Лабораторна робота №4

Студента групи КН-11

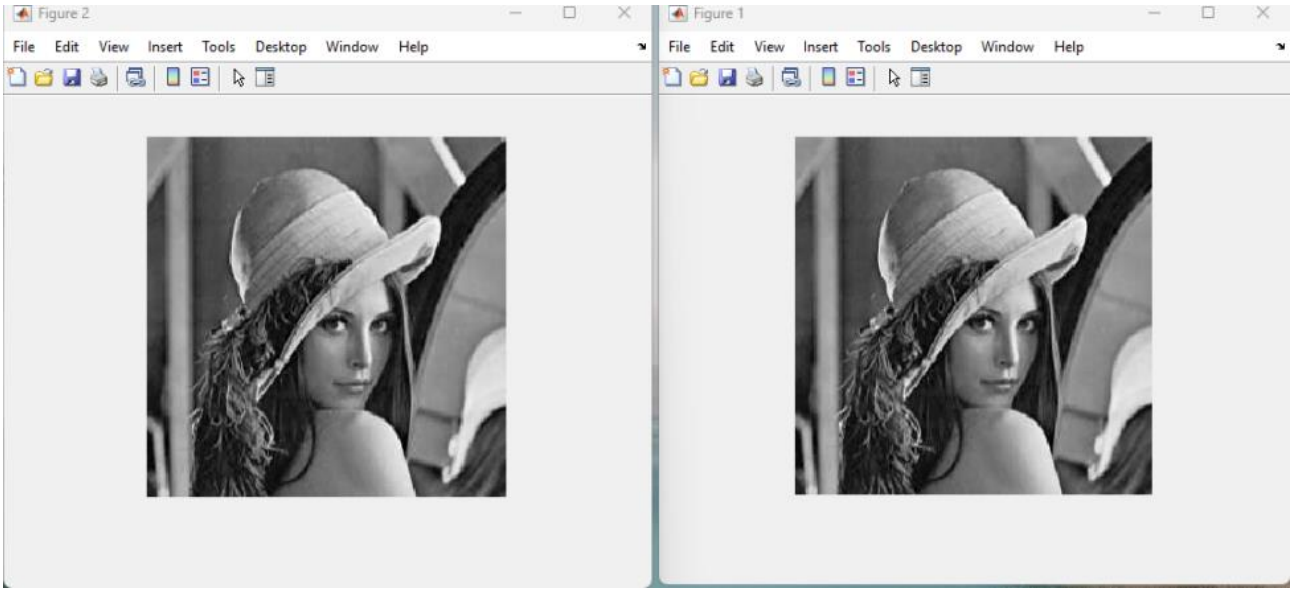
Сеня Тараса

З дисципліни Комп’ютерна графіка

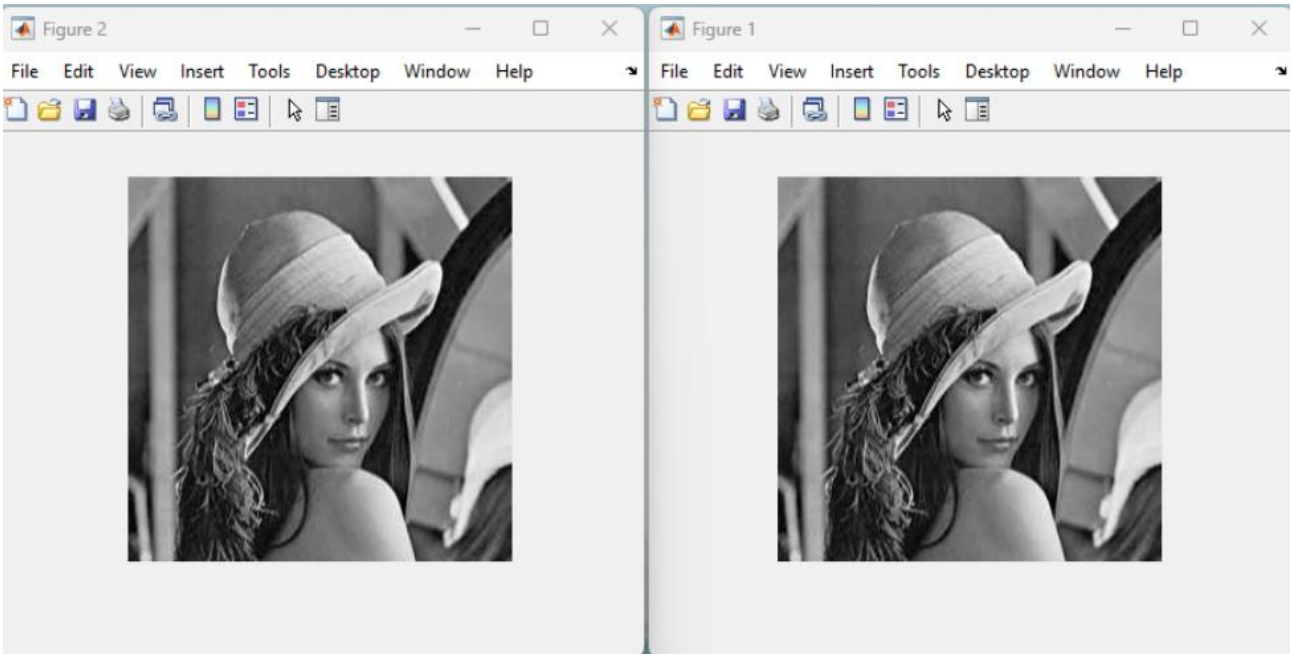
Виконання

Приклад 4.1

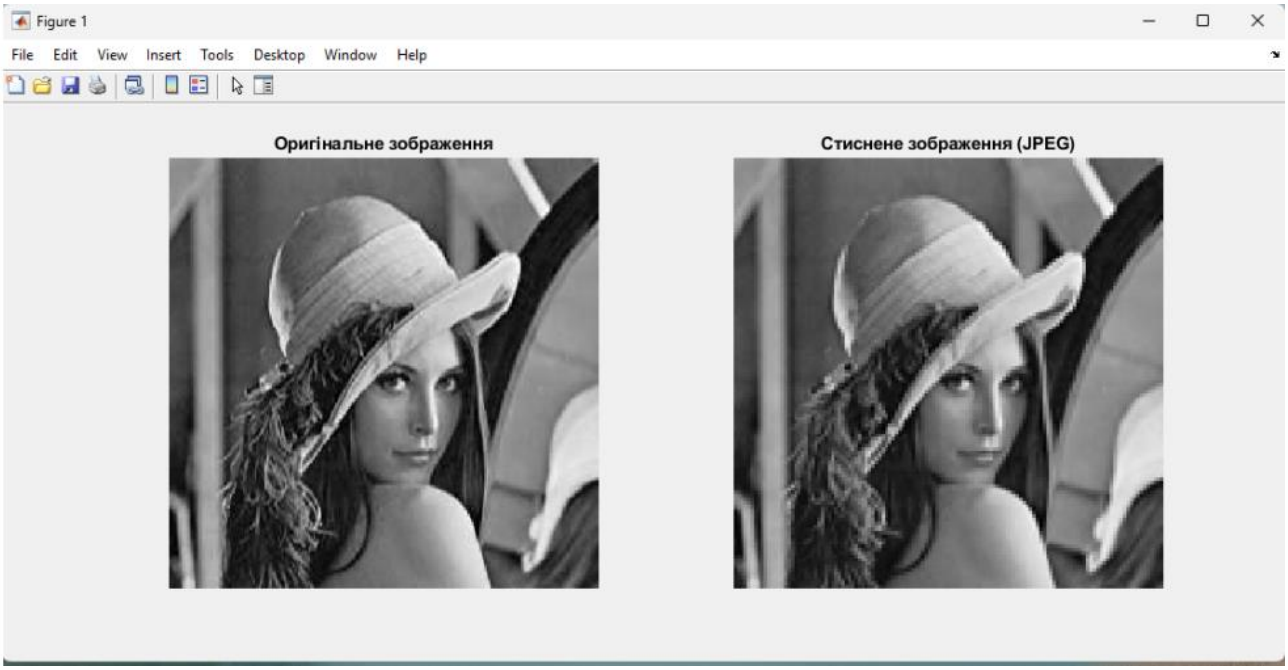
Кодер



Декодер



Стиснення зображення



1. В чому полягає суть алгоритму JPEG?

Алгоритм JPEG (Joint Photographic Experts Group) - це стандарт для стиснення зображень з втратами, який широко використовується для зменшення обсягу файлів при збереженні цифрових фотографій та інших різновидів зображень. Основна суть алгоритму JPEG полягає в використанні двох основних етапів: перетворення зображення та квантування.

1. **Перетворення зображення:**
   - Зображення перетворюється з простору кольорів RGB (чи іншого простору) у простір YCbCr, де Y відповідає за яскравість (лума), а Cb та Cr за хромінанс (колірності). Це перетворення дозволяє використовувати властивість людського зору, за якою люди більше чутливі до змін яскравості, ніж до змін кольору.

2. **Дискретне косинусне перетворення (DCT):**
   - Зображення розділяється на блоки розміром 8x8 пікселів, і для кожного блоку використовується дискретне косинусне перетворення (DCT). DCT перетворює блоки пікселів у частотний домен, де більшість енергії сконцентровані в низьких частотах.

