Image and Video Compression Laboratory Outline of implemented Techniques

Yanchen Huang

Technical University of Munich

Matrikel-Nr: 03715208

Email: yanchen.huang@tum.de

Chang Liu

Technical University of Munich

Matrikel-Nr: 03707882

Email: ge29ney@tum.de

I Introduction

Optimization methods: PCA for color space conversion, deblocking filter, DWT and SPIHT algorithm are implemented in our project and explained in detail in this outline.

II Color Space Conversion: PCA

Conversion between RGB and YCbCr usually follows ITU-R BT.601 standard, which is a fixed space conversion matrix. However, different images have different color properties, to preserve more color information for each image, a better conversion algorithm such as PCA is needed. Based on the reference paper, "A better Color Space Conversion Based on Learned Variances For Image Compression[1]", an optimal RGB-YCbCr convert matrix for each image is defined in equation (1) as T_{enc} and $Offset_{enc}$.

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} x1 & x2 & x3 \\ y1 & y2 & y3 \\ z1 & z2 & z3 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} Y\_offset \\ Cb\_offset \\ Cr\_offset \end{bmatrix} \tag{1}$$

Each row in T_{enc} is the direction of the new YCbCr space's base axis, to find the optimal axis, PCA is applied. Firstly, the whole picture is divided into 8*8 grids, each pixel in the grid should minus the mean value within the same grid. Secondly, covariance matrix of size 3*3 is computed and PCA is utilized to find eigenvalues and eigenvectors. Stack the eigenvectors based on the descend order of the corresponding eigenvalues to form our T_{pca} as follow.

$$\begin{bmatrix} x_p1 & x_p2 & x_p3 \\ y_p1 & y_p2 & y_p3 \\ z_p1 & z_p2 & z_p3 \end{bmatrix}$$

Thirdly, based on the reference paper, according to YCbCr's range restrictions, scaling is done based on following equation(2).

$$\begin{aligned} &[x1, x2, x3] = L1\_normalize([x_p1, x_p2, x_p3]) * 219/255 \\ &Scale_{Cb} := 224/255/(|y_p1| + |y_p2| + |y_p3|) \\ &[y1, y2, y3] = [y_p1, y_p2, y_p3] * Scale_{Cb} \\ &Scale_{Cr} := 224/255/(|z_p1| + |z_p2| + |z_p3|) \\ &[z1, z2, z3] = [z_p1, z_p2, z_p3] * Scale_{Cr} \end{aligned} \tag{2}$$

$Offset_{enc}$ is defined as equation(3).

$$\begin{aligned} &Y\_offset = 16 \\ &Cb\_offset = -1 * sum\_negative(y1, y2, y3) * 255 + 16 \\ &Cr\_offset = -1 * sum\_negative(z1, z2, z3) * 255 + 16 \end{aligned} \tag{3}$$

In decoding, instead of using T_{enc}^{-1}, T_{dec} is computed through least square method, i.e. linear regression. The convert formula is shown in equation(4).

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = T_{dec} * (\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} - \begin{bmatrix} Y\_offset \\ Cb\_offset \\ Cr\_offset \end{bmatrix}) \tag{4}$$

To further optimize the result, polynomial curve fitting is applied to the converted RGB image, i.e. use converted RGB image and original RGB image to train coefficient for a polynomial mapping of degree 3 to find the best match.

III Deblocking filter

Deblocking filter is applied to reduce blocking distortion by smoothing the block edges. Based on H.264/AVC Loop filter[2], we first take vertical and horizontal edges of 8*8 macroblock as depicted in Fig. 1.

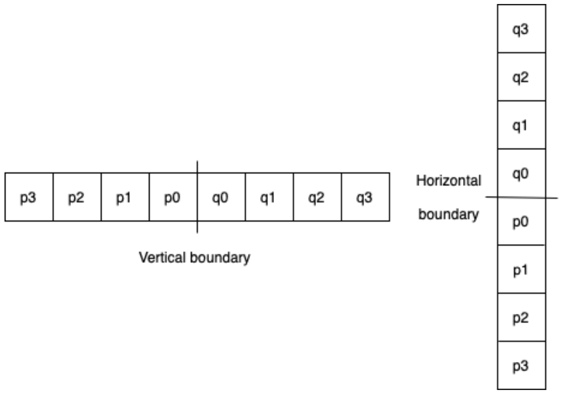

Figure 1: *Pixels adjacent to vertical and horizontal boundaries*

Secondly, to determine the true boundary for applying deblock filter, two conditions shown as follow should be fulfilled, notice that in our project, thresholds α and β are prefixed.

- |p0 - q0| < α
- |p1 - q1| < β

Thirdly, a 4-tap linear filter is applied with inputs p1, p0, q0, q1 if the filter is applicable. Output value would be assigned to position p0 and q0 as the new value for the filtered boundary.

IV Sparsity based adaptive quantization

The basic idea of this algorithm is using the sparsity of the image to adptively choose the quantization table, namely the qScale. The sparsity shows the detail distribution in each area of the image. The sparsity can be determined by the larger parameters of each DCT Block.
The SparsityMap is determined by counting the number of larger parameters in each DCT block.

