KOLEJ UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY

Assignment

BMCS3003 DISTRIBUTED SYSTEMS AND PARALLEL COMPUTING

2021/2022

Student's name/ ID Number	:	Ryan Kho Yuen Thian (2204097)
Student's name/ ID Number	:	Ong Weng Kai (2203309)
Student's name/ ID Number	:	Yong Zee Lin (2203770)
Programme	:	Bachelor of Computer Science in Data Science
Tutorial Group	:	2
Date of Submission to Tutor	:	17/9/2024

Near Lossless Image Compression using Discrete Cosine Transform (DCT)

CHAPTER 1: INTRODUCTION

In today's digital landscape, the importance of image storage and transmission spans many applications (Alzahrani & Albinali 2021), from social media and digital libraries to medical imaging and scientific documentation. As high-resolution imaging technology becomes more widespread, file sizes increase exponentially, demanding more storage and bandwidth. Various image compression techniques are employed to manage these challenges, categorised into lossy and lossless compression (GnosisX 2023). Lossy compression achieves higher rates by discarding some data, which may not be suitable for all applications. In contrast, lossless compression preserves all original data, allowing for exact reconstruction, critical for precision applications (GnosisX 2023). Lossless image compression is vital in applications where image integrity and accuracy are paramount. This includes medical imaging, where accurate depiction of images influences diagnostics and treatment; digital art and archival, where preserving original artwork without detail loss is essential; and scientific research, where detailed visual data must be accurately recorded and analysed (Ungureanu et al. 2024).

Despite its importance, lossless image compression faces significant challenges due to its computational complexity. The key task is to eliminate redundancy in image data without losing information (Bothra 2024). This requires sophisticated algorithms to detect and encode redundancies for full restoration upon decompression. Techniques include Huffman coding, arithmetic coding and the Lempel-Ziv-Welch (LZW) algorithm. These methods involve complex, computationally expensive operations, especially with large high-resolution images. The challenge is further compounded by the need for real-time processing in many applications. For example, in medical imaging systems like MRI and CT scans, images must be quickly processed and compressed to facilitate timely diagnostics and treatments (Alkawaz, Mydin & Johar 2022). Scientific research often involves collecting and analysing large volumes of image data that need efficient compression and storage. The demand for higher resolutions and expanding datasets has made the computational costs of lossless compression a critical bottleneck, affecting system speed, efficiency, and increasing storage and transmission costs. Therefore, developing advanced compression algorithms to manage growing image data complexity while reducing computational requirements is essential. Enhancements in this area could significantly improve data handling speed and efficiency, contributing to technological and scientific advancements.

To summarise, lossless image compression is vital for maintaining data integrity in critical fields, but the high computational cost of these algorithms presents a challenge as image resolutions and datasets continue to grow. Addressing these challenges is essential for the advancement of digital imaging applications, making it a key area of focus in both research and industry. To tackle this issue, we propose the use of parallel computing, which can significantly accelerate the process of near lossless image compression and decompression, especially given its computationally intensive nature. We will utilise three parallel platforms: OpenMP, CUDA and MPI. Our work mainly focuses on parallelizing a Transform algorithm, specifically DCT and IDCT. Finally, we will compare the execution times in both serial and parallel approaches, measure the performance gains and assess the quality of the images.

Research Question: How can parallel computing techniques be effectively applied to accelerate near-lossless image compression and decompression processes? If applied effectively, what would be the gain in speed?

CHAPTER 2: LITERATURE REVIEW

2.1 Lossless Image Compression

Lossless compression reduces file sizes without sacrificing information, maintaining the original quality of images. It employs algorithms that rewrite files efficiently, allowing the original to be restored without quality loss, making it ideal for applications where data integrity is crucial, such as text documents, software, and certain media files (Adobe 2022; Larmier 2023). This process involves indexing non-essential information and compressing the data accordingly. The advantages of lossless compression include maintaining high image quality, making it ideal for digital portfolios and client deliveries, and allowing images to be restored to their original form for further editing or printing. It also improves website performance by speeding up loading times and file transfers. However, it doesn't reduce file sizes as much as lossy compression, which can be limiting with large volumes of images. Lossless images tend to be larger than their lossy counterparts, potentially affecting web page responsiveness (see Figure 2.1). Common file types using lossless compression include PNG, BMP, RAW, GIF, ZIP, and WAV, each serving purposes from web graphics to high-quality audio storage (Adobe 2022).

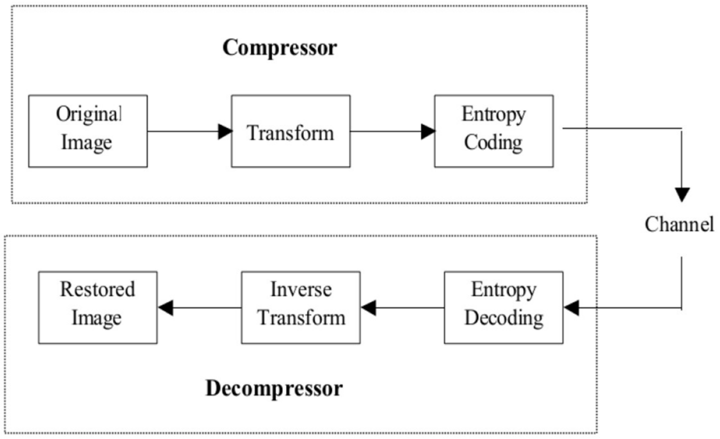


Figure 2.1 Comparison of Lossy and Lossless Compressions

Figure 2.2 Lossless Compression Flow

Lossless compression algorithms typically involve two steps (see Figure 2.2). First, the original image is converted into a format that reduces inter-pixel redundancy. Second, an entropy encoder eliminates coding redundancy. The lossless decompression process is the exact reverse of this (Gupta & Nagar 2020).

