Digital Image Processing Notebook

Junchi Wang

April 2025

1 ELEC70078 Digital Image Processing

1.1 Image Acquisition

An image is a projection of a 3D scene into a 2D projection plane. An image can be defined as a function of two variables (x, y) as:

$$f(x, y) : \mathbb{R}^2 \to \mathbb{R}$$

where, for each position (x, y) in the projection plane, $f(x, y)$ defines the light intensity at this point.

1.2 Sampling and Quantization

The analogue image needs to be converted into a digital image by sampling and quantization (Digitize).

Obviously, the higher the sample rate, the better the quality. The more quantization levels, the better the quality.

1.3 Four Main Parts of Digital Image Processing

1. Image Transforms
2. Image Enhancement
3. Image Restoration
4. Image Compression

1.4 Natural Signals and Compression

The natural signals are sparse, and that's why they can be compressed.

2 2D Discrete Fourier Transform (DFT)

2.1 1D DFT

The generic form of a one-dimensional signal transform:

$$g(u) = \sum_{x=0}^{N-1} T(u, x) f(x), \quad \text{for } 0 \le u \le N - 1$$

$T(u, x)$ is a function of u, x called the **forward transformation kernel**.

$$\mathbf{g} = T \cdot \mathbf{f}$$

For DFT, we have:

$$T(u, x) = e^{-j2\pi \frac{ux}{N}}$$

The generic form of inverse transform:

$$f(x) = \frac{1}{N} \sum_{u=0}^{N-1} I(u,x) g(u), \quad \text{for } 0 \leq x \leq N-1$$

$I(u,x)$ is called the **inverse transformation kernel**.

$$\mathbf{f} = I \cdot \mathbf{g}, \quad \mathbf{g} \in \mathbb{R}^N$$

$$I = \frac{1}{N} e^{j2\pi \frac{ux}{N}} = \frac{1}{N} T^* = T^{-1}$$

2.2 Matrix Properties

Recall:

$$\frac{1}{N} T^* = T^{-1}, \quad T^* T = N \cdot I, \quad T^T T = N \cdot I$$

This implies that the matrix is **orthogonal**.

2.3 2D Transform

The generic form of a 2D transform:

$$g(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} T(u,x) T(v,y) f(x,y)$$

2.4 Separable and Symmetric Transforms

A 2D transform is **separable** if:

$$T(u,x,v,y) = T_1(u,x) T_2(v,y)$$

A 2D transform is **symmetric** if:

$$T(u,x) = T_2(v,y) = T(x,y)$$

$$f(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} I(u,x) I(v,y) g(u,v)$$

2.5 Separable Transform Expansion

$$\begin{aligned}
g(u,v) &= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} T_1(u,x) T_2(v,y) f(x,y) \\
&= \sum_{y=0}^{N-1} T_2(v,y) \left(\sum_{x=0}^{M-1} T_1(u,x) f(x,y) \right) \\
&= \sum_{y=0}^{N-1} T_2(v,y) F(u,y)
\end{aligned}$$

Where $F(u,y)$ is the **intermediate image**.

3 Properties of DFT

3.1 Energy Preservation (Parseval's Theorem)

$$\|g\|^2 = \|f\|^2$$

For images, this can be written as:

$$\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} |f(x,y)|^2 = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} |g(u,v)|^2$$

3.2 Energy Compaction

Transformed image values close to the origin correspond to **low frequency**.

Most energy of the transformed image is concentrated in a small area close to the origin.

$$\text{Low Frequency} \leftrightarrow \text{Smooth Surface}$$

$$\text{High Frequency} \leftrightarrow \text{Edge}$$

4 The 2D Discrete Cosine Transform (DCT)

4.1 1D DCT

The one-dimensional DCT is defined as:

$$C(u) = a(u) \sum_{x=0}^{N-1} f(x) \cos\left[\frac{(2x+1)u\pi}{2N}\right], \quad 0 \le u \le N-1$$

where the scaling factor $a(u)$ is:

$$a(u) = \begin{cases} \sqrt{\frac{1}{N}}, & \text{if } u = 0 \\ \sqrt{\frac{2}{N}}, & \text{if } u = 1, 2, \ldots, N-1 \end{cases}$$

Inverse Transform

The inverse 1D DCT is given by:

$$f(x) = \sum_{u=0}^{N-1} a(u) C(u) \cos\left[\frac{(2x+1)u\pi}{2N}\right]$$

The signal is projected onto **real sinusoids** instead of complex exponentials.

