**Homework 3**

Wenyuan Shao

Problem 1:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Initial Values | d | 0 | 0.75 | 1.5 | 2.25 | 3 |
| Iteration 1 | r | 0.525 | 1 | 1.8825 | 2.466 |  |
| d | 0 | 0.7625 | 1.4413 | 2.1743 | 3 |
| Iteration 2 | r | 0.525 | 1 | 1.7767 | 2.39 |  |
| d | 0 | 0.7625 | 1.3883 | 2.0833 | 3 |
| Iteration 3 | r | 0.525 | 1 | 1.7767 | 2.39 |  |
| d | 0 | 0.7625 | 1.3883 | 2.0833 | 3 |

So, = 0; = 0.7625; = 1.3883; = 2.0833; = 3;

= 0.525; = 1; = 1.7767; = 2.39;

1. Quantized array = [0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3]

Dequantized array = [0.525, 0.525, 0.525, 0.525, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1.7767, 1.7767, 1.7767, 1.7767, 1.7767, 1.7767, 2.39, 2.39, 2.39, 2.39, 2.39, 2.39, 2.39]

MSE = 0.0357

1. Quantized array = [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2]

Dequantized array = [0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5]

MSE = 0.1041

1. Quantized array = [0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3]

Dequantized array = [0.525, 0.525, 0.525, 0.525, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1.8825, 1.8825, 1.8825, 1.8825, 1.8825, 1.8825, 1.8825, 1.8825, 2.4660, 2.4660, 2.4660, 2.4660, 2.4660]

MSE = 0.0419

1. The MSE of the question (d) is the smallest.

MSE(d) < MSE(b) < MSE(c)

Problem 2.

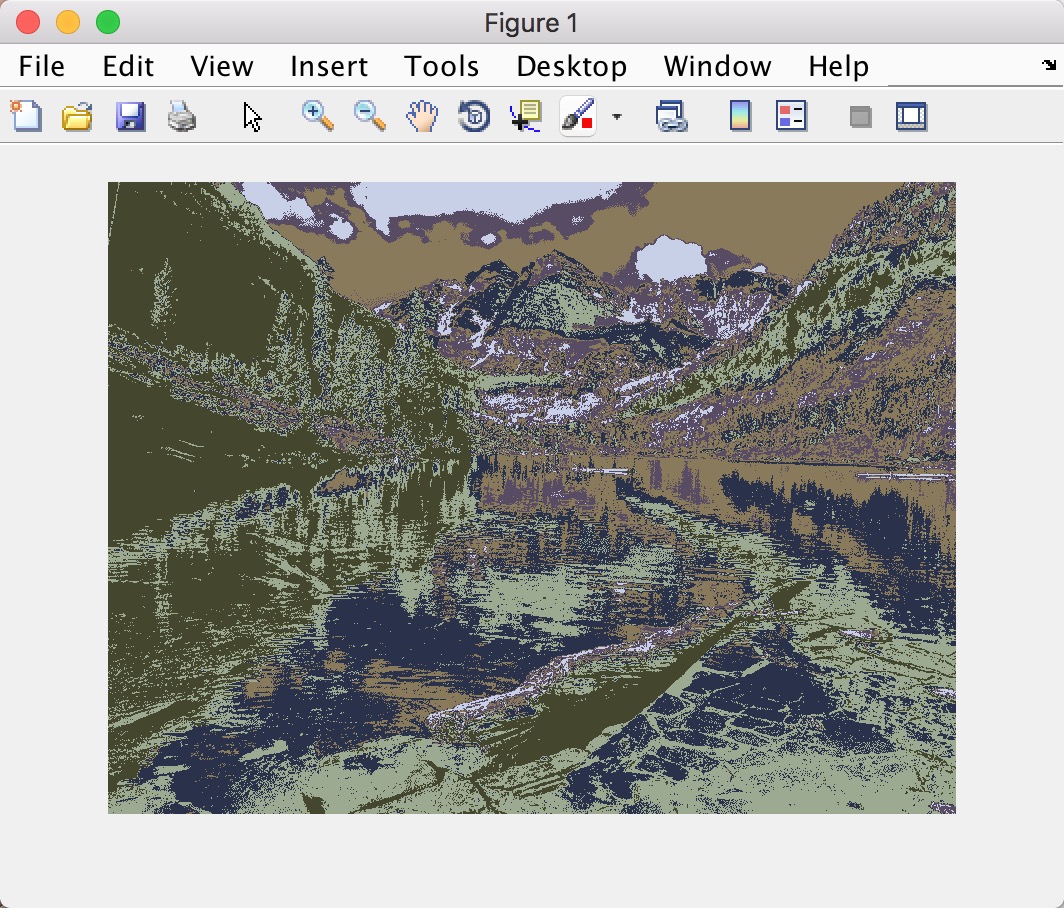
1. Enropy = 7.1776 (to the original picture)

