EE669 2015Spring

Homework#2 Report

Name: Shanglin YANG
ID: 3795329308

Problem 1: QM Coder
Part A: Written Questions

Problem 1 QM Coder

PART A Written Question.

Seate	Input	MPS	Qe	A	C	Output
10	\	0	299A	10000	0000	\
10	1	0	299A	299A	D666	1
9				533A	ACCC	1
8				A668	5998	
8	1		32BA	32B4	CD4C	1
7				6568	9A98	1
6				CDD0	3530	
6	0		3C3D	8E93	353D	
6	0	0	3C3D	5256	3530	0
7				A4AC	6A60	
7	0		375E	6D4E	6A60	0+1
8				DA9C	DAC0	Carry=1
8	1	0	32B4	32BA	7CA8	0+1
7				6568	F950	1+1=0 Carry=1
6				CAD0	F2A0	
6	0	0	3C3D	8E93	F2A0	
6	0	0	3C3D	5256	F2A0	1+1=0 Carry=1
7				A4AC	E540	
7	1	0	375E	375E	528E	0
				6EBC	A51C	1 Carry=1
6	0	0	3C3D	DD78	4A38	

The output is 1111011D001 (11 bits).

Part B: Encoding using QM encoder

1.2.1 Abstract and Motivation

QM encoder is an adaptive binary arithmetic coding procedure which is widely used in entropy. The QM coder is a binary arithmetic coder, implying that it codes a stream of only two symbols 0 and 1. A source with multiple symbols can also be coded by decomposing each symbol using a binary decision tree. It involves Recursive interval subdivision and conditional probability. Advantages of a binary arithmetic coder lie in that it generates compressed data as a finite precision fraction which identifies an interval on the number line. Each symbol is encoded or decoded on the basis of a probability estimate which determines where the binary arithmetic coder splits the interval into two subintervals. In standard process, we use the general QE table to do the estimation. In the context based. The current symbol determines which subinterval becomes the new interval. When the size of the new interval drops below a minimum value, renormalization shifts the precision until it is greater than or equal to the minimum size. With each shift, a bit is produced for the compressed data stream.

1.2.2 Approach and Procedures

Basic Describtion: In the first question. We do the standard QN coding. The main process involving initial(), coding process depending on of on LPS & MPS symbol , normal process and flush the left symbol.

First, initial the register and parameter. Using the **Initenc()** function.

```
void Initenc()
{
   A_register = 0x10000; // A register
   C_register = 0; // C register
   s = 0;  // State
   CT = 11; // Out number count
   MPS = 0; // The symbol with high frequncy
   Outbuff = 0; //Output BUFF
   BPST = 0; // The flag of the first symbol
}
```

Then, we need map the symbol into the binary stream , here I use two ways of mapping. The first way is directly to map symbol of each bits to binary stream;

The second the way is to map each symbol of plane to the bit streams.

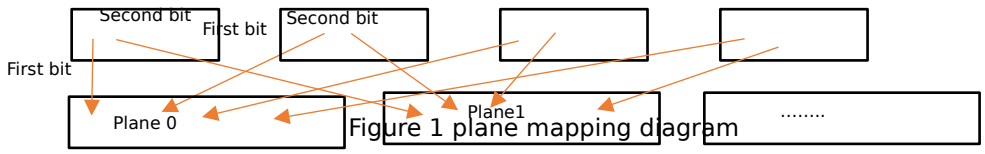


Figure 1 plane mapping diagram

After which , we encode symbol based on whether the symbol is MPS , there are two function to solve the symbol, void Code_LPS() and Code_MPS().When the size of the new interval drops below a minimum value or the interval is less than 0x8000, we need to renormalize the register, Renorm().Only this moment ,we use Byte_out() to output the MSB of the C register. In the byte_out process, we need to choose to solve the carry conflicts. We stacked 0xffs in the stack, based on the current symbol to decide whether output the symbol in the stack. There are Output_stacked_zeros(), Output_stacked_0xffs(), Stuff_0().

At last, use the flush() to output the left symbol in C register, there we

generate as much as zeros at the tail of the register as well as keep the interval unchanged. Then cut all the left zero and add two 0XFFFF to the tail for decoder recognize the end of the files.

In the context based QM_CODE, we need the context table involving building the using the table, which is decided by the rule. For n=0, it is the standard QM coder .For context rule n=1, the context is set to 1 if the previous bit is 1. The context is set to 0 if the previous bit is 0. For context rule n=2, the previous 2 bits has ‘00’, ‘01’, ’10’, ‘11’, four cases. The context is set 0 to 3 corresponding to the binary value. For context rule n=3, the context is set 0 to 7 corresponding to the binary value of previous 3 bits ‘000’ to ‘111’. For context rule n=4, the context is set 0 to 15 corresponding to the binary value of previous 4 bits ‘0000’ to ‘1111’.

For the initial data ,take them as n=0, when we get enough symbol to build the table ,we change it to the =1,2 or3. As shown. Here I use the reference code and the function encode(value, context) to get the QM coder.

```
if (file_start < (rule_num))
{

      qm->encode(value, 0);
      file_start++;
      buffvector.push_back(value);
      context <<= 1;
      context += value;
}
else
{
      qm->encode(value, context);
      context <<= 1;
      context += value;
      switch (rule_num)
      {
      case 0:context &= 0x00; break;
      case 1:context &= 0x01; break;
      case 2:context &= 0x03; break;
      case 3:context &= 0x07; break;
      }
```

For image file, I use the Zigzag preprocccesing the enchance the Encoder.

1.2.3 Result

Table 1 Standard QM compression results using different mapping function comparison

		Basic Mapping		Plane Mapping	
File Type	Input size(Byte)	Compressed File Size(Byte)	Compressed File ratio	Compressed File Size(Byte)	Compressed File ratio
Binary	65536	6801	10.38%	8271	**12.62%**
Test	8358	8609	103.00%	6182	**73.97%**
Audio	65536	66,754	101.86%	62,965	**96.08%**
imaginary	65536	67,399	102.84%	63,587	**97.03%**

Table 2 CABC QM compression results comparison

		Standard		n=1	
File Type	Input size(Byte)	Compressed File Size(Byte)	Compressed File ratio	Compressed File Size(Byte)	Compressed File ratio
Binary	65536	6801	10.38%	1914	2.92%
Test	8358	8609	103.00%	8635	103.31%
Audio	65536	66,754	101.86%	66253	101.09%
imaginary	65536	67,399	102.84%	66962	102.18%
		n=2		n=3	
File Type	Input size(Byte)	Compressed File Size(Byte)	Compressed File ratio	Compressed File Size(Byte)	Compressed File ratio
Binary	65536	1885	2.88%	1864	2.84%
Test	8358	8562	102.44%	8483	101.50%
Audio	65536	66139	100.92%	65926	100.60%
imaginary	65536	66528	101.51%	66509	101.48%

Table 3 Preprocessed image compression results comparison

	Basic image		Image with preprocessing	
Input size(Byte)	65536			
Context rule	Compressed File Size(Byte)	Compressed File ratio	Compressed File Size(Byte)	Compressed File ratio
n=1	66962	102.176%	66910	102.097%
n=2	66528	101.514%	66398	101.315%
n=3	66509	101.485%	66199	101.012%

1.2.3 Discussion

(A) Comparison and Conclusion of context based QM coding

From the table 1, we can see that QM coed does not get good result except binary.dat. The reason for that is that in the binary.dat, there are long sequence of 1 or 0, and skewed distribution of two symbol. While in other three files the symbol (1 or 0) are almost equally distributed and there are not long sequence of continue symbol. Considering the characteristic of QM encoder, it generate code when receiving LPS or A is two small (requiring renormalization), so if sequence change frequently, it generate more codes, and if symbols are equally distributed, the QE is almost 0.5, in the MPS_CODE, A=A-QE, the A decreases more quickly.

The result remind us to get the symbol stream as more as satisfying the requirement of the QM encoder--- skewed distribution and long continue sequence. So that, we can use different mapping function and preprocessing.

(B) The influence of Mapping Function

In this process, I use two mapping function, From the Table 1 we can see except binary.dat, the result have improvements when using the plane mapping.

Although the symbols of different samples are not correlated, but for the data of different planes, there exist relation. For example, for the sample in Text, which are all less than 128, then the first 4 bits are zeros. If we connect these zeros together, we get long sequence of the zero which is the local optimization for the QM coder.

The reason for the exception is that the binary.dat itself has good condition for the QM

encoder, while the aim of the plane mapping is to improve the continuity of the symbol, it may break the original condition of file.

The result reminds us again the performance of EM coder is related to the characteristic of input samples, we can change the input symbols using different mapping functions to get better results.

(C) Comparison and Conclusion of context based QM coding

From the table2, we can find the context based QM coding improve the performance of the QM coder compared to the standard QM coder, especially for the binary.dat. Meanwhile, the more previous symbol used (Higher n), the better results we can get.

The reason for that is that in the context based, we estimate QE considering the previous symbols, which give us the more accurate estimation of the MPS and QE, in other words, we could get more continuous complex symbol stream and more skewed distribution. When we increase the n, we use more previous symbols leading the better understand of the contest---more accurate estimation.