2.2 Discrete Cosine Transform (DCT)

2.2.1 Introduction

DCT is a mathematical operation similar to the Fast Fourier Transform (FFT), used to transform a signal from one representation to another. For instance, an image is a 2D signal interpreted by the human visual system. DCT converts this spatial information into numerical "frequency" or "spectral" data, enabling easier manipulation for compression purposes (Stanford University n.d.). This transform is widely used in image processing applications like image compression, noise reduction and watermarking, particularly as the standard for JPEG compression. DCT, an orthogonal transform with a fixed set of basis functions, is preferred over other transforms like SVD, DWT and Hadamard due to its efficient energy compaction and low computational complexity (Aravindh & Sakthivel 2020; Raid et al. 2014). It concentrates energy in lower frequencies and reduces the blocking artefact effect, where boundaries between sub-images become noticeable. In DCT, the signal is analysed by separating it into individual frequency components, such as low, mid and high-frequency components (Aravindh, S & Sakthivel 2020). Type-II DCT is the most commonly used variant, often referred to simply as "the DCT." Its inverse, known as Type-III DCT, is called "the inverse DCT" or "the IDCT" (Kaushik & Nain 2014). The general equation for a 2D DCT for an N by M image is shown in Figure 2.3.

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

$$\alpha(u) = \begin{cases}\sqrt{\frac{1}{N}} & \text{if } u = 0\\ \sqrt{\frac{2}{N}} & \text{if } u \neq 0\end{cases}$$

$$\alpha(v) = \begin{cases}\sqrt{\frac{1}{M}} & \text{if } v = 0\\ \sqrt{\frac{2}{M}} & \text{if } v \neq 0\end{cases}$$

```
for (i = 0; i < N; i++) {
  for (j = 0; j < N; j++) {
    temp = 0.0;
    for (x = 0; x < N; x++) {
      for (y = 0; y < N; y++) {
        temp += Cosines[x][i] *
          Cosines[y][j] *
          Pixel[x][y];
      }
    }
    temp *= sqrt(2 * N) * Coefficient[i][j];
    DCT[i][j] = INT_ROUND(temp);
  }
}
```

Figure 2.3: Equation for 2D DCT

Figure 2.4: Algorithm to Calculate DCT Matrix

In Figure 2.3, the input image has dimensions N by M, with pixel intensity at row i and column j denoted as F(i,j), and DCT coefficients at row u and column v represented as F(u,v). Scaling factors are α(u) and α(v). Most image signal energy is concentrated at low frequencies (upper left of the DCT matrix), allowing compression by neglecting small values in the lower right (higher frequencies) with minimal visible distortion (Kaushik & Nain 2014). If DCT is applied to an N x N square matrix of pixel values, which would produce an N x N matrix of frequency coefficients, N is typically set to 8 as larger blocks would require significantly more computational time although it would provide better compression

(Stanford University n.d.). The algorithm for the DCT matrix calculation is shown in Figure 2.4, with the IDCT equation in Figure 2.5.

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

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{if } u = 0 \\ \sqrt{\frac{2}{N}} & \text{if } u \neq 0 \end{cases}$$

$$\alpha(v) = \begin{cases} \sqrt{\frac{1}{M}} & \text{if } v = 0 \\ \sqrt{\frac{2}{M}} & \text{if } v \neq 0 \end{cases}$$