GrayEntropy = 6.6177

1. Entropy of G’ = 0.7949 (the matrix type is double)

Entropy of G’ = 2.378872043027084 (the matrix type is uint8)

:



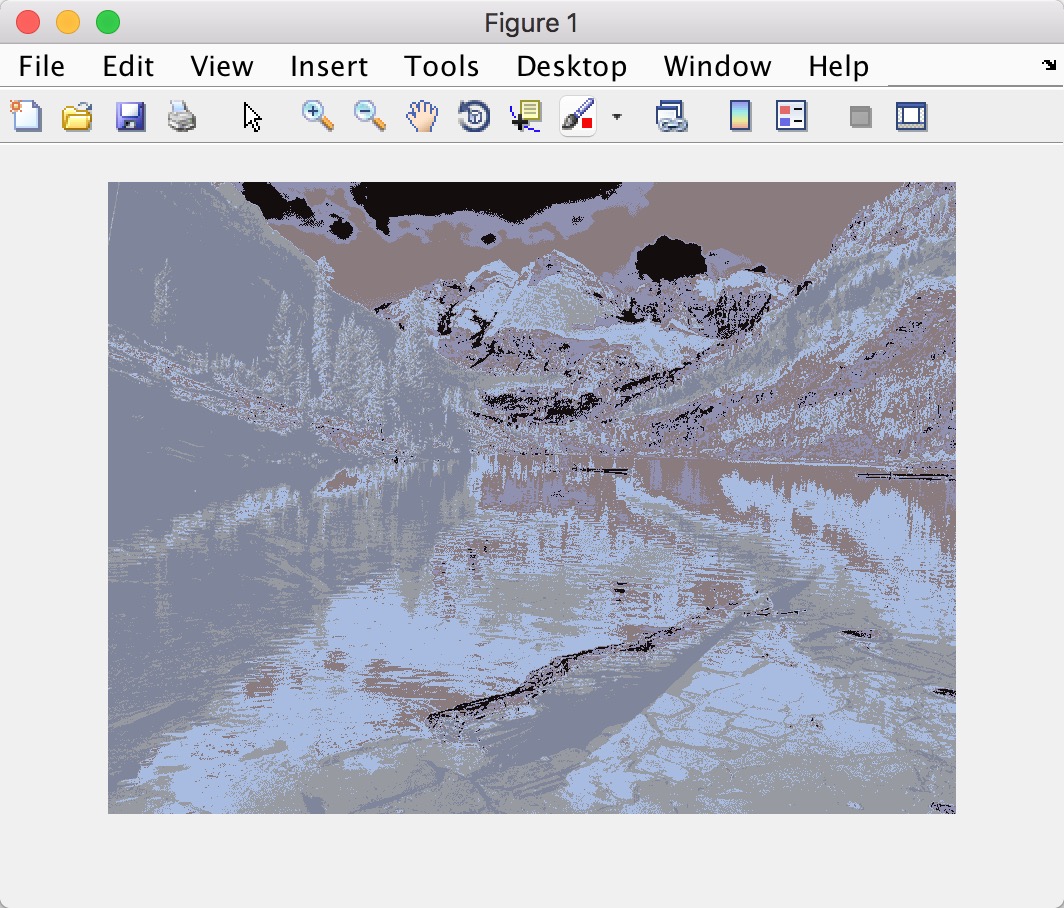
(use imshow() function to show the image)

SNR = 19.419938240237750

1. Entropy of G’ = 0.7949 (matrix type: double)

Entropy of G’ = 2.378872043027084 (matrix type: uint8)

:



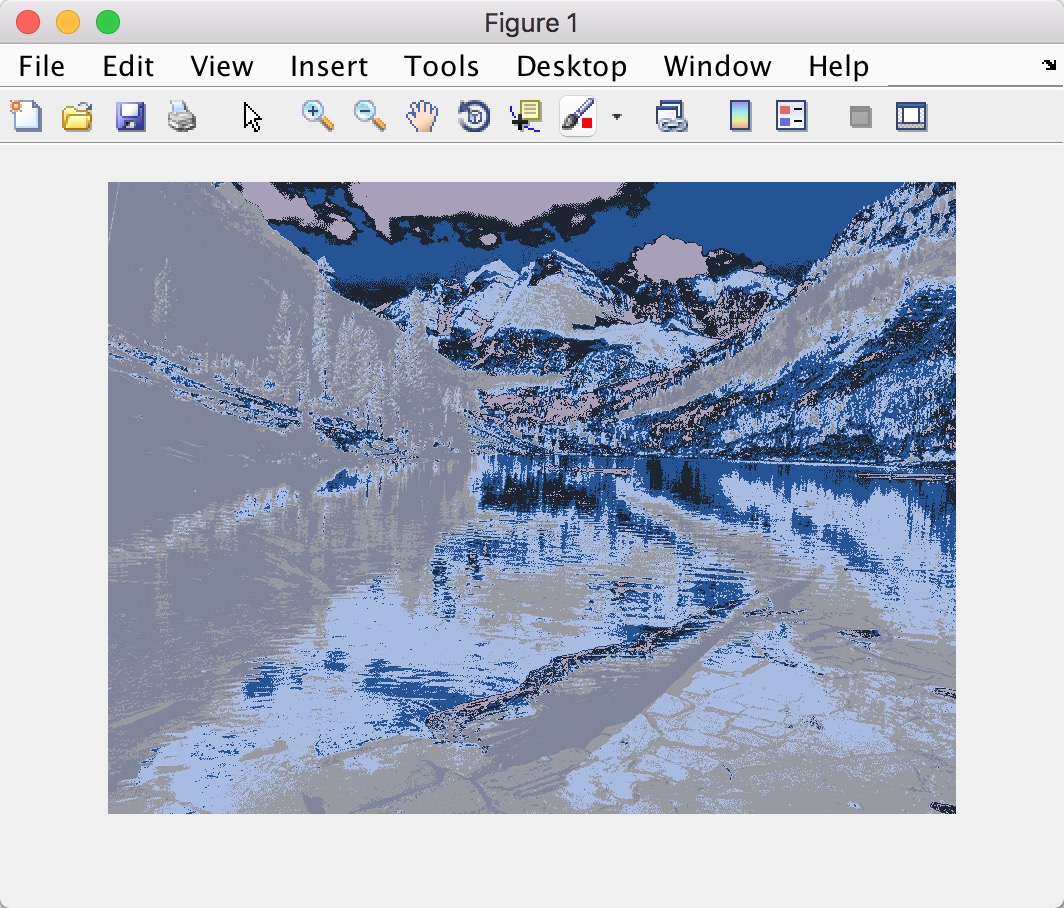
(use imshow() function to show the image)

SNR = 19.510872974663588

1. Entropy of G’ = 0.7886 (matrix type: double)

Entropy of G’ = 2.405483178698946 (matrix type: uint8)

:



(use imshow() function to show the image)

SNR = 19.566819125943390

1. Entropy(b) = Entropy(c) > Entropy(d) (for double)

Entropy(b) = Entropy(c) < Entropy(d) (for uint8)

I think this is caused by the different between types, double type will surely increase the number of samples which will result in bigger entropy.

SNR(b) < SNR(c) <SNR(d)

The picture of question(d) has the best quality.

Problem 3.

