BPG Research Report by Eish Kapoor, Gavin Jampani, and Vivek Ponnala

Table of contents

Introduction and Background

Images can record richer content than other data forms such as text, so they are widely used in the modern information age. Simultaneously, as the resolution of imaging technologies improves, the average size of images grows, implying that a single image requires more storage space and a larger communication line bandwidth for image transfer, which becomes a problem in many cases [1].

Massive images, for example, can provide more comprehensive data for analysis in remote sensing, but they will require a large amount of memory during storage of the spaceborne acquisition and large bandwidth for image transmission to the ground system. This is also true for applications running on portable or mobile devices [1]. As a result, image compression is critical, and it must meet a number of criteria. A compressed image should require less storage space as well as less bandwidth or transmission time. Simultaneously, real-time online browsing and higher quality are desired [1].

Lossless compression and lossy compression are the two types of image compression. Lossless compression primarily accomplishes compression by reducing information redundancy. After lossless image compression, there is no information distortion. It is appropriate for some situations requiring ideal image information restoration, but its compression ratio (CR) is limited. Lossy compression, on the other hand, can easily break through the upper limit of the compression ratio and achieve a compression ratio of 10 times or more [1].

Retaining image quality is an important consideration in lossy compression. This can be solved via two methods. The first method is the iterative procedure. This method approaches the desired value of the chosen quality metric step by step. The accuracy of this method is usually satisfactory but multiple iterations cause sufficient time for compression. The second method is prediction. This method extracts and calculates information from an original image's properties in order to predict image quality after compression/decompression. This method does not require iterations and can be implemented quickly; however, the prediction process is dependent on the compression algorithm, and the accuracy needs to be improved. This method performs the first compression step using the average rate-distortion curve obtained offline and then uses the image quality after the first compression as feedback for the second compression step to provide the desired image quality. This method has been tested on AGU, ADCTC (based on discrete cosine transform - DCT), and SPIHT (based on discrete wavelet transform - DWT) compression encoders [1].

The BPG encoder is a new compression technique that outperforms JPEG in terms of compression quality and image size. The encoder has quickly gained popularity due to its superior performance, particularly in portable devices and online encoding applications[1]. The BPG encoder is based on the HEVC encoder, which is regarded as a significant advancement in compression technology. HEVC has a high coding efficiency due to the clever method used to reduce the coding area (pixels) [1]. There are two types of lossy compression coding: predictive coding and transform coding [1]. Predictive coding involves predicting the current data value using a model and encoding the difference between the predicted value and the actual value of the sample; predictive coders include SZ, FPZIP, and others. Transcoding is the process of converting the original data into another space in order to generate more data close to zero[1].

Algorithmic Analysis

BPG Technologies

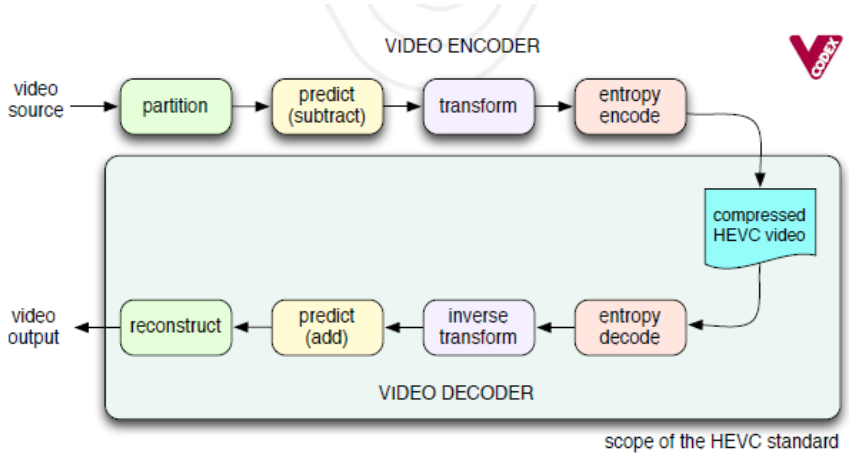

Figure 1: Basic diagram of the HEVC encoder and decoder [15]