Figure 2.5: Equation for IDCT

2.2.2 Related Works (DCT)

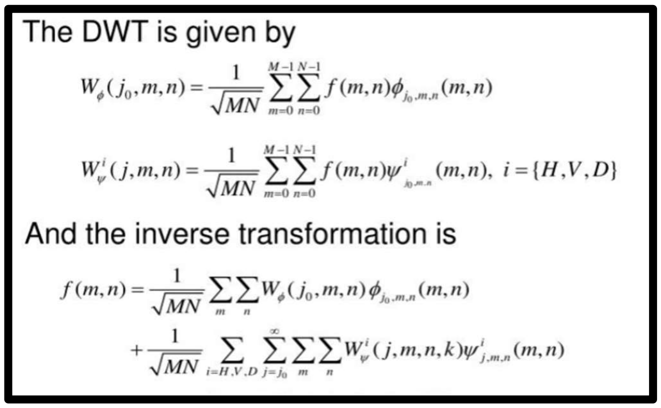
Interest in lossless DCT has grown, with Hasan & Mia (2017) proposing a method where images are divided into 8x8 or 16x16 blocks, and 2D DCT is applied to each block. The DCT coefficients are quantized, encoded and transmitted. At the receiver's end, these coefficients are decoded, and the inverse 2D DCT is applied to reconstruct the image. This method leverages the fact that many DCT coefficients are close to zero in typical images, allowing for discarding them with minimal quality loss (Hasan & Mia, 2017). Dingra et al. (2018) also introduced a similar approach where the image is segmented into 8×8-pixel blocks. DCT is applied to each block sequentially, from left to right and top to bottom. Each block undergoes compression via quantization, and the resulting compressed blocks are stored in a more compact form. To reconstruct the image, the compressed blocks are decompressed using IDCT. Lossless DCT image compression is crucial in healthcare to prevent data loss that could lead to incorrect diagnoses and treatment complications. Joshi & Mishra (2019) proposed using an improvised DCT (DCT & IDCT) for medical image lossless compression, effectively reducing both compression size and time and enhancing the compression ratio, making it applicable to various types of medical images. This approach shows promise for future application in telemedicine and represents a promising area for significant advancements in medical image compression. Singh et al. (2014) also agreed with this by applying DCT and LZW lossless techniques for image compression in biomedical imaging, resulting in a compressed image without any information lost.

2.3 Discrete Wavelet Transform (DWT)

2.3.1 Introduction

It is used in image compression due to its ability to analyse and modify image data at various resolution levels into a series of wavelet coefficients, representing the image across various frequency bands with varying degrees of detail. These coefficients are separated into high-frequency components, capturing detailed aspects (edges and textures), and low-frequency components that represent the smoother, broader areas of the image. Its power lies in its hierarchical approach to compression, where each decomposition level provides a finer granularity of the image's frequency components. This allows significant flexibility in how compression is applied, enabling higher compression ratios without critical losses in visual quality. By retaining more of the low-frequency components and selectively compressing the high-frequency parts, DWT minimises data size and preserves important visual elements of the image (Starosolski 2020).

Moreover, DWT is effective in handling various types of image redundancies, including spatial, temporal and pixel redundancies. It reduces spatial redundancy by compressing similar patterns or colours, addresses temporal redundancy in video compression by reducing data across similar frames, and manages pixel redundancy by encoding image variations more efficiently. DWT's multi-resolution nature allows for a more nuanced approach to compression, which can adaptively compress different parts of an image according to their specific details and importance. In practical applications, DWT has proven to be particularly advantageous for compressing high-resolution images and multispectral data from satellite imagery, where maintaining the integrity and quality of the data is paramount. The adaptability of DWT to various data types and its effectiveness in reducing file sizes while maintaining quality make it a preferred choice for modern image compression needs (Starosolski 2020). The equation for DWT is shown below in Figure 2.6.



The DWT is given by

$$W_\phi(j_0,m,n)=\frac{1}{\sqrt{MN}}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}f(m,n)\phi_{j_0,m,n}(m,n)$$

$$W_\psi^i(j,m,n)=\frac{1}{\sqrt{MN}}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}f(m,n)\psi'^{i}_{j_0,m,n}(m,n),\ i=\{H,V,D\}$$

And the inverse transformation is

$$f(m,n)=\frac{1}{\sqrt{MN}}\sum_m\sum_n W_\phi(j_0,m,n)\phi_{j_0,m,n}(m,n)$$
$$+\frac{1}{\sqrt{MN}}\sum_{i=H,V,D}\sum_{j=j_0}^{\infty}\sum_m\sum_n W_\psi^i(j,m,n,k)\psi'^{i}_{j,m,n}(m,n)$$

Figure 2.6 DWT and inverse transformation formulas

2.3.2 Related Works (DWT)

Abdulazeez et al. (2020) thoroughly examined wavelet transform (WT) techniques, focusing on their application in image processing. Wavelet transforms, particularly the Discrete Wavelet Transform (DWT), are valuable for decomposing images into different frequency components, capturing various details and features. This makes DWT crucial for reducing data size while preserving image quality, ideal for compression and transmission. The paper demonstrated DWT's effectiveness in image compression by describing that the process involves dividing an image into multiple frequency bands, each capturing distinct aspects of the image. These bands are then processed through various levels of filtering, which isolates and encodes the essential features while minimising redundancy. This approach enhances image compression efficiency, including in schemes like JPEG 2000, making it crucial for modern data management and transmission (Abdulazeez et al. 2020).

Yadav, Gangwar & Singh (2012) highlighted the efficacy of wavelet transform-based techniques, particularly DWT, in addressing the growing demands for storage and communication bandwidth. DWT reduces data redundancy by decomposing images into wavelet coefficients across multiple scales and applying iterative low-pass and high-pass filtering. This process generates four sub-images representing different frequency details, which are then processed through thresholding, where coefficients below a set threshold are discarded to reduce image size while retaining essential information. The effectiveness of DWT is assessed using Peak Signal-to-Noise Ratio (PSNR), indicating the quality of the reconstructed image. The implementation aims to balance high compression ratios with minimal image quality loss, making DWT a robust method for efficient, scalable image compression. It overcomes limitations of previous standards like JPEG and adapts to the evolving needs of mobile multimedia and internet communications, offering a flexible solution for modern data management challenges.

