International Journal of Computer Engineering in Research Trends
ISSN: 2349-7084/ https://doi.org/10.22362/ijcert.v10i8.786
Volume 10, Issue 8, August 2023

Research Paper

Optimizing Digital Image Quality Assessment: Format Selection and Methods

Kayithi Varshitha[1], Battina Sneha Sree [2], Salikuti Varunika Reddy[3], Semarla Sreeja Reddy[4], K Venkatesh Sharma[5]

[1,2,3,4] *B.Tech Student ,Department of Computer Science & Engineering, CVR College of Engineering, Rangareddy Dist, Telangana, India*
[5] *Professor, Department of Computer Science & Engineering, CVR College of Engineering, Rangareddy Dist, Telangana, India*

e-mail: varshitha.kayithi@gmail.com, snehabattina2003@gmail.com, varunikasalikuti2000@gmail.com, sreejareddy072@gmail.com, venkateshsharma.cse@gmail.com

**Corresponding Author: venkateshsharma.cse@gmail.com*

Received: 11/07/2023, Revised: 21 /07/2023, Accepted: 08/08/2023 Published: 16/08/2023

Abstract: - *The evaluation of computer-stored digital image quality currently relies on subjective measures based on user preferences. The landscape includes a plethora of diverse file formats with unique features that can either enhance or detract from image quality. Selecting the optimal format is a challenge for both average users and system developers, requiring a comprehensive understanding of available formats, their attributes, implications on image quality, and the specific image data they will handle. This places undue stress on users, potentially leading to unsuitable format choices. Consequently, making an informed initial decision becomes paramount. This study aims to identify prevalent quality assessment methods used in related industries for images stored across a range of file formats, and to propose effective implementation strategies. Drawing upon insights from these methods and delving into popular file formats and compression techniques, I propose practical suggestions on a general level. In a bid to enhance my comprehension of format-related challenges and their impact on image quality, a key aspect of this research involves developing a graphics library. This software component facilitates seamless conversion between numerous popular graphics formats, thereby fortifying the understanding of format issues and compression/storage effects on image quality.*

Keywords- *Image Quality Evaluation, File Format Selection, Quality Assessment Methods, Compression Techniques, Graphics Library Development, Format-Related Challenges*

---

1. Introduction

The realm of graphics file formats and conversion applications has sparked my curiosity since my early engagement with computing. Personal experiences using graphic images for coursework prompted questions about the multitude of formats and storage methods available. This project offers a chance to explore these questions and gain insights into the world of graphics files.

At the project's outset, my familiarity with this field was casual, encompassing a general understanding of bitmaps while lacking specific knowledge about formats, compression techniques, and overall graphic image structures [1]. Given my career aspirations in this domain, studying the measurement of image 'quality' and its link to format selection emerged as a natural and engaging research pursuit.

The learning curve of this endeavor has proven steeper than previous projects, culminating in the development of my first comprehensive software product. My prior knowledge of C and basic skills in Pascal from Borland Delphi were insufficient for the project's scope. Consequently, I undertook a significant learning process to master these languages for creating file conversion software [2].

This research delves into the principles of image storage, including compression and decompression, color spaces, color system conversion, image display, file format conversion, and advanced techniques like real-time full-motion video enhancement.

The fundamental objectives of this paper include:

Contextualizing the Importance of File Formats and Compression Methods: Highlight the ubiquity of digital data across diverse applications, including images, audio, video, and documents. And Emphasize that the choice of file format and compression method impacts data integrity, storage efficiency, transmission speed, and user experience.

Challenges in Digital Data Management: Discuss the challenges associated with managing and sharing diverse forms of digital content, such as large file sizes, bandwidth limitations, and the need for quick data access. Point out that addressing these challenges requires an understanding of various file formats and compression techniques.

Addressing the Need for Structured Approach: Introduce the provided methodology as a structured approach for comprehensively understanding and comparing different file formats and compression methods. Explain that the methodology is designed to assist in making informed decisions based on the specific requirements of data storage, transmission, and quality preservation.

The subsequent sections of this paper have been structured as follows: In Section 2, a comprehensive literature survey is presented. Section 3 delves into the realm of File Formats and Compression Methods. The intricate interplay of Factors Affecting Image Quality and Higher Level Considerations is explored in Section 4. Section 5 navigates through the intricacies of Graphic File Handling. Finally, the paper is brought to a conclusion in Section 6, which also outlines potential avenues for future research and development.

