Wei-Ming Liu
CSCI 576
Assignment 2 Extra Credit

In this assignment, I implemented some algorithms and compared the performance of them. First, let me explain the algorithms:

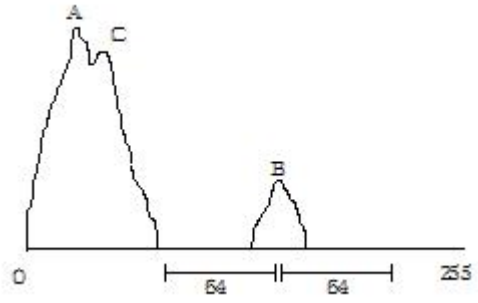
Algorithms

1. Uniform quantization:

When m = 1, it uses uniform quantization. First, it calculates how many bits needed to represent indexes of n vectors, and then divide the number of bits by 3. If the number is dividable by 3, each of R, G, B channel gets the same number of bits. Otherwise, the channel with higher deviation gets 1 more bit than other channel.

When x bits are assigned for a channel, the color space in that channel is separated into 2^xsections. The middle point of each section is chosen as a representative vector.

Finally the number of representative vectors may be bigger than n. So we have to eliminate some vectors. First eliminate the vectors which represent least pixels.



2. Peak Searching Algorithm:

My inspiration of this algorithm came from histogram. When we see a histogram of a picture, it usually looks like this. There are some “peaks”, which mean some pixels having similar color are clustering together. My algorithm is to find these peaks.

A high peak is not necessarily an important peak. For example, peak C is higher than B, but it’s less important than B if A has been selected as a representative vector. So, I think the “radius” of a peak is more important. When a peak has radius r, it means all the colors within the distance r away from the peak are lower than the peak. For example, peak B has radius more than 64, but less than 128.

We start by searching a peak with highest radius (256). When we find a peak, we select it as a representative vector. Then we reduce the radius and search for peaks with lower radius. Just repeat this until we have enough vectors.

The implementation of this algorithm uses a 3-D histogram, rather than this 1-D histogram. For each point in the color space, we construct a (2 * radius) x (2 * radius) x (2 * radius) cube, using the point as the center of this cube, and see if there exist any bigger point in this cube.

Choose m = 2 to use this algorithm.

3. Weighted Count Algorithm:

When we are deciding whether to choose a color as a representative vector, there are some factors to consider:

First factor is the number of pixels near this color. If there are many pixels near this color, it's good to choose this color as a representative vector because all these pixels can be represented by this vector and largely reducing error quantifier.

The other factor to consider is the distance between this color and currently selected vectors. If this color is far away from all the selected vectors, it's good to choose this color as a representative vector because all the pixels near this color are far away from currently selected vectors, it's better to have a vector close to them. This will also largely reduce error quantifier.

Therefore, we have to consider these two factors for each color in color space. It will take a lot of time to compute these two factors precisely. To save time, I simply use the number of pixels for a color to represent the number of pixels near this color, and use summation of distances in each channel to represent distance. When we have these two numbers, the problem is to find out a formula to calculate the weight of this color using these two numbers. (We count the number of pixels of a certain color and weight it by the distance to closest selected vector. This is why I call it "weighted count")

Choose m = 3 to use this algorithm.