4.2 2D DCT

The two-dimensional DCT is:

$$C(u,v) = a(u)a(v) \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \cos\left[\frac{(2x+1)u\pi}{2M}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

Inverse 2D DCT

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} a(u)a(v)C(u,v) \cos\left[\frac{(2x+1)u\pi}{2M}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

4.3 Remarks

- DCT projects signals onto real cosine bases.
- DCT has better **energy compaction** properties compared to the DFT.
- It is considered the real-valued version of the DFT.

5 The Walsh-Hadamard Transform

5.1 Binary Representation

Each integer x can be represented in binary as:

$$x = (b_{n-1}(x)\ b_{n-2}(x)\ \dots\ b_0(x))$$

Example: For $x = 6$

$$b_2(x) = 1, \quad b_1(x) = 1, \quad b_0(x) = 0$$

5.2 Definition

The Walsh transform is defined as:

$$W(u) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} f(x) \prod_{i=0}^{n-1} (-1)^{b_i(x) \cdot b_{n-1-i}(u)}$$

or equivalently:

$$W(u) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} f(x) (-1)^{\sum_{i=0}^{n-1} b_i(x) \cdot b_{n-1-i}(u)}$$

Where:

$$N = 2^n, \quad b_i(x) \in \{0, 1\}$$

5.3 Inverse Transform

$$f(x) = \frac{1}{\sqrt{N}} \sum_{u=0}^{N-1} W(u) \prod_{i=0}^{n-1} (-1)^{b_i(x) \cdot b_{n-1-i}(u)}$$

or equivalently:

$$f(x) = \frac{1}{\sqrt{N}} \sum_{u=0}^{N-1} W(u) (-1)^{\sum_{i=0}^{n-1} b_i(x) \cdot b_{n-1-i}(u)}$$

5.4 Matrix Property

We can show:

$$I = N \cdot T^{-1} = T^T$$

This implies that we can easily get the inverse matrix.

6 2D Walsh and Hadamard Transforms

6.1 2D Walsh Transform

$$W(u,v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)+b_i(y)b_{n-1-i}(v)}$$

Inverse transform:

$$f(x,y) = \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} W(u,v)\prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)+b_i(y)b_{n-1-i}(v)}$$

6.2 Hadamard Transform

The Hadamard transform is similar to the Walsh transform:

$$H(u,v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)(-1)^{\sum_{i=0}^{n-1}[b_i(x)b_{n-1-i}(u)+b_i(y)b_{n-1-i}(v)]}$$

6.3 Domain Transformation and Sequencing

The Walsh and Hadamard transforms move data from one domain to another (e.g., from spatial to frequency). We interpret frequency as the number of zero-crossings or sign changes in the basis vector. This is referred to as **sequency**.

Ordered Basis Example (Hadamard Matrix):

$$\frac{1}{\sqrt{N}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

Rows correspond to increasing sequency from 0 to 7.

7 Karhunen–Loève Transform (KLT)

7.1 Eigenvalues and Eigenvectors

Given a covariance matrix C, the KLT relies on solving:

$$C\mathbf{e} = \lambda\mathbf{e}$$

where:

- $\mathbf{e}$: eigenvector
- λ: eigenvalue

7.2 Image Vector Representation

Suppose we have n images sampled at the same location (x,y), forming a vector:

$$\vec{X}_{x,y} = \begin{bmatrix} x_1(x,y) \\ x_2(x,y) \\ \vdots \\ x_n(x,y) \end{bmatrix}$$

7.3 Mean Vector and Covariance Matrix

The mean vector:

$$\vec{m}_x(x,y) = \mathbb{E}[\vec{X}(x,y)] = \begin{bmatrix} m_1(x,y) & m_2(x,y) & \dots & m_n(x,y) \end{bmatrix}^T$$

Mean value for component i:

$$m_i = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} x_i(k,l)$$

Covariance matrix:

$$\mathrm{Cov}(\vec{X}) = \mathbb{E}[(\vec{X} - \vec{m}_x)(\vec{X} - \vec{m}_x)^T] \quad \text{of size } n \times n$$

7.4 Covariance Matrix Properties

- The covariance matrix is symmetric and positive definite.
- $\mathbf{x}^T C \mathbf{x} > 0$ for all non-zero real vectors $\mathbf{x}$.

