

## CSL461: Digital Image Analysis

### Lab 4: Image Compression

---

#### Aim:

- To get hands-on experience implementing simple image compression algorithms

#### Let's get started!

- Create a directory structure to hold your work for this course and all the subsequent labs:
  - Suggestion: `CSL461/Lab4`

#### Run-Length Encoding (RLE)

- Is one of the algorithms discussed in class that attempts to remove spatial redundancy
- Given an input image, the algorithm should output:
  - `X Y I1 R1 I2 R2 I3 R3 ...`
  - Where X & Y are the dimensions of the input image and the pairs `(In Rn)` indicate that Intensity `(In)` is repeated `Rn` times
- Example:
  - Given one row image/signal: `[10 10 11 11 11 11 15 16 16 16]`
  - Output of RLE: `[ 1 10 10 2 11 4 15 1 16 3]`
- Task:
  - **Write your own** Run-Length Encoding and Decoding functions:
    - `[outCode] = myRLE( inImg )`
    - `[outImg] = myRLD( inCode )`
  - Where `inImg` is the input image, `outCode` is the RLE format
  - `inCode` is the RLE format and `outImg` is the code transformed to image format

#### Huffman Coding

- Is one of the algorithms discussed in class that attempts to remove coding redundancy
- Details of the algorithm have been discussed in class (Lecture slides L15, L16)
- Task:
  - **Write your own** Huffman Encoding and Decoding functions:
    - `[out] = myHuffmanEncode( in )`
    - `[out] = myHuffmanDecode( in )`

#### Exercise (`TestCompression.m`):

- Three sample images are provided as part of this lab: `Fig1.tiff`, `Fig2.tiff`, `Fig3.tiff`
1. Compress these images using RLE
    - a. Save the files in an appropriate format with names: `Fig1_rle`, `Fig2_rle`, `Fig3_rle`

2. Compress these images using Huffman Coding
  - a. Save the files in an appropriate format with names: Fig1\_hc, Fig2\_hc, Fig3\_hc
3. Compress the RLE encoded format using Huffman Coding
  - o Input: Fig1\_rle, Fig2\_rle, Fig3\_rle
  - o Output: Fig1\_rle\_hc, Fig2\_rle\_hc, Fig3\_rle\_hc
- o Calculate the compression factors and redundancies (using the formula from class) for each of the three compression methods in steps 1 – 3 and for each of the images. Document them in the [README](#) file
- o Comment on the efficiency of the above algorithms with respect to the content of these images in the [README](#) file
4. Decompress the \_rle images using RLD
  - a. Save the files in an appropriate format with names: Fig1\_rld, Fig2\_rld, Fig3\_rld
5. Decompress these \_hc using Huffman decoding
  - a. Save the files in an appropriate format with names: Fig1\_hd, Fig2\_hd, Fig3\_hd
6. Decompress the \_rle\_hc files using Huffman decoding followed by RLD
  - o Input: Fig1\_rle\_hc, Fig2\_rle\_hc, Fig3\_rle\_hc
  - o Output: Fig11, Fig22, Fig33
- o Calculate the difference between original and decoded versions and show that both RLE and Huffman coding are LOSSLESS compression algorithms
  - a. For each of the decompression steps 4 – 6, display original, decompressed and their difference (scaled appropriate)
  - b. Use one figure for each step (4 – 6) with results for each image show in one figure using subplot

### Submitting your work:

- o All source files and class files as one tar-gzipped archive.
  - When unzipped, it should create a directory with your ID. Example: **P2008CS1001–L1** (NO OTHER FORMAT IS ACCEPTABLE!!! Case sensitive!!!)
  - **Negative marks if the TA has to manually change this to run his/her scripts!!**
- o Source / class files should include the following: (Case-Sensitive file names!!)
  - `myHuffmanEncoding.m` / `myHuffmanDecoding.m`
  - `myRLE.m` / `myRLD.m`
  - `TestCompression.m`
  - [README](#)
- o **Negative marks for any problems/errors in running your programs**
- o Submit/Upload it to Moodle