Class 5: Image Processing Techniques

Table of Contents

Introduction

Image processing is a fundamental aspect of computer vision and machine learning. It involves manipulating and analyzing images to extract meaningful information or enhance their quality. This comprehensive guide delves into essential image processing techniques, providing detailed explanations, mathematical notations, equations, and practical examples. The key topics covered include:

- Pixels
- Color Spaces (RGB, HSV)
- Image Normalization
- Data Augmentation (Flipping, Rotation, Cropping)
- Convolution and Filtering
- Essential Preprocessing Techniques

By the end of this guide, readers will have a solid understanding of how to preprocess image data effectively for model training, ensuring optimal performance and accuracy.

1. Pixels

1.1 Definition

A **pixel** (short for picture element) is the smallest unit of a digital image or graphic that can be displayed and represented on a digital display device. Pixels are the building blocks of digital images, each representing a single point in the image.

In color images, each pixel typically contains multiple color components (e.g., Red, Green, and Blue in the RGB color space), allowing for the representation of a wide range of colors through the combination of these basic colors.

1.2 Mathematical Representation

An image can be represented as a two-dimensional matrix or array of pixel intensity values. For a grayscale image, the intensity at each pixel location ((i, j)) is a single scalar value:

$$\mathbf{I} = \begin{bmatrix} I_{11} & I_{12} & \dots & I_{1n} \\ I_{21} & I_{22} & \dots & I_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ I_{m1} & I_{m2} & \dots & I_{mn} \end{bmatrix}$$

where:

- $\mathbf{I}$ is the image matrix.
- I_{ij} represents the intensity value at row i and column j.
- m is the number of rows (height of the image).
- n is the number of columns (width of the image).

For a color image in the RGB color space, the image is represented by three matrices corresponding to the Red, Green, and Blue channels:

$\mathbf{I}_{RGB} = \{\mathbf{R}, \mathbf{G}, \mathbf{B}\}$

Each of $\mathbf{R}$, $\mathbf{G}$, and $\mathbf{B}$ is an $m \times n$ matrix representing the intensity values of the respective color channel.

1.3 Example

Grayscale Image Example:

Consider a simple 3×3 grayscale image:

$$\mathbf{I} = \begin{bmatrix} 200 & 150 & 100 \\ 150 & 100 & 50 \\ 100 & 50 & 0 \end{bmatrix}$$

Each element I_{ij} represents the intensity at that pixel, with values typically ranging from 0 (black) to 255 (white) in 8-bit images.

Color Image Example:

For a 2×2 color image in RGB space:

Red channel $\mathbf{R}$:

$$\mathbf{R} = \begin{bmatrix} 255 & 128 \\ 64 & 0 \end{bmatrix}$$

Green channel $\mathbf{G}$:

$$\mathbf{G} = \begin{bmatrix} 0 & 64 \\ 128 & 255 \end{bmatrix}$$

Blue channel $\mathbf{B}$:

$$\mathbf{B} = \begin{bmatrix} 128 & 255 \\ 0 & 64 \end{bmatrix}$$

2. Color Spaces

A **color space** is a specific organization of colors, enabling reproducible representations of color in both analog and digital representations. The choice of color space can significantly impact the processing and analysis of images.

2.1 RGB Color Space

The **RGB color space** is an additive color model where colors are obtained by combining Red, Green, and Blue light in various intensities. It's widely used in digital imaging devices like cameras and displays.

2.1.1 Representation

In the RGB color space, each pixel's color is represented by a three-dimensional vector:

$$\mathbf{p} = \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

where R, G, and B are the intensity values for the Red, Green, and Blue channels, respectively. These values typically range from 0 to 255 in 8-bit images.

An entire RGB image can be represented as three $m \times n$ matrices:

$$\mathbf{I}_{RGB} = \{\mathbf{R}, \mathbf{G}, \mathbf{B}\}$$

2.1.2 Example

Consider a pixel with the following RGB values:

$$\mathbf{p} = \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 120 \\ 200 \\ 150 \end{bmatrix}$$

This pixel has a moderate red intensity, high green intensity, and medium blue intensity, resulting in a specific shade of color when displayed.

2.2 HSV Color Space

The **HSV color space** represents colors using three components:

- **Hue (H):** Represents the color type, ranging from 0° to 360°.
- **Saturation (S):** Represents the vibrancy of the color, ranging from 0 to 1.
- **Value (V):** Represents the brightness of the color, ranging from 0 to 1.

HSV is particularly useful for tasks where color descriptions align more closely with human perception.

