EC7212 – Computer Vision and Image Processing
Take Home Assignment 1

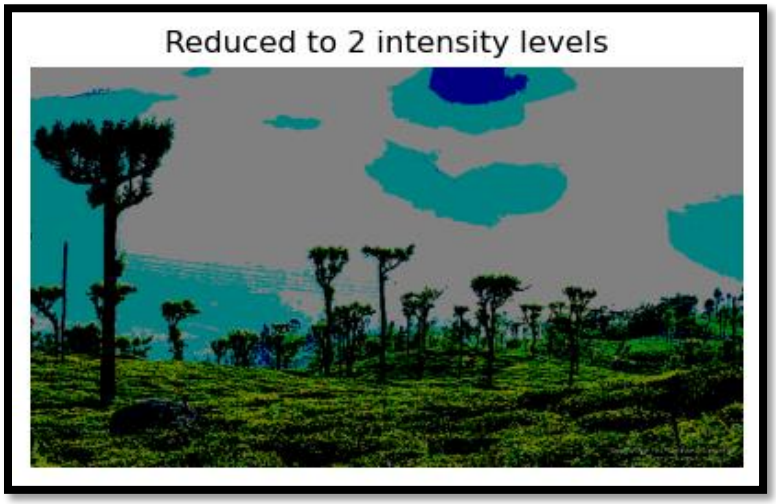
Index No: EG/202/3886
Name: De Zoysa R.N.C.
Github Repo: https://github.com/NisalDeZoysa/EC7212-CVIP-Assignment-01.git

1. To reduce the number of intensity levels in an image from 256 to 2, in integer powers of 2. The desired number of intensity levels needs to be a variable input to your program.

```
def reduce_intensity_levels(image, levels):
    factor = 256 // levels
    reduced_image = (image // factor) * factor
    return reduced_image

for level in [2, 4, 8, 16, 32, 64, 128]:
    reduced = reduce_intensity_levels(image, level)
    show_image(f"Reduced to {level} intensity levels", reduced)
```

Results:



Reduced to 4 intensity levels

Reduced to 8 intensity levels

Reduced to 16 intensity levels

Reduced to 32 intensity levels

Reduced to 64 intensity levels

Reduced to 128 intensity levels

2. Load an image and then perform a simple spatial 3x3 average of image pixels. Repeat the process for a 10x10 neighborhood and again for a 20x20 neighborhood.

```
3.
4. def spatial_average(image, kernel_size):
5.     return cv2.blur(image, (kernel_size, kernel_size))
6.
7. for size in [3, 10, 20]:
8.     blurred = spatial_average(image, size)
9.     show_image(f"Spatial Average {size}x{size}", blurred)
10.
```

Results:

