Jpeg-xl vs Avif vs Heic vs BCN vs Astc vs Webp vs Qoi vs Png vs Jpeg vs Jpeg2000 vs Exr vs Hdr vs Tiff

Multi Image Codec Introduction/Comparison

Larry Lin, Zhouyang He, Martin Yang

1. Introduction

- Compression rate

  As digital storage continues to expand exponentially while being more cost-effective, the typical consumer might be less concerned about image file size. But increasing the compression rate still remains impactful for AI researchers, professionals in graphic, large-scale storage clusters, and the load of overall network traffic.

- Decoding speed

  The speed of decoding may appear sufficient for the average user, but loading a single PNG might slow down the MCU in an embedded system. Even for modern PCs, the decoding rate can be unacceptable, as newer image codecs increase complexity by a factor of 10-100 or more. Also, most importantly, texture decoding has been a huge problem that requires GPU hardware and dedicated decoder.

- Compression Algorithm

  Many traditional algorithms did not consider the use of progressive or block encoding and were not designed with parallelism in mind. Concepts like resolution and different scales are not considered. Additionally, insufficient metadata design during standardization has caused significant problems for color management and future decoding.

- Field specific support

  Typical image codecs do not contain the information for field specific requirements（dynamic range, norm etc）, so that for 3d/vfx/photo/hdr, displaying, post processing, layer editing, and render passing is not possible.

2. Research Extend Content

- With the advent of newer formats such as AVIF and HEVC, the encoding rate has notably increased. However, the decoding time has also seen a 10x increase, and 20-50x for encoding time .
- While jxl manages to reach the encoding rate of AVIF and HEVC and keep the de/encoding at faster speed, it still requires sufficient power to convince google.
- Dedicated format has been standardized for different purpose such as exr/hdr for the field of vfx/3d, bc6/bc7/astc/etc for gpu achieving on the fly fast decompression, but field specific codec introduced problem like architectural limitation, unacceptable file size (single image over 2Gb)
- Improving algorithms, adding Simd or native specific optimization can largely improve performance in different magnitudes. But simultaneously increase the complexity of porting the codec to alternative platforms and architectures, as well as implementing it at register transfer level.

—----------

Research result:

Jpeg-xl:

universal, fast enc/dec, support for lossy/lossless/frames/hdr

Encode steps:

Image to adaptive chunks for var-dct(8x8-64x64 into 4x4-8x4) (lossy)

encode in layers of quality for progressive rendering

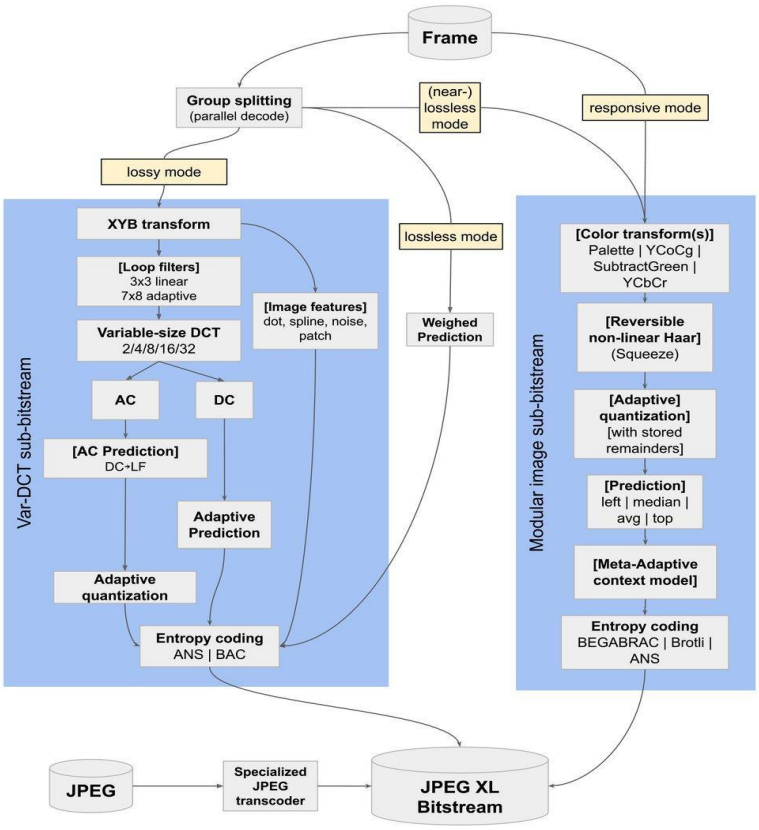
Luma S LM difference color space transform(XYB)

