JPEG Family: Near-Lossless Compression Method Comparsion Study

Pengzhan Hao
phao3@binghamton.edu

December 18, 2016

Abstract

The Joint Photographic Experts Group (JPEG) is a well-known committee which owns lots of image standards. Popluar lossy image compression standard JPEG is first work of them. After JPEG format became more popluar, JPEG committee released serveral new image compression method and corresponding image format, including JPEG2000, JPEG-LS, JPEG XT and etc. In my course project, I will concentrate two mainly used format, JPEG2000 and JPEG-LS, throughly state their technolgies and algorithms. Briefly introducation of my implemetation and comparsion these two algorithms and JPEG will be also covered in this report.

1 Introducation

The Joint Photographic Experts Group is the joint committee between ISO/IEC JTC1 and ITU-T (formerly CCITT) that created the JPEG, JPEG 2000, and JPEG XR standards[1]. Across all image formats, JPEG is one of the most important and most famous format. During our course lectures, we was introduced very well about how jpeg using discrete cosine transform and components discarding to achieve high compress rate. In my project, I will compare it with other formats of JPEG family to see how it real works and whether it fit to most common senarios under some predefined workload.

1.1 JPEG

JPEG is a commonly used method of lossy compression for digital images, particularly for those images produced by digital photography. The degree of compression can be adjusted, allowing a

selectable tradeoff between storage size and image quality. JPEG typically achieves 10:1 compression with little perceptible loss in image quality.[2] The JPEG compression algorithm has good performance on realistic scenes with smooth variations of tone and color. This algorithm is baesd on discrete consine transform, which is a mathematical opertaion transform spatial domain into the frequency domain. Discarding a large of information in color hue and intensity that can't be recognized by human perception. Discrete consine transform was desinged to be lossy and image quilaty would have a great probability to be affected. However, in JPEG standard, lossless compression is also supported in standard worklord. Aprat for strictly lossless, JPEG can also achieve a near lossless level if we apply transform and coding carefully.

1.2 JPEG2000

JPEG 2000 (JP2) is an image compression standard and coding system. It was created by the Joint Photographic Experts Group committee in 2000 with the intention of superseding their original discrete cosine transform-based JPEG standard (created in 1992) with a newly designed, wavelet-based method.[3] JPEG2000's wavelet-based method can deal with variety length of input stream, which means compression process will no longer corp images into fix-size blocks. This progress creatively solved JPEG's block effect, an horrible user feeling especially in JEPG's correspoding video codec standard - MPEG. This new feature also provide a probablity of compressing images by region of interests. JPEG2000 also provide a brunch of new techniques for different working senarios. As of 2016, there were still few digital cameras that shoot photos in the JPEG-2000 (.JP2, .J2K) format, and support for reading these photos is still fairly limited in feature mobile phones as well as smartphones and tablet PCs running various operating systems.[3]

1.3 JPEG-LS

JPEG-LS was defined to address the need for effective lossless and near-lossless compression of continuous-tone still images. JPEG-LS is especially suited for low-complexity hardware implementations of very moderate complexity, while at the same time providing state-of-the-art lossless compression performance.[4] JPEG-LS was developed in 1998, as the first edtion of official version of lossless JPEG. JPEG-LS use some new concepts for compression, which provide some great experienment results for following image standards. In real world application, it doesn't have lots of influence as JPEG2000 and JPEG. In my project, I only will demonstrate the idea and provide some basic results for this part.

In following sections, I will generally talk about basic compress method in section 2. Both JPEG2000 and JPEG-LS will be covered in this part. My implementation of JPEG2000 will be introduced in the third section. After that, in section 4, I will talks about how my implementation works and how it compares with each other. With a short summary, I will basicly talk how to run my program and list my references and acknowledgement in rest of my report.

2 Compression Method

2.1 JPEG 2000

JPEG2000 include a standard work process, which composed by three parts: Discrete wavelet transform, quantization and coding. Except these stardard process stage, some other techniques such as region of interests coding, color decorreletion will not be covered in this section.