2. Literature Survey

The investigation into current image quality assessment practices reveals the multifaceted challenges associated with accurately measuring image quality across different file formats. To gain a comprehensive understanding, the survey encompasses several key aspects:

2.1 Image Quality Measurement Challenges

The diversity of subjective human perception serves as a foundational challenge in objectively assessing image quality. With each individual possessing unique visual abilities and preferences, relying solely on human judgment for quality assessment leads to inherent biases. Factors such as equipment disparities, variations in human vision, environmental conditions, and personal biases influence perception and create inconsistencies in judgment. Equipment limitations, including monitor quality and graphics capabilities, introduce distortions that skew quality perceptions. The complexity of human vision means that fine differences, which might be captured by advanced formats, are often undetectable to the naked eye. Moreover, variations in lighting conditions and viewer bias further compound the challenge of consistent image quality assessment [3].

2.2 Industry Insights

Feedback obtained from industry sources sheds light on the nuanced considerations guiding image format choices. Jeffrey Glover from ASAP Inc. underscores the significance of speed versus quality, particularly in applications like web graphics. The prioritization of speed over high quality necessitates format choices based on the intended purpose of the image. The National Remote Sensing Centre (NRSC) [4] emphasizes user judgment in applications like color mapping of remote sensing scans, highlighting the absence of standardized methods for quality assessment. Their experience underscores challenges arising from platform and software compatibility when transferring images across various formats. Similarly, Nick Efford from the Centre of Medical Imaging Research (CoMIR) [5] discusses the balance between lossless and lossy compression, influenced by the importance of high compression for some applications and the retention of quality for others.

2.3 Avenues for Improvement

The literature survey accentuates the need for a systematic and scientific approach to image quality measurement that transcends individual perceptions. The challenges posed by subjective judgment, equipment limitations, and variations in human vision necessitate a method that filters out these biases. The industry feedback underscores the complexity of format selection, where specific application requirements drive decisions, and a lack of standardized methods is apparent.

In light of these findings, this study endeavors to address the challenges highlighted in the literature survey. It aims to propose methods for measuring image quality objectively across diverse graphic file formats, considering the complexities outlined by industry experts. By amalgamating theoretical insights, practical knowledge, and a scientific approach, this research seeks to contribute to the development of a more standardized and objective image quality assessment framework that takes into account the intricacies of format selection, human perception, and application-specific demands.

3. File Formats and Compression Methods

In the world of computer systems and applications, various file formats have emerged to cater to specific software applications and use cases. These formats are designed to store and represent different types of data. Additionally, compression methods have been developed to reduce the size of files, making them easier to store and transmit [6]. Let's explore these concepts in more detail:

3.1 File Formats: File formats are standardized structures used to organize and store data in digital form. They vary depending on the type of data they are designed to store, such as images, audio, video, or documents. Here are three primary categories of file formats, along with their advantages, disadvantages, and examples:

Three primary categories of file formats include:

3.1.1 Vector-based Formats: These formats represent images using mathematical statements that define objects as vectors. Originally limited to lines, these formats can now construct complex objects from various curves and

polygons. Vector formats are commonly used in Computer-Aided Design (CAD)[7] for technical drawings and mechanical designs. Due to their compact storage method, compression is less effective for vector formats.

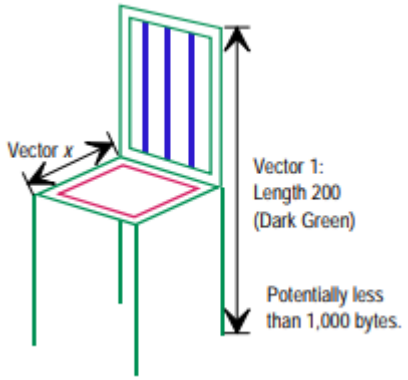

Figure 1. Vector Representation of a Chair

Advantages:
Scalability: Vector images can be resized without losing quality because they are based on mathematical equations.
Small File Sizes: Vector files are typically smaller compared to bitmap files, making them suitable for graphics that need to be scaled.

Disadvantages:
Limited Realism: Vector graphics might not capture intricate details of real-world scenes as effectively as bitmaps.
Complexity: Representing some complex images using vectors can be challenging.
Examples: SVG (Scalable Vector Graphics), AI (Adobe Illustrator), EPS (Encapsulated PostScript)

3.1.2 Bitmap-based Formats: Bitmap formats [8] store images as grids of pixels, where each pixel carries color information. These formats are suitable for representing real-world scenes and detailed images. Bitmap-based formats support compression techniques to reduce file sizes.

