DEPARTMENT OF ELECTRICAL AND INFORMATION ENGINEERING

FACULTY OF ENGINEERING

UNIVERSITY OF RUHUNA

EE7204 – COMPUTER VISION AND IMAGE PROCESSING

14th March 2024

SADEEPA P.M.A.S. (EG/2019/3726)

01. 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.

```python
def reduceIntensityLevels(image, levels):
    max_val = 255
    factor = max_val / (levels - 1)
    return np.round(image / factor) * factor

# Loading image
image = cv2.imread('./img.jpg')

# Code to reduce intensity levels
while True:
    levels = int(input("Enter the number of intensity levels [2, 4, 8, 16, ...]: "))
    if levels > 0 and math.log2(levels).is_integer():
        break
    else:
        print("Please enter valid number for intensity levels that is an integer power of 2.")

outputIntensityLevels = reduceIntensityLevels(cv2.cvtColor(image, cv2.COLOR_BGR2GRAY), levels)

# Display Original Image
cv2.imshow( winname: 'Original Image', image)
# Display Intensity Levels Image
cv2.imshow( winname: 'Reduced Intensity Levels Image', reduceIntensityLevels)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Figure 1: Code for Question 1

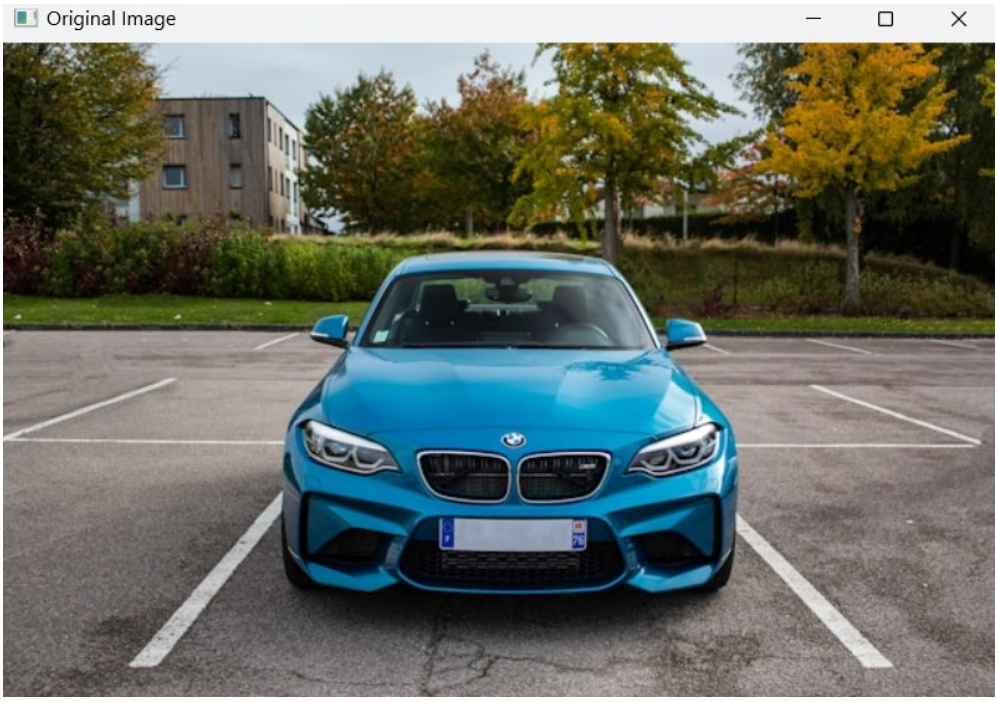

Figure 2: Original Image

Figure 3: Reduced Intensity Levels Image

02. 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.

```
4  def applyAverageFilter(img, kernel_size):
5      return cv2.blur(img, ksize: (kernel_size, kernel_size))
6
7  # Loading image
8  image = cv2.imread('img.jpg')
9
10 # Apply average filter with different kernel sizes
11 output_image_3x3 = applyAverageFilter(image, kernel_size: 3)
12 output_image_10x10 = applyAverageFilter(image, kernel_size: 10)
13 output_image_20x20 = applyAverageFilter(image, kernel_size: 20)
14
15 # Display Original Image
16 cv2.imshow( winname: 'Original Image', image)
```