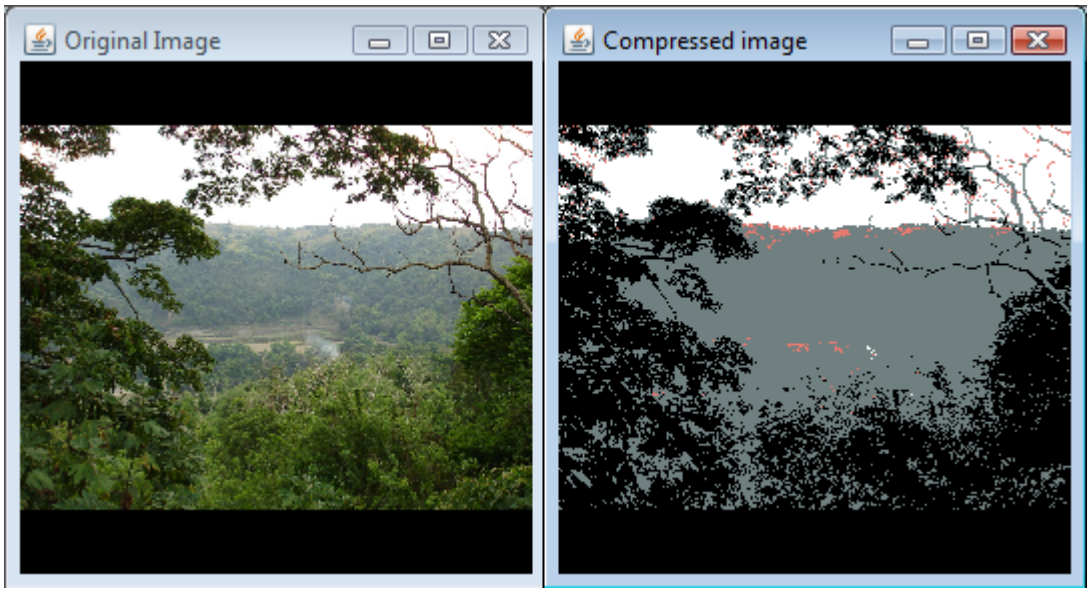
Problems and Solutions

1. Peak Searching Algorithm:

When the Peak Searching Algorithm is used, there are some problems.

When n = 4, it gave me this output. (See next page) I can hardly see any red pixels in the original image, why there is a peak in that area? Well, I think there exist some red pixels, which may be noise, scattered in the image. Actually, a single pixel can be a peak in color space, if there is no other pixel in its neighborhood.

To solve this problem, I used the anti-aliasing function in HW1 to process my image before the histogram was generated. This problem was solved because the noise was eliminated when doing anti-aliasing.



2. Weighted Count Algorithm:

I started by using the following formula:

weighted count of a color = the number of pixels * the distance to closest selected vector

Then, always choose the color which has biggest weighted count at the moment.

When n = 8, the picture looks like this:

In the original picture, blue pixels and black pixels are outnumbered other color pixels. This is why the output looks like this. Therefore, I put more weighted on distance, and used the following formula:

weighted count of a color = the number of pixels * (the distance to closest selected vector)^2

weighted count = count * distance^2

weighted count = count * distance^3

I tried several different formulas, and it turns out the following formula has best result:

weighted count of a color = the number of pixels * (the distance to closest selected vector)^4

This is the output image when n = 8

Result

Uniform algorithm on sample1

n = 4, r = 0, g = 1, b = 1 (bits)

n = 8, r = 1, g = 1, b = 1 (bits)

n = 16, r = 1, g = 1, b = 2 (bits)

n = 32, r = 1, g = 2, b = 2 (bits)

Peak searching algorithm on sample1

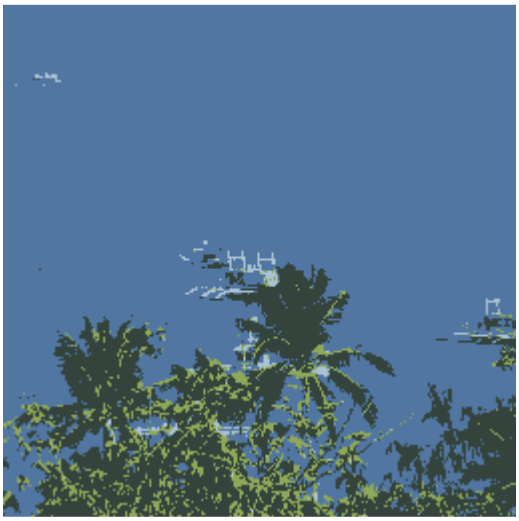
n = 4

n = 8

n = 16

n = 32

Weighted count algorithm on sample1



n = 4

n = 8

n = 16

n = 32

Comparison table:

Sample_1: 4 colors

Algorithm	R error quantifier	G error quantifier	B error quantifier	Overall Quantifier
Uniform	3223757	3167334	2227911	8619002
Peak Searching	594979	570968	709441	1875388
Weighted count	584491	661641	840798	2086930

Sample_1: 8 colors

Algorithm	R error quantifier	G error quantifier	B error quantifier	Overall Quantifier
Uniform	1230127	3167334	2227911	6625372
Peak Searching	555078	543343	708298	1806719
Weighted count	487639	545616	724583	1757838

Sample_1: 16 colors

Algorithm	R error quantifier	G error quantifier	B error quantifier	Overall Quantifier
Uniform	1230127	3167334	585071	4982532
Peak Searching	498494	484512	532466	1515472
Weighted count	417666	482185	637015	1536866

Sample_1: 32 colors

Algorithm	R error quantifier	G error quantifier	B error quantifier	Overall Quantifier
Uniform	1230127	1431812	585071	3247010
Peak Searching	410169	396640	484877	1291686
Weighted count	298804	345995	422459	1067258

Uniform algorithm on sample2

n = 4, r = 1, g = 0, b = 1 (bits)

n = 8, r = 1, g = 1, b = 1 (bits)

n = 16, r = 2, g = 1, b = 1 (bits)

n = 32, r = 2, g = 1, b = 2 (bits)