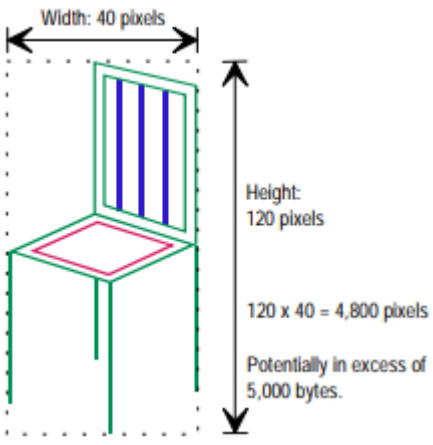

Figure 2. Bitmap Representation of a Chair
Advantages:

- **Realism:** Bitmap images can represent intricate details, making them suitable for photographs and realistic graphics.
- **Widespread Compatibility:** Bitmaps are supported by a wide range of software and devices.

Disadvantages:
File Size: Bitmaps can become large in size, especially for high-resolution images.
Limited Scalability: Enlarging bitmap images can lead to pixelation and loss of quality**.**

Examples: JPEG (Joint Photographic Experts Group), PNG (Portable Network Graphics), BMP (Bitmap)

3.1.3 Metafile-based Formats: Metafile formats [9] store both vector and bitmap data and are often used for exchanging graphics data between different software applications. They offer a compromise between vector and bitmap capabilities.
Advantages:
Versatility: Metafiles can store both vector and bitmap data, offering a balance between scalability and realism.
Interoperability: They can be exchanged between different software applications without losing data fidelity.
Disadvantages:
Complexity: Metafiles can be more complex to implement due to the need to handle both vector and bitmap data.
Examples: PDF (Portable Document Format), WMF (Windows Metafile), EMF (Enhanced Metafile)

3.2 Bitmap Compression Algorithms

3.2.1 Symmetric and Asymmetric Compression: Suppose you're creating an animated movie using the MPEG format[10]. During creation, symmetric compression is employed, involving a complex and time-consuming process to ensure optimal compression. However, during playback on regular devices, the emphasis shifts to fast decompression to maintain real-time playback performance. Conversely, if you're working on a real-time object tracking system, you might prioritize asymmetric compression. Compression is quicker to keep up with the fast-paced tracking, while decompression can be more intricate to achieve better compression ratios.
3.2.2 Non-Adaptive, Semi-Adaptive, and Adaptive Encoding: Imagine you're working on an image editing software that offers compression for saved projects. For non-adaptive encoding, you might choose a predefined dictionary containing frequently used patterns like common shapes. Semi-adaptive encoding involves analyzing the data to identify recurring patterns before compressing, offering improved compression compared to non-adaptive methods. Adaptive encoding excels further by dynamically adjusting the dictionary during compression, achieving the best compression ratios.

3.2.3 Lossless vs. Lossy: Consider a digital photo album app. For family photos, you might choose a lossless compression method to preserve image quality, ensuring that decompressed images match the originals. On the other hand, for an app focused on sharing memes, you might opt for lossy compression like JPEG. This would remove subtle color variations and fine details that might not be noticeable

to the human eye, resulting in smaller file sizes while maintaining acceptable image quality for the intended purpose.

3.2.4 Compression Methods: Compression methods are techniques used to reduce the size of files while preserving their essential data. There are two main categories of compression: lossless and lossy[11].

3.2.4.1 Lossless Compression: Lossless compression techniques reduce file size without losing any data. These methods are crucial when data integrity is essential. Common lossless methods include:

- Run-Length Encoding (RLE)[12]: Replaces repeated sequences of data with a code representing the sequence and its count. Effective for images with large areas of the same color.
- Lempel-Ziv Welch (LZW): A dictionary-based method that replaces recurring data sequences with references to a dictionary. LZW [13] is widely used in formats like GIF and TIFF.

Consider a simple image represented by a sequence of pixel values:

$$AAAABBBCCDAA$$

Using RLE compression, this sequence would be encoded as: $4A3B2C1D2A$
Here, "4A" means four consecutive 'A' values, "3B" means three consecutive 'B' values, and so on.
Advantages of RLE:

- Extremely simple and easy to implement.
- Highly effective for images with large areas of the same color.