3. **Квантування:**
   - Після DCT коефіцієнти блоків квантуються. Квантування полягає в поділі коефіцієнтів на числа заздалегідь визначені, зазвичай за допомогою матриці квантування. Великі коефіцієнти, які внесли б малу вклад у якість зображення, зменшуються до менших значень.

4. **Кодування:**

- Квантовані коефіцієнти кодуються за допомогою алгоритму кодування, такого як кодування довжин серій (Huffman coding). Це дозволяє представити коефіцієнти з більшими значеннями меншою кількістю біт, що допомагає в зменшенні розміру файлу.

Після цих етапів створюється стисле представлення зображення з втратами. Чим більше значень у матриці квантування і менше біт використовується для кодування, тим більша стисліть файлу, але тим менша якість зображення. JPEG дозволяє користувачеві вибирати рівень стиснення для балансу між розміром файлу та якістю.

2. Назвіть основні кроки алгоритму кодування зображень.

Алгоритм кодування зображень включає ряд основних кроків, які допомагають зменшити обсяг даних зображення або відео без значного втрати якості. Основні кроки алгоритму кодування зображень включають:

1. **Перетворення кольорів та простору кольорів:**
   - Зазвичай зображення подаються у форматі RGB (червоний, зелений, синій), але перетворення в інші простори кольорів, такі як YCbCr, може поліпшити ефективність стиснення. Простір YCbCr розділяє яскравість (Y) від хромінансу (Cb та Cr), використовуючи особливості людського зору, де більше чутливості до яскравості, ніж до кольору.

2. **Дискретне косинусне перетворення (DCT):**
   - Зображення розбивається на блоки (наприклад, 8x8 пікселів), і для кожного блоку використовується дискретне косинусне перетворення (DCT). DCT допомагає концентрувати енергію в низьких частотах, що поліпшує ефективність стиснення.

3. **Квантування:**
   - Коефіцієнти DCT квантуються, тобто округлюються або зменшуються до менших значень. Великі коефіцієнти, які б незначно вплинули на сприйняття людини, зменшуються, що дозволяє ефективніше кодувати дані.

4. **Кодування:**
   - Квантовані коефіцієнти кодуються за допомогою різних методів кодування. Одним з основних методів є кодування довжин серій (Huffman coding), яке використовується для представлення коефіцієнтів з високою ймовірністю.

