Adaptive Edge-Aware Image Compression: Enhancing JPEG with Dynamic Block Partitioning and Quantization

by

Fevzi Babaoğlu

June 2025

ABSTRACT

Adaptive Edge-Aware Image Compression: Enhancing JPEG with Dynamic Block Partitioning and Quantization

The standard JPEG algorithm's reliance on fixed-size blocks causes conspicuous blocking artifacts, particularly at higher compression levels. This project addresses these limitations by developing an innovative, adaptive, edge-aware image compression system that dynamically adapts block sizes and quantization levels in response to local image content. The primary innovation is the use of edge detection to control block partitioning adaptively, so that smaller blocks capture detail in high-edge-density regions, and larger blocks efficiently compress smoother regions. This, together with adaptive quantization based on block size and user-adjustable quality targets, improves JPEG by dynamically controlling block size and quantization. The system accommodates multiple color spaces and user control through an easy-to-use GUI, enabling flexible experimentation and intuitive interaction for both novice and advanced users.

Evaluation against baseline JPEG on classic test images and the LIVE Image Quality Assessment Database, based on metrics like compression ratio, PSNR, SSIM, MS-SSIM, and LPIPS, confirmed its efficacy. When compression prioritized, the adaptive approach provided 25-55% file size savings while typically maintaining or even improving image quality. When quality prioritized, it produced significant 10-30% quality increases, often without file size increases. The evaluation also highlighted the effectiveness of color spaces such as OKLAB and YCoCg and the robustness of 4:2:0 chroma subsampling. These factors, along with the overall performance benefits, certify the system's potential as a more efficient, perceptually aware JPEG alternative.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

AC	Alternating Current
AEÜ	Archiv für Elektronik und Übertragungstechnik
AHE	Adaptive Histogram Equalization
AJPG	Adaptive JPEG
ANS	Asymmetric Numeral Systems
App	Application
AVC	Advanced Video Coding
bpp	Bits Per Pixel
BPG	Better Portable Graphics
CC	Creative Commons
CIE	Commission Internationale de l'Éclairage
CLAHE	Contrast Limited Adaptive Histogram Equalization
CODEC	Coder-Decoder
CPU	Central Processing Unit
CV	Computer Vision
CVF	The Computer Vision Foundation
dB	Decibels
DC	Direct Current
DCT	Discrete Cosine Transform
DWT	Discrete Wavelet Transform
ECCV	The European Conference on Computer Vision
EOTF	Electro-Optical Transfer Function
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HDR	High Dynamic Range
HE	Histogram Equalization
HEVC	High Efficiency Video Coding
HP	Hewlett-Packard
HVS	Human Visual System
I/O	Input/Output
I2CT	International Conference for Convergence of Technology
IDCT	Inverse Discrete Cosine Transform
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization

ITU	International Telecommunication Union
ITU-R	ITU Radiocommunication Sector
JIT	Just-In-Time
JND	Just Noticeable Difference
JPEG	Joint Photographic Experts Group
JSON	JavaScript Object Notation
LPIPS	Learned Perceptual Image Patch Similarity
LIVE	Laboratory for Image & Video Engineering
MPEG	Moving Picture Experts Group
MS-SSIM	Multiscale Structural Similarity Index Measure
MSE	Mean Squared Error
PIQ	Perceptual Image Quality
PNG	Portable Network Graphics
PQ	Perceptual Quantizer
PSNR	Peak Signal-to-Noise Ratio
QF	Quality Factor
Rec.	Recommendation
RGB	Red Green Blue
sRGB	Standard Red Green Blue
SAIL	Stanford Artificial Intelligence Laboratory
SSIM	Structural Similarity Index Measure
VGG	Visual Geometry Group
WebP	Web Picture format
WCG	Wide Color Gamut
1D	One-dimensional
2D	Two-dimensional

1. INTRODUCTION

Digital images are an integral part of today's communication, infiltrating every aspect of our daily lives. The amount of visual information created and exchanged is immense; an estimated 1.94 trillion images were captured in the year 2024 alone [1], which is roughly equivalent to 61450 images per second. This material is then widely disseminated across different platforms with daily sharing numbers reaching about 6.9 billion images on WhatsApp, 3.8 billion on Snapchat, 2.1 billion on Facebook, and 1.3 billion on Instagram [1]. Such a large amount of visual data requires effective image compression methods. To put the load of data into perspective, a single uncompressed Full HD (1920x1080 pixels) color image at 24 bits per pixel requires 6.2 Megabytes of storage [2]. The 61450 images that are captured each second would, if all uncompressed Full HD images, require more than 380 Gigabytes of storage each second. Without effective compression, the storage demands for this amount of imagery would be unreasonable, and transmission capabilities would be overwhelmed. So, image compression is not merely a technical convenience but an enabling technology that facilitates the seamless flow and consumption of visual information in the digital era, making rich visual content both manageable and widely accessible. Although lossless formats such as PNG (Portable Network Graphics) preserve every detail, they offer more limited file size reduction for photographic content. Conversely, lossy formats such as JPEG (Joint Photographic Experts Group) formats, by selectively discarding some information less noticeable by human eye, establish JPEG as the de facto standard for photographic images across the web and digital devices for decades [2].

In spite of its popularity and widespread use, the standard JPEG algorithm, which relies on dividing an image into fixed-size 8x8 pixel blocks for its Discrete Cosine Transform and subsequent quantization, exhibits well-known limitations. These are especially evident at higher compression ratios, when blocking artifacts may occur along the boundaries of the 8x8 grid, and detailed texture or small details in the image may be lost or excessively blurred. The uniform quantization of the whole image, scaled by a single quality factor, fails to consider the different complexity and perceptual significance of different image regions. Perceptually important complex and textured regions may not receive enough bits to preserve their integrity and may be over-quantized and consequently end up with blocking artifacts. Enhancing these aspects can bring significant advantages: a better trade-off between visual quality and file size reduction, i.e., smaller file sizes for the same visual quality or better quality images for the same file sizes. That would lead to lower storage costs for consumers and large services, less bandwidth usage for data transfer, and an overall enhanced visual experience for consumers, with fewer distracting artifacts and better preservation of critical

image features like edges and textures. Addressing these shortcomings is crucial to satisfy the growing demand for high-quality visual content in a world constrained by storage and bandwidth.

This project addresses these challenges by proposing an ”Adaptive Edge-Aware Image Compression” system. The general concept is to augment the standard JPEG pipeline with dynamic block partitioning and adaptive quantization techniques depending on local image content, more specifically edge information. Rather than JPEG's fixed 8x8 block approach, the new system begins by examining the image to determine edges and regions of high detail. According to this analysis, the image is divided into blocks of varying sizes; small blocks are employed to capture intricate details in high edge density regions, thereby preserving their sharpness, and large blocks are employed to effectively compress smoother, less complex areas. Following this adaptive partitioning, quantization levels are dynamically adjusted for these blocks according to their size and user-specified quality targets. This provides for intelligent allocation of bits, increased fidelity where there is visual importance and more extreme compression where it will be less noticeable. Additionally, the system supports modern color spaces, which can provide greater decorrelation of color components or better perceptual uniformity than conventional YCbCr, and provides users with an easy-to-use graphical user interface to configure compression parameters as needed and see real-time previews. The ultimate objective is to create a more perceptually aware and versatile compression system than standard JPEG with the aim of achieving improved compression efficiency and image quality.

Accordingly, this report is structured as follows. Chapter 2 reviews an overview of the related background on JPEG improvements, other similar adaptive schemes to the one being proposed, machine learning for compression, and pre/post-processing methods. Chapter 3 records the methodology of the system, including standard JPEG algorithms, color space conversion, image processing filters, quadtree partitioning, and quality measures. Chapter 4 explains project analysis and design, including the system's functional and non-functional specifications, and its architectural design with use-case diagrams and sequence diagrams. Chapter 5 describes the implementation details, such as overall system architecture, color conversion, step-by-step compression and decompression pipeline processes, and graphical user interface development. Chapter 6 explains the process of testing and evaluation, including datasets used, performance measures tracked, baseline benchmark, various comparison approaches, and in-depth results analysis. Lastly, Chapter 7 concludes the report by summarizing the important findings and project contributions, explaining how the proposed system fulfills its objectives, and providing recommendations on possible future research and development in adaptive image compression.

2. BACKGROUND

This section provides an overview of significant advances in JPEG algorithms, adaptive compression algorithms, machine learning algorithms, and pre/post-processing methods eliminating JPEG compression artifacts while maintaining or improving image quality.

2.1. Improving JPEG Algorithm

Alter, Durand, and Froment [3] introduced a JPEG artifact removal algorithm based on adapted total variation (TVα) minimization with weights that vary spatially (5.0-10.0 near block edges, 1.0-2.0 within blocks) to overcome the 8×8 DCT block pattern. Their method solves a constrained optimization problem which retains the original quantized DCT coefficients and optimizes the modified TV functionality in a subgradient descent optimization with projection on the constraint set. Although theoretically proven to converge within an infinite number of iterations, experiments give best performance with very few iterations for compression ratios below 30. Such computational efficiency allows the method to be generally applicable with both higher visually and quantitatively improved outcomes than standard JPEG decompression (for Lena: 1.17 dB PSNR improvements at compression ratio 30, to 30.39 dB from 29.22 dB with traditional decompression) without sacrificing important image information like edges and textures, or introducing blur or perceptual detail loss.

Douak, Benzid, and Benoudjit [4] developed a DCT-based image compression algorithm of color images using adaptive block scanning for improvement of the efficiency of compression with high-quality images. It applies color transformation, block-based DCT, and iterative thresholding by the bisection method for achieving a certain image quality in terms of PSNR. The key contribution lies in their adaptive scanning technique, which scans each DCT block using four different scanning techniques (see Figure 2.1) to identify the encoder that produces the maximum sequence of zeros at the end of each block. It is combined with a proposed two-role encoder (TRE) [5] and Huffman coding [6] for further compression. Experimental testing on standard test images showed their algorithm surpassed the standard JPEG approach, achieving comparable PSNR values with considerably lower bit rates. Experimental result indicated that 16×16 blocks had the optimum performance in YCbCr domain with an average PSNR of 32.56 dB at only 0.90 bpp for test images compared to JPEG's 32.57 dB at 1.03 bpp, demonstrating a practical algorithm for efficient compression of color images with around 13% greater efficiency in compression.

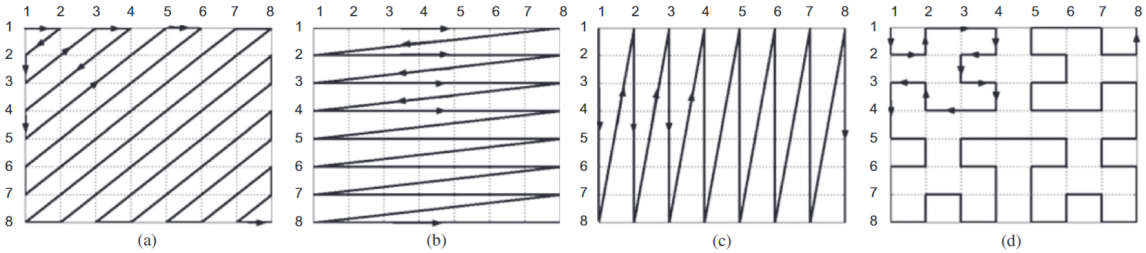

Figure 2.1. The different ways for DCT Blocks scanning: (a) zigzag, (b) horizontal, (c) vertical, and (d) Hilbert. [4]

Lin and Dong [7] introduced an adaptive downsampling algorithm that significantly improves low bit rate image compression. They choose best downsampling adaptively in terms of local visual importance, compared with the previous methods which used uniform downsampling ratios. Every 16×16 macroblock is handled by the algorithm and one of four downsampling modes (DSM) is selected from: no downsampling (DSM_0), 1/2 downsampling in vertical direction (DSM_1), 1/2 downsampling in horizontal direction (DSM_2), or 1/4 downsampling in both directions (DSM_3). For each macroblock, the algorithm first calculates coding distortion from standard JPEG compression as a starting point. Then it compares other DSMs paired with appropriate quantization modes ((QM_0)-QM_3) to determine the best pair of minimizing mean squared error without increasing nonzero DCT coefficients. This helps for more quality quantization of retained coefficients within the same bit budget. When the downsampling mode is selected, the algorithm does not apply downsampling along directions of high spatial variation as a means of preserving perceptually important features. They simply reconstruct full-resolution DCT coefficients from the downsampled coefficients at the decoder by using derived mathematical formulas, without spatial interpolation being a choice. Experiments proved that their method improved the critical bit rate (maximum bit rate at which downsampling performs better than traditional methods) to 1.2 bpp (from 0.15-0.41 bpp in the literature) with PSNR improvements up to 3.4 dB at extremely low bit rates (< 0.2 bpp).

Alakuijala et al. [8] developed Guetzli, a perceptual-guided JPEG encoder with better compression ratios than current well-known encoders and has visually equivalent image quality. Guetzli employs Butteraugly, its perceptual distance metric, to allocate bits wisely according to three significant visual features: yellow light to blue sensitivity, decreased spatial resolution of the human eye for blue, and visibility of fine structures as a function of surrounding visual activity. A two-stage optimization approach is employed by the encoder: first tuning global quantization tables, and then selective zeroing of DCT coefficients that are less visually important in every block. The zeroing coefficient accounts for around

90% of the compression quality of Guetzli by synchronizing block substitution in adjacent blocks to create visually consistent output. This creates more even quality loss, favoring preservation for perceptually significant areas and applying higher compression to regions that are less perceptible. Although the output of Guetzli is actually standard-compliant JPEG, its extremely high computational complexity renders Guetzli best for compression of static content where encoding time is not crucial. Benchmark comparisons against libjpeg and mozjpeg showed significant improvements, achieving 29-45% file size reduction at similar perceptual quality.

Abu and Ariffin [9] presented a novel psychovisual model for near-lossless compression of images based on psychoacoustic threshold principles applied to the visual realm. Their approach applies a large two-dimensional DCT directly to whole 240×320 pixel rectangular image frames rather than the standard 8×8 JPEG blocks. The algorithm replaces the usual quantization tables with psychovisual threshold curves as just noticeable difference (JND) parameters, which determine the most favorable frequency signals to be maintained based on human sensitivity to vision. The work uses a scale factor coding technique borrowed from audio processing, where frequency signals are divided into 64 coefficient blocks, with the highest value of each block serves as a scale factor. Experimentation using 96 images showed improved performance over standard JPEG compression: PSNR of 37.40 dB (natural images) and 36.42 dB (synthetic images) with similarity measures of virtually 99.5%. Compression efficiency was comparable with high-quality JPEG (Q75) with DC/AC luminance bit lengths of 4.86/2.77 bits as opposed to JPEG's 5.67/1.62 bits, but with improved visual quality and no blocking artifacts.

Wang, Lee, and Chang [10] have suggested a systematic approach to generating JPEG quantization tables based on human visual system (HVS) models. The algorithm uses an HVS model and a uniform quantizer to create perceptually optimal quantization tables for JPEG compression. In their method, the vision system is modeled as a modulation transfer function (MTF) that accounts for varying sensitivity across different spatial frequencies and viewing angles. Applying the HVS frequency weighting matrix to DCT coefficients, investigators built quantization tables where every visual contribution is weighted according to its perceptual importance. Unlike display resolution-independent standard JPEG tables, their approach can easily be tailored to specific viewing conditions and display resolutions. Testing on the standard 512×512 images achieved superior performance compared to JPEG default tables with improvements of 0.2-2.0 dB in PSNR for various compression rates (0.25-1.0 bpp) and superior performance compared to several other state-of-the-art compression methods without increased computational complexity in encoder or decoder.

2.2. Similar Adaptive JPEG Algorithms

Feischl and Hackl [11] have introduced an adaptive image compression algorithm that enhances JPEG compression by making use of optimum mesh refinement techniques over the traditional fixed size 8×8 pixels blocks. Their approach integrates concepts from adaptive finite element methods to image compression and divides images into content complexity-dependent variable-sized rectangular elements. The algorithm starts with a coarse grid and progressively refines it by selecting elements with the highest modified error value, splitting the chosen elements into four children. This is done until the compression error is below a specified tolerance. One of the significant mathematical contributions is their proof that the ”refinement property” is satisfied with high probability (greater than 99.99% with appropriate parameters), offering an almost optimal image subdivision. The authors demonstrated mathematically that their algorithm produces quasi-optimal (almost optimal) subdivision grids with near-minimal approximation error. On empirical tests, their adaptive compression achieved substantial storage savings on large JPEG files with negligible loss of quality, reducing file sizes by up to 30-40% compared to regular JPEG compression.

Ernawan, Abu, and Suryana [12] designed an adaptive JPEG image compression algorithm using a psychovisual model to improve the compression efficiency compared to standard JPEG algorithms. Although JPEG compression generally applies the quality factors (QF) to scale quantization tables uniformly, it degrades image visual quality at higher compression levels. The researchers created custom quantization tables based on human visual system characteristics by investigating the contribution of each DCT coefficient to image reconstruction. The authors derived mathematical functions that yielded adaptive quantization tables to maintain visually significant information, particularly protecting DC coefficients that contain vital visual information. Experimentation revealed that their adaptive psychovisual threshold approach delivered improved quality reconstructed images at lower bit rates compared to standard JPEG compression. Visual inspection on enlarged images guaranteed perceptually enhanced quality output with minimal compression artifact.

Rosenholtz and Watson [13] designed a perceptually adaptive JPEG coding scheme that optimizes compression by setting quantization dynamically at the block level. Their scheme extends the JPEG standard by scaling the quantization matrix for each 8×8 block according to a contrast-sensitive, light-adaptation, and contrast masking visual model. Unlike texture-based approaches, their algorithm selects multipliers that produce constant perceptual error across image blocks and maintain low bitrate. Their experiments showed a 22% bitrate reduction (1.29 to 1.01 bpp) while perceptual quality is maintained. Interestingly, their analysis showed that luminance masking contributed to the majority of the adaptive coding

gain, contrary to the common industry practice of using texture-based activity measures. This finding challenges existing adaptive coding practices and suggests that perceptually-optimized multipliers can enhance compression efficiency.

Ramos and Hemami [14] developed "quadpression," an edge-adaptive JPEG compression algorithm that provides improved image quality at comparable bit rates with standard JPEG. The method adopts a two-stage strategy to identify and rank visually relevant regions: first, a Canny edge detector identifies edge pixels after Gaussian blurring; second, quadtree decomposition recursively divides the image based on edge occurrence (used at a 5% threshold), dividing subimages into smaller blocks only if they hold significant edge content. This yields a hierarchical organization where smaller sub-images (8×8, 16×16) are more active visually and larger ones (32×32, 64×64) less active, as one can see in Figure 2.2. Unlike uniform Q-factor scaling for regular JPEG, quadpression applies different quantization scaling factors for DCT coefficients based on the size of the sub-image containing each 8×8 block, more precise quantization in smaller (more active) regions and less precise quantization in larger (less active) regions. The DC coefficient is not quantized at all to minimize blocking artifacts. Experiments on standard images showed that quadpression generates rate-distortion curves within 0.015-0.03 bpp of standard JPEG but with noticeably better visual quality, particularly around strong edges and textures, and reduced blockiness in flat regions.

Figure 2.2. The edges and the quadtree decomposition of the Lena image [14]

Ouyang et al. [15] created a dynamic JPEG CODEC based on an adaptive quantization system to achieve optimized frame storage with high-quality images. The core algorithm uses a two-step process: one, it uses a four-channel multiplexing structure where the grayscale images are processed through two parallel channels while the chrominance information is down-sampled in separate channels, in which its system structure is shown in Figure 2.3; two, it uses an adaptive quantization process that dynamically adjusts compression quality based on previous frame storage usage. This feedback loop maintains storage at around 80% by modifying quantization table coefficients and encoding the quality level as a prefix in the data stream header. The system also utilizes a ping-pong buffer organization using dual-port,

dual-frequency SRAMs (Static Random-Access Memory) for efficiently bridging various clock domains, where one SRAM writes (PING mode) and another reads (PONG mode). Implementation results showed a reduction in the frame storage of 76.60% and a boost in the latency of 77.10% compared to the baseline JPEG CODEC. The system can support as high as 1080p image compression and a total of six levels of compression quality, the highest being at 87.5%. Their approach accommodates real-time processing with consistent storage utilization while maintaining image quality within tolerable ranges.

