DEPARTMENT OF COMPUTER ENGINEERING

FACULTY OF ENGINEERING

UNIVERSITY OF RUHUNA

EC7212 – Computer Vision and Image Processing

Take Home Assignment 1

DISSANAYAKA D.M.S.C

EG/2020/3905

Image-Processing-Experiments-Take Home 1

GitHub Repository with source code**:**

1 Introduction

This report details the implementation of various image processing techniques using Python. The operations include reducing intensity levels, applying spatial average filters, rotating images, and performing block-wise average down sampling. These techniques are applied to analyze and manipulate a sample image, with results visualized and saved for comparison.

Figure 1-1 Original Image

2 Methodology

The image processing tasks were performed using the OpenCV library for image manipulation and Matplotlib for visualization. The methodology involved:

- Loading the image in grayscale and color formats.
- Implementing functions for intensity level reduction, spatial averaging, image rotation, and block-wise averaging.
- Processing the image with varying parameters like intensity levels, kernel sizes, block sizes.
- Saving the processed images in designated output folders for analysis.

2.1 Code

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

def intensity_level_reduction(image, levels):
    factor = 256 // levels
    reduced = (image // factor) * factor
    return reduced

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

def rotate_image_by_angle(image, angle):
    h, w = image.shape[:2]
    matrix = cv2.getRotationMatrix2D((w / 2, h / 2), angle, 1.0)
    rotated = cv2.warpAffine(image, matrix, (w, h))
    return rotated

def blockwise_average_downsample(image, block_size):
    h, w = image.shape
    downsampled = image.copy()
    for y in range(0, h - block_size + 1, block_size):
        for x in range(0, w - block_size + 1, block_size):
            block = image[y:y + block_size, x:x + block_size]
            avg = int(np.mean(block))
            downsampled[y:y + block_size, x:x + block_size] = avg
    return downsampled

def plot_and_save(title, image, filename, cmap='gray'):
    plt.figure(figsize=(4, 4))
    plt.imshow(image, cmap=cmap)
```

```
    plt.title(title)
    plt.axis('off')
    plt.tight_layout()
    plt.savefig(filename, bbox_inches='tight')
    plt.close()

def main():
    image_path = "image.jpg"
    gray_img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    color_img = cv2.imread(image_path)

    os.makedirs("outputs", exist_ok=True)
    os.makedirs("outputs/intensity outputs", exist_ok=True)
    os.makedirs("outputs/average filters outputs", exist_ok=True)
    os.makedirs("outputs/rotated outputs", exist_ok=True)
    os.makedirs("outputs/blockwise outputs", exist_ok=True)

    #  Intensity level reduction
    for level in [2, 4, 8, 16, 32, 64, 128, 256]:
        reduced_img = intensity_level_reduction(gray_img, level)
        filename = f"outputs/intensity outputs/intensity_{level}_levels.png"
        plot_and_save(f"Intensity {level} Levels", reduced_img, filename)

    #  Spatial average filtering
    for k in [3, 10, 20]:
        filtered_img = spatial_average_filter(gray_img, k)
        filename = f"outputs/average filters
outputs/average_filter_{k}x{k}.png"
        plot_and_save(f"{k}x{k} Average Filter", filtered_img, filename)

    #  Rotate image by 45 and 90 degrees
    rotated_45 = rotate_image_by_angle(color_img, 45)
    rotated_90 = cv2.rotate(color_img, cv2.ROTATE_90_CLOCKWISE)
    plot_and_save("Rotated 45°", cv2.cvtColor(rotated_45, cv2.COLOR_BGR2RGB),
"outputs/rotated outputs/rotated_45.png", cmap=None)
    plot_and_save("Rotated 90°", cv2.cvtColor(rotated_90, cv2.COLOR_BGR2RGB),
"outputs/rotated outputs/rotated_90.png", cmap=None)

    #  Block-wise average downsampling
    for b in [3, 5, 7]:
        downsampled_img = blockwise_average_downsample(gray_img, b)
        filename = f"outputs/blockwise outputs/blockwise_avg_{b}x{b}.png"
        plot_and_save(f"Blockwise Avg {b}x{b}", downsampled_img, filename)

    print(" All images processed and saved in the 'outputs/' folder.")

