EC7212 – Computer Vision and Image Processing

Take Home Assignment 1 - Final report

Maithreepala U.G.R.D.

EG/2020/4062

Computer Engineering

This project mainly focus on the basic image processing operations using Python and OpenCV. The goal is to perform four different tasks on a grayscale image.

Github link : https://github.com/RangaDM/EC7212-Assignment-1

Code :

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

def reduce_intensity_levels(image, num_levels):
  step = 256 // num_levels
  lookup_table = np.zeros(256, dtype=np.uint8)
  for i in range(256):
    lookup_table[i] = (i // step) * step
  return cv2.LUT(image, lookup_table)

def spatial_averaging(image, kernel_size):
  return cv2.blur(image, (kernel_size, kernel_size))

def rotate_image(image, angle):
  if angle == 90:
    return cv2.rotate(image, cv2.ROTATE_90_CLOCKWISE)

  height, width = image.shape[:2]
  center = (width // 2, height // 2)
  matrix = cv2.getRotationMatrix2D(center, angle, 1.0)

  # Compute new bounding dimensions
  cos = np.abs(matrix[0, 0])
  sin = np.abs(matrix[0, 1])
  new_width = int((height * sin) + (width * cos))
  new_height = int((height * cos) + (width * sin))
```

```
    # Adjust matrix for translation
    matrix[0, 2] += (new_width / 2) - center[0]
    matrix[1, 2] += (new_height / 2) - center[1]

    return cv2.warpAffine(image, matrix, (new_width, new_height))

def block_averaging(image, block_size):
    """
    Reduce spatial resolution by averaging non-overlapping blocks.

    Args:
        image: Grayscale image
        block_size: Size of the block (e.g., 3, 5, 7)

    Returns:
        Block-averaged image
    """
    height, width = image.shape
    result = image.copy()

    for y in range(0, height, block_size):
        for x in range(0, width, block_size):
            block = image[y:y+block_size, x:x+block_size]
            if block.size == 0:
                continue
            avg = np.mean(block, dtype=np.uint8)
            result[y:y+block_size, x:x+block_size] = avg

    return result

def main():
    image_path = input("Enter the path to your image: ")
    image = cv2.imread(image_path)
```

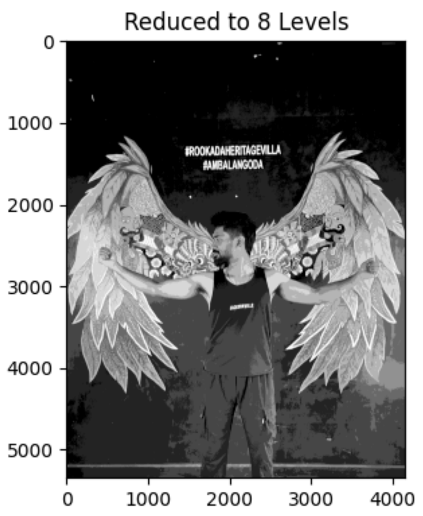
```
if image is None:
    print("Error: Unable to load the image.")
    return

gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# --- 1. Intensity Reduction ---
num_levels = int(input("Enter number of intensity levels (power of 2, e.g., 2, 4, 8...): "))
if num_levels not in [2, 4, 8, 16, 32, 64, 128, 256]:
    print("Invalid input! Must be a power of 2 between 2 and 256.")
    return
reduced_image = reduce_intensity_levels(gray_image, num_levels)

# --- 2. Spatial Averaging ---
avg_3x3 = spatial_averaging(gray_image, 3)
avg_10x10 = spatial_averaging(gray_image, 10)
avg_20x20 = spatial_averaging(gray_image, 20)

# --- 3. Rotation ---
rotated_45 = rotate_image(gray_image, 45)
rotated_90 = rotate_image(gray_image, 90)

# --- 4. Block Averaging (Spatial Downsampling) ---
block_3x3 = block_averaging(gray_image, 3)
block_5x5 = block_averaging(gray_image, 5)
block_7x7 = block_averaging(gray_image, 7)

# Display All Results
plt.figure(figsize=(16, 12))

# Row 1: Original + Intensity + Blur
plt.subplot(3, 4, 1)
plt.imshow(gray_image, cmap='gray')
plt.title('Original Grayscale')

plt.subplot(3, 4, 2)
plt.imshow(reduced_image, cmap='gray')
plt.title(f'Reduced to {num_levels} Levels')

plt.subplot(3, 4, 3)
plt.imshow(avg_3x3, cmap='gray')
plt.title('3x3 Average Blur')

plt.subplot(3, 4, 4)
plt.imshow(avg_10x10, cmap='gray')
plt.title('10x10 Average Blur')

# Row 2: Blur + Rotations
plt.subplot(3, 4, 5)
plt.imshow(avg_20x20, cmap='gray')
plt.title('20x20 Average Blur')

plt.subplot(3, 4, 6)
plt.imshow(rotated_45, cmap='gray')
plt.title('Rotated 45°')

plt.subplot(3, 4, 7)
plt.imshow(rotated_90, cmap='gray')
plt.title('Rotated 90°')

# Row 3: Block Downsampling
plt.subplot(3, 4, 9)
plt.imshow(block_3x3, cmap='gray')
plt.title('3x3 Block Averaged')

plt.subplot(3, 4, 10)
plt.imshow(block_5x5, cmap='gray')
```

```
    plt.title('5x5 Block Averaged')

    plt.subplot(3, 4, 11)
    plt.imshow(block_7x7, cmap='gray')
    plt.title('7x7 Block Averaged')