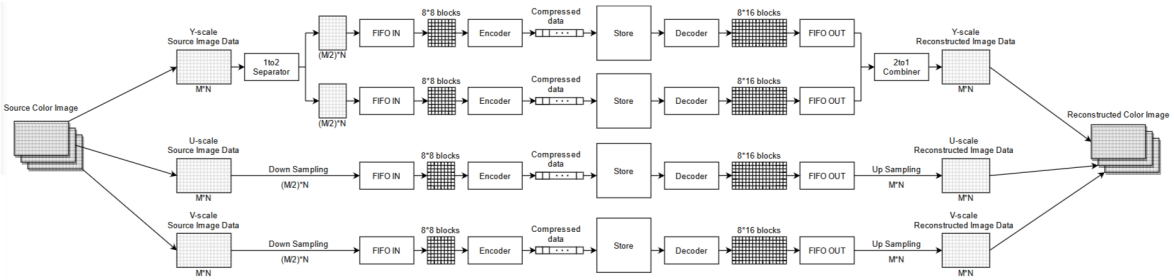

Figure 2.3. System architecture [15]

2.3. ML Based Image Compression Algorithms

Rippel and Bourdev [16] suggested an algorithm for compression through a neural network that outperforms all existing codecs in terms of real-time computational efficiency. They used an autoencoder structure with three new components: a pyramidal feature extraction system that scans images in an array of scales, an adaptive coding block that efficiently quantizes and encodes features, and a codelength regularization system that balances compression efficiency and image quality. The system recursively examines images at different scales, aligns data across these scales, quantizes the resulting features to 6-bit precision before submitting them to adaptive arithmetic coding. They also enhanced visual quality through adversarial training with a multiscale discriminator network. Benchmarking against standard datasets showed their method produces files 2.5 times smaller than JPEG and JPEG 2000, 2 times smaller than WebP, and 1.7 times smaller than BPG across all settings and yet at still amazingly fast processing speeds (approx. 9ms encode, 10ms decode on GPU)—the first ML-based compression algorithm to surpass commercial codecs while executing in real-time.

Ballé, Laparra, and Simoncelli [17] presented an end-to-end rate-distortion optimized image compression scheme with nonlinear transforms that outperforms JPEG and JPEG 2000. Their method combines a nonlinear analysis transform, uniform quantization, and nonlinear synthesis transformation with biologically-inspired generalized divisive normalization (GDN) as part of its feature. The system jointly optimizes all parameters for rate-distortion

performance using stochastic gradient descent, employing a continuous relaxation strategy in order to avoid nondifferentiability while training. Testing showed superior rate-distortion performance to standard codecs, with their methodology achieving superior PSNR values at lower bit rates (e.g., 27.01 dB versus 26.61 dB for JPEG 2000 at 0.113 bit/px). Above all, the approach has impressive visual quality improvements across all the test images and bit rates, particularly preserving edge contours and eliminating blocking artifacts, with MS-SSIM values always greater (e.g., 0.9039 vs. 0.8860 for JPEG 2000 at comparable bit rates), confirming both subjective and objective quality gains.

Reich et al. [18] introduced a new differentiable approximation of JPEG image compression that is much better than other methods. Their method produces a differentiable equivalent of JPEG that closely resembles the standard non-differentiable version for all quality levels and with high performance particularly at lower qualities where previous methods fail. In reviewing existing methods, they discovered significant flaws in handling discretization processes and rounding approximations. Their method uniquely incorporates differentiable approximations for critical JPEG operations like quantization, scaling of quantization tables, and image clipping, while maintaining gradient information at all stages. The approach is differentiable with respect to input images, JPEG quality, quantization tables, and color conversion parameters. Their implementation outperforms previous differentiable methods by 3.47dB PSNR on average across quality settings and by 9.51dB PSNR for extremely compressed images.

2.4. Evaluation of Preprocessing and Post-Processing for Image Compression

JR and SP [19] designed a better edge detection method for image segmentation compared to conventional methods. Their approach enhances the conventional Canny edge detection method by incorporating histogram equalization techniques (in particular CLAHE) along with noise reduction filters (bilateral/Gaussian) as preprocessing steps, as shown in Figure 2.4. Through extensive experimentation on various samples of images, they showed that the ensuing combined algorithms yield improved edge detection along with improved values of PSNR. EQU+GAUSSIAN+CANNY delivered a 66.89% improvement in PSNR over other methods in the literature. Additionally, for various datasets, CLAHE+BILATERAL+CANNY and CLAHE+GAUSSIAN+CANNY yielded similar results. The researchers demonstrated that adaptive equalization techniques with adequate filtering are better compared to conventional edge detection techniques in maintaining edge integrity and minimizing added noise in the histogram equalization process.

Figure 2.4. Proposed edge detection technique [19]

Zhang and Gunturk [20] suggested an adaptive bilateral filtering method for compression artifact reduction, which adjusts filter parameters automatically based on image content. In their method, first, they identify the texture regions by calculating block standard deviations and block boundary discontinuities from edge transition measurements. It then adjusts the bilateral filter's spatial parameter (σd) inversely proportional to the texture strength to preserve details, and adjusts intensity parameter (σr) proportionally to boundary discontinuity measurements to eliminate blocking artifacts. The detected discontinuities are then smoothed across blocks using linear interpolation to prevent artifact shifting. This adaptive approach delivered improved PSNR metrics (30.49 dB compared to 30.37 dB for standard bilateral filtering at 0.22 bit-rate) and greatly improved blocking artifact metrics with MSDSt values of 526 compared to 3841 for standard filtering at 0.20 bit-rate. The approach is optimally balanced between artifact minimization and detail preservation with low filter thresholds for the overall image and more aggressive filtering utilized where required. Visual assessment confirmed the adaptive method preserves textures at a higher level and minimizes blocking artifacts compared to other methods.

Kawar et al. [21] developed an approach for JPEG artifact correction using Denoising Diffusion Restoration Models (DDRM). They generalize diffusion models to nonlinear inverse problems by representing JPEG decoding as a "pseudo-inverse" of JPEG encoding. The algorithm replaces linear operators in the update rule of DDRM by JPEG encoding/decoding operations so that it can make use of pretrained unconditional diffusion models without requiring task-specific training. In ImageNet-1K tests, their method surpassed baseline JPEG decoding for all quality factors significantly, with PSNRs of 2.44 dB for QF=5 (25.90 dB vs. 23.46 dB), 2.25 dB for QF=10 (28.37 dB vs. 26.12 dB), and 2.05 dB for QF=30 (31.58 dB vs. 29.53 dB). Their approach demonstrated exemplary generalization ability, maintaining high quality at extremely low quality factors and generalizing to other non-linear inverse problems such as image dequantization.

Li et al. [22] demonstrate a system that eliminates JPEG compression artifacts while enhancing image contrast, addressing the issue of how contrast enhancement can inadver-

tently over-enhance compression artifacts. Their technique involves three principal steps (see Figure 2.5). First, they decompose the image into the structure and texture layers with total variation regularization (λ values 0.02-0.05 depending on compression level). Second, they regularize the structure layer containing few artifacts directly. Third, they decompose the texture layer through a novel two-stage approach: (1) building a scene detail mask through DCT coefficient analysis to isolate regions of actual details versus artifacts, followed by soft matting (masking with soft edges) to fit this mask to structure boundaries; and (2) selectively deblocking at 8×8 boundary positions in areas of details and completely eliminating content in non-detail regions. Finally, they reconstruct the layers with a scale factor that is either derived from Taylor series approximation for tone control or optical model for dehazing algorithms. Compared to traditional deblocking methods used as pre/post-processing, their integrated approach delivered better SSIM scores and qualitative results, while maintaining faster runtime (15s vs 20-287s for other algorithms).

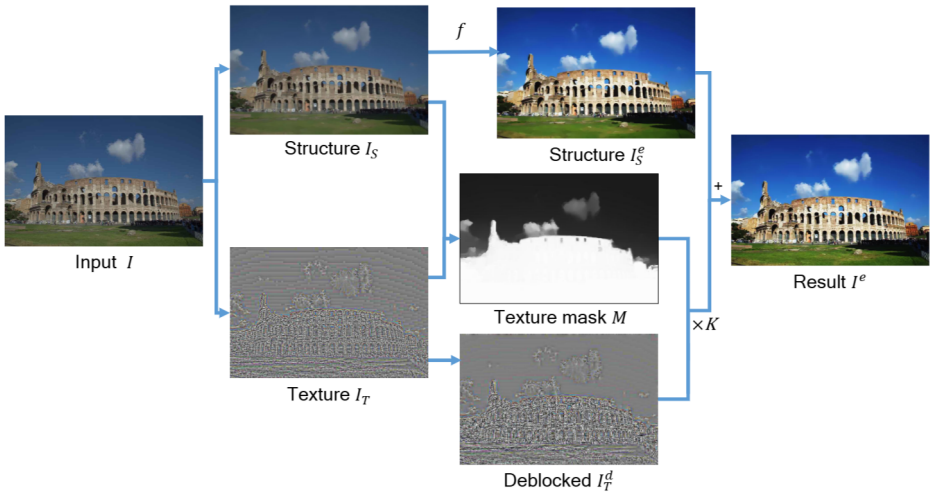

Figure 2.5. The overview of the proposed method. [22]

The advances of JPEG optimization, adaptive compression architectures, and artifact removal algorithms reflect the ongoing pursuit for balance between efficiency of compression and perceptual quality of images. The improvements provide a significant platform to address the limitations of traditional techniques to preserve fine details at high compression ratios. Supported by this platform, the following Methodology section describes the technical framework used by the proposed adaptive JPEG compression system. It elaborates on the integration of standard JPEG workflows with adaptive preprocessing strategies, color space transformations, and quality measures based on perception—components needed to maximize compression and reduce artifacts.

3. METHODOLOGY

This chapter describes the fundamental techniques involved in adaptive JPEG compression techniques. It covers standard JPEG image compression techniques, color space transformation, other image preprocessing techniques utilized, and various quality metrics.

3.1. Standard JPEG Algorithms

Standard JPEG algorithms form the foundation of lossy digital image compression. This section describes three primary standards: the original JPEG format that employs discrete cosine transform (DCT) and quantization, JPEG2000 which employs wavelet transforms, and JPEG XL, the latest development in JPEG standards.

3.1.1. JPEG

JPEG (ISO/IEC 10918) [23], developed by the Joint Photographic Experts Group, has proved to be the de facto image compression standard due to its high compression ratio, good visual quality, and computational efficiency. JPEG, in contrast to lossless images like PNG, compresses more, and therefore it is suitable for web graphics, digital photos, and applications that handle large volumes of images.

One of JPEG's strongest points is its speed, a necessity for real-time applications. The Discrete Cosine Transform (DCT), on which JPEG compression is based, is a computationally efficient function that could be calculated efficiently even on low-power hardware. Another important advantage is taking advantage of the human perception in order to compress data, since the human vision system is highly sensitive to luminance over chrominance. By color space conversion, JPEG allows subsampling of color information, enabling high compression at minimal loss of perception. Variable quality control is also supported by JPEG such that users can control the tradeoff between image quality and file size. This makes it suited for applications ranging from high-quality image capture to transmission over low-bandwidth channels. Last but not least, JPEG is widely supported. JPEG images can be encoded and decoded with nearly all digital hardware and web browsers.

JPEG compression follows a multistep process that progressively reduces file size while preserving essential visual information. The encoder and decoder frameworks of JPEG can be seen in Figure 3.1 and Figure 3.2, respectively.

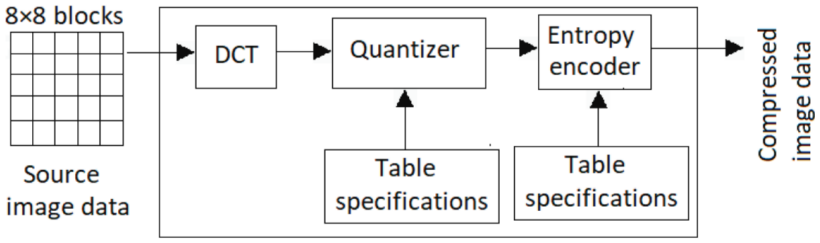

Figure 3.1. JPEG image coding encoder framework [24]

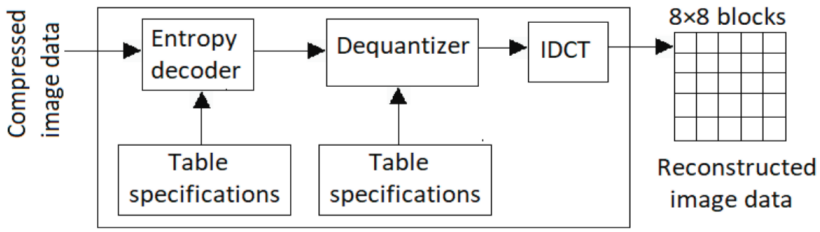

Figure 3.2. JPEG image coding decoder framework [24]

The JPEG encoding process, as illustrated by the framework in Figure 3.1, unfolds through the following key stages:

- **Color Space Conversion:** Most digital images are in RGB color space. JPEG first converts them into YCbCr color space (see Section 3.2.3) to separate luminance (Y) from chrominance (Cb, Cr).
- **Chroma Subsampling:** For data reduction, JPEG applies chroma subsampling, typically 4:2:0, where the chrominance components are maintained at half the resolution of the luminance component. This compresses the file size significantly with imperceptible loss of quality.
- **Block Splitting:** The image is divided into 8×8 pixel blocks, and each one of them is treated separately. Locality here makes the Discrete Cosine Transform (DCT) easier and facilitates efficient compression.
- **Discrete Cosine Transform (DCT):** DCT is computed for each 8×8 block, which converts the spatial pixel values into frequency components (see Figure 3.3). The transformation concentrates most of the energy in low-frequency coefficients. The mathematical expression for the transformation is given in Equation 3.1.

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \quad \text{for } k = 0, \ldots, N-1. \tag{3.1}$$

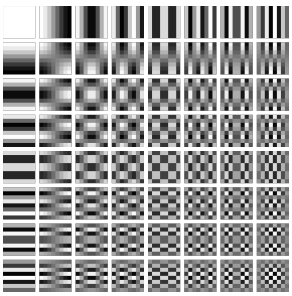

Figure 3.3. Two-dimensional DCT frequencies from the JPEG DCT [25]

- **Quantization:** Quantization reduces the precision of DCT coefficients by dividing them by a predefined quantization matrix and rounding. Low-frequency components are preserved with higher precision, while high-frequency details (fine textures, noise) are heavily quantized.
- **Zigzag Ordering & Entropy Coding:** Zigzag ordering sequences DCT coefficients in a way that arranges zeros for more effective compression (see Figure 3.4). Lossless Huffman coding [6] is utilized afterward to obtain the final data.

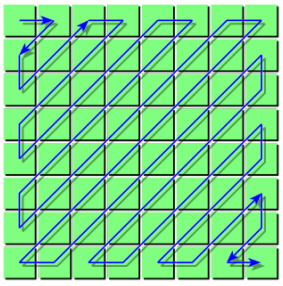

Figure 3.4. Zigzag ordering of JPEG image components [26]

Although effective, JPEG has a number of drawbacks. Since images are processed in 8×8 blocks, noticeable blocking artifacts can occur at high compression rates. High-frequency detail (textured areas, fine lines) is lost due to over-quantization. Chroma subsampling can lead to visible color bleeding in gradients due to YCbCr's inefficient separation of luminance and chrominance.

JPEG is the most widely used image compression standard due to its tradeoff between quality, efficiency, and speed. Despite its limitations, its universal support and adaptability ensure that JPEG will remain as de facto standard for a significant time in the future [2].

3.1.2. JPEG2000

JPEG 2000, standardized as ISO/IEC 15444 [27], is an image compression standard that has been created to supersede the original JPEG standard with greater compression capability and a feature set. JPEG 2000 uses the Discrete Wavelet Transform (DWT) in place of the Discrete Cosine Transform (DCT) used in the original JPEG and provides higher compression ratios and image quality. The encoder structure may be seen in Figure 3.5.

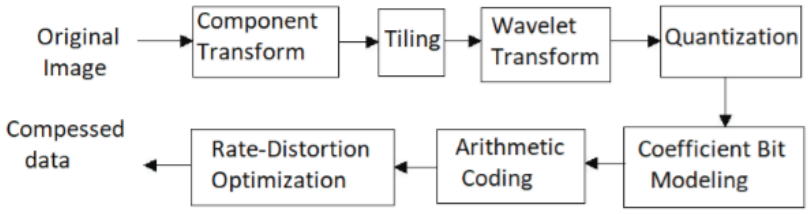

Figure 3.5. JPEG 2000 image coding encoder framework [24]

The standard also supports progressive image decoding, enabling images to be viewed at progressively greater and greater resolution and quality as more data is received. It also employs advanced entropy coding techniques, such as Embedded Block Coding with Optimal Truncation (EBCOT), to achieve higher compression ratio.

Though it is technically superior, JPEG 2000 has seen limited application in consumer hardware, primarily due to computational intensity and inadequate hardware and software support. It has, however, seen niche applications in fields like medical imaging, digital cinema, and image archiving.

3.1.3. JPEG XL

JPEG XL [28], designed as an improved successor to standard JPEG, enhances it by incorporating a variable-block-size Discrete Cosine Transform (DCT), supporting block sizes from 2×2 to 256×256, in contrast to the fixed 8×8 blocks of standard JPEG. Entropy coding is significantly improved using Asymmetric Numeral Systems (ANS) [29] instead of Huffman coding [6] for more efficient compression. Modular encoding is another significant aspect where different parts of an image are processed separately, improving the efficiency of parallel decoding and progressive rendering. JPEG XL also allows for lossless recompression of existing JPEG images. This makes it possible to transition smoothly from JPEG with full backward compatibility.

3.2. Color Space Transformations

Color space transformations are essential to achieve effective JPEG compression. The conversion of RGB images to luma-chroma separated color spaces enables chroma subsampling, exploiting the human visual system's insensitivity to color details as opposed to brightness, resulting in considerable file size reduction without perceptual quality degradation.

3.2.1. Color Space: Standard RGB (sRGB)

sRGB (Standard Red Green Blue) [30] is a widely adopted, de facto color space developed by HP and Microsoft in 1996 and later standardized by the IEC. It was developed to match typical display characteristics of consumer-grade monitors and printers and hence became the default color space for web-based applications, digital cameras, and other imaging devices. It is based on the Rec. 709 primaries with a D65 white point and employs a non-linear gamma correction that is consistent with human visual perception.

The design choices underlying sRGB, including its specific gamut and tone reproduction characteristics, reflect a balance between device compatibility and color accuracy. While its gamut is somewhat smaller than those offered by more advanced color spaces, sRGB's consistency and widespread support have made it a de facto standard for most digital imaging applications. Its role in standardizing color reproduction has been crucial for ensuring that images and videos maintain a consistent appearance across various devices and platforms.

3.2.2. Color Space: CIEXYZ

The CIE XYZ color space [31] is a foundational color space in colorimetry established by the International Commission on Illumination (CIE) in 1931. Unlike RGB and other color spaces with directly visible primaries, XYZ is founded upon three hypothetical primaries that cannot be physically achieved but provide mathematical advantages. The imaginary primaries allow all colors to be represented using only positive coefficients, eliminating the negative coefficients of the earlier color models.

The chromaticity diagram, shown in Figure 3.6, maps the gamut—the entire range of human-visible colors, as a horseshoe-shaped area with pure spectral colors on the curved edge and non-spectral purples on the straight edge. The XYZ color space encompasses the entire gamut of human color vision and serves as a device-independent reference against which other color spaces are defined and measured. In practical applications, XYZ func-

tions as an intermediate connection space when converting between different color models. The XYZ color space remains foundational in scientific color work, and standardized color measurements.

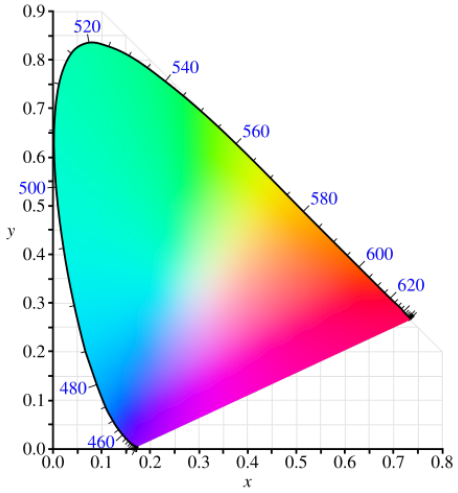

Figure 3.6. The CIE 1931 color space chromaticity diagram. [32]

3.2.3. Color Space: YCbCr

YCbCr, defined in ITU-R BT.601 [33], is a color space that separates image information into a luminance component (Y) and two chrominance components (Cb and Cr). The Y channel is luminance, and Cb is the blue-difference and Cr is the red-difference chrominance components. This is done in order to take advantage of the higher sensitivity of the human visual system to luminance than color differences. A sample image and its Y', Cb, Cr separate components are illustrated in Figure 3.7.