(D)The preprocessing of image

From the table3, we can see that applying zigzag scanning to image.dat improve the overall compression performance in context-based QM encoders.
The reason for that is that there exist correlation of pixels in neighbors in image, so the zigzag scanning increases the chances of successive 1s or 0s in sequences-Better condition for QM coder.

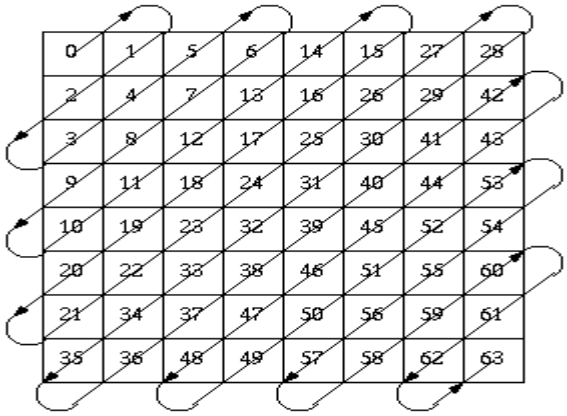


Figure2 Zigzag processing diagram

Problem 2: Scalar and Vector Quantization
Part A: Lloyd-Max Scalar Quantizer Design

2.1.1 Abstract and Motivation

Quantization, in mathematics and digital signal processing, is the process of mapping a large set of input values to a (countable) smaller set – such as rounding values to some unit of precision. A device or algorithmic function that performs quantization is called a quantizer. Scalar quantization is the most common type of quantization: it maps a scalar input value x to a scalar output value y. Quantization is important for the later processing and the performance such as encoder compression ratio. It is a important part in lossy compression process. Lloyd-Max scalar quantizer is a typical application of scalar quantization. It is used in a Euclidean space, the distance function serves as a measure of similarity between points. And get the optimization of the Quantization.

2.1.2 Approach and Procedures

At first, we need to form a training set to train the quantizer. Read the image data of trainset into quantizer and get the histogram the trainset. the 3 images: chem.256,house.256 and moon.256 and collect the histogram of each image.

The next step is to initialize the quantizer. For 3 bits quantizer, we need 7 initial thresholds besides 0 and 255. For 5 bits quantizer, the initialization method is the same, but we need 31 initial thresholds besides 0 and 255 and the range is total pixel numbers divide by 32.
I use the function void Initial(int size,double *weight) ,size is the 8 for 3 bits quantizer and 32 for 5 bits quantizer. *weight record the histogram of the train set. I get the initial thresholds by recording the pixel whose cumulative number reaches the integral multiples of the range, where range is total pixel numbers divide by 8. Considering the accuracy, I use double number rather than int number to represent the threshold.

Then we need to do the iteration for the optimization.
When the iterative optimization process stops, we can get the trained thresholds and ranges. They now should not be integers so we should round them to its nearest integer. And then we can apply the trained thresholds and ranges to other images for quantization. Separately input each image , record the histogram, quantinizing and calculate the related outcomes.

2.1.3 Result

(a) Histogram of each image
(b)

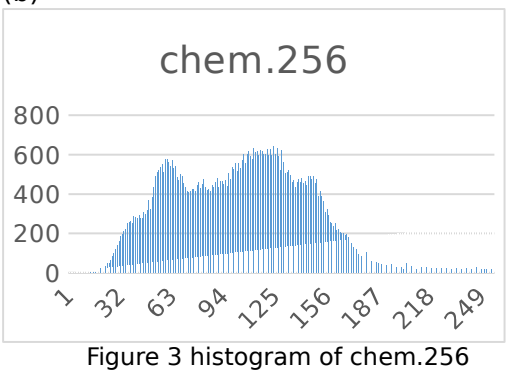


Figure 3 histogram of chem.256

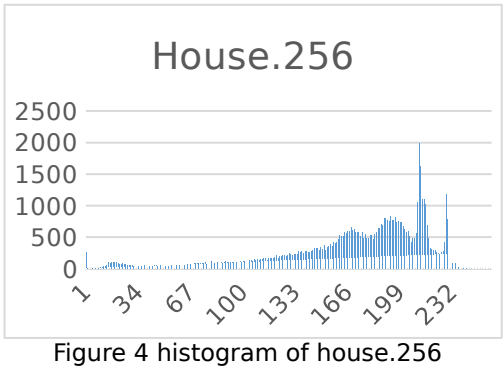


Figure 4 histogram of house.256

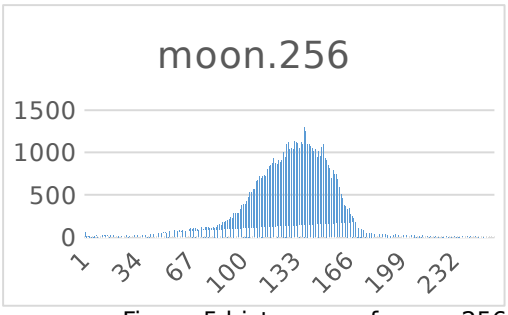


Figure 5 histogram of moon.256

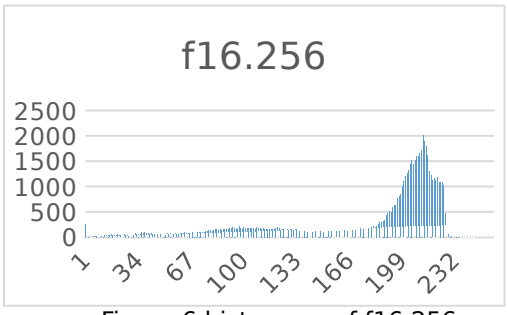


Figure 6 histogram of f16.256

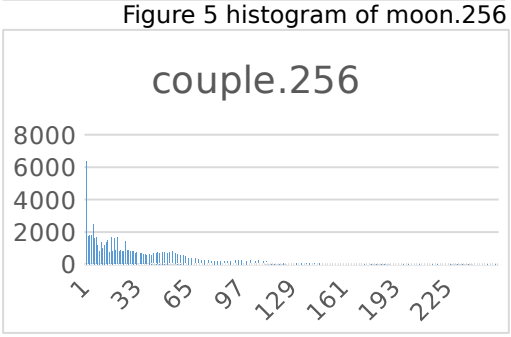


Figure 7 histogram of couple.256

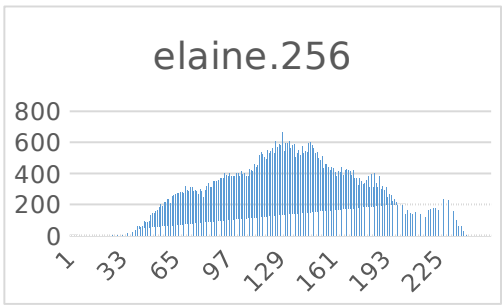


Figure 8 histogram of elaine.256

(b) PSNR of each iteration and number of iteration
3 bits quantizer:

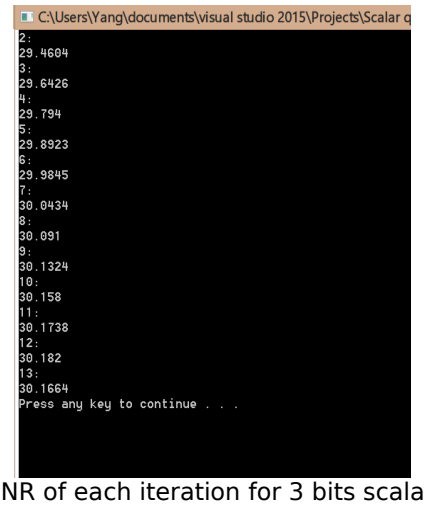


Figure 9 PSNR of each iteration for 3 bits scalar quantizer

Number of iteration=13;

5 bits quantizer:

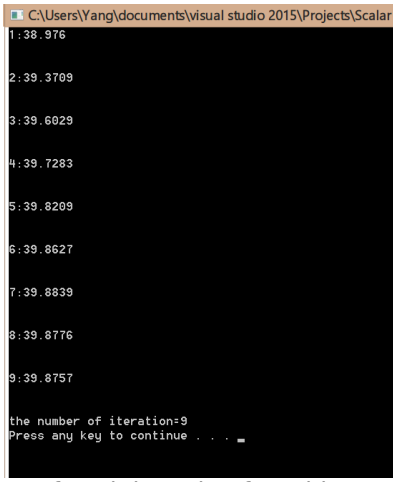
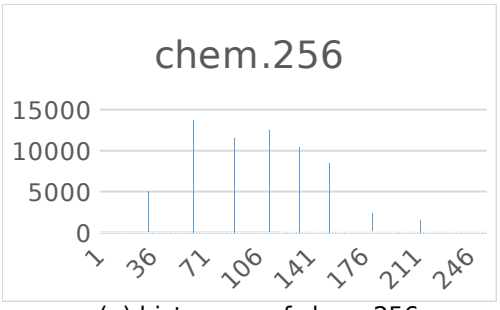