Disadvantages of RLE:
Limited compression efficiency for complex data with minimal repeated sequences.
May not achieve significant compression for random or complex patterns.

Lempel-Ziv Welch (LZW): Lempel-Ziv Welch (LZW) is a widely used dictionary-based compression method that's particularly effective for compressing text and data with repeating patterns.
How LZW Works: LZW builds a dictionary of frequently occurring patterns in the data. It then replaces these patterns with shorter codes, effectively reducing the size of the data. As the compression process continues, the dictionary is expanded to accommodate new patterns.
Example: Let's compress the text "ABABABAABABAABABAAA":

- The initial dictionary contains single characters: A, B.
- As the compression progresses: "A" and "B" remain as they are, "AB" is assigned a new code, and so on.
- The compressed output might look like a sequence of codes: 1 2 3 4 5 6 7 4 5 8 9.

Advantages of LZW:

- Effective for compressing text and data with repetitive patterns.
- Can achieve higher compression ratios compared to simple methods like RLE.

Disadvantages of LZW:

- Relatively more complex to implement than simple methods.
- May not be as efficient for data with minimal repetition.

Advantages:
No Data Loss: Lossless compression retains all original data, making it suitable for text documents and archival purposes.
Exact Reconstruction: Decompressed data is identical to the original, ensuring data integrity.
Disadvantages:
Modest Compression Ratio: Lossless compression doesn't achieve as high a compression ratio as lossy methods.
Limited for Media: Less effective for compressing multimedia files like images and audio.
Examples: ZIP (lossless data compression), FLAC (Free Lossless Audio Codec), PNG (lossless image compression)

3.2.4.2 Lossy Compression: Lossy compression reduces file size by discarding some data that might not be easily noticeable to the human eye or ear. These methods are suitable for multimedia content where slight quality loss is acceptable. Common lossy methods include:

- Huffman Coding: Assigns shorter codes to more frequently occurring symbols in the data stream. While it's efficient, it doesn't guarantee optimal storage.
- Arithmetic Coding: Represents data as a floating-point number between 0 and 1, allowing optimal storage. It achieves better compression efficiency than Huffman coding.

Advantages:
Higher Compression: Lossy methods achieve higher compression ratios, reducing file sizes significantly.
Suitable for Media: Effective for compressing multimedia files where slight quality loss is acceptable.
Disadvantages:
Quality Loss: Lossy compression results in a reduction of data quality, which may be noticeable in some cases.
Irreversible: Decompressed data won't be identical to the original, potentially leading to data loss.
Examples: JPEG (lossy image compression), MP3 (lossy audio compression), MPEG (lossy video compression)

Understanding file formats and compression methods is essential for effectively managing and sharing digital content. Selecting the right format and compression technique depends on factors such as the type of data, the desired level of quality, the available storage or bandwidth, and the intended use of the files. Balancing advantages and disadvantages is key to making informed decisions about which formats and compression methods to employ.

Pixel Packing: Imagine you're creating an app that displays pixel art. To efficiently store pixel art images, you implement pixel packing. Instead of using a full byte (8 bits) for each pixel, you group pixels together in a byte, allocating specific bits for each pixel's color value. For example, you could store two 4-bit pixel values in a single

byte. This technique reduces wasted storage and can lead to effective compression of pixel art images.

3.2.5 Huffman Coding:

Huffman Coding [14] is a variable-length compression algorithm that generates shorter bit codes for symbols that are more likely to appear in a data stream, resulting in efficient compression. The key features of Huffman Coding are:

- **Probability-based Coding:** The algorithm takes advantage of the probabilities of symbols occurring in the data. Symbols that are more probable to appear are assigned shorter bit codes, and less probable symbols are assigned longer bit codes.
- **Unique Prefix Attribute:** Huffman Coding ensures that no code is a prefix of another code, which allows for unambiguous decoding even when the codes are of varying lengths.

To illustrate Huffman Coding, let's consider the ASCII data string "BBAAABCBDEEBBEBBBAEBDAEDBEBEBBDAB" and compress it using this algorithm.

Frequency Calculation and Tree Construction:

First, calculate the frequency of each symbol in the data. In this example:

Table 1. Huffmann Coding – Symbol Frequency

Symbol	Frequency
A	6
B	15
C	1
D	4
E	7

Construct a Huffman tree by repeatedly merging the two lowest frequency symbols into a parent node with a frequency equal to the sum of their frequencies. This process is illustrated in Fig. 4.8.