1. NO answer for question a.
2. E (1,0,0) = 0.9411 (double) 2.7191 (uint8)

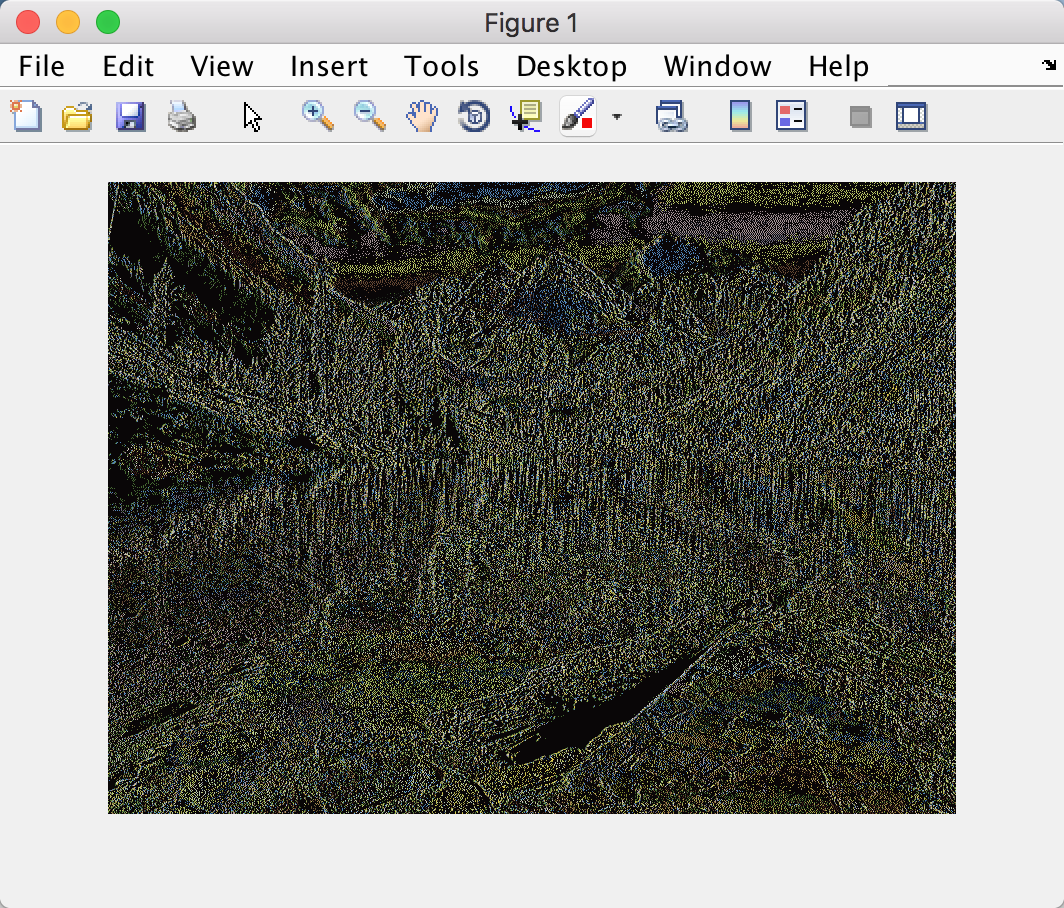
E (0,0,1) = 0.9433 (double) 2.8167 (uint8)

E (0.5,0,0.5) = 0.9780 (double) 2.9214 (uint8)

E (1,-1,1) = 0.9804 (double) 3.0793 (uint8)

E (0.75,-0.5,0.75) = 0.9868 (double) 3.0092(uint8)

