Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

Бакалаврська кваліфікаційна робота на тему:

«РОЗРОБКА ПРОГРАМНОГО МОДУЛЮ ДЛЯ АВТОМАТИЗОВАНОГО СТИСНЕННЯ ЗОБРАЖЕНЬ У КОМП'ЮТЕРНО-ІНТЕГРОВАНИХ СИСТЕМАХ»

Виконав: студент IV курсу, групи 1АКІТ-21б
спеціальності 151 – Автоматизація та
комп'ютерно-інтегровані технології
(шифр і назва спеціальності)
Владислав ГОРІ
(прізвище та ініціали)
Керівник: к.т.н, доцент, доцент кафедри АІІТ
Володимир КОЦЮБИНСЬКИЙ
(прізвище та ініціали)
«17» 06 2025 р.
Рецензент: к.т.н, доцент, професор кафедри КСУ
Микола БИКОВ
(прізвище та ініціали)
«17» 06 2025 р.

Допущено до захисту
завідувач кафедри АІІТ
Олег БІСІКАЛО
«18» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти перший (бакалаврський)
Галузь знань 15 - Автоматизація та приладобудування
Спеціальність Автоматизація та комп'ютерно-інтегровані технології
Освітньо-професійна програма Інтелектуальні комп'ютерні системи управління

ЗАТВЕРДЖУЮ
завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«18» 06 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Горі Владиславу Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Розробка програмного модулю для автоматизованого стиснення зображень у комп'ютерно-інтегрованих системах»
керівник роботи Коцюбинський Володимир Юрійович, к.т.н., доцент кафедри
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «20» березня 2025 року №97
2. Термін подання студентом роботи 10.06.2025 р.
3. Вихідні дані до роботи:
   - тип цифрового зображення – півтонові та кольорові;
   - мінімальна роздільна здатність зображення 128 x 128 пікселів, максимальна роздільна здатність 2048 x 2048 пікселів;
   - коефіцієнт стиснення 2 – 200.
   - Виграш у стисненні від 0,5% до 20% у порівнянні з класичним алгоритмом JPEG.
4. Зміст текстової частини (перелік питань, які потрібно розробити) Вступ. Аналіз сучасних методів стиснення зображень. Застосування ортогональних перетворень для стиснення зображень. Розробка програмного модулю для стиснення зображень. Висновки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень) схема стиснення зображення по стандарту JPEG, алгоритм стиснення зображень, експериментальні дослідження, середньоквадратичне відхилення значень пікселів стислого зображення від оригіналу, порівняння методів за критерієм якості PSNR

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконав прийняв
1-3	к.т.н. доцент каф. АІІТ Володимир ГАРМАШ		

7. Дата видачі завдання 20.03.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Термін виконання		Примітка
1	Вибір, узгодження та затвердження теми БКР	03.10.2024	09.10.2024	Вик
2	Аналіз предметної області та опрацювання літературних джерел.	12.03.2025	29.03.2025	Вик
3	Постановка задач для вирішення в БКР	01.04.2025	26.04.2025	Вик
4	Вирішення поставлених задач	29.04.2025	10.05.2025	Вик
5	Підтвердження достовірності отриманих результатів	13.05.2025	24.05.2025	Вик
7	Оформлення пояснювальної записки та графічної частини	27.05.2025	31.05.2025	Вик
8	Перевірка ПЗ на відповідність вимогам	03.06.2025	07.06.2025	Вик
9	Попередній захист, доопрацювання, та рецензування БКР	10.06.2025	15.06.2025	Вик
10	Захист БКР	16.06.2025	23.06.2025	Вик

Студент _____ Владислав ГОРІ
(підпис) (ім'я та прізвище)

Керівник роботи _____ Володимир КОЦЮБИНСЬК
(підпис) (ім'я та прізвище)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 83 сторінок формату А4, включаючи додатки, що містить 9 рисунків. Список використаних джерел містить 20 найменувань.

Робота присвячена розробці програмного модуля для автоматизованого стиснення зображень у комп'ютерно-інтегрованих системах на основі модифікованого алгоритму JPEG. Метою є підвищення ефективності стиснення цифрових зображень шляхом модифікації алгоритму JPEG, що забезпечує високий коефіцієнт стиснення при мінімальних спотвореннях. Проведено аналіз існуючих методів стиснення, досліджено можливості вдосконалення алгоритму JPEG та його переваги для комп'ютерно-інтегрованих систем. Запропоновано модифікацію M-JPEG, яка зберігає трудомісткість стандартного алгоритму, але забезпечує до 21% виграшу в стисненні. Розроблено програмний модуль, який реалізує запропонований метод, та проведено експерименти, що підтверджують його ефективність у підвищенні продуктивності обробки зображень у комп'ютерно-інтегрованих системах.