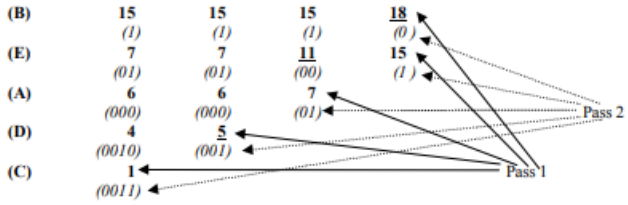

Figure 3. Huffmann Coding –Bit-Code Representation

Table 2. Huffmann Coding – Results

Symbol	Frequency	Bit-Code	Total bits
A	6	000	18
B	15	1	15
C	1	0011	4
D	4	0010	16
E	7	01	14
	33 (bytes)		67 (bits)

Generating Bit Codes:

Starting from the root of the Huffman tree, traverse the tree to assign bit codes to each symbol based on their position in the tree. As you move left, assign a '0', and as you move right, assign a '1'.

The resulting bit codes for each symbol are as follows:

- A: 00
- B: 1
- C: 010
- D: 011
- E: 001

Note that the shorter codes are assigned to more frequent symbols.

1. **Compression:**
   - Replace the symbols in the original data with their corresponding Huffman codes. The compressed data is a sequence of these variable-length codes.

In the provided example, the data string "BBAAABCBDEEBBEBBBAEBDAEDBEBEBBDAB" is compressed using Huffman Coding. The symbols with higher frequencies ('B' and 'A') are assigned shorter codes, while symbols with lower frequencies ('C', 'D', and 'E') are assigned longer codes. This way, the more common symbols are represented by fewer bits, resulting in overall compression.

Huffman Coding is widely used in various data compression formats due to its ability to efficiently represent frequent symbols with shorter codes, leading to effective compression while maintaining lossless data representation.

3.2.6 Arithmetic Coding:

Arithmetic Coding [15] is a compression algorithm that uses fractional values to represent a string of symbols. Unlike Huffman Coding, which assigns fixed-length codes to symbols, Arithmetic Coding assigns variable-length codes based on the probabilities of symbols occurring in the data. This allows for optimal storage of any string through the use of arithmetic operations.

Key features of Arithmetic Coding are:

- **Floating Point Representation:** A string is represented as a floating point number between 0 and 1. This number is uniquely decoded to reconstruct the original symbol stream.
- **Probability-based Encoding:** Probability values for each symbol in the string are calculated. Each symbol is then assigned a range within the interval [0, 1] based on its likelihood of occurrence.

To demonstrate Arithmetic Coding, let's consider encoding the phrase "BILL GATES" using the probability distribution shown in Fig. 4.10.

2. **Probability Calculation and Range Assignment:**
   - Calculate the probability of each symbol in the string: A, B, E, G, I, L, S, and T.
   - Assign a range within [0, 1] for each symbol based on its probability. The higher the probability, the larger the assigned range.
3. **Encoding the String:**

- Begin with a range of [0.0, 1.0].
- Encode each symbol by proportionally subdividing the range based on the assigned probability ranges.
- At each step, update the range to the subrange corresponding to the current symbol being encoded.

4. **Decoding the Encoded Value:**
   - To decode the encoded value, the decompressor examines the most significant digit of the value.
   - The decompressor uses the symbol probability distribution table to determine which symbol corresponds to the current range.
   - It narrows down the range based on the symbol's probability distribution and decodes each symbol accordingly.

In the provided example, the phrase "BILL GATES" is encoded using Arithmetic Coding with the probability distribution shown in Fig. 4.10. The process involves iteratively narrowing down the range based on the probabilities of the symbols. The resulting encoded value is 0.2572167752, which can be uniquely decoded to reconstruct the original string.

Arithmetic Coding achieves efficient compression by encoding symbols based on their probability distributions, enabling the representation of complex data with variable-length codes. It provides a powerful alternative to other compression algorithms like Huffman Coding, allowing for optimal storage of strings with varying probabilities.

3.3 Advanced Image Formats:

In this section, the author discusses three advanced methods for compressing and storing images: JPEG, MPEG, and Fractal. These methods are briefly introduced, as a comprehensive study of each one goes beyond the scope of the current project.