The main technology that BPG uses is a part of the HEVC encoder and decoder. HEVC, also known as H.265, is an improvement over the old video encoder and decoder standard H.264. HEVC requires less bandwidth and has an improved image quality. **Figure 1** shows the entire video encoder and decoder process, however, our research focused mainly on the encoder part. BPG specifically uses the intra-frame encoding of HEVC only and not the inter-frame encoding of HEVC. The reason is inter-frame is used for moving images in a video and BPG format is for coding digital images hence it only needs the intra-frame encoding. Intra-frame happens within a single frame which is what BPG needs since it compresses single images.

The first stage of the HEVC encoder, the partition stage, is part of the initialization phase where each input image is broken into multiple smaller subunits. The rest of the stages are part of the actual encoding phase. The second stage is the prediction stage where the encoder subtracts a prediction from a subunit. The prediction stage in HEVC includes the intra and inter prediction, but BPG only uses the intra prediction since it only compresses single images. There are 3 main types of frames: I-frame (used for intra frame prediction to reduce spatial redundancy), P-frame (used for inter frame prediction to hold only the changes in the image from the previous frame

), and B-frame (used for inter frame prediction of bidirectional frames/slices (macroblocks)). The transform stage handles the transforming and quantizing the residual signal from the prediction (the difference between the original picture unit and the prediction). One of the most popular transforms is the Discrete-Cosine Transform (DCT). The last stage is the entropy encoding which, for the general HEVC, includes popular algorithms like Huffman Encoding or Arithmetic encoding.

BPG Algorithms

In this section, we will be going step by step over the algorithms and steps necessary for encoding in the Better Portable Graphics format.

Initialization

The BPG encoder takes in images of type JPEG or PNG, and it requires the input image to meet certain requirements for compression. To do so, it surveys the image's metadata, color space, alpha, and pixel depths. Two requirements of note are that BPG requires that the amount of data to indicate the color of each pixel, or bit depth, to be 8 bits and the color space to either be true color or grayscale. If all these prerequisites are met, the encoder moves onto the second phase [2].

Intra-Prediction

The Better Portable Graphics format follows the High Efficiency Video Coding standard, and in that standard there are two different aspects of prediction: inter-prediction and intra-prediction. Since BPG works with images, the BPG algorithm is solely based on the intra-prediction aspect of the HEVC [11]. The HEVC contains three steps for its intra-prediction: reference sample array construction, sample prediction, and post-processing. In implementing these three steps, there are three main categories: angular prediction, DC prediction, & planar prediction. No matter the category, the first step is reference sample array generation, done at block sizes ranging from 4x4 to 32x32 samples. It is enacted by extrapolating sample values, which are also run through a sample substitution process that differentiates it from previous standards. This expands the available set of intra-prediction modes available by removing the dependency on neighboring samples. These sample values are extracted into the aforementioned blocks, which are then further divided into Coding Tree Units (CTUs) and Coding Units(CUs). On these coding units, one of the three categories of prediction is run. Here is the breakdown of each of them:

- **Angular** - The mode extracts samples from the reference samples based on the given directional orientation.
- **Planar** - Plane prediction mode in which each sample value is calculated by assuming an amplitude surface with a smooth gradient in both x and y directions.
- **DC** - Each sample value is calculated by taking the average of the reference samples. These reference samples are the neighboring top and left blocks.