2.4 Haar Wavelet Transform (HWT)

2.4.1 Introduction

Haar Wavelet Transformation (HWT) is constructed to transform a list of numbers into an approximation that can be easily transmitted and reconstructed. The transformation can be summarised by taking pairs of numbers. HWT can also compute their averages and differences and send these transformed values. This allows the data trends to be captured with large differences indicating significant changes and small differences indicating minor changes. HWT can be represented using a matrix formulation, which can allow the transformation to be applied to vectors of even length. The transformation matrix computes both averages and differences. Its inverse can be derived by doubling the transpose of the matrix. HWT is defined using an orthogonal matrix and the inverse is its transpose. It also involves lowpass and highpass filters, which can help in analysing the data by averaging similar values and highlighting differences. For digital grayscale images, the HWT can be applied to the image matrix to transform it. This involves multiplying the image matrix by the HWT matrix and its transpose (Khan et al. 2019; Why Do Math? 2011).

The 2D HWT compresses the image data and conserves most of the energy in one corner of the transformed matrix. This will let it in many near-zero values elsewhere and make it suitable for image compression. When HWT is applied, it can further improve compression. By repeatedly transforming the blurred part of the image, more energy is conserved. After that, the compression becomes more efficient. The iterative HWT process maintains invertibility and allows the original image to be reconstructed from the compressed form. In conclusion, the HWT has limitations such as short filter lengths and converting integers to irrational numbers, which are addressed by more sophisticated wavelet filters used in practical applications (Khan et al. 2019; Why Do Math? 2011). The HWT equation is shown in Figure 2.7.

$$\boxed{\psi(t) = \begin{cases} 1 & t \in [0, 1/2) \\ -1 & t \in [1/2, 1) \\ 0 & t \notin [0, 1) \end{cases}}$$

$$\boxed{\psi_i^j(t) = \sqrt{2^j}\psi(2^j t - i),\ j = 0, 1, \ldots \text{ and } i = 0, 1, \ldots, 2^j - 1 .}$$

Figure 2.7 HWT Equation

2.4.2 Related Works (HWT)

Khan et al. (2019) have demonstrated HWT's effectiveness in compressing images with minimal quality loss. The research aimed to identify the transformer algorithm that achieves high compression rates while preserving image quality. The experiments demonstrated that the optimised HWT achieved 97.8% accuracy, significantly speeding up image compression and decompression while maintaining image quality. It also outperformed DCT and RLE in terms of higher Peak Signal to Noise Ratio (PSNR) and better Compression Ratio (CR) (Khan et al. 2019). This makes HWT valuable for high-quality image preservation in fields like medical imaging and multimedia storage.

Numerous studies have explored image compression using the HWT for focusing on its computational simplicity and efficiency. However, based on the project results and various comparative studies by Xiao (2001), they indicate that HWT's discontinuous nature often results in poorer compression performance compared to more advanced wavelets like Daubechies or biorthogonal wavelets. Although the HWT is fast, it struggles with smooth gradients and leads to less effective compression ratios.

2.5 Algorithm Chosen

i) Comparisons between works using DCT & DWT

Research by Dingra et al. (2018) shows that DCT is more effective and reliable than DWT for lossless image compression, allowing for better size reduction and image reconstruction. Besides achieving better compression, DCT allows for the original image to be reconstructed whenever necessary, which is not possible with the DWT technique (Dingra et al., 2018). Moreover, according to Joshua, Arrivukannamma & Sathiaseelan (2016), who conducted research comparing the results of new lossy and lossless image compression methods using transform coding techniques, specifically DCT and DWT, DCT achieves a higher compression ratio while avoiding blocking artifacts and allows for effective localization in both spatial and frequency domains. Furthermore, with reference to MSE and PSNR values, DCT outperforms DWT with high compression ratio and large coefficients (Joshua, Arrivukannamma & Sathiaseelan 2016).

ii) Comparisons between works using DCT & HWT

A study on lossless image compression of medical images indicates that DCT is superior to HWT in compressing PNG and TIF images, including CT-grey and MRI-colour images, with better results in terms of compressed image size and compression ratio. However, for JPG format images, such as those from gastrointestinal endoscopy, DCT performs better with grayscale images, while HWT excels with colour images, raising an open question for further research. Despite this, HWT consistently outperforms DCT in compression time across all image types and formats (Alzahrani & Albinali 2021)

We will use the DCT algorithm. DCT is simpler and easier to understand than HWT and DWT. Its straightforward matrix operations, efficient algorithms, and intuitive use of cosine functions make it less resource-intensive and easier to implement and conceptualise.