Locally adjusted adaptive quantization

Block, lines, noise in compressed representation for reference

Entropy encoding with Asymmetric Numeral System + lz77(optional)

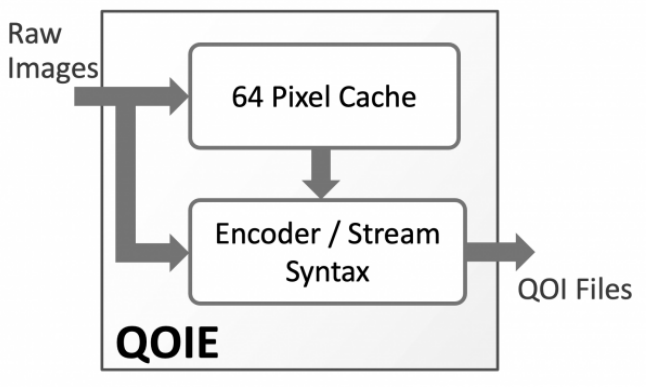
Weighted self correcting predictor adjusted per context with modeling (lossless)



Avif Heic:

Avif and Heic are part of the Av1/Hevc encoder where the frame based compression are used for storing individual image but achieving almost double the compression ratio of Png

Qoi Png：dc rate, dc speed, usage, improvement, limitation



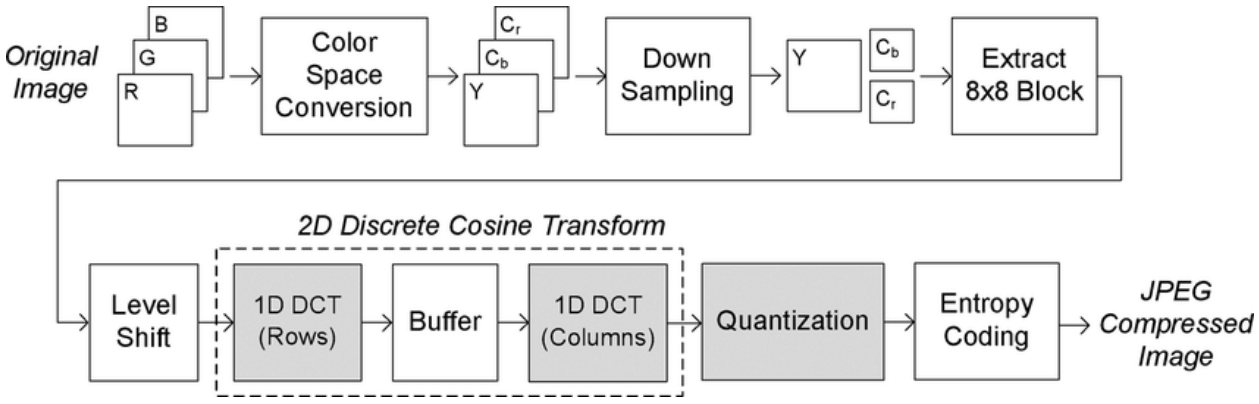
The QOI algorithm compresses RGB or RGBA images with 8 bits per color without any loss.

PNG employs the DEFLATE compression algorithm, a non-patented, lossless data compression technique. DEFLATE is a combination of two main components: LZ77 (Lempel-Ziv 1977) and Huffman coding. The LZ77 algorithm, also known as sliding window compression, identifies repeated sequences of data and replaces them with references to a single copy. This mechanism proves especially effective for

compressing images with areas of uniform color or patterns. Huffman coding, on the other hand, is a variable-length encoding method that assigns shorter codes to more frequently occurring values, optimizing the representation of the data.