Ключові слова: стиснення, зображення; алгоритм JPEG, коефіцієнт стиснення.

ABSTRACT

The bachelor's qualification work consists of 83 pages of A4 format, including appendices, containing 9 figures. The list of used sources contains 20 items.

The work is devoted to the development of a software module for automated image compression in computer-integrated systems based on a modified JPEG algorithm. The goal is to increase the efficiency of digital image compression by modifying the JPEG algorithm, which provides a high compression ratio with minimal distortion. An analysis of existing compression methods was conducted, possibilities for improving the JPEG algorithm and its advantages for computer-integrated systems were investigated. A modification of M-JPEG was proposed, which retains the complexity of the standard algorithm, but provides up to 21% gain in compression. A software module was developed that implements the proposed method, and experiments were conducted to confirm its effectiveness in increasing the productivity of image processing in computer-integrated systems.

Keywords: compression, image; JPEG algorithm, compression ratio.

ЗМІСТ

ВСТУП

Актуальність. У сучасних умовах швидкість обробки та завантаження даних є ключовим фактором ефективності вебсайтів і комп'ютерно-інтегрованих систем. Зображення, які становлять значну частину цифрового контенту, потребують оптимізації для зменшення розміру файлів, що є критично важливим для мобільного інтернету з обмеженою пропускною здатністю. Зі зростанням обсягів інформації в цифрових технологіях стиснення зображень стає необхідним для економії ресурсів і забезпечення стабільної роботи систем. Галузі, пов'язані з обробкою даних, активно працюють над удосконаленням методів стиснення, щоб мінімізувати втрати якості при зменшенні розміру файлів. Розробка програмного модуля для автоматизації цього процесу сприяє підвищенню продуктивності систем, зниженню навантаження на канали зв'язку та прискоренню обробки інформації. Автоматизоване стиснення зображень забезпечує швидший доступ до даних, що особливо важливо для хмарних сервісів і потокових платформ. Воно також сприяє зниженню енергоспоживання серверів і пристроїв, що відповідає сучасним вимогам до енергоефективності. Крім того, оптимізація зображень покращує користувацький досвід, зменшуючи час завантаження сторінок і підвищуючи загальну швидкодію вебдодатків. У контексті розвитку технологій штучного інтелекту та обробки великих даних, автоматизоване стиснення є ключовим елементом для масштабування інформаційних систем. Таким чином, впровадження таких рішень має стратегічне значення для забезпечення конкурентоспроможності сучасних цифрових платформ.

Мета роботи – підвищення коефіцієнта стиснення цифрових зображень, шляхом створення програмного модуля для автоматизованого стиснення цифрових зображень у комп'ютерно-інтегрованих системах із максимальним збереженням якості.

Об'єкт дослідження – процес стиснення цифрових зображень.

Предмет дослідження – алгоритми стиснення, зокрема модифікації JPEG, для використання в автоматизованих модулях комп’ютерно-інтегрованих систем.

Завдання дослідження:

1. Проаналізувати сучасні методи автоматизованого стиснення зображень.
2. Створити програмний модуль на основі модифікованого алгоритму JPEG.
3. Провести тестування модуля та оцінити його ефективність у реальних умовах.

Практична цінність полягає в розробці програмного модуля, який оптимізує стиснення зображень із збереженням високої якості. Метод сприяє економії ресурсів і прискоренню завантаження зображень в інтернеті, що особливо актуально для мобільних користувачів із обмеженим з’єднанням. Впровадження модуля підвищує ефективність роботи комп’ютерно-інтегрованих систем, знижуючи навантаження на сервери та канали зв’язку. Крім того, він забезпечує покращення користувацького досвіду завдяки швидшій обробці та відображенню візуального контенту.

1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ СТИСНЕННЯ ЗОБРАЖЕНЬ

1.1 Аналіз об’єкту автоматизації

У сучасних комп'ютерно-інтегрованих системах (КІС), зокрема в таких галузях як промисловість, медицина, безпека, телекомунікації та мультимедіа, зображення є важливим елементом для візуалізації, аналізу та передачі інформації