```
18  # Average Filter
19  cv2.imshow( winname: '3x3 Average Filter', output_image_3x3)
20  cv2.imshow( winname: '10x10 Average Filter', output_image_10x10)
21  cv2.imshow( winname: '20x20 Average Filter', output_image_20x20)
22
23  cv2.waitKey(0)
24  cv2.destroyAllWindows()
```

Figure 4: Code for Question 2

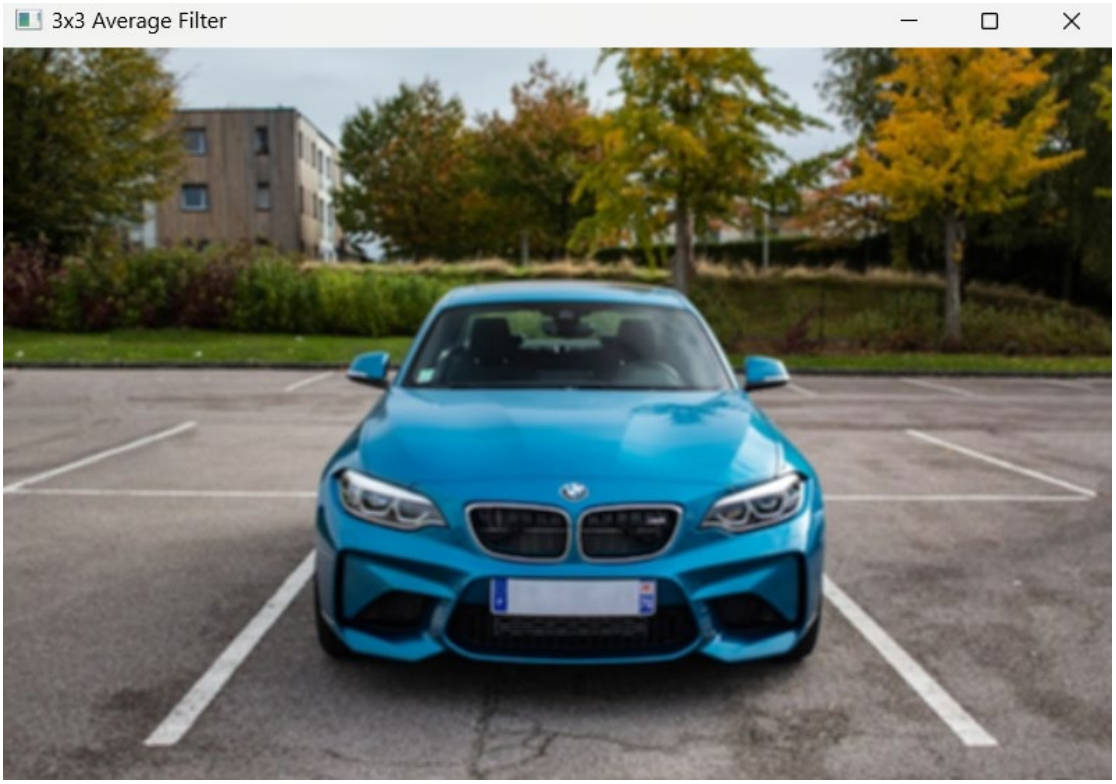

Figure 5: Output Image Of 3x3 avg filter

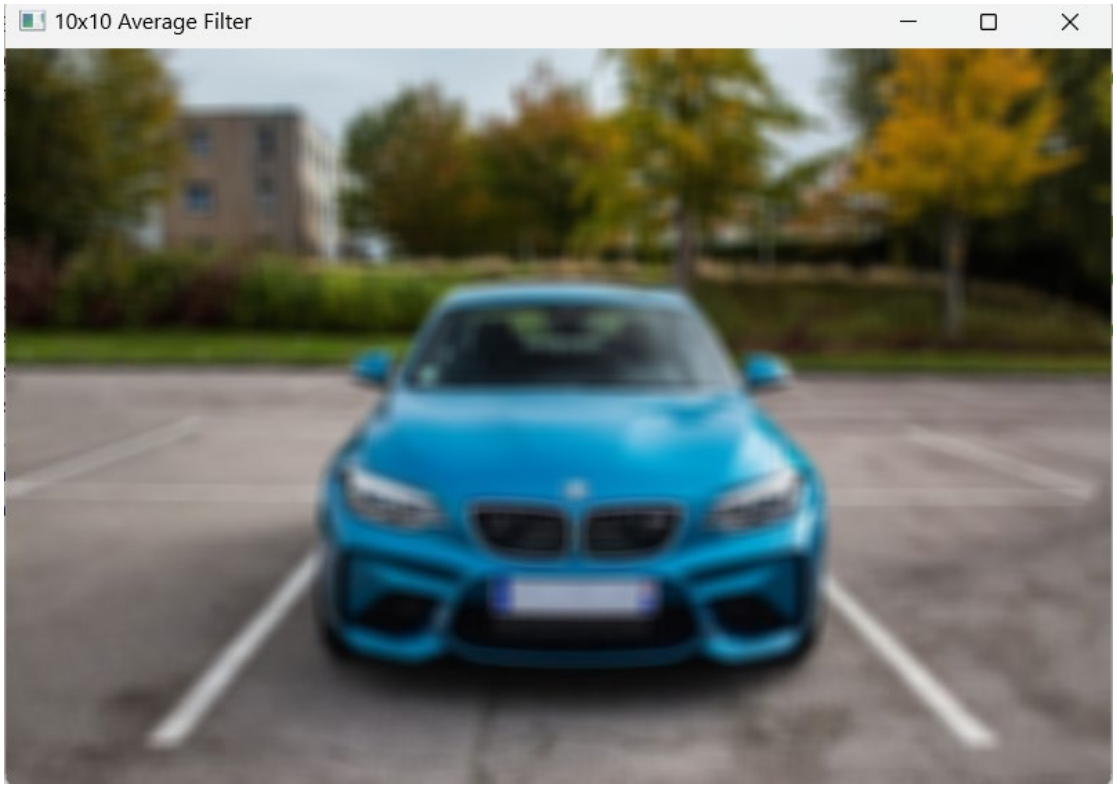

Figure 6: Output Image Of 10x10 avg filter

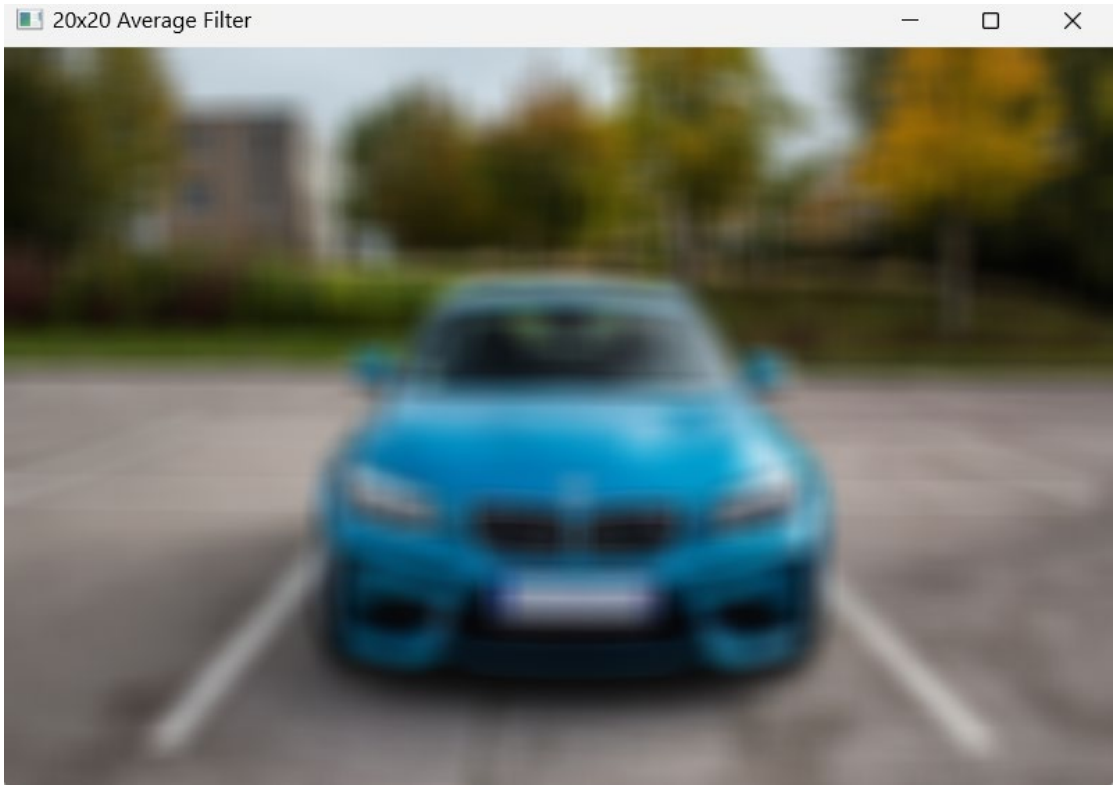

Figure 7: Output Image Of 20x20 avg filter

03. Rotate an image by 45 and 90 degrees.

