Chapter 2

The Motion-JPEG decoder application

The Motion-JPEG video format is composed of a succession of JPEG still pictures. It is used by several digital cameras and camcorders to store video clips of a relatively small size. With Motion-JPEG, each frame of video is captured separately and compressed using the JPEG algorithm. JPEG stands for Joint Photographic Experts Group, a standardization committee that gave its name to the still picture compression algorithm it defined. It is a lossy compression algorithm, meaning that the decompressed image is not totally identical to the original image. Its goal is to reduce the size of natural color images as much as possible without affecting the quality of the image as experienced by the human eyes.

2.1 Algorithm overview

The JPEG compression algorithm splits an image in blocks of 8×8 pixels, then translates each block into the frequency domain, eliminates the high using a per image filter, and compress the resulting blocks using an Huffman encoding with a per image dictionnary. The blocks are also called macroblock or MCU for MacroBlock Unit. The resulting bitstream is made of sequences of raw data separated by markers that identify data. The format of the sections is given in section A.

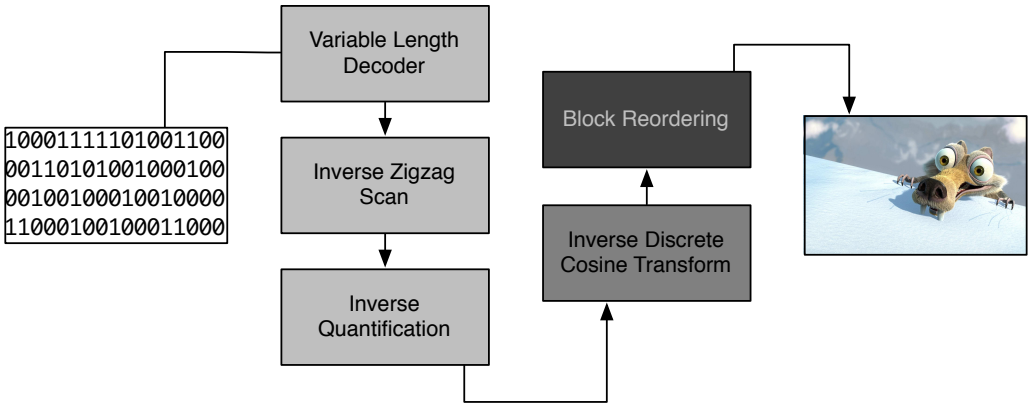

Figure 2.1: The JPEG decoder

The Motion-JPEG algorithm can run sequentially (figure 2.1) or be decomposed in a parallel version in order to take advantage of the parallelism of its target platform. A possible parallelized version can be straightforwadly build from the sequential version. It is presented figure 2.2.

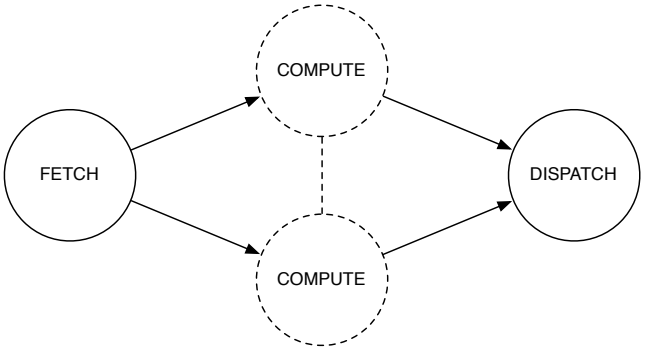

Figure 2.2: A parallel version of the Motion-JPEG algorithm

2.2 The FETCH task

This task performs a rough analysis of the input stream in order to extract the JPEG markers and identify the corresponding data. For each image, it executes the following operations:

- **Variable Length Decoding**: using the Huffman tables from the DHT section, and the image size from the SOF section
- **Inverse Quantization and Inverse Zigzag Scan**: using the inverse quantization table from the DQT Section), some of the SOF Section, and some of the SOS section

2.2.1 Variable length decoding

The variable length decoding operation is based on Huffman codes. Huffman codes are called *prefixed* code, because no symbol can be the prefix of another symbol. Figure 2.3 illustrates this.

a	11
b	0
c	101
d	1001
e	1000

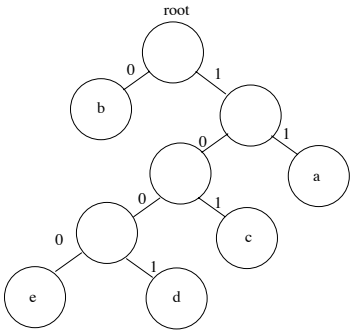

Figure 2.3: Decoding the bitstream `0001101011000` leads to `bbbabce`

2.2.2 Inverse Quantization and Inverse Zigzag Scan

The inverse quantization operation multiplies each value in a macroblock by a constant which value depends on the position of the value in the block Each image of the flow has its own constant table in order to optimize the compression. The inverse zigzag scan reorders the data of a macroblock as depicted figure 2.4.

2.3 The COMPUTE task

This task performs Inverse Discrete Cosine Transform operation (IDCT). It is implemented as a doubly nested sum, thanks to the Loëffler's algorithm, that uses the a minimal number of multiplications to accomplish this computation.

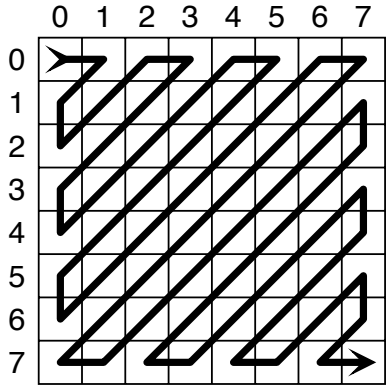

Figure 2.4: The inverse zigzag scan operation

2.4 The DISPATCH task

This operation reorganise the macroblocks comming from the different compute processes into a picture. Once the picture is finalized, it is sent to the framebuffer.