7.5 Transform Matrix Construction

Let matrix A be formed by stacking the eigenvectors of the covariance matrix C:

$$A = \begin{bmatrix} e_1^T \\ e_2^T \\ \vdots \\ e_n^T \end{bmatrix}, \quad \text{with corresponding eigenvalues } \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$$

Transformation:

$$\mathbf{y} = A(\mathbf{x} - \vec{m}_x)$$

It's called the Karhunen–Loève Transform.

7.6 Proof of Zero-Mean Output

$$\vec{m}_y = \mathbb{E}[\mathbf{y}] = 0$$

Proof:

$$\begin{aligned}
\vec{m}_y &= \mathbb{E}[\mathbf{y}] \\
&= \mathbb{E}[A(\mathbf{x} - \vec{m}_x)] \\
&= A\mathbb{E}[\mathbf{x} - \vec{m}_x] \\
&= A(\mathbb{E}[\mathbf{x}] - \vec{m}_x) \\
&= A(\vec{m}_x - \vec{m}_x) = 0
\end{aligned}$$

7.7 Covariance in Transformed Domain

$$\mathrm{Cov}(\mathbf{y}) = A\mathrm{Cov}(\mathbf{x})A^T$$

Matrix A and its transpose:

$$A = \begin{bmatrix} e_1^T \\ e_2^T \\ \vdots \\ e_n^T \end{bmatrix}, \quad A^T = [e_1 \ e_2 \ \dots \ e_n]$$

Diagonalization:

$$\mathrm{Cov}(\mathbf{x})A^T = C_x[e_1 \ \dots \ e_n] = [\lambda_1 e_1 \ \dots \ \lambda_n e_n]$$

$$A\mathrm{Cov}(\mathbf{x})A^T = \begin{bmatrix} e_1^T \\ \vdots \\ e_n^T \end{bmatrix} [\lambda_1 e_1 \ \dots \ \lambda_n e_n] = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix}$$

This is a diagonal matrix of variances—**uncorrelated components**.

7.8 Reconstruction and Compression

Inverse KLT:

$$\mathbf{x} = A^T\mathbf{y} + \vec{m}_x$$

Keeping only top K components:

$$\hat{\mathbf{x}} = A^T\mathbf{y}_K + \vec{m}_x$$

7.9 Mean Square Error (MSE)

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n} \|x_i - \hat{x}_i\|^2 = \|x - \hat{x}\|^2 = \sum_{j=K+1}^{n} \lambda_j$$

7.10 Interpretation

The Karhunen–Loève Transform allows for decorrelation and energy compaction:

- Most of the signal energy is concentrated in the first few transformed coefficients.
- Enables compression by retaining only significant components.

Illustration: Compressing correlated images by concentrating information into a smaller number of principal components.

8 Image Enhancement

8.1 Classification

Image enhancement techniques can be classified into:

- **Spatial-domain methods** (operate on (x, y))
- **Frequency-domain methods** (operate on (u, v))

8.2 Intensity Transformations

Let $s = T(r)$, where:

- r: input grey level at pixel $f(x, y)$
- s: output grey level at pixel $g(x, y)$

Image Negatives

$$s = T(r) = L - 1 - r$$

Piecewise Linear Transformation

Often used to stretch or compress intensity ranges.

8.3 Log Transformation

$$s = c \cdot \log(1 + r)$$

Enhances the dynamic range of low intensity values.