5. **Кодування або стиснення розміру файлу:**

- Закодовані коефіцієнти зберігаються у файлі, що може піддаватися додатковому стисненню (наприклад, використовуючи алгоритми стиснення без втрат, такі як gzip).

Ці кроки дозволяють створити компактне представлення зображення або відео з використанням більш ефективного кодування та мінімізації зайвих даних.

3. В чому полягає ідея ентропійного кодування?

Ідея ентропійного кодування полягає в тому, щоб використовувати короткі коди для представлення часто зустрічаючихся символів та довші коди для менш ймовірних символів. Це базується на інформаційній теорії та понятті ентропії, яка вимірює "несподіваність" символу у випадковому потоці.

Основні принципи ентропійного кодування:

1. **Кодування для менш ймовірних символів:**
   - Якщо символ має менш ймовірний випадок зустрічі, йому надається більше довгий код. Це дозволяє скоротити середню довжину кодування для менш ймовірних символів і зменшити загальний обсяг даних.

2. **Кодування для часто зустрічаючихся символів:**
   - Часто зустрічаючимся символам присвоюються коротші коди, що дозволяє подати їх з меншим обсягом біт. Це забезпечує ефективніше представлення для часто вживаних символів.

3. **Безперервне використання бітів:**
   - Ентропійне кодування дозволяє використовувати біти більш ефективно, роблячи коротші коди для частіших подій і довші коди для рідше зустрічаючихся подій.

4. **Природна адаптація до ймовірностей:**
   - Спроектоване так, щоб більш ймовірні символи отримували коротші коди, а менш ймовірні — довші. Це сприяє зменшенню середньої кількості біт, потрібних для представлення символів, і покращенню ефективності кодування.

Застосування ентропійного кодування може значно зменшити обсяг даних у випадку, коли є нерівномірність в ймовірностях входження символів у потік. Одним із прикладів ентропійного кодування є алгоритм Хаффмана, який найчастіше використовується для стиснення даних в алгоритмах стиснення без втрат, таких як ZIP або gzip.

4. Опишіть спрощену структуру кодера JPEG.

Структура кодера JPEG може бути спрощена, але в основі її лежать основні етапи, необхідні для створення стислого зображення. Основні компоненти кодера JPEG включають:

1. **Перетворення кольорів та простору кольорів:**
   - Зображення зазвичай перетворюється з простору кольорів RGB в простір YCbCr. Це дозволяє використовувати особливості людського зору, де більша чутливість до яскравості (Y) порівняно з кольором (Cb та Cr).

2. **Дискретне косинусне перетворення (DCT):**
   - Зображення розділяється на блоки (зазвичай 8x8 пікселів), і для кожного блоку використовується дискретне косинусне перетворення (DCT). Це перетворення концентрує енергію в низьких частотах, що покращує ефективність стиснення.

3. **Квантування:**
   - Коефіцієнти DCT квантуються за допомогою матриці квантування. Великі коефіцієнти, які внесли б малу вклад у якість зображення, зменшуються, що дозволяє ефективніше кодувати дані.

4. **Кодування коефіцієнтів:**
   - Зквантовані коефіцієнти DCT кодуються за допомогою алгоритму Хаффмана або інших методів кодування. Це дозволяє представити значення коефіцієнтів з більшими ймовірностями входження меншою кількістю біт.

5. **Збереження вихідного файлу:**
   - Зкодовані коефіцієнти та інша необхідна інформація (наприклад, таблиці квантування та таблиці Хаффмана) зберігаються у вихідному файлі формату JPEG.

Це спрощена структура кодера JPEG, яка взагалі описує основні етапи стиснення зображення. В реальності кодер JPEG може включати додаткові етапи для оптимізації та покращення якості стисненого зображення.

5. Опишіть спрощену структуру декодера JPEG.