2.1.1 Discrete Wavelet Transform

Wavelet transform is one of the new analysis transform method. As a successer of fourier analysis transform, wavelet transform can dealing with different input window to have a dynamic float frequency domain which can focus on details on a serial signal. Adaptive on different length and frequency can be well apply on any input stream and provide good results than fourier transform. Discrete wavelet transform is an upgrade and special application of wavelet transform. To deal with discrete signals, DCT can be apply well and offer great seperation of scalable signals such as a frame of video of a static image. Haar discrete wavelet transform is the basic filter of wavelet transform and we can simply represent it as following equation:
Given two numbers a and b, we have the following discrete wavelet transform:

$$(a, b) \rightarrow ((a + b)/2, (a - b)/2)$$

We can easily apply this formula to any length of array:

$$\underbrace{(a_1, a_2, \ldots, a_{2k-1}a_{2k})}_{2k} \rightarrow \underbrace{(\frac{a_1 + a_2}{2}, \frac{a_3 + a_4}{2}, \ldots \frac{a_{2k-1} + a_{2k}}{2}, \frac{a_1 - a_2}{2}, \frac{a_3 - a_4}{2}, \ldots, \frac{a_{2k-1} + a_{2k}}{2})}_{2k}$$

Haar transform just has a different arguments of input stream, which has a $\sqrt{2}$ for all signals.[5] So if we write a matrix for haar discrete transform, it will looks like:

$$\tilde{W}_n\mathbf{I} = \begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \sqrt{2}/2 & \sqrt{2}/2 & & 0 & 0 \\ \vdots & & & & \ddots & & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \sqrt{2}/2 & \sqrt{2}/2 \\ -\sqrt{2}/2 & \sqrt{2}/2 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & -\sqrt{2}/2 & \sqrt{2}/2 & & 0 & 0 \\ \vdots & & & & \ddots & & \vdots \\ 0 & 0 & 0 & 0 & \cdots & -\sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ \vdots \\ v_{n-2} \\ v_{n-1} \\ v_n \end{bmatrix} = \mathbf{O}$$

Haar wavelet transform is most common used wavelet. We can consider this matrix or equation to a simple way as two filters. We define haar filter $h = (h_0, h_1) = (\sqrt{2}/2, \sqrt{2}/2)$. This filter is also called as lowpass filter,since it averages paris of numbers, it tends to reproduce two values that are simliar and send to 0 to numbers that are near opposites of each other. We also build the bottom half of the HWT a highpass filter. In this case, we can get $g = (g_0, g_1) = (-\sqrt{2}/2, \sqrt{2}/2)$ to show differences between nearby pixels.[5]

Though haar transform is easily to be used and I implemented it as my compression, JPEG2000 has its own standard in wavelet chosen. JPEG2000 pick biorthogonal CDF 5/3 wavelet for lossless compression and CDF 9/7 as lossy one. Compare with haar filters, CDF 5/3 wavelet filter looks like in table 1. If we using lifting-based filtering for the 5/3 analysis filter, we can make some change across

Table 1: CDF (Le Gall) 5/3 Analysis Filter Coefficients [6]

K	Lowpass Filter	Highpass Filter
0	6/8	1
±1	2/8	1/2
±2	-1/8	

these coefficients. By dealing with discrete domain, we can have such operation[6]:

$$O_{2n+1} = I_{2n+1} - \left\lfloor \frac{I_{2n} + I_{2n+2}}{2} \right\rfloor$$

$$O_{2n} = I_{2n} - \left\lfloor \frac{I_{2n-1} + I_{2n+1} + 2}{4} \right\rfloor^{1}$$

We can apply wavelet transform on any serial signals, and similary, it can also been used on 2D signals, just apply wavelet transform on both x axis and y axis. We can either choose do wavelet

[1] This equation is already been quantiaed.

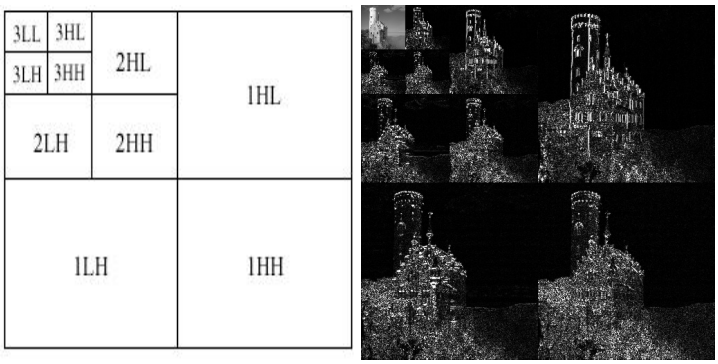

Figure 1: 3 Stage DWT [6]

transform on any region, which means we can compress our target image recusively until to a $2 * 2$ size block. According to JPEG2000 standard, it will only apply 3 stage transform on lowpass coefficients.[3]. Figure 1 shows the results of if we apply a 3 stage discrete wavelet transform on a picture.