Spatial Average 10x10

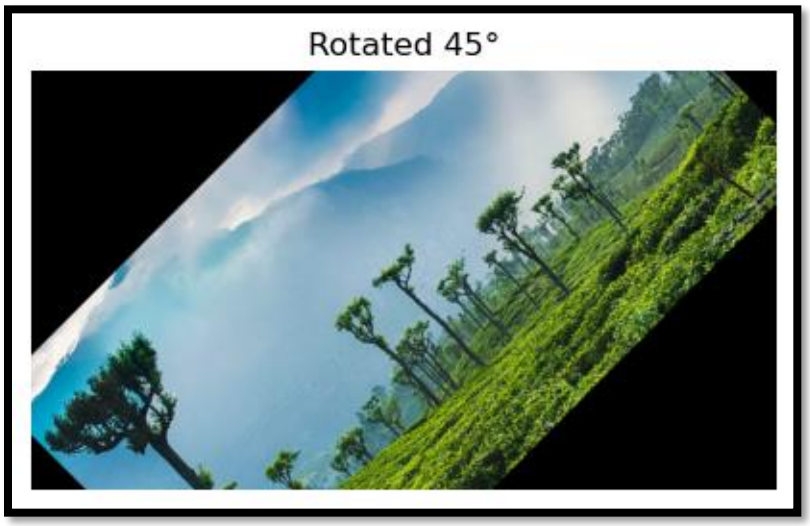
Spatial Average 20x20

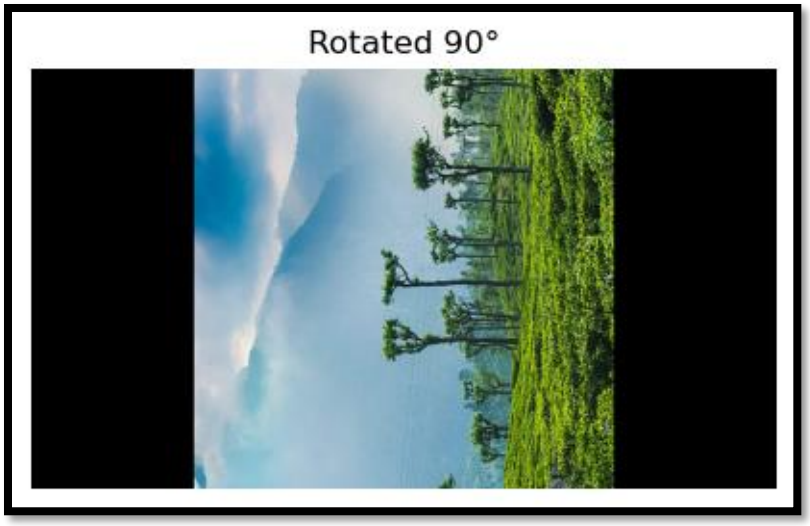
3. Rotate an image by 45 and 90 degrees.

```
# Rotation
def rotate_image(image, angle):
    (h, w) = image.shape[:2]
    center = (w // 2, h // 2)
    matrix = cv2.getRotationMatrix2D(center, angle, 1.0)
    return cv2.warpAffine(image, matrix, (w, h))

rotated_45 = rotate_image(image, 45)
rotated_90 = rotate_image(image, 90)
show_image("Rotated 45°", rotated_45)
show_image("Rotated 90°", rotated_90)
```

Results:



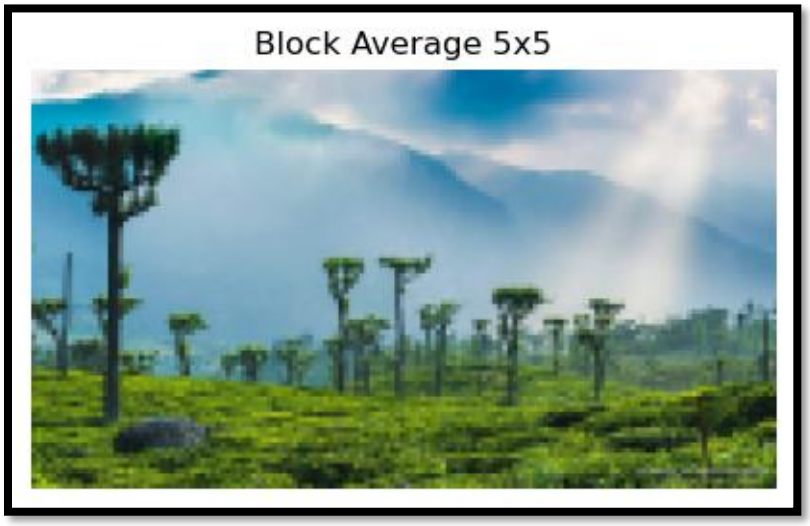


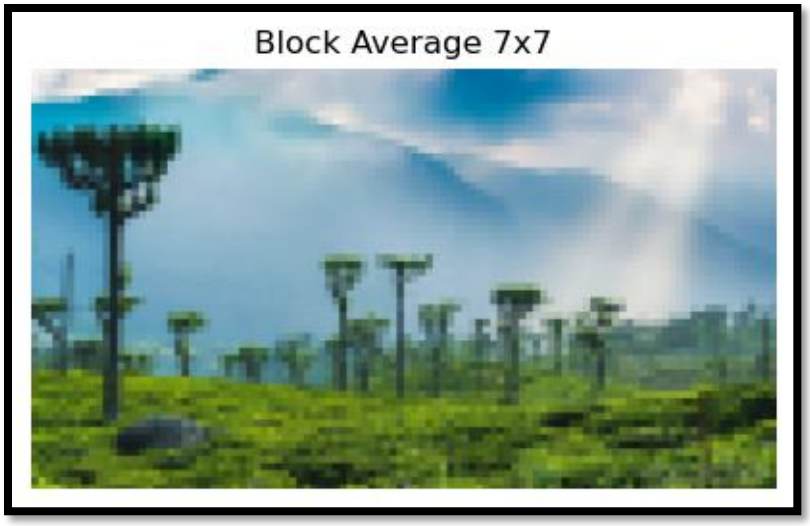
4. For every 3×3 block of the image (without overlapping), replace all the corresponding 9 pixels by their average. This operation simulates reducing the image spatial resolution. Repeat this for 5×5 blocks and 7×7 blocks.

```
# Block Averaging
def block_average(image, block_size):
    h, w = image.shape[:2]
    result = image.copy()
    for y in range(0, h, block_size):
        for x in range(0, w, block_size):
            block = image[y:y+block_size, x:x+block_size]
            if block.size == 0:
                continue
            avg_color = block.mean(axis=(0, 1), dtype=int)
            result[y:y+block_size, x:x+block_size] = avg_color
    return result

for block in [3, 5, 7]:
    reduced = block_average(image, block)
    show_image(f"Block Average {block}x{block}", reduced)
```

Results:




Block Average 7x7