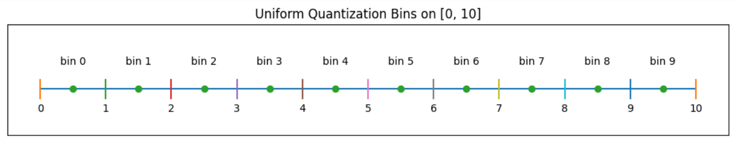
- Output of DCT is real-valued
  - Each $A[i, j]$ is a real number
- Recall that we studied Huffman coding for discrete alphabets
  - Each symbol took values in a discrete set (e.g., $\{1, 2, 3, \ldots, M\}$)
- So how to convert these real-valued symbols into discrete values?

# Quantization

- Mapping a continuous (real-valued) to a finite set of $N$ discrete levels
- Simplest form: **Uniform Quantization**
    - Divide into equal-width intervals and map to the nearest representative
- Suppose a number $x$ takes values in $[0, 10]$
    - We want to quantize it such that it can take $N = 10$ possible values

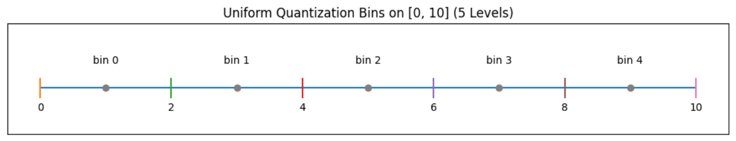
# Quantization

- **Uniform Quantization**
  - Divide into equal-width intervals and map to the nearest representative
- Suppose a number $x$ takes values in $[0, 10]$
  - We want to quantize it such that it can take $N = 10$ possible values



# Quantization

- Suppose a number $x$ takes values in $[0, 10]$
  - We want to quantize it such that it can take $N = 5$ possible values




- Quantization: Takes input $x$ and outputs bin id from $\{0, 1, \ldots, N - 1\}$
- Dequantization: Outputs $\hat{x}$ as the 'reconstruction' (midpoint of the bin)

# Quantization

- Can we recover the exact original value $x$ from $\hat{x}$, i.e., after quantization and dequantization?
- What happens as we decrease $N$?

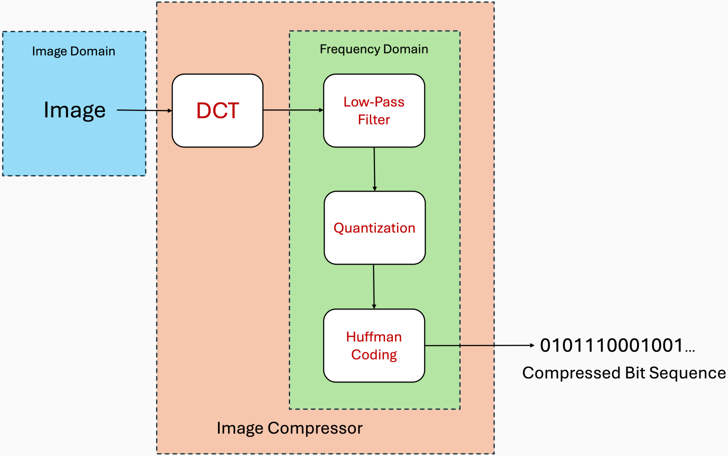
# Quantization

- Can we recover the exact original value after quantization?
  - No - we are throwing away preciseness
- What happens as we decrease $N$?
  - We are discarding more

# Back to Image Compression

- Given image $X$
- Apply 2D-DCT to get frequency domain representation $A$
- Apply low-pass filter on $A$ to get $\tilde{A}$ using cutoff $k_c$
- Quantize each $\tilde{A}[i, j]$ with $N$ levels
- Huffman Coding
  - Compute empirical frequencies
  - Use these for Huffman algorithm
  - Encode
- Compressed bit sequence!

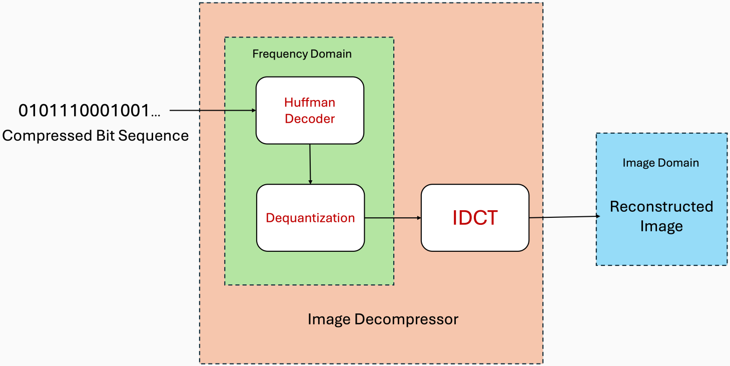
# Image Compressor



# Decompression

- Given compressed bit sequence
- Apply Huffman decoding
- Dequantize to get $\hat{A}[i, j]$
- Apply 2D-IDCT to get reconstructed image $\hat{X}$

# Image Decompressor



# Reconstruction

- Compress image $X$ and then decompress it to get reconstructed image $\hat{X}$
- Are they the same?
    - Is $X[m, n] = \hat{X}[m, n]$ for each pixel $m, n$?