CHAPTER 3: METHODOLOGY

3.1 Serial Code for DCT & IDCT

The C++ serial implementations of DCT & IDCT are shown in Figures 3.1 and 3.2. DCT will be applied to small blocks (e.g., 8x8 pixels) of the image and each block will be transformed from spatial to frequency domain, where it might be easier to identify and encode patterns. IDCT will be used to transform the compressed data back into the spatial domain. The codes for DCT & IDCT are adapted from the formulae in Literature Review and the Python code written by Sherpa (2018) in Kaggle.

```
void dct2(const Mat& src, Mat& dst) {
    int M = src.rows;
    int N = src.cols;
    dst.create(M, N, CV_64F);

    for (int u = 0; u < M; ++u) {
        for (int v = 0; v < N; ++v) {
            double sum = 0.0;
            for (int x = 0; x < M; ++x) {
                for (int y = 0; y < N; ++y) {
                    double cosX = cos(M_PI / M * (x + 0.5) * u);
                    double cosY = cos(M_PI / N * (y + 0.5) * v);
                    sum += src.at<double>(x, y) * cosX * cosY;
                }
            }
            double C_u = (u == 0) ? sqrt(1.0 / M) : sqrt(2.0 / M);
            double C_v = (v == 0) ? sqrt(1.0 / N) : sqrt(2.0 / N);
            dst.at<double>(u, v) = C_u * C_v * sum;
        }
    }
}
```

Figure 3.1: Serial Code for 2D DCT

```
void idct2(const Mat& src, Mat& dst) {
    int M = src.rows;
    int N = src.cols;
    dst.create(M, N, CV_64F);

    for (int x = 0; x < M; ++x) {
        for (int y = 0; y < N; ++y) {
            double sum = 0.0;
            for (int u = 0; u < M; ++u) {
                for (int v = 0; v < N; ++v) {
                    double C_u = (u == 0) ? sqrt(1.0 / M) : sqrt(2.0 / M);
                    double C_v = (v == 0) ? sqrt(1.0 / N) : sqrt(2.0 / N);
                    double cosX = cos(M_PI / M * (x + 0.5) * u);
                    double cosY = cos(M_PI / N * (y + 0.5) * v);
                    sum += C_u * C_v * src.at<double>(u, v) * cosX * cosY;
                }
            }
            dst.at<double>(x, y) = sum;
        }
    }
}
```

Figure 3.2: Serial Code for 2D IDCT

3.2 Parallel Computing Techniques

i) OpenMP Platform

- **Parallelize the Loops & Using Work Sharing Constructs:** Since the calculation of each element in the **dst** matrix (i.e., each DCT coefficient in the **dct2** function and each pixel value in the **idct2** function) is independent of others, the most straightforward approach is to parallelize the loops. This involves the **#pragma omp parallel for** directive to distribute the iterations of these loops across available threads. This makes use of OpenMP's work-sharing constructs (the **for** directive), allowing the workload of computing the DCT coefficients across different **u** and **v** indices to be shared among the threads.
- **Reduction of Shared Data:** In both DCT and IDCT functions, the computation of **sum** involves accumulating values in a manner that could lead to race conditions if not handled properly. To manage this, OpenMP provides a **reduction** clause that can be used with the parallel for directive to perform thread-safe reductions. This ensures that each thread has a private copy of the sum variable, and all individual sums are combined at the end of the parallel region.
- **Precomputing/Hardcoding Cosine Values:** Precomputing/hardcoding the cosine values speeds up the DCT and IDCT functions by eliminating the need for repetitive trigonometric calculations during runtime. This allows OpenMP threads to focus on parallelizing matrix operations rather than recalculating values, resulting in improved performance.

ii) CUDA Platform

- **Parallelizing Computations:** The DCT and IDCT involve nested loops where each output element is computed independently. These computations can be performed in parallel. A dct/idct kernel function will be defined that processes 8x8 blocks of image data using 64 threads per block, where each thread is responsible for one element of the block. It will leverage shared memory to reduce global memory accesses, perform the DCT/IDCT computation with precomputed cosine values, and ensure data consistency with __syncthreads(). This efficient parallel approach accelerates DCT/IDCT calculations on the GPU by utilising block-based processing and effective memory usage.
- **Precomputing/Hardcoding and Transferring Cosine Values:** To enhance performance, cosine values will be hardcoded into the program and then transferred to the GPU constant memory. Constant memory is read-only memory space that is cached and optimised for read operations. It's ideal for storing data that does not change during kernel execution, such as precomputed cosine values used in DCT and IDCT calculations.

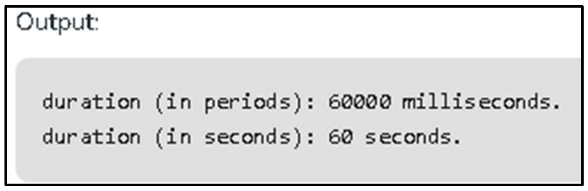
iii) MPI Platform

- **Distributing Portions of the Image:** An image will be divided equally among the processes available so that each process performs operations on its own image portion. Image division will be based on the number of rows of the image. **MPI_Scatter** will be used to distribute portions of the image among the available processes. **MPI_Gather** will be used to gather the results of the processes back to the root process to form the final image.
- **Data Broadcast:** The **MPI_Bcast** function will be employed to broadcast the necessary details to all MPI processes. This ensures that all processes have the necessary data to perform their computations.
- **Processing:** Since the image read will be a coloured image, each MPI process has to deal with all 3 channels (red, green and blue) when performing operations, such as DCT and IDCT.
- **Precomputing/Hardcoding Cosine Values:** By precomputing and hardcoding the cosine values, it eliminates the need to compute these values at runtime. The DCT and IDCT involve a series of cosine operations, which can be computationally expensive.