Figure 3.7. A color image and its Y′, CB and CR components. [34]

YPbPr (analog version of YCbCr) from gamma-corrected R'G'B' is derived as:

$$\begin{bmatrix} Y' \\ P_B \\ P_R \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & -0.418688 & -0.081312 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} \tag{3.2}$$

YCbCr provides benefit in digital image and video compression. By allocating more bandwidth to the Y channel and subsampling the chrominance components (usually in 4:2:2 or 4:2:0 formats), systems can reduce data with little noticeable loss of quality. This forms the basis of most compression standards such as JPEG, MPEG, and H.264/AVC.

3.2.4. Other Relevant Color Spaces

Modern imaging demands more computationally efficient and perceptually effective color representations than traditional YCbCr. In this study, we review alternative, modern color spaces that offer advantages in computational efficiency, perceptual uniformity, and HDR and wide gamut capability for new compression and display applications.

YCoCg [35] is a color space that separates image data into a single luma component (Y) and two chroma components, Co (offset orange) and Cg (offset green). It is also less computationally intensive because RGB to YCoCg can be done with only additions and bit-shifts. With a theoretical coding gain of 4.21 dB over approximately 3.5 dB for YCbCr variants, the performance of YCoCg is superior and has been added in professional video compression standards like H.264/AVC and HEVC for its effectiveness in preserving high-fidelity RGB details. The RGB to YCoCg components conversion is described in Equation 3.3.

$$\begin{bmatrix} Y \\ Co \\ Cg \end{bmatrix} = \begin{bmatrix} 0.25 & 0.50 & 0.25 \\ 0.50 & 0.00 & -0.50 \\ -0.25 & 0.50 & -0.25 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.3}$$

OKLab [36] is a perceptually uniform color space, meaning that numeric color value differences are more proportional to perceived differences by the human eye. It transforms traditional CIE XYZ coordinates through a series of operations, including linear matrix transformations and a non-linear operation (a cube root) on intermediate color signals. Equation 3.4 defines this whole process of transformation. Such perceptive qualities make OKLab extremely appropriate to be utilized in applications involving color manipulation processes

like color mixing, generating smooth gradients, color picking, and image editing, where predictability and visually accurate results are of the utmost importance.

$$\begin{bmatrix} L \\ a \\ b \end{bmatrix} = \begin{bmatrix} 0.21 & 0.79 & 0.00 \\ 1.98 & -2.43 & 0.45 \\ 0.03 & 0.78 & -0.81 \end{bmatrix} \left(\begin{bmatrix} 0.82 & 0.36 & -0.13 \\ 0.03 & 0.93 & 0.04 \\ 0.05 & 0.26 & 0.63 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \right)^{\frac{1}{3}} \tag{3.4}$$

ICaCb [37], designed for high dynamic range (HDR) and wide color gamut encoding, optimizes quantization to better align with human visual perception. This color space offers better luminance-chrominance decorrelation, with perceptual uniformity obtained by minimizing visible distortions in HDR content. Compared to conventional color models, ICaCb offers hue linearity and just-noticeable difference (JND) uniformity, which is very well adapted to HDR applications where visual quality matters.

ICtCp, proposed in ITU-R BT.2100 [38], is tailored for HDR and wide color gamut imaging, addressing issues such as hue shifts, luminance errors, and banding that are common in HDR content. By more evenly distributed color differences according to JND perception, ICtCp achieves smoother gradients and better hue continuity. This makes it particularly suitable for HDR broadcast, where color accuracy at high brightness levels needs to be maintained.

JzAzBz [39] is a perceptually uniform color space that works well with HDR and wide color gamut imagery. It is an improvement over older models with an even distribution of just noticeable differences (JNDs) and improved hue linearity and lightness scaling.

3.3. Smoothing Filters

Smoothing filters remove image noise and therefore enhance the reliability of edge detection by emphasizing significant structural features and repressing undesirable artifacts. Smoothing is essential in supporting the performance of subsequent edge detection algorithms, ensuring outstanding results of image salient features that are highlighted by eliminating random noises.

Gaussian filter is a smoothing filter that reduces noise and smoothes images by convolving them with a kernel derived from the Gaussian function (named after Carl Friedrich Gauss), given in Equation 3.5. This kernel assigns higher weight to pixels near the center

and progressively smaller weights to further ones, producing a natural blurring effect that preserves the overall structure of the image without high-frequency noise.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.5}$$

Bilateral filtering [40] is an advanced edge-preserving smoothing filter, adept at reducing noise while keeping sharp edges. It uniquely weights nearby pixels based on both their spatial proximity and their intensity similarity (how similar the pixel values are). Unlike traditional blurring that solely rely on spatial proximity, this dual consideration ensures that only pixels with comparable values significantly contribute to the smoothing process, thus preventing edge deformation. This makes it highly effective for tasks like denoising, texture removal, and HDR tone mapping.

3.4. Contrast Enhancement Methods

Contrast enhancement is a fundamental image processing operation that enhances the visual contrast between image regions by redistributing the pixel intensities, improving clarity, visibility of details, and overall perceptual quality.

Histogram Equalization (HE) is a contrast enhancing technique that redistributes pixel intensity values of an image to produce a flat histogram, thereby expanding the dynamic range and improving visual separability. By flattening and spreading intensity distribution (see Figure 3.8), HE enhances dark or light region details and exposes buried features. This process is particularly effective in global contrast enhancement for grayscale images or luminance channel of color images.

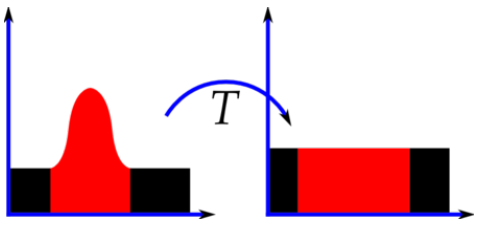

Figure 3.8. Histograms of an image before and after equalization. [41]

Adaptive Histogram Equalization (AHE) [42] is a contrast enhancement method that increases the perceptibility of an image by locally modifying contrast instead of applying a global operation to the entire image. In contrast to traditional histogram equalization that

increases contrast of the entire image, AHE computes separate histograms for small regions of the image and applies local intensity transformations, making it more appropriate for low-contrast or non-uniformly illuminated images. Local processing improves fine details but might add noise to smooth regions. AHE has uses in medical imaging, remote sensing, and computer vision.

Contrast Limited Adaptive Histogram Equalization (CLAHE) [43] is a variation of AHE that limits contrast improvement to prevent over-enhancement noise in uniform regions. CLAHE clips peak histogram values and redistributes excess pixels (refer to Figure 3.9), preventing excessive contrast with preservation of detail. CLAHE was intended initially for medical imaging but now has many usages in radiography, microscopy, and computer vision for low-contrast image enhancement.

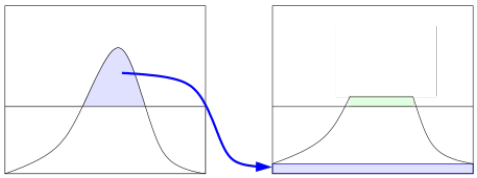

Figure 3.9. Excess redistribution in contrast-limited adaptive histogram equalization. [44]

3.5. Edge Detection Methods

Edge detection is a group of mathematical methods aimed at detecting edges, defined as curves in a digital image where the pixel intensity abruptly changes or, more formally, has discontinuities. Edge detection is an essential tool in image processing and computer vision, particularly feature detection and feature extraction.

3.5.1. Sobel Operator

The Sobel operator (Sobel–Feldman operator) [45] is a discrete differentiation technique in image processing, developed by Irwin Sobel and Gary Feldman at Stanford Artificial Intelligence Laboratory in 1968. The method employs two 3×3 convolution kernels, horizontal (Gx) and vertical (Gy), to approximate the image's intensity gradient. This gradient information is then used to identify edges, which are the abrupt changes in pixel values.

Realized as an isotropic 3×3 gradient operator, the Sobel algorithm uses integer-valued separable filters and is hence computationally inexpensive and real-time implementable. It

operates on just a 3×3 pixel neighborhood, achieving a good compromise between localized processing and minimal computational cost. The existence of rotational symmetry allows for uniform edge detection independently of orientation. Its reliance upon discrete intensity approximations remains widely applicable across many fields such as medical imaging, robotics, and digital photography.

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \tag{3.6}$$

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A} \tag{3.7}$$

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2} \tag{3.8}$$

Following the convolution of the image with the horizontal kernel (Equation 3.6) and the vertical kernel (Equation 3.7), the gradient components obtained are combined using Equation 3.8, which yields the final edge magnitude.

3.5.2. Canny Edge Detector

The Canny edge detection algorithm, proposed by John Canny in 1986 [46], is a multi-stage approach for detecting edges in images while minimizing noise and preserving structural accuracy. It defines three key criteria for an optimal edge detector: good detection (low false positives and false negatives), good localization (accurate edge placement), and minimal response (one response per edge). The method applies Gaussian smoothing to reduce noise, computes image gradients to identify edge strength and direction, and then applies non-maximum suppression to thin edges. Hysteresis thresholding is finally applied to remove weak edges while retaining strong, continuous edges.

Compared to simpler edge detectors, Canny's method optimizes detection accuracy and localization by adapting to varying noise conditions through multi-scale analysis and gradient tracking and is therefore widely used in image processing and computer vision. The application of an example image and Canny edge detector is given in Figure 3.10..

Figure 3.10. Canny edge detection applied to a photograph [47]

3.6. Quadtree Data Structure

Data structures are specialized formats for organizing, storing, and managing data. Examples include arrays, trees, and graphs, each tailored to optimize a specific task.

A quadtree [48] is a tree data structure used for searching and retrieving spatial data in two-dimensional space. It partitions an area recursively into four quadrants (refer to Figure 3.11), in which every node contains a point of data and up to four child nodes representing the space's further subpartitions. The structure supports efficient searching, insertion, and region-based queries.

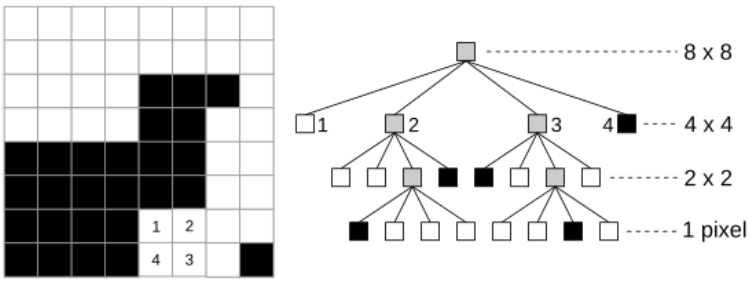

Figure 3.11. A bitmap and its compressed quadtree representation [49]

3.7. Quality Assessment Metrics

Quality Assessment Metrics are essential to assess the efficacy of compression algorithms in a way that the decrease in the size of the data does not affect the integrity or usability of the original content. The metrics offer objective criteria to quantify fidelity, identify distortions, and describe perceptual degradation, thereby allowing systematic optimization of the compression parameters.

3.7.1. Peak Signal-to-Noise Ratio (PSNR)

Peak Signal-to-Noise Ratio (PSNR) is a commonly utilized measure to quantify the quality of reconstructed or compressed signals by calculating the ratio of the maximum possible power of a signal to the noise power distorting it. Being a logarithmic quantity on the decibel scale, it provides a convenient numerical comparison between an original image and a processed one, with larger values relating to increased fidelity. While computationally straightforward and mathematically appealing, PSNR weighs pixel-level difference rather than perceptual significance, and thus it is a traditional yet limited tool for measuring lossy compression and reconstruction methods.

$$\text{MSE} = \frac{1}{m\,n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \tag{3.9}$$

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}_I^2}{\text{MSE}} \right) \tag{3.10}$$

The PSNR metric relies on Mean Squared Error, which is calculated as the mean of squared pixel differences between original (I) and processed (K) images as per Equation 3.9. Equation 3.10 then defines PSNR by calculating a logarithmic ratio of the maximum possible value of squared pixel intensity to this MSE to yield decibel value for image quality.

3.7.2. Structural Similarity Index Measure (SSIM)

Structural Similarity Index Measure (SSIM) [50] is an image quality measure that approximates the similarity of two images on a basis of individual consideration of luminance, contrast, and structural information and subsequent combination into a single measure of similarity. In contrast to mean squared error (MSE) or PSNR, which are purely pixelwise measures of difference, SSIM acknowledges that the visual system is more attentive to structural distortions than to absolute differences. The metric is based on comparing local intensity distributions over small image segments, and hence allows perceptual distortions to be assigned more significant. SSIM therefore performs well for assessing the quality of compression, denoising, and other image processing methods affecting structural integrity.

$$\text{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{3.11}$$

Table 3.1. Notations for SSIM calculation

Symbol	Description
μ_x, μ_y	Pixel sample means of images x and y
σ_x^2, σ_y^2	Sample variances of x and y
σ_{xy}	Sample covariance between x and y
c_1, c_2	Stabilization constants $(k_i L)^2$
L	Dynamic range of pixel values
k_1, k_2	Default constants (0.01, 0.03)

Equation 3.11 employs these statistical components, listed in Table 3.1, to compute image similarity. SSIM gives a more perceptually consistent assessment compared to pixel-difference-based measures by comparing local means for luminance, variances for contrast, and covariance for structural similarity.

3.7.3. Multiscale Structural Similarity Index Measure (MS-SSIM)

Multiscale Structural Similarity Index Measure (MS-SSIM) [51] is an extension of the SSIM that combines image features at various spatial resolutions by low-pass filtering and downsampling techniques. Unlike SSIM, which evaluates structural similarity at one scale only, MS-SSIM enhances this measure by comparing images at varying resolutions in order to approach human perceptual mechanisms that process images at various scales. MS-SSIM's capability in dealing with different viewing conditions, compression artifact, and high-resolution images is remarkable. Empirical testing demonstrates that MS-SSIM is more highly correlated with subjective image quality judgments of humans compared to SSIM, thus making it more appropriate for use in practical applications such as image restoration and compression.

3.7.4. Learned Perceptual Image Patch Similarity (LPIPS)

Learned Perceptual Image Patch Similarity (LPIPS) [52] is a perceptual image quality metric that is based on comparing deep feature embeddings rather than pixel-wise differences. Unlike traditional metrics such as PSNR and SSIM, LPIPS addresses image comparison based on high-level perceptual similarity by employing deep neural network features. This approach enables LPIPS to align more closely with human visual perception, making it particularly well-suited to the evaluation of image compression, super-resolution, and generative

models. LPIPS can also be fine-tuned further for improved correlation with human judgment, and hence is a robust and flexible metric for perceptual similarity tasks.

3.8. Implementation Tools

This section outlines the technologies and platforms employed to develop the proposed adaptive JPEG compression system. By leveraging modern programming paradigms and modular architectures, the chosen tools facilitate seamless integration of preprocessing, compression, and quality assessment workflows while enabling rapid prototyping and validation.

3.8.1. Python

Python, recognizable by its official logo (Figure 3.12), has become widely utilized in the area of image processing because of its extensive range of libraries supporting various computational requirements, from basic image manipulation to sophisticated algorithmic application. These libraries facilitate efficient implementation of tasks such as image transformation, filtering, and frequency-domain analysis, which are essential components of adaptive compression processes. Python's independence from architecture allows code to run on any operating system and hardware environment with ease, ensuring uniform consistency of performance regardless of the deployment platform on local PCs, servers, or cloud platforms. Python's minimalist syntax and consistent package management tools simplify dependency resolution and environment copying, facilitating collaboration and reproducibility. Quick prototyping through interactive development environments is also supported by Python for iterative testing and refinement of compression strategies.

Figure 3.12. The Python logo [53]

3.8.2. Libraries

NumPy is the fundamental library for numerical computing that enables effective manipulation of pixel data as multidimensional arrays. It supports critical operations in color space conversion, enabling effective processing of image data across formats. With its C-optimized backend and close integration with other libraries, it is a necessity for manipulating and analyzing image data at scale.

Numba speeds up color space algorithms by compiling Python functions into efficient machine code using Just-In-Time (JIT) compilation. This is essential for real-time use, like iterative color correction or batch processing, where performance bottlenecks in plain Python would otherwise impede efficiency. With loop and matrix acceleration, Numba closes the gap between prototyping convenience and execution speed.

ImageIO provides basic image I/O support with read/write capability for many formats. Its NumPy support allows direct translation of image data into arrays, minimizing the need for pre/post-processing. Pillow serves primarily to provide a bridge between image data and the GUI elements of Tkinter, converting raw pixel arrays into something Tkinter's display modules can use.

OpenCV-Python delivers a comprehensive collection of optimized image processing routines, such as frequency-domain transformation, filtering, and matrix resizing. It features robust edge detection, contrast normalization, and noise reduction algorithms that are executed by a high-performance C++ backend, delivering efficiency at both compression and decompression stages. OpenCV's versatility and efficiency render it a cornerstone for adaptive operations that entail precise matrix manipulation.

PIQ and LPIPS provide complementary metrics for the evaluation of compression quality. PIQ focuses on structural and pixel-wise quality metrics (e.g., SSIM, MS-SSIM, PSNR), while LPIPS attempts to measure perceptual similarity with respect to human visual judgment using deep learning. Together, they enable a balanced evaluation of visual fidelity and technical precision in compressed images.

Tkinter is Python's built-in GUI toolkit, wrapping the mature Tcl/Tk framework to provide a rich set of widgets—buttons, menus, dialogs, canvases—without extra installs. Its simple, object-oriented API and cross-platform consistency let you build desktop apps quickly, focusing on functionality rather than low-level GUI details. By abstracting complex algorithms into user-friendly interfaces, it bridges algorithmic complexity with user accessibility.

The methods described herewith are now set to form a technical framework for adaptive JPEG compression, including preprocessing, color space transforms, and perceptual quality measures. Having set such a technical basis, the next chapter formalizes these principles into solid specifications.

4. ANALYSIS & DESIGN

This chapter outlines the in-depth analysis and design process of the project. This entails the establishment of the functional and non-functional requirements and the design elements including the use case diagrams and the sequence diagrams.

4.1. Analysis

The analysis section entails a comprehensive consideration of the functional and non-functional requirements of the project. This describes what the system should do and how it should perform.

4.1.1. Functional Requirements

Functional requirements specify the system's core behavior and what the system is intended to achieve.

The system shall accept input and produce output in any well-known image type as well as the custom AJPG file format. This requirement will enable the system to handle the great majority of standard formats available and enable users to easily incorporate the tool in existing workflows. By accommodating common formats and the suggested AJPG format, the system offers flexibility and wide interoperability and minimizes the requirement to preprocess or translate images prior to the use of the utility.

The system shall provide an option to perform batch processing of images. Batch processing makes the system much more efficient when dealing with high numbers of images. One operation can be used to configure and execute a number of encoding or decoding tasks by users, which will reduce manual intervention. This can prove very beneficial in production environments where resource and time optimization are the key issues and overall productivity can be boosted.

The system shall allow users to select the color space, quality range, and block size range for compression. Providing the capability to tailor color space, quality, and size enables users to fine-tune the operation based upon their particular needs. Such flexibility accommodates a broad range of usage cases, from high fidelity applications to those requiring low file size.

It enables users to try out varying setups and determine the ideal compromise between image quality and the efficiency of the resulting file size.

The system shall offer a preview of the processed image and display relevant evaluation metrics. In providing both live preview and quantitative evaluation metrics, the system will allow users to instantly determine the effects of their selected parameters on the quality of the resulting image. This feedback loop in real time is essential when users need to make iterative adjustments, and they can do so effectively and promptly. Providing different metrics also addresses different users' needs in terms of evaluation so that users will get a more comprehensive assessment of image fidelity.

The system shall encode standard images into the AJPG format and decode AJPG files back into standard image formats. This requirement underlines the bidirectional conversion capability of the system in terms of conversion. Encoding the images to the AJPG format aids in taking advantage of specialized means of compression, while decoding the files ensures users can switch back to widely accepted image formats. This two-way capability is important in terms of real-world usage.

The system shall store chosen settings (color space, quality range, block size range) as metadata within the AJPG files. Embedding the compression settings directly into the AJPG file as metadata provides an internal record of how the image was processed. This feature supports reproducibility, allowing the same settings to be applied during decoding.

4.1.2. Non-Functional Requirements

Non-functional requirements identify how a system should work. They define the system's quality attributes such as performance, usability, availability, and scalability.

The system shall provide a graphical user interface (GUI) that is easy to understand and operate—users must be able to complete the core workflows within one minute. An easy-to-use GUI is vital in order to ensure novice and expert users can easily use and navigate the system. The core workflows such as loading images; adjusting color space, quality, and block-size parameters; and executing compression or decompression operation up to 10 images must be completed within the one-minute threshold. A GUI with proper design reduces the learning process and the likelihood of user errors and ensures a good user experience. Also, it aids in increased user adoption and satisfaction as it makes functionalities accessible.