3.3.1 JPEG:

JPEG stands for Joint Photographic Experts Group, which is the organization that developed this image compression standard. JPEG is particularly designed for compressing and storing photographic images efficiently. It addresses the challenge of removing unnecessary information from an image while maintaining its visual quality.

Lossy Compression and Precision:

Lossy compression involves reducing the file size of an image by discarding some data, with the goal of achieving higher compression ratios. However, the challenge lies in deciding which information can be safely removed without significantly affecting the image's perceptual quality. Basic methods like reducing the precision of pixel values might not work well because their impact varies depending on the image content.

JPEG's Approach:

JPEG was developed to address this challenge by providing a way to achieve compression without sacrificing the perceived quality of the image. It offers a standardized approach to compressing images, specifically focused on photographic content. The JPEG format includes both lossless and lossy compression options, but the text focuses on the lossy component.

3.3.2 JPEG JFIF Format:

The JPEG JFIF (JPEG File Interchange Format) was created as a standard format for storing JPEG-compressed images. It provides a way to represent images in a compressed form while retaining visual fidelity.

Lossy Compression in JPEG:

JPEG's lossy compression algorithm is designed to intelligently remove information that is less perceptually significant. It analyzes the image data and employs techniques such as quantization and Discrete Cosine Transform (DCT) to reduce the amount of data while preserving the essential visual features.

Visual Quality Preservation:

JPEG's strength lies in its ability to achieve significant compression ratios while maintaining an image's visual quality that appears acceptable to the human eye. This makes it an excellent choice for storing and sharing photographic images on the web and in various applications.

3.3.3 JPEG JFIF Format:

The JPEG format employs the JPEG File Interchange Format (JFIF) for storing compressed images. This standard ensures compatibility across different devices and software.

Application and Example:

JPEG compression is highly effective for photographic images that contain complex textures and color gradients. For instance, a landscape photograph with the intricate details of trees, mountains, and skies can be compressed using JPEG. The resulting image maintains a visually pleasing appearance while having a significantly reduced file size, making it suitable for web sharing and storage.

5. Comparison of Factors Affecting Image Quality and Higher Level Considerations

Table 3. Summary of Factors Affecting Image Quality and Higher Level Considerations

Aspect	Image Format Factors	Higher Level Factors
Pixel Resolution	Supported resolutions vary; higher resolution with fewer colors is more discernible	Impact on image quality, speed, and storage
Colour Depth	Varied depths including 1-bit, palette-based, high-color, true-color, and more	Affects image realism, storage efficiency, and processing time
Compression	Lossless and lossy methods; choice depends on image content; Run-Length	Speed and processing overheads; impact on file size

	encoding example	
Colour Space	Various color models (e.g., RGB, HLS); RGB and HLS models are commonly used	Accuracy in color representation; compatibility with systems
Multiple Image Storage	Storing related images together in the same file to reduce redundancy	Efficient data storage and cataloguing
Speed	Format speed varies; uncompressed data is faster; processing time influenced by complexity	Real-time applications; decompression time; codec complexity
File Size	Uncompressed files are large; efficient compression decreases file size	Trade-off between speed and storage
System Resources	Format performance affected by system resources; floating-point math enhances quality	Format must be compatible with available system resources
Application	Suitability for the intended application is crucial; considerations for storage, compression, and speed	Format's capability to fulfill specific application needs

Above table is to provide a comprehensive comparison of various factors that affect image quality in the context of image formats. It outlines both the technical attributes related to image format factors and the higher-level considerations that impact the quality and suitability of image formats for different applications. The table aims to highlight the key aspects that contribute to the overall quality of an image format and help users understand how these factors play a role in choosing the right format for specific purposes. By presenting this information in a structured format, the table facilitates a clear understanding of the complexities involved in evaluating and selecting image formats.

4.1 Suggestions on Measuring Image Quality

For measuring image quality and evaluating the suitability of different image formats. The section introduces two approaches: exhaustive testing and quality rating.

A. Exhaustive Testing: The exhaustive testing approach involves systematically evaluating various image formats by subjecting them to a predefined set of test images that represent a wide range of application scenarios. The text suggests using a benchmark system to compare popular image formats based on factors like colour depth, resolution, compression and decompression speed, etc. This approach aims to provide a comprehensive overview of each format's performance across different scenarios.

