DEPARTMENT OF COMPUTER ENGINEERING

FACULTY OF ENGINEERING

UNIVERSITY OF RUHUNA

EC7212 – Computer Vision and Image Processing

Take Home Assignment 1

DISSANAYAKE D.M.M.I.T

EG/2020/3912

Image-Processing-Experiments-Take Home 1

GitHub Repository with source code: https://github.com/ThisaruDissanayake/Take-Home1Image-Processing-Experiments.git

1 Introduction

This report documents the implementation of four image processing operations using Python, OpenCV, NumPy, and Matplotlib. The experiments were conducted on a sample image (test4.jpg) featuring a parrot, and the results are saved as processed images.

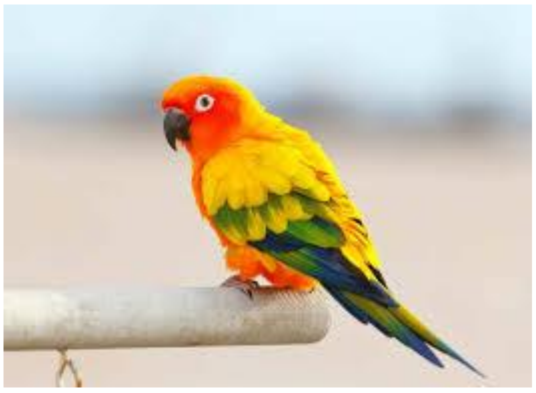
Figure 1-1 Original Image

2 Methodology

The following tasks were implemented using the provided Python script:

2.1 Reduce the Number of Intensity Levels

- Objective: Reduce the image's intensity levels from 256 to integer powers of 2 (2, 4, 8, 16, 32, 64, 128, 256).
- Method: The pixel values are scaled by dividing by a factor (256 / levels) and multiplying back to the nearest level.

2.2 Spatial Averaging

- Objective: Apply a simple spatial average filter with 3x3, 10x10, and 20x20 neighborhoods.
- Method: A normalized kernel of ones is used with cv2.filter2D to compute the average.

2.3 Image Rotation

- Objective: Rotate the image by 45 and 90 degrees.

- Method: A rotation matrix is generated using cv2.getRotationMatrix2D and applied with cv2.warpAffine.

2.4 Reduce Spatial Resolution

- Objective: Simulate reduced spatial resolution by averaging non-overlapping 3×3, 5×5, and 7×7 blocks.
- Method: Each block is replaced with its mean value using NumPy operations.

3 Code Implementation

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

# Function to reduce intensity levels
def reduce_intensity_levels(image, levels):

    if not (levels & (levels - 1) == 0 and levels != 0):
        raise ValueError("Number of intensity levels must be a power of 2.")


    if len(image.shape) == 3:
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    factor = 256 // levels
    reduced_image = (image // factor) * factor
    return reduced_image

# Function to perform spatial averaging
def spatial_average(image, kernel_size):

    if len(image.shape) == 3:
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)


    kernel = np.ones((kernel_size, kernel_size), dtype=np.float32) /
(kernel_size * kernel_size)
    averaged_image = cv2.filter2D(image, -1, kernel)
    return averaged_image

# Function to rotate image
def rotate_image(image, angle):


    (h, w) = image.shape[:2]
```

```python
    center = (w // 2, h // 2)

    M = cv2.getRotationMatrix2D(center, angle, 1.0)
    rotated_image = cv2.warpAffine(image, M, (w, h))
    return rotated_image

# Function to reduce spatial resolution by block averaging
def reduce_spatial_resolution(image, block_size):

    if len(image.shape) == 3:
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    h, w = image.shape
    new_h, new_w = h // block_size, w // block_size
    reduced_image = np.zeros((new_h * block_size, new_w * block_size),
dtype=np.uint8)

    # Process each block
    for i in range(0, h - block_size + 1, block_size):
        for j in range(0, w - block_size + 1, block_size):
            block = image[i:i+block_size, j:j+block_size]
            avg = np.mean(block).astype(np.uint8)
            reduced_image[i:i+block_size, j:j+block_size] = avg

    return reduced_image

def main():
    # Load an image
    image = cv2.imread('D:/7 Semester/computer vision/Take
Home1/code/test4.jpg')
    if image is None:
        print("Error: Could not load the image.")
        return


    intensity_levels = [2, 4, 8,16,32,64,128,256]
    for levels in intensity_levels:
        reduced_intensity = reduce_intensity_levels(image, levels)
        cv2.imwrite(f'reduced_intensity_{levels}.jpg', reduced_intensity)
        plt.imshow(reduced_intensity, cmap='gray')
        plt.title(f'Intensity Levels: {levels}')
        plt.show()