3.3 Performance Evaluation & Other Evaluation Methods

The following is a list of ways that we will use to compare the original images with the compressed ones.

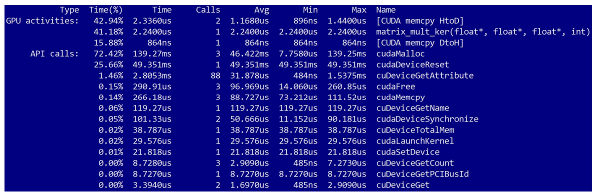
- **Measure the time taken to compress and decompress the image for serial and parallel methods, and then evaluate the speedup gained from using parallel methods:** We will use a special timer from the **<chrono>** library in C++. Figure 3.3 shows an example. To compute the speedup gained, the time taken for the serial method will be divided by the time taken for each parallel method. On CUDA, we can use `**nvprof**`, an NVIDIA command-line profiler tool for collecting and analysing performance data from CUDA applications, helping identify and optimise performance bottlenecks on NVIDIA GPUs. (Figure 3.4).



```
Output:

duration (in periods): 60000 milliseconds.
duration (in seconds): 60 seconds.
```

Figure 3.3: The example output to show the duration



```
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   42.94%  2.3360us         2  1.1680us     896ns  1.4400us  [CUDA memcpy HtoD]
                   41.18%  2.2400us         1  2.2400us  2.2400us  2.2400us  matrix_mult_ker(float*, float*, float*, int)
                   15.88%     864ns         1     864ns     864ns     864ns  [CUDA memcpy DtoH]
      API calls:   72.42%  139.27ms         3  46.422ms  7.7580us  139.25ms  cudaMalloc
                   25.66%  49.351ms         1  49.351ms  49.351ms  49.351ms  cudaDeviceReset
                    1.46%  2.8053ms        88  31.878us     484ns  1.5375ms  cuDeviceGetAttribute
                    0.15%  290.91us         3  96.969us  14.060us  260.85us  cudaFree
                    0.14%  266.18us         3  88.727us  73.212us  111.52us  cudaMemcpy
                    0.06%  119.27us         1  119.27us  119.27us  119.27us  cuDeviceGetName
                    0.05%  101.33us         2  50.666us  11.152us  90.181us  cudaDeviceSynchronize
                    0.02%  38.787us         1  38.787us  38.787us  38.787us  cuDeviceTotalMem
                    0.02%  29.576us         1  29.576us  29.576us  29.576us  cudaLaunchKernel
                    0.01%  21.818us         1  21.818us  21.818us  21.818us  cudaSetDevice
                    0.00%  8.7280us         3  2.9090us     485ns  7.2730us  cuDeviceGetCount
                    0.00%  8.7270us         1  8.7270us  8.7270us  8.7270us  cuDeviceGetPCIBusId
                    0.00%  3.3940us         2  1.6970us     485ns  2.9090us  cuDeviceGet
```

Figure 3.4: Sample output of nvprof

- **Comparing image similarity or quality:** We use Mean Squared Error (MSE) and Structural Similarity Index (SSIM) to assess image similarity. Lower MSE values indicate greater similarity between images, while higher SSIM values (close to 1) also indicate greater similarity. These methods are used in addition to visually comparing the images.

3.4 System Design

The proposed system design is shown below in Figure 3.5. The ideas were adapted from the lossless image compression using DCT works by Hasan & Mia (2017) and Mandyam et al. (1995). Mandyam et al. (1995) stated that to produce any compression, DCT coefficients must be quantized, as storing them at full precision doesn't achieve compression. His proposal was to evaluate the DCT matrix to only B digits past the decimal point. RLE is initially used to remove consecutive repeated zeros in the latent space. Afterward, the Lempel-Ziv-Markov chain algorithm (LZMA), a powerful lossless compression method, is applied to the RLE output to further reduce the file size prior to transmission.

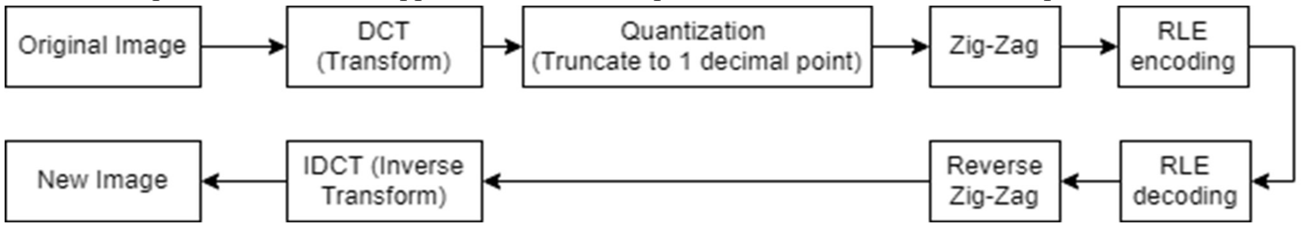


Figure 3.5: Proposed System Design

For the dataset, we will use a collection of high-resolution and large coloured images to test the effects of using the proposed parallel computing platforms and techniques. The whole process illustrated in Figure 3.5 will be timed for each approach. The duration will cover from the moment an image is read into the program to the moment the decompressed image is written. The parallel computing techniques mentioned earlier will be applied to Figure 3.5.

