VSP Final Project / Image Coding Contest

李豪韋 103061527
陳俐安 101061117

Overview

The project is focused on images coding, especially lossy-coding, which is widely used in recent years. This document mainly consists of several sections: 1) implementation, which describes the ways we used in images coding in detail; 2) results, which demonstrates images after coding (real cases) and the evaluation plots (BD curve); 3) discussion, which contains something worth mentioning but is not required in the specification of project reports.

The source code is already publicly available on GitHub://HW-Lee/ImageCodec (hyperlink).

Implementation

- Coding Process

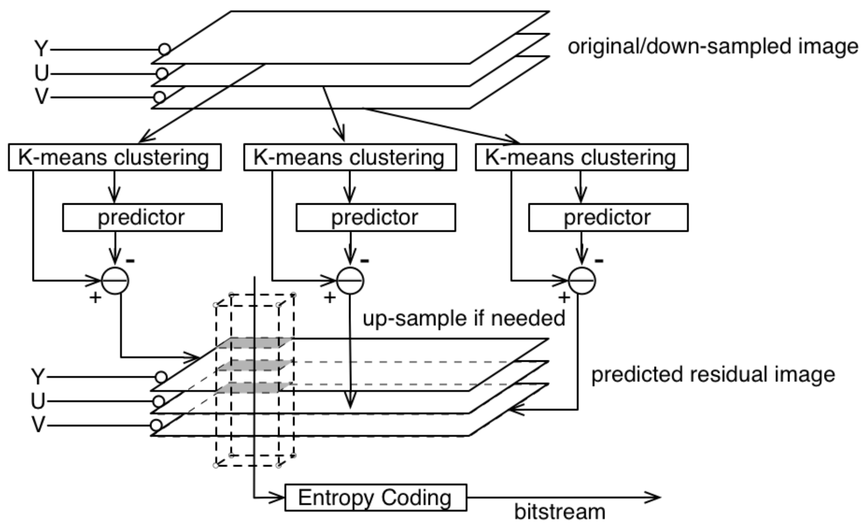

Figure 1: level 1 coding, which uses 1) k-means clustering, to cluster YUV value for decreasing variance of values; 2) predictors, to intra-predict the image and coding residual to decreasing entropy; 3) up-sampler, to interpolate value in U/V such that Y/U/V contain the same width and height. It is main encoder of the system.

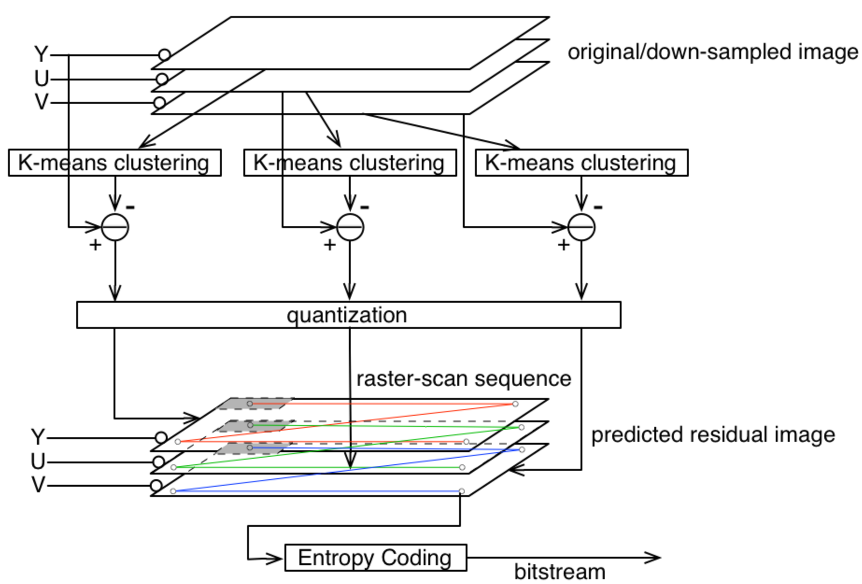

Figure 2: level 2 coding (optional), which uses same components of level 1. It is optionally used for compensating PSNR caused by quantization, namely k-means clustering, and encodes the rest of information abandoned by level 1. Notice that this level still has a quantization process, brings more parametric flexibility to find a much better coding parameters.