2.2.1 Conversion from RGB to HSV

Converting an RGB pixel to HSV involves several steps.

1. **Normalize RGB values to [0, 1]:**

$$R' = \frac{R}{255}, \quad G' = \frac{G}{255}, \quad B' = \frac{B}{255}$$

2. **Compute maximum and minimum values among R', G', B':**

$$C_{\max} = \max(R', G', B'), \quad C_{\min} = \min(R', G', B')$$

3. **Compute the difference (Δ):**

   $$\Delta = C_{\max} - C_{\min}$$

4. **Compute Hue (H):**

   $$H = \begin{cases} 0^\circ, & \text{if } \Delta = 0 \\ 60^\circ \times \left(\frac{G'-B'}{\Delta} \mod 6\right), & \text{if } C_{\max} = R' \\ 60^\circ \times \left(\frac{B'-R'}{\Delta} + 2\right), & \text{if } C_{\max} = G' \\ 60^\circ \times \left(\frac{R'-G'}{\Delta} + 4\right), & \text{if } C_{\max} = B' \end{cases}$$

5. **Compute Saturation (S):**

   $$S = \begin{cases} 0, & \text{if } C_{\max} = 0 \\ \frac{\Delta}{C_{\max}}, & \text{otherwise} \end{cases}$$

6. **Compute Value (V):**

   $$V = C_{\max}$$

2.2.2 Example

Convert RGB pixel $(R = 255,\ G = 128,\ B = 64)$ to HSV:

1. **Normalize RGB values:**

   $$R' = \frac{255}{255} = 1.0, \quad G' = \frac{128}{255} \approx 0.502, \quad B' = \frac{64}{255} \approx 0.251$$

2. **Compute $C_{\max}$, $C_{\min}$, and Δ:**

   $$C_{\max} = \max(1.0,\ 0.502,\ 0.251) = 1.0$$

   $$C_{\min} = \min(1.0,\ 0.502,\ 0.251) = 0.251$$

   $$\Delta = C_{\max} - C_{\min} = 1.0 - 0.251 = 0.749$$

3. **Compute Hue (H):**

   Since $C_{\max} = R'$:

   $$H = 60^\circ \times \left(\frac{G' - B'}{\Delta} \mod 6\right) = 60^\circ \times \left(\frac{0.502 - 0.251}{0.749} \mod 6\right) = 60^\circ \times (0.335) \approx 20.1^\circ$$

4. **Compute Saturation (S):**

   $$S = \frac{\Delta}{C_{\max}} = \frac{0.749}{1.0} = 0.749$$

5. **Compute Value (V):**

   $$V = C_{\max} = 1.0$$

Result:

$$(H,\ S,\ V) = (20.1^\circ,\ 0.749,\ 1.0)$$

3. Image Normalization

Image normalization is the process of adjusting pixel intensity values to a common scale, enhancing contrast, or preparing data for further processing. It improves the convergence of machine learning models by ensuring that features are on a similar scale.

3.1 Min-Max Normalization

Min-Max Normalization scales data to a fixed range, usually [0, 1] or [-1, 1].

3.1.1 Mathematical Formula

Given an original pixel value I_{ij}, the normalized pixel value I'_{ij} is computed as:

$$I'_{ij} = a + \left(\frac{I_{ij} - I_{\min}}{I_{\max} - I_{\min}} \right) \times (b - a)$$

where:

- $I_{\min}$ and $I_{\max}$ are the minimum and maximum pixel intensity values in the image.
- $[a,\ b]$ is the desired range for normalization.

For normalization to [0, 1]:

$$I'_{ij} = \frac{I_{ij} - I_{\min}}{I_{\max} - I_{\min}}$$

3.1.2 Example

Normalize an image with pixel values ranging from 50 to 200 to the range [0, 1]:

Given:

- $I_{\min} = 50$
- $I_{\max} = 200$
- Original pixel $I_{ij} = 125$

Compute normalized pixel:

$$I'_{ij} = \frac{125 - 50}{200 - 50} = \frac{75}{150} = 0.5$$

Thus, a pixel value of 125 is normalized to 0.5.

3.2 Z-Score Normalization

Also known as **standardization**, Z-score normalization scales data to have a mean of 0 and a standard deviation of 1.

3.2.1 Mathematical Formula

Given an original pixel value I_{ij}, the standardized pixel value I'_{ij} is computed as:

$$I'_{ij} = \frac{I_{ij} - \mu}{\sigma}$$

where:

- μ is the mean of all pixel values in the image.
- σ is the standard deviation of the pixel values.