References (TAR UMT Harvard Referencing Style)

Abdulazeez, A., Zeebaree, D., Zebari, D., Zebari, G. & Adeen, I. 2020, 'The applications of Discrete Wavelet Transform in image Processing: a review,' *Journal of Computing and Data Mining*, vol. 1, no. 2, viewed 3 August 2024, <https://publisher.uthm.edu.my/ojs/index.php/jscdm/article/download/7215/3935/30309>.

Adobe 2022, '*Lossless Compression: A Complete Guide | Adobe*', viewed 30 July 2024, <https://www.adobe.com/uk/creativecloud/photography/discover/lossless-compression.html>.

Alkawaz, M., Mydin, M. & Johar, M. 'X-Ray, MRI and CT scan Medical Image Compression using Huffman Coding', in *2022 IEEE Symposium on Industrial Electronics & Applications (ISIEA)*, 16-17 July 2022, IEEE Xplore, Langkawi Island, Malaysia, 2022, pp. 1-6.

Alzahrani, M. & Albinali, M. 2021, 'Comparative Analysis of Lossless Image Compression Algorithms based on Different Types of Medical Images', in *2021 International Conference of Women in Data Science at Taif University (WiDSTaif)*, 30-31 March 2021, IEEE Xplore, Taif, Saudi Arabia, pp. 1-5.

Aravindh, S. & Sakthivel S, 'Implementation of Two-Dimensional (2D) Discrete Cosine Transform (DCT) using Reversible Gates', in *Journal of Physics: Conference Series Volume 1716* , 1 December 2020, IOP Publishing Ltd, Vellore Institute of Technology, Chennai, India, pp. 1-1.

Bothra, K. 2024, *Lossless Compression: Advantages, disadvantages, & Types,* viewed 30 July 2024, <https://seahawkmedia.com/design-glossary/lossless-compression-advantages-disadvantages-and-types/>.

Dingra, E., Kour, E., Dar, H. & Rashid, H. 2018, 'Software for Lossless Compression Using DCT and Wavelet-Based Image Compression Technique Using MATLAB', *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 7, no. 6, viewed 29 July 2024, <https://www.ijirset.com/upload/2018/june/80_11_Software.pdf>.

GnosisX. 2023, *Lossy vs. Lossless Compression: A Detailed Look*, viewed 30 July 2024, <https://medium.com/@contact_45426/lossy-vs-lossless-compression-a-detailed-look-7284db2f9a93>.

Gupta, S. & Nagar, D. 2020, 'Image Compression: A Review', *International Journal of Advanced Research in Science, Communication and Technology*, vol. 9, no. 5, viewed 30 July 2024, <https://ijarsct.co.in/Paper462.pdf>.

Hasan, A. & Mia, M. 2017, ‘Lossless Image Compression using Discrete Cosine Transform (DCT)’, *Department of Computer Science and Engineering, Jagannath University*, viewed 29 July 2024, <https://www.researchgate.net/publication/323771460_Lossless_Image_Compression_using_Discrete_Cosine_Transform_DCT>.

Joshi, H. & Mishra, P. 2019, ‘Medical Image Lossless Compression Using Improvised DCT’, *International Journal of Computer Sciences and Engineering*, vol. 7, no. 4, viewed 29 July 2024, <https://doi.org/10.26438/ijcse/v7i4.682685>.

Joshua, T., Arrivukannamma, M. & Sathiaseelan, J. 2016, ‘Comparison of DCT and DWT Image Compression’, *International Journal of Computer Science and Mobile Computing*, vol. 5, no. 4, viewed 29 July 2024, <https://www.ijcsmc.com/docs/papers/April2016/V5I4201635.pdf>.

Kaushik, E & Nain, E 2014, ‘Image Compression Algorithms Using Dct’, *Journal of Engineering Research and Applications*, vol. 4, no. 4, viewed 21 July 2024, <https://www.ijera.com/papers/Vol4_issue4/Version%201/BG044357364.pdf>.

Khan, S., Nazir, S., Hussain, A., Ali A. & Ullah A. 2019, ‘An efficient JPEG image compression based on Haar wavelet transform, discrete cosine transform, and run length encoding techniques for advanced manufacturing processes’, *Measurement and Control*, vol. 52, no. 9-10, viewed 2 August 2024, <https://journals.sagepub.com/doi/10.1177/0020294019877508>.

Larmier, M. 2023, *Lossy vs Lossless Image Compression: What’s the Difference?*, viewed 30 July 2024, <https://imagify.io/blog/lossless-vs-lossy-image-compression/>.

Mandyam, G., Ahmed, N. & Magotra, N. 1995, ‘DCT-based scheme for lossless image compression’, *Society of Photo-Optical Instrumentation Engineers (SPIE)*, vol. 2419, <https://doi.org/10.1117/12.206386>.

Raid, A., Khedr, W., El-dosuky, M. & Ahmed, W. 2014, ‘JPEG image compression using discrete cosine transform - A survey’, *International Journal of Computer Science & Engineering Survey*, vol. 5, no. 2, viewed 21 July 2024. <https://arxiv.org/pdf/1405.6147>.