- Coding Format Specification

TAG	BITS	DESCRIPTION
$\log_2(\text{dsr})$	1+1	down-sample rate (of width/height)
width	16	image width
height	16	image height
format	1 or 2	4:2:0 | 4:2:2 | 4:4:4
m_Y	3	bits per class symbol of Y
y-center[0]	8	the first Y center value
⋮	⋮	
y-center[end]	8	the last Y center value
m_U	3	bits per class symbol of U
u-center[0]	8	the first U center value
⋮	⋮	
u-center[end]	8	the last U center value
m_V	3	bits per class symbol of V
v-center[0]	8	the first V center value
⋮	⋮	
v-center[end]	8	the last V center value
k-predictor	3	predictor no. of k-means cluster id
N_{k}	32	# of k-symbols
k-symbol[0]	m_{YUV}	the first symbol value
⋮	⋮	
k-symbol[end]	m_{YUV}	the last symbol value

TAG	BITS	DESCRIPTION
m_{table}	8	# of bits of Huffman table
WL_{max}	m_{table}	max word length of the table
entries(1)	m_{table}	# of entries of word length 1
⋮	⋮	
entries(end)	m_{table}	# of entries of word length WL_max
k-bitstream		content of k-symbol
Q_{res}	8	quantization constant of the residual
N_{res}	8	# of residual value symbols
res-symbol[0]	8	the first symbol value
⋮	⋮	
res-symbol[end]	8	the last symbol value
m_{table}	8	# of bits of Huffman table
WL_{max}	m_{table}	max word length of the table
entries(1)	m_{table}	# of entries of word length 1
⋮	⋮	
entries(end)	m_{table}	# of entries of word length WL_max
res-bitstream		content of res-symbol

Level 2

Figure 3: Coding format (top-to-bottom, left-to-right)

- Methods Description

1. K-means clustering: reference from wiki: K-means_clustering (hyperlink)

   The selected K of Y/U/V (denoted $k_Y/k_U/k_V$) is power of 2, (e.g. $k_Y = 2^{m_Y}$, $m_Y \in \mathbb{Z}$) for purposes of 1) using the minimal number of bits to represent the maximal number of symbols; 2) controlling the number of bits per pixel. ($m = m_Y + m_U + m_V$)

2. Negative values handling after subtraction

   Even though the value range will be doubled after prediction (i.e. from $[0,\, 2^N - 1]$ to $[-2^N + 1,\, 2^N - 1]$), it can still be mapped into $[0,\, 2^N - 1]$ fortunately. Therefore, 'circular subtraction' has been employed for making residual range keeps the same range as raw range. Circular subtraction is implemented with the mathematical form: $x \ominus_N y \equiv (x - y) \mod N$. Similarly, circular addition is also defined as $x \oplus_N y \equiv (x + y) \mod N$.

3. Entropy coding: encoded with Canonical Huffman Coding (hyperlink)

   The codebook in level 1 consists of symbols which represent information of a pixel, but that in level 2 consists of symbols which represent information of a component. (raster-scan order)

4. Down-sampling: for low-bitrate constraints, down-sample the image before encoding.

   The way choosing the representative value: find the median in each $\mathrm{dsr}_w \times \mathrm{dsr}_h$ block.

5. Predictors: use adjacent previous blocks (A, B, C) to predict the current block (D).

C	B
A	D

~~Pred0(X) = 0~~	Pred3(X) = C	Pred6(X) = B + (A - C)/2
Pred1(X) = A	Pred4(X) = A + B - C	Pred7(X) = (A + B)/2
Pred2(X) = B	Pred5(X) = A + (B - C)/2	Pred8(X) = (3A + 3B - 2C)/4

Results

1. Sample Image (see more on GitHub://HW-Lee/ImageCodec/results (hyperlink))

original image

bitrate: 0.6835, PSNR: 27.2617

bitrate: 0.5075, PSNR: 26.9286

bitrate: 0.9608, PSNR: 27.5097

2. BD curve (note that the real encoded file/bitrate is always slightly smaller)

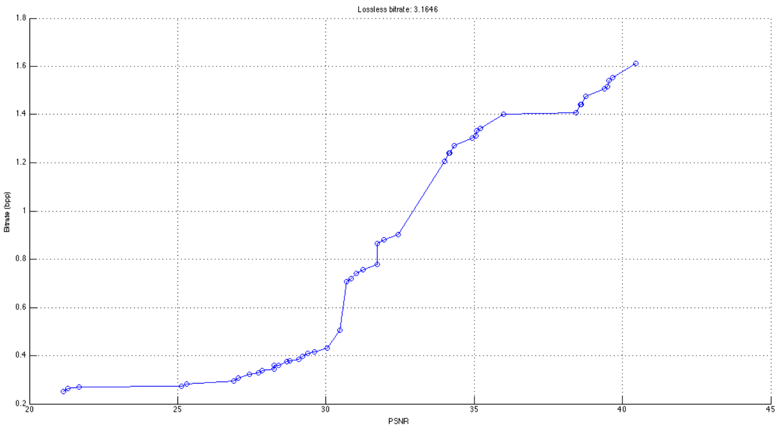

Image 1 (hyperlink)