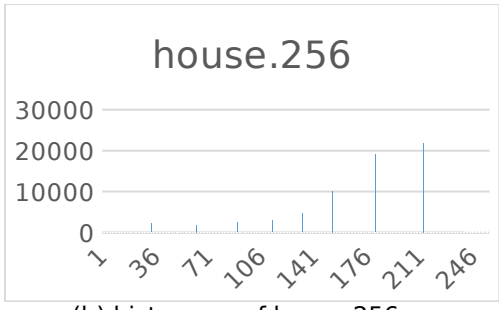
Figure 10 PSNR of each iteration for 3 bits scalar quantizer

Number of iteration=9;

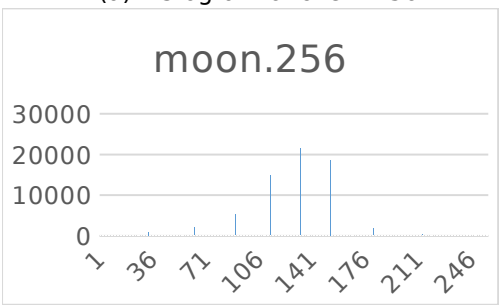
(c) Histogram of 6 images after quantization



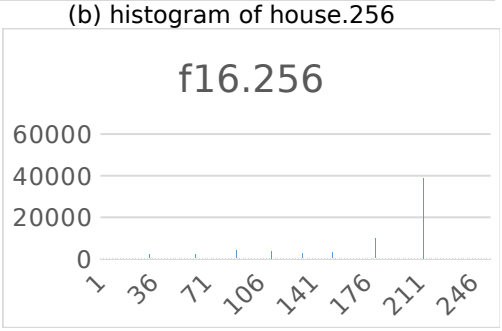
(a) histogram of chem.256



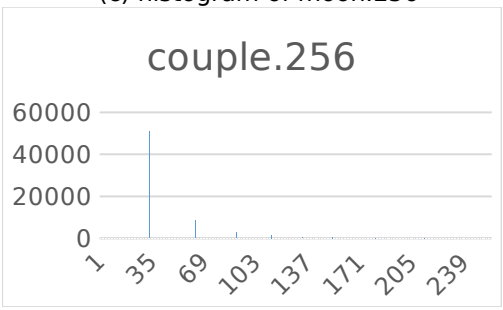
(b) histogram of house.256



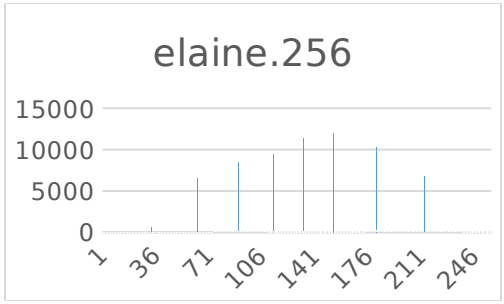
(c) histogram of moon.256



(d) histogram of f16.256

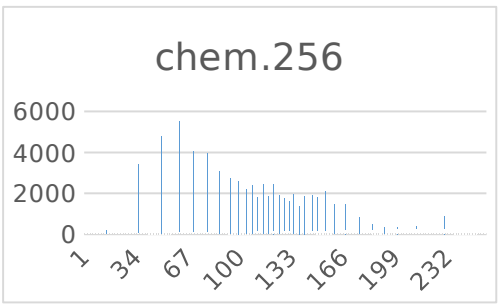


(e) histogram of couple.256

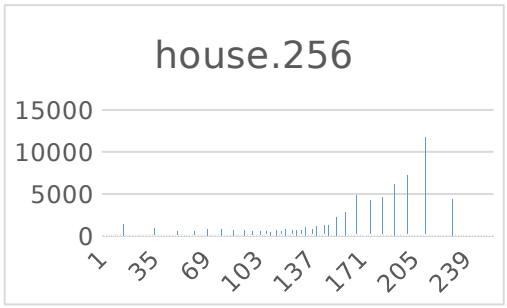


(f) histogram of elaine.256

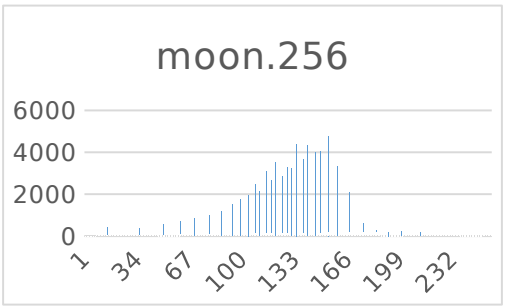
Figure 11 3 Bits Scalar Quantization Results



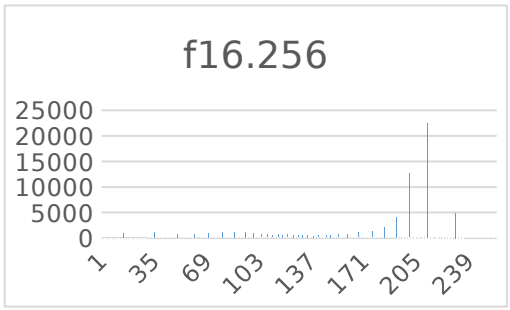
(a) histogram of chem.256



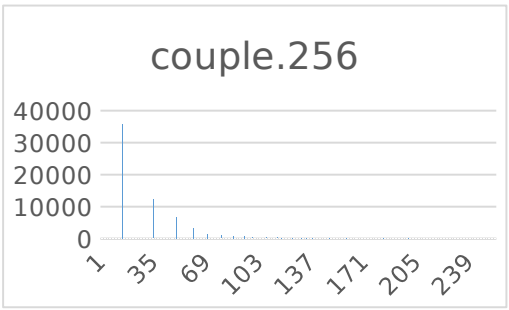
(b) histogram of house.256



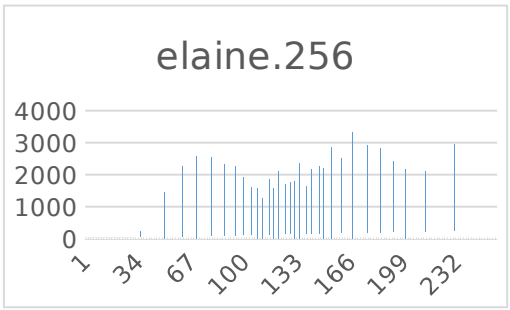
(c) histogram of moon.256



(d) histogram of f16.256



(e) histogram of couple.256



(f) histogram of elaine.256

Figure 12 5 Bits Scalar Quantization Results

(C)PSNR of each image

Table 4 PSNR of Each Image

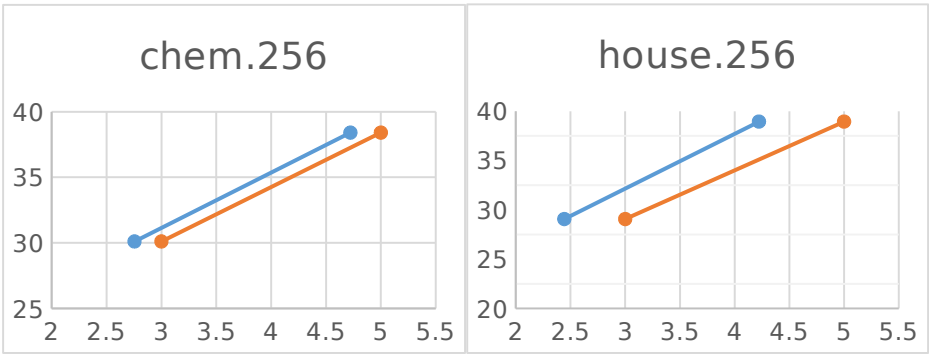
	chem.256	house.256	moon.256	f16.256	couple.256	elaine.256
3 bits	30.110958	29.068063	31.634426	29.678554	22.924221	30.12165278
5 bits	38.406195	38.947413	43.604585	36.451545	31.393431	40.91542348

(D)Entropy of Each Image

Table 5 Entropy of Each Image

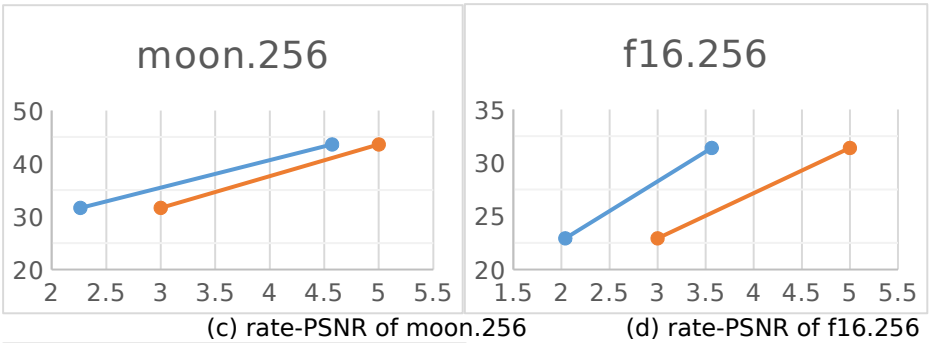
	chem.256	house.256	moon.256	f16.256	couple.256	elaine.256
3 bits	2.7549027	2.443573	2.2632416	2.0399941	1.13856869	2.821996236
5 bits	4.7223869	4.2222203	4.5719835	3.5631845	2.27197916	4.88566716

