JPEG_HW Report

R10922124 林家毅

Overview

本次作業是實作 JPEG Decode 流程的其中三個步驟：

- `read_block()`: 利用 Huffman Decoding 讀取 8*8 的 block
- `anti_q()`: 執行 anti-quantization
- `anti_zz()`: 執行 anti-zigzag

以下會分別說明各函數的實作細節

Functions

read_block()

已知每個 8*8 block 都包含 1 個 DC 係數和 63 個 AC 係數，且因為 JPEG Encoding 是將 DC、AC 係數分別先做 DPCM 和 Run Length Encoding、再做 Huffman Encoding，所以在 decode 的時候就需要把順序倒過來。

其中，DC 係數 decoding 已經由助教完成，所以在這邊只需要做 AC 係數 decoding，在 Huffman Decoding 的時候，是呼叫 `get_symbol_len()` 到 AC Huffman Tree 利用 Variable Length Code 的性質探索直到 leaf，而 leaf 存的就是 8-bit Run Length Code；在 Run Length Code 中，高位元 4 bits 代表 AC 係數為 0 的個數，低位元 4 bits 的值代表要再從 input 取多少 bits 來作為非零 AC 係數，其中需要注意的是，若找到的非零 AC 係數最高位元是 0，則需要取它的 1's complement 才是真正的 AC 係數；另外也需注意 8-bit Run Length Code 中，0x00 是保留給 End of Block，代表剩下的所有 AC 係數都是 0，而 0xF0 則是代表接下來有 16 個連續 0 的 AC 係數。依照上述 decode 規則，找到 63 個 AC 係數，即完成此步驟。

anti_q()

取得一個 block 的 DC、AC 係數後，下一步要做的是 anti-quantization，由於 encode 時做的 quantization 是把 DC、AC 係數除以 quantization table 對應位置的值取整

數，因此在做 anti-quantization 的時候，只需將 block 中每一個值乘以 quantization table 對應值即可。

anti_zz()

由於在 encode 時會對 block 中的每個值從低頻區到高頻區做 zigzag scan，因此 anti-zigzag 的目的是將 DC、AC 係數放回它們在 DCT matrix 的對應位置；只需掃過 block 一次，將係數放到正確位置即可。

Reference

1. Understanding and Decoding a JPEG Image using Python