The system shall be capable of achieving comparable (within ±5% tolerance range) or superior image quality relative to the standard JPEG algorithm at similar compression ratios (within ±5% tolerance range). This requirement is highlighting the necessity of the system to either match or surpass the quality of images provided by the conventional JPEG standard. By providing competitive compressed image quality, the system can be adopted with certainty by users depending on the preservation of high image fidelity. It acts as a performance standard, upholding the role of the system as a complement to existing compression methods.

The system shall be capable of achieving comparable (within ±5% tolerance range) or superior compression ratios relative to the standard JPEG algorithm at similar quality scores (within ±5% tolerance range). Compression ratio is a principal performance indicator of image processing systems. The requirement ensures the system is concerned with image quality as well as minimizing the size of files. By providing equivalent or better compression ratios compared to baseline JPEG, the system can assist users in saving storage capacity and transmission bandwidth without trading off image detail.

The system shall run on Windows, Linux, and macOS, or any operating system that has Python and the required libraries installed. Cross-platform compatibility guarantees users across different operating systems access and use the system without worrying about the implications of compatibility problems. Such wide support is imperative if the system is to serve the varied needs of the potential users, starting from personal desktop use to enterprise-wide deployment. It also reflects a commitment to versatility, making the system adaptable to various technological ecosystems.

4.2. Design

This section introduces the design of the adaptive image compression system. The goal of the system is to provide flexible, content-aware compression while maintaining a user-friendly interface that allows fine-tuned control over image quality and compression behavior.

Figure 4.1 illustrates the Use Case diagram of the system, indicating main user functions and interactions. The users can adjust critical settings such as color space, quality range, and block size range, while previewing the result in real-time on a sample image. This preview feature is enhanced by option to refresh the view upon changes and to analyze evaluation metrics for an unbiased measure of compression quality. The figure also shows the main compression and decompression workflow, which necessarily includes the File System for the input image selection and, later, for the storage of the processed outputs, thus defining

a complete workflow. It addresses the whole process of the user from the very first step of parameter selection and interactive previewing through to the ultimate output generation and storage. In all, the diagram provides a clear, high-level picture of the system design, highlighting its flexible, user-centered approach to image compression that allows users to make knowledgeable decisions about the whole processing chain.

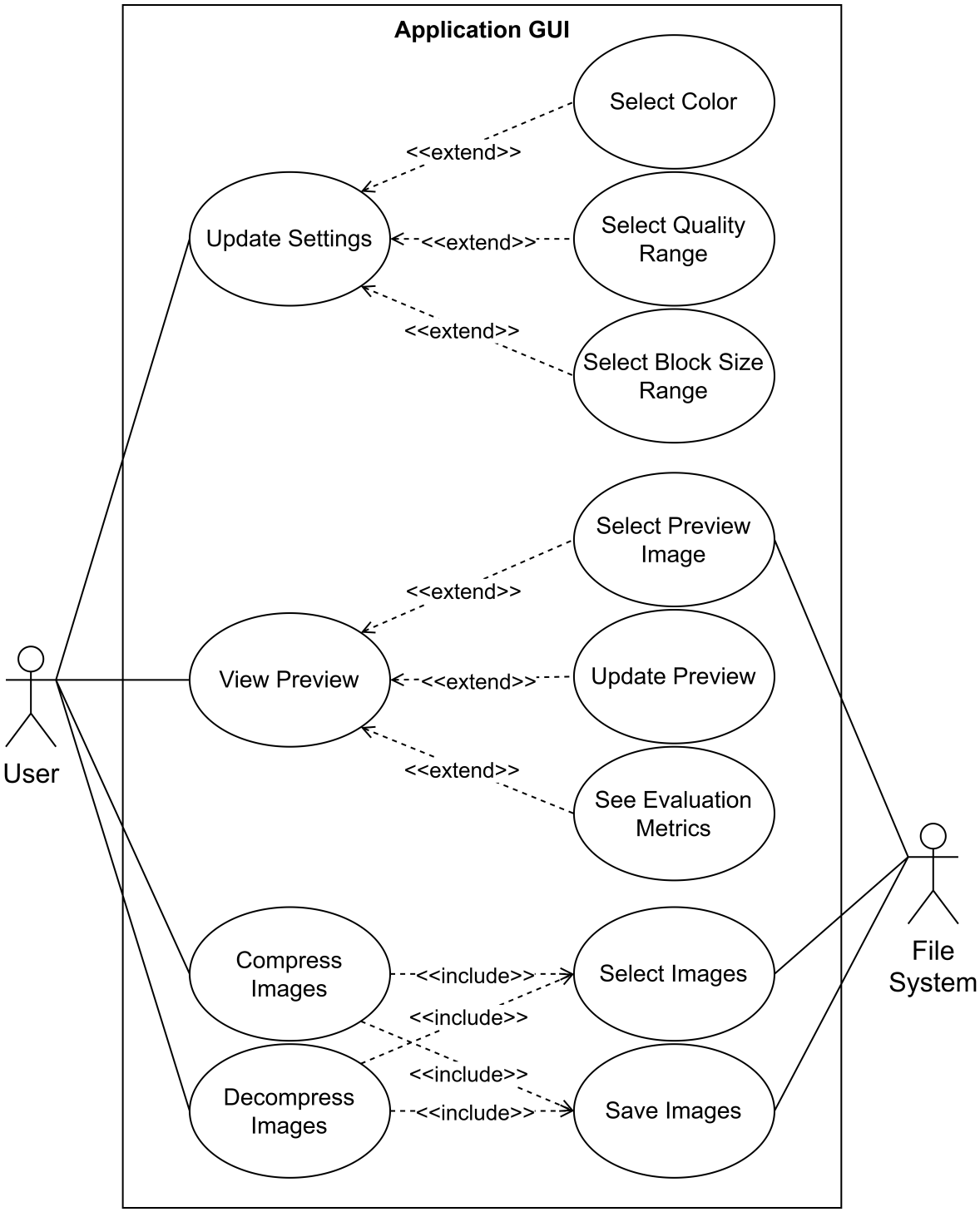

Figure 4.1. Application GUI Use Case Diagram

Figure 4.2 is the sequence diagram of the graphical user interface (GUI) interactions of the adaptive image compression system. This figure illustrates how the system responds to user actions—such as adjusting settings, previewing compression results, and performing image compression or decompression—through communication between modules like JpegApp, ControlPanel, PreviewPanel, and Jpeg. The sequence begins with the initialization of key GUI components, after which the user interacts with the interface to select compression parameters. These involve color selection, quality and block size slider adjustment. These settings are of central significance, as they directly determine the tradeoff between the pictorial quality of the compressed image and the level of file size reduction achieved, enabling users to tune the output to their own particular needs. These settings having been defined, the system submits the settings through the ControlPanel and updates the application configuration appropriately.

In the preview stage, the user selects a sample image to preview the result of the compression parameters specified. The system acts on this request by initially displaying the original image and then performing compression and decompression. These actions are delegated to the Jpeg module, which handles the core image processing logic, which can be seen in Figures 4.3 and 4.4. After obtaining the decompressed result, the application calculates the compression ratio and evaluation metrics. These metrics, along with the processed preview image, are returned and displayed to the user within the PreviewPanel, providing immediate feedback on how well the chosen parameters balance quality and compression.

The last section of the diagram illustrates the overall compression and decompression process. In this case, the users choose images to process and start the compression or decompression process. This can be a single image or a batch of several files, offering flexibility for different user workflows. The JpegApp coordinates with the Jpeg module to carry out compression or decompression on all selected images. The individual images are processed and the results are saved.

Together, these figures present a detailed and structured view of the system's operation, demonstrating how the interface and backend components work in harmony to deliver adaptive, user-driven image compression.

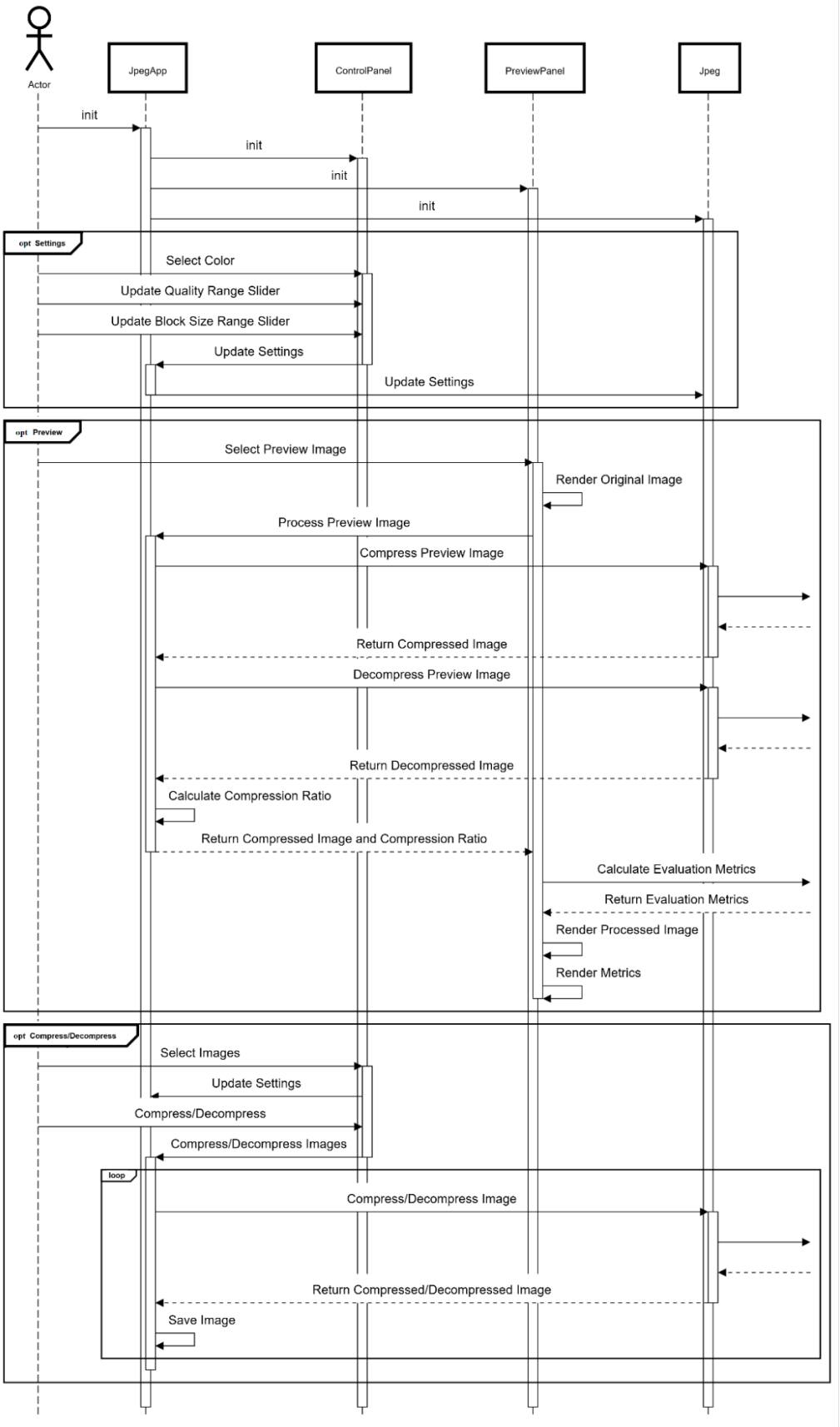

Figure 4.2. User-GUI Sequence Diagram

Figure 4.3 is a sequence diagram illustrating the adaptive image compression process in the system. The system begins with the call to the compression function, thereby initiating a chain of preprocessing and transform operations that set up the image for effective encoding. The initial operation is the transformation of the input image into a suitable color space, followed by downsampling to achieve data reduction in the chrominance layers—a common step in image compression algorithms. Chroma subsampling exploits the reduced sensitivity of the human vision system to color information as opposed to luminance and therefore enables significant data reduction with little perceived effect on the quality of the image.

Next, the system enters the Block Split Loop, where edge detection and spatial partitioning are applied. For each layer, the system employs a Canny edge detector, which is enhanced through several preprocessing filters: CLAHE (Contrast Limited Adaptive Histogram Equalization, see Section 3.4), Gaussian blur, and bilateral filtering (see Section 3.3). These methods sharpen edges and reduce noise, making the edge map more credible for the further partitioning procedures. Then the extracted edge data is forwarded to the quadtree (see Section 3.6) module, which recursively divides the image into irregular blocks according to edge complexity, forming a hierarchical tree structure. The leaf nodes of the tree correspond to the final blocks for compression.

Following partitioning of the image, the specified blocks are extracted by the system and a Discrete Cosine Transform (DCT) is carried out on each of them. The operation then enters the Adaptive Quantization Loop, where quantization parameters are adaptively selected based on the size of every block. For each block, a quality factor and corresponding quantization matrix are calculated, allowing fine-grained control over the fidelity of compression. Each DCT-transformed block is quantized accordingly, significantly reducing data volume while preserving perceptual quality.

The final stage is Entropy Encoding, where the quantized data is saved by serializing it. General image metadata, which includes color space, quality and block size ranges, is written, followed by a loop over all image layers. Each layer undergoes zigzag ordering—a technique that organizes DCT coefficients for more effective entropy coding—before lossless compression is applied to further reduce file size. Metadata corresponding to each and every layer, along with the compressed data, is stored, creating a compact, organized representation of the original image. The procedure ends with the compressed image ready for storage or sending, which now occupies a significantly reduced size with minimal compromise in quality.

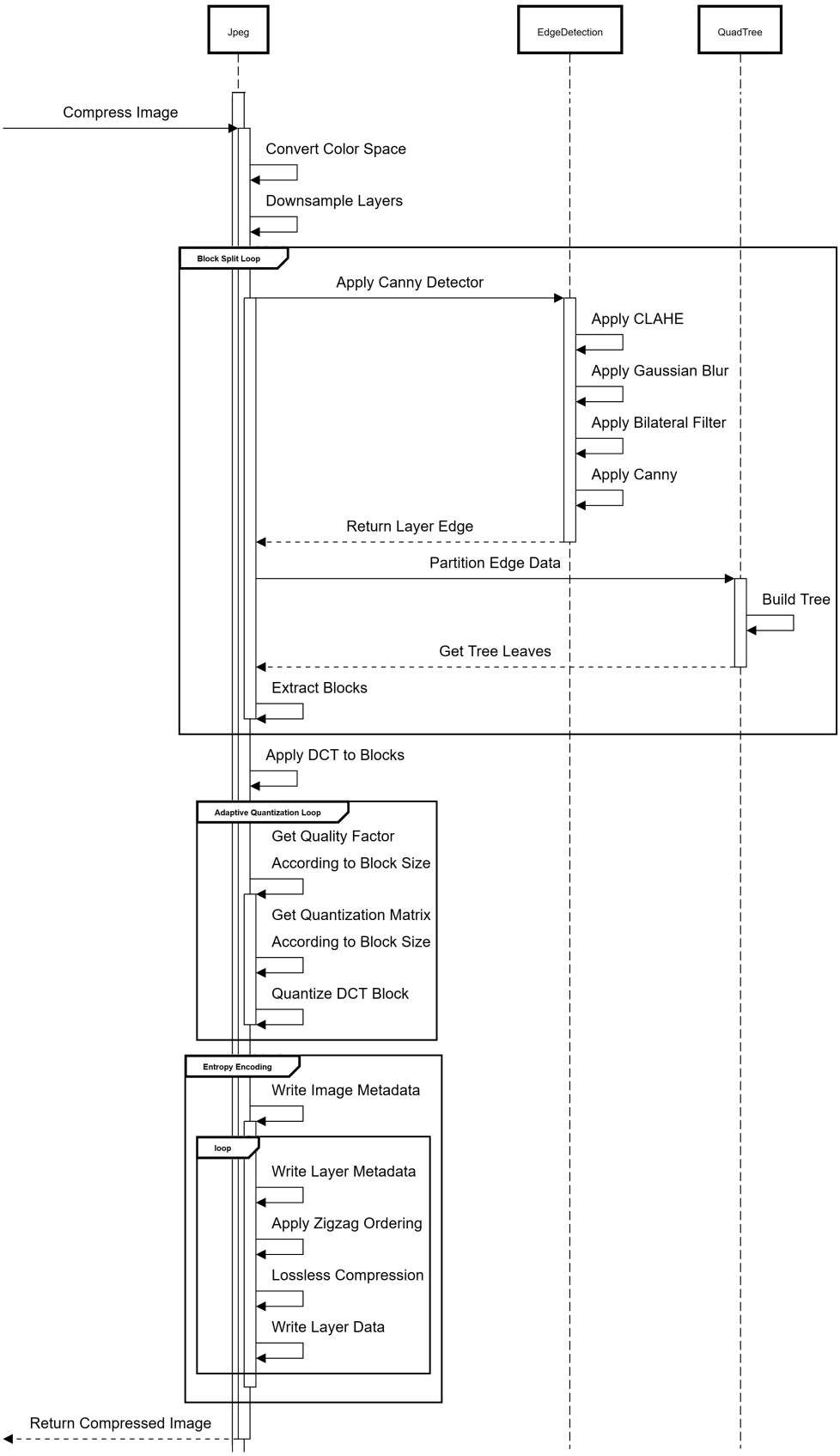

Figure 4.3. Image Compression Sequence Diagram

Figure 4.4 illustrates the decompression process, the reverse of the transformations in the compression pipeline (shown in Figure 4.3). The operation begins with the Entropy Decoding phase, where the image metadata in the encoded file is read to initialize the decompression environment. The metadata has parameters required for decoding and reconstructing layer sizes so that the system knows how the compressed file is organized. Accurate interpretation of this metadata is crucial as it dictates how the subsequent streams of compressed data for each block and layer are to be handled and reconstructed.

The system initiates a decoding loop to process each layer of the image separately. It extracts metadata for each layer and uses it to decode the size of quadtree leaf blocks that have been created during compression. The adaptive block partitioning information is necessary to properly map the decoded data to its spatial location in the image. The compressed image data is then read, and lossless decompression is applied. The inverse of the zigzag reordering operation is subsequently performed to restore the frequency-domain representation of the image blocks to their original arrangement.

Following entropy decoding, the Adaptive Dequantization Loop begins. It processes each DCT block by calculating its quality factor and corresponding quantization matrix, based on its size. The blocks are then transformed back to the spatial domain through the Inverse Discrete Cosine Transform (IDCT).

After the reconstruction of individual image blocks, the system assembles them as follows: With the leaf sizes deciphered previously, the blocks are rearranged to their original locations within each image layer. The merging process rebuilds each layer's continuous pixel grid from the blocks that were processed individually. The resulting layers are upscaled to their native resolution. Then, the image is transformed from its internal color representation to the standard RGB format, completing the decompression process and producing the fully restored image.

The adaptive image compression system design is expressed in a series of complementary diagrams that collectively describe user interaction and internal algorithms. The use case and GUI sequence diagrams emphasize user actions and interface responses, whereas the technical sequences describe compression and decompression processes step-by-step. They collectively provide a unified and solid view of the system's functionality.