(E) Plot of rate-PSNR for each image
*red line rate-PSNR after quantinization
*blue line theoretical rate -PSNR



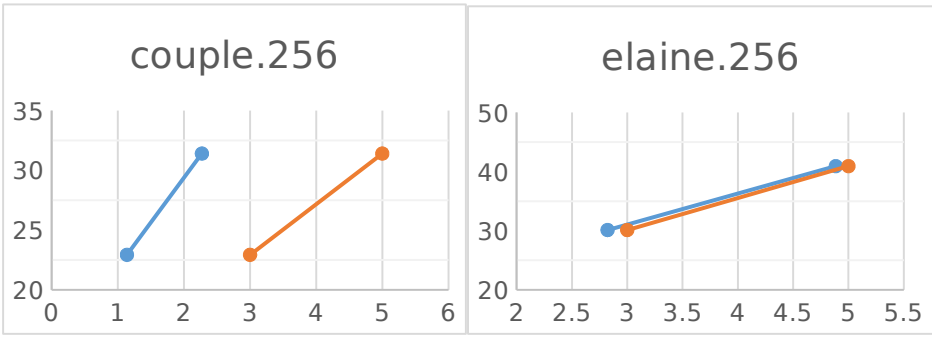
(a) rate-PSNR of chem.256

(b) rate-PSNR of house.256



(c) rate-PSNR of moon.256

(d) rate-PSNR of f16.256



(e) rate-PSNR of couple.256

(f) rate-PSNR of elaine.256

Figure13 Rate-PSNR Plane for Each Image

2.1.4 Discussion
(A) Discussion of Initialization
Good Initialization not only leads us to the global optimization rather than the local optimization, but also get the result quicker using less iteration;
I use the function void **Initial(int size,double *weight)** ,size is the 8 for 3 bits quantizer and 32 for 5 bits quantizer. *weight record the histogram of the train set. I get the initial thresholds by recording the pixel whose cumulative number reaches the integral multiples of the range, where range is total pixel numbers divide by 8(or 32). Considering the accuracy, I use double number rather than int number to represent the threshold. For example , Here is the threshold of initial table for 3 bits quantizer.

```
[1]  72.606504065040653     double
[2]  103.42660178426601     double
[3]  119.80886758538047     double
[4]  133.42984907769704     double
[5]  146.85574148874784     double
[6]  164.05341506129596     double
[7]  194.96000000000001     double
```

Figure14 Thresholds of initial table for 3 bit quantizer

(B) Discussion of train of Quantizer

To form the training set, I add the histograms of chem.256, house.256 and moon.256 together. Although for each image the distribution is skewed, the sum histogram is relatively uniform, but has more pixels in the middle intensity range. And kind of equally distribution.

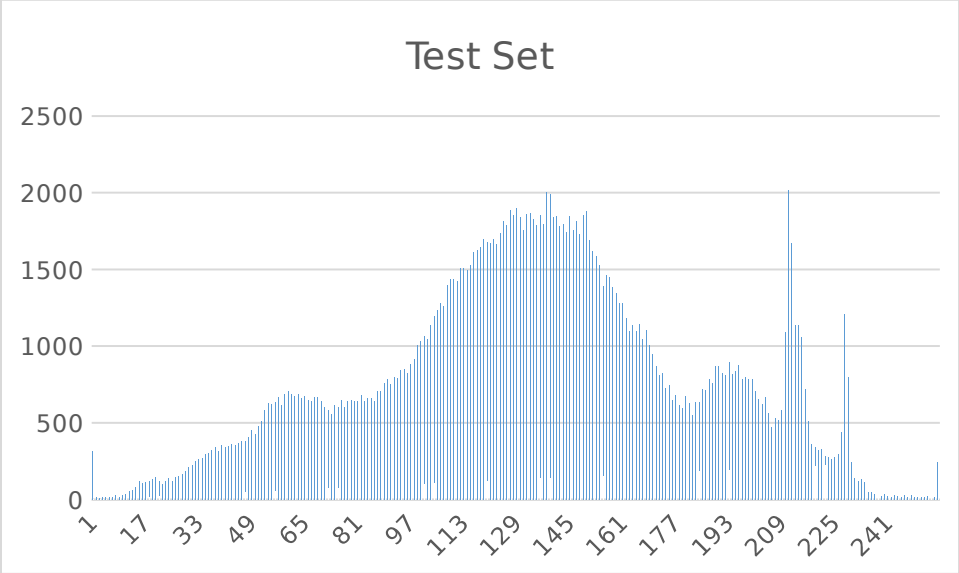


Figure14 Histogram of the train set

From the each PSNR OF 3 bits SQ and 5 bits SQ, find PSNR increases as the iteration goes on. We can see that the This indicates that the optimization process is getting better scalar quantizer as the iterationincreases. The iteration number for 3 bits SQ is 13 and for 5 bits SQ is 9. The iteration numbe is related to the initialization of the quantizer. If the quantizer has a good initialization, it can converge more quickly so that the iteration number would be smaller. Otherwise, it needs more iteration to find the minimum point.

Besides, we can find the PSNR increase as we use more bits, and the difference is near 12 db. which satisfy that using more bits decrease noise and each bit increase almost 6 db.

However as last PSNR decrease little, the reason is the result kind of over the optimization but the result difference is still less than the limit. The result is acceptable.

(C) Discussion of results after Quantization and relation to the train set

For the indicator of “good” quantization, the main idea is that the quantization pane should represent the “features” of the original image, more generally, two image should be similar. The key standard indicator we can use is the PSNR.

In fact, the key characteristic of good quantization is for symbol with high frequency, there should be smaller interval and more quantization. And the local histogram in small interval should be close.

Back to the result of 6 image, for example in chem.256’s histogram, both 3 bits quantization result and 5bits quantization result have a similar contour as the original one, while the contour of couple.256’s has a changed both in 3 bits and 5 bits quantization result compared with the original one. Which imply the quantization for the chem.256 is better than couple.256.

The fact agrees with the PSNR results, for chem.256 get 30.110958 at 3 bits and 38.406195 at 5 bits, while for the couple.256 ,they are only 22.9242421 and 31.3934131

Reason for this difference is the trainset we choose, as said below, the train set histogram has kind of uniform distribution and more data in the middle area. So

,for chem.256 ,which has similar histogram can get better result while couple.256 whose histogram differ mush ccompared to the train set get the bad results.

From the rate-distortion plane, we can see that 5 bits SQ has a better result than 3 bits SQ. The result of 5 bit is more accurate. Meanwhile, chem.256, house.256 and f16.256 has similar PSNR results in 3 bits quantization. But The PSNRs are distinct when 5 bits SQ is applied. Moon.256 has the best result while couple.256 has the worst result. It is because the effect of quantization is more clear when the bit increase.as we talk about, the effect of the train set choose is more clear when it comes to 5 bits.

All these results reminds us the important of the choose of train set, which has big influence on the final results. For general case we can use uniform or Gaussian distribution, but for special case, the context based result is better.

(D) Discussion of Rate-PSNR Plane

We can see that couple.256 has the smallest entropy and Elaine.256 has the largest entropy. An image with a more uniform distributed histogram has larger entropy than an image with a less uniform distributed histogram. The entropy provide us the theoretically optimal word length.

Compare the two line in the Rate-PSNR Plane, we can find the better quantization we get, the rate-PSNR line is almost near to the theoretically optimal results. The bigger difference between the two line ,meaning there are more redundancy after quantinizing process. The two plot lines in Elaine.256 are close, which means that its code word length is reaching its optimal theoretical entropy bound. The two plot lines in couple.256 are far away from each other, which means its code word length is far away from its optimal theoretical entropy bound.

Part B: Vector Quantization

2.2.1 Abstract and Motivation

Vector quantization (VQ) is a classical quantization technique from signal processing which allows the modeling of probability density functions by the distribution of prototype vectors. It was originally used for data compression. It works by dividing a large set of points (vectors) into groups having approximately the same number of points closest to them. Each group is represented by its centroid point, as in k-means and some other clustering algorithms.

TSVQ has been studied extensively in vector quantization from the perspective of data compression.

Here is how the algorithm works:

1. First we apply k-means to get 2 centroids or prototypes within the entire data set. This provides us with a boundary between the two clusters, which would be a straight line based on the nearest neighbor rule.

2. Next, the data are assigned to the 2 centroids.

3. Then, for the data assigned to each centroid (call them a group), apply 2 centroid k-means to each group separately. The initialization can be done by splitting the centroid into two. Note that data points channeled to different centroids are treated separately.

4. Repeat the above step.

5.

2.2.2 Approach and Procedures

First , we need to do the image block ,meaning we input the image data block by block , convert it into vector ,then output as a stream of vector.

I first use the train set image to get 3 total trainset vector stream with dimension equal to 4, 16 and 64. Then convert 3 test image each into 3 testset vector stream with dimension equal to 4, 16 and 64. Now we get 3 test_set and 9 train_set.

Use the reference code to get the codebooks and the result of the quantization.

For Standard vector quantizaton ,for example ,using command "**./stdvq -t trainset -c bookd_16 -d 4 -f 16**" to get the codebook with dimension=4 and vector size=16, the trainset has dimension equal to 4;

Then using command"**./stdvqe -c book4_16 -i testset_couple_4 -o out_couple_4_16**" to do the quantization of the couple_test whose dimension is 4 using the codebooks I just get.