DEFLATE operates by first applying LZ77 compression, followed by Huffman coding. During LZ77 compression, the algorithm searches for repeated substrings within the data, creating a dictionary of these substrings and their corresponding references. The Huffman coding step then assigns variable-length codes to the dictionary entries based on their frequency of occurrence. The result is a compressed representation of the original data, where redundancy has been reduced without any loss of information.

Jpeg Jpeg2000 Webp：usage, improvement, limitation

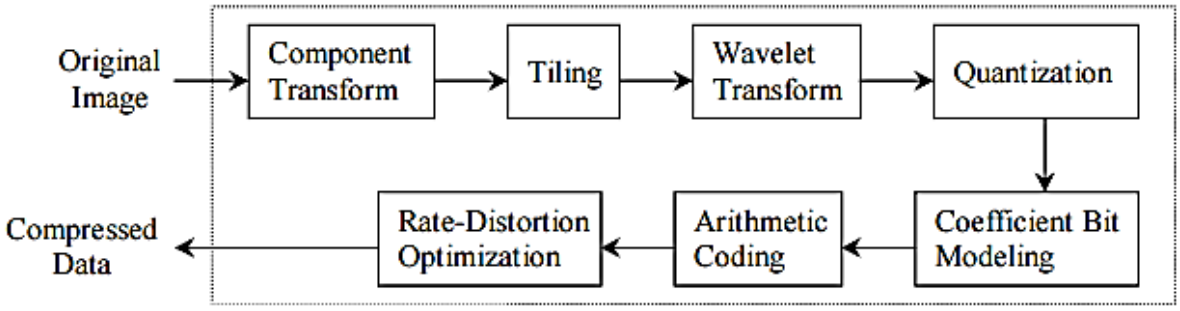


When JPEG first receives the raw data, it employs a color space transformation from RGB (Red, Green, Blue) to YUV (Luma, Chrominance). This transformation is based on the fact that the human visual system is more sensitive to changes in brightness (luminance or Y component) than to changes in color (chrominance or U and V

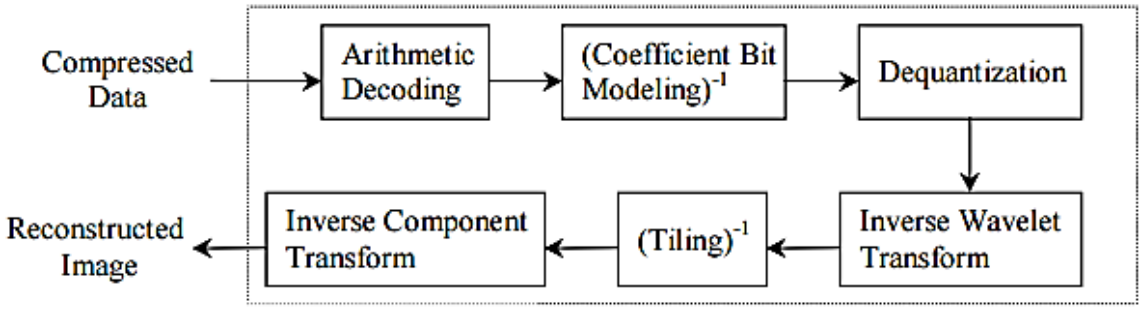
components). By separating luminance and chrominance information, JPEG can take advantage of the human eye's characteristics to achieve more efficient compression. Following that, JPEG divides the YUV image into smaller blocks, typically 8x8 pixels. This block-based approach facilitates the application of compression techniques to individual units of the image. After that, JPEG employs a mathematical transformation known as the discrete cosine transform (DCT) to convert the image data from the spatial domain to the frequency domain. The DCT identifies patterns and variations in intensity within each block, allowing for the separation of image information into different frequency components. High-frequency components, representing fine details and rapid changes in intensity, are then quantized, meaning that their precision is reduced. This quantization step introduces some loss of information, contributing to the characteristic "lossy" nature of JPEG compression. Finally, JPEG uses entropy coding, specifically Huffman coding, to further compress the data. Huffman coding assigns variable-length codes to different symbols based on their frequency of occurrence. This coding scheme is particularly effective at representing more common values with shorter codes, resulting in a more compact representation of the image data.