    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    main()
```

Results :

1. Reduce Intensity Levels

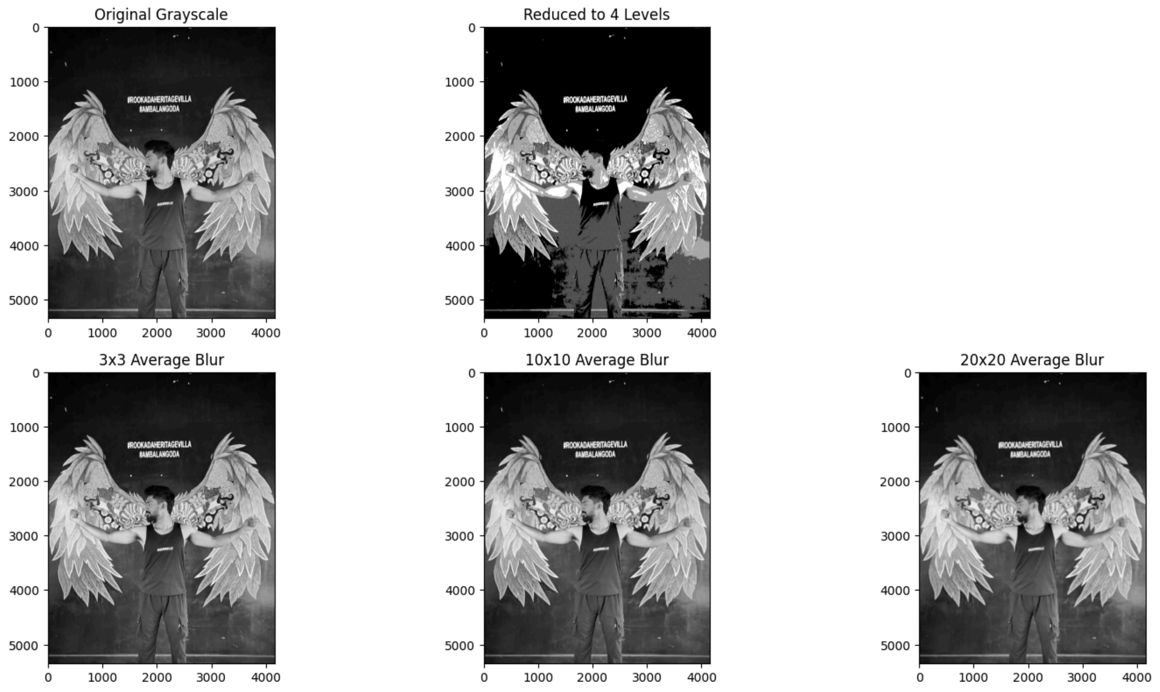
Images normally have 256 levels of brightness (from 0 to 255). For the assignment, I wrote a program to reduce the number of brightness levels to a smaller number, like 2, 4, 8, etc. This helps simulate how images look with fewer shades of gray. The user should enter how many levels they want, and the program adjusts the image accordingly.



2. **Spatial Averaging (Smoothing)**

In this part, I applied a blurring (smoothing) technique to the image using three different square regions:

- A small 3x3 region
- A medium 10x10 region
- A large 20x20 region

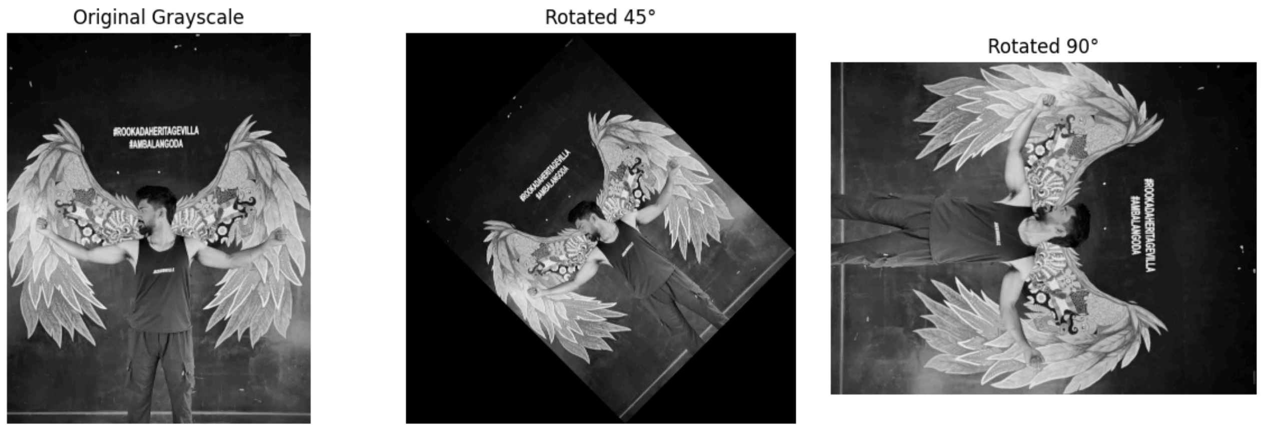


This means each pixel is replaced by the average value of its surrounding pixels. Larger regions result in a more blurred image. This technique is useful to reduce noise.

3. Rotate Image

I rotated the input image by:

45 degrees, and 90 degrees.



The 90-degree rotation uses a built-in OpenCV function. The 45-degree rotation is done using a mathematical transformation, which rotates the image around its center and resizes it to fit the new shape.

4. Block Averaging (Reduce Spatial Resolution)

In this part, I divided the image into small square blocks (3x3, 5x5, and 7x7) without overlapping, and replaced all pixels in each block with the average value of that block. This makes the image look more pixelated or low-resolution, which simulates reducing the image quality. It's like seeing a zoomed-out or simplified version of the image.