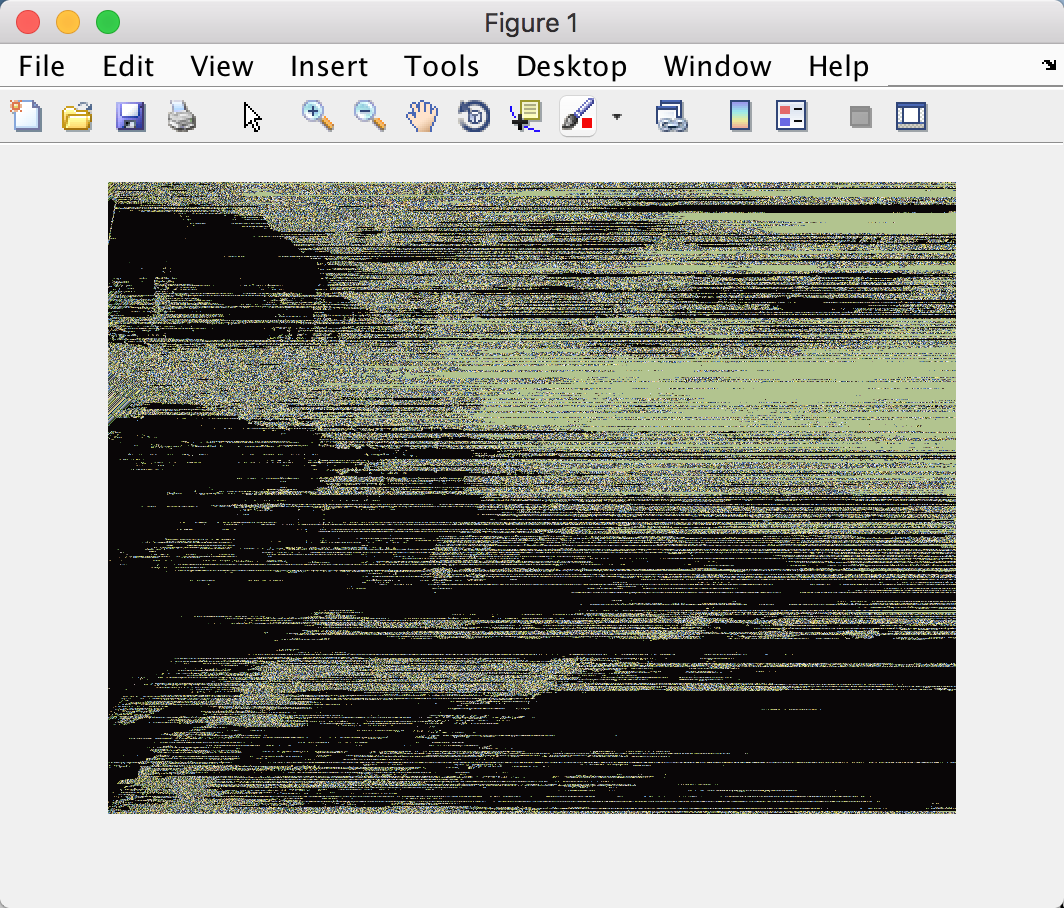
So if (a,b,c) = (1,0,0) the result has the smallest entropy. The image E is:



(use imshow() function to show the image)

1. Entropy = 0.098227050677435 (matrix type: double)

Entropy = 2.072230056288189 (matrix type: uint8)



(use imshow() function to show the image)

SNR = -4.093905948755968

1. The image has a bad quality. In my opinion, it is because the DPCM doesn’t have a good performance. To make the DPCM perform better, we could assume that R(I,j) has some relation with more points around it.

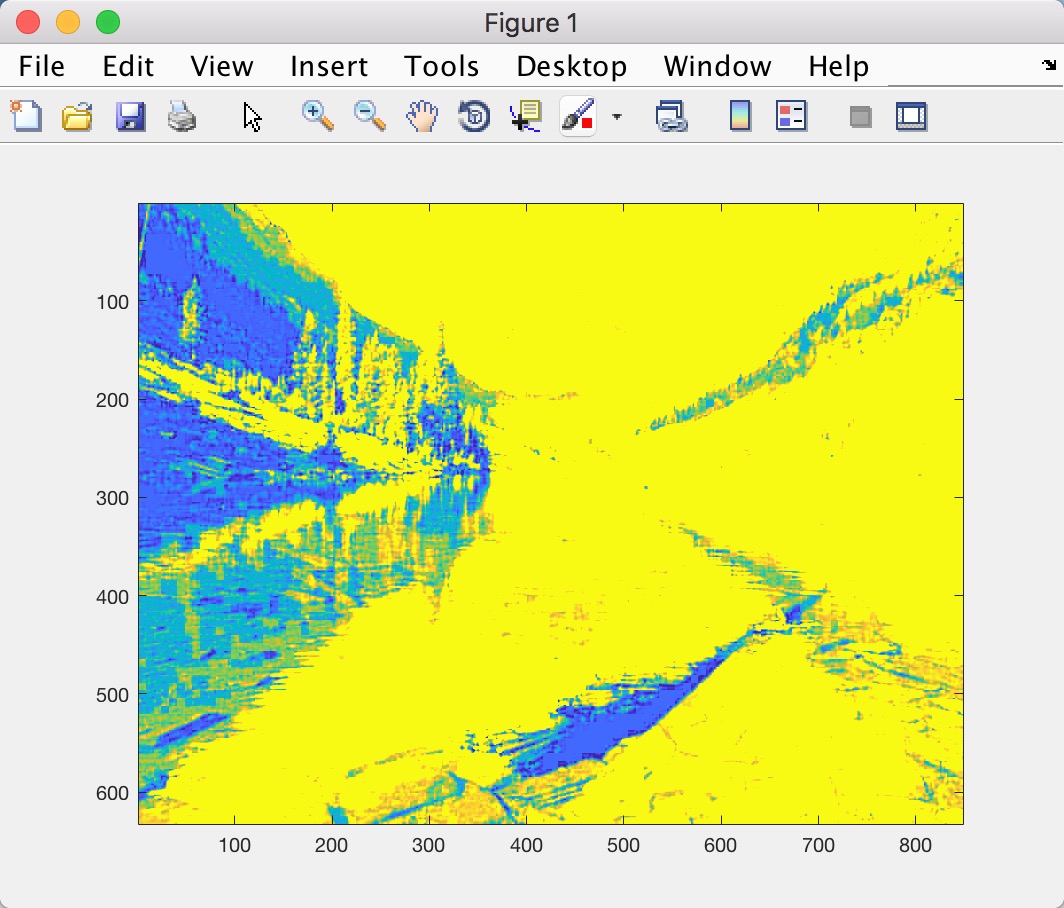
Problem 4.