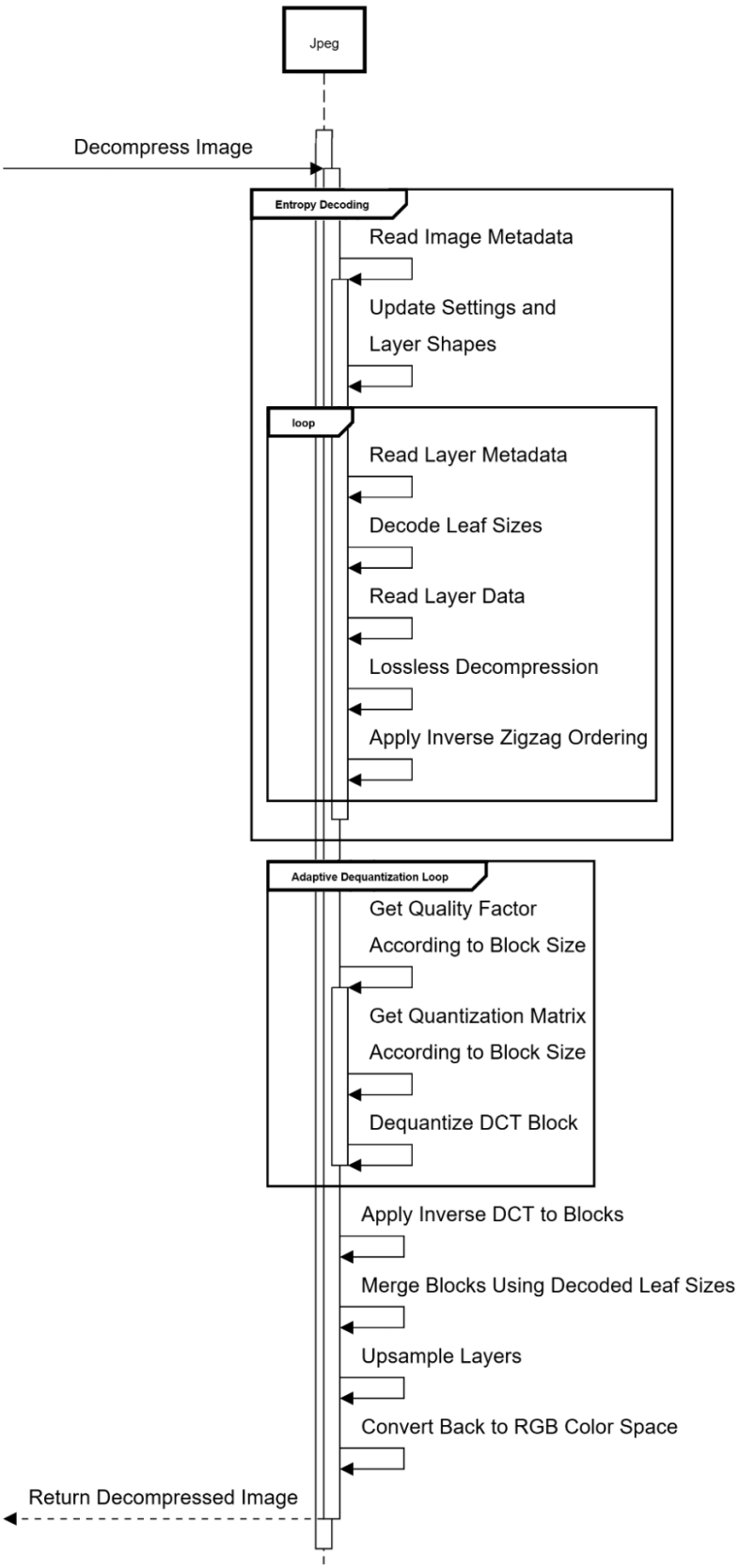

Figure 4.4. Image Decompression Sequence Diagram

5. IMPLEMENTATION

This chapter outlines the implementation of the adaptive, edge-aware image compression system proposed previously. It begins with the high-level architecture, then details individual components and their interactions. Subsequent sections will dive into the specifics of color-space conversion and compression/decompression algorithms along with GUI implementation making these features available to end users.

5.1. Overall Architecture

The implementation illustration in Figure 5.1 shows the transformation of an image into a compressed file and its subsequent reconstruction. Input image first goes through color conversion prior to compression, depending upon parameters that are tunable and predetermined, results in the final AJPG file. The reverse process includes decompression of the AJPG file and then color conversion to produce an output image.

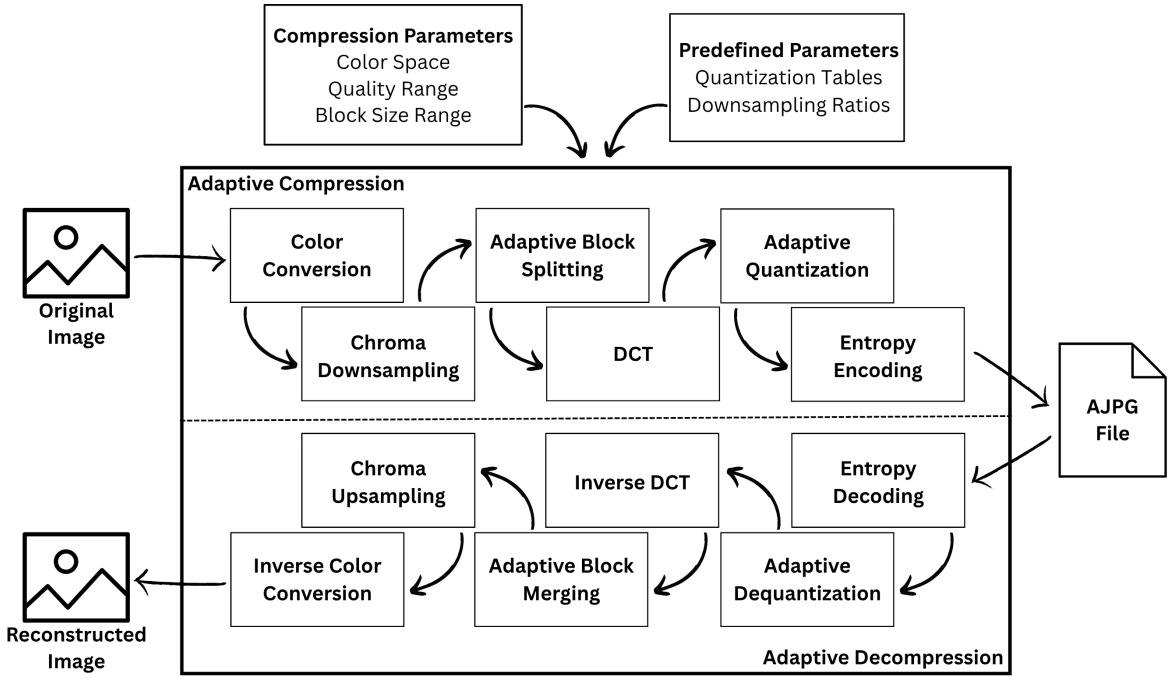

Figure 5.1. Overall System Architecture

The behavior of the system is essentially controlled by a given set of compression parameters supplied at configuration time. These parameters specify essential aspects of the process such as internal processing color space, desired range for quality levels, and range

for block sizes. All these parameters which are configurable allow for direct user control of tradeoffs in compression. In contrast, the quantization tables for luminance and chrominance and the downsampling factors for the color components are fixed by the codec and cannot be altered by the user.

The compression pipeline converts an incoming image into the custom AJPG format in a series of operations. The image first goes through preprocessing in which conversion from a standard input color space (sRGB) into an internal processing space (e.g., YCbCr, YCoCg, Oklab) and chroma downsampling is done. Then, it is followed by edge detection for determining image content, which directs the hierarchical partitioning algorithm for dividing the image into variable sized blocks based on local complexity and block size range setting. Each block is then converted from pixel information into frequency coefficients using Discrete Cosine Transform (DCT). Subsequently, adaptive quantization, where the quantization strength for each block's coefficients is determined adaptively depending upon block size and overall quality parameters – smaller blocks from detailed regions get finer quantizing, while bigger blocks from smooth regions get coarser quantizing. Finally, the quantized coefficients are prepared for storage via entropy coding, which includes coefficient reordering (zigzag scanning), packaging required metadata (image dimensions, settings, partitioning structure), and applying lossless compression to the coefficient data stream, resulting in final AJPG byte stream.

The decompression pipeline reverses the compression process in order to reconstruct the image from the AJPG file. It first reads the file's metadata and extracts image dimensions, original compression settings, and encoded partitioning structure. These parameters are then used to configure the decompression engine. Next, coefficient data is decompressed, and the partitioning structure is reconstructed. With this structure, the coefficient stream is properly divided, and coefficients for every block are reconstructed using an inverse zigzag scan. Following this, adaptive dequantization occurs, in which coefficients are scaled using quantization parameters based on block size and quality range stored. Inverse DCT transforms coefficients into spatial representation for every block. The blocks are then combined based on the decoded partitioning structure in order to reconstruct complete image layers. Finally, post-processing operations, such as chrominance upsampling and conversion into standard display color space (sRGB), result in fully reconstructed output image.

This is the high-level framework for the adaptive compression approach. The specific algorithms and implementation choices for each phase – color conversion, the step-by-step procedures for compression and decompression, and user interface – are described in detail in the following sections.

5.2. Color Conversion

An important initial step in the compression pipeline is to translate the input image data from its standard source color space, sRGB, into an internal color representation more suitable for efficient compression. Standard sRGB, while very prevalent for display, luminance and chrominance data are not explicitly separated and has a non-linear correspondence with perceived brightness due to its gamma correction. Many compression techniques, including the DCT-based approach used here, benefit significantly from operating in color spaces that decorrelate color components, separate luminance (brightness) from chrominance (color), or are perceptually more uniform. This separation allows quantization schemes to be optimized separately for brightness and color information, utilizing the human visual system's lower sensitivity to chromatic detail over luminance.

The implementation is also flexible in that it supports color space transformation for a set of established color spaces (see Section 3.2). The fact that there are several color spaces allows the system to support diverse source content characteristics and compression goals, as each space offers varying advantages for perceptual uniformity, component decorrelation, or computational simplicity. These are:

- **YCbCr:** A widely used family in image and video compression (like standard JPEG) for its effective separation of luma (Y) and chroma (Cb, Cr).
- **YCoCg / YCoCg-R:** Simple and computationally efficient linear transformations offering good decorrelation, with the '-R' variant providing reversibility even with integer arithmetic.
- **OKLAB:** A modern color space designed for perceptual uniformity, meaning numerical differences correspond more closely to perceived color differences.
- **ICtCp and ICaCb:** Spaces designed primarily for High Dynamic Range (HDR) and Wide Color Gamut (WCG) content.
- **JzAzBz:** Another perceptually uniform space, particularly optimized for HDR and WCG applications, aiming for uniformity in color difference perception.

The conversion is handled by a dedicated component that routes data between sRGB and the selected target internal space. As transformations between many advanced color spaces are usually based on an intermediate representation (commonly CIE XYZ), the implementation follows established multi-step routes where required.

These transformations consist of both linear operations (primarily matrix multiplications defined for each color space pair) and nonlinear functions. The nonlinearities are the standard sRGB gamma correction and, in HDR-focused spaces, the PQ EOTF (Electro-Optical Transfer Function) and its inverse. These are more computationally demanding than simple linear transforms.

Following the color space conversion, the data is normalized before it is fed into the main stages of compression. The normalization is performed by shifting the data components based on predefined midpoints and scale factors specific for each color space. The aim is to map the component values into a fixed numerical range, [-127, 127]. This normalized range offers consistency and is required during the subsequent DCT and quantization stages. During decompression, a corresponding denormalization step reverses this, restoring the components to their original scale within the color space.

Given the computational cost these conversions would likely involve, especially for large image arrays, the implementation exploits fast numerical libraries (NumPy) for the array operations. Besides, the computationally heavy processes, particularly the nonlinear transfer operations of the matrix multiplications, are accelerated through Numba's Just-In-Time (JIT) compilation into highly optimized machine code, often with automatic parallelization over any number of available CPU cores, thus reducing the time consumed in processing the color conversion. Such a JIT compilation approach gives a good balance, maintaining the development speed of Python while achieving close to native performance for performance-critical, computation-intensive sections of the color conversion pipeline.

5.3. Compression

Following color transformation detailed in Section 5.2, the core compression pipeline processes the image data into final AJPG file. The procedure consists of several stages aimed at reducing data redundancy while adaptively preserving visual information. The stages encompass chroma downsampling, edge-guided adaptive block partitioning, frequency transformation using Discrete Cosine Transform (DCT), adaptive block quantization, and entropy encoding to effectively package the data.

5.3.1. Downsampling

Before the initial spatial processing, for every chosen processing color space, chrominance layers are downsampled according to predefined ratios. This common practice, reduc-

ing the color resolution in comparison to luminance, exploits human eye's lower sensitivity to chromatic detail. Downsampling is achieved using pixel area relation resampling, well-suited to reduce layer dimensions while mitigating aliasing artifacts. The target dimensions for each layer are based on the original image dimensions and on the predefined downsampling factors specific to the color space chosen.

5.3.2. Adaptive Block Partitioning

A key innovation of the system is avoidance of the fixed-block method of standard JPEG. In this case, the image is spatially partitioned into variable-sized square blocks based on local content complexity, guided by available edge information. The adaptive partitioning is performed with the edge detection results to create a hierarchical quadtree structure across the image plane.

The process begins with the generation of an edge map. A Canny-based edge detection algorithm is used, together with a suite of preprocessing operations, as seen in Figure 5.2, in order to improve robustness and accuracy. Contrast Limited Adaptive Histogram Equalization (CLAHE) is used to enhance local contrast, followed by Gaussian and bilateral filtering operations to remove noise without removing important edges. Rather than employing constant values, the Canny algorithm here uses dynamic hysteresis thresholds derived from the intensity distribution of the preprocessed image—the 10th percentile for the lower and the 30th for the upper threshold—which makes the edge detection more adaptive. With adaptive thresholds, according to the intensity distribution of the image, the edge detection is more robust against global image brightness or contrast variations compared to using constant thresholds. The result is a binary edge map that indicates edges or significant textures.

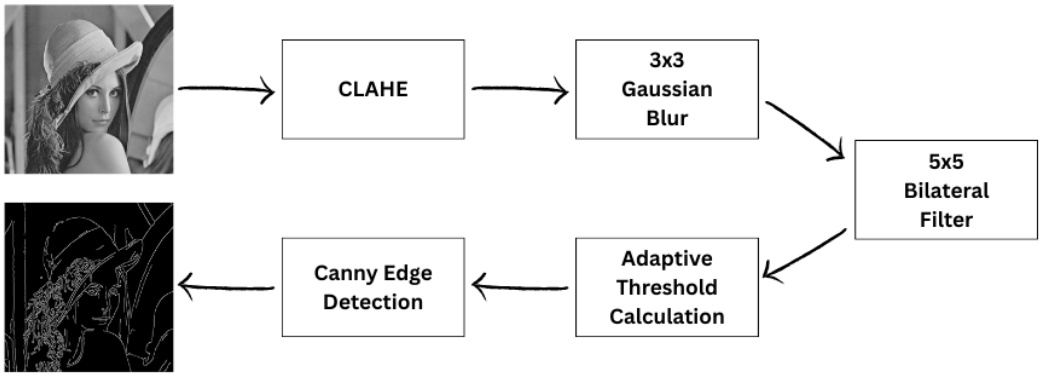

Figure 5.2. A diagram illustrating the edge detection pipeline

The edge map serves as a guide to the quadtree construction. The image area is divided recursively into four quadrants of equal size. The decision to divide any given region is dependent on whether it meets some conditions: if the size of the region exceeds the chosen maximum block size limit, or its size is above the chosen minimum limit and it has detected edge pixels according to the edge map, then it is split. This recursive splitting continues until a region is either of the smallest possible block size or within the size limits and has no edges. These terminal regions are the leaf nodes of the quadtree. An example with the block size range of 8-128 can be seen in Figure 5.3.

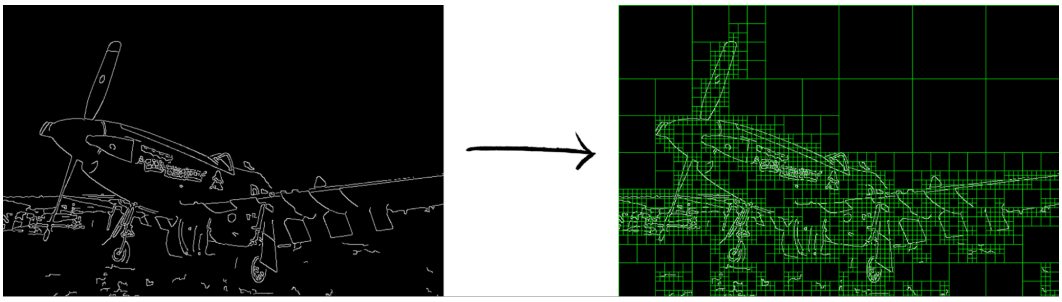

Figure 5.3. A visual example of edge map and the resulting quadtree partitioning

Consequently, large leaf nodes correspond to relatively uniform image areas, while regions rich in detail or edges are covered by a hierarchy of smaller leaf nodes. The process accommodates arbitrary image dimensions by initially basing the quadtree on a root size encompassing the entire image, ensuring all pixels are included.

5.3.3. Block Processing

With the adaptive partitioning established, each leaf node of the quadtree defines a block for further processing, primarily consisting of frequency transformation and quantization.

Initially, the two-dimensional Discrete Cosine Transform (DCT) is applied to every extracted block. This transformation converts the spatial pixel data into a frequency domain representation (see Figure 3.3), effectively compacting the block's energy in a small number of low-frequency coefficients in the top-left corner of the resulting coefficient block.

Following the DCT, the crucial lossy operation of quantization is performed adaptively. For each block of DCT coefficients, quantization involves several operations. First, a specific quality factor is determined for the block. This quality factor is calculated between user-defined minimum and maximum quality settings based on the size of the present block in

relative to the allowed range of block sizes. A logarithmic scaling is employed to guarantee that quality adjustments are linear between choices of block sizes (4, 8, 16, …). This block-size dependent quality factor, is computed as shown in Equation 5.1. Hence, small blocks, typically representing detailed areas, are provided with a higher quality factor (resulting in finer quantization), while large blocks from smooth areas are provided with a lower quality factor (resulting in coarser quantization). This follows the understanding that blocks of smaller sizes tend to be used on detailed and complex regions where fidelity must be most accurately preserved, but blocks of larger sizes in areas of smoother surfaces can tolerate more aggressive quantization without significant perceptual loss.

$$Q = \begin{cases} \lfloor \frac{q_{min}+q_{max}}{2} \rfloor & \text{if } b_{min} = b_{max} \\ \lfloor q_{min} + (q_{max} - q_{min}) \left(1 - \frac{\log(b/b_{min})}{\log(b_{max}/b_{min})}\right) \rfloor & \text{if } b_{min} \neq b_{max} \end{cases} \tag{5.1}$$

Second, an appropriate base quantization matrix (luminance or chrominance) is selected. This base matrix is then scaled, based on the block-dependent quality factor obtained previously, using a standard scaling equation, determined as detailed in Equation 5.2. The actual scaled matrix, is then calculated as shown in Equation 5.3. Subsequently, the scaled matrix is resized, by bilinear interpolation, to match the size of the current variable-sized DCT block.

$$S = \begin{cases} \frac{5000}{q} & \text{if } q < 50 \\ 200 - 2q & \text{if } q \geq 50 \end{cases} \tag{5.2}$$

$$M_{scaled} = \left\lfloor \frac{S \cdot M_{default} + 50}{100} \right\rfloor \tag{5.3}$$

Finally, each DCT coefficient within the block is divided by the corresponding element of the size-dependent, quality-scaled quantization matrix, and rounded to the nearest integer. Division and rounding discard information, particularly for the high-frequency coefficients that are typically less perceptually significant. This adaptive quantization, coupled with the adaptive block sizing, is fundamental for achieving the optimal compromise between detail preservation and compression efficiency. For optimal performance, the dynamically calculated quantization matrices for varying block sizes and quality levels are precomputed and cached when settings are initialized or updated.

5.3.4. Entropy Encoding

The final stage organizes the quantized coefficients and necessary metadata into the AJPG byte stream. The 2D quantized integer coefficient block is initially transformed into a 1D sequence using a zigzag scan pattern (refer to Figure 3.4). The pattern scans the block diagonally, usually storing coefficients in order of lowest to highest frequency. The arrangement has a tendency to cluster the many zero coefficients that are generated due to quantization, and this renders the subsequent lossless compression more efficient.

Before the encoding of the coefficient data, metadata required for decompression is packaged. Global metadata, including the original image dimensions, the number of layers, the color space identifier, the range of quality and block sizes used during compression, and original file extension, is serialized into JSON and written to the output stream, preceded by its length.

In addition, metadata for every layer that indicates the quadtree structure must be encoded in a form where the decompressor can reconstruct the variable-sized blocks in the correct manner. This is successfully achieved by recording the sequence of decisions made during the traversal of the quadtree generated during the partitioning phase. As the traversal visits each potential node location (starting with the root and progressing recursively from top-left, top-right, bottom-left, bottom-right quadrants), a compact state code is generated:

- **’00’ (Leaf):** Indicates a leaf node. The decompressor knows this region corresponds to a single block, and no further subdivision occurs for this branch.
- **’01’ (Split):** Indicates an internal node. The decompressor knows it must recursively process the four child quadrants of this node.
- **’10’ (Null/Empty):** Indicates that this potential quadrant does not correspond to a valid node within the image context (e.g., it falls entirely outside the actual image boundaries defined by the layer shape). The decompressor skips this branch.

An example of the encoding can be seen in Figure 5.4. The packed bitstream result of the example will be: *01 01 00 01 00 00 00 00 00 00 00 10 10*.

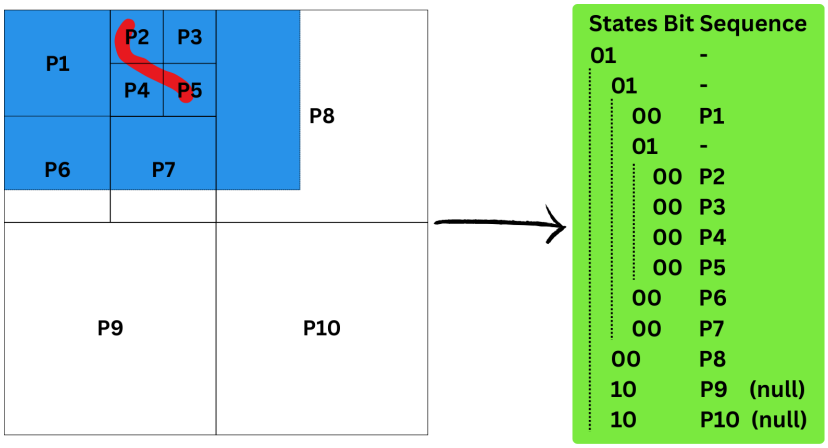

Figure 5.4. A diagram illustrating the quadtree encoding *(On the left, a simple partitioned quadtree with the blue region representing the original image. On the right, the resulting states bit sequence top-to-bottom matched with the quadtree regions)*

The sequence of 2-bit state codes is a compact, linear representation of the entire tree structure. This structural information is also written to the stream, preceded by its length and root node size of the layer.

