**Zach Wilson**

**-Lossy (Irreversible) Data Compression-**

**Digital Signal Processing**

**4/13/2018**

Imagine you and your family are about to embark on a wonderful vacation to a tropical island in the Caribbean, the ancient castles of Europe, or the other worldly beauty of New Zealand. In order to get there, you have to fly. The challenge of all trips is packing enough materials, clothes, and toiletries to get you through the trip, without breaking the bank with baggage fees from airlines for the extra storage space needed.

Because you aren’t willing to pay baggage fees, you know you need to reduce baggage weight. You need to *compress* your items. Your options are to either simply take the bare necessities, or to make a list and buy the missing items at your destination. In a way, this is much like data transfer in the digital world. You have either ‘Lossless’ or ‘Lossy (Irreversible)’ data compression algorithms. In this paper, the focus will be the secondary option. It is the minimalist approach, to say the least. What is the smallest number of items that we can take on the vacation and still have enough to get us through.

The internet is the airline traffic of the digital world. The trillions of downloads, uploads, and streams that happen daily in this world have to be streamlined in order to keep the world running. Most compression algorithms run automatically in services that drop useless information. This is most commonly seen in multimedia messages, internet streaming, as well as telephone communications. The ultimate goal is “How much information can be dropped from the signal before the end user notices”. The best lossy compression algorithms can remove all unnecessary information from a signal that still leaves enough data there to reproduce at the other end without any noticeable degradation.

In audio, digital marketing, and the film industry, it is common to have different levels of compression. The original file, known as the Master Lossless File, contains all of the original information. The firm will then compress copies of this file to different ratios depending on the service it is to be used for. Examples of this are a video posted to a streaming service. The film corporation that created a movie has the ultimate video file with all of the original information. However, once the video is available on Netflix, compression algorithms will reduce unnecessary information in the file and allow it to be streamed over the internet faster. Videos can be significantly reduced before movie patrons would even notice. The user watching Netflix usually won’t notice the drop in quality, and enjoys the tradeoff of being able to instantly watch the movie as opposed to waiting for it to download.

Multimedia is one of the most extreme examples. When sending videos over a cellular broadband network, the most important goal of users is that the video gets to the recipient in a reasonable time frame. Videos are very large files when not compressed, and would take a very long time to send all original data. The priority is the end goal in this case, so the data compression algorithms significantly reduce the video quality with noticeable degradation. This is an example, as opposed to the Netflix example earlier, where poor quality is accepted for instantaneous data transfer.

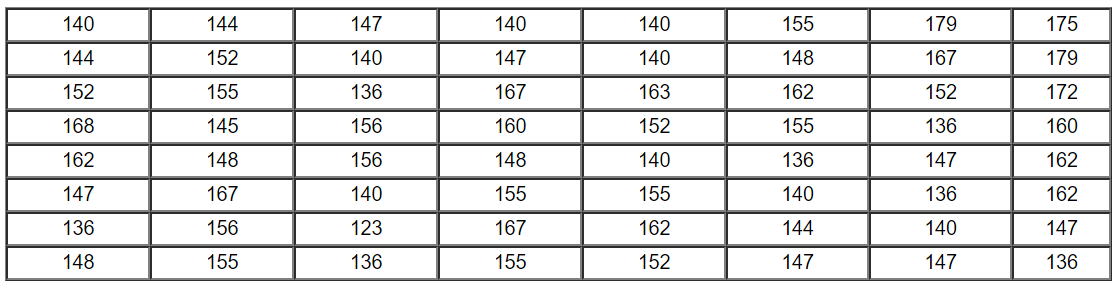
While multimedia messages do at times consist of videos, the majority consists of pictures. The industry standard for pictures on a computer is a JPEG file. A JPEG file was created on the basis that the human eye can only see a specific range of frequencies, as well as only being able to distinguish a specific quality level in photographs. That means the majority of the information stored in a picture can be thrown out as useless in order to preserve hardware storage space. The industry standard is known as the “JPEG Baseline Algorithm”. This algorithm is so efficient, it can compress images to nearly 0.1 times the original size (Stanford).

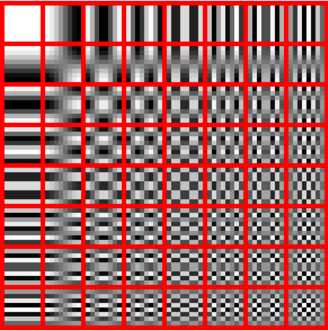


**Figure 1.** A picture of a forest in Greece, the red square marks the pixels of focus. Imagine the square is 8x8 pixels.

The JPEG Baseline Algorithm is a lossy compression sequence that relies on three main steps. It performs a DCT, or the “Discrete Cosine Transform”. The next step is to create quantized coefficients, and finally to encode the information via Huffman encoding. The principle operation of this algorithm is to break up the picture into small pieces. For the image seen above in Fig. 1, imagine small 8 by 8 squares being drawn on to the picture to fence in a total of 64 pixels. This is present in the visible red square. These pixels are then converted from standard RGB to YCbCr. The Y in this form represents the luminance, while the Cb and Crrepresent the blue and red difference in the chrominance. A two dimensional matrix is then formed with the x and y spatial coordinates of the pixel being the row and column position in the matrix, with the value at that point representing the intensity for either the luminance or the chrominance. The luminance is essentially a grey scale version of the picture with the chrominance adding the color. Table 1 below gives an example matrix borrowed from (Stanford).

