Basics of Image Analysis and Rendering

KOUHAIL Oumaima
BENJELLOUN Elghali
LACHGAR Samia
LABZAE Kawtar

27/4/2025

1 General Concept of Digital Image and Its Types

A *digital image* is a discrete representation of a two-dimensional visual scene. Mathematically, it can be modeled as a function

$$f : \{0, 1, \dots, M-1\} \times \{0, 1, \dots, N-1\} \;\to\; \mathbb{R}^k$$

where (i, j) are pixel coordinates and k is the number of channels (e.g. $k = 1$ for gray-scale, $k = 3$ for RGB). Each pixel value $f(i, j)$ encodes intensity or color.

Types of Digital Images

- **Binary images:** $k = 1$ with values in $\{0, 1\}$, often for masks or logical maps.
- **Gray-scale images:** $k = 1$ with intensity $0 \leq f(i, j) \leq L - 1$, where L is the number of gray levels (commonly $L = 256$).
- **Indexed (paletted) images:** Pixel values index into a colormap of true-color entries.
- **True-color (RGB) images:** $k = 3$ channels, each channel $R, G, B \in [0, 255]$ in 8-bit representation.
- **Multispectral / Hyperspectral images:** $k > 3$ bands, capturing reflectance in many narrow spectral bands.

2 Concept of Color Spaces

A *color space* is a mapping between color values and physical quantities of light. It provides a coordinate system for color representation.

- **RGB:** Device-dependent additive model with axes (R, G, B).

- **CMYK:** Subtractive model for print: (C, M, Y, K).
- **HSV / HSL:** Cylindrical transforms of RGB into hue, saturation, and value/lightness.
- **CIE XYZ:** Device-independent space based on human vision; linear combinations of cone responses.
- **CIE L*a*b*:** Perceptually uniform space defined by

  $$L^* = 116\, f\big(Y/Y_n\big) - 16, \quad a^* = 500\big[f(X/X_n) - f(Y/Y_n)\big], \quad b^* = 200\big[f(Y/Y_n) - f(Z/Z_n)\big]$$

  where $f(t) = t^{1/3}$ for $t > 0.008856$.

3 Gray-Scale Codification

Gray-scale codification maps color or multi-channel data to a single intensity value. A common formula (ITU-R BT.601) is

$$Y \;=\; 0.299\, R \;+\; 0.587\, G \;+\; 0.114\, B\,,$$

where Y is the luminance. This respects human perception, weighting green most heavily. Quantization then maps Y to an integer range $[0, L-1]$.

4 General Workflow of Image Analysis

The typical pipeline for image analysis consists of:

1. **Acquisition:** Capture via sensors, producing raw pixel data.
2. **Pre-processing:** Noise reduction (e.g. Gaussian filtering), contrast enhancement (histogram equalization), geometric corrections.
3. **Segmentation:** Partition image into meaningful regions (thresholding, clustering, active contours).
4. **Feature Extraction:** Compute descriptors (edges, textures via Gabor filters, keypoints via SIFT).
5. **Classification/Interpretation:** Assign labels or interpret structures using statistical or machine learning models.
6. **Post-processing:** Refine results (morphological operations: erosion/dilation defined by

   $$(I \ominus B)(x) = \min_{b \in B} I(x+b), \quad (I \oplus B)(x) = \max_{b \in B} I(x-b),$$

   etc.).

5 Three Levels of Image Processing / Analysis

Low-Level: Pixel-based operations without semantic understanding (e.g. filtering, edge detection).

Mid-Level: Grouping pixels into features or regions (segmentation, region growing, shape analysis).

High-Level: Semantic interpretation (object recognition, scene understanding, inference).

6 Rendering in Computer Graphics

Definition and Significance

Rendering is the process of generating a 2D image from 3D scene data (geometry, materials, lights). It is crucial for visualization, simulation, and entertainment.

Rendering Pipeline and Stages

1. **Modeling:** Creation of geometric primitives (meshes, curves).
2. **Transformation:** Apply model, view, and projection matrices:

   $$\mathbf{p}_{\text{clip}} = P\,V\,M\,\mathbf{p}_{\text{model}},$$

   where M=model, V=view, P=projection.
3. **Clipping & Culling:** Remove primitives outside view frustum.
4. **Rasterization:** Convert primitives to fragments/pixels (see Section 8).
5. **Shading:** Compute color using lighting models (Phong, Blinn–Phong, or full *rendering equation*):

   $$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_\Omega f_r(x, \omega_i, \omega_o)\, L_i(x, \omega_i)\, (\omega_i \cdot n)\, d\omega_i.$$
6. **Output Merging:** Depth test, blending, produce final framebuffer.

7 Real-Time vs Offline Rendering

Real-Time Rendering: Targeting interactive rates ($\geq 30\,$fps). Emphasizes performance using simplified lighting (e.g. screen-space effects, precomputed radiance).

Offline Rendering: No strict time constraints; focuses on physical accuracy (path tracing, global illumination, Monte Carlo integration).

8 Rasterization and Its Role in Rendering

Rasterization is the scan-conversion of vector primitives (triangles, lines) into discrete pixels.

- **Triangle Setup:** Compute edge functions $E_i(x, y) = ax+by+c$ to test point inclusion.
- **Barycentric Coordinates:** Interpolate vertex attributes:

$$(u, v, w) = \left(\tfrac{A_{PBC}}{A_{ABC}}, \tfrac{A_{APC}}{A_{ABC}}, \tfrac{A_{ABP}}{A_{ABC}}\right), \quad \mathbf{attrib}_P = u\,\mathbf{attrib}_A + v\,\mathbf{attrib}_B + w\,\mathbf{attrib}_C.$$

- **Algorithms:** Bresenham's line algorithm for edges, midpoint tests.
- **GPU Implementation:** Massively parallel fragment shaders execute per-pixel shading.