Finally, the 1D sequences of zigzag-scanned coefficients of all blocks within a layer are concatenated. The combined sequence is then compressed using a standard lossless data compression algorithm, LZ77 [54] compression followed by Huffman coding [6]. The resulting compressed data block for the layer is written to the output stream, prefixed again by its length.

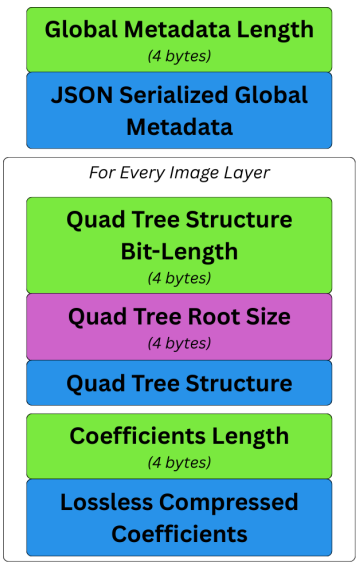

Figure 5.5. A diagram illustrating the AJPG file format structure

The resulting AJPG file structure, which is visualized in Figure 5.5, thus consists of a global metadata header, followed by data blocks for each image layer, where each layer block contains its quadtree structural information and the losslessly compressed stream of its quantized DCT coefficients.

5.4. Decompression

The decompression process reconstructs the image by methodically reversing the procedures performed under compression. The system takes the compressed AJPG file as input and progressively reconstructs the image data starting with entropy decoding to recover coefficients and structure information, followed by dequantization, inverse DCT, block merging, chroma upsampling, and finally, color conversion back to sRGB.

5.4.1. Entropy Decoding

This initial stage reads the AJPG file stream to get metadata and recover the quantized DCT coefficients of each block. The global metadata is read first. The length is deduced from the initial four bytes, allowing the subsequent serialized metadata (JSON containing original dimensions, color space identifier, quality and block size ranges, original file extension) to be read and parsed. This information is crucial for configuring the decompressor parameters to the same parameters used in the original compression to maintain consistency, particularly for adaptive dequantization and determining the final output dimensions.

The information for each image layer is then processed sequentially. For every layer, the structural header representing the quadtree partitioning is read. This involves reading the stored bit length and root size, followed by the byte array containing the sequence of 2-bit state codes. This sequence of states is then decoded by conceptually traversing the quadtree grid, starting from the root size, in the exact same order as during encoding. By reading the state codes sequentially during traversal, the decoder determines the size of each leaf node encountered and implicitly knows their spatial order. This process reconstructs the list of leaf sizes which dictates the dimensions of the blocks whose coefficient data follows. This reverses the concept illustrated previously in Figure 5.4.

Following the structural header, the compressed coefficient data for the layer is read. Its length is obtained from the stream, the corresponding data block is read, and the lossless decompression algorithm (complementary to the one used in compression) is applied. This

yields the original sequence of concatenated, zigzag-scanned, quantized integer DCT coefficients for all blocks in that layer.

Using the reconstructed list of leaf sizes, the 1D coefficient stream is then partitioned. Each partition corresponds to one block, with its length determined by the square of the respective leaf size. Each partition undergoes the inverse zigzag scan pattern such that the 1D sequence is transformed back to a 2D array of quantized DCT coefficients of the block.

5.4.2. Image Layer Reconstruction

Once the 2D blocks of quantized coefficients are recovered for a layer, they are processed to reconstruct the spatial domain pixel values. The first step is adaptive dequantization, which reverses the lossy quantization applied during compression. For each recovered block, its size along with the quality range settings is used to calculate the exact same block-specific quality factor that was used during compression. Each integer coefficient in the quantized block is then multiplied element-wise by the corresponding value.

After dequantization, the Inverse Discrete Cosine Transform (IDCT) is applied to each block. This transforms the frequency domain coefficients back into spatial domain values. Finally, the individual spatial blocks must be reassembled into a complete image layer, reversing the partitioning process, using the reconstructed quadtree structure. A blank canvas corresponding to the layer's dimensions is created. The blocks are placed onto this canvas in their correct spatial locations, effectively tiling the layer according to the partitioning determined during compression.

5.4.3. Upsampling and Color Conversion

The final post processing operation prepares the reconstructed layers for output. Initially, pixel values are denormalized using color space parameter derived from the file's metadata, reversing the normalization applied for compression and restoring the correct value ranges within the processing color space. Subsequently, downsampled chrominance layers are upsampled, via bilinear interpolation, to the original image resolution. The fully constructed layers are subsequently converted from processing color space back to the sRGB space by the appropriate inverse transformation. This yields the final pixel data for the output image, completing a sequence that mirrors the compression pipeline in reverse.

5.5. Graphical User Interface (GUI)

To facilitate user interaction with the adaptive compression system and meet the usability nonfunctional requirement, a Graphical User Interface was designed using Python's native Tkinter library. The GUI allows user-friendly controls for adjusting compression parameters, managing files for batch processing, and observing the effect of chosen settings in real-time, thereby abstracting the system's more complex algorithmic details into a user-friendly workflow.

The main application window, shown in Figure 5.6, is a clean two-panel layout. The left panel holds the Control Panel, which is dedicated for parameter configuration and batch operations, while the right panel features the Preview Panel, designed for real-time visual feedback and quantitative assessment.

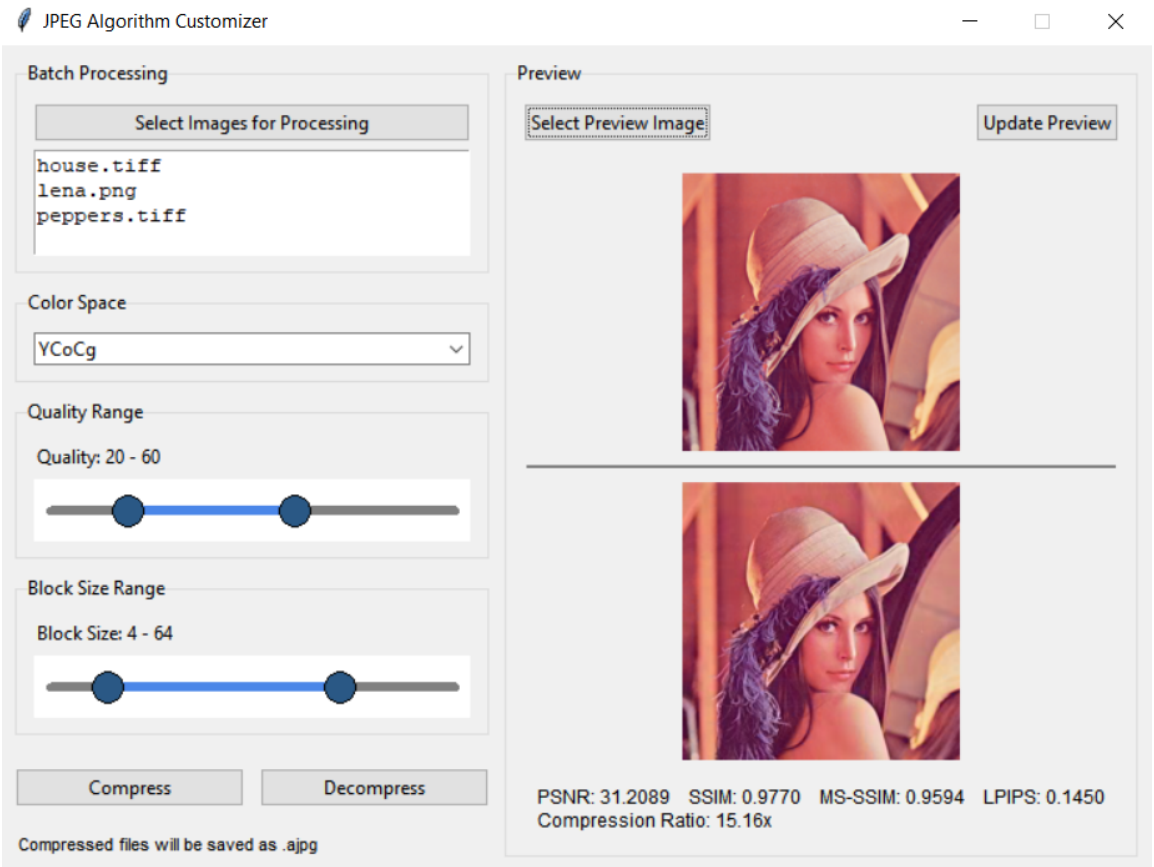

Figure 5.6. The Graphical User Interface (GUI)

The Control Panel aggregates all user-adjustable settings. At the top, a "Batch Processing" section that allows users to select multiple image files or previously compressed

AJPG files from a simple file dialog. The names of selected files are displayed in the text field below the selection button. Below batch processing, compression parameters can be individually adjusted. A drop-down list allows the user to select the desired internal color space from the implemented options (e.g., YCoCg, OKLAB, YCbCr). The key adaptive parameters are controlled by custom slider widgets. The Quality Range slider allows the user to specify the minimum and maximum quality values the adaptive quantization algorithm interpolates between. The same applies to the Block Size Range slider, which can be used to specify the lower and upper block sizes (represented as powers of 2, for example, 4x4 to 64x64) that will be used by the adaptive partitioning algorithm. Dedicated ”Compress” and ”Decompress” buttons trigger batch processing of the chosen files. Compression uses the currently configured GUI parameters, whereas decompression automatically sets itself up based on the parameters embedded in the metadata of each chosen AJPG file.

The Preview Panel provides valuable real-time feedback. It has ”Select Preview Image” button, allowing the user to load any image for testing, and an ”Update Preview” button for manually refreshing the processed image if needed. The middle section of this panel is a canvas divided vertically. The top portion displays the currently selected original preview image, while the bottom portion displays the effect of compressing and then decompressing that same image using the currently active configuration in the Control Panel. The top-and-bottom comparison allows immediate visual assessment of compression artifacts or detail preservation. Below the processed image, essential quantitative metrics are displayed. These include standard metrics—Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Metric (SSIM), Multi-Scale SSIM (MS-SSIM)—and the perceptual metric, Learned Perceptual Image Patch Similarity (LPIPS), and calculated compression ratio. This combination of visual comparison and quantitative metrics empowers users to make informed decisions while balancing compression efficiency and image quality.

Overall, GUI successfully translates the system’s complex adaptive capabilities into an easy-to-use interface, offering convenient access and evaluation of the compression process.

With the Graphical User Interface now fully described, completing the system’s frontend aspects, this chapter has detailed the complete construction of the adaptive image compression system, from its high-level architecture and algorithmic components to the GUI. Having completed the system, the system’s performance will be tested and evaluated in the next chapter in terms of how well it meets the design goals.

6. TESTING & EVALUATION

This chapter describes the empirical validation of the proposed adaptive image compression system. The system performance is measured quantitatively by a series of controlled experiments. The analysis focuses on assessment of compression effectiveness and image quality preservation, investigating the impact of key parameters, as well as making a comparison with baseline standard within different optimization scenarios.

6.1. Evaluation Methodology

This section describes experimental setup, datasets, performance metrics, and comparison strategies utilized to systematically analyze the adaptive image compression system presented. The approach was devised to deliver quantitative performance measures along with qualitative knowledge about the behavior of the system in different scenarios.

6.1.1. Datasets

The evaluation utilized a combination of standard test images and a larger benchmark database to ensure comprehensive testing across different types of image content. Widely recognized standard test images, including Baboon, House, Jelly Beans, Lena, and Peppers, were used. These are commonly utilized in image processing studies due to their rich diversity, covering fine textures, sharp edges, smooth surfaces, and varied color distributions, making them suitable for illustrating specific algorithmic behaviors and facilitating comparisons with existing literature. For more extensive statistical analysis and robustness testing, LIVE Image Quality Assessment Database Release 2 ([50], [55], [56]) was also utilized. This database offers a wider pool of images where performance can be assessed over a broader category of naturalistic image contents.

6.1.2. Performance Metrics

Two key aspects were assessed to evaluate performance in this system—the compression efficiency and quality of reconstructed images. Compression efficiency was measured in terms of Compression Ratio, where the original uncompressed data size of the image divided by the compressed AJPG file size, as shown in Equation 6.1. Higher ratio values signify more effective data reduction.

$$\text{Compression Ratio} = \frac{\text{uncompressed data size}}{\text{compressed data size}} \tag{6.1}$$

Image quality evaluation was based on various objective scores, whose underlying principles and interpretations were discussed previously in Section 3.7, which are Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), Multi-Scale SSIM (MS-SSIM), and the Learned Perceptual Image Patch Similarity (LPIPS) score. These models were selected for measuring distinct aspects of image fidelity, from pixel-level precision (PSNR) through to structural likeness (SSIM, MS-SSIM) and human perceptual similarity (LPIPS).

To enable direct comparisons with the baseline JPEG, quality measures were normalized to be presented as ratios. For a specified image and target baseline quality level, adaptive method PSNR, SSIM, MS-SSIM, and LPIPS scores are divided by the respective scores from the baseline, according to Equation 6.2. For PSNR, SSIM, and MS-SSIM, in which higher scores are better quality, a score above 1.0 indicates better performance from the adaptive approach. In contrast, for LPIPS, where lower scores are better quality, a score below 1.0 indicates better performance. A Composite Score was also computed in terms of a weighted average of said individual quality ratios, defined in Equation 6.3. This score gives a single, even measure of overall quality performance against the baseline, formulated so a score above 1.0 implies a general improvement across quality dimensions.

$$\text{Metric Ratio} = \frac{\text{adaptive method metric value}}{\text{baseline metric value}} \tag{6.2}$$

$$\text{Composite Score} = 0.1\,PSNR_{\text{ratio}} + 0.2\,SSIM_{\text{ratio}} + 0.35\,MS\_SSIM_{\text{ratio}} + 0.35\,\frac{1}{LPIPS_{\text{ratio}}} \tag{6.3}$$

6.1.3. Baseline Benchmark

Functionality of the designed adaptive system was primarily benchmarked against standard JPEG, which chosen for its availability and universality as a point of comparison in lossy image compression. The baseline utilized a typical JPEG configuration: employing the YCbCr color space, fixed 8x8 DCT blocks, a single global quality factor used everywhere throughout the image, and 4:2:0 chroma subsampling. This specific subsampling pattern was used since it is a very common tradeoff in standard JPEG implementations, achieving very effective data reduction in the chrominance, halving resolution in both horizontal and

vertical directions. Comparison to standard JPEG baseline is meant to observe if and whether the adaptive method can achieve superior performance to a practical and common standard configuration.

6.1.4. Comparison Strategy and Scenarios

The adaptive compression method was compared against the baseline JPEG as outlined in the previous section. For every comparison point, the baseline JPEG was configured to a specific target level of quality (10, 25, 50, 75, and 90). The parameters of the adaptive system were then analyzed to identify settings that maximized the performance for four strategic priorities, which were mainly centered around either maximizing Compression Ratio or Composite Quality Score, corresponding to various possible use-cases for trading off compression gain and quality preservation with respect to that particular baseline performance:

- **Compression Hard:** This scenario aimed to maximize the Compression Ratio. The only constraint on the Composite Score was the universal requirement that it (and the ratios of its constituent metrics) met the universal 5% tolerance, allowing for slightly worse quality images than the baseline.
- **Compression Soft:** This scenario also aimed to maximize Compression Ratio, but with the additional requirement that the resulting Composite Score should be a minimum of 1.0. This forces the optimization to find the highest compression possible while guaranteeing non-degrading overall quality.
- **Quality Hard:** This scenario aimed to maximize the Composite Score. The only constraint on the Compression Ratio was the universal requirement that it met the universal 5% tolerance, allowing for slightly larger files than the baseline.
- **Quality Soft:** This scenario also aimed to maximize the Composite Score, but with the additional requirement that the resulting Compression Ratio should be a minimum of 1.0. This forces the optimization to find the highest quality possible without increasing the file size relative to standard JPEG.

These four instances have been selected to thoroughly test the adaptive compression system against the specified non-functional requirements for differing operation priorities. The ”Hard” scenarios particularly test the system to significantly improve one of the measures (Compression Ratio or Composite Score) while maintaining the other within the 5% tolerance requirement. The ”Soft” scenarios impose a stricter non-degradation constraint (where the

secondary measure value ought to be ≥ 1.0), testing the ”superior” performance by not allowing even minor degradation on the secondary metric when optimizing the primary metric. This dual approach of ”Hard” and ”Soft” targets for compression and quality enables a multi-dimensional explanation of the system performance over a spectrum of potential use-cases, from extreme optimization with minimal acceptable compromises to balanced improvement guaranteeing no degradation in the secondary metric.

6.1.5. Parameter Space Exploration

The evaluation involved an exhaustive search across the adaptive system's configurable parameters to identify optimal settings for the scenarios defined above. This grid search approach ensured that all possible combinations of the following parameters and their specified ranges were systematically tested:

- **Color Spaces:** Six distinct processing color spaces were evaluated: YCbCr, YCoCg, OKLAB, ICaCb, ICtCp, and JzAzBz.
- **Block Size Ranges:** For the adaptive block partitioning, the system was tested with various minimum and maximum block size limits. The specific block dimensions considered for setting the limits were 4, 8, 16, 32, 64, and 128 pixels. All valid combinations were explored, covering a wide spectrum from very fine to very coarse ranges, such as 4-4, 4-8, ..., 4-128, 8-8, ..., 8-128, and so on, up to 128-128.
- **Quality Ranges:** For the adaptive quantization process, the system was tested with various minimum and maximum quality level limits. The specific quality values considered for setting the limits were 10, 25, 50, 75, and 90. All valid combinations were explored, covering a wide spectrum from very low to very high quality constraint ranges, such as 10-10, 10-25, ..., 10-90, 25-25, ..., 25-90, and so on, up to 90-90.
- **Chroma Subsampling:** Five chroma subsampling schemes were tested to identify their influence on the tradeoff between compression and quality. These schemes varied in the level of reduction of chrominance resolution in relation to luminance. The 4:4:0 mode contained full horizontal chrominance resolution and half vertical resolution. The 4:2:2, halved the chrominance resolution horizontally but maintained it vertically. The 4:2:0 scheme, which is commonly used, halved chrominance resolution in both the horizontal and vertical planes, producing one-quarter of the total chrominance samples of full resolution. More extreme subsampling was investigated with 4:1:1, which quartered horizontal chrominance resolution but maintained full vertical resolution, and with 4:1:0, which quartered horizontal resolution and halved vertical resolution,

producing one-eighth of the total chrominance samples. This selection of schemes covers a range from mild to aggressive reduction of color information.

The selection of these specific parameters and their respective ranges was driven by the core objectives of this research. The exploration of various block size ranges and quality ranges is critical, as adaptive block partitioning is one of the central innovations within this project, and understanding its behavior for a wide range of block sizes is crucial. The inclusion of newer color spaces along with the traditional YCbCr was motivated by the desire to investigate their capability for higher decorrelation and perceptual uniformity, meaning higher quality and efficiency in compression. Finally, testing various chroma subsampling schemes was essential to evaluate how each color space responds varying levels of subsampling and impact the overall quality-compression tradeoff. This comprehensive approach ensures a robust evaluation of the adaptive system's capabilities.

6.2. Results and Analysis

This section presents empirical findings from the systematic evaluation of the proposed adaptive image compression system, based on results from the comprehensive grid search (Section 6.1.5). For this analysis, performance metrics for each parameter configuration were averaged across all test images. The evaluation first explores the influence of chroma subsampling and different color spaces, and provides an overview of system performance under different optimization scenarios. Subsequently, dominant settings for other key parameters are discussed, concluding with the identification of high-performing configurations.