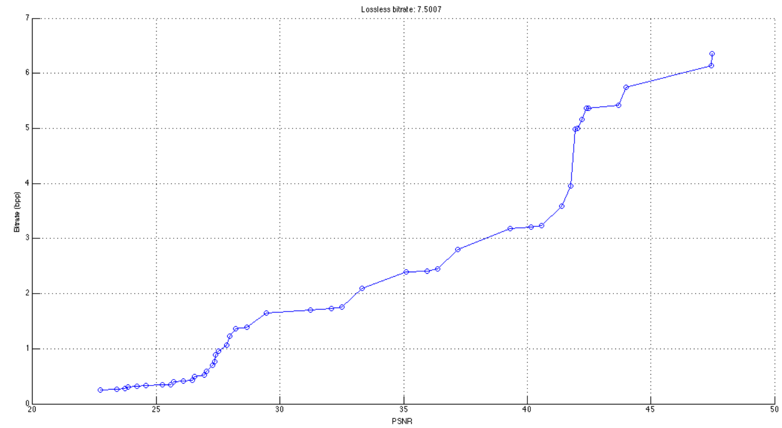

Image 2 (hyperlink)

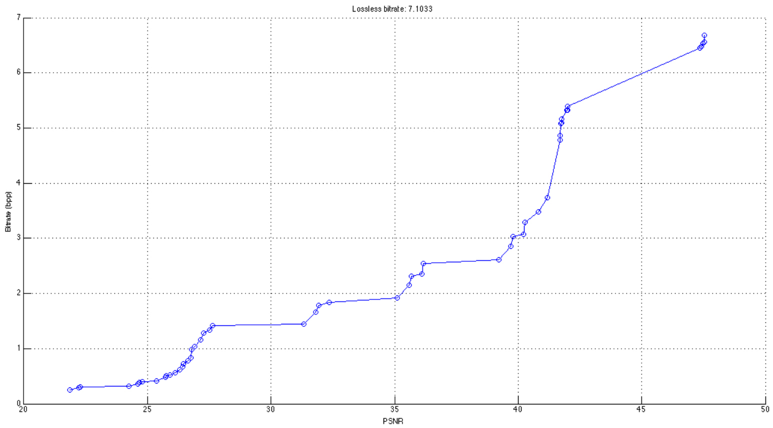

Image 3 (hyperlink)

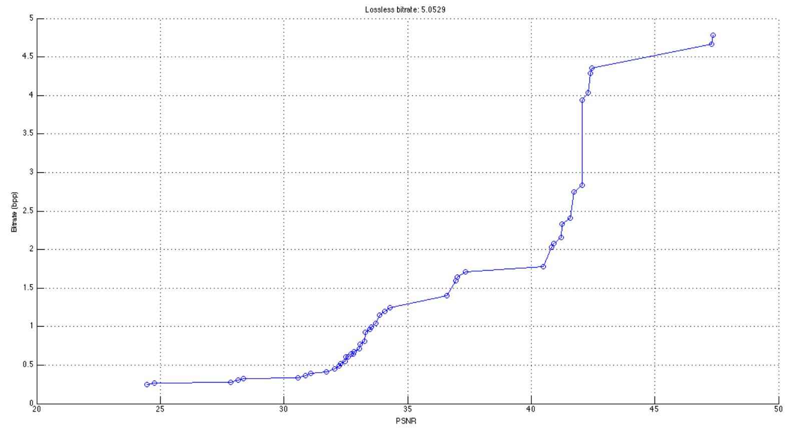

Image 4 (hyperlink)

