**TCV 3151 – Computer Vision**

**Lab 2**

**Section 1: Theory**

The following questions will serve as a guide in confirming your understanding in the image processing field. Answer all of the questions below.

1. Name the three fields that deal with images.
2. Name the three basic types of signals. Give the formulas for each of the signal.
3. Compare and contrast sampling and quantization.
4. Calculate the number of bits required to store the following images:
5.  image with 8 gray levels
6. An  image with 256 gray levels

**Section 2: Practical**

The steps below provide an introduction on how to handle images in OpenCV. Follow the instructions to start working with images in the OpenCV environment.

*Problem 1: Down sampling an image*

1. Read in the image ‘logo.png’ and convert it to grayscale. Display the image to see how it looks like. What is the size of this image?
2. Subsample the image by half of the original image size. Display the image to see whether you get the desired output. You can use the following program as a guide.

(height, width) = I.shape

reduction\_step = 2

rows\_keep = range(0, height, reduction\_step)

cols\_keep = range(0, width, reduction\_step)

I2 = np.zeros((len(rows\_keep), len(cols\_keep)), dtype=np.uint8)

for i, row\_value in enumerate(rows\_keep):

for j, col\_value in enumerate(cols\_keep):

# Write your code here

*Note*: This is what you get when you print i and row\_value, i.e. print(i, row\_value)

0 0

1 2

2 4

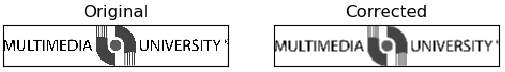
3 6

4 8

1. Now, subsample the image by quarter of the original image size. What do you observe?

*Problem 2: Correcting the Aliasing Effect*

Resizing the image size is a desirable step in image processing due to limitation in processing resources. However, resizing an image will sometimes cause the aliasing effect as shown on the left figure below. What can you do to overcome the problem? In other words, what can you do to minimize the aliasing effect as shown in the right figure?



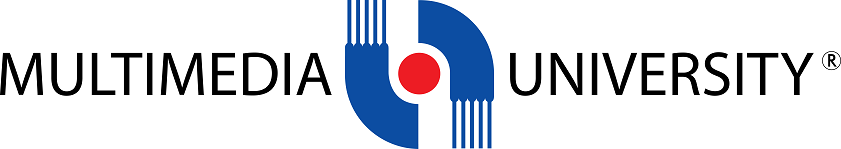
*Problem 3: Working with Bitwise Operation*

1. Read in the images ‘foreground.png’ and ‘background.png’.
2. Apply the following bitwise operations on the images:

|  |  |  |
| --- | --- | --- |
| **No.** | **Operation** | **Function** |
| 1. | AND | cv2.bitwise\_and |
| 2. | OR | cv2.bitwise\_or |
| 3. | XOR | cv2.bitwise\_xor |
| 4. | AND | cv2.bitwise\_not |

*Problem 4: Putting Logo on images*

Read in the images ‘mmu.jpg’ and ‘logo.png’. Use the bitwise operations in Problem 3 to add the logo on the image as follows:



**LOGO**



**IMAGE**



**LOGO ON IMAGE**