8.4 Power-Law Transformations

$$s = T(r) = c \cdot r^{\gamma}$$

Used for **gamma correction**:

- $\gamma > 1$: compress higher intensities, darkens image
- $\gamma < 1$: expand higher intensities, brightens image

8.5 Gray Level Slicing

Highlights specific ranges of grey values:

- Preserve intensities in a certain range
- Suppress others or keep them unchanged

8.6 Bit-Plane Slicing

Assume each pixel is represented by 8 bits. Separate an image into its binary layers:

- **MSB (Most Significant Bit)** layers hold visually significant data
- **LSB (Least Significant Bit)** layers capture fine details

9 Histogram Equalization

Consider an image with intensity levels $r_k \in [0, L-1]$, and image size $M \times N$.

9.1 Histogram Definitions

- The number of pixels with intensity r_k is n_k.
- The histogram of the image is the function: $h(r_k) = n_k$.
- The normalized histogram is:

  $$p(r_k) = \frac{n_k}{MN}$$

  This represents the probability of a pixel having grey value r_k.

9.2 Image Contrast Types

- Low contrast: narrow histogram.
- High contrast: widely spread histogram.
- Dark image: histogram biased to low intensity values.
- Bright image: histogram biased to high intensity values.

9.3 Transformation Function

We want a transformation $s = T(r)$, with $T'(r) > 0$, satisfying:

- Monotonically increasing
- One-to-one mapping
- $0 \leq T(r) \leq L-1$ for $0 \leq r \leq L-1$

9.4 Continuous Case

Let $p_r(r)$ be the probability density function (PDF) of input intensities.

We want to find a transformation $s = T(r)$ such that $p_s(s)$ is **uniformly distributed**.

Assume:

$$p_s(s)\,ds = p_r(r)\,dr \Rightarrow p_s(s) = p_r(r)\frac{dr}{ds}$$

To make $p_s(s) = \frac{1}{L-1}$, we integrate:

$$\int_0^s p_s(s)\,ds = \int_0^r p_r(r)\,dr = \frac{1}{L-1}\int_0^s ds = \frac{s}{L-1} \Rightarrow s = (L-1)\int_0^r p_r(r)\,dr$$

9.5 Discrete Version

$$s_k = (L-1)\sum_{j=0}^{k} p(r_j)$$

9.6 Effect of Histogram Equalization

- Enhances image contrast
- Makes the light distribution more uniform
- More visually pleasing for human perception

9.7 Example (Illustrative)

A dark image (e.g., most pixels at intensity 1) is transformed to use a broader intensity range (e.g., intensity 1 to 3), improving visibility and spreading the histogram.

9.8 Histogram Specification (Matching)

We seek a transformation $z = T(r)$ such that the output image has a specified probability density function $p_z(z)$.

Step 1: Equalize $p_r(r)$

$$s = (L-1)\int_0^r p_r(r)\,dr = T(r)$$

Step 2: Equalize $p_z(z)$

$$s = (L-1)\int_0^z p_z(t)\,dt = G(z)$$

Step 3: Invert to get z

$$z = G^{-1}(s) = G^{-1}(T(r)), \quad \text{or } s = T(r) = G(z)$$

Conceptual Flow

$$r \xrightarrow{T(r)} s \xrightarrow{G^{-1}(s)} z$$

Interpretation:

- Adjusts the histogram to match a specified shape (e.g., bimodal, Gaussian).
- Useful for style matching or enhancing specific contrast profiles.

9.9 Local Histogram Equalization

Instead of applying histogram equalization globally, apply it to small regions to enhance local details.

Advantages:

- Enhances contrast in small areas.
- Preserves edge and texture information better.

Process:

- Slide a window (e.g., 3x3 or 5x5) across the image.
- Apply histogram equalization to each window.
- Combine results to form the final image.

10 Spatial Filters in Image Processing

10.1 Model with Noise

$$g(x, y) = f(x, y) + \eta(x, y)$$

Where $\eta(x, y)$ is white noise.

White Noise: A sequence of uncorrelated random variables with zero mean and finite variance. If it follows a normal distribution with zero mean, it's called *white Gaussian noise*.

10.2 Filter Kernels and Convolution

Convolution:

$$g(x, y) = \sum_{k=-K}^{K} \sum_{l=-L}^{L} w(k, l) f(x + k, y + l)$$

Box Filter (Low-Pass):

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \cdot \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

This filter is **separable**.

10.3 Gaussian Filter

Gaussian Kernel:

$$w(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Smooths the image.
- Gaussian in spatial domain remains Gaussian in frequency domain.
- Acts as a low-pass filter.