1. As far as I am concerned, the question means we need to generate a unique quantizer for each 8\*8 block. In this way, the whole picture will surely has a better quality. However, it will be hard to transform all of the quantizers to the decoder.

The bitrate wil be: 0.022030098049306 (for double type matrix)

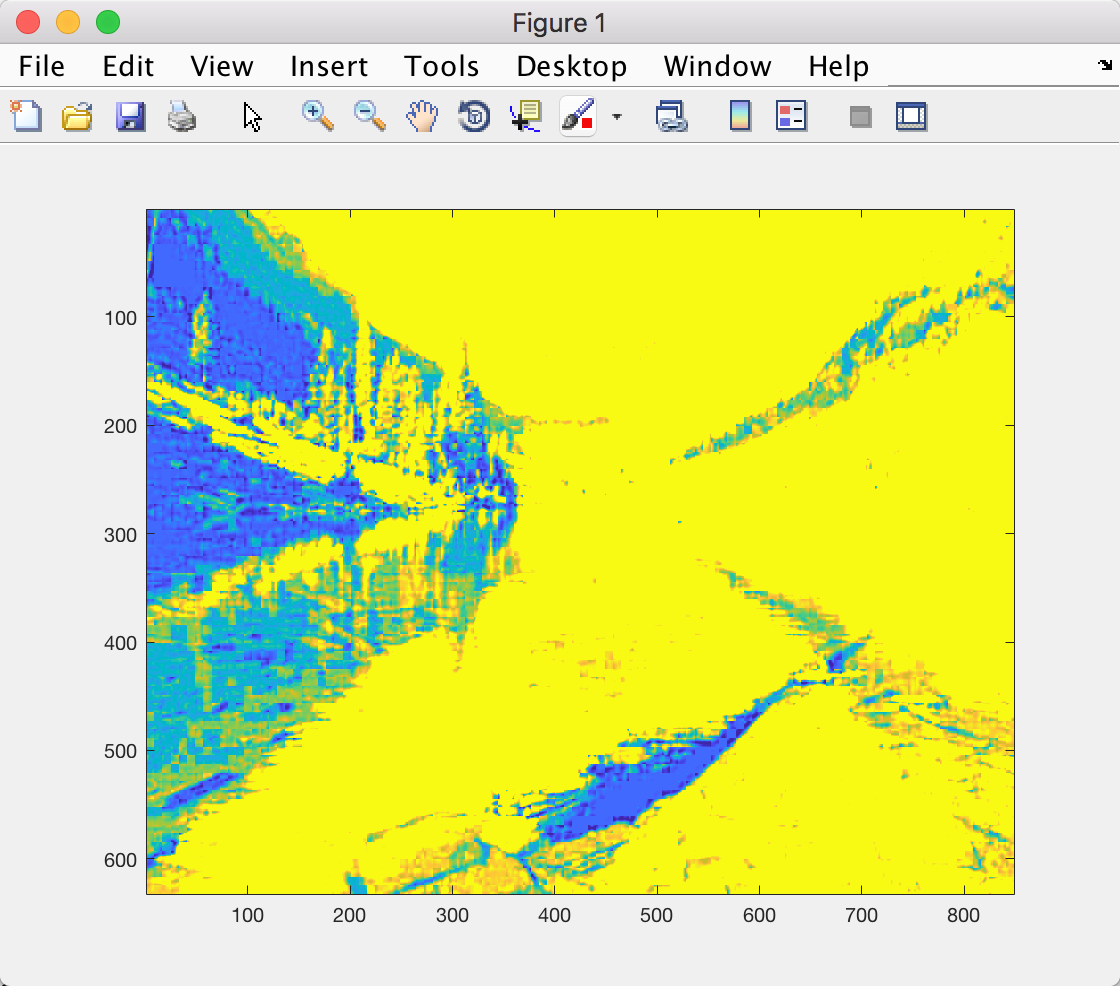
7.614065749294735 (for uint8 type matrix)