For TSVQ, for example, using command "**./tsvq -t trainset -c book4_16 -d 4 -r 4** " to get the codebooks with dimension =4 and 4.0 bits per vector, then using command "**/tsvqe -c book4_16 -i testset_couple_4 -o out_couple_4_16**" to do the qutization of the couple_test whose dimension is 4 using the codebooks I just get.

2.2.3 Results

Table 6 Entropy and distortion of Each Pair using Standard VQ

	Dimension	D=4			D=16			D=64		
	Vector Size	S=16	S=32	S=64	S=16	S=32	S=64	S=16	S=32	S=64
couple	entropy	1.434005	2.128132	2.834615	1.224747	1.845075	2.596398	1.153452	1.930759	2.272109
	distortion	1295.6	510.9876	294.7688	8056.086	4094.298	2578.48	42860.44	35078.45	22523.22
	Rate (bpv):	4	5	6	4	5	6	4	5	6
	Vectors encoded:	16384	16385	16386	4096	4096	4096	1024	1024	1024
f16	entropy	3.114008	3.907905	4.654266	3.032821	3.663895	4.601357	2.992541	3.82082	4.66801
	distortion	1079.893	538.8117	395.7807	7535.661	5820.549	4842.092	57186.1	43020.61	39365.44
	Rate (bpv):	4	5	6	4	5	6	4	5	6
	Vectors encoded:	16384	16385	16386	4096	4096	4096	1024	1024	1024
eplain	entropy	3.86174	4.346165	5.137041	3.621775	4.231128	5.167947	3.840949	4.51629	5.352405
	distortion	291.2213	219.7298	158.8136	2894.68	2246.708	1821.204	24841.68	20311.65	18415.61
	Rate (bpv):	4	5	6	4	5	6	4	5	6
	Vectors encoded:	16384	16385	16386	4096	4096	4096	1024	1024	1024

Table 7 Entropy and distortion of Each Pair using TSVQ

	Dimension	D=4			D=16			D=64		
	Vector Size	S=16	S=32	S=64	S=16	S=32	S=64	S=16	S=32	S=64
couple	entropy	2.129609	2.196629	2.514425	1.526943	2.145567	2.196087	1.333761	1.46113	1.613886
	distortion	467.0264	448.4392	393.9658	7053.948	3044.469	2955.485	41136.27	40487.76	39823.1
	Rate (bpv):	4	5	6	4	5	6	4	5	6
	Vectors encoded:	16384	16385	16386	4096	4096	4096	1024	1024	1024
f16	entropy	3.174766	3.64364	4.033267	2.888786	3.704145	4.223799	2.796668	3.488397	4.22493
	distortion	792.0332	551.7991	342.7233	8859.616	6019.736	5049.806	57038.21	49179.72	42868.72
	Rate (bpv):	4	5	6	4	5	6	4	5	6
	Vectors encoded:	16384	16385	16386	4096	4096	4096	1024	1024	1024
eplain	entropy	3.476358	4.172108	4.413373	3.523286	4.0131	4.629034	3.527616	4.105534	4.647739
	distortion	320.8605	229.0663	180.8058	3033.648	2466.864	2063.08	25744.4	23565.29	21376.42
	Rate (bpv):	4	5	6	4	5	6	4	5	6
	Vectors encoded:	16384	16385	16386	4096	4096	4096	1024	1024	1024

2.2.4 Discussion

(A) Discussion on Standard VQ

For distortion, we can see the least distortion is the 4 dimension and 64 vector size, and the max distortion is the 64 dimension and 16 vector size. We can find using same dimension, when you use bigger vector size, the distortion become less; using same vector size, when you using bigger block(dimension),the distortion become bigger. And the influence of block is bigger than the vector size. For example, for couple.256, Using 16 dimension and 64 vector size leads to distortion equal to 2578, while using the 64 dimension and 16 vector size leas to 42860.

Reason for that using larger block meaning you remove more information for using 1 vector to represent the whole block. And use more vector size, meaning there are more region to accommodate the original image which leads to the improvement of accuracy.

For entropy, which is the theoretical of the code word length, from the table we can see, when using larger vector size, the entropy also increases. And when using same vector size, the bigger dimension, the less entropy we will get.

Reason for the relation is that, when you using more vector size, you generate more position which leans to the increase of information to encode and leads to the increase of the entropy, Considering the block is to using a general value to represent same dimension values, the bigger the block, the total distribution of the data is more uniform, thus the entropy as well as decrease.

(B) Discussion on TSVQ

The characteristic of the data is almost same as Standard VQ.

For distortion, we can see the least distortion is the 4 dimension and 64 vector size, and the max distortion is the 64 dimension and 16 vector size. We can find using same dimension, when you use bigger vector size, the distortion become less; using same vector size, when you using bigger block(dimension),the distortion become bigger. And the influence of block is bigger than the vector size. For example, for couple.256, Using 16 dimension and 64 vector size leads to distortion equal to 2578, while using the 64 dimension and 16 vector size leas to 42860.

Reason for that using larger block meaning you remove more information for using 1 vector to represent the whole block. And use more vector size, meaning there are more region to accommodate the original image which leads to the improvement of accuracy.

For entropy, which is the theoretical of the code word length, from the table we can see, when using larger vector size, the entropy also increases. And when using same vector size, the bigger dimension, the less entropy we will get.

Reason for the relation is that, when you using more vector size, you generate more position which leans to the increase of information to encode and leads to the increase of the entropy, Considering the block is to using a general value to represent same dimension values, the bigger the block, the total distribution of the data is more uniform, thus the entropy as well as decrease.

(C) Comparison of Standard VQ and TSVQ

For distortion, from the table, we can not see the direct relation between the Standard VQ and TSVQ,. They are almost at the same level. In fact, considering the distortion result is under the influence of the parameter set ,so it hard to tell which is better, but we can said both the algorithm can get acceptable results

But for entropy, we can see the entropy of TSVQ is almost less than the data of Standard VQ , reason for that could be explained in two aspect ,the first is that the TSVQ generate new cell based on the statistic results which could better generate a more uniform distribution, while standard VQ just split cell accordingly.

The second reason is also the influence of the parameter set , which influence the result of the standard VQ, in this process ,the movement direction may unable to generate a more uniform histogram.

Problem 3: JPEG Compression Standard

Part A: Discrete Cosine Transform (DCT) and Quantization for JPEG

3.1.1 Abstract and Motivation

DCT is widely used in signal processing, including audio compression technique MP3 and image compression technique JPEG. It is used in JPEG for blocked based quantization to preserve main information and discard detail information in an image. In this section, I would implement DCT and discuss its quantization performance in JPEG.

3.1.2 Approach and Procedures

At first, we read the given block image "lena.raw" data and subtract each pixel by128. There should be 4 four 8x8 blocks, and for each block, we use DCT transformation. Use the equations below to get the DCT coefficient matrix.

(6)

After getting the DCT matrix, we can obtain the quantized matrix by dividing DCT matrix with quantization matrix. Then we can use the quantized matrix for quantization in JPEG.
Then generate the new quantization matrix for factor=10 and factor=90 using equation below.

(7)

3.1.3 Result

(A) Quantized Matrix for quantization matrix Q_{50}

-1	-2	5	1	1	-2	-1	0	6	12	2	-2	0	0	0	0
5	-3	4	1	0	0	0	0	-3	1	-8	6	2	0	0	0
-2	6	-2	1	0	0	0	0	-2	-6	0	1	-2	1	0	0
1	0	-1	1	0	0	0	0	4	-3	-2	-3	0	0	-1	0
0	0	0	-1	0	0	0	0	2	1	1	0	0	0	0	0
-1	0	0	0	0	0	0	0	-1	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-12	6	7	-3	-1	-3	-1	-1	6	0	8	1	-3	0	0	0
3	-1	-2	-3	0	0	0	0	-1	-8	9	2	0	-1	0	0
1	-2	4	-1	0	1	0	0	-2	5	-3	-2	0	1	1	-1
-1	1	0	0	0	0	0	0	0	2	-1	0	-1	0	0	0
0	1	0	0	0	0	0	0	-2	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure15 Quantized Matrix for quantization matrix Q_{50}

(B) Quantization Matrix Q_{10} AND Quantized Matrix

80	55	50	80	120	200	255	305
60	60	70	95	130	290	300	275
70	65	80	120	200	285	345	280
70	85	110	145	255	435	400	310
90	110	185	280	340	545	515	385
120	175	275	320	405	520	565	460
245	320	390	435	515	605	600	505
360	460	475	490	560	500	515	495

