Assignment for Chapter 02: Image Representation and Basic Operations

Objective:

To understand the representation of digital images and perform fundamental operations such as cropping, resizing, flipping, and rotation.

Assignment Tasks

Part 1: Theory

1. Explain the following image representations with examples:
   - Grayscale Images
   - RGB Images
   - Binary Images
2. Describe the significance of **image resolution** and **aspect ratio**.
3. Discuss the difference between **lossy** and **lossless** image compression with examples.
4. Write the mathematical formula for resizing an image using bilinear interpolation.
   Example:
   $$I(x, y) = \sum_{i=0}^{1} \sum_{j=0}^{1} w(i, j) \cdot f(x + i, y + j)$$

Part 2: Practical Tasks

1. **Image Cropping**
   - Write a program to crop a specific region (e.g., top-left 100x100 pixels) of an image and display it.
2. **Image Resizing**
   - Resize the image to:
     - Half its original dimensions.
     - Double its original dimensions.
   - Ensure the aspect ratio is preserved.
3. **Image Flipping**
   - Perform horizontal, vertical, and diagonal flips on the image.

4. **Image Rotation**
    - Rotate the image by:
        - 90 degrees clockwise.
        - 45 degrees around its center.
5. **Aspect Ratio Adjustment**
    - Resize the image to a fixed width of 300 pixels, adjusting the height to maintain the aspect ratio.
6. **Arithmetic Operations**
    - Add and subtract two images of the same dimensions.
    - Multiply an image by a scalar value to increase its brightness.
7. **Bitwise Operations**
    - Perform AND, OR, and XOR operations on two binary images.

Sample Solutions

Part 1: Theory Solutions

1. **Image Representations**
    - **Grayscale**: Single intensity channel, values range from 0 (black) to 255 (white).
    - **RGB**: Three channels representing red, green, and blue.
    - **Binary**: Pixel values are 0 or 1, representing black and white.
2. **Image Resolution**:
   The number of pixels in an image, affecting its clarity. Aspect ratio maintains the width-to-height proportion.
3. **Lossy vs Lossless Compression**:
    - **Lossy**: JPEG, sacrifices quality for smaller size.
    - **Lossless**: PNG, preserves original quality.

Part 2: Practical Solutions

1. **Image Cropping**

```
import cv2
import matplotlib.pyplot as plt

image_path = "/path/to/image.jpg"
image = cv2.imread(image_path)
cropped_image = image[0:100, 0:100]  # Crop top-left 100x100 region

plt.imshow(cv2.cvtColor(cropped_image, cv2.COLOR_BGR2RGB))
plt.title("Cropped Image")
plt.axis("off")
plt.show()
```

2. **Image Resizing**

```
# Half and double the dimensions
half_size = cv2.resize(image, None, fx=0.5, fy=0.5, interpolation=cv2.INTER_LINEAR)
double_size = cv2.resize(image, None, fx=2.0, fy=2.0, interpolation=cv2.INTER_LINEAR)

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(half_size, cv2.COLOR_BGR2RGB))
plt.title("Half Size")
plt.axis("off")

plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(double_size, cv2.COLOR_BGR2RGB))
plt.title("Double Size")
plt.axis("off")
plt.show()
```

3. **Image Flipping**

```
# Horizontal, Vertical, and Diagonal Flips
flip_horizontal = cv2.flip(image, 1)
flip_vertical = cv2.flip(image, 0)
flip_diagonal = cv2.flip(image, -1)

plt.figure(figsize=(12, 6))
plt.subplot(1, 3, 1)
plt.imshow(cv2.cvtColor(flip_horizontal, cv2.COLOR_BGR2RGB))
plt.title("Horizontal Flip")
plt.axis("off")

plt.subplot(1, 3, 2)
plt.imshow(cv2.cvtColor(flip_vertical, cv2.COLOR_BGR2RGB))
plt.title("Vertical Flip")
plt.axis("off")
```

```
plt.subplot(1, 3, 3)
plt.imshow(cv2.cvtColor(flip_diagonal, cv2.COLOR_BGR2RGB))
plt.title("Diagonal Flip")
plt.axis("off")
plt.show()
```

4. **Image Rotation**

```
# Rotate by 90 degrees clockwise
rotated_90 = cv2.rotate(image, cv2.ROTATE_90_CLOCKWISE)

# Rotate by 45 degrees around center
(h, w) = image.shape[:2]
center = (w // 2, h // 2)
rotation_matrix = cv2.getRotationMatrix2D(center, 45, 1.0)
rotated_45 = cv2.warpAffine(image, rotation_matrix, (w, h))

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(rotated_90, cv2.COLOR_BGR2RGB))
plt.title("90 Degrees Rotation")
plt.axis("off")

plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(rotated_45, cv2.COLOR_BGR2RGB))
plt.title("45 Degrees Rotation")
plt.axis("off")
plt.show()
```

5. **Aspect Ratio Adjustment**

```
new_width = 300
aspect_ratio = image.shape[0] / image.shape[1]
new_height = int(new_width * aspect_ratio)
resized_image = cv2.resize(image, (new_width, new_height), interpolation=cv2.INTER_LIN

plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
plt.title("Aspect Ratio Preserved")
plt.axis("off")
plt.show()
```

6. **Arithmetic Operations**

```
brightened_image = cv2.add(image, 50)  # Increase brightness
darkened_image = cv2.subtract(image, 50)  # Decrease brightness

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
```

```
plt.imshow(cv2.cvtColor(brightened_image, cv2.COLOR_BGR2RGB))
plt.title("Brightened Image")
plt.axis("off")

plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(darkened_image, cv2.COLOR_BGR2RGB))
plt.title("Darkened Image")
plt.axis("off")
plt.show()
```

7. **Bitwise Operations**

```
binary1 = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)[1]
binary2 = cv2.threshold(cv2.flip(gray_image, 1), 127, 255, cv2.THRESH_BINARY)[1]

bitwise_and = cv2.bitwise_and(binary1, binary2)
bitwise_or = cv2.bitwise_or(binary1, binary2)
bitwise_xor = cv2.bitwise_xor(binary1, binary2)

plt.figure(figsize=(18, 6))
plt.subplot(1, 3, 1)
plt.imshow(bitwise_and, cmap="gray")
plt.title("Bitwise AND")
plt.axis("off")

plt.subplot(1, 3, 2)
plt.imshow(bitwise_or, cmap="gray")
plt.title("Bitwise OR")
plt.axis("off")

plt.subplot(1, 3, 3)
plt.imshow(bitwise_xor, cmap="gray")
plt.title("Bitwise XOR")
plt.axis("off")
plt.show()
```

Submission Instructions

1. Submit your Python code as a `.py` or Jupyter notebook (`.ipynb`).
2. Include a PDF report with:
   - Answers to theory questions.
   - Screenshots of outputs from practical tasks.
   - Explanations and observations.
3. Zip all files into a single folder named `Chapter02_Assignment_YourName.zip`.