Transform & Quantization

Transforms are next applied to the residual signal resulting from the intra-prediction. Residual blocks of this signal are then divided into 4x4, 8x8, 16x16, or 32x32 blocks and inputted into a dimensional forward transform of the same NxN size. In the case of the BPG format, the transform used is the Discrete Cosine Transform, or DCT[10]. The 2D DCT transform is as follows:

$$F(u,v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \Lambda(i).\Lambda(j).cos\left[\frac{\pi.u}{2.N}(2i+1)\right] cos\left[\frac{\pi.v}{2.M}(2j+1)\right].f(i,j)$$

Figure 2: 2D DCT Transform Equation

The reason that DCT is used is because it has several algorithmic advantages that lend itself to stronger compression efficiency. Some of these advantages include[8]:

- Basis vectors are orthogonal and have favorable energy compaction, both of which are desirable traits for compression efficiency.
- DCT matrix can be specified using a small number of unique elements. A matrix of size $2^M x 2^M$ can be specified by 2^M -1 unique elements.
- DCT even basis vectors are symmetric, odd basis vectors non-symmetric. This property reduces arithmetic complexity.

The result of DCT transform being applied is transform coefficients. These transform coefficients are then quantized, which consists of division by a quantization step size. This step size is determined by a quantization parameter[10].

Entropy Encoding

The quantized transform coefficients are then entropy encoded, which uses statistical properties in such a way that the number of bits representing data is logarithmically proportional to the data's probability. It is a lossless compression scheme with an input of a series of syntax elements. These syntax elements describe the partition properties, such as quantization parameter, type of offsets, and more. The specific method used for entropy coding is CABAC, which stands for Context-Based Adaptive Binary Arithmetic Encoding [2]. CABAC is not unique to HEVC- it was also used in H.264/AVC; the data dependencies of the implementations are what distinguishes the two. CABAC consists of binarization, binary arithmetic coding, and context modeling. The figure below shows the CABAC process:

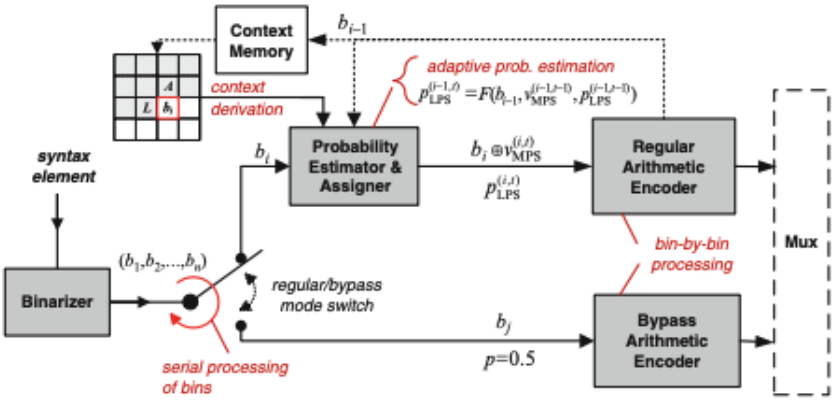

Figure 3: CABAC process

What differentiates the CABAC from the previous iterations of it in H.264 were its improvements in transform coefficient coding, specifically in the coefficient grouping, and improved ordering of significance flags. It also improved on the throughput bottlenecks, reducing coded bins and context modeling dependencies. The CABAC, through these multitudes of steps, successfully eliminates redundancy in the code and in the bitstream. The output bitstream results in a successful encoding.

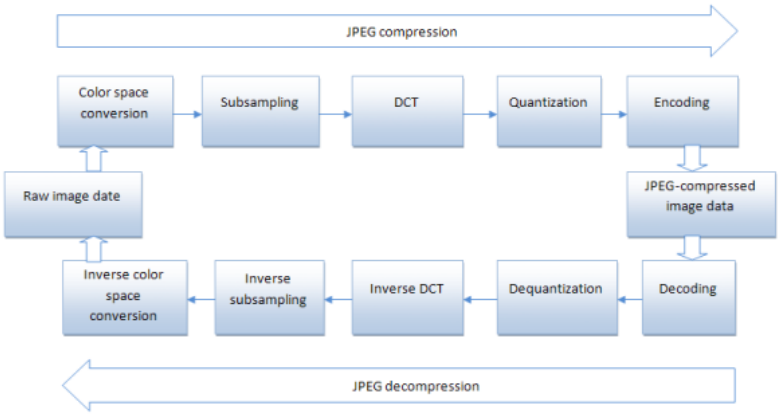

Figure 4: JPEG encoder diagram [14]