Figure16 Quantization matrix Q_{10}

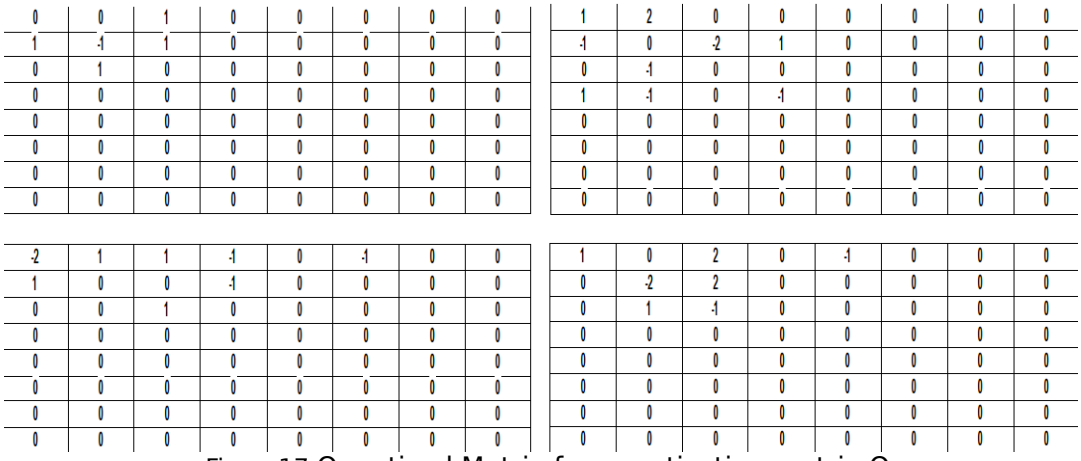


Figure17 Quantized Matrix for quantization matrix Q_{10}

(C) Quantization Matrix Q_{90} AND Quantized Matrix

3	2	2	3	4	8	10	12
2	2	2	3	5	11	12	11
2	2	3	4	8	11	13	11
2	3	4	5	10	17	16	12
3	4	7	11	13	21	20	15
4	7	11	12	16	20	22	18
9	12	15	17	20	24	24	20
14	18	19	19	22	20	20	19

Figure18 Quantization matrix Q_{10}

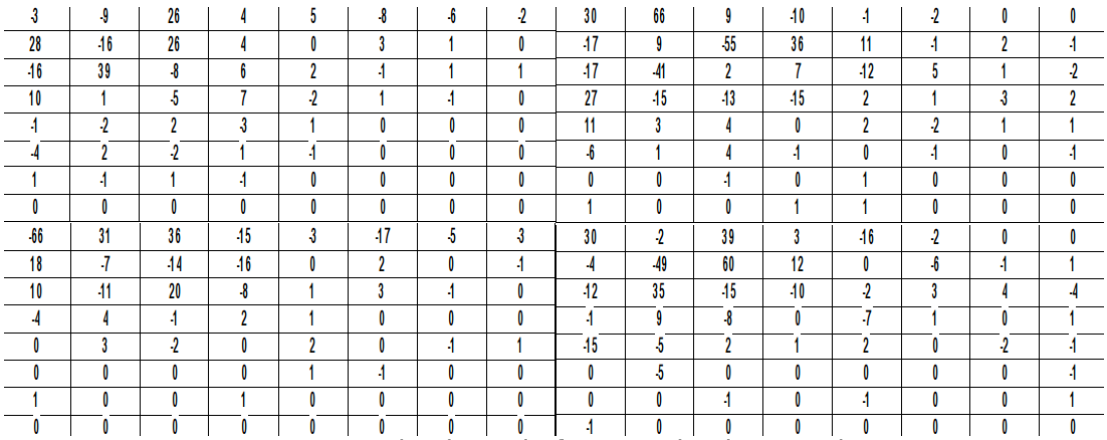


Figure19 Quantized Matrix for quantization matrix Q_{90}

3.1.4 Discussion

(A) Nonlinear quantization

The DCT and Quaninization are both similar to the low pass filter which remove the symbol of high frequency away, which is reasonable for human is not sensible for these symbol. From the table , we can see for each block, the bigger values are concentrated on the left -up ,while the minor values are located at the right-down area. We use nonlinear quantization that remove

more high frequency value while keep more of the low frequency value.

(B) Comparison of different factor

Compare three quantized matrices, the content of information in Quantized Matrix for quantization matrix Q90 is the biggest, while only little information remain for Quantized Matrix for quantization matrix Q10.

Thus bigger factor is , the more value (symbol) were removed and the worse image quality is . But from the aspect of entropy coding, in the Quantized Matrix for quantization matrix Q10, there are lots of successive 0s ,which is the good condition for entropy coding The bigger factor brings more '0' and the redundancy of the symbol stream which allow us to get bigger compression ratio for entropy coding. So the factor=10 get the worst image quality but best suited for entropy coding.

Part B: JPEG Compression Quality Factor

3.2.1 Abstract and Motivation

JPEG uses a lossy form of compression based on the discrete cosine transform (DCT). This mathematical operation converts each frame/field of the video source from the spatial (2D) domain into the frequency domain (a.k.a. transform domain.) A perceptual model based loosely on the human psychovisual system discards high-frequency information, i.e. sharp transitions in intensity, and color hue. In the transform domain, the process of reducing information is called quantization. In simpler terms, quantization is a method for optimally reducing a large number scale (with different occurrences of each number) into a smaller one, and the transform-domain is a convenient representation of the image because the high-frequency coefficients, which contribute less to the overall picture than other coefficients, are characteristically small-values with high compressibility. The quantized coefficients are then sequenced and losslessly packed into the output bitstream. Nearly all software implementations of JPEG permit user control over the compression-ratio (as well as other optional parameters), allowing the user to trade off picture-quality for smaller file size. In embedded applications (such as miniDV, which uses a similar DCT-compression scheme), the parameters are pre-selected and fixed for the application.

3.2.2 Approach and Procedures

Using the **jpeg_6b.zip,** for example, using the command **"**./cjpeg -quality 100 clock.bmp> clock_qua100.jpg**"** to get the jpg with factor=100**;**
Then using the command "./djpeg clock_qua100.jpg > clock_qua100.raw" to get the raw data and calculate the PSNR compared to the origin data.

3.2.3 Result

Table 8 clock.bmp Compression Result

Factor	100	60	40	20	10	1
Input_file size	66614	66614	66614	66614	66614	66614
Output_file size	37487	6795	5440	4045	3152	1847
compression ratio	56.27%	10.20%	8.17%	6.07%	4.73%	2.77%
PSNR	58.4932	35.8304	34.0552	31.5612	28.7632	18.7949

(a) original image

(b) factor 100

(c) factor 60

(d) factor 40 (e) factor 20 (f) factor 10

(g) factor 1

Figure20 The jpg of different factor

3.2.4 Discussion

Observing the quantization matrix , we find the value of quantization matrix with little factor is bigger. Considering we divide the DCT parameter with the value in the matrix, the bigger value in the matrix , we will lose more information during the division.

Meantime, we can see from the Quantized Matrix that bigger value concentrate on the left-up and in the right down area ,there are almost all zeros. Considering we do block DCT for the whole image , the information is losing by block ,the information we lose ,the more will see the effect of the block

Thus, By decreasing the quality factor, we are reducing high frequency components in the quantized matrix. The smaller the quality factor is, the more high frequency components are removed. When the quality factor is too small, we remove too many high frequency components in the quantized matrix and it starts to hurt the image quality. Human eyes can sense the loss of content when too many elements are zeros in the quantized matrix. From Figure 17, we can see that there are no obvious differences between original image, factor 100 and factor 60 images. This is because JPEG compression only removes the high frequency components which human eyes cannot differentiate at this point.

But When the factor is 40, we can fins that the background of the image is not as smooth as the images above. When the factors there exist the boundary of the blocks. Especially when it comes to 20 and 10, we can see obvious artificial blocks in the image. When the factor is 1, we can each block

one by one clearly. Considering we do the block DCT quantization , This is caused by the loss of too many components during the DCT quantization in JPEG.

Part C: Post-Processing of JPEG Encoded Images

3.3.1 Abstract and Motivation

During the jpeg encoded process, to get better compression ratio, more information is lost. Considering the result and discussion above ,using low factor could improve the compression performance, while however, more information have been dropped during the quantization, which leads to bad image quality. One of the most common and important problem is the blocking artifact.

Post-processing aims to solve these problems and improve the quality of image. It is a process of enhancing image. In this section, we solve the main problem of blocking artifact. In which, I choose and implement three method to solve the problem.

3.3.2 Approach and Procedures

At first, I convert the ten image into raw dat. use the jpeg_6b.zip provided and remove the head_files .(also could transfer to bmp and then read the raw , but this way is easier.)

Example command: .

/djpeg clock1.jpg> clock_pro_1.raw

./djpeg pepper1.jpg> pepper_pro_1.raw

Then come to the three methods:

(A) My method

Since the blocking artifact happens along the block boundary, I use the blocks Gaussian filter to smooth the boundary. Firstly, do it vertically, find each vertical boundary, and do Gaussian low pass filter to the 2 pixel along the boundary for each row; then do it horizontally, find each horizontal boundary, and do Gaussian low pass filter to the 2 pixel along the boundary for each column. For the value outside the image area, I set the corresponding boundary value as their value.