    kernel_sizes = [3, 10, 20]
    for size in kernel_sizes:
        averaged_image = spatial_average(image, size)
        cv2.imwrite(f'averaged_{size}x{size}.jpg', averaged_image)
        plt.imshow(averaged_image, cmap='gray')
        plt.title(f'Spatial Average: {size}x{size}')
        plt.show()


    angles = [45, 90]
    for angle in angles:
        rotated_image = rotate_image(image, angle)
        cv2.imwrite(f'rotated_{angle}_degrees.jpg', rotated_image)
        plt.imshow(cv2.cvtColor(rotated_image, cv2.COLOR_BGR2RGB))
        plt.title(f'Rotation: {angle} degrees')
        plt.show()


    block_sizes = [3, 5, 7]
    for size in block_sizes:
        reduced_resolution = reduce_spatial_resolution(image, size)
        cv2.imwrite(f'reduced_resolution_{size}x{size}.jpg',
reduced_resolution)
        plt.imshow(reduced_resolution, cmap='gray')
        plt.title(f'Reduced Resolution: {size}x{size} blocks')
        plt.show()

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

4 Results

4.1 Intensity Level Reduction:

- Images: reduced_intensity_2.jpg to reduced_intensity_256.jpg.
- Observation: The image transitions from a binary-like appearance (2 levels) to the full grayscale original (256 levels), with increasing gray shades as levels increase.

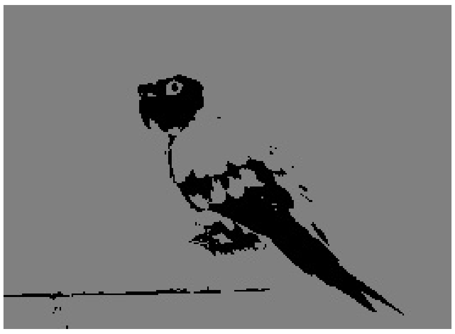
Figure 4-1 Reduced intensity level 2 image

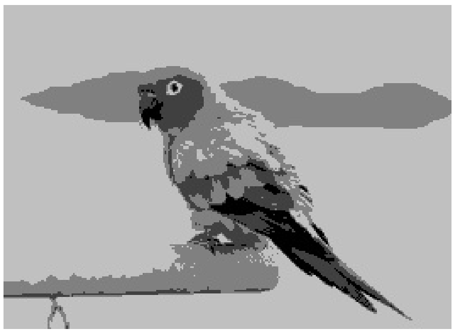
Figure 4-2 Reduced intensity level 4 image

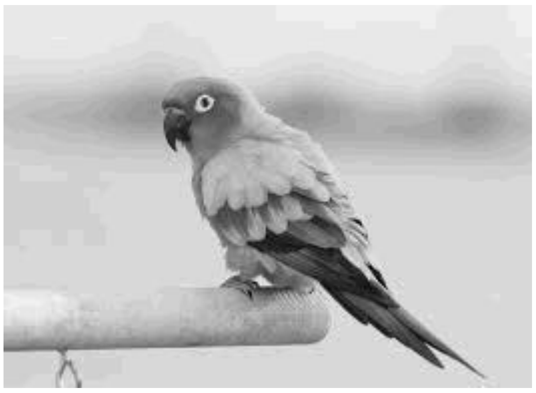
Figure 4-3 Reduced intensity level 32 image

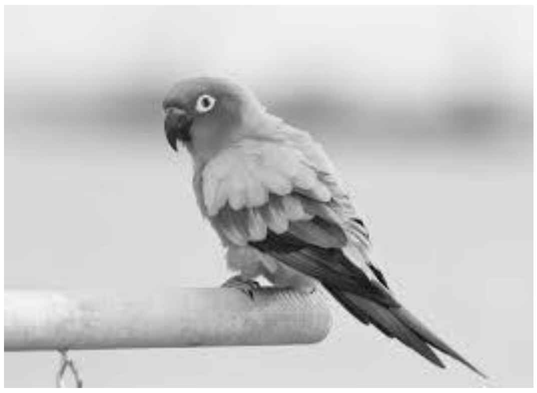
Figure 4-4 Reduced intensity level 256 image

4.2 Spatial Averaging**:**

- Images: averaged_3x3.jpg, averaged_10x10.jpg, averaged_20x20.jpg.
- Observation: The parrot’s image becomes progressively smoother, with 3x3 showing slight blur, 10x10 losing feather details, and 20x20 heavily blurred.