Algorithm 1 : determine the Sparsity

Input: the blockwise DCT transformed image I_{DCT};
Output: $SparsityMap$ for the I_{DCT};

for $Block_{ij} \in I_{DCT}$ **do**
  Threshold=GAMMA×Max coefficient in DCT Block;
  N= numel(coefficients⩾threshold);
  SparityMap(i,j)=N;
end for

The value GAMMA is image specific. In this paper I take the GAMMA value as 0.015 for the Foreman video sequence. According to the SparsityMap it is shown, that which block contains more details. It means that the greater the number in SparsityMap is, the more details information that block has.
Then the quantization scale can be adptively chosen according to the SparsityMap. The basic concept is:

Algorithm 2 :determine the qScale

Input: The $SparsityMap$; the qScale baseline $qScale$;
Output: Suitable qScale $qScale_{new}$ for each individual Block;
 find the maximum value S_{max} in $SparsityMap$;
 for $SparsityMap_{ij} \in SparsityMap$ **do**
  chose the suitable stepsize Δ of $qScale$;
  $steps$=$SparsityMap_{ij}/(S_max\times(1/8))$;
  $qScale_{new}$= $qScale$+$\Delta\times(4-steps)$;
 end for

The $SparsityMap$ tells us how much detailed information is contained in each block. And we totally divide them into 8 levels(8 contains most detail information).And for the up level, namely $level \in$ [5:8],the quantization factor qScale will be increased. otherwise it will be decreased.
The other step of the whole compression is basically the same as follow.

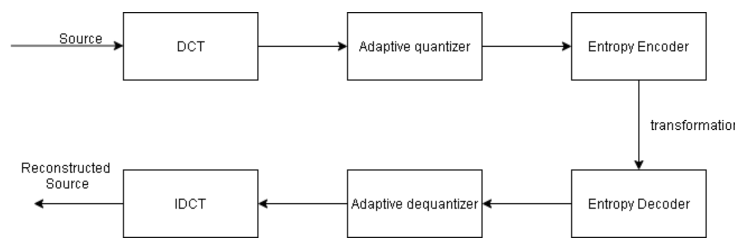

Figure 2: *DCT based Image/video compression*

V DWT-SPIHT based image and video compression

The progress of DWT-SPIHT based image and video compression is very similar to the previous DCT based Compression. The Difference is that instead of dividing the source image to many blocks and do DCT for each block, DWT-SPIHT based compression do n-level DWT for the whole image and quantize them with SPIHT algorithm.

A DWT

when we do the DWT transform we use the *Bior*4.4 filter.[1]The DWT process of the image is as follows.
the DWT transformed image will be saved as I_{DWT},which seems like a multi-divided quadrate. and then seems it contains too much

Algorithm 3 :n levels DWT

Input: The *sourceimage*; filters Lo_D,Hi_D
Output: DWT Transformed image I_{DWT};
LL=*sourceimage*;
for *level* $\in [1:n]$ **do**
L=LL*Lo_D;
LL= (downsampled L)*Lo_D;
LH=(downsampled L)*Hi_D;
H=LL*Hi_D;
HL= (downsampled H)*Lo_D;
HH=(downsampled H)*Hi_D;
Sort each computed part in to I_{DWT}.
end for

coefficient. The I_{DWT} should be quantized and compressed with SPIHT algorithm. The SPIHT algrithm is very similar to EZW algorithm.

B SPIHT algorithm

The SPIHT algorithm based on EZW algorithm. It applies the same Successive-Approximation Quantization. The main difference of the SPIHT algorithm to EZW algotirhm is that the SPIHT algorithm used a patial orientation tree to more efficiently represent the coefficient structure. The "patial orientation tree" allows it eliminate some root, that except the root all the other node are unimportant. After the SPIHT Algorithm the DWT Transformed image data will be quantized

References

[1] Li, Manman. A Better Color Space Conversion Based on Learned Variances For Image Compression. CVPR Workshops (2019).

[2] H.264/AVC Loop Filter. https://www.vcodex.com/h264avc-loop-filter/. Accessed 02.02.2020.

[3] Youngjun Yoo Antonio O. and Bin Yu. adptive quantization of image subbands with efficient overhead rate selection. https://www.researchgate.net/publication/3671825. Accessed 31.01.2020.

[4] K. S. Thyagarajan. Still Image and video compression with Matlab. ISBN 978-0-470-48416-6

[5] David Fridovich-Keil and Grace Kuo. Compressed Sensing for Image Compression.

[6] Gregory Wallace, The JPEG Still Image Compression Standard,

[7] Prof. Dr.-Ing. Eckehard Steinbach. Multidimensional signal processing Lecture Skript

[8] SPHIT Algorithm. http://www.ws.binghamton.edu/fowler/fowler%2

Algorithm 4 :SPIHT algorithm

Initialization ;
compute Threshol s
LIP=all elements in Tree root
LSP=empty;
for $n \in [1 : N]$ **do**
  Significance Map Encoding(Sorting Pass)
  for $Coeff(i, j) \in LIP$ **do**
    search all coefficient in LIP
    if $coefficient \geqslant threshold$ **then**
      S_{ij}=1;
    else
      S_{ij}=0;
    end if
    According to the S table fill the LSP Table
  end for
  Process LIS
  for $searchallelemetinLIS$ **do**
    if $Type = D$ **then**
      S_{ij}= sign(D_{ij});
      if $S_{ij} = 1$ **then**
        add item in the LSP list;
      else
        add item the LIP list;
      end if
    else$Type = L$
      S_{ij} = sign(L_{ij});
    end if
  end for
  Refinement Pass
  Process LSP
  for *all element in LSP list except those just added above*
do
    Output the nth most significant bit of coeff
  end for
  Update;
  Update n.
end for