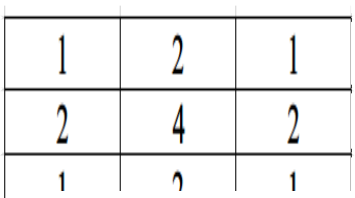


Figure21Gaussian block

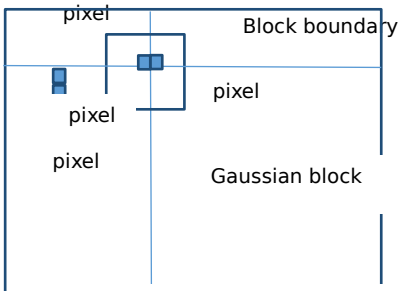


Figure22 Detail for the implement ion

(B)Deblocking_filter method

The key idea of the Deblocking_filter method is the vector using and the classification of smooth regions and complex regions. Using different methods to solve the different kind of region.

It also based on the boundary, but rather than using block area, it choose a linear direction vector as the object. It make decision based on the difference (change) between each 2 successive value, if it less than threshold T1, F(v)++, after calculate the F(v), compare it to the threshold T2, if it larger than T2 and the biggest difference along the vector is less than the twice of corresponding quantization value Qe, it take it as smooth region ,else if F(v)<T2,take it as default region. Neither satisfied, unchanged the vale.

If the region is smooth, the smooth de-block filter is called. In this filter, we use a nine-tab

smooth filter to smooth the whole block.

If the region is default, the default de-block filter is called. In this mode, we smooth the two block boundaries only and the four-point DCT is used as the analysis tool to analyze the frequency

components in the block. The key is to modify the value of v4 and v5, which is along the boundary.

The proposed deblocking filter is applied for all of the block boundaries along the horizontal edges first, and then along the vertical edges. If a pixel value is changed by the previous filtering operation, the updated pixel value is used for the next filtering.

The detailed algorithm flow chart is as following, in the implementation, set c1=2, c2=5, c3=8, T1=2, T2=6 and QP using the corresponding position quantization value in Qe_{50}.

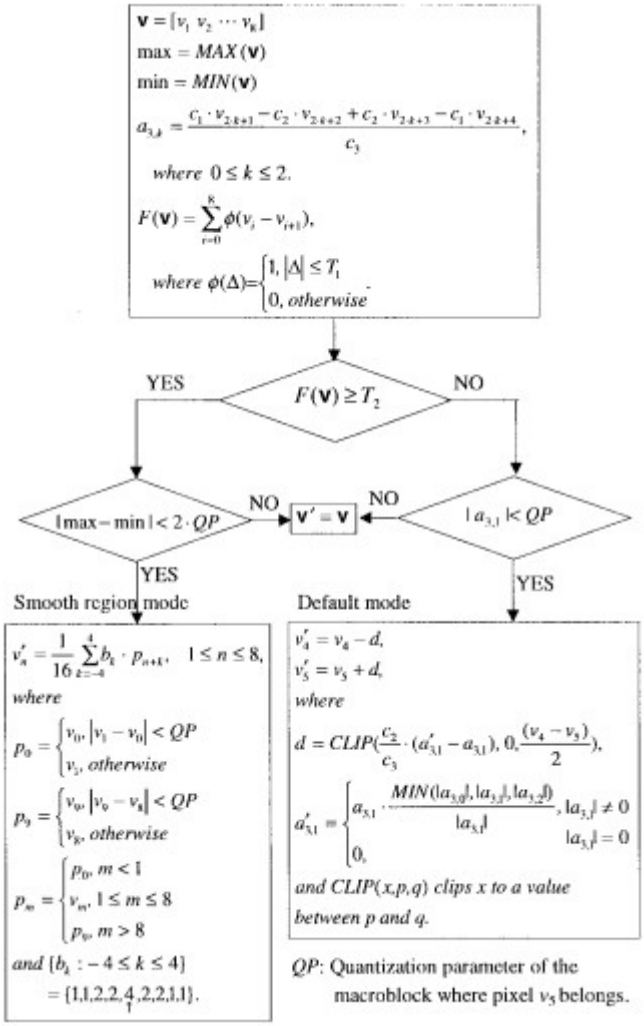


Figure 23 Deblocking Scheme

(C)Reapplying_JPEG method

For Reapplying_JPEG method , first, we should got the raw data of the image, then shift it in direction (i, j),meaning we change the pixel data with the corresponding data of the direction.

If the dirction is(vec,hem)

Converdata[i][j] =Imagedata_com[i][j-hem];

Converdata_2[i][j] =Converdata[i+vec][j];

If the data isrequired from the out of the image we just replicate the value let :

Converdata [i][j] =Converdata[i][j];

For vec and hem belongs to[-3,4] for each image there are 64 shifted images.

Then we do all 64 image JPED encoding and decoding ,which generate new 64 raw image data.

For each raw image ,we shift it back new_vec=-vec,new_hem=-hem;

After all data shifting back process finished, we add all raw data for each pixel together then divide it by 64,then the average data image is our results.

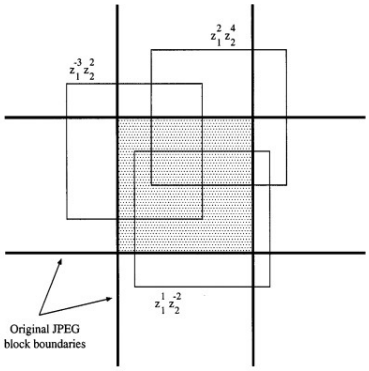


Figure 24 Direction vector explanation diagram

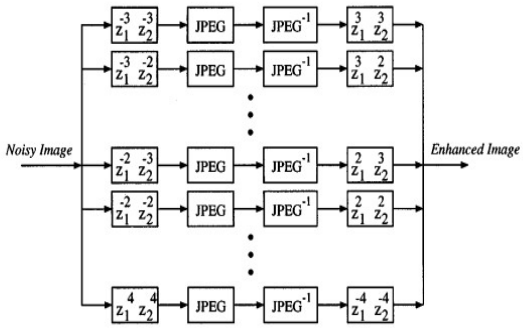


Figure 25 Reapplying_JPEG system

3.3.3 Results
(A) PSNR COPARISSON

Table 9 PSNR for clock.raw

File name	clock1	clock2	clock3	clock4	clock5
PNSR_Before	27.7701	29.5013	30.7266	32.2411	33.4376
PSNR_After(GU)	28.0888	29.4355	30.2448	31.1646	31.7664
Gain	0.3187	-0.0658	-0.4818	-1.0765	-1.6712

Table 10 PSNR for pepper.raw

File name	pepper1	pepper2	pepper3	pepper4	pepper5
PNSR_Before	23.6089	24.9313	25.8745	26.9761	27.7683
PSNR_After(GU)	23.9958	25.2352	26.1011	27.0578	27.7271
Gain	0.3869	0.3039	0.2266	0.0817	-0.0412

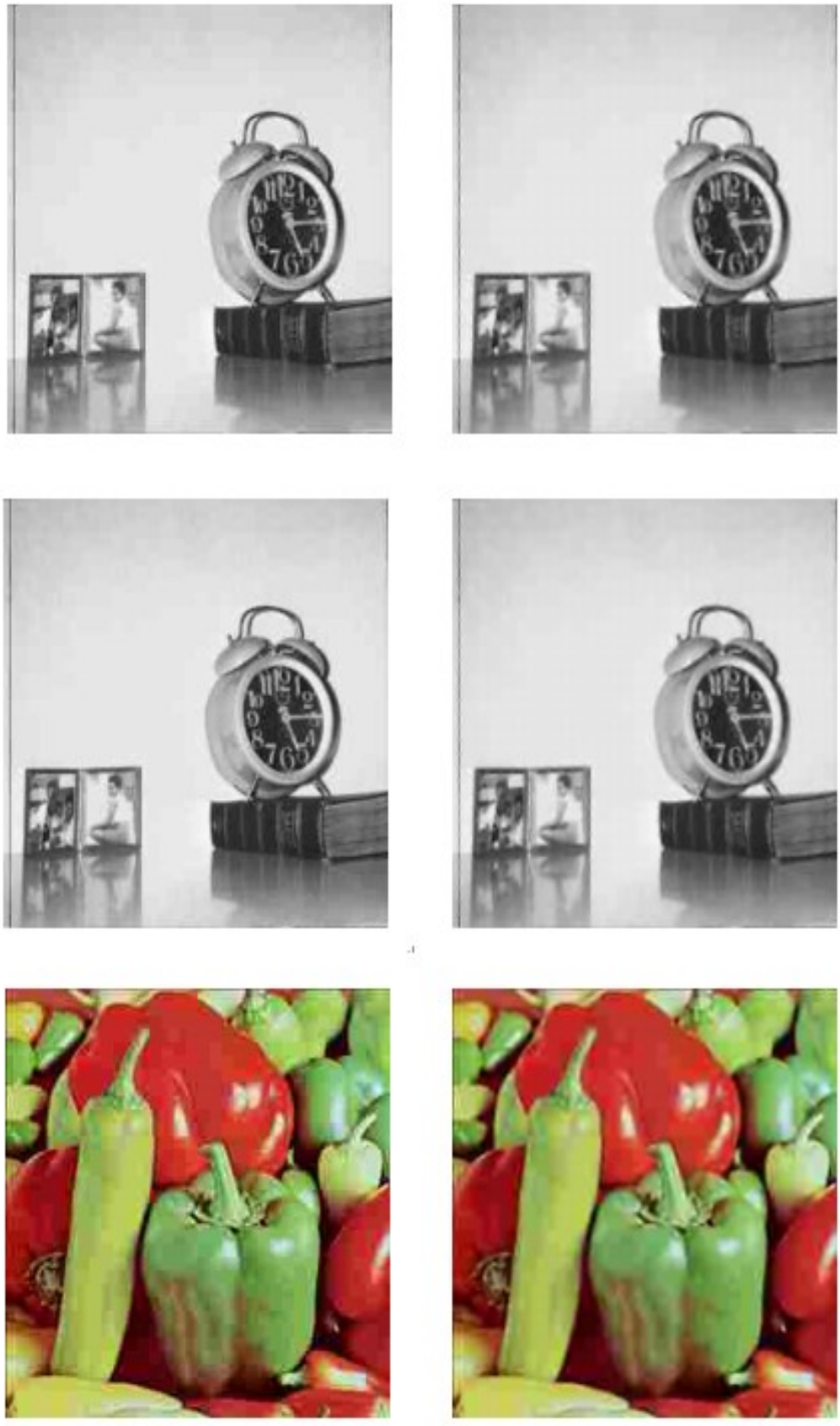


Figure 26 original image (left) and post_processing image (right)