2.1.2 Quantization

Quantization is one necessary process for lossy compression. JPEG2000 was designed for lossless compression at beginning, but added lossy compression support after. In my project, I concentrate on near lossless compression of JPEG family, which need this quantization stage of compression processing. Quantization basically designed to reduce distinct value of given stream. When the number of discrete symbols in a given stream is reduces, the stream becomes more compressible.

In JPEG standard, after doing discrete cosine transform, we have a list of $8 * 8$ blocks. So JPEG's quantization will use a pre-defined quantization matrix to divide these blocks in order to have a bias but less complex blocks.

In JPEG2000, there isn't a pre-defined block size which means static quantization matrix is not exist. So I take an arbitry design to decrease entropy of image stream. Basically, two steps will be taken. First, after discrete wavelet transform, most items of results will be a float number rather than an integer.If we assume using static length coding, in most of systems, a float number need at least 4 bytes to store. After observing data, we can figure out that after 3 stage DWT, the biggest number of whole matrix is about 1500 and minimum number is about -20. So we just need at least 11 bits to store each item. If we assume using variety length coding, we can decrease about $1/3$ of distinct values, which will provide approximately $4/7$ deducation of coding length. So we need to first arbitry transform all data from float type to integers. Second, after taking round opertaion, we can find a lot of consederately small value in first stage's LH, HL, HH coeffiences. These small differences only show slightly

different between two pixels of original image, but it takes more places to store them. Thus, we need the second step of quantization which is round more data close to 0.

2.1.3 Coding

JPEG2000 use arithmetic coder for low level coding operations. Embedded Block Coding with Optimal Truncation (EBCOT) is first step of coding. EBCOT is an implementation of compress by region of interests. It has some concepts of round to zero which has been covered in my quantization design. Basically, in this encoding process, each bit plane of the code block gets encoded in three so-called coding passes, first encoding bits (and signs) of insignificant coefficients with significant neighbors (i.e., with 1-bits in higher bit planes), then refinement bits of significant coefficients and finally coefficients without significant neighbors. The three passes are called Significance Propagation, Magnitude Refinement and Cleanup pass, respectively.[3] After EBCOT, bits will be encoded by a context-driven binary arithmetic coder, namely the binary MQ-coder. Unfortunately, MQ-coder and EBCOT are high computable uneffcience. In EBCOT, algorithm takes a great complexity. So, I didn't consider use EBCOT and MQ-coder as coding both in design and my own implementation. Instead of these techniques, I choose huffman coding, the one used in JPEG standard, as my coding system.

2.2 JPEG-LS

JPEG-LS use three important concepts which is prediction, context modealing and Golomb codes. For each pixel in an image, it will be gradients first, and them pass to fixed predicotr and adaptive correction, after prediction, context modeler will check is all predication is correctr, if not, then it will modify correction components, and start fixed predictor and adaptaive correction again. After finally converged, result will be pass to the golomb coder, which be compress in serial.
In LOCO-I JPEG-LS, fixed predictor is used following equation[11][12]:

$$\hat{x}_{MED} = \begin{cases} min(a,b) & if c \geq max(a,b) \\ max(a,b) & if c \leq min(a,b) \\ a+b-c & otherwise. \end{cases}$$

After prediction, context determination will decide whether predict error rate is below than threshold or equal to 0. There is a shiftable correction value which can be modify by process, after determination, correction value can be computed as the rounded average[11]:

$$C' = \lceil D/N \rceil$$

$$D = N * C' + B'$$

where IntegerB' satisfies $-N < B' \leq 0$. It is redily verified, by setting up a singple recursion. And the bias computing code looks like:

Algorithm 1 Bias Computation

```
1:  B = B + ε                                   ▷ accumulate prediction residual
2:  N = N + 1                                   ▷ update occurrence cunter
3:  if then B ≤ −N
4:      C = C − 1, B = B + N
5:      if then B ≤ −N
6:          B = −N + 1
7:      end if
8:  else if then B > 0
9:      C = C + 1, B = B − N
10:     if then B > 0
11:         B = 0
12:     end if
13: end if
```

After previewsly steps, result can be passed to golomb coder to be encoded. I won't generally talks about this part and all JPEG-LS's implementation, compare to JPEG2000, it was not my focusing.