For the JPEG Encoder, there are five main steps, which are the following: RGB to YCbCr color space conversion, preprocessing for DCT transformation, DCT transformation, coefficient quantization and lossless encoding. First, a digital image in RGB format that is a combination of red, green and blue channels is converted to YCbCr color channels. Y is the brightness of the image, Cb is the blue difference relative to the green color and Cr is the red difference relative to the red color. The next step is JPEG chroma sampling. The luminance channel is maintained at full resolution, but the chrominance channels are down sampled by 2:1 horizontally or 2:1 vertically since the eye is unable to distinguish between different colors but can differentiate between changes in brightness or contrast. Luminance and chrominance components of the image are divided up into an array of 8*8 pixel blocks and padding is provided to ensure blocks on the right and bottom are full [4].

8*8 pixel blocks fed into a process that performs forward 2D DCT, which generates 64 values. The above is the equation for the forward 2D DCT. The purpose of the 2D DCT is to convert spatial information into spectral information. The next step is the quantization process. Values in the quantization matrix are chosen to preserve low frequency and discard noise. There are different quantization matrices for Y,Cb and Cr color space. The next step is Huffman encoding which increases memory efficiency by 70%. Instead of just storing the pixels, the pixels and their frequency are stored. The final step is zigzag scanning. All the DCT AC terms are collected via a zigzag. The DCT DC term is calculated based on the previous quantized DCT DC value [4].

Experiment

Experimental Design

We used 18 images from a dataset that said “Images are available without any prohibitive copyright restrictions” [11]. Our images contain a mix of RGB for color comparison and Grayscale. They also contain a mix of different file sizes such as 8-bit, 16-bit, and 16-bit linear. The 16-bit linear is raw and unprocessed and parts of the generated image are not recognizable by the human eye. We used 4 different metrics:

- **Compression Ratio:** We compare compression ratios to see how much coding, Inter pixel and psycho visual redundancy from the image have been reduced. Furthermore, it gives us a better understanding of how much the images have been compressed by either image compression algorithm.

$$Compression\ ratio\ =\ \frac{Total\ number\ of\ bits\ in\ initial\ image}{Total\ number\ of\ bits\ in\ compressed\ image}$$

Equation 1: Compression Ratio [8]

- **Peak Signal-to-Noise Ratio (PSNR):** The ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. We use this to see the difference in image quality.

$$PSNR\ =\ 20\ *\ log_{10}(\frac{MAX\ PIXEL}{RMSE})$$

Equation 2: PSNR [8]

- **Mean-Squared Error (MSE):** The average of the squares of the errors

$$MSE\ =\ \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}[I(i,j)\ -\ C(i,j)]^{\ 2}$$

Equation 3: MSE [8]

- **Root Mean-Squared Error (RMSE):** Measures the average magnitude of error

$$RMSE\ =\ \frac{1}{\sqrt{MN}}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}|I(i,j)\ -\ C(i,j)|$$

Equation 4: RMSE [8]

Procedure and Implementation

For the procedure and implementation for our experiment, there are three main stages: converting the format, writing the script, setting the augments and passing images through BPG and JPEG encoder. From our dataset of test RGB and gray-scale images, the RGB images were in ppm format and the gray-scale images were in pgm format. For the JPEG encoder, the images have to be in bmp format so they can be converted to jpeg. Similarly, for the BPG encoder, the images have to be in png format so they can be converted to BPG. For converting the format of the images, we used https://convert.io [13]. The next step is writing our script for the BPG and JPEG encoder.

