PA171 - Project documentation

Michal Jankovič

December 5, 2021

This document describes a (lossy and lossless) compression scheme designed and implemented as a project for PA171 Digital Image Filtering.

1 Compression scheme

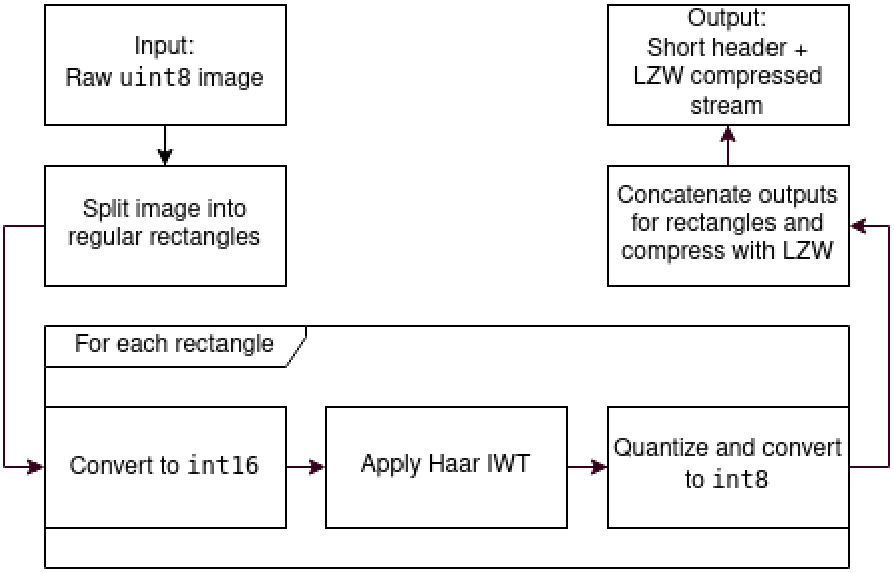
Input:
Raw uint8 image
Split image into regular rectangles
For each rectangle
Convert to int16
Apply Haar IWT
Quantize and convert to int8
Concatenate outputs for rectangles and compress with LZW
Output:
Short header + LZW compressed stream

Figure 1: Compression scheme

The lossy compression scheme is based around the Haar integer wavelet transform. Input `uint8` image with values in the $[0, 255]$ range is split into regular rectangular chunks (chunks touching the right and bottom border have different sizes). These are then converted to `int16` and passed through the wavelet transform recursively until only one approximation coefficient remains.

Resulting wavelet coefficients are transformed to fit inside the single-byte `int8` range of $[-128, 127]$, and to enable better compression ratios. The 2D wavelet

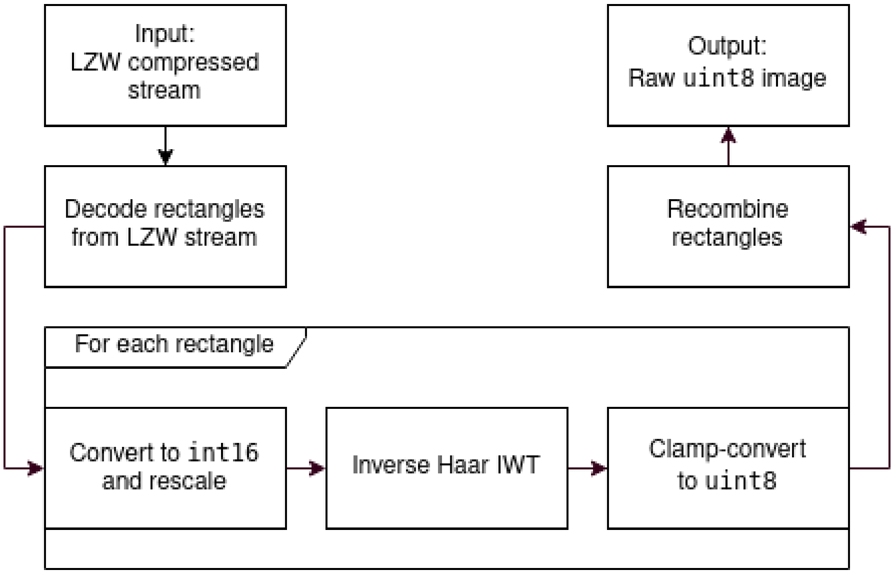

Figure 2: Decompression scheme

transform (whether it is applied just once or recursively) yields 3 kinds of coefficients that must be handled differently:

- Approximation coefficients - obtained by averaging neighboring pixels and thus in the same range as source values - $[0, 255]$. They are not quantized as they contain the most important information. Only transformation applied is a shift by -128 to fit in the `int8` range.
- Vertical / horizontal detail coefficients - obtained by differentiating neighboring pixels, these have double the range of source values - $[-256, 255]$. To fit the `int8` range, they must be quantized by dividing by at least a factor of 2. A larger factor is chosen for the detail components obtained by the first few iterations of the wavelet transform.
- Diagonal detail coefficients - obtained by twice differentiating neighboring pixels, these have four times the range of source values - $[-512, 511]$. They are quantized by twice the quantization factor for the vertical / horizontal coefficients.

Outputs from transform and quantization of each rectangle are concatenated and compressed with LZW, using all possible 8-bit values as the initial dictionary. The LZW dictionary grows up-to a maximum word size of 16 bits. When the maximum word size is reached, the dictionary is flushed.

A short header (88 bytes) is attached to the compressed LZW stream, indicating image dimensions and all the compression parameters to allow configuring the decoder for decompression.

2 CLI application

The compression / decompression is implemented as a CLI application.

Compressing an image with default lossy compression:

```
pa171_compress image.bmp image.small
```

Compressing an image with lossless compression:

```
pa171_compress -l0 image.bmp image.small
```

Compressing an image with lossy compression level 32:

```
pa171_compress -l32 image.bmp image.small
```

Compression levels range from 0 to 64. 0 means that the input is only passed through LZW compression without transform and quantization (lossless compression). Otherwise, a quantization factor of 2x the compression level is used for the horizontal / vertical details of the first level of wavelet decomposition and 4x for diagonal details. This factor is divided by 8 for the detail components of the second level, and so on until a minimum quantization factor of 2 for vertical / horizontal details and 4 for diagonal.

Decompressing an image (with any compression level):

```
pa171_decompress image.small image.bmp
```

2.1 Compilation

Requirements:

- `gcc >= 10.2.0`
- `CMake >= 3.17`
- `conan >= 1.43.0` (`pip install conan`)

```
mkdir build && cd build
conan install .. -b missing
cmake .. -DCMAKE_TOOLCHAIN_FILE=conan_toolchain.cmake
cmake --build .
```

Resulting binaries are in `build/bin`.