# Reconstruction

- Compress image $X$ and then decompress it to get reconstructed image $\hat{X}$
- Are they the same?
  - Is $X[m,n] = \hat{X}[m,n]$ for each pixel $m,n$? — **NO!**
- Which steps lead to this discrepancy?
  - If I just did $A = \text{DCT}(X)$ and then $\hat{X} = \text{IDCT}(A)$, then what would happen?

# Reconstruction

- Compress image $X$ and then decompress it to get reconstructed image $\hat{X}$
- Are they the same?
  - Is $X[m,n] = \hat{X}[m,n]$ for each pixel $m,n$? — **NO!**
- Which steps lead to this discrepancy?
  - If I just did $A = \text{DCT}(X)$ and then $\hat{X} = \text{IDCT}(A)$, then what would happen?
- Lossy steps:
  - Low-pass Filter
  - Quantization

# Lossy Compression

- The reconstruction is not the same as the original
- Quantifying the difference:

$$\mathsf{mse}(\hat{X}, X) = \frac{1}{L_x L_y} \sum_{m=0}^{L_x - 1} \sum_{n=0}^{L_y - 1} \big( X[m,n] - \hat{X}[m,n] \big)^2$$

- MSE is easy to compute - but does not always align well with human perception

# DFT: Back to Frequency Domain Representations

# Recall...

- Recall where we started...
    - Any periodic signal $x(t)$ with period $T$ can be represented using the **Fourier Series**

$$x(t) = b_0 + \sum_{j=1}^{\infty} a_j \sin\left(2\pi \frac{j}{T} t\right) + b_j \cos\left(2\pi \frac{j}{T} t\right)$$

- Then we went to even signals...
    - Continuous-time even periodic signal
        - Fourier Cosine Series
    - Continuous-time time-limited signal - Even periodic extension
        - Fourier Cosine Series
    - Discrete-time time-limited signal - Even periodic extension
        - Discrete Cosine Transform

# Discrete Time Signal

- Consider discrete time signal of length $L$: $y[0], y[1], \ldots, y[L-1]$
  - What if we did a periodic extension?
  - What representation would this be similar to?
  - Will we need both sines and cosines?

# Discrete Time Signal

$$y[n] = b_0 + \sum_{j=1}^{\infty} a_j \sin\left(2\pi \frac{j}{L} n\right) + b_j \cos\left(2\pi \frac{j}{L} n\right)$$

- Do we need summation till infinity?
- $L$ degrees of freedom - how many terms do we need?

# Discrete Fourier Transform

**Fact**

Given discrete time signal $y[0], \ldots, y[L-1]$, then $y[n]$ for $n = 0, \ldots, L-1$ can be represented as

$$y[n] = b_0 + \sum_{j=1}^{\frac{L-1}{2}} a_j \sin\left(2\pi \frac{j}{L} n\right) + b_j \cos\left(2\pi \frac{j}{L} n\right)$$

Moreover, the coefficients are unique for given signal.

- Number of terms is $\frac{L-1}{2} + \frac{L-1}{2} + 1 = L$[1]

[1] assumed that $L$ is odd, if $L$ is even then additional $\cos(\cdot)$ term is taken

# Important Remark

- Given discrete time signal of length $L$
- DCT and DFT are two different ways of representing the signal
  - Both are valid
  - DFT used both sines and cosines
  - DCT just uses cosines
- Depends on our choice and the application
  - DFT was obtained by periodic extension
  - DCT was obtained by even periodic extension
  - The *extension* is a construct we did

# Concluding Frequency Representations

# Frequency Domain Representations

- Different representations based on what signal we are representing
  - Fourier Series
  - Fourier Cosine Series
  - Discrete Fourier Transform
  - Discrete Cosine Transform

# Concluding...

- Joseph Fourier
    - Developed this theory to solve the heat equation - how heat flows through solids
- Many applications:
    - Fundamental to quantum mechanics
    - Image Processing (current discussion)
    - Communication systems (soon...)
    - Acoustics

# Frequency Domain Representations

| Time-domain representation | | Frequency-domain representation | | |
|---|---|---|---|---|
| **Time type** | **Periodic / Time-limited** | **Name** | **Frequency type** | **Periodic / Frequency-limited** |
| Continuous | Yes | Fourier Series $\{b_0, a_1, b_1, \ldots\}$ | Discrete | No |
| Discrete | Yes (finite-length) | Discrete Fourier Transform (DFT) $\{b_0, a_1, \ldots, b_{(L-1)/2}\}$ | Discrete | Yes |
| Continuous | Not time-limited (aperiodic) | Fourier Transform $\{X(f)\}$ | Continuous | No |
| Discrete | Not time-limited (aperiodic) | Discrete-Time Fourier Transform (DTFT) $\{X(f)\}$, periodic | Continuous | Yes |

# Thank You!