3.2.2 Example

Standardize an image with pixel values:

$$\mathbf{I} = \begin{bmatrix} 100 & 150 & 200 \\ 150 & 200 & 250 \\ 200 & 250 & 300 \end{bmatrix}$$

1. **Compute Mean (μ):**

$$\mu = \frac{1}{9}(100 + 150 + 200 + 150 + 200 + 250 + 200 + 250 + 300) = \frac{1800}{9} = 200$$

2. **Compute Standard Deviation (σ):**

$$\sigma = \sqrt{\frac{1}{9}\sum(I_{ij} - \mu)^2} = \sqrt{\frac{1}{9}(10000 + 2500 + 0 + 2500 + 0 + 2500 + 0 + 2500 + 10000)} = \sqrt{\frac{30000}{9}} = \sqrt{3333.33} \approx 57.735$$

3. **Standardize Pixel $I_{11} = 100$:**

$$I'_{11} = \frac{100 - 200}{57.735} = \frac{-100}{57.735} \approx -1.732$$

4. Data Augmentation

Data augmentation is a technique used to increase the diversity and size of a dataset by applying various transformations to existing data. This helps improve the robustness and generalization ability of machine learning models.

4.1 Flipping

Flipping involves mirroring the image along a specific axis.

4.1.1 Horizontal Flip

Mathematical Representation:

For an image of width n, the horizontally flipped pixel I'_{ij} is:

$$I'_{ij} = I_{i,\, n-j+1}$$

This operation mirrors the image along the vertical axis passing through the center of the image.

4.1.2 Vertical Flip

Mathematical Representation:

For an image of height m, the vertically flipped pixel I'_{ij} is:

$$I'_{ij} = I_{m-i+1,\, j}$$

This operation mirrors the image along the horizontal axis passing through the center of the image.

4.1.3 Example

Given a 3×3 image:

$$\mathbf{I} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Horizontal Flip:

$$\mathbf{I}_{\text{hflip}} = \begin{bmatrix} 3 & 2 & 1 \\ 6 & 5 & 4 \\ 9 & 8 & 7 \end{bmatrix}$$

Vertical Flip:

$$\mathbf{I}_{\text{vflip}} = \begin{bmatrix} 7 & 8 & 9 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{bmatrix}$$

4.2 Rotation

Rotating an image involves rotating it by a specified angle θ around its center.

4.2.1 Mathematical Representation

The new coordinates (x', y') of a pixel after rotation are given by:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x - x_c \\ y - y_c \end{bmatrix} + \begin{bmatrix} x_c \\ y_c \end{bmatrix}$$

where:

- (x, y) are the original coordinates.
- (x_c, y_c) are the coordinates of the center of the image.
- θ is the rotation angle in radians.

4.2.2 Example

Rotate a point $(x, y) = (2, 3)$ around the center $(x_c, y_c) = (1, 1)$ by $90°$ (or $\theta = \frac{\pi}{2}$ radians):

1. **Translate point to origin:**

   $$x_{\text{shifted}} = x - x_c = 2 - 1 = 1$$

   $$y_{\text{shifted}} = y - y_c = 3 - 1 = 2$$

2. **Apply rotation matrix:**

   $$\begin{bmatrix} x'_{\text{shifted}} \\ y'_{\text{shifted}} \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{\pi}{2}\right) & -\sin\left(\frac{\pi}{2}\right) \\ \sin\left(\frac{\pi}{2}\right) & \cos\left(\frac{\pi}{2}\right) \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} -2 \\ 1 \end{bmatrix}$$

3. **Translate point back:**

   $$x' = x'_{\text{shifted}} + x_c = -2 + 1 = -1$$

   $$y' = y'_{\text{shifted}} + y_c = 1 + 1 = 2$$

Result:

The point $(2, 3)$ is rotated to $(-1, 2)$.

4.3 Cropping

Cropping involves extracting a specific region of interest (ROI) from an image.

