

DMET 501 – Introduction to Media Engineering

### Project

(Due on December 30<sup>th</sup>, 2022 at 11:59PM)

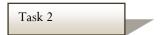


In this task you are required to apply quantization on the image using 9 Levels.

Function Signature : def show\_image\_information(image):

Input: The image you read using Image.open('/content/image.jpg')

Expected Output: return the unique colors before and after quantization each as a list.



In this task you are required to **compute the consecutive runs** for each row of your **quantized image**. This is not the Run-Length Encode, meaning that if a color appeared more than once not continuously in the same row, it will have **separated/different runs**.

The format of a single run should be a tuple that looks as follows:

(row, first coumn of the run, last column of the run, color)

Example:

```
[(0,0,0,2),(0,1,3,1),(0,4,5,2),(0,6,7,1),(1,0,5,2),(1,6,7,3)]
```

Function Signature : def compute\_runs(image, unique\_values):

Input: image: the quantized image,  $unique\_values:$  a list of colors after quantization

Expected Output: a list of tuples each being a single run using the format mentioned above and similar to the example.



DMET 501 – Introduction to Media Engineering

# Project

(Due on December 30<sup>th</sup>, 2022 at 11:59PM)

Task 3

In this task you are required to **compute the Run-Length Encode** for the **quantized image**. You may use your output/function implemented in **Task2**. **However**, **you don't have to.** If a color appeared more than once not continously in the same row, it will have only 1 tuple representing all the runs in the row.

### Example:

```
[(0,1,3,6,7,1), (0,0,0,4,5,2), (1,0,5,2), (1,6,7,3)]
```

Function Signature : def compute\_RLE(image, unique\_values) :

Input: image: the quantized image, unique\_values: a list of colors after quantization

Expected Output: a list of tuples each being a single run using the following format.

(row,{first coumn of the run, last column of the run}\*, color)

HELPER FUNCTIONS

These functions are to help you write your code, you are not obliged to use them.

```
def get_size(image):
```

Get the dimensions for the image (width, height).

Example Output: (61, 61).



DMET 501 - Introduction to Media Engineering

# Project

(Due on December 30<sup>th</sup>, 2022 at 11:59PM)

```
def get pixel value(img,col,row):
```

Get the intensity of a single pixel.

Example Output *color at pixel* (col=2, row=4) in img: 150.

```
def quantization(image, n):
```

Quantize the image colors to n levels.

Output: Image after quantization as a list.

```
def get unique values(image):
```

Get the unique values of colors in an image.

Ouput: pair of (list of unique colors in image, length of the list of unique colors in image).

```
def extract row color(arr,row,color):
```

Takes as an input a list of tuples having the format discussed in Task2 and extracts/filters the tuples having the same row row and same color color and return them in a list.

#### Input Example:

```
arr=[(0,0,0,2),(0,1,3,1),(0,4,5,2),(0,6,7,1),(1,0,5,2),(1,6,7,3)]
extract_row_color(arr, 0, 1)
Output Example:
[(0, 1, 3, 1), (0, 6, 7, 1)]
```

```
def merge_row_color(filtered, row, color):
```

Takes as input a list of tuples having the same row & same color, returns a merged tuple for the start/ finish of the run.

#### Input Example:

```
filtered=[(0, 1, 3, 1), (0, 6, 7, 1)]
merge_row_color(filtered, 0, 1)
Output Example:
  (0, 1, 3, 6, 7, 1)
```