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

2.2 Results

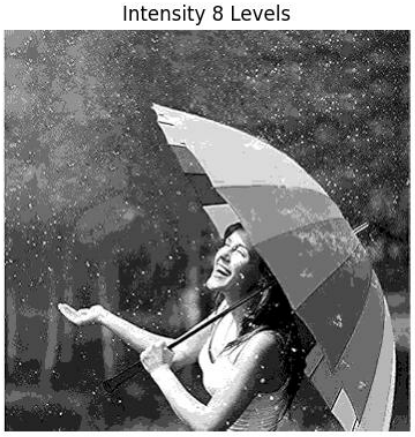
Intensity Levels

The image was processed with intensity levels of 2, 4, 8, 16, 32, 64, 128, and 256. The reduction is most noticeable at lower levels creating a highly quantized appearance.

Intensity 2 Levels

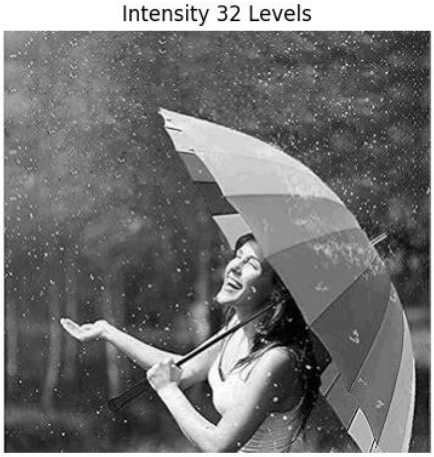
Intensity 4 Levels

Intensity 8 Levels



Intensity 16 Levels

Intensity 32 Levels



Intensity 64 Levels

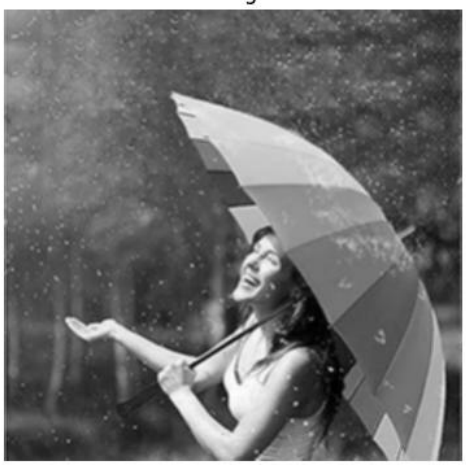
Intensity 128 Levels

Intensity 256 Levels

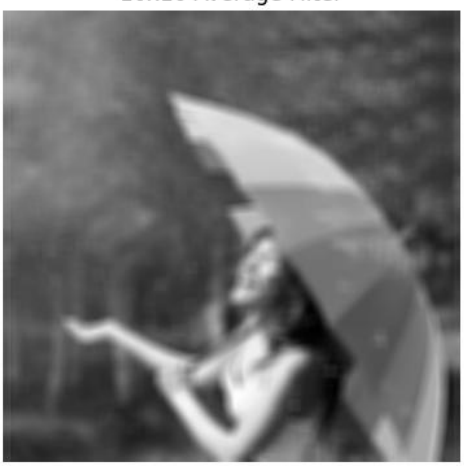
Spatial Average Filters

Applied with 3x3, 10x10, and 20x20 kernels. Larger kernels (e.g., 20x20) resulted in significant blurring, smoothing out details

3x3 Average Filter



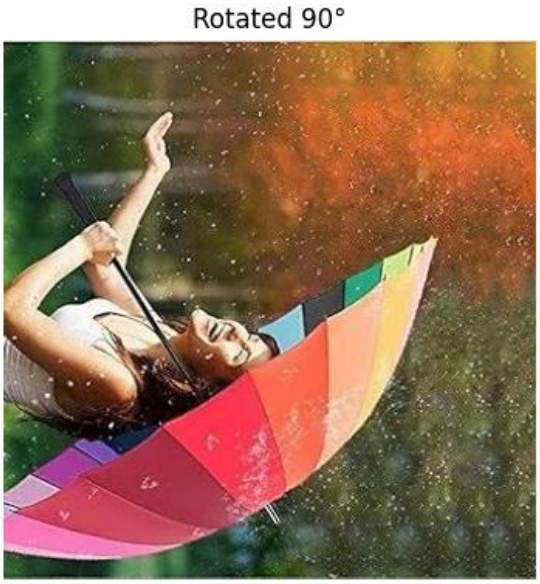
10x10 Average Filter



20x20 Average Filter

Rotations

The image was rotated by 45° and 90°, preserving the color information and showing clear orientation changes.



Block-wise Average Downsampling

Performed with 3x3, 5x5, and 7x7 blocks. Larger blocks led to a more pronounced reduction in spatial resolution, with a blocky appearance.

Blockwise Avg 5x5

Blockwise Avg 7x7