Programming Assignment P4

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-ex04.tar.gz`.

Exercise P4 (24 P)

This Exercise implements JPEG from Lecture 11 with the same overall pipeline, but a few changes. It uses the arithmetic coding from P2 instead of Huffman coding, applies the modified integer YUV transformation of JPEG2000 instead of the standard YCbCr transform and supports different quantisation and chroma subsampling techniques.

P4.1: Chroma Subsampling

Supplement the routines `subsample` and `upsample` in the C programme `ic19_jpeg_light.c` with missing code. These are used to down- or upscale the chroma channels Cb and Cr. The routines should perforam a simple arithmetic averaging over all pixels that are merged into one for subsampling. For upsampling, simply copy the coarse resolution value into all corresponding fine resolution pixels.

P4.2: DCT

Supplement the routine `block_DCT` in the C programme `ic19_jpeg_light.c` with missing code. It implements the DCT from Lecture 11 on $N \times N$ blocks.

P4.3: Reordering

Supplement the routine `init_zigzag_table` n the C programme `ic19_jpeg_light.c` with missing code. It provides a lookup table for the zig-zag pattern of the reordering step.

P4.4: Entropy Coding

Supplement the routine `block_encode` n the C programme `ic19_jpeg_light.c` with missing code. You only have to fill in small code snippets that assign categories and category indices to DC coefficients and implement the ZRL exception with runlengths larger than 15. The actual entropy coding and storage to the hard disk is already implemented.

P4.5: Testing Parameters

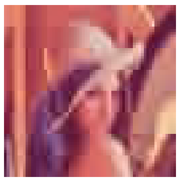
Compilation and Execution

In order to compile your programme please use the contained makefile: In the folder `ic19-ex04`, simply type `make` (on a Unix system).

To test the compiled programme, execute

```
./ic19_jpeg_light -i input/lena_mini.pgm -o lena_mini_jpeg
```

The newly created images show the DCT coefficients and the reconstruction obtained from standard parameters. You should obtain an error of ≈ 147.5 at a compression ratio of $\approx 27:1$ with the following reconstruction:



Try to modify the parameters q and s by appending the options -q or -s to the programme call (both should be integers ≥ 1). Use the image kodim23.ppm for your tests. What happens for $q = 1$ and $s = 1$? What happens for large q and s?

Submission

Submission: Submit via mail to peter@mia.uni-saarland.de until **Monday, 01.07.2019, 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. Also include the makefile and the answers to all questions of P4, as well as the images you produced. Include all of these files in an archive named `ex04_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** `lena_mini.ppm` 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 function `pow(x,y)` which computes x^y.