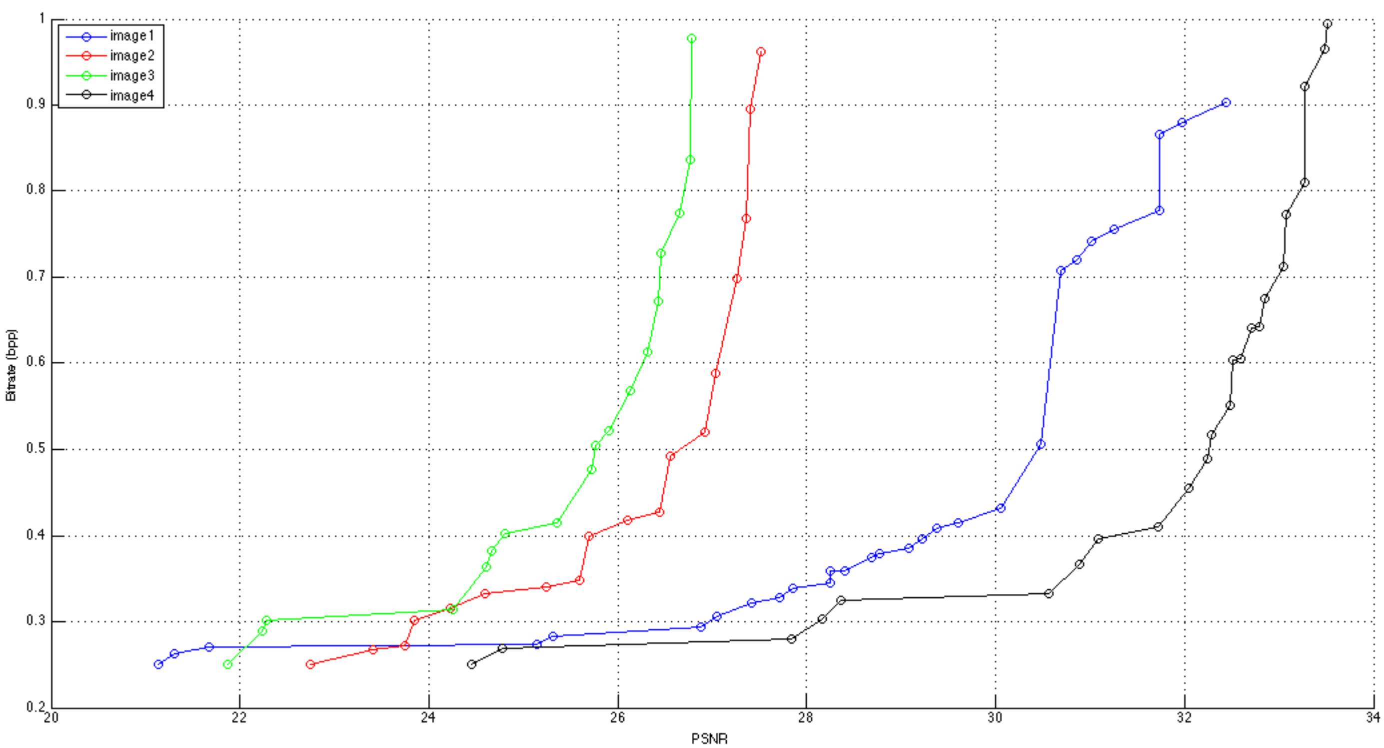

Figure 4: All images where bpp is ranged from 0 to 1

Discussion

1. Huffman Table v.s. Golomb-Rice Table

   Huffman Table requires larger time consumption ($O(n^2)$), and it is powerful after seeing all data to be compressed. On the other hand, Golomb-Rice Table requires less ($O(n)$), and can be constructed without seeing any contents (fast, easily reusable). However, Huffman Table is able to fit any distribution (because it constructs itself after seeing data) and thus reach a better compression ratio. After all, this project is not focused on time consumption. Therefore, Huffman table is employed finally.

2. Transform Coding

   In practice, data after DCT has lower entropy than those after prediction, generally, but just saves about 10% bitrate. Due to considerations of developing time and project variations, it is just implemented but not applied. In addition, prediction is an operation similar to decorrelation, so we expect the efficiency of DCT will not be significant. Finally, DCT is not used.

Supplementary and Links

1. Quick start and APIs: https://github.com/HW-Lee/ImageCodec/blob/master/README.md
2. References: http://sun.aei.polsl.pl/ rstaros/papers/s2006-spe-sfalic.pdf
3. Presentation slides: https://github.com/HW-Lee/ImageCodec/blob/master/report/VSP_Slides.pdf