4.3.1 Mathematical Representation

Let $(x_0,\ y_0)$ be the top-left coordinate of the ROI, and w and h be its width and height. The cropped image $\mathbf{I}_{\text{crop}}$ consists of pixels:

$$\mathbf{I}_{\text{crop}} = \{I_{ij} \mid x_0 \leq i < x_0 + w, \quad y_0 \leq j < y_0 + h\}$$

4.3.2 Example

Given a 5×5 image:

$$\mathbf{I} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix}$$

Crop a 3×3 region starting from $(x_0,\ y_0) = (1,\ 1)$:

$$\mathbf{I}_{\text{crop}} = \begin{bmatrix} 1 & 2 & 3 \\ 6 & 7 & 8 \\ 11 & 12 & 13 \end{bmatrix}$$

5. Convolution and Filtering

5.1 Convolution Operation

5.1.1 Mathematical Definition

Convolution is a fundamental operation in image processing used for filtering and feature extraction. It involves sliding a kernel (also known as a filter) over an image to compute a weighted sum of pixel values, producing a new image that emphasizes certain features.

The convolution of an image $\mathbf{I}$ with a kernel $\mathbf{K}$ is defined as:

$$O_{ij} = \sum_{m=-k}^{k} \sum_{n=-l}^{l} I_{i+m,\, j+n} \cdot K_{k+m,\, l+n}$$

where:

- O_{ij} is the output pixel value at position $(i,\ j)$.
- $I_{i+m,\, j+n}$ is the input pixel value at position $(i+m,\ j+n)$.
- $K_{k+m,\, l+n}$ is the kernel value at position $(k+m,\ l+n)$.
- The kernel size is $(2k+1) \times (2l+1)$.

5.1.2 Properties of Convolution

- **Linearity**: Convolution is a linear operation.

  $$\mathbf{I} * (a\mathbf{K}_1 + b\mathbf{K}_2) = a(\mathbf{I} * \mathbf{K}_1) + b(\mathbf{I} * \mathbf{K}_2)$$

- **Commutativity**:

  $$\mathbf{I} * \mathbf{K} = \mathbf{K} * \mathbf{I}$$

- **Associativity**:

  $$\mathbf{I} * (\mathbf{K}_1 * \mathbf{K}_2) = (\mathbf{I} * \mathbf{K}_1) * \mathbf{K}_2$$

- **Distributivity**:

  $$\mathbf{I} * (\mathbf{K}_1 + \mathbf{K}_2) = \mathbf{I} * \mathbf{K}_1 + \mathbf{I} * \mathbf{K}_2$$

5.2 Common Filters

Convolution is used with different kernels to achieve various effects in image processing.

5.2.1 Edge Detection

Edge detection filters highlight regions in the image with high intensity gradients.

Sobel Operator:

- **Horizontal Sobel Kernel** $\mathbf{K}_x$:

  $$\mathbf{K}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

- **Vertical Sobel Kernel** $\mathbf{K}_y$:

  $$\mathbf{K}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Prewitt Operator:

- **Horizontal Prewitt Kernel**:

  $$\mathbf{K}_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

- **Vertical Prewitt Kernel**:

$$\mathbf{K}_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Laplacian Operator:

- **Laplacian Kernel**:

$$\mathbf{K} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

5.2.2 Blurring

Blurring filters smooth the image by reducing noise and detail.

Gaussian Blur:

The Gaussian kernel is defined by the Gaussian function:

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

An example of a 3×3 Gaussian kernel with $\sigma = 1$:

$$\mathbf{K} = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Mean Filter (Box Blur):

A simple averaging filter:

$$\mathbf{K} = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

5.2.3 Sharpening

Sharpening filters enhance edges and fine details in the image.

Unsharp Masking:

The sharpening kernel:

$$\mathbf{K} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

High-Pass Filter:

Emphasizes high-frequency components:

$$\mathbf{K} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

5.2.4 Examples

Applying the Sobel Operator for Edge Detection

Given a grayscale image:

$$\mathbf{I} = \begin{bmatrix} 10 & 10 & 10 \\ 10 & 20 & 10 \\ 10 & 10 & 10 \end{bmatrix}$$

Compute the gradients:

- Horizontal gradient $G_x = \mathbf{I} * \mathbf{K}_x$
- Vertical gradient $G_y = \mathbf{I} * \mathbf{K}_y$

Compute the gradient magnitude:

$$G = \sqrt{G_x^2 + G_y^2}$$

Calculations:

At the center pixel (position $(2,\ 2)$):

- **Compute** G_x:

$$\begin{aligned} G_x &= (-1) \times 10 + 0 \times 10 + 1 \times 10 \\ &+ (-2) \times 10 + 0 \times 20 + 2 \times 10 \\ &+ (-1) \times 10 + 0 \times 10 + 1 \times 10 \\ &= 0 \end{aligned}$$

- **Compute** G_y:

$$\begin{aligned} G_y &= (-1) \times 10 + (-2) \times 10 + (-1) \times 10 \\ &+ 0 \times 10 + 0 \times 20 + 0 \times 10 \\ &+ 1 \times 10 + 2 \times 10 + 1 \times 10 \\ &= 0 \end{aligned}$$

- **Gradient Magnitude**:

$$G = \sqrt{0^2 + 0^2} = 0$$

Result: The center pixel does not represent an edge in this case.