```
def rotateImage(img, angle):
    rows, cols = img.shape[:2]
    M = cv2.getRotationMatrix2D( center: (cols / 2, rows / 2), angle, scale: 1)
    return cv2.warpAffine(img, M, dsize: (cols, rows))

# Loading image
image = cv2.imread('img.jpg')

# Rotate image by 45 & 90 degrees
rotated_45 = rotateImage(image, angle: 45)
rotated_90 = rotateImage(image, angle: 90)

# Display the results
cv2.imshow( winname: 'Original Image', image)
```

```
# Display Original Image
cv2.imshow( winname: 'Original Image', image)

# Display Rotated Image
cv2.imshow( winname: 'Rotated by 45 Degrees', rotated_45)
cv2.imshow( winname: 'Rotated by 90 Degrees', rotated_90)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Figure 8: Code for Question 3

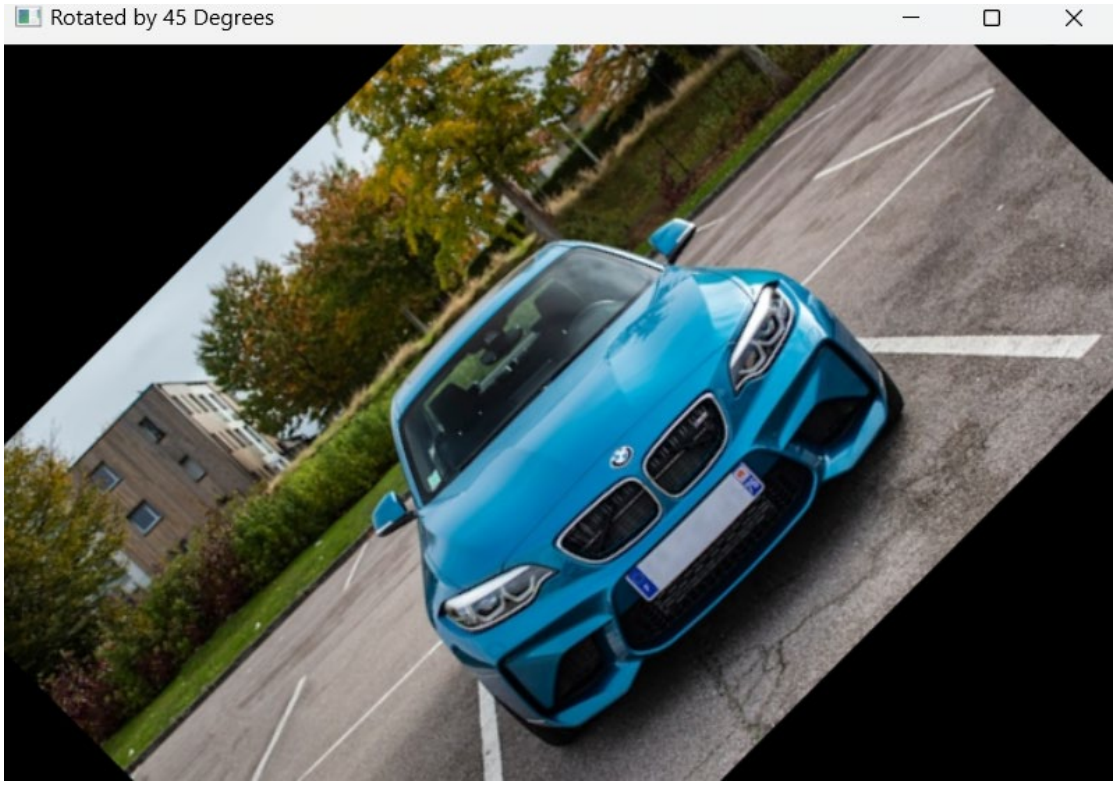

Figure 9: Image rotated by 45 Degrees

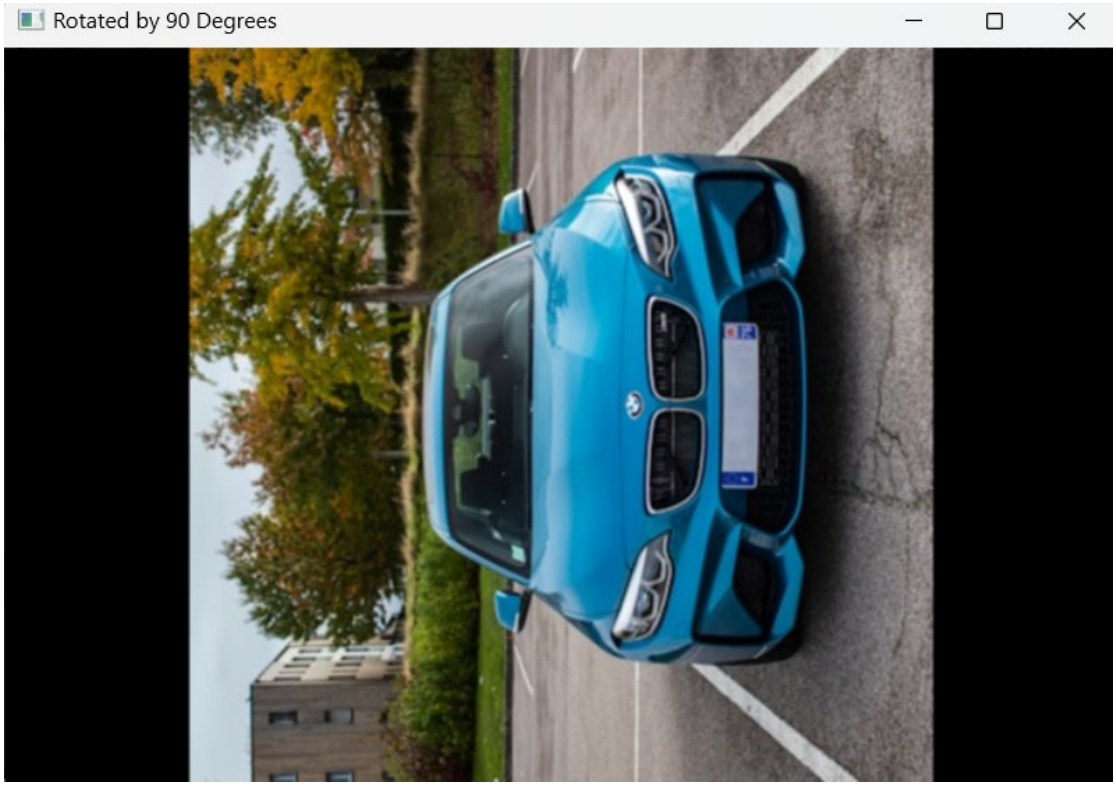

Figure 10: Image rotated by 90 Degrees

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

```python
def reduce_resolution(img, block_size):
    rows, cols = img.shape[:2]
    for row in range(0, rows, block_size):
        for col in range(0, cols, block_size):
            block = img[row:row+block_size, col:col+block_size]
            avg = np.mean(block)
            img[row:row+block_size, col:col+block_size] = avg
    return img

# Loading image
image = cv2.imread('img.jpg')

# Reduce resolution with different block sizes.
output_image_3x3_block = reduce_resolution(cv2.cvtColor(image, cv2.COLOR_BGR2GRAY).copy(), block_size: 3)
output_image_5x5_block = reduce_resolution(cv2.cvtColor(image, cv2.COLOR_BGR2GRAY).copy(), block_size: 5)
output_image_7x7_block = reduce_resolution(cv2.cvtColor(image, cv2.COLOR_BGR2GRAY).copy(), block_size: 7)

# Display Original Image
cv2.imshow( winname: 'Original Image', image)

# Display Block Avg Filter
cv2.imshow( winname: '3x3 Block Average', output_image_3x3_block)
cv2.imshow( winname: '5x5 Block Average', output_image_5x5_block)
cv2.imshow( winname: '7x7 Block Average', output_image_7x7_block)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Figure 11: Code for Question 4

Figure 12: Output image 3x3

Figure 13: Output image 5x5

Figure 14: Output image 7x7