Sethi, N., Krishna, R., & Arora, R. n.d., ‘Image Compression Using Haar Wavelet Transform’, *Computer Engineering and Intelligent Systems*, viewed 10 August 2024, <https://core.ac.uk/download/pdf/234644233.pdf>.

Sherpa, 2018, *2D DCT2&3 Implementation*, ver. 2, computer programme, viewed 3 August 2024, <https://www.kaggle.com/code/thesherpafromalabama/2d-dct2-3-implementation/notebook>.

Singh, S., Agrawal, A., Tripathi, M. & Vaish, S. 2014, ‘Image Compression on Biomedical imaging using DCT and LZW Lossless Approach’, in *International Conference on Advances in Engineering and Technology*, 2014, Institute of Research Engineers and Doctors, pp. 16 - 19.

Stanford University n.d., ‘*The Discrete Cosine Transform (DCT)*, *Lossy data compression: JPEG*’, viewed 21 July 2024, <https://cs.stanford.edu/people/eroberts/courses/soco/projects/data-compression/lossy/jpeg/dct.htm>.

Starosolski, R. 2020, 'Hybrid adaptive lossless image compression based on discrete Wavelet Transform', *Entropy*, vol. 22, no. 7, viewed 18 August 2024, <https://doi.org/10.3390/e22070751>.

Ungureanu, V., Negirla, P., & Korodi, A. 2024, ‘Image-Compression techniques: classical and “Region-of-Interest-Based” approaches presented in recent papers’, *Sensors*, vol. 24, no. 3, viewed 30 July 2024, <https://doi.org/10.3390/s24030791>.

Why Do Math? 2011, ‘*Haar Wavelet Transformation’,* viewed 2 August 2024, <https://www.whydomath.org/node/wavlets/hwt.html.>.

Xiao, P. 2001, ‘*Image Compression by Wavelet Transform*’, East Tennessee State University, viewed 3 August 2024, Electronic Theses and Dissertations Paper 58, <https://dc.etsu.edu/cgi/viewcontent.cgi?article=1108&context=etd>.

Yadav, R., Gangwar, S. & Singh, H. 2012 'Study and analysis of wavelet based image compression techniques', *International Journal Of Engineering, Science And Technology*, vol. 4, no. 1, viewed 3 August 2024, <http://dx.doi.org/10.4314/ijest.v4i1.1S.>.

Name(s):Ryan Kho Yuen Thian, Ong Weng Kai, Yong Zee Lin   Programmer: RDS3S1   Group:2   Date:17/9/2024

Criteria	Weak	Average	Good	Excellent	Mark
Introduction	No or very little discussion on existing problem and the project The proposed project already exists, or with very minor change. No discussion or very little of introduction given to the related system or technology. 0-9	Little discussion on existing problem and introduction of proposed project. Minor ideas are modified from existing system(s). Introduction to the related system is given, but no evaluation provided. 10-18	Good discussion and evaluation of existing problem and the proposed project. Ideas modified from existing system, with some creative ideas are added. Good discussion and evaluation of the related system. 19-24	A very good discussion and evaluation of existing problem and the proposed project. Majority of the ideas are creative. A very good discussion and evaluation of the related system. 25-30	
Literature Review	Contents are retrieved directly from the literature without any paraphrasing. Selected literature was from unreliable sources (e.g. Web sources). No critical evaluation was provided. 0-9	Summary is lengthy, contents are retrieved directly from the literature without any critical evaluation. Selected literature was from unreliable sources. Literary supports were vague or ambiguous. 10-18	Summary is concise and clear. Able to identify key constructs and variables related to the research questions, from the relevant, reliable theoretical and research literature. Discussions on the validity, opinions, arguments and evidence are presented. 19-24	Summary is concise and clear, which integrates critical and logical details from the peer-reviewed theoretical and research literature. Attention is given to different perspectives, conditionality, threats to validity, and opinion vs. evidence. 25-30	
Methodology	No discussion or very little of introduction given to the methods applied. Brief design is provided but lack of explanation. Algorithm suggested is not appropriate or less relevant. 0-9	Brief description of system design, with some explanations. Introduction to the related application of the methods is given, but no evaluation provided. Algorithm suggested is relevant, but lack of understanding or explanation. 10-18	System design is well-illustrated, and with clear explanation. Good discussion and evaluation of the methods applied. Algorithm suggested is relevant and good explanation shows understanding. 19-24	System design is well-illustrated, with good explanation. Good discussion and evaluation of the relevant and practical methods applied. Algorithm suggested is relevant and good explanations to relate to the proposed project. 25-30	
Reference	No proper referencing is done. Only rely on website content, but no research papers. Reference list is not provided. 0-2	Reference list is provided but not with Harvard Referencing standard. Some mixture of reference sources. 3-5	Referencing is done properly. Some mixture of reference sources with at least one journal. 6-8	Rich mixture of reference sources especially good quality of research papers. Proper citations are done whenever necessary. Reference list follows proper Harvard Referencing standard and the information are complete. 9-10	
				Sum of scores	
				Final score = sum of scores/100*40 (base 40%)	