NOVA SCHOOL OF SCIENCE & TECHNOLOGY

Advanced Topics in Digital Image Processing

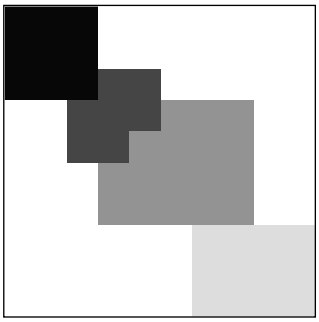
LESSON 1 - *Lossless Image Compression*

Objective:

With this assignment it is intended to test the coding and decoding algorithms of Huffman and Shannon-Fano applied in the lossless image compression.

Procedure:

1. Create a python script file to open and show the image (aula1.bmp)
2. Add huffman.py to the project and import it to the script



Encoding

Considering the image above (10x10 pixels)

1. Calculate and display the image histogram using a plot (use matplotlib) and on console (as shown on the right)

```
hist = cv.calcHist([img],[0],None,[256],[0,256])

plt.plot(hist), plt.xlim([0, 256])
plt.show()
```

```
-> 7 - 9
-> 69 - 7
-> 147 - 17
-> 221 - 12
-> 255 - 55
```

2. How would you encode this image without compression? How many bits would be required to represent the image (without code-book)?
3. What is the result of Huffman compression applied to this example?
    1. Build the Huffman tree (on paper or excel) by coding intensity levels according to the rule: 1-left and 0-right.
    2. Confirm the generated codes with the auxiliary function Huffman

```
huffman_dict = h.huffman(img.flatten())
print(huffman_dict)
```

    3. Generate the code-book of this encoding (on paper or excel).
    4. Write a script to encode the provided image using the generated huffman dictionary and calculate the length of the generated code. (print only the bits string)

4. What compression ratio is obtained compared to a simple representation?
5. Repeat the same operation with Shannon-Fano encoding (on paper or excel)

Decoding (on paper)

6. Consider that an image with 6x6 pixels generated the encoding below using Huffman Optimized code and containing the optimized code book:

0000 0101 0000 0010 1010 1011 1110 1000 0001 0110 1000 0001 0110 1000 0001 0110 1000 0001 0111 0110 0000 0001 1101 1011 1111 1110 1

1. What is huffman tree?
2. What is the original image?

Entropy Analysis

7. Implement a function that receives an image and returns the entropy of each of the RGB channels.

```
def entropy_image(img):
        …

print(entropy_image(img))
```

8. Calculate the entropy image after applying a blurring operation.