While the compression achieved by JPEG is significant and allows for the efficient storage and transmission of images, the process inherently introduces some degree of image degradation due to the quantization of high-frequency components. The balance between compression ratio and image quality can be controlled by adjusting compression settings, allowing users to make trade-offs based on their specific needs. JPEG compression is widely used for photographic images on the web and in various

digital media, where the compromise in image quality is generally acceptable for the benefits of reduced file size and faster transmission.



Encoder Processing Steps



Decoder Processing Steps

JPEG 2000 represents a significant advancement in image compression technology, utilizing wavelet transforms to achieve efficient encoding and compression. Unlike the original JPEG format, which relies on discrete cosine transforms (DCT), JPEG 2000 employs wavelet transforms, specifically the discrete wavelet transform (DWT). The wavelet transform allows for a more adaptive representation of image details across different scales and resolutions.

One notable feature of JPEG 2000 is its ability to provide superior image quality at lower bit rates compared to traditional JPEG. This is due in part to the efficient representation of image information through the wavelet transform, allowing for a more accurate capture of both high and low-frequency components. The use of wavelets enables the creation of a multi-resolution representation of the image, facilitating more effective compression while preserving essential details.

JPEG 2000 supports both lossless and lossy compression, providing flexibility to users based on their specific requirements. In lossy compression, users can adjust the compression ratios to achieve a balance between file size reduction and acceptable image quality, making it suitable for a wide range of applications. In scenarios where preservation of every detail is critical, JPEG 2000's lossless compression option ensures that no information is sacrificed during the compression process.

WebP[2] is a versatile image format designed explicitly for web-based applications, combining elements of both lossless and lossy compression to optimize image delivery. In its lossy compression mode, WebP utilizes the VP8 methodology for frame prediction, a technique that efficiently predicts subsequent frames based on preceding ones, reducing redundancy in the image data. This predictive approach allows for significant compression while maintaining perceptual image quality, making it ideal for scenarios where a balance between file size and visual fidelity is crucial, such as web graphics and multimedia content.

On the other hand, WebP's lossless compression mode employs a series of transformative operations applied to the image data without sacrificing any visual information. These operations include techniques like color indexing, entropy encoding, and spatial prediction. By carefully managing the representation of pixel values and patterns within the image, lossless WebP compression achieves compression ratios comparable to other lossless formats, making it suitable for applications where preserving every detail is paramount, such as graphics with text and logos.

BCn Astc:

Texture compression format features fast decompression for on the fly frame and block specific decompression in multi resolution for the purpose of rendering, the algorithm also considers implementation in hardware but architecture might be limited (astc on arm only)

Exr Hdr Tiff：

The OpenEXR(exr)[3] is specifically designed to handle high-dynamic-range images, allowing for the storage of pixel values with a wide range of intensities that is often used in film and vfx. This is crucial in applications where preserving details in both dark and bright areas of an image is essential. It uses a floating-point format to represent pixel values. This allows for precise representation of color and intensity values, avoiding issues like clipping or loss of information. It supports the storage of multiple image layers, each with its own set of channels. This makes it well-suited for complex images that consist of multiple passes, such as render elements in computer graphics. It allows

users to define custom channels with different data types (e.g., float, half, uint) for flexibility in representing various types of data, such as color, depth, and other auxiliary information. It supports a tiled storage format, which can improve performance in certain scenarios, especially when working with specific regions of an image. It allows the inclusion of metadata within the image file. This metadata can include information about the image, camera settings, and other details relevant to the production pipeline.