10.4 Order-Statistic Filters

Median Filter:

$$g(x,y) = \text{median}\{f(x+i, y+j)\}, \quad i = -N, \ldots, N,\ j = -M, \ldots, M$$

- Effective for removing "salt and pepper" noise.
- Non-linear filtering.

10.5 High-Pass Filtering

$$\frac{1}{9}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Properties:

- The sum of coefficients must be zero.
- Emphasizes edges and suppresses smooth areas.

10.6 High-Boost Filtering

Formula:

$$\text{High-boost image} = A \cdot I_{\text{orig}} - I_{\text{low}} = (A-1) \cdot I_{\text{orig}} + I_{\text{high}}$$

Application: Enhances edges more strongly than high-pass filtering.

11 Edge Detection

Edges are abrupt changes in pixel intensity.

11.1 Gradient and Derivatives

Change rate:

$$\frac{\partial f}{\partial x} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

Discrete approximation ($\Delta x = 1$):

$$\frac{\partial f}{\partial x} \approx f(x+1) - f(x)$$

11.2 Second Derivative (Laplacian)

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) - 2f(x) + f(x-1)$$

11.3 Gradient Magnitude

$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$\|\nabla f(x,y)\|_2 = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \quad \text{(Euclidean norm)}$$

$$\|\nabla f(x,y)\|_1 \approx \left|\frac{\partial f}{\partial x}\right| + \left|\frac{\partial f}{\partial y}\right| \quad \text{(Faster to compute)}$$

11.4 Laplacian Operator

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Used to detect areas of rapid intensity change.

12 Image Restoration

12.1 Degradation Model

$$g(i,j) = H[f(i,j)] + n(i,j)$$

For Linear Time (or Space) Invariant (LTI/LSI) systems:

$$g(i,j) = f(i,j) * h(i,j) + n(i,j)$$

Where * denotes convolution.

12.2 Types of Degradation Functions

- **Uniform motion blur:**

$$h(i,j) = \begin{cases} \frac{1}{(2L+1)^2}, & -L \leq i,j \leq L \\ 0, & \text{otherwise} \end{cases}$$

- **Atmospheric turbulence:**

$$h(i,j) = e^{-k(i^2+j^2)^{5/6}}$$

- **Out-of-focus blur:**

$$h(i,j) = \begin{cases} \frac{1}{\pi R^2}, & i^2 + j^2 \leq R^2 \\ 0, & \text{otherwise} \end{cases}$$

12.3 Restoration Quality Metrics

- **BSNR (Blurred Signal-to-Noise Ratio):**

$$\text{BSNR} = 10 \log_{10} \left(\frac{1}{MN} \sum_{i,j} [E\{f(i,j)\} - \bar{f}(i,j)]^2 \Big/ \sigma_n^2 \right)$$

- **ISNR (Improvement SNR):**

$$\text{ISNR} = 10 \log_{10} \left(\frac{\sum_{i,j} [f(i,j) - g(i,j)]^2}{\sum_{i,j} [f(i,j) - \hat{f}(i,j)]^2} \right)$$

12.4 Lexicographic Ordering

Transform image matrix to a 1D vector $\mathbf{y} = H\mathbf{f} + \mathbf{n}$. Matrix H represents the Toeplitz-block structure.

12.5 Inverse Filtering

$$\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)}, \quad f(i,j) = \text{IDFT}[\hat{F}(u,v)]$$

Noise Amplification Problem:

$$\hat{F}(u,v) = \frac{F(u,v)H(u,v) + N(u,v)}{H(u,v)} = F(u,v) + \frac{N(u,v)}{H(u,v)}$$

12.6 Modified Inverse Filter

$$H'(u,v) = \begin{cases} \frac{1}{H(u,v)}, & \text{if } |H(u,v)| \geq \varepsilon \\ 0, & \text{otherwise} \end{cases}$$