If we assume that the coder and decoder know the quantizer, the compression ratio is:0.15625



(Use image() function to show the image)

SNR = 18.962413384233190



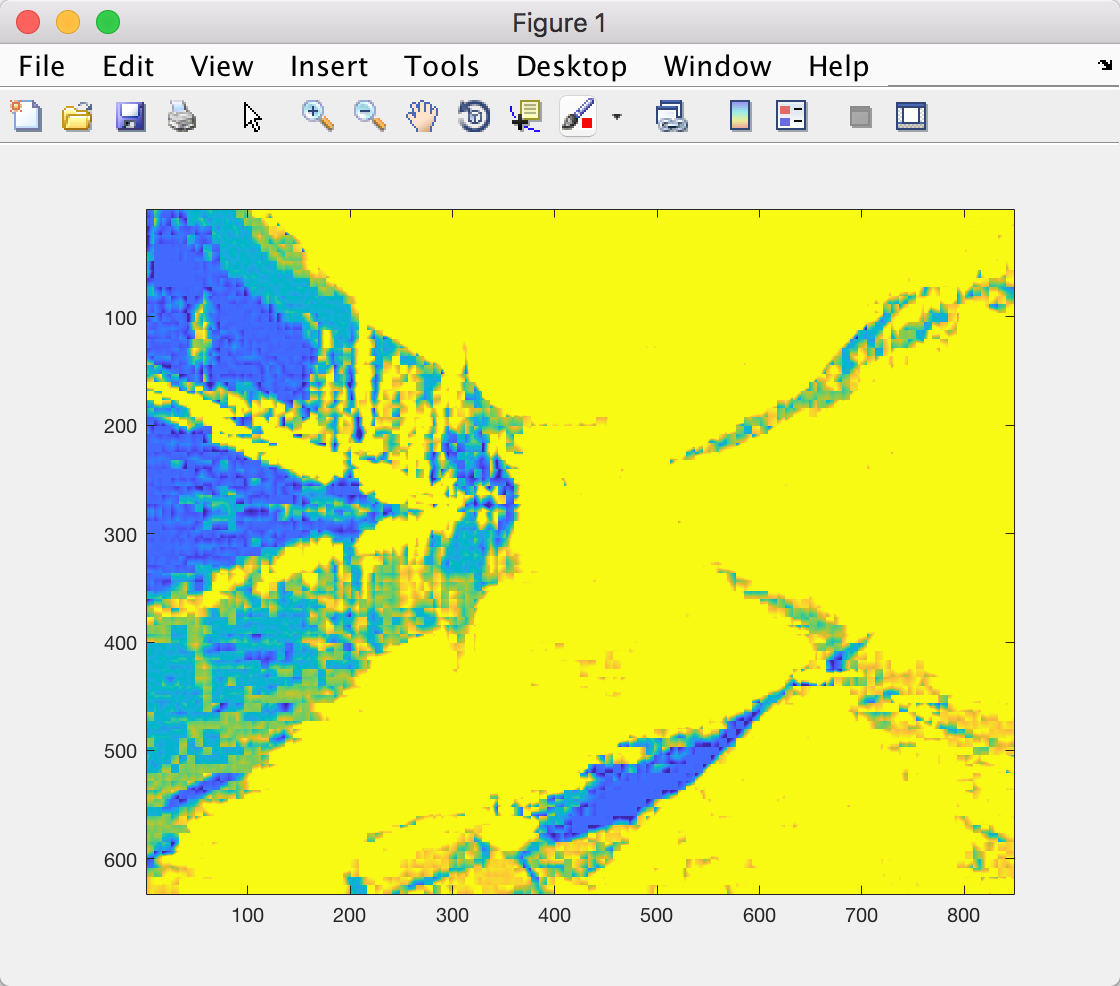
(Use image() function to show the image)

SNR = 17.830448187369820

Bitrate = 0.017463837768157 (for double type matrix)