6. Essential Techniques for Preprocessing Image Data for Model Training

Preprocessing is a critical step in preparing image data for machine learning models. Proper preprocessing can improve model accuracy, reduce training time, and enhance generalization.

6.1 Resizing

Adjusting images to a uniform size is essential when input dimensions are fixed for a model.

6.1.1 Mathematical Representation

Given an original image $\mathbf{I}$ of size $m \times n$ and a desired size $M \times N$, the scaling factors are:

$$s_x = \frac{M}{m}, \quad s_y = \frac{N}{n}$$

Each pixel I'_{ij} in the resized image corresponds to:

$$I'_{ij} = I_{\frac{i}{s_x}, \frac{j}{s_y}}$$

6.1.2 Interpolation Methods

- **Nearest Neighbor Interpolation**:

  Assigns the value of the nearest pixel.

- **Bilinear Interpolation**:

  Considers the closest 2×2 neighborhood of known pixel values surrounding the unknown pixel.

- **Bicubic Interpolation**:

  Uses the closest 4×4 neighborhood, resulting in smoother images.

Bilinear Interpolation Formula:

Given four neighboring pixels $Q_{11}, Q_{12}, Q_{21}, Q_{22}$ at coordinates $(x_1, y_1), (x_1, y_2), (x_2, y_1), (x_2, y_2)$, the interpolated value at (x, y) is:

$$I(x, y) = \frac{1}{(x_2 - x_1)(y_2 - y_1)} [Q_{11}(x_2 - x)(y_2 - y) + Q_{21}(x - x_1)(y_2 - y) + Q_{12}(x_2 - x)(y - y_1) + Q_{22}(x - x_1)(y - y_1)$$

6.2 Mean Subtraction

Subtracting the mean pixel value centers the data, making it zero-centered.

6.2.1 Mathematical Formula

Compute the mean pixel value μ:

$$\mu = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij}$$

Subtract the mean from each pixel:

$$I'_{ij} = I_{ij} - \mu$$

6.2.2 Example

Given:

$$\mathbf{I} = \begin{bmatrix} 50 & 60 \\ 70 & 80 \end{bmatrix}$$

Compute mean:

$$\mu = \frac{1}{4}(50 + 60 + 70 + 80) = 65$$

Subtract mean:

$$\mathbf{I}' = \begin{bmatrix} 50 - 65 & 60 - 65 \\ 70 - 65 & 80 - 65 \end{bmatrix} = \begin{bmatrix} -15 & -5 \\ 5 & 15 \end{bmatrix}$$

6.3 Standardization

Standardization scales data to have zero mean and unit variance.

6.3.1 Mathematical Formula

Given mean μ and standard deviation σ:

$$I'_{ij} = \frac{I_{ij} - \mu}{\sigma}$$

Compute standard deviation:

$$\sigma = \sqrt{\frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (I_{ij} - \mu)^2}$$

6.3.2 Example

Using the previous example:

Compute standard deviation:

$$\sigma = \sqrt{\frac{1}{4}[(-15)^2 + (-5)^2 + 5^2 + 15^2]} = \sqrt{\frac{1}{4}(225 + 25 + 25 + 225)} = \sqrt{125} = 11.18$$

Standardize:

$$\mathbf{I}' = \begin{bmatrix} \frac{-15}{11.18} & \frac{-5}{11.18} \\ \frac{5}{11.18} & \frac{15}{11.18} \end{bmatrix} \approx \begin{bmatrix} -1.34 & -0.45 \\ 0.45 & 1.34 \end{bmatrix}$$

6.4 Histogram Equalization

Histogram equalization improves the contrast of images by redistributing pixel intensities.

6.4.1 Mathematical Concept

1. **Compute the histogram** $h(I)$ of the image.
2. **Compute the cumulative distribution function (CDF)**:

   $$cdf(I) = \sum_{k=0}^{I} h(k)$$

3. **Normalize the CDF**:

   $$cdf_{\text{norm}}(I) = \frac{cdf(I) - cdf_{\text{min}}}{(mn) - cdf_{\text{min}}} \times (L - 1)$$

   where L is the number of possible intensity levels (e.g., 256 for 8-bit images).