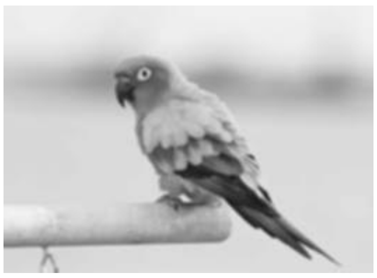
Figure 4-5 Averaged_3x3 Image

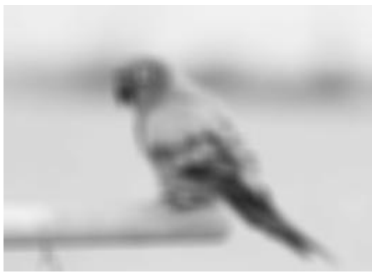
Figure 4-6 Averaged_10x10 Image

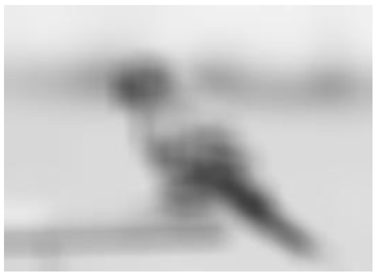
Figure 4-7 Averaged_20x20 Image

4.3 Image Rotation**:**

- Images: rotated_45_degrees.jpg, rotated_90_degrees.jpg.
- Observation: The parrot is rotated by 45 and 90 degrees, though some edge cropping occurs due to fixed output size.

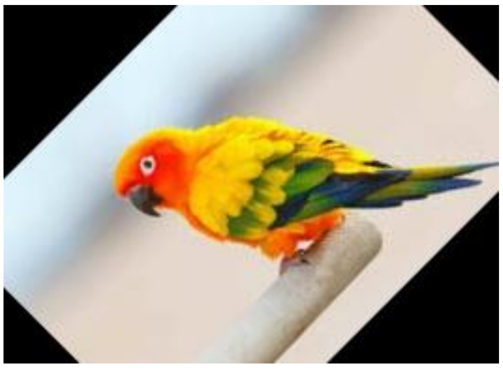
Figure 4-8 Rotated_45_degree image

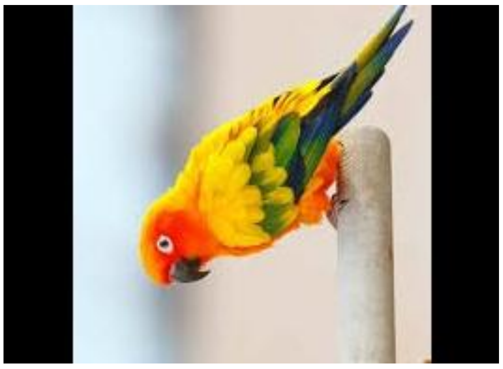
Figure 4-9 Rotated_90_degree image

4.4 Spatial Resolution Reduction**:**

- Images: reduced_resolution_3x3.jpg, reduced_resolution_5x5.jpg, reduced_resolution_7x7.jpg.
- Observation: The image becomes increasingly blocky and less detailed, with 3x3 retaining the parrot’s shape, 5x5 showing more blur, and 7x7 reducing it to major outlines.

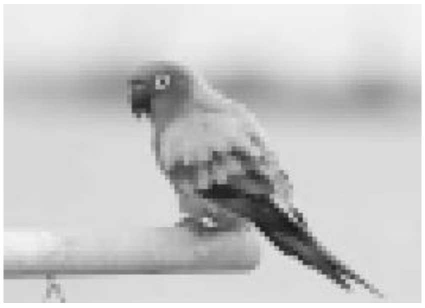
Figure 4-10 Resolution_3x3 image

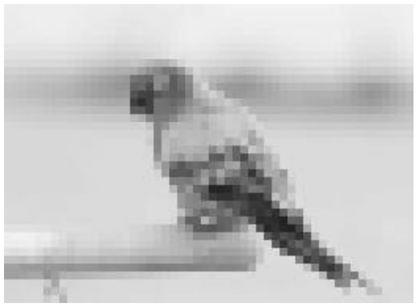
Figure 4-11 Resolution_5x5 image

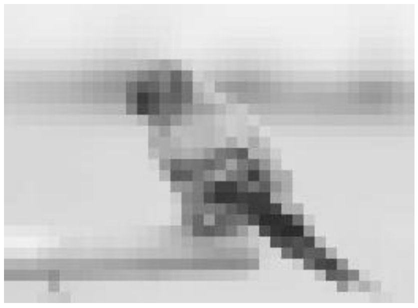
Figure 4-12 Resolution_7x7 image