To demonstrate this approach, two test images are introduced: a 4-bit test pattern (image (a)) and a 24-bit Kodak Photo-CD picture (image (b)). These images are saved in various formats such as BMP, GIF, JPEG (different quality levels), and PCX. The results of the tests, including file sizes, compression percentages, and loading/saving times, are presented in a table 4. The table highlights the strengths and weaknesses of each format in terms of compression efficiency and processing speed for these specific test images.

The exhaustive testing approach provides insights into how each format performs under different conditions. It helps identify which formats are better suited for specific types of images, providing users with valuable information to make informed decisions.

Table 4. File Sizes and Compression of Test Images

Image Format	**File Size (bytes)**	**Compression (%)**	**Load (secs)**	**Save (secs)**
a BMP	19,320	benchmark	-	-
BMP (+RLE)	760	3.93	-	-
GIF	947	4.90	0.14	0.16
JPEG (max.)	5,550	28.73	0.32	0.27
JPEG (low)	3,671	19.00	0.25	0.20
PCX	2,177	11.27	0.28	0.26
b BMP	4,285,232	benchmark	0.21	0.21
BMP (+RLE)	N/A *	N/A	N/A	N/A
GIF	768,934 ‡	17.94	0.14	0.16
JPEG (max.)	348,408	8.13	0.32	0.27
JPEG (low)	74,811	1.75	0.25	0.20
PCX	3,915,808	91.38	0.28	0.26

*Note: 24-bit RLE compression is not generally supported in BMP. † Quantized to 256-entry colour palette.

B. Quality Rating: The quality rating approach aims to develop a set of ratings that objectively assess the quality of an image format for a given test image. This method involves analyzing the relationships between various factors, such as colour depth, compression algorithms, and image characteristics. By quantitatively measuring how these factors impact the final image output, developers can assign ratings to formats based on their suitability for specific types of images.

For instance, the text discusses how certain formats are affected by specific image characteristics. GIF images, with limited colours represented by an internal palette, may perform well for certain types of images but not for others. JPEG compression, on the other hand, has a complex relationship between quality factors and image content, requiring analysis at the block level to assess its impact.

By assigning quality ratings to different formats based on these relationships, developers can evaluate which formats are well-suited for specific types of images. This

approach offers a more systematic and less subjective method for measuring image quality compared to exhaustive testing.

While the text presents the concepts and potential benefits of both approaches, it acknowledges that the details of implementing a rating system are complex and require further research and development. The idea of automating the analysis of formats using a software system is proposed as a potential solution to make the evaluation process more efficient and consistent. However, due to the complexity and scope of such a system, it's suggested that this could be a topic for future work.

5. Graphic File Handling

5.1 Graphic File Viewer:

A graphic file viewer is software designed to display and render various types of graphic files on a computer screen. It allows users to open and view images, graphics, and other visual content stored in different file formats. These viewers provide a visual representation of the graphics and often include features for zooming, panning, rotating, and navigating through the image. Graphic file viewers are commonly used for viewing photographs, illustrations, diagrams, icons, and other types of visual content.

Challenges in Graphic File Viewing:

1. **File Format Compatibility:** Different graphic file formats (JPEG, PNG, GIF, BMP, TIFF, etc.) have distinct structures and compression methods. Creating a viewer that can handle multiple file formats and render them accurately can be complex.
2. **Performance:** Rendering high-resolution images smoothly can be demanding on system resources. Ensuring smooth scrolling and zooming, especially for large images, can be a challenge.
3. **Color Management:** Displaying colors accurately across different devices and monitors can be a challenge due to variations in color profiles, settings, and calibration.
4. **Metadata Handling:** Images often contain metadata such as EXIF data (camera settings, timestamps), which needs to be parsed and displayed accurately.
5. **Interactivity:** Implementing interactive features like zooming, rotating, and panning while maintaining image quality and performance can be technically challenging.

5.2 Graphic File Conversion and Its Importance

Graphic File Converter:

A graphic file converter is software designed to transform graphic files from one format to another. It allows users to convert images from formats that might not be suitable for certain applications to formats that are more appropriate. For example, converting a high-resolution image to a compressed format for web use or converting a photograph to an icon format for use in software applications.

