Image Processing Lecture 8
Hough Transform, Color Theory and Compression

Yonghao Lee

December 10, 2025

Contents

1 Module 1: The Hough Transform

The Hough Transform is a global method used to find shapes (like lines or circles) in an image. Its biggest strength is robustness: it can find shapes even if they are broken, dashed, or partially covered.

1.1 1. Line Detection

To detect lines, we map points from the *image space* to a *parameter space*.

1.1.1 A. The Problem with Slope-Intercept

We usually describe a line as $y = ax + b$.

- **Issue:** Vertical lines have an infinite slope ($a = \infty$). Computers cannot store infinity, so this method fails for vertical edges.

1.1.2 B. Polar Representation (d, θ)

To solve the infinity problem, we use polar coordinates:

$$d = x\cos(\theta) + y\sin(\theta) \tag{1}$$

Where:

- d (rho): The shortest distance from the origin to the line.
- θ (theta): The angle of the line's normal vector.

In Simple Terms

The Duality Principle:

- **One Point in Image → Sine Wave in Parameter Space:** A single dot in the image could belong to infinite lines. In parameter space, this forms a curve.
- **Intersection → Line:** If two curves in parameter space cross at a specific point (d, θ), that point represents the single line connecting the two image dots.

1.2 2. The Algorithm (Voting)

1. **Edges:** Detect edge pixels (using Canny, etc.).
2. **Accumulator:** Create a 2D matrix representing all possible (d, θ) pairs. Initialize to zero.
3. **Vote:** For every edge pixel, calculate every possible line passing through it and increment the corresponding cell in the matrix.
4. **Peak Finding:** The cells with the highest values (most votes) represent the actual lines in the image.

1.3 3. Circle Detection

Equation: $(x - a)^2 + (y - b)^2 = r^2$.

- **Complexity:** A circle needs 3 parameters (a, b, r), requiring a 3D accumulator which is slow $(O(N^3))$.
- **Optimization:** Use the gradient direction. The center of the circle must lie along the gradient vector of the edge, reducing the search space.

2 Module 2: Color Theory

2.1 1. Physics and Biology

Color is a perception, not a physical property.

- **The Spectrum:** Visible light ranges from ∼400nm (Blue) to ∼700nm (Red).
- **The Eye:**
  - **Rods:** Sensitive to intensity (night vision), not color.
  - **Cones:** 3 types (Long, Medium, Short). They integrate the light spectrum into just 3 signals (L, M, S).

2.1.1 Metamerism

Because we compress an infinite spectrum into 3 values, different physical light mixtures can look identical. A yellow laser looks the same as a mix of Red + Green light. These matches are called **Metamers**.

2.2 2. Color Spaces

Space and **Description**
RGB and Additive mixing (Screens). Not perceptually uniform.
HSV and Intuitive (Hue, Saturation, Value). Good for user selection.
CIE-Lab and Perceptually uniform. Euclidean distance matches human vision.
YUV and Separates Luminance (Y) from Chrominance (UV). Used in TV/Compression.

2.3 3. Color Histograms

A distribution of colors in an image.

- **Robustness:** Histograms do not change if the image is rotated or scaled.
- **Comparison:** We compare images by calculating the distance (e.g., Earth Mover's Distance) between their histograms.

3 Module 3: Image Compression

3.1 1. Why Compress?

Raw images are massive (HD Video = 75MB/sec). Compression exploits **redundancy**:

- **Spatial Correlation:** Neighboring pixels are usually similar.
- **Temporal Correlation:** Successive video frames are nearly identical.

3.2 2. Entropy and Lossless Coding

Entropy measures the "unpredictability" of information. Rare events have high information; common events have low information.

$$Entropy = -\sum p(k) \log_2 p(k) \tag{2}$$

3.2.1 Lossless Algorithms

- **Huffman Coding:** Assigns short binary codes to frequent symbols and long codes to rare symbols.
- **Run Length Encoding (RLE):** Stores pairs of $(Color, Length)$. Best for binary images (Fax).
- **LZW:** Dictionary-based. Replaces repeated patterns with an index.

3.3 3. The JPEG Standard (Lossy)

JPEG uses a "Transform Coding" pipeline to discard invisible information.

3.3.1 Step A: Color Conversion ($RGB \to YC_bC_r$)

Separates Brightness (Y) from Color (C_b, C_r). Since the eye is bad at seeing color detail, we can lower the resolution of C_b and C_r without anyone noticing.

3.3.2 Step B: Discrete Cosine Transform (DCT)

Splits the image into 8×8 blocks and converts pixels into **frequencies**.

- **Top-Left:** Low frequencies (smooth gradients). Most important.
- **Bottom-Right:** High frequencies (noise/texture). Least important.

> **In Simple Terms**
>
> **Why DCT?** It concentrates the image energy. Instead of 64 equal pixels, you get a few "strong" coefficients in the corner and many near-zero coefficients everywhere else.

3.3.3 Step C: Quantization (The Magic Step)

We divide the DCT values by a **Quantization Table** and round to integers.

- We divide High Frequencies by **large numbers** $\to$ They become 0.
- We divide Low Frequencies by **small numbers** $\to$ They stay accurate.
- This is the **only** step where information is permanently lost.

3.3.4 Step D: Zig-Zag Scan and Coding

We read the 8×8 matrix in a Zig-Zag pattern. This puts the non-zero values first and creates a long run of zeros at the end, which is very easy to compress with RLE.