EXR offers a few different data compression schemes. PIZ compression method is a lossless method. It applies a discrete wavelet transform to pixel data and it encodes the result with Huffman encoding method. A discrete wavelet transform(DWT) has different wavelet filters to choose from and some of them can be used for lossless compression. Then DWT applies a wavelet filter to an image with a high and low frequency filter and followed by downsampling by two to keep the sample rate. The data passing the high frequency filter is called detail and the data passing the low frequency filter is called approximation. It is possible to apply DWT to the approximation information until suitable numbers of level. After applying DWT to the data, the obtained coefficients are quantized with a uniform quantizer and apply Huffman encoding to compress the data.

High Dynamic Range(HDR)[4] image is a format used in film, photography and other industries, and hdr image files contain a wide range of luminance values allowing for a

scene with varying levels of brightness and contrast. It uses floating point representation to store a wide range of values including very large and very small ones. Due to its file size, compress algorithms are necessary to efficiently store and transmit images such as tone mapping algorithms. HDR images are usually stored uncompressed or compressed in other file extensions like TIFF or EXR.

HDR image specification is defined by the International Telecommunication Union (ITU). It includes resolution, digital representation, transfer functions, etc.. Rec.2100 specifies three resolutions: 1920x1080, 3840x2160 and 7680x4320. The color value could be 10 or 12 bits per component.The specified color spaces are RGB, YCbCr, and ICtCp. ICtCp is derived from RGB color space. First RGB is transformed into LMS color space, then applied to the nonlinear transfer function, and then converted to ICtCp. The transfer functions,hybrid log–gamma(HLG) and perceptual quantizer(PQ), are optical-electro transfer functions. PQ is designed to align with human perceptual sensitivity to brightness by ensuring a perceptually uniform distribution of code values. This means that equal perceptual steps in code values accurately correspond to equal perceptual steps in brightness, providing an effective representation of the wide dynamic range found in HDR content. HLG is a broadcast-oriented HDR format designed for backward compatibility. It enables the transmission of high dynamic range (HDR) content over standard dynamic range (SDR) infrastructure. HLG simplifies the production and broadcasting workflow by offering a single signal that can be correctly displayed on both HDR and SDR screens, eliminating the need for additional metadata. This emphasis on compatibility makes HLG a practical choice for broadcast television applications.

HDR is commonly known as HDR imaging technique.In the HDR imaging, there are two steps: radiance mapping and tone mapping. The radiance map serves as a repository for information on radiance values at each location in the scene. Creating this map involves capturing multiple exposures of the same scene at various levels of exposure, spanning a broad spectrum of luminance values. The radiance map is created from the images by extracting the radiance values linked to each pixel. Each pixel within the radiance map corresponds to a specific point in the scene, encapsulating details about the light intensity at that particular spot. To make the HDR content suitable for monitors or screens, tone mapping comes into play. Tone mapping entails compressing the dynamic range, ensuring compatibility with the limited capabilities of standard displays. It enhances local contrast; bringing out details in highlights and shadows; and adjusts the overall brightness and contrast of the image.

Tagged Image File Format (TIFF) is widely utilized within the realms of design, photography, and desktop publishing industries. TIFF files are characterized by their capacity to retain detailed data, primarily employing lossless compression methods such as Lempel-Ziv-Welch compression. However, it is worth noting that due to their substantial file size, TIFF images are not considered optimal for implementation in web design or applications prefer short loading times. Alternatives like JPEG or PNG formats are typically favored in these contexts.

TIFF format is capable of storing bilevel, grayscale, palette-color, and full-color images in multiple color spaces. TIFF supports both lossy and lossless compression methods, providing the flexibility to balance between storage space and processing time based on the needs of the applications utilizing the format. TIFF is not limited to processors, operating systems, or file systems.

TIFF is not only an image file extension and it is also an image container. Each subfile within TIFF uses a data structure called an image file directory (IFD) to handle image entries in the subfile. Each IFD contains one or several entries and is managed by its tag. The tags are arbitrary 16-bit numbers and define its dimensions, color information, compression method, and other details.