Peak searching algorithm on sample2

n = 4

n = 8

n = 16

n = 32

Weighted count algorithm on sample2

n = 4

n = 8

n = 16

n = 32

Comparison table:

Sample_2: 4 colors

Algorithm	R error quantifier	G error quantifier	B error quantifier	Overall Quantifier
Uniform	2015715	4107098	2184778	8307591
Peak Searching	1929401	1224899	2137311	5291611
Weighted count	1695859	1292043	2011386	4999288

Sample_2: 8 colors

Algorithm	R error quantifier	G error quantifier	B error quantifier	Overall Quantifier
Uniform	2015715	2040876	2184778	6241369
Peak Searching	1366298	906108	1319651	3592057
Weighted count	1310056	912553	1322819	3545428

Sample_2: 16 colors

Algorithm	R error quantifier	G error quantifier	B error quantifier	Overall Quantifier
Uniform	935175	2040876	2184778	5160829
Peak Searching	871759	652502	856462	2380723
Weighted count	769369	551486	739359	2060214

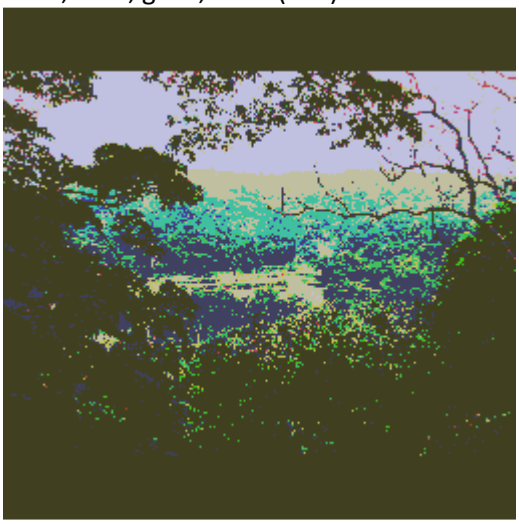
Sample_2: 32 colors

Algorithm	R error quantifier	G error quantifier	B error quantifier	Overall Quantifier
Uniform	935175	2040876	983286	3959337
Peak Searching	721483	529665	582801	1833949
Weighted count	610928	467693	537706	1616327

Uniform algorithm on sample3

n = 4, r = 1, g = 0, b = 1 (bits)

n = 8, r = 1, g = 1, b = 1 (bits)



n = 16, r = 1, g = 1, b = 2 (bits)

n = 32, r = 2, g = 1, b = 2 (bits)

Peak searching algorithm on sample3

n = 4

n = 8

n = 16

n = 32

Weighted count algorithm on sample3

n = 4

n = 8

n = 16

n = 32

Comparison table:

Sample_3: 4 colors

Algorithm	R error quantifier	G error quantifier	B error quantifier	Overall Quantifier
Uniform	2804201	5118298	3190075	11112574
Peak Searching	1214992	1437489	1149688	3802169
Weighted count	861702	920052	761975	2543729

Sample_3: 8 colors

Algorithm	R error quantifier	G error quantifier	B error quantifier	Overall Quantifier
Uniform	2804201	2803138	3190075	8797414
Peak Searching	595587	528686	694577	1818850
Weighted count	775658	833221	725274	2334153

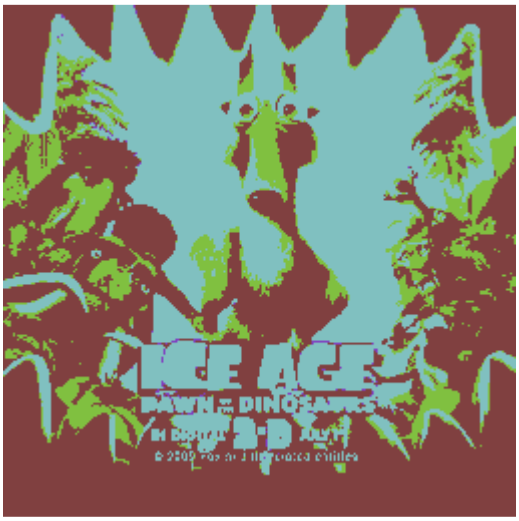
Sample_3: 16 colors

Algorithm	R error quantifier	G error quantifier	B error quantifier	Overall Quantifier
Uniform	2804201	2803138	1446017	7053356
Peak Searching	393014	383305	366454	1142773
Weighted count	527435	476825	482621	1486881

Sample_3: 32 colors

Algorithm	R error quantifier	G error quantifier	B error quantifier	Overall Quantifier
Uniform	1365355	2803138	1446017	5614510
Peak Searching	304383	296401	335549	936333
Weighted count	334898	352425	380130	1067453