(B) Deblocking filter

Table 11 PSNR for clock.raw

File name	clock1	clock2	clock3	clock4	clock5
PNSR_Before	27.7701	29.5013	30.7266	32.2411	33.4376
PSNR_DB	28.2731	29.9494	31.0638	32.4469	33.4329
Gain	0.503	0.4481	0.3372	0.2058	-0.0047

Table 12 PSNR for pepper.raw

File name	pepper1	pepper2	pepper3	pepper4	pepper5
PNSR_Before	23.6089	24.9313	25.8745	26.9761	27.7683
PSNR_DB	23.8974	25.1872	26.1112	27.1697	27.9291
Gain	0.2885	0.2559	0.2367	0.1936	0.1608

Figure 27 original image (left) and post_processing image (right)

(C) Reappling _JPEG method

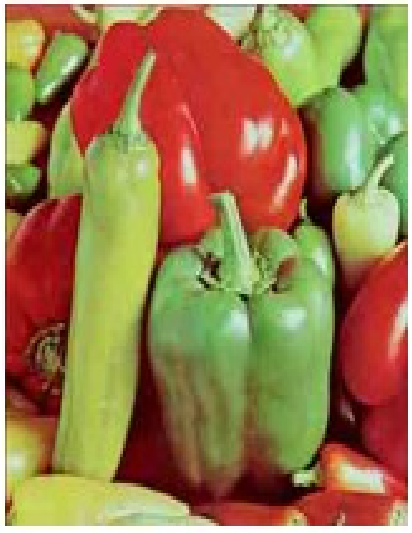
Table 13 PSNR for clock.raw

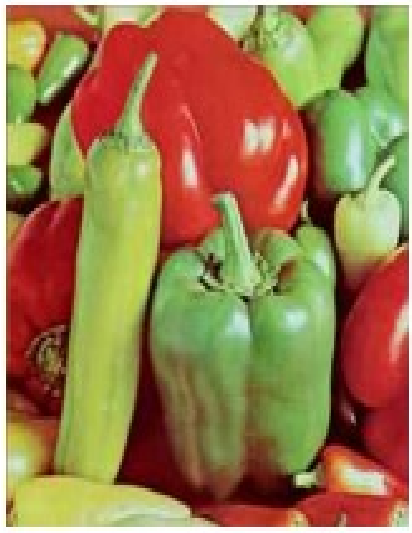
File name	clock1	clock2	clock3	clock4	clock5
PNSR_Before	27.7701	29.5013	30.7266	32.2411	33.4376
PSNR_RA	27.7701	29.5013	30.7266	32.2411	33.4376
PSNR_RA	28.0551	29.7344	31.0152	32.6226	33.3557
Gain	0.285	0.2331	0.2886	0.3815	-0.0819

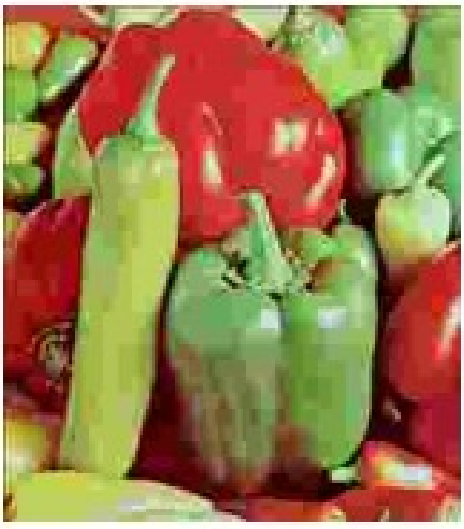
Table 14 PSNR for pepper.raw

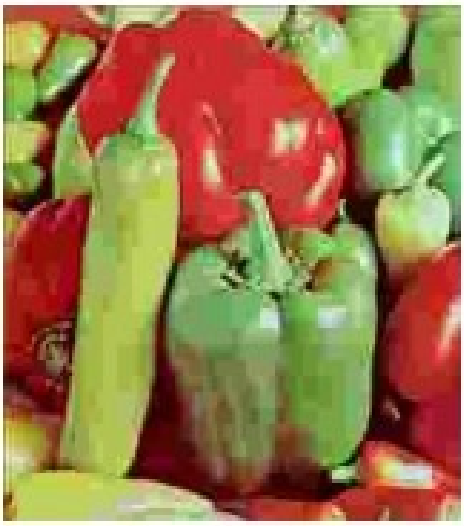
File name	pepper1	pepper2	pepper3	pepper4	pepper5
PNSR_Before	23.6089	24.9313	25.8745	26.9761	27.7683
PSNR_RA	23.6089	24.9313	25.8745	26.9761	27.7683
PSNR_RA	23.9776	25.2637	26.148	27.1349	27.7845
Gain	0.3687	0.3324	0.2735	0.1588	0.0162

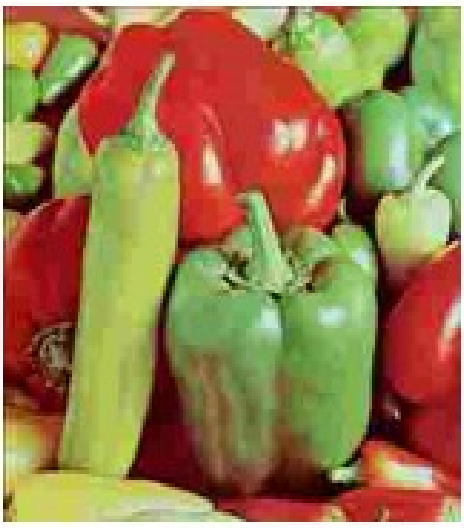










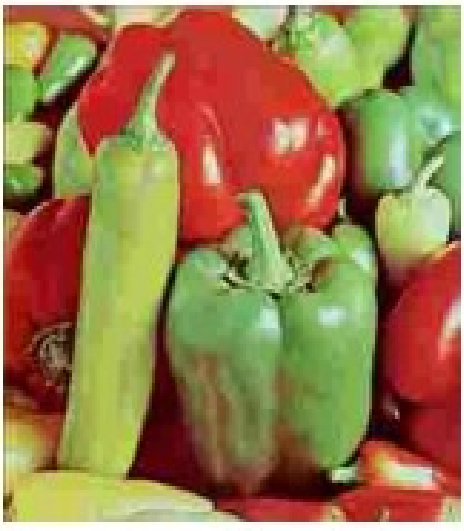


Figure 28 original image (left) and post_processing image (right)

3.4.4 Discussion

(A) General Discussion

From all figure above we can see clearly that we can see that the three method all have some improvement affect on the image. Although it cannot completely remove the artificial block, the blocks in the processed image is not as obvious as the ones in original images.

From the PSNR ,we can also see the improvement of the PSNR for most image, which also show the improvement of the methods.

However, there are some inconsistencies between the subjective quality and objective quality, especially for clock5, the objective quality: PSNR of processed image is smaller than PSNR of original image. But in subjective quality, we can clearly see that processed image has less artificial blocks than the original one. Comparison shown below. The reason for it could be although proproscessing such as low frequency filter , may drop or change some imformation , but humanity is not sensitive to these change comparing to the artificial blocks. In fact, if we see the value of each pixel, we can find that the data is more smooth than original even not in the block boundary. But from the perspect of global image we find it unnoticeable.

This inconsistency also indicates the fact that PSNR may not be a good indicator for image quality assessment. It only compares the differences between two images but doesn't consider human visual systems and pixels correlation.

Figure 29 original image (left) and post_processing image (right)

(B) Comparison of Deblocking method and Reapplying method

Compare from the subjective aspect ,we can see the Reapplying method get better result than the Deblocking method; From the PSNR ,we can see for clock image , Reapplying method have less PSNR gain compared to the Deblocking method,which may imply it kind of sacrificing the original image data during the shift and re-shift in the non_boundary area;

Besides ,the he computational complexity of Reapplying method easier to implement than the Deblocking approaches, and the computational complexity is also smaller.

Figure 30 Deblocking image (left) and Reapplying image (right)