EC7212: COMPUTER VISION AND IMAGE PROCESSING

ASSIGNMENT 01

NAME : JATHUSAN U.

REG No. : EG / 2020 / 3977

SEMESTER: 07

DATE : 21 / 06 / 2025

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.

Figure 1: Original Image

Code :

question_1.py > ...

```python
import cv2
import numpy as np
img = cv2.imread("image/image1.jpg", cv2.IMREAD_GRAYSCALE)
if img is None:
    raise ValueError("Image not found.")
cv2.namedWindow("Original Image", cv2.WINDOW_NORMAL)
cv2.resizeWindow("Original Image", 600, 600)
cv2.imshow("Original Image", img)
max_level = 8
initial_level = 3

def update_intensity(level):
    if level < 0 or level > max_level:
        raise ValueError(f"Level must be between 0 and {max_level}.")

    num_levels = 2 ** level
    step = 256 // num_levels

    reduced_image = np.floor(img / step) * step
    reduced_image = reduced_image.astype(np.uint8)

    cv2.imshow("Reduced Intensity Image at Level " + str(level) + "",

cv2.createTrackbar("Intensity Level", "Original Image", initial_level,
update_intensity(initial_level)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Result：

Figure 2: Grayscale Image

Result :

Figure 3: Quantized Grayscale Images

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.

Code :

```python

import cv2
import numpy as np
img = cv2.imread("image/image1.jpg")
if img is None:
    raise ValueError("Image not found or unable to load.")

window_name = ["Original Image", "3x3 Average Image", "10x10 Average Image", "20x20 Average Image"]

for name in window_name:
    cv2.namedWindow(name, cv2.WINDOW_NORMAL)
    cv2.resizeWindow(name, 600, 600)

average_3x3 = cv2.blur(img, (3, 3))
average_10x10 = cv2.blur(img, (10, 10))
average_20x20 = cv2.blur(img, (20, 20))

cv2.imshow(window_name[0], img)
cv2.imshow(window_name[1], average_3x3)
cv2.imshow(window_name[2], average_10x10)
cv2.imshow(window_name[3], average_20x20)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Result :

Figure 4: Output of Simple spatial averaging

3. Rotate an image by 45 and 90 degrees.

Code :

```
def rotate_image(image, angle):
    (h, w) = image.shape[:2]
    center = (w / 2, h / 2)

    M = cv2.getRotationMatrix2D(center, angle, 1.0)

    cos = np.abs(M[0, 0])
    sin = np.abs(M[0, 1])
    new_width = int((h * sin) + (w * cos))
    new_height = int((h * cos) + (w * sin))

    M[0, 2] += (new_width / 2) - center[0]
    M[1, 2] += (new_height / 2) - center[1]
    return cv2.warpAffine(image, M, (new_width, new_height))
rotated_image = rotate_image(img, 45)
rotated_image_90 = rotate_image(img, 90)

window_name = ["Original Image", "Rotated Image 45 Degrees", "Rotated Image 90 Degrees"]

for name in window_name:
    cv2.namedWindow(name, cv2.WINDOW_NORMAL)
    cv2.resizeWindow(name, 600, 600)
cv2.imshow(window_name[0], img)
cv2.imshow(window_name[1], rotated_image)
cv2.imshow(window_name[2], rotated_image_90)
```

Result :

Figure 5: Output of Image rotation

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.

Code :

```
def reduce_resolution(image, block_size):
    processed_image = np.copy(image)

    for i in range(0, image.shape[0], block_size):
        for j in range(0, image.shape[1], block_size):
            roi = image[i:i + block_size, j:j + block_size]

            if roi.shape[0] != block_size or roi.shape[1] != block_size:
                continue

            avg_color = np.mean(roi, axis=(0, 1), dtype=np.float32)

            processed_image[i:i + block_size, j:j + block_size] = avg_color
    return processed_image.astype(np.uint8)

processed_images = [reduce_resolution(img, block_size) for block_size in block_sizes]

cv2.namedWindow("Original Image", cv2.WINDOW_NORMAL)
cv2.resizeWindow("Original Image", 600, 600)
cv2.imshow("Original Image", img)

for i, processed_image in enumerate(processed_images):
    window_name = f"Processed Image (Block Size {block_sizes[i]})"
    cv2.namedWindow(window_name, cv2.WINDOW_NORMAL)
    cv2.resizeWindow(window_name, 600, 600)
    cv2.imshow(window_name, processed_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Result :

Figure 6: Output of Averaging for every 3×3, 5x5, 7x7 blocks

GitHub Link

https://github.com/Jathusan19/ComputerVison_1