The relation between chroma subsampling schemes and color space has a significant impact on both compression ratio and preservation of quality. Figure 6.1 displays these trade-offs by graphing the mean Compression Ratio and Composite Score for each subsampling scheme in all five color spaces, relative to the standard JPEG compression baseline on 4:2:0 subsampled YCbCr. The starred bars indicate the best subsampling scheme for that measure in each color space. For (a) ICaCb and (b) ICtCp, the 4:1:1 subsampling scheme usually offers the best Compression Ratio and Composite Score. On the other hand, for perceptually optimized color spaces (c) OKLAB and (e) YCoCg and traditional (d) YCbCr, the 4:2:0 subsampling scheme usually offers the most balanced or optimal performance. Specifically, 4:2:0 offers the best Compression Ratio and Composite Score for (c) OKLAB and (d) YCbCr. For (e) YCoCg, 4:2:0 exhibits the optimal Compression Ratio, and 4:4:0 has a rather better Composite Score, but 4:2:0 is still highly competitive in terms of quality. From this, the optimal subsampling plan identified for each color space is hardcoded into the system and

not user-configurable. This highlights 4:2:0's ability to achieve competitive compression without drastically sacrificing quality in these latter color spaces. It was also observed that configurations utilizing the JzAzBz color space (not depicted in Figure 6.1) consistently failed to surpass the baseline JPEG performance in the given optimization constraints, and hence did not feature in the following analyses.

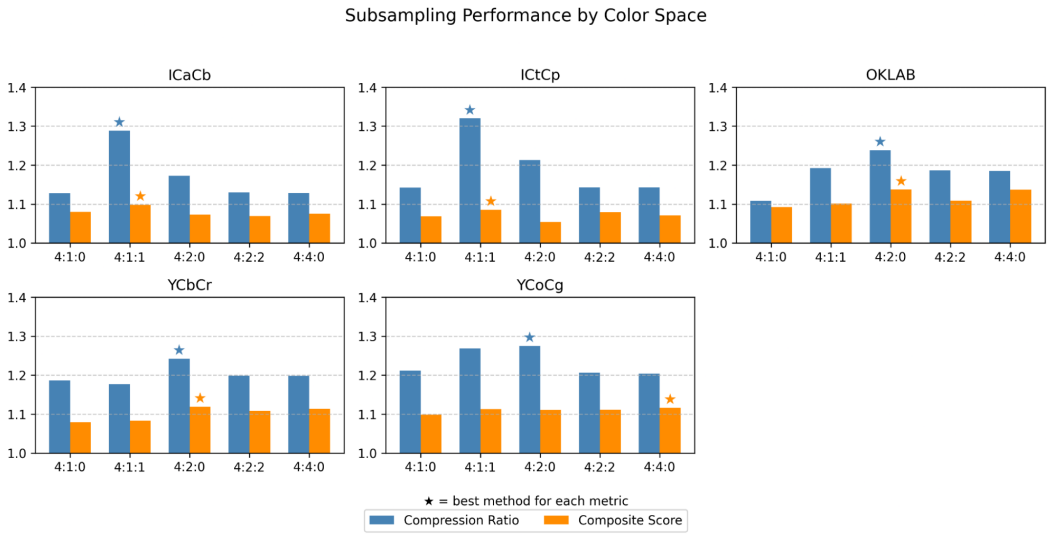

Figure 6.1. Subsampling Performance by Color Space: *(a) ICaCb, (b) ICtCp, (c) OKLAB, (d) YCbCr, and (e) YCoCg.*

A summary of the performance of the adaptive system across the four given optimization strategies (Compression Hard, Compression Soft, Quality Hard, Quality Soft) and the five baseline JPEG quality levels is illustrated in Figure 6.2. The top panel of the figure illustrates the achieved Compression Ratios compared to the baseline JPEG, and the bottom panel presents the corresponding Composite Score comparison ratios. When focusing on maximizing file size reduction, the "Compression Hard" method consistently achieved the highest compression ratios, ranging from approximately 1.60 (60% improvement at baseline Q10) to 1.32 (Q90), while maintaining the 5% tolerance for the Composite Score. The "Compression Soft" strategy, which mandated a Composite Score of 1.0 or better, also delivered significant compression, the ratios varying from 1.47 (Q10) to 1.32 (Q90), effectively balancing file size reduction with guaranteed non-degradation. Conversely, the "Quality Hard" approach, aiming to maximize the Composite Score with a 5% tolerance for file size increase, resulted in maximum quality gains, which had Composite Scores of up to 1.27 (Q50). The "Quality Soft" approach, maximizing quality while ensuring no file size increase, also resulted in notable quality improvement, which had Composite Scores of up to 1.22. This figure clearly demonstrates the adaptive system's ability to navigate the compression-quality tradeoff effectively based on the specified optimization goals.

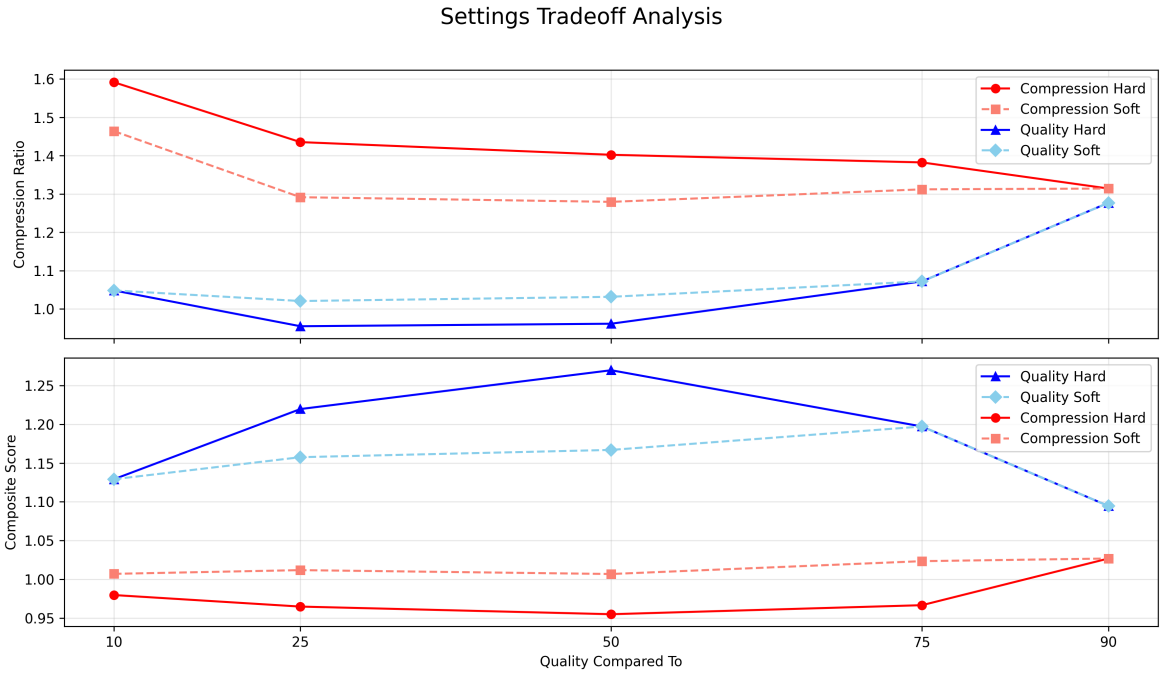

Figure 6.2. Settings Tradeoff Analysis

Further insights into the system's behavior are gained by examining the dominant parameter settings that proved optimal under different conditions. Figure 6.3 reveals that OKLAB and YCoCg were the most frequent color spaces selected. Overall, OKLAB is utilized more when the primary goal is aggressive compression ("Compression Hard"), particularly at greater baselines of quality. YCoCg, on the other hand, is strongly evident in "Compression Soft" and is typically chosen for quality-based methods at both lower and higher levels of baseline quality. This suggests that the system best exploits the special characteristics of these color spaces – perhaps OKLAB's perceptual uniformity helping extreme compression, and YCoCg's component separation helping quality preservation or compression ratio under different constraints.

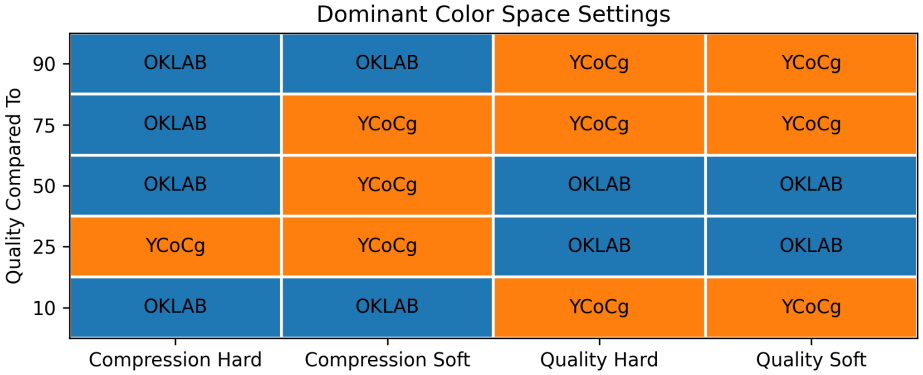

Figure 6.3. Dominant Color Space Settings

Though various subsampling schemes showed promise in isolated tests (as shown in Figure 6.1), Figure 6.4 shows that when integrated into the full evaluation over all parameters, the 4:2:0 subsampling mode for chroma was the general winner. This overall preference, particularly when compared against the baseline of standard JPEG using YCbCr with 4:2:0 subsampling, likely results from its robust equilibrium between compression and perceptual quality. This balance synergizes well with the dominant OKLAB and YCoCg color spaces and the system's adaptive mechanisms, consistently providing the most effective overall solution to meet the defined optimization goals.

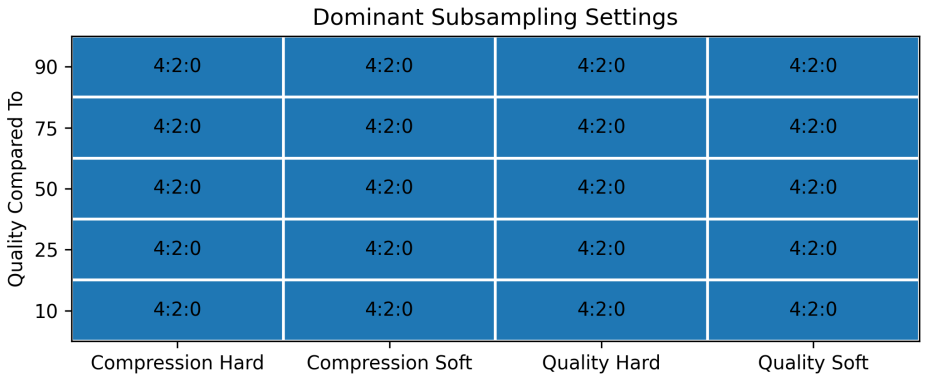

Figure 6.4. Dominant Subsampling Settings

The optimal adaptive block partitioning parameters also varied, as can be seen in Figure 6.5 and Figure 6.6. Analyzing Figure 6.5, the minimum block size of 16x16 was most consistently selected for most scenarios. This suggests that a 16x16 minimum is a suitable overall-purpose starting option for partitioning. However, the occasional selection of 8x8 minimum indicates that in certain optimization situations, allowing the system to begin partitioning at a lower level is advantageous for capturing more intricate image details.

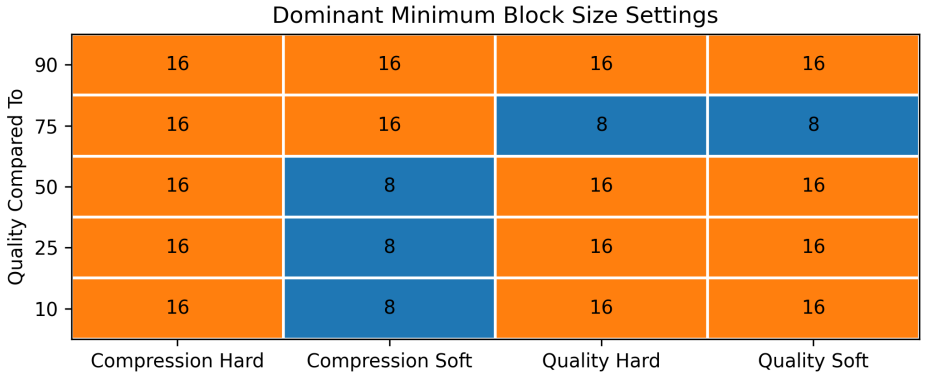

Figure 6.5. Dominant Minimum Block Size Settings

Figure 6.6, illustrating the dominant maximum block size configurations, demonstrates greater variability depending on the optimization strategy and baseline quality. The selection of higher maximums was common in scenarios where maximizing compression for smoother image regions was a priority. Conversely, frequent appearance of smaller maximum block sizes, sometimes as small as 8x8, suggests that such blocks are favored in other situations, particularly where preserving fine detail or meeting high quality standards is critical. The variance across different baseline quality points and optimization priorities clearly demonstrates the flexibility of the system to vary the block size range.

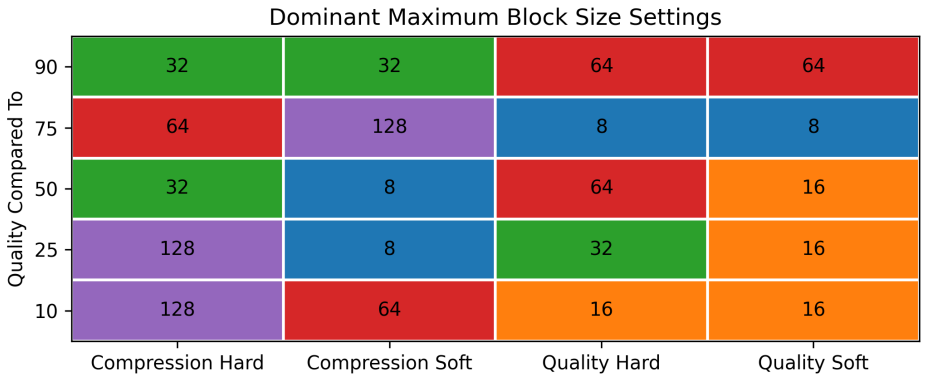

Figure 6.6. Dominant Maximum Block Size Settings

The dominant settings for the adaptive quality range, as shown in Figure 6.7 and Figure 6.8, also adapted to the scenario. The selected minimum quality frequently shows a strong bias towards aligning with the baseline quality level, indicating that the baseline often serves as a reference point for the least aggressively quantized regions. Instances where the chosen minimum surpasses the baseline reflect an effort to enhance quality even in the ”worst” regions of the image, compared to the standard JPEG.

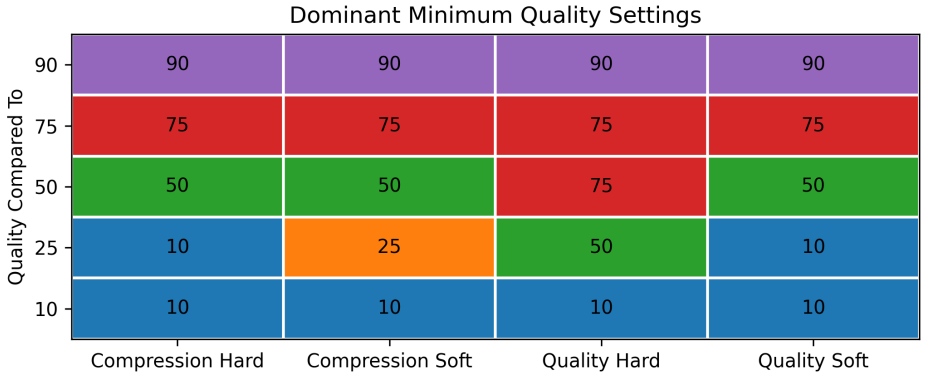

Figure 6.7. Dominant Minimum Quality Settings

For the maximum quality, parameters are often pushed substantially above the corresponding baseline quality. This allows the system to allocate very fine quantization to high-detail regions of the image, preserving fidelity. Conversely, in compression-focused scenarios, maximum quality is always exactly the same as the baseline JPEG quality level. This leads to the system utilizing an adaptive range but not attempting to have any of the image extend above baseline quality level.

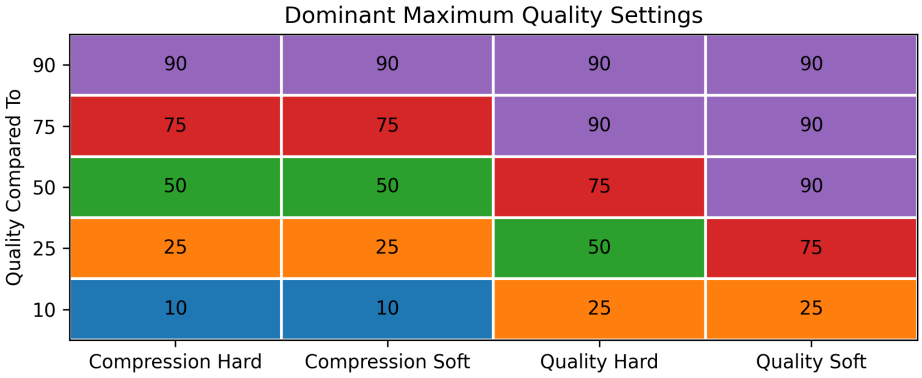

Figure 6.8. Dominant Maximum Quality Settings

To concretely illustrate the adaptive method's practical performance, the standard Lena test image was processed using both standard JPEG and the proposed adaptive system under various optimization priorities: "Compression Soft" and "Quality Soft". The detailed quantitative metrics and specific compression settings for these comparisons are presented systematically. The detailed quantitative metrics and compression settings for the standard JPEG results at five baseline quality levels (10, 25, 50, 75, and 90) are presented in Table 6.1.

Table 6.1. Quantitative Metrics for Standard JPEG [23] Compression of Lena

Color Space	Quality	PSNR Ratio	SSIM Ratio	MS-SSIM Ratio	LPIPS Ratio	Compression Ratio
YCbCr	10	27.5140	0.8992	0.8898	0.2960	31.2945
YCbCr	25	30.4478	0.9606	0.9457	0.1645	19.3312
YCbCr	50	32.0558	0.9808	0.9666	0.1065	13.0576
YCbCr	75	33.2675	0.9905	0.9764	0.0547	8.7981
YCbCr	90	34.8496	0.9964	0.9844	0.0152	5.1784

Metrics for the adaptive method under the ”Compression Soft” scenario (maximizing compression with Composite Score ≥ 1.0) are presented in Table 6.2, and for ”Quality Soft” (maximizing quality with Compression Ratio ≥ 1.0) in Table 6.3. These tables detail specific compression settings, absolute PSNR, SSIM, MS-SSIM, LPIPS ratio values, and the image's absolute Compression Ratio. Furthermore, both tables include key comparative columns for direct comparison with standard JPEG results (from Table 6.1):

- **'Quality Comp. To' (Quality Compared To):** The standard JPEG quality level (from Table 6.1) used as the benchmark for the row's adaptive result.
- **'Abs. Comp. Ratio' (Absolute Compression Ratio):** The actual compression ratio achieved for the image.
- **'Compos. Score vs Stan.' (Composite Score vs Standard):** The adaptive method's Composite Score relative to the benchmarked standard JPEG (1.0x = equivalent).
- **'Comp. Ratio vs Stan.' (Compression Ratio vs Standard):** The adaptive method's Compression Ratio relative to the benchmarked standard JPEG (1.0x = equivalent).

Examining Tables 6.2 and 6.3, particularly the 'Compos. Score vs Stan.' and 'Comp. Ratio vs Stan.' columns alongside the other parameters, allows direct assessment of the adaptive system's performance on Lena. It is crucial to note that these tables show the application of the previously determined best parameter settings (found from the broader parameter analyses outlined in earlier sections, summarized in Figures 6.3-6.8) to this specific image. Therefore, while the primary metric for each ”Soft” scenario was targeted, the secondary metric's performance reflects how these generalized optimal settings perform on Lena and might not always meet the non-degradation constraint (being ≥1.0) for this single image.

Table 6.2 details ”Compression Soft” optimal settings on Lena, where the initial goal was maximizing compression while ensuring the Composite Score was at least equivalent to standard JPEG. For Lena, these settings achieved significantly higher compression ratios, ranging from 1.24x (at Q50) up to an impressive 1.54x (at Q10), compared to standard JPEG. The Composite Score remained generally competitive, often close to 1.0, though at Q90 it dipped to 0.91x, illustrating that generalized settings might not perfectly meet secondary constraints on every individual image, though these values are still within or very close to the universal 5% tolerance. The often identical 'Quality Range' settings imply Lena's compression gains stem primarily from adaptive block sizing utilizing varied 'Block Size Ranges'.