3 Implementation

In my project, I follow my design of section 3 to implement my work. To make my work content clearly, I draw everything on the process in Figure 2. Exactly as figure 2 shows, bmp files will be read into work process, first transformed to a YUV color domain. For Y, U, V channels, it applies wavelet transform to each of them. After quantization and encoding, estimated file size can be output and image will be write into a jp2 format file. Then, after decoding, inverse wavelet transform and inverse color coversion, program will output a bmp file. Then comparsion utlity will judge differential of original and target bmp file to infrustrate percentage of error.

3.1 Haar Wavelet Transform

My implementation of haar wavelet transform is very simple. For each stage, program will judge start and end pixel, and apply 2d DWT on target region.

After Haar wavelet transform, three channel should look like Figure 3:

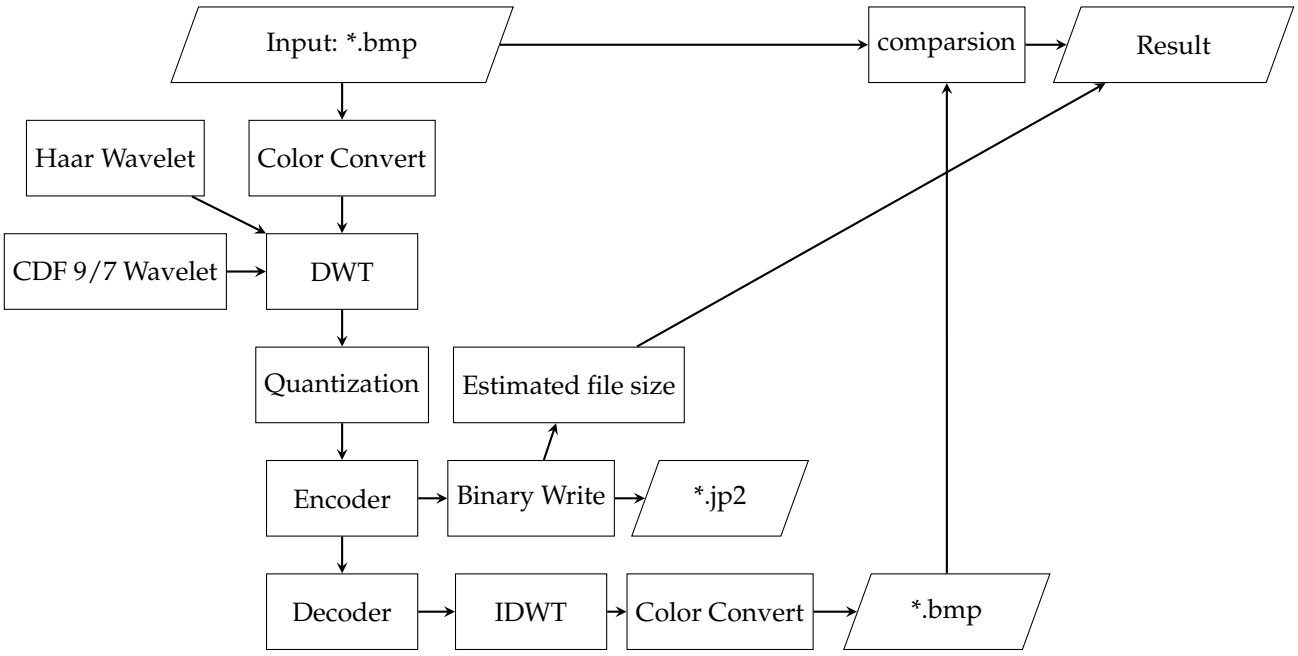

Figure 2: Process of JPEG2000 Implementation

3.2 CDF 9/7 Wavelet Transform

CDF 9/7 is implementation by article "how does JPEG 2000 work?".[7] It turns that it doesn't work well even under very casual test experiments. Attaching my code below, which is mainly about transform, to prevent result bias, we also need to do some optimation in the end of this function, but that part is not such important, so I will discard it.

3.3 Huffman Coding

Huffman coding is definately not the hardest part of this project, but I spent much more time on this part. Huffman coding is based on translate table and variety length content. I used the most common way to first build huffman tree according to pixel value appearances. After that, I will translate pixels by each and send them to a static length buffer. Everytime the buffer is full, it will automatically write to the hard disk.

To prevent decoding is absolutely right, metadat such as height, width, super code and length of translate table will be first write into filesystem. Then huffman tree will be recusively write only by pairs of $< key, value >$. After translate table written to the file, translation and content write will start. If the buffer is still not full at end of last pixel, it will automatically find the last bit of sequence.