Структура декодера JPEG включає етапи, які відновлюють оригінальне зображення з закодованого файла формату JPEG. Основні компоненти декодера JPEG виглядають наступним чином:

1. **Зчитування файлу JPEG:**

- Декодер починає зчитування вмісту файлу JPEG. Це включає в себе прочитання заголовків, таблиць квантування, таблиць Хаффмана та інших необхідних даних.

2. **Декодування коефіцієнтів:**
   - Декодер відновлює коефіцієнти DCT, які були закодовані на етапі кодування. Для цього використовуються таблиці квантування та таблиці Хаффмана.

3. **Інверсне квантування:**
   - Зквантовані коефіцієнти відновлюються до оригінальних значень, використовуючи матриці квантування.

4. **Інверсне Дискретне Косинусне Перетворення (IDCT):**
   - Застосовується інверсне дискретне косинусне перетворення для отримання блоків пікселів із відновленими коефіцієнтами DCT.

5. **Інверсне перетворення кольорів:**
   - Якщо простір кольорів був змінений під час кодування, декодер повертається до вихідного простору кольорів, наприклад, з YCbCr до RGB.

6. **Відновлення оригінального зображення:**
   - Відновлені блоки пікселів об'єднуються, щоб сформувати оригінальне зображення.

7. **Збереження декодованого зображення:**
   - Отримане декодоване зображення може бути збережене в пам'яті або виведене на вихідний пристрій.

Це спрощена структура декодера JPEG, яка взагалі відображає процес відновлення оригінального зображення з закодованого файлу JPEG. При реалізації декодера може бути також використано додаткові оптимізації та методи для покращення продуктивності та якості декодованого зображення.

6. Опишіть структуру програми кодека JPEG мовою MATLAB.

Структура програми кодека JPEG в MATLAB може бути організована наступним чином. У цьому описі враховується, що кодек включає етапи кодування та декодування. Важливо зауважити, що реальна реалізація може бути складнішою, і цей опис є лише загальним напрямком.

```matlab
% Кодер JPEG
function encoded_data = jpeg_encoder(input_image)
```

```
    % Етап кодування

    % 1. Перетворення кольорів та простору кольорів
    YCbCr_image = rgb2ycbcr(input_image);

    % 2. Дискретне косинусне перетворення (DCT)
    dct_coefficients = apply_dct(YCbCr_image);

    % 3. Квантування
    quantized_coefficients = apply_quantization(dct_coefficients);

    % 4. Кодування коефіцієнтів
    encoded_data = huffman_encode(quantized_coefficients);
end

% Декодер JPEG
function output_image = jpeg_decoder(encoded_data)
    % Етап декодування

    % 1. Декодування коефіцієнтів
    quantized_coefficients = huffman_decode(encoded_data);

    % 2. Інверсне квантування
    dct_coefficients = inverse_quantization(quantized_coefficients);

    % 3. Інверсне Дискретне Косинусне Перетворення (IDCT)
    YCbCr_image = inverse_dct(dct_coefficients);

    % 4. Інверсне перетворення кольорів
    output_image = ycbcr2rgb(YCbCr_image);
end

% Допоміжні функції (необхідно реалізувати)
function YCbCr_image = rgb2ycbcr(input_image)
    % Функція перетворення кольорів з RGB в YCbCr
    % ...
end

function dct_coefficients = apply_dct(YCbCr_image)
    % Функція застосування Дискретного Косинусного Перетворення
    % ...
end

function quantized_coefficients = apply_quantization(dct_coefficients)
```

```
    % Функція квантування коефіцієнтів DCT
    % ...
end

function encoded_data = huffman_encode(quantized_coefficients)
    % Функція кодування коефіцієнтів за допомогою Хаффмана
    % ...
end

function quantized_coefficients = huffman_decode(encoded_data)
    % Функція декодування коефіцієнтів за допомогою Хаффмана
    % ...
end

function dct_coefficients = inverse_quantization(quantized_coefficients)
    % Функція інверсного квантування
    % ...
end

function YCbCr_image = inverse_dct(dct_coefficients)
    % Функція інверсного Дискретного Косинусного Перетворення
    % ...
end

function output_image = ycbcr2rgb(YCbCr_image)
    % Функція інверсного перетворення кольорів з YCbCr в RGB
    % ...
end
```

Це загальна структура кодека JPEG в MATLAB. Функції, які використовуються для кожного етапу, повинні бути реалізовані окремо. Реалізація деяких функцій, таких як застосування DCT чи квантування, може бути більш складною та включати ряд додаткових оптимізацій.