Table 6.2. Quantitative Metrics for Adaptive Compression (Compression Soft) of Lena

Quality Comp. To	Color Space	Quality Range	Block Size Range	PSNR Ratio	SSIM Ratio	MS-SSIM Ratio	LPIPS Ratio	Abs. Comp. Ratio	Compos. Score vs Stan.	Comp. Ratio vs Stan.
10	YCoCg	10-10	16-128	28.24	0.91	0.91	0.30	48.24	1.01x	1.54x
25	YCoCg	25-25	8-128	30.75	0.96	0.95	0.18	24.49	0.98x	1.27x
50	YCoCg	50-50	8-128	32.26	0.98	0.97	0.11	16.21	0.99x	1.24x
75	YCoCg	75-75	16-32	33.55	0.99	0.98	0.05	11.51	1.02x	1.31x
90	OKLAB	90-90	16-32	34.86	0.99	0.98	0.02	7.18	0.91x	1.39x

Conversely, Table 6.3 presents ”Quality Soft” optimal settings on Lena, where the parameter search prioritized maximizing the Composite Score while ensuring the compression ratio was at least equivalent to standard JPEG. These settings consistently delivered notable perceived quality improvements for Lena, evidenced by Composite Scores ranging from 1.09x (at baseline Q25) up to a substantial 1.31x (at Q75) compared to standard JPEG. The compression ratio generally maintained parity with the baseline, often exceeding 1.0x and thus fulfilling the secondary constraint, though occasionally dipping slightly below, again reflecting the performance of generalized settings on a specific image.

Table 6.3. Quantitative Metrics for Adaptive Compression (Quality Soft) of Lena

Quality Comp. To	Color Space	Quality Range	Block Size Range	PSNR Ratio	SSIM Ratio	MS-SSIM Ratio	LPIPS Ratio	Abs. Comp. Ratio	Compos. Score vs Stan.	Comp. Ratio vs Stan.
10	YCoCg	10-25	16-16	29.85	0.94	0.94	0.23	33.66	1.13x	1.08x
25	YCoCg	10-75	16-16	31.97	0.98	0.97	0.13	19.11	1.09x	0.99x
50	YCoCg	50-90	16-16	33.28	0.99	0.98	0.06	12.60	1.24x	0.96x
75	YCoCg	75-90	16-16	34.12	0.99	0.98	0.03	9.39	1.31x	1.07x
90	YCoCg	90-90	16-128	35.06	1.00	0.99	0.01	6.61	1.09x	1.28x

In summary, these tables show optimal ”Compression Soft” and ”Quality Soft” settings on Lena. Prioritizing compression, adaptive block sizing was key to higher compression. For quality, benefits often stemmed from superior fixed configurations (e.g., YCoCg, 16x16 blocks), rather than adaptive allocation. This highlights the system’s effective use of its components, even with generalized optimal configurations on individual images.

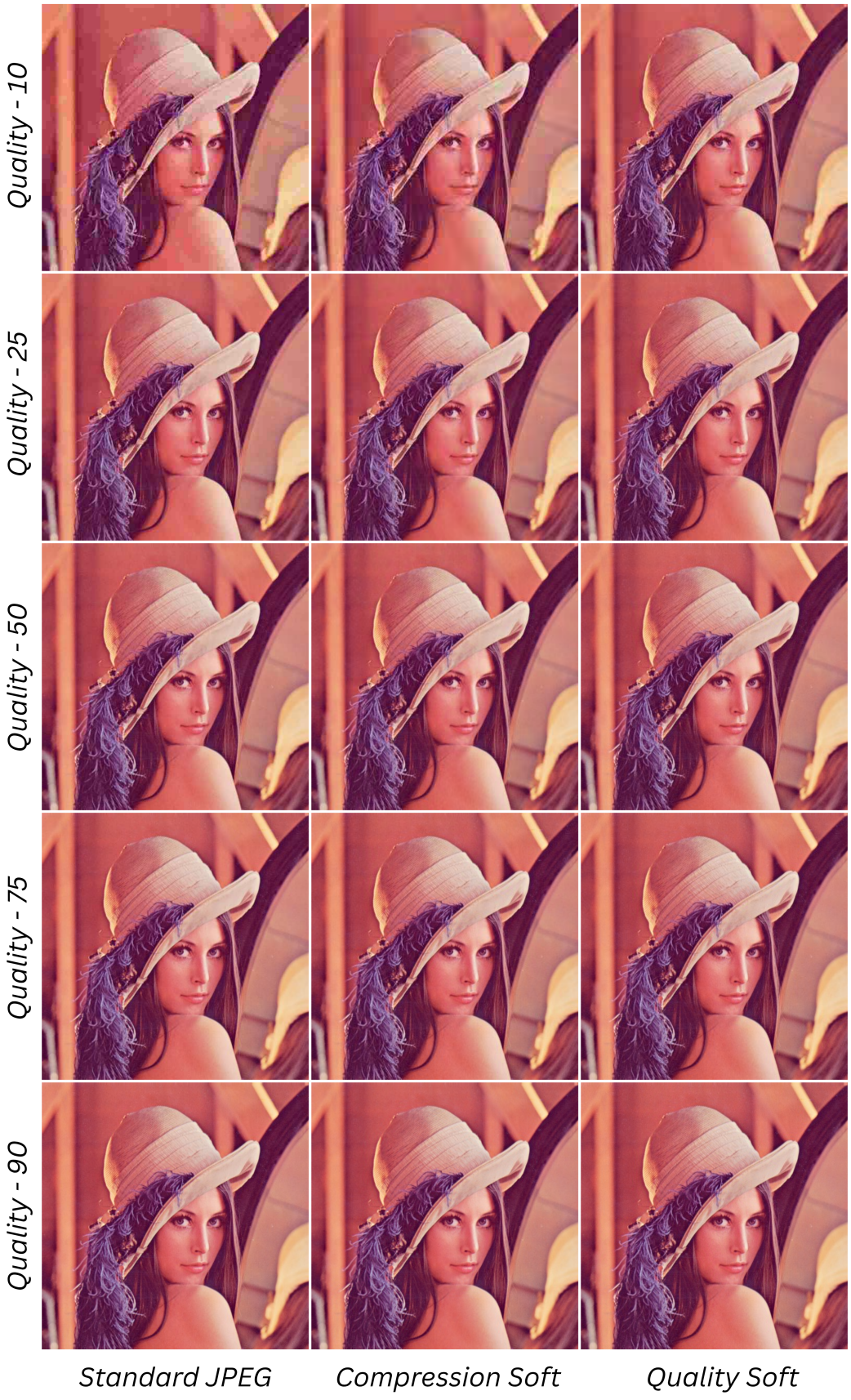

Figure 6.9. Lena Image Comparison *(Columns: Standard JPEG, Adaptive Compression Soft, Adaptive Quality Soft)*

Figure 6.9 complements these tables visually, providing a side-by-side view of the compressed Lena images. The figure presents a 5x3 grid layout where each row is allocated to a specific baseline JPEG target quality (10, 25, 50, 75, and 90) matching the format of the preceding tables. The left column in each row displays the Lena image compressed by standard JPEG (detailed in 6.1). The middle column contains the Lena image compressed with the adaptive ”Compression Soft” method (detailed in Table 6.2). The right column shows the Lena image compressed with the adaptive ”Quality Soft” method (detailed in Table 6.3).

A visual analysis of Figure 6.9 reveals distinct trends across different quality levels. At lower baseline qualities (Q10, Q25), standard JPEG images (left) show pronounced blockiness, especially in the background and facial features. ”Quality Soft” images (right) are visually superior, with significantly reduced artifacts and better detail, appearing smoother. ”Compression Soft” images (middle) also improve upon standard JPEG, though not good as ”Quality Soft”. As baseline quality increases to mid range (Q50, Q75), standard JPEG and ”Compression Soft” images become nearly indistinguishable, while ”Quality Soft” versions still offer subtle improvements, especially in smoother gradients and texture rendering. At the highest baseline (Q90), all three versions are visually very similar, with only minimal, if any, perceptible differences. This visual comparison, supported by the quantitative data in the tables, allows for an assessment of differences in artifact suppression, detail preservation, overall file size reduction, and the visualization of perceptual differences that may not be fully captured by metrics alone, under different optimization priorities. The setup clearly demonstrates the adaptive system’s ability for superior compression efficiency while maintaining or improving quality (”Compression Soft”) and significantly enhancing perceptual quality without file size penalty (”Quality Soft”), especially at lower to mid-quality baselines.

The comprehensive empirical evaluation confirms the proposed adaptive image compression system’s significant advantages over standard JPEG across diverse optimization objectives. The system effectively enhances both the compression and the quality by strategically selecting parameters: OKLAB and YCoCg emerged as dominant color spaces, 4:2:0 chroma subsampling proved optimal when integrated into the full optimization, and block size and quality ranges were dynamically adjusted based on the specific compression or quality priority. This adaptive method consistently allowed the system to, for example, reduce file size by 25-55% without perceptible visual degradation, or alternatively, increase perceived quality by 10-30% while maintaining or reducing file size. All these findings, supported by quantitative performance across varied settings as well as specific image assessment for Lena, validate the substantial benefits of this edge-aware, adaptive compression method.

7. CONCLUSION & FUTURE WORK

In an era dominated by digital imagery, effective image compression is crucial for efficient storage and transmission. While standard JPEG is widely used, its fixed-size block processing and non-adaptive quantization have a tendency to create noticeable artifacts and a suboptimal trade-off between file size and perceived quality—e.g., blocking in complex areas and insufficient compression in smooth areas—making the demand for an enhancement.

This project presents an adaptive, edge-aware image compression system that aims to overcome these JPEG limitations using a smarter, perceptually driven approach. The key innovation is edge detection guided adaptive block partitioning, using smaller blocks for high-detail areas and larger blocks for smooth regions, coupled with adaptive quantization determined by block size and user-defined quality targets. The system operates on input images by initially converting the color space, followed by Canny-based edge detection to direct quadtree-based partitioning. Variable-size blocks are then subjected to Discrete Cosine Transform (DCT), adaptive quantization, and entropy encoding. Support for modern color spaces and an intuitive GUI with real-time feedback further enhance practical utility.

Empirical evaluation against baseline JPEG confirmed the efficacy of the system. When prioritizing compression, the adaptive approach gained 25-55% file size saving while preserving or enhancing image quality in terms of a composite score of PSNR, SSIM, MS-SSIM, and LPIPS. Conversely, when prioritizing quality, it provided a 10-30% quality gain, often without increasing, and sometimes even lowering, file size. Results emphasized the efficiency of color spaces such as OKLAB and YCoCg, 4:2:0 chroma subsampling, and the importance of adaptive block sizes, with visual findings supporting reduced blocking artifacts and better detail preservation, particularly at lower to mid-range qualities.

Several areas for future work could further enhance the system: allowing user-configurable individual chroma subsampling ratios per each axis; refining quantization matrix creation from basic rescaling by considering advanced computational techniques or by leveraging precomputed and experimentally derived matrices; using modern entropy coders such as Asymmetric Numeral Systems; lossless compression of metadata; and optimizing execution time for large images or batch operations using parallelization, compiled languages, or GPU acceleration. At last, further quantitative testing over extensive scenarios, along with thorough subjective visual testing involving human observers, is still necessary in order to completely validate and optimize the perceptual quality of the adaptive algorithm.

Bibliography

[1] M. Broz, *Photo statistics: How many photos are taken every day?* `https://photutorial.com/photos-statistics/`, [Accessed 30-05-2025], 2025.

[2] G. Hudson, A. Léger, B. Niss, I. Sebestyén, and J. Vaaben, "Jpeg-1 standard 25 years: Past, present, and future reasons for a success," *Journal of Electronic Imaging*, vol. 27, no. 4, pp. 040 901–040 901, 2018.

[3] F. Alter, S. Durand, and J. Froment, "Adapted total variation for artifact free decompression of jpeg images," *Journal of mathematical Imaging and Vision*, vol. 23, no. 2, pp. 199–211, 2005.

[4] F. Douak, R. Benzid, and N. Benoudjit, "Color image compression algorithm based on the dct transform combined to an adaptive block scanning," *AEU-International Journal of Electronics and Communications*, vol. 65, no. 1, pp. 16–26, 2011.

[5] R. Benzid, F. Marir, and N.-E. Bouguechal, "Electrocardiogram compression method based on the adaptive wavelet coefficients quantization combined to a modified two-role encoder," *IEEE Signal Processing Letters*, vol. 14, no. 6, pp. 373–376, 2007.

[6] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[7] W. Lin and L. Dong, "Adaptive downsampling to improve image compression at low bit rates," *IEEE Transactions on Image Processing*, vol. 15, no. 9, pp. 2513–2521, 2006.

[8] J. Alakuijala, R. Obryk, O. Stoliarchuk, Z. Szabadka, L. Vandevenne, and J. Wassenberg, "Guetzli: Perceptually guided jpeg encoder," *arXiv preprint arXiv:1703.04421*, 2017.

[9] N. A. Abu and M. R. K. Ariffin, "A novel psychovisual model on an independent video frame for an almost lossless compression," in *2014 10th International Conference on Information Assurance and Security*, IEEE, 2014, pp. 60–65.

[10] C.-Y. Wang, S.-M. Lee, and L.-W. Chang, "Designing jpeg quantization tables based on human visual system," *Signal Processing: Image Communication*, vol. 16, no. 5, pp. 501–506, 2001.

[11] M. Feischl and H. Hackl, "Adaptive image compression via optimal mesh refinement," *Computational Methods in Applied Mathematics*, vol. 24, no. 2, pp. 325–343, 2024.

[12] F. Ernawan, N. A. Abu, and N. Suryana, "An adaptive jpeg image compression using psychovisual model," *Advanced Science Letters*, vol. 20, no. 1, pp. 26–31, 2014.

[13] R. Rosenholtz and A. B. Watson, "Perceptual adaptive jpeg coding," in *Proceedings of 3rd IEEE International Conference on Image Processing*, IEEE, vol. 1, 1996, pp. 901–904.

[14] M. G. Ramos and S. S. Hemami, "Edge-adaptive jpeg image compression," in *Visual Communications and Image Processing'96*, SPIE, vol. 2727, 1996, pp. 1082–1093.

[15] Y. Ouyang, X. Wang, G. Shi, L. Chen, and F. An, "A dynamic jpeg codec with adaptive quantization table for frame storage compression," in *2022 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, IEEE, 2022, pp. 119–122.

[16] O. Rippel and L. Bourdev, "Real-time adaptive image compression," in *International Conference on Machine Learning*, PMLR, 2017, pp. 2922–2930.

[17] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," *arXiv preprint arXiv:1611.01704*, 2016.

[18] C. Reich, B. Debnath, D. Patel, and S. Chakradhar, "Differentiable jpeg: The devil is in the details," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 4126–4135.

[19] L. J. JR and J. V. R. SP, "Enhanced edge detection for image segmentation and its real-time implementation," in *2024 28th International Symposium on VLSI Design and Test (VDAT)*, IEEE, 2024, pp. 1–6.

[20] M. Zhang and B. K. Gunturk, "Compression artifact reduction with adaptive bilateral filtering," in *Visual Communications and Image Processing 2009*, SPIE, vol. 7257, 2009, pp. 413–423.

[21] B. Kawar, J. Song, S. Ermon, and M. Elad, "Jpeg artifact correction using denoising diffusion restoration models," *arXiv preprint arXiv:2209.11888*, 2022.

[22] Y. Li, F. Guo, R. T. Tan, and M. S. Brown, "A contrast enhancement framework with jpeg artifacts suppression," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II 13*, Springer, 2014, pp. 174–188.

[23] T. Recommendation, "Ccitt t. 81," 1993.

[24] S. N. Kumar, M. V. Bharadwaj, and S. Subbarayappa, "Performance comparison of jpeg, jpeg xt, jpeg ls, jpeg 2000, jpeg xr, hevc, evc and vvc for images," in *2021 6th International Conference for Convergence in Technology (I2CT)*, IEEE, 2021, pp. 1–8.

[25] Devcore, *8x8 dct (discrete cosine transformation)*, Public domain. Retrieved March 31, 2025, 2012. [Online]. Available: `https://commons.wikimedia.org/wiki/File:DCT-8x8.png`.

[26] A. Khristov, *Zig-zag pattern used in en:jpeg entropy coding*. Public domain. Retrieved March 31, 2025, 2007. [Online]. Available: https://commons.wikimedia.org/wiki/File:JPEG_ZigZag.svg.

[27] I. IEC, "Iso/iec 15444,"

[28] J. Alakuijala, R. Van Asseldonk, S. Boukortt, *et al.*, "Jpeg xl next-generation image compression architecture and coding tools," in *Applications of digital image processing XLII*, SPIE, vol. 11137, 2019, pp. 112–124.

[29] J. Duda, "Asymmetric numeral systems: Entropy coding combining speed of huffman coding with compression rate of arithmetic coding," *arXiv preprint arXiv:1311.2540*, 2013.

[30] M. Anderson, R. Motta, S. Chandrasekar, and M. Stokes, "Proposal for a standard default color space for the internet—srgb," in *Color and imaging conference*, Society of Imaging Science and Technology, vol. 4, 1996, pp. 238–245.

[31] T. Smith and J. Guild, "The cie colorimetric standards and their use," *Transactions of the optical society*, vol. 33, no. 3, p. 73, 1931.

[32] BenRG, *Cie 1931 xy color space diagram*. Public domain. Retrieved March 31, 2025, 2009. [Online]. Available: https://commons.wikimedia.org/wiki/File:CIE1931xy_blank.svg.

[33] B. Series, "Studio encoding parameters of digital television for standard 4: 3 and widescreen 16: 9 aspect ratios," *International Telecommunication Union, Radiocommunication Sector*, 2011.

[34] Mike1024, *Barns grand tetons ycbcr separation*, Public domain. Retrieved March 31, 2025, 2006. [Online]. Available: https://commons.wikimedia.org/wiki/File:Barns_grand_tetons_YCbCr_separation.jpg.

[35] H. Malvar and G. Sullivan, "Transform, scaling & color space impact of professional extensions," *ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q*, vol. 6, 2003.

[36] B. Ottosson, *A perceptual color space for image processing*, https://bottosson.github.io/posts/oklab/, [Accessed 12-04-2025], 2020.

[37] J. Froehlich, T. Kunkel, R. Atkins, *et al.*, "Encoding color difference signals for high dynamic range and wide gamut imagery.," in *CIC*, 2015, pp. 240–247.

[38] B. Series, "Image parameter values for high dynamic range television for use in production and international programme exchanges," *International Telecommunication Union, Radiocommunication Sector*, 2025.

[39] M. Safdar, G. Cui, Y. J. Kim, and M. R. Luo, "Perceptually uniform color space for image signals including high dynamic range and wide gamut," *Optics express*, vol. 25, no. 13, pp. 15 131–15 151, 2017.

[40] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*, IEEE, 1998, pp. 839–846.

[41] Zefram, *Schematische darstellung einer histogrammeinebnung*, Public domain. Retrieved March 31, 2025, 2006. [Online]. Available: `https://commons.wikimedia.org/wiki/File:Histogrammeinebnung.png`.

[42] S. M. Pizer, E. P. Amburn, J. D. Austin, *et al.*, "Adaptive histogram equalization and its variations," *Computer vision, graphics, and image processing*, vol. 39, no. 3, pp. 355–368, 1987.

[43] K. J. Zuiderveld *et al.*, "Contrast limited adaptive histogram equalization.," *Graphics gems*, vol. 4, no. 1, pp. 474–485, 1994.

[44] Vswitchs, *Excess redistribution in contrast-limited adaptive histogram equalization.* CC0 1.0 Universal – Public Domain Dedication. Retrieved March 31, 2025, 2011. [Online]. Available: `https://commons.wikimedia.org/wiki/File:Clahe-redist.svg`.

[45] I. Sobel, "An isotropic 3x3 image gradient operator," *Presentation at Stanford A.I. Project 1968*, Feb. 2014.

[46] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[47] JonMcLoone, *Edge detection applied to a photograph*, Creative Commons Attribution-Share Alike 3.0 Unported. Retrieved March 31, 2025, 2010. [Online]. Available: `https://commons.wikimedia.org/wiki/File:%C3%84%C3%A4retuvastuse_n%C3%A4ide.png`.

[48] R. A. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta informatica*, vol. 4, pp. 1–9, 1974.

[49] W. Muła, *Bitmap 8x8 pixels repsented with compressed quadtree.* Public domain. Retrieved March 31, 2025, 2008. [Online]. Available: `https://commons.wikimedia.org/wiki/File:Quad_tree_bitmap.svg`.

[50] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[51] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, Ieee, vol. 2, 2003, pp. 1398–1402.

[52] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 586–595.

[53] Python Software Foundation, *The python logo*, Retrieved: 14 May 2025. [Online]. Available: `https://www.python.org/community/logos/`.

[54] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on information theory*, vol. 23, no. 3, pp. 337–343, 1977.

[55] H. R. Sheikh, Z. Wang, L. Cormack, and A. C. Bovik, "Live image quality assessment database release 2 (2005)," *URL http://live.ece.utexas.edu/research/quality*, 2005.

[56] H. R. Sheikh, M. F. Sabir, and A. C. Bovik, "A statistical evaluation of recent full reference image quality assessment algorithms," *IEEE Transactions on image processing*, vol. 15, no. 11, pp. 3440–3451, 2006.