Algorithm 2 2D Haar Discrete Cosine Transform

```
1: function HAAR_DWT(Matrix, height, width)
2:     for h = 0 to height do
3:         for w = 0 to width do
4:             Matrix[h][w] = sqrt(2) * (Matrix[h][w*2] + Matrix[h][w*2+1])/2
5:             w = w + 2
6:         end for
7:         h = h + 1
8:     end for
9:     for w = 0 to width do
10:        for h = 0 to height do
11:            Matrix[h][w] = sqrt(2) * (Matrix[h*2][w] + Matrix[h*2+1][w])/2
12:            h = h + 2
13:        end for
14:        w = w + 1
15:    end for
16: end function
17:
18: function HAAR_DWT(Matrix, stage)
19:    h = Matrix.height
20:    w = Matrix.width
21:    for s = 0 to stage do
22:        HAAR_DWT(Matrix, h/pow(2,s), w/pow(2,s))
23:    end for
24: end function
```

Then it will fulfill the buffer by the other binary number.

When decoding the file, it will inverse do what previews encoding's steps. Python has very good support on dictionary lookup based on cpython translator, so both encoding and decoding can be done very fast.

3.4 File Organization

This project is organized in common development structure and use git as version control.

Under *img* directory, it has some final images of my work. All images except *basel*2.*bmp* is generated by program. Experiement images are not included. (P.s some expereiment iamges are really big and made my github crashed once, make me set head again! Just complaint a while, Prof. you won't see this sentences, I guess)

Under *src* directory, it has two sub directories. jp2 is source code of my JPEG2000 implementation and jls is R srcipt for JPEGLS.

Under *utils* direcotry, some utilities are included. *cmp.py* is used for comparsion error rate (loss rate). *trans.py* is a template for write a image preprossing program.

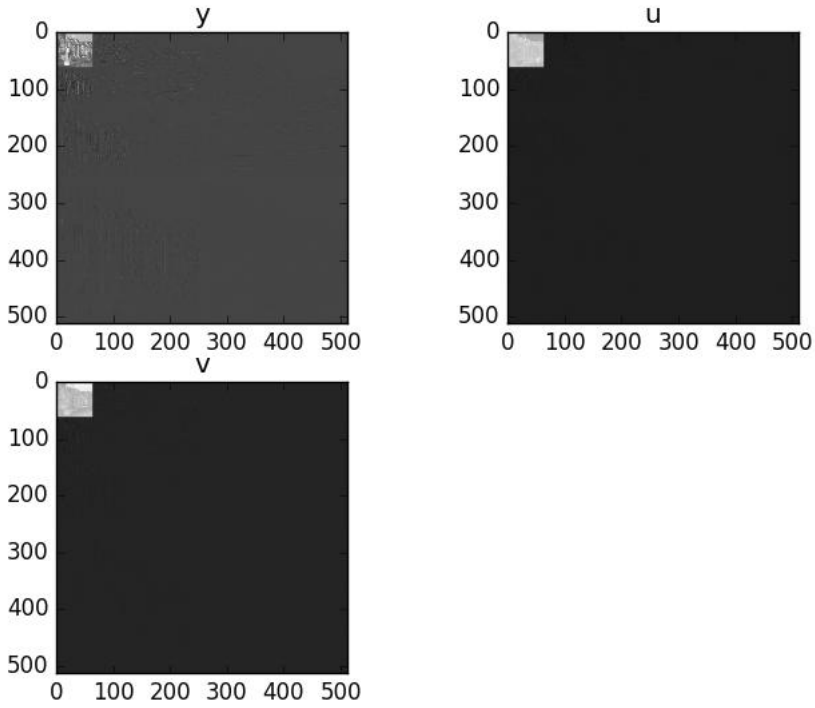

Figure 3: Y,U,V channels after DWT

4 Comparsion

4.1 Metrics

To measure image quality and compress rate, I pre-defined two metrics for each of them. Image quality is no doubt the most important measurement of this project. I compute the average differential between each componenets, which means for each color channel, the target image and original image should keep same.

$$Lossrate = \frac{\sum_{h=1}^{height} \sum_{w=1}^{width} \sum_{c=1}^{3} |Original[h][w][c] - Target[h][w][c]|}{height * width * 3}$$

After testing some images based on libjpeg and HPJPEG-LS library, I set 1 as a threshold of distinguish near-lossless and lossy compression. However, implementation might have some problem and correctness issue,so I set it as 1.5.