TIFF IFD typically includes:

- Image dimensions specify the width and height of the image in pixels;
- Color information is about the color space used in the image, such as RGB, CMYK, grayscale, etc;
- Compression information defines the compression method used to reduce file size (e.g., LZW compression);
- Bits per Sample indicates the number of bits used to represent each color component.
- Image Data stores a value pointing to the actual image data in the file;
- And Various other entries can include information like date and time of creation, software used, authorship, etc.

TIFF images mainly use lossless compression method and are also capable of using lossy compression like JPEG. TIFF supports a number of compression methods and Lempel–Ziv–Welch (LZW) compression is one of them. LZW compression is a dictionary based compression scheme. LZW compression commonly uses a table with 4096 entries and the first 256 entries in the table represent single bytes for the input. The table starts with a set of individual characters as entries and is typically initialized with the unique symbols present in the input data.

The encoding process begins with:

1. Read a input character
2. If the current string exist in the table:
    a. If yes, read next input character, and then go to step 2
    b. If not, create a new entry with the current string and output the code of the current string without the last character. Then drop the sequence before the last character and go to step 1
3. Repeat until reaching the end of the input.

Decoding is similar to encoding and the full dictionary table does not need to be provided from the encoder. The encoder provides the table with the initial symbols or informs the decoder about the table they used like ASCII.The algorithm builds the original file with the table and uses revealed characters to reconstruct the dictionary table with the encoder algorithm in order to restore the original file. However, you could encounter a code that is not yet recorded in the dictionary during decoding. This

happens when the previous sequence repeats itself. The algorithm will use the first single symbol of the last code and append it to the end of the code and make a new entry into the dictionary.

The LZW compression is not the best compression method to use in very small files or files with minimal redundancy. The dictionary overhead could exceed the size of the original files. The dictionary size could also be a problem with a large file. It could require large memory space during encoding and decoding. These two problems could add up together. If a large file with minimal redundancy is compressed with LZW, it will lead to a huge dictionary table and the compressed file will be much larger than the original.

Statistic:

Format capabilities:

Jpeg2000: 4294x4294 mp + 16384 channel

Png: 2147x2147 mp + 4 channel

Jpeg-xl: 1073x1073 mp + 4099 channel

Jpeg: 65x65 kp + 4? channel

webp: 16x16 kp + 4 channel

Avif: 8x4 kp + 5 channel

Heic: 8x4 kp + 5 channel

(Statistic below may vary depending on implementation and hardware)

platform: 5900x win10 single threaded,

software: rust binary linked with static built lib: libav, libaom, libjxl, qoi, qoixx, libwebp

quality: ls: lossless, q2: PSNR ~44, q10: PSNR ~36

lib: zune/rs/base/libav - implementation

Avg ratio: average compression ratio (compressed / uncompressed)

total undecoded: total undecoded source file size

Test image set:

chx: total channel count, width*height*channel

channel is 4 if the format is RGBA/YCCK/CMYK/… or 3 if RGB/YUV/…

painting: 50 image 229478784chx,

screenshot: 50 image 254187069chx,

photo: 50 image 2388787200chx,

internet img: 50 image 184670298chx,

All 4 test set is encoded into 3 quality preset using ffmpeg cli

(note that lossy/lossless is not supported in some format)

type	quality	lib	Img set	Avg time	speed (mpix/s)	Avg ratio	undecoded size
Jpeg	q2	zune	painting	0.2870s	799.4658	0.0092	16.16MB
			screenshot	0.3804s	668.1482	0.008	15.53MB
			photo	2.3203s	1029.5376	0.009	164.51MB
			internet img	0.1684s	1096.4663	0.0079	11.17MB
	q10		painting	0.1774s	1293.6583	0.0033	5.82MB
			screenshot	0.1716s	1481.1214	0.0031	5.92MB
			photo	1.4124s	1691.274	0.0026	46.71MB
			internet img	0.1242s	1486.8959	0.0034	4.84MB
	q2	libav-native	painting	0.4930s	465.5143	0.0092	16.16MB
			screenshot	0.4791s	530.5261	0.008	15.53MB
			photo	2.4954s	957.2675	0.009	164.51MB
	q10		painting	0.4305s	533.0159	0.0033	5.82MB
			screenshot	0.4222s	602.1067	0.0031	5.92MB
			photo	1.5563s	1534.8741	0.0026	46.71MB
Jpeg 2000	q2	libav-libopenjpeg	painting	6.5305s	35.4161	0.0116	20.51MB
			screenshot	6.4451s	49.1176	0.0074	17.93MB
			photo	27.8776s	85.6884	0.0054	97.91MB