12.7 Constrained Least Squares (CLS) Restoration

Objective:

$$\min \|y - Af\|^2 + \lambda \|Cf\|^2$$

Solution:

$$\hat{F}(u,v) = \frac{H^*(u,v)}{|H(u,v)|^2 + \lambda |C(u,v)|^2} G(u,v)$$

12.8 Wiener Filter

Assume:

$$\mathbb{E}[f] = \mathbb{E}[y] \Rightarrow \text{Linear estimator: } \hat{f} = W\mathbb{E}[y]$$
$$\text{Error independent of distortion: } \mathbb{E}[(f - \hat{f})(y - \mathbb{E}[y])^T] = 0$$

Wiener Solution in Frequency Domain:

$$W(u,v) = \frac{H^*(u,v) S_{ff}(u,v)}{|H(u,v)|^2 S_{ff}(u,v) + S_{nn}(u,v)}$$

If $H(u,v) = 1$:

$$W(u,v) = \frac{S_{ff}(u,v)}{S_{ff}(u,v) + S_{nn}(u,v)} = \frac{\text{SNR}}{\text{SNR} + 1}$$

- If SNR $\gg 1$: $W(u,v) \approx 1$
- If SNR $\ll 1$: $W(u,v) \approx \text{SNR}$

13 Image Compression

13.1 Overview

Compression reduces data size in images, video, or audio. It involves two processes:

- **Encoding** (compress)
- **Decoding** (reconstruct)

Compression Ratio:

$$CR = \frac{\text{Input size}}{\text{Output size}} \quad \text{e.g., } \frac{64\text{kbps}}{4\text{kbps}} = 16$$

13.2 Lossless vs. Lossy Compression

- **Lossless:** $\hat{f} = f$
- **Lossy:** $\text{SNR}_{\hat{f}} < \text{SNR}_f$

SNR (Signal-to-Noise Ratio):

$$\text{SNR} = \frac{\text{signal energy}}{\text{noise energy}}$$

14 Information Theory Basics

14.1 Self-Information

$$I(s_i) = \log_2 \left(\frac{1}{p_i} \right) = -\log_2 p_i$$

14.2 Entropy

$$H(S) = \sum_i p_i I(s_i) = -\sum_i p_i \log_2 p_i \quad \text{(bits/symbol)}$$

14.3 Source Model

- **Source:** sequence of symbols from a finite alphabet.
- **DMS (Discrete Memoryless Source):** symbols generated independently.

14.4 Average Length

$$L_{avg} = \sum_{i=1}^{n} p_i \cdot l_i \quad \text{(length of codewords)}$$

$$L_{avg} \geq H(S)$$

15 Huffman Coding

- Prefix code, optimal for a given probability model.
- Binary tree representation.
- $H(S) \leq L_{avg} < H(S) + 1$

Decoding: Traverse the tree based on bits.

16 Coding Extensions and Redundancy

16.1 Nth Extension

Encoding blocks of N symbols improves efficiency:

$$H(S) \leq \frac{L_{avg}}{N} < H(S) + \frac{1}{N}$$

16.2 Redundancy

$$\text{Redundancy} = L_{avg} - H(S)$$

17 Differential Coding

Difference Encoding:

$$g(x, y) = f(x, y) - f(x-1, y) \quad \text{(horizontal)} \quad \text{or} \quad f(x, y) - f(x, y-1)$$

Prediction Residual:

$$r = y - x, \quad \text{where } y = f(a, b, c)$$

18 JPEG Compression Pipeline

1. Divide image into 8×8 blocks.
2. Apply 2D DCT.
3. Quantize DCT coefficients.
4. Use zig-zag scan to serialize.
5. Apply:
    - Differential coding (DC)
    - Run-length coding (AC)
    - Huffman encoding

18.1 DCT and Quantization

$$Z_{i,j} = \left\lfloor \frac{Y_{i,j}}{Q_{i,j}} \right\rfloor$$

18.2 Differential Coding (DC Coefficient)

$$\Delta DC = DC_i - DC_{i-1}$$

18.3 Run-Length Coding (AC Coefficients)

- Use zig-zag order to group zeros.
- Encode using (run, value) pairs.
- Each pair Huffman encoded.

18.4 JPEG Final Structure

$$\text{Original} \to \text{DCT} \to \text{Quantization}$$

$$\text{DC: differential + Huffman}, \quad \text{AC: zig-zag + run-length + Huffman}$$