As for compression ratio, I used an estimation of encoded file as the standard. For huffman encoded

Algorithm 3 2D CDF 9/7 Discrete Cosine Transform

$Arraya = [-1.586, -0.053, 0.883, 0.444]$ ▷ 9/7 CDF Filter Coefficiences
function HAAR_DWT($Matrix, height, width$)
  for $h = 0$ to $height$ **do**
    for $w = 0$ to $width - 1$ **do**
      $Matrix[h][w] += a_1 * (Matrix[h][w-1] + Matrix[h][w+1])$
      $w = w + 2$
    end for
    for $w = 1$ to $width$ **do**
      $Matrix[h][w] += a_2 * (Matrix[h][w-1] + Matrix[h][w+1])$
      $w = w + 2$
    end for
    for $w = 0$ to $width - 1$ **do**
      $Matrix[h][w] += a_3 * (Matrix[h][w-1] + Matrix[h][w+1])$
      $w = w + 2$
    end for
    for $w = 1$ to $width$ **do**
      $Matrix[h][w] += a_4 * (Matrix[h][w-1] + Matrix[h][w+1])$
      $w = w + 2$
    end for
    $h = h + 1$
  end for
end function

file:

$$EstimatedFileSize = \frac{Metadata * 8 + SymbolTableLength * 4 * 2 + content}{8} Bytes$$

I didn't find any image dataset for compress test, to enforce my result, I use some crawlers download some random pictures with different keywords. Only images larger than $512 * 512$ can be accepted. After have a dataset with more than 300 images. I used *trans.py* write a automative script, crop all of them to $512 * 512$ and save as bmp format.

4.2 Image Quality

Based on pre-defined metrics, I gather data for all three type JPEG-family compression. Ploting of first 50 test data are in following figure 4. And in table 2, we can see all result of 300 images.

We can easily get some conclusion according to table 2 and figure 4. Obviously, JPEG was not designed for near-lossless compression, but it do have very great result even compared with JEPG2000 or JPEG-LS. JPEG2000 and JPEG-LS have samiliar results, especially in average error rate and Maximum Error Rate. But in fact, JPEG-LS was running in matlab, its core function is provided by HP Lab[8], and did have very stable performance than JPEG2000.

Also, JPEG2000 are better in computation time, I didn't notice that in a special section, but according to serveral times running timestamp, it has a great decreasing on running time compare with

JPEG-LS.

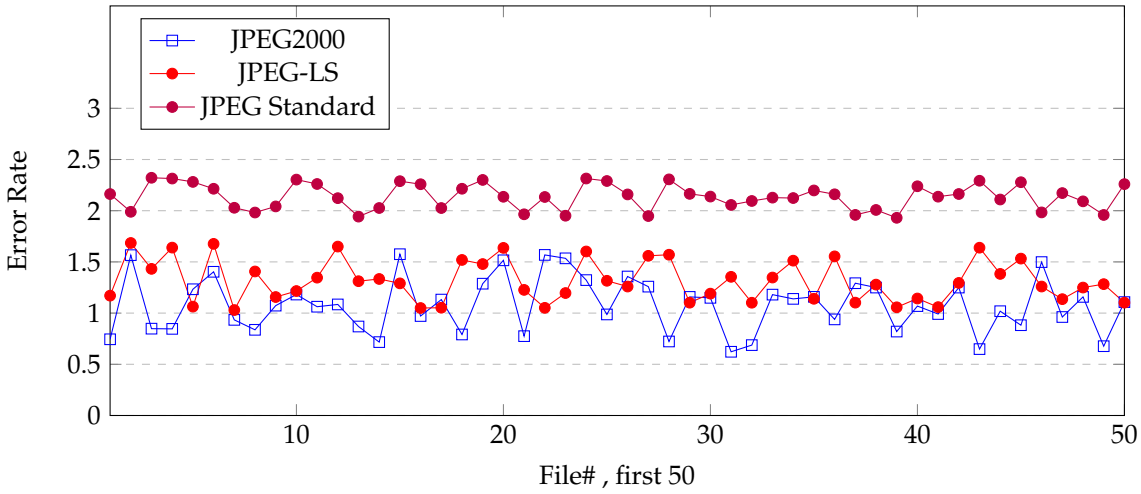

Figure 4: Image Quality Comparison: Acroos JPEG, JPEG2000 and JPEG-LS

Table 2: Summary of Quality Comparison

Compress Standard	JPEG	JPEG2000	JPEG-LS
Average Error Rate	2.312	1.310	1.329
Maximum Error Rate	2.797	2.255	2.319
Minimum Error Rate	1.813	0.399	0.810

4.3 Compression Ratio

Compression ratio is also an important metric for image compress method comparison. Due to some reason, we can't get exactly size of file. But as we demonstrate in previews section, we used a method to estimate approximate size of compressed image. Table 3 shows briefly results of all datasets, and figure 5 shows first 50 files' file size.