Challenges in Graphic File Conversion:

1. **Lossy vs. Lossless Conversion:** Some formats, like JPEG, use lossy compression, meaning they discard some image data to reduce file size. Converting from a lossy format to another format can lead to quality loss. Deciding between lossless and lossy conversion requires considering the trade-off between file size and image quality.
2. **Color Space and Bit Depth:** Different formats support varying color spaces (RGB, CMYK) and bit depths. Converting between formats with different color representations and bit depths can lead to color shifts and loss of detail.
3. **Transparency and Alpha Channels:** Handling transparency and alpha channel information during conversion can be challenging, especially when converting to formats that have different ways of representing transparency.
4. **Aspect Ratio and Dimensions:** Converting images while maintaining the correct aspect ratio and dimensions is important to prevent distortion.
5. **Batch Conversion:** Implementing the ability to convert multiple images in a batch can be complex, especially when handling variations in input files and desired output formats.

Both graphic file viewers and converters are essential tools for working with visual content, and their development involves addressing these technical challenges to ensure accurate rendering and conversion of images across various formats and devices.

6. Conclusion

In conclusion, this paper pursuit of evaluating graphics file formats and developing a comprehensive viewer and converter unveiled a range of challenges and achievements. While objectives spanning research, format analysis, and software development were met with varying degrees of success, the endeavor illuminated the complexities of the field. Challenges included obtaining industry insights on quality measurement, formulating generalized rating methods, and grappling with software integration issues like the elusive PCX format and display limitations. However, successful accomplishments encompassed an understanding of compression techniques, application of Borland Delphi and Windows API, and the implementation of clipboard functionality and online help. This multifaceted journey emphasized the intricate nature of graphics handling, fostering a deeper understanding for future endeavors in this domain.

Potential future work involves broadening the graphics library to encompass additional formats and incorporating advanced image processing capabilities.

References

[1] Fan, Zhigang, and Ricardo L. De Queiroz. "Identification of bitmap compression history: JPEG detection and quantizer estimation." *IEEE Transactions on Image Processing* 12.2 (2003): 230-235.

[2] Faircloth, B. C. (2006). GMCONVERT: file conversion for GENEMAPPER output files. Molecular Ecology Notes, 6(4), 968-970.

[3] Chandler, D. M. (2013). Seven challenges in image quality assessment: past, present, and future research. *International Scholarly Research Notices*, *2013*.

[4] Dathan, C. S. (2017). *Land evaluation and land use planning in Eruthavoor watershed of Western Ghat region using GIS and remote sensing* (Doctoral dissertation, Department of Soil Science and Agricultural Chemistry, College of Agriculture, Vellayani).

[5] Basha, S. S., & Prasad, K. S. (2009). Segmentation of Medical Images Using Morphological Image Processing. *i-Manager's Journal on Future Engineering and Technology*, *4*(3), 37.

[6] Wiggins, R. H., Davidson, H. C., Harnsberger, H. R., Lauman, J. R., & Goede, P. A. (2001). Image file formats: past, present, and future. *Radiographics*, *21*(3), 789-798

[7] Karima, M., Sadhal, K., & McNeil, T. (1985). From paper drawings to computer-aided design. *IEEE computer graphics and applications*, *5*(02), 27-39.

[8] Sharifinejad, A., & Mehrpour, H. (2001, October). A novel bitmap based matching criterion for MPEG video coding. In *Proceedings. Ninth IEEE International Conference on Networks, ICON 2001.* (pp. 20-24). IEEE.

[9] Lien, C. Y., Teng, H. C., Chen, D. J., Chu, W. C., & Hsiao, C. H. (2009). A Web-based solution for viewing large-sized microscopic images. Journal of digital imaging, 22, 275-285.

[10] Salomon, D. (2007). A concise introduction to data compression. Springer Science & Business Media.

[11] Storer, J. A. (1987). *Data compression: methods and theory*. Computer Science Press, Inc..

[12] Golomb, S. (1966). Run-length encodings (corresp.). *IEEE transactions on information theory*, *12*(3), 399-401.

[13] Kinsner, W., & Greenfield, R. H. (1991, May). The Lempel-Ziv-Welch (LZW) data compression algorithm for packet radio. In *[Proceedings] WESCANEX'91* (pp. 225-229). IEEE.

[14] Moffat, A. (2019). Huffman coding. *ACM Computing Surveys (CSUR)*, *52*(4), 1-35.

[15] Howard, P. G., & Vitter, J. S. (1994). Arithmetic coding for data compression. *Proceedings of the IEEE*, *82*(6), 857-865.