The four main scripts written and used were: bpgenc.c, metrics.py, huffmanencoding.py and jpegencoder.py. The Huffman encoding python file gets the frequency of different pixels and stores the pixels and the frequency instead of storing just the pixels. Used for improving efficiency and saving 70% of memory. The Jpeg encoder python file implements the following: rgb to ycbcr color space conversion, preprocessing for DCT transformation, DCT transformation, Coefficient quantization [6]. The BPG encoder C file does the following: converting color space and preprocessing image, implementing HEVC encoder, which has prediction(I frames, P frames), transformation and quantization, and entropy encoding. In entropy encoding, a unique prefix-free code is assigned to each unique symbol that occurs in the input. These entropy encoders compress data by replacing each fixed-length input symbol with corresponding variable length prefix-free output code word [3]. The metrics python file consists of the following image quality parameters: MSE, RMSE, PSNR and compression ratio.

Setting the augments is the last stage of the implementation. For the BPG encoder, since we have RGB and gray-scale images, there are two cases. For the BPG encoder, we set the augments for the gray-scale images like the following: -b 8 -e x265 -m 9. For the gray-scale images in the BPG encoder, the bit depth is 8, HEVC encoding is enabled, the compression level is 9 and the YCbCr color space is used. For the RGB images, we set similar augments but we change the color space to RGB to specifically handle RGB 8-bit, 16-bit, and 16-bit linear images [3]. For the JPEG encoder, we set the default quantization parameter as 95 [6].

Results for Gray 8-bit

mode	Format	Images encoded	CR	PSNR(db)	MSE(db)	RMSE(db)
BPG	gray8bit	city.png	13.24	32.66	35.24	5.936
	gray8bit	fireworks.png	13.20	40.48	5.8208	2.412
	gray8bit	flower.png	25.04	41.12	5.024	2.241
JPEG	gray8bit	city.bmp	12.87	34.59	22.60	4.753
	gray8bit	fireworks.bmp	77.85	39.02	8.143	2.854
	gray8bit	flower.bmp	24.64	45.90	1.671	1.292

Figure 3: BPG vs JPEG gray 8-bit results

Results for Gray 16-bit

mode	Format	Images encoded	CR	PSNR(db)	MSE(db)	RMSE(db)
BPG	gray16bit	city.png	43.64	80.79	35.79	5.980
	gray16bit	fireworks.png	68.2	88.36	6.258	2.502
	gray16bit	flower.png	188.9	89.11	5.270	2.290
JPEG	gray16bit	city.bmp	12.78	82.79	22.60	4.760
	gray16bit	fireworks.bmp	77.57	87.16	8.261	2.874
	gray16bit	flower.bmp	24.83	94.16	1.647	1.283

Figure 4: BPG vs JPEG gray 16-bit results

Results for Gray 16-bit Linear

mode	Format	Images encoded	CR	PSNR(db)	MSE(db)	RMSE(db)
BPG	gray16bitlinear	city.png	173.7	86.06	10.63	3.260
	gray16bitlinear	fireworks.png	91.89	90.24	4.057	2.014
	gray16bitlinear	flower.png	225.4	94.84	1.409	1.187
JPEG	gray16bitlinear	city.bmp	75.16	86.86	8.860	2.977
	gray16bitlinear	fireworks.bmp	115.8	87.07	8.433	2.903
	gray16bitlinear	flower.bmp	177.7	108.8	0.057	0.239

Figure 5: BPG vs JPEG gray 16-bit linear results

Results for RGB 8-bit

mode	Format	Images encoded	CR	PSNR(db)	MSE(db)	RMSE(db)
BPG	rgb8bit	city.png	7.125	32.04	40.66	6.376
	rgb8bit	fireworks.png	14.57	39.48	7.323	2.706
	rgb8bit	flower.png	32.05	41.03	5.124	2.260
JPEG	rgb8bit	city.bmp	9.86	29.35	75.48	8.688
	rgb8bit	fireworks.bmp	69.86	34.71	21.87	4.688
	rgb8bit	flower.bmp	73.76	35.87	16.83	4.103