Uniform algorithm on sample4



n = 4, r = 0, g = 1, b = 1 (bits)

n = 8, r = 1, g = 1, b = 1 (bits)

n = 16, r = 1, g = 1, b = 2 (bits)

n = 32, r = 2, g = 1, b = 2 (bits)

Peak searching algorithm on sample4

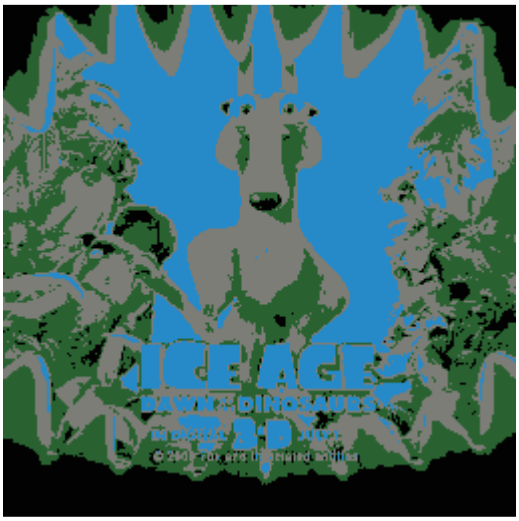
n = 4

n = 8

n = 16

n = 32

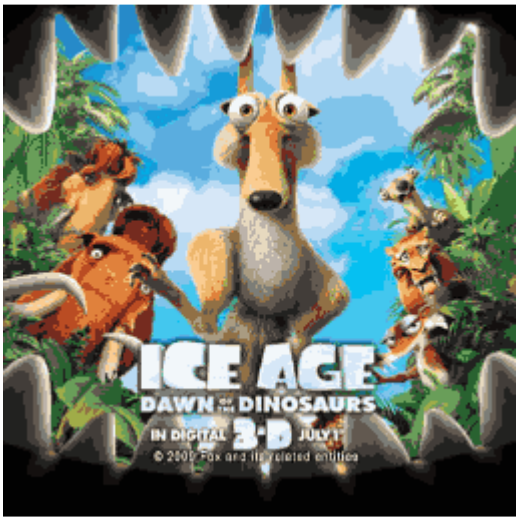
Weighted count algorithm on sample4



n = 4

n = 8

n = 16



n = 32

Comparison table:

Sample_4: 4 colors

Algorithm	R error quantifier	G error quantifier	B error quantifier	Overall Quantifier
Uniform	4182559	2015240	2221250	8419049
Peak Searching	3718549	2454249	1770695	7943493
Weighted count	3608880	2501430	1540189	7650499

Sample_4: 8 colors

Algorithm	R error quantifier	G error quantifier	B error quantifier	Overall Quantifier
Uniform	2099223	2015240	2221250	6335713
Peak Searching	3123936	1417017	1365034	5905987
Weighted count	1734010	1029067	1097659	3860736

Sample_4: 16 colors

Algorithm	R error quantifier	G error quantifier	B error quantifier	Overall Quantifier
Uniform	2099223	2015240	1089818	5204281
Peak Searching	1191695	1130149	732950	3054794
Weighted count	991793	851228	731651	2574672

Sample_4: 32 colors

Algorithm	R error quantifier	G error quantifier	B error quantifier	Overall Quantifier
Uniform	2099223	1120536	1089818	4309577
Peak Searching	868354	683452	622018	2173824
Weighted count	726510	552411	555506	1834427

Conclusion:

Both advanced algorithms are better than uniform algorithm in performance. In sample1 and sample3, these two advanced algorithms with n = 4 are better than uniform algorithm with n = 32. In sample2 and sample4, these two advanced algorithms are still better than uniform algorithm, but not that much.

The error quantifier of two advanced algorithms are very low in sample1 when n = 4. The deviation of color in this picture is low, so advanced algorithms can perform much better than uniform algorithm. Uniform algorithm wastes some vectors for the color which rarely exist in the picture.

Actually the error quantifier is not a perfect indicator of the quality of compressed pictures. When n = 4 in sample1, the error quantifier of peak searching algorithm is lower than weighted count algorithm. However, the visual quality of weighted count algorithm is better. (We can still see the boat) The reason is that peak searching algorithm chose black as a representative vector, and weighted count algorithm chose white as a representative vector. The black pixels in the image outnumber white pixels, so reducing quantization error of black pixels will contribute more to overall error quantifier.

The weighted count algorithm takes much more time to compute than peak searching algorithm. The reason is that I used a reduced histogram in peak searching algorithm. (It's a hierarchical search) (reduce 256x256x256 color space to 64x64x64) The weighted count algorithm may take less time if I implemented this method.