7.588477663129759 (for uint8 type matrix)

Compression ratio = 0.09375



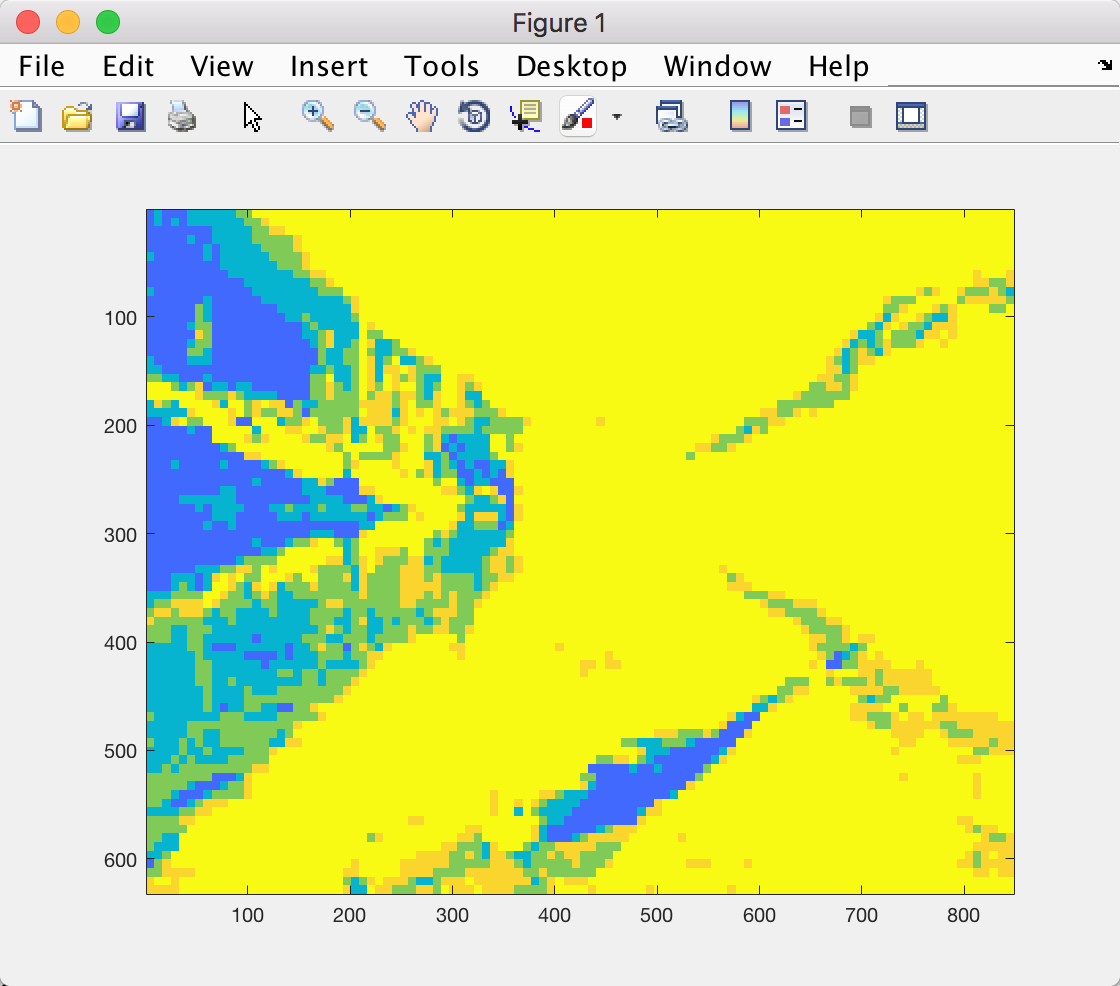
(Use image() function to show the image)

SNR = 16.607291880329310

Bitrate = 0.017202164272909 (for double type matrix)

7.520418096755978 (for uint8 type matrix)

Compression ratio = 0.03125



(Use image() function to show the image)

SNR = 14.925232783487346

Bitrate = around 0(for double type matrix)

3.737440826951699 (for uint8 type matirx)

Compression ratio = 0.015625

1. SNR(b) > SNR(c) > SNR(d) > SNR(e)

Bitrate(e) < Bitrate(d) < Bitrate(c) < Bitrate(a)

For the picture quality, we can hardly tell there are any difference between and . While, the quality of is not so good. The quality of really bad.