Figure 6: BPG vs JPEG RGB 8-bit results

Results for RGB 16-bit

mode	Format	Images encoded	CR	PSNR(db)	MSE(db)	RMSE(db)
BPG	rgb16bit	city.png	21.01	80.21	40.91	6.396
	rgb16bit	fireworks.png	53.68	87.61	7.446	2.728
	rgb16bit	flower.png	148.38	89.00	5.406	2.325
JPEG	rgb16bit	city.bmp	9.859	77.55	75.48	8.688
	rgb16bit	fireworks.bmp	69.86	82.91	21.97	4.688
	rgb16bit	flower.bmp	73.75	84.07	16.83	4.103

Figure 7: BPG vs JPEG rgb 16-bit results

Results for RGB 16-bit Linear

mode	Format	Images encoded	CR	PSNR(db)	MSE(db)	RMSE(db)
BPG	rgb16bitlinear	city.png	106.56	84.99	13.61	3.689
	rgb16bitlinear	fireworks.png	94.15	90.37	3.943	1.986
	rgb16bitlinear	flower.png	206.9	94.94	1.376	1.173
JPEG	rgb16bitlinear	city.bmp	15.99	82.98	21.61	4.649
	rgb16bitlinear	fireworks.bmp	100	82.00	27.11	5.206
	rgb16bitlinear	flower.bmp	111.3	98.42	0.6184	0.7864

Figure 8: BPG vs JPEG rgb 16-bit linear results

BPG Image Comparison

Figure 9: Decoded RGB 16-bit BPG Image of flower

JPG Image Comparison

Figure 10: Encoded RGB 16-bit JPEG Image of flower

Graphs

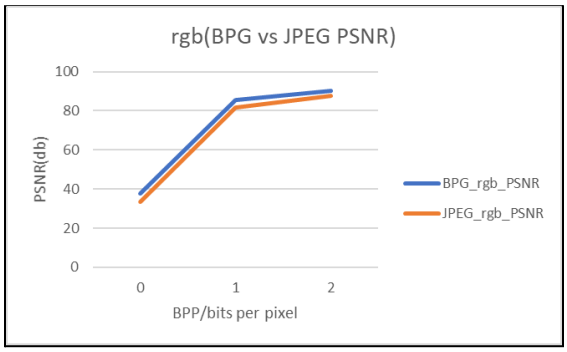

Figure 11: PSNR comparison of BPG and JPEG RGB images

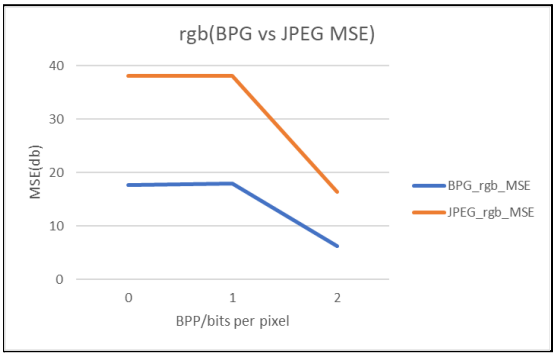

Figure 12: MSE comparison of BPG and JPEG RGB images

Evaluation

We see BPG relatively offers a much higher compression ratio than JPEG, a higher PSNR and MSE than JPEG. For RGB images, BPG is a clear winner in terms of image quality. For grayscale images, BPG and JPEG images perform relatively the same but BPG offers a much higher compression ratio. BPG on average gives a much better compression ratio since it gives a higher dynamic range and there are less rounding errors in the BPG decoder [7]. BPG clearly outperforms JPEG for PSNR since the quantization matrix has lower values for the RGB color space, and the YCbCr color space. Additionally, entropy encoding is performed. BPG provides a much lower MSE since BPG uses the HEVC encoder, to improve image quality so the compressed image is quite similar to the original image. As mentioned earlier, this is because the HEVC encoder uses I-frames which are used to compress single images. In our dataset rgb16bitlinear performs the best between BPG and JPEG since there is a high compression ratio and PSNR [2]. Additionally, raw images do not have to be processed much, leading to less noise in the image itself. However, we have good image quality metric results for JPEG and BPG since they both use 2D DCT which achieves compression ratios of 37:1 before distortion appears [5].