**Table 1.** An example input matrix that has been shifted by 128, taken from (Stanford).





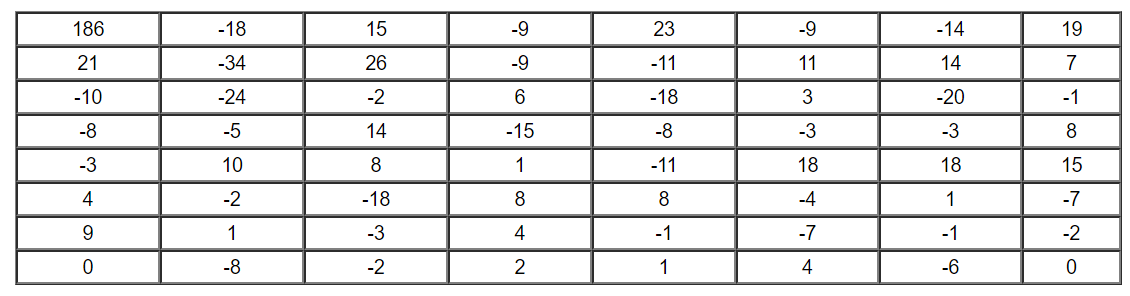
**Figure 2.** Above is a representation of the possible frequency combinations resulting from the DCT (Wikipedia).

**(1)**

\*The above equation is sourced from (Stanford).

Once the matrix is obtained, (see Table 1), it is shifted by 128 to put the center point value at zero. The shifted matrix is then transformed from its two dimensional spatial domain into the frequency domain with the DCT. The specifics of how the DCT works will not be covered in depth due to word restraints on this paper, however it creates a combination of different cosine functions and generates a matrix of transform coefficients. Fig. 2 above shows 64 quadrants that relate spatially to the pixel input values. The coefficient acts as a measure of the presence of each one of the 64 quadrants in Fig. 2. The top left of Fig. 2 represents no color transition and lower frequencies, while the bottom right represents the higher frequencies. Table 2 below represents the DCT output matrix, with the converted values borrowed from (Stanford).

**Table 2.** An example DCT output matrix, taken from (Stanford).



Once a matrix of DCT coefficients is calculated, the next step is to quantize the coefficients. This is where the “lossy” part of the JPEG algorithm takes place. Looking at Table 2, it can be seen that the top left values are greater than the bottom right values. This is referred to as the DC and AC values respectively. The DC magnitude values are much larger than the AC values, as to be expected. The more AC the value, the less it will contribute to the overall picture. Below in Table 3 is an example of a common quantization matrix for 50% compression (Wikipedia). The DCT coefficients are divided by the corresponding quantization matrix value, and then rounded to the nearest integer. The purpose of this is to scale all of the values, as well as zeroing out the vast majority of the non-important frequencies. The quantized coefficient results are presented in Table 4.

**Table 3.** A common quantization matrix for 50% compression, values taken from (Wikipedia)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 16 | 11 | 10 | 16 | 24 | 40 | 51 | 61 |
| 12 | 12 | 14 | 19 | 26 | 58 | 60 | 55 |
| 14 | 13 | 16 | 24 | 40 | 57 | 69 | 56 |
| 14 | 17 | 22 | 29 | 51 | 87 | 80 | 62 |
| 18 | 22 | 37 | 56 | 68 | 109 | 103 | 77 |
| 24 | 35 | 55 | 64 | 81 | 104 | 113 | 92 |
| 49 | 64 | 78 | 87 | 103 | 121 | 120 | 101 |
| 72 | 92 | 95 | 98 | 112 | 100 | 103 | 99 |

**Table 4.** The resultant quantized coefficient values.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 12 | -2 | 2 | -1 | 1 | 0 | 0 | 0 |
| 2 | -3 | 2 | 0 | 0 | 0 | 0 | 0 |
| -1 | -2 | 0 | 0 | 0 | 0 | 0 | 0 |
| -1 | 0 | 1 | -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 |

Table 4 above shows the data compression achieved through the quantization process. The higher frequency values are zeroed out, leaving only the lower frequencies to reconstruct the image. The final steps of the process are a lossless encoding process using Huffman Encoding, which essentially removes any zero coefficients before transfer.

While the idea of keeping every bit of information is appealing, it is completely unnecessary. The human brain can only filter and process a limited amount of information. Engineers are always working to exploit this principle to reduce hardware storage demands as well as speed up transfer speeds. The amount of information compressed is completely dependent upon the situation. The example given in this document shows the incredible ways that engineers can reduce unnecessary information from a picture. The resulting JPEG image would look exactly identical to the human eye, yet only fifty percent of the original data was used. That is the power of lossy data compression algorithms.

**References**

“Data Compression”. *Stanford*. Accessed 10 April 2018.

<https://cs.stanford.edu/people/eroberts/courses/soco/projects/datacompression/overview/index.htm>

“JPEG”. *Wikipedia.* 11 April 2018.

<https://en.wikipedia.org/wiki/JPEG#Discrete_cosine_transform>

“YCbCr”. *Wikipedia*. 3 October 2017.

<https://en.wikipedia.org/wiki/YCbCr>

“Lossy Compression”. *Wikipedia*. 7 March 2018. <https://en.wikipedia.org/wiki/Lossy_compression>

“JPEG DCT, Discrete Cosine Transform (JPEG Pt2)- Computerphile” *YouTube*, uploaded by Computerphile, 22 May 2015, <https://www.youtube.com/watch?v=Q2aEzeMDHMA&t=687s>.

“JPEG”. *Joint Photographic Expert’s Group*. jpeg.org/about.html. Accessed 12 April 2018.

<https://jpeg.org/about.html>