Table 3: Summary of Compression Rate Comparison

Compress Standard	JPEG	JPEG2000	JPEG-LS
Average Compress Ratio	19.43%	48.98%	53.44%
Best Compress Ratio	15.75%	42.41%	40.18%
Worst Compress Ratio	25.41%	61.62%	71.96%

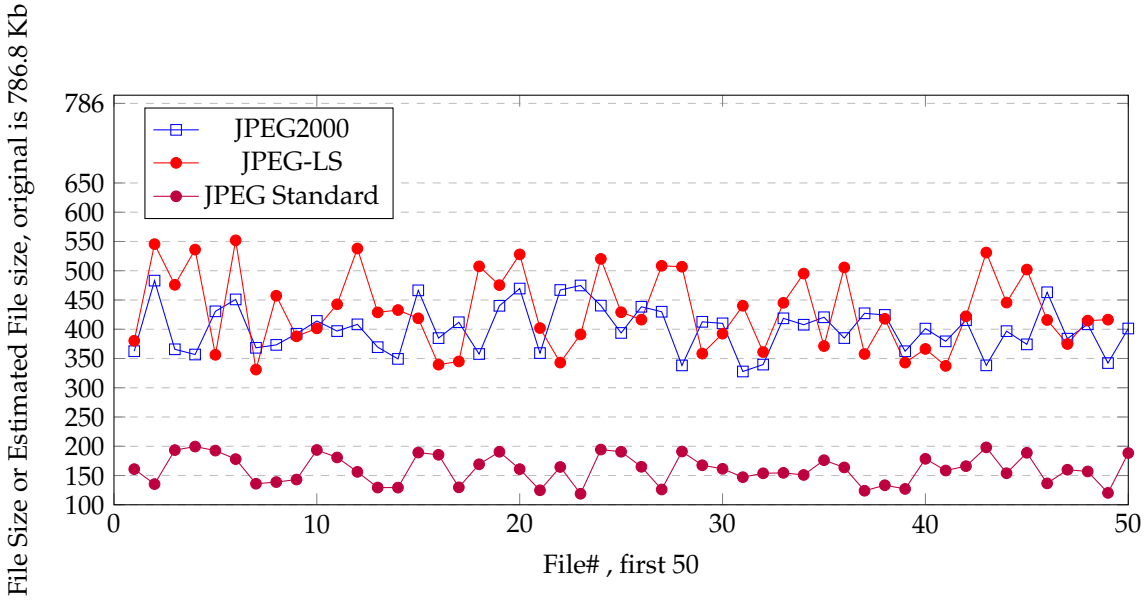

Figure 5: Image Compression Rate Comparison: Acroos JPEG, JPEG2000 and JPEG-LS

We can check figure 5 and table 3. It's obviously that JPEG standard library has a great compression rate. Across all files, it can compress under 20%, which is a great result. As for JPEG2000 and JPEG-LS, we can notice that these two show less differences, but JPEG2000 is slightly better than JPEG-LS. In fact, due to huffman coding is not original coding system for JPEG2000 and JPEG-LS, we can't say that these two compression method works bad on these images. Also, project implementation can also influent results. If we use standard *libjpeg* to transform this image sets to JPEG2000 format, we can get average size of 335.49 Kilobytes under lossless.

5 Summary and Future

In this report, I stated some concepts, algorithms and showed the results of final project. We can see that JPEG family is working step by step to provide high compression ratio method. From JPEG to JPEG2000, it did make a lot of progress on compression. JPEG2000 as mainly implenmentation of my project, some algorithms and techniques are still not been done until this last day. But results shows a lot acroos these three steps of JPEG family.

JPEG as always, shows robust and stable in compression. It has considerally larger error rate, but with its great performance on compress ratio, we can say that JPEG is still the best across these three methods. JPEG2000 and JPEG-LS recived similiar results both on compress rate and quality, but

JEPG2000 works a little better than JPEG-LS. As first generation of JPEG's lossless image standard, JPEG-LS didn't get popularity's fever. The successor JPEG2000's stanard library have much better optimization on JPEG-LS's disadvantages, but it still not that heat until now.