Reflection and Next Steps

Through this experiment, we were able to understand the differences between BPG and JPEG. There are a couple of improvements or things we could try for next time:

- **Quantization:** Change quantization metrics for BPG and JPEG encoder to evaluate differences in PSNR, MSE and RMSE
- **Block size:** Change block size to 16*16, 32*32 to see the effect on SSIM
- **Transforms:** We used 2D DCT, there are plenty of others we can try using like 3D DCT, DST, DWT, WHT, DFT to see changes in PSNR, MSE and RMSE
- **Change entropy encoding methods for better memory efficiency:** arithmetic encoding
- **Different direction:**
    - Use different file formats such as FILF and WebP to see image compression results
    - Compare differences if BPG uses lossless compression and JPEG uses lossy

References

[1] “An Approach to Better Portable Graphics (BPG) Compression with Providing a Desired Quality.” *IEEE Xplore*, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9349289&tag=1[Accessed: 26-Nov-2021].

[2] *A Hardware Architecture for Better Portable Graphics (BPG ...* http://smohanty.org/Publications_Conferences/2015/Mohanty_IEEE-iNIS-2015_BPG-Architecture.pdf

[3] “BPG Image Format.” *BPG Image Format*, https://bellard.org/bpg/

[4] Dias, Danoja. “JPEG Compression Algorithm.” *Medium*, Break the Loop, 4 May 2017, https://medium.com/breaktheloop/jpeg-compression-algorithm-969af03773da

[5] *Different Transforms for Image Compression - Core.ac.uk* https://core.ac.uk/download/pdf/25807932.pdf

[6] fangwei123456. “Python-JPEG-Encoder.” GitHub, https://github.com/fangwei123456/python-jpeg-encoder

[7] Owano, Nancy. “BPG Image Format Judged Awesome versus JPEG.” Tech Xplore - Technology and Engineering News, Tech Xplore, 17 Dec. 2014, https://techxplore.com/news/2014-12-bpg-image-format-awesome-jpeg.html

[8] “Performance Evaluation of High Quality Image Compression Techniques.” *IEEE Xplore*, https://ieeexplore.ieee.org/abstract/document/6968643?signout=success

[9] Radha Krishna A.N, R. K. A. (2013). JPEG Encoder using Discrete Cosine Transform & Inverse Discrete Cosine Transform. *IOSR Journal of Electronics and Communication Engineering*, *5*(4), 51–56. https://doi.org/10.9790/2834-0545156

[10] *Saliency-Based Image Compression Using Walsh Hadamard ...* https://www.researchgate.net/profile/A-Andrushia/publication/319175696_Saliency-Based_Image_Compression_Using_Walsh-Hadamard_Transform_WHT/links/5bf2667d299bf1124fdd0a98/Saliency-Based-Image-Compression-Using-Walsh-Hadamard-Transform-WHT.pdf

[11] Sze, V., Budagavi, M., & Sullivan, G. J. (Eds.). (2014). High Efficiency Video Coding (HEVC). *Integrated Circuits and Systems*. Published. https://doi.org/10.1007/978-3-319-06895-4

[12] “The New Test Images.” *The New Test Images - Image Compression Benchmark*, https://imagecompression.info/test_images/

[13] “The Ultimate File Converter.” *The Ultimate File Converter*, https://convert.io/

[14] “Understanding of JPEG Encoding Parameters.” *Graphics Mill*, https://www.graphicsmill.com/docs/gm5/UnderstandingofJPEGEncodingParameters.htm

[15] “HEVC: An introduction to high efficiency coding,” Vcodex. [Online]. Available: https://www.vcodex.com/hevc-an-introduction-to-high-efficiency-coding/. [Accessed: 12-Dec-2021].