4. **Map the original pixel values** to equalized values:

   $I'_{ij} = cdf_{\text{norm}}(I_{ij})$

6.4.2 Example

Consider a grayscale image with pixel values:

$$\mathbf{I} = \begin{bmatrix} 52 & 55 & 61 \\ 59 & 79 & 61 \\ 85 & 59 & 65 \end{bmatrix}$$

1. **Compute histogram** $h(I)$.
2. **Compute CDF**.
3. **Normalize CDF and map pixels**.

Due to the complexity, we can refer to standard algorithms or software tools for computation.

7. Advanced Techniques

7.1 Fourier Transform

The **Fourier Transform** decomposes an image into its sinusoidal components, analyzing the frequency content of the image.

7.1.1 Mathematical Definition

The 2D Discrete Fourier Transform (DFT) of an image $\mathbf{I}$ of size $m \times n$ is:

$$F(u,\ v) = \sum_{x=0}^{m-1} \sum_{y=0}^{n-1} I(x,\ y) e^{-j2\pi\left(\frac{ux}{m} + \frac{vy}{n}\right)}$$

The inverse DFT is:

$$I(x,\ y) = \frac{1}{mn} \sum_{u=0}^{m-1} \sum_{v=0}^{n-1} F(u,\ v) e^{j2\pi\left(\frac{ux}{m} + \frac{vy}{n}\right)}$$

where $j = \sqrt{-1}$.

7.1.2 Applications

- **Filtering in the Frequency Domain**: High-pass and low-pass filters.
- **Image Compression**: JPEG uses the Discrete Cosine Transform (DCT), a variant of Fourier Transform.
- **Pattern Recognition**: Identifying repetitive patterns.

7.1.3 Example

Applying a Low-Pass Filter:

1. **Compute the DFT** $F(u,\ v)$ of the image.
2. **Multiply** $F(u,\ v)$ by a filter $H(u,\ v)$ that attenuates high frequencies.
3. **Compute the inverse DFT** to get the filtered image.

7.2 Wavelet Transform

The **Wavelet Transform** provides a time-frequency representation of the signal, offering advantages over Fourier Transform in analyzing non-stationary signals.

7.2.1 Mathematical Definition

The Continuous Wavelet Transform (CWT) of a signal $f(t)$ is:

$$W(a,\ b) = \int_{-\infty}^{\infty} f(t)\psi^*\left(\frac{t-b}{a}\right) dt$$

where:

- $\psi(t)$ is the mother wavelet.
- a is the scale parameter.
- b is the translation parameter.
- $\psi^*(t)$ is the complex conjugate of $\psi(t)$.

7.2.2 Applications

- **Image Compression**: Used in JPEG 2000.
- **Denoising**: Removes noise while preserving important features.
- **Feature Extraction**: Captures localized features in images.

7.2.3 Example

Wavelet Denoising:

1. **Apply Wavelet Transform** to the noisy image.
2. **Threshold the wavelet coefficients** to remove noise.
3. **Reconstruct the image** using the inverse Wavelet Transform.

8. Conclusion

This comprehensive guide has explored essential image processing techniques crucial for preparing image data for machine learning model training. Understanding these techniques enables practitioners to enhance image quality, extract meaningful features, and improve model performance.

Key takeaways include:

- **Pixels** are the fundamental units of digital images.
- **Color Spaces** like RGB and HSV provide different ways to represent color information.
- **Image Normalization** ensures consistent pixel value ranges.
- **Data Augmentation** increases dataset diversity and robustness.
- **Convolution and Filtering** are powerful tools for feature extraction.
- **Preprocessing Techniques** like resizing, mean subtraction, and standardization are vital for model readiness.
- **Advanced Techniques** like Fourier and Wavelet Transforms offer deeper insights into image content.

9. References

1. **Gonzalez, R. C., & Woods, R. E. (2008).** *Digital Image Processing*. Pearson.
2. **Goodfellow, I., Bengio, Y., & Courville, A. (2016).** *Deep Learning*. MIT Press.
3. **OpenCV Documentation:** https://docs.opencv.org/
4. **Szeliski, R. (2010).** *Computer Vision: Algorithms and Applications*. Springer.
5. **Pratt, W. K. (2007).** *Digital Image Processing: PIKS Scientific Inside*. Wiley-Interscience.
6. **Mallat, S. (1999).** *A Wavelet Tour of Signal Processing*. Academic Press.
7. **Strang, G., & Nguyen, T. (1996).** *Wavelets and Filter Banks*. Wellesley-Cambridge Press.