7. Завдання функції quantization.

Функція квантування (`quantization`) в кодеці JPEG відповідає за редукцію точності значень DCT-коефіцієнтів для піксельних блоків. Основна ідея полягає в тому, щоб зменшити кількість біт, які використовуються для представлення кожного коефіцієнта, тим самим стискати дані. Типова реалізація цієї функції виглядає приблизно так:

```matlab
```

```
function quantized_coefficients = quantization(dct_coefficients,
quantization_matrix)
   % dct_coefficients: DCT-коефіцієнти
   % quantization_matrix: матриця квантування

   % Розмір блока
   block_size = size(dct_coefficients);

   % Ініціалізація вихідної матриці
   quantized_coefficients = zeros(block_size);

   % Прохід по кожному коефіцієнту
   for i = 1:block_size(1)
      for j = 1:block_size(2)
         % Квантування
         quantized_coefficients(i, j) = round(dct_coefficients(i, j) /
quantization_matrix(i, j));
      end
   end
end
```

У цій функції `dct_coefficients` - це матриця DCT-коефіцієнтів для певного блока, а `quantization_matrix` - матриця квантування, яка визначає, як сильно кожен коефіцієнт має бути квантований. В результаті отримуємо матрицю `quantized_coefficients` зі значеннями коефіцієнтів після квантування.

Матриця квантування може варіюватися в залежності від якості, яку ви хочете досягти у JPEG-зображенні. Зазвичай вона передбачає більше втрати для високочастотних коефіцієнтів, щоб дозволити більш ефективне стиснення.

8. Завдання функції dequantization.

Функція де-квантування (`dequantization`) в кодеку JPEG відповідає за відновлення оригінальних DCT-коефіцієнтів після квантування. Основна ідея полягає в множенні квантованих коефіцієнтів на відповідні значення матриці квантування. Ось приблизна реалізація у MATLAB:

```matlab
function dequantized_coefficients = dequantization(quantized_coefficients,
quantization_matrix)
   % quantized_coefficients: квантовані DCT-коефіцієнти
   % quantization_matrix: матриця квантування
```

```
    % Розмір блока
    block_size = size(quantized_coefficients);

    % Ініціалізація вихідної матриці
    dequantized_coefficients = zeros(block_size);

    % Прохід по кожному квантованому коефіцієнту
    for i = 1:block_size(1)
        for j = 1:block_size(2)
            % Де-квантування
            dequantized_coefficients(i, j) = quantized_coefficients(i, j) *
quantization_matrix(i, j);
        end
    end
end
```

У цій функції `quantized_coefficients` - це матриця квантованих DCT-коефіцієнтів для певного блока, а `quantization_matrix` - матриця квантування, яка використовувалася під час квантування.

Після виклику цієї функції отримаємо матрицю `dequantized_coefficients` зі значеннями коефіцієнтів після де-квантування. Ці значення слід використовувати на етапі інверсного Дискретного Косинусного Перетворення (IDCT) для отримання оригінальних пікселів блоку.

9. Критерії оцінювання ступеня стиснення.

