Image Compression

Dr. P. Peter
Mathematical Image Analysis Group

Summer term 2019
Saarland University

Programming Assignment 2

You can download the files from the web page

`http://www.mia.uni-saarland.de/Teaching/ic19.shtml`

To unpack the archive, use the command `tar xzvf ic19-ex02.tar.gz`.

Exercise P2 (16 P)
Supplement the routines `encode_adaptive_wnc` and `decode_adaptive_wnc` in the C programme `ic19_adaptive_arithmetic.c`
with missing code such that it In order to compile your programme please use the contained makefile: In the folder `ic19-ex02`, simply type `make` (on a Unix system). To test the compiled programme, execute

```
./ic19_adaptive_arithmetic -i input/mini.pgm -o minidecoded
```

The newly created image `minidecoded.pgm` should be identical to the input image. In order to assist you with debugging, the file `mini_reference.txt` contains the debugging logfile of the reference implementation. This allows you were your programme might fail. If your implementation works on `mini.pgm`, try some more realistic images: `trui` and `kodim21`.

Submission

Submission: Submit via mail to peter@mia.uni-saarland.de until **Monday, 27.05.2019, 12.15**. Please submit in groups of up to 3 students. You have to submit your completed *source code* together with a *text file* containing the names and matriculation numbers of all group members. Include all of these files in an archive named `ex02_firstname_lastname.tar.gz` (zip is also okay). You can use the first and last name of any of your group members.

Hints

- Calling the programme without parameters will display a list of optional parameters.

- For debugging, use the option `-D debugfile.txt`. The programme will then write some default debug-information into the specific textfile. **Be careful: These textfiles become very large if you compress actual images with high resolutions. They are most useful with simple examples like `mini.pgm`** Feel free to add your own debug information to your code by following the example

  ```
  if (debug_file != 0) {
     fprintf(debug_file,"n:%ld, s:%ld, r:%f, M:%ld \n",n,s,r,M);
  }
  ```

  where `%ld` lets you print integers and `%f` lets you print floats.

- In C, you might have to do some type casting. Consider these two examples with integers (long) and float values (double):

  ```
  long a;
  double b;
  a = (long)b; b = (double)a;
  ```

- You might need the functions `ceil(x)`, `floor(y)`, and `pow(x,y)`. The first two are self-explanatory and the latter computes x^y.