Though JPEG2000 can have lossless compress image with about 30% compression ratio, it still not able to beat PNG.[3] Also, Google's webp became more popluar with chrome's development. WebP has exceeded PNG and JPEG2000 both on lossy and lossless compression.[9][10] Some labratory standard even can achieve a considerable lower ratio than WebP.

Image compression is the base of image store and usage. Compare to image application, it more focus on low level, mathmatical operations. These works seems boring, but provide everything to a high class application in modern image processing research domain.

6 Acknowledgement

I would like to express my special thanks of gratitude to Prof. Lijun Yin who gave me the best lectures and opportunity to do this start-up project. I have to say that image processing is the best course I've taken in Binghamton University. Secondly, I would like to credit HP Labs, your homepage give me a direction for study JPEG-LS. I wouldn't finish this project without these knowledges. At last, I would like to thank to my friend, Pan Huo, who inspire me on vision and image processing. Honestly, I would not take this course without inspiration of your research at my last undergrad year. Best wishes to your research.

7 Appendix A: How to run the software

Note: Though every python file can be run, but only main.py and cmp.py matters.

```
cd src/jp2/
python main.py encode [input_file] [ouput_file]
cd ../../utils/
# input file is the original file, output_file is our target file for
    comparsion
python cmp.py [input_file] [output_file]
```

8 Appendix B: Bench Experiment Script to Image data sets

```
#!/bin/python
def list_and_operate(path=os.getcwd(), operator=[]):
```

```
    flist = [f for f in os.listdir(path) if os.path.isfile(os.path.
        join(path, f))]
    for i in flist:
        operator[0][3] = path+'/'+i
        operator[0][4] = path+'/'+i.split('.')[0]+'.new.bmp'
        bash = subprocess.Popen(operator[0], stdout=subprocess.PIPE,
            stderr=subprocess.PIPE)
        out, err = bash.communicate()
        print(out)
        operator[1][3] = path+'/'+i
        operator[1][4] = path+'/'+i.split('.')[0]+'.new.bmp'
        bash = subprocess.Popen(operator[1], stdout=subprocess.PIPE,
            stderr=subprocess.PIPE)
        out, err = bash.communicate()
        print(out)

operator = [
        ["python", "src/jp2/main.py", "encode", "", ""],
        ["python", "utils/cmp.py", "", ""]
]
```

References

[1] Wikipedia: Joint Photographic Experts Group Wiki,
https://en.wikipedia.org/wiki/Joint_Photographic_Experts_Group

[2] Wikipedia: JPEG Wiki, https://en.wikipedia.org/wiki/JPEG

[3] Wikipedia: JPEG 2000 Wiki, https://en.wikipedia.org/wiki/JPEG_2000

[4] JPEG: Overview of JPEG-LS, https://jpeg.org/jpegls/

[5] How Math Led to the JPEG2000 Standard:Haar Wavelet Transformation,
http://www.whydomath.org/node/wavlets/hwt.html

[6] C. Christopoulos, A. Skodras, and T. Ebrahimi. âĂIJThe JPEG2000 Still Image Coding: An Overview,âĂİ IEEE Transactions on Consumer Electronics, Vol. 46, No. 4, pp. 1103-1127, November 2000,
http://etro.vub.ac.be/čhchrist/paper_ieee_ce_jpeg2000_Nov2000.pdf

[7] Prof. Edward Aboufadel. âĂIJJPEG 2000: The Next Compression Standard using wavelet technologyâĂİ,
http://faculty.gvsu.edu/aboufade/web/wavelets/student_work/EF/how-works.html

[8] HP Labs, LOCO-I/JPEG-LS Home Page
http://www.labs.hp.com/research/info_theory/loco/

[9] Google, A new image format for the Web
https://developers.google.com/speed/webp/docs/compression

[10] Wikipedia, WebP Wiki
https://en.wikipedia.org/wiki/WebP

[11] M. Weinberger, G. Seroussi, G. Sapiro, "The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS", Hewlett-Packard Laboratories Technical Report No. HPL-98-193R1, November 1998, revised October 1999. IEEE Trans. Image Processing, Vol. 9, August 2000, pp.1309-1324.
http://www.hpl.hp.com/loco/HPL-98-193R1.pdf

[12] M. Weinberger, G. Seroussi, G. Sapiro, "LOCO-I: A Low Complexity, Context-Based, Lossless Image Compression Algorithm," Proc. IEEE Data Compression Conference, Snowbird, Utah, March-April 1996.
http://www.hpl.hp.com/loco/dcc96copy.pdf