# Intensity Histogram

## Outline
- What is histogram
  - How to calculate histogram
- Grayscale transformations and their effects on histograms
  - Negative image
  - Brightness adjustment
  - Contrast adjustment
  - Gamma correction
  - Color balance

---

## What is Histogram?
- A function that depicts the distribution of intensity levels in a given image.
  - The input of the function is between [0, gray_level).
  - The output depicts how often the intensity is used in the image.

---

## Histogram Function
- Unnormalized histogram:
  - \( H(x) \): # of pixels whose intensities are \( x \).
  - Values of \( H(x) \) are integers.
  - \( \sum H(x) = \) total # of pixels.
- Normalized histogram:
  - \( h(x) \): % of pixels whose intensities are \( x \).
  - Values of \( h(x) \) are float numbers between [0,1].
  - \( h(x) = \frac{H(x)}{\text{total # of pixels}} \).
  - \( \sum h(x) = 1 \).

---

## Calculation of Histogram
- Initialize \( H[] \) and \( h[] \) arrays.
- For each pixel \( (q, p) \):
  - Increment \( H[F[q][p]] \).
- Normalize \( h[] \) arrays.

---

## What Can We Tell from Histogram?
- Low contrast
- Underexposed
- Overexposed
- High contrast

---
Grayscale Transformation

- Characterizes:
  - The calculation is solely based on the intensity of the current pixel
  - Convert one grayscale value to another grayscale value
  - Some transformation functions are invertible
  - The corresponding inverse function can be used to restore the original image

Negative

- The simplest grayscale transformation
- Objective:
  - Reverse the color or intensity of the image
- Input:
  - Original image: \( F[p, q] \)
- Output:
  - New image: \( G[p, q] \)

Implementation for Negative

```java
for (int y = 0; y < height; y++)
    for (int x = 0; x < width; x++) {
        Color clr = new Color(source.image.getRGB(x, y));
        int red = 255 - clr.getRed();
        int green = 255 - clr.getGreen();
        int blue = 255 - clr.getBlue();
        target.image.setRGB(x, y, red << 16 | green << 8 | blue);
    }
```

Brightness Adjustment

- Objective:
  - Make the image brighter or darker than before.
- Input:
  - Original image: \( F[p, q] \)
  - Brightness adjustment: offset
  - \( -\text{gray-level} < \text{offset} < \text{gray-level} \)
- Output:
  - New image: \( G[p, q] \)

What Happened?

```java
for (int y = 0; y < height; y++)
    for (int x = 0; x < width; x++) {
        Color clr = new Color(source.image.getRGB(x, y));
        int red = clr.getRed() + offset;
        int green = clr.getGreen() + offset;
        int blue = clr.getBlue() + offset;
        target.image.setRGB(x, y, red << 16 | green << 8 | blue);
    }
```

Revised Implementation

```java
for (int y = 0; y < height; y++)
    for (int x = 0; x < width; x++) {
        Color clr = new Color(source.image.getRGB(x, y));
        int red = clr.getRed() + offset;
        int green = clr.getGreen() + offset;
        int blue = clr.getBlue() + offset;
        target.image.setRGB(x, y, red < 8 ? 8 : red);
    }
```
**Contrast Adjustment**

- **Objective:**
  - Make the image contrast stronger or weaker
- **Input:**
  - Original image: F[p,q]
  - Contrast adjustment: coeff.
- **Output:**
  - Updated image: F[p,q]

**Implementation for Contrast Adjustment**

```java
for (int y=0; y<height; y++)
    for (int x=0; x<width; x++) {
        Color clr = new Color(source.image.getRGB(x, y));
        int red = (int)(clr.getRed() * coeff + offset);
        int green = (int)(clr.getGreen() * coeff + offset);
        ...
        target.image.setRGB(x, y, red << 16 | green << 8 | blue);
    }
```

**Gamma Correction**

- Some transformations are defined using power functions:
  - T(x) = c·x^γ
  - c and γ are positive constants
- Widely used in correcting the power-law response of CRT monitors
- Called Gamma correction

**Gamma Correction for Monitors**

- γ = 0.5
- γ = 2
- γ = 1

**Log Transformations**

- Defined using log function
  - T(x) = c·log(1+x)
  - c is a constant
  - x is assumed to be positive
- For display images with large dynamic range
  - Such as Fourier spectrum
  - Input 0~10^6 -> output 0~6

**Color Balance**

- Global adjustment of the intensities of the colors
- The goal is to render specific colors (e.g. neutral colors) correctly
- Can be achieved by adjusting RGB channels separately
Color Balance Results