Ступінь стиснення в кодекуванні JPEG може бути оцінений за допомогою різних критеріїв. Декілька загальновизнаних метрик включає:

1. **Стиснення біт:**
   - Це відношення розміру стиснутого файла до розміру оригінального файла. Чим менше це відношення, тим ефективніше стиснення. Формула:

$$\text{Стиснення біт} = \frac{\text{Розмір оригінального файлу}}{\text{Розмір стиснутого файлу}}$$

   \[Стиснення біт = \frac{Розмір оригінального файлу}{Розмір стиснутого файлу} \]
2. **Втрати інформації (SNR - Signal-to-Noise Ratio):**
   - Це відношення енергії сигналу (оригінального зображення) до енергії шуму (різниці між оригінальним і відновленим зображенням). Чим більше SNR, тим менше втрати інформації. Формула:

$$SNR = 10 \cdot \log_{10} \left(\frac{\text{Сума квадратів амплітуд сигналу}}{\text{Сума квадратів амплітуд шуму}} \right)$$

\[SNR = 10 \cdot \log_{10}\left(\frac{\text{Сума квадратів амплітуд сигналу}}{\text{Сума квадратів амплітуд шуму}}\right) \]

3. **Помилка середньоквадратичного відхилення (MSE - Mean Squared Error):**
   - Це середнє значення квадратів різниці між пікселями оригінального і відновленого зображення. Чим менше MSE, тим краще. Формула:

$$MSE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} (I(i,j) - \hat{I}(i,j))^2$$

\[MSE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} (I(i,j) - \hat{I}(i,j))^2 \]

де \(M \) і \(N \) - розміри зображення, \(I(i,j) \) - піксель оригінального зображення, \(\hat{I}(i,j) \) - піксель відновленого зображення.

4. **Помилка пікового сигналу (PSNR - Peak Signal-to-Noise Ratio):**
   - Це логарифмічне відношення максимальної можливої енергії сигналу до енергії шуму. Використовується для вимірювання якості стисненого зображення. Формула:

$$PSNR = 10 \cdot \log_{10} \left(\frac{\text{Максимально можлива амплітуда сигналу}^2}{MSE} \right)$$

\[PSNR = 10 \cdot \log_{10}\left(\frac{\text{Максимально можлива амплітуда сигналу}^2}{MSE}\right) \]

Ці метрики допомагають кількісно оцінити якість стиснення та рівень втрат інформації при використанні алгоритмів стиснення, таких як JPEG. Важливо враховувати, що великі значення PSNR та малі значення MSE вказують на кращу якість відновленого зображення.

10. Як провести оцінювання якості реконструйованого зображення?

Оцінювання якості реконструйованого зображення включає в себе використання різних метрик та візуальних оцінок. Ось деякі способи оцінювання якості:

1. **Візуальна оцінка:**
   - Спостерігайте зображення і визначайте, наскільки воно відповідає оригінальному з точки зору контрасту, колірної точності та розмірів деталей. Візуальна оцінка може бути суб'єктивною, але важлива для визначення того, наскільки зображення задовольняє конкретні потреби.

2. **Peak Signal-to-Noise Ratio (PSNR):**
   - Обчисліть PSNR, щоб отримати кількісну оцінку різниці між оригінальним і відновленим зображенням. Вище PSNR вказує на кращу

якість відновлення. Цей підхід особливо корисний, коли доступно оригінальне зображення для порівняння.

3. **Structural Similarity Index (SSI):**
   - Це метрика, яка оцінює візуальну подібність між оригінальним і відновленим зображенням. Вона враховує якість, контраст і структурні зміни в зображенні.

4. **Mean Squared Error (MSE):**
   - MSE визначає середню квадратичну помилку між кожним пікселем оригінального і відновленого зображення. Менше MSE вказує на кращу якість.

5. **Блочні або локальні метрики:**
   - Оцінка якості на рівні окремих блоків чи регіонів може допомогти виявити артефакти або дефекти, які можуть вплинути на загальну якість.

6. **Оцінка часу виконання:**
   - В разі реального часу чи обробки великих обсягів даних важливо враховувати час виконання алгоритмів стиснення. Високе стиснення може призводити до затримок у виконанні.

Зазвичай для повноцінної оцінки якості реконструйованого зображення використовують комбінацію цих методів, бо вони доповнюють один одного та надають повнішу картину якості стиснення.