Png	ls	zune	painting	0.6431s	359.6451	0.0499	88.09MB
			screenshot	0.5190s	609.9225	0.023	55.66MB
			photo	7.1821s	332.6044	0.0492	897.40MB
			internet img	0.4260s	433.316	0.0348	48.98MB
		libav-native	painting	1.0354s	223.3753	0.0499	88.09MB
			screenshot	1.0376s	305.0898	0.023	55.66MB
			photo	8.4504s	282.6826	0.0492	897.40MB
Qoi	ls	qoi-rs	painting	0.3136s	737.5565	0.0583	102.93MB
			screenshot	0.2304s	1374.0865	0.0268	64.78MB
			photo	3.1686s	753.8827	0.062	1129.83MB
			internet img	0.1848s	999.3335	0.0451	63.54MB
		qoi-base	painting	0.4850s	630.8209	0.0441	102.93MB
			screenshot	0.3823s	886.4785	0.0251	64.78MB
			photo	5.3713s	592.9766	0.0465	1129.83MB
			internet img	0.2930s	840.3162	0.0338	63.54MB
		libav-native	painting	0.6773s	341.489	0.0583	102.93MB
			screenshot	0.6113s	517.8574	0.0268	64.78MB
			photo	5.8066s	411.3895	0.062	1129.83MB

Jpeg xl	q2	libav-libjxl	painting	1.6240s	142.4178	0.0048	8.53MB
			screenshot	2.4667s	128.3366	0.0031	7.60MB
			photo	13.7915s	173.2067	0.0053	96.67MB
	q10		painting	3.7099s	62.3432	0.0014	2.47MB
			screenshot	4.6551s	68.0051	0.001	2.35MB
			photo	27.6358s	86.4381	0.0011	20.15MB
	ls		screenshot	13.1740s	24.0297	0.0143	34.54MB
			painting	14.2891s	16.1862	0.0317	55.88MB
Avif	q10	libav-libaom	painting	1.0814s	212.1999	0.0024	4.26MB
			screenshot	0.8433s	301.4283	0.0017	3.26MB
			photo	5.1950s	459.82	0.0016	29.21MB
	q2		painting	1.6353s	140.3305	0.0083	14.61MB
			screenshot	1.2959s	196.1526	0.0054	10.53MB
			photo	10.0046s	238.7679	0.007	126.73MB
	ls		painting	4.7294s	48.5214	0.0399	69.92MB
			screenshot	3.1763s	80.0254	0.0222	42.96MB
			photo	27.0594s	88.2794	0.0233	424.44MB

Uncovered graphical format:

Common or used in specific field:

gif, ico, xmb, dds, bmp, pnm, flif, rast, tga, s3tc, bc45, etc, pvrtc, eac, dxtc,

svg, raws/container(psd/clip/ai/pdf/epsxcf/etc)

uncommon:

Bgp, fits, pcx, jpeg-xr, sgi, pik, ecw, jpeg-xt, …. +100 more

The Future of Image Format:

JPEG, PNG

—----------

3. Expected deliverables and a rough biweekly time schedule

   Oct. 21: Background research

   Nov. 4: Midterm update

   Nov.20: Demo and adjustment

   Dec: Final report and presentation

4. website url; any resources needed, references if any, etc

   https://www.cojan.net/

5. References:

   1. https://jpegxl.info
   2. An image format for the Web
   3. Technical Introduction to OpenEXR
   4. "BT.2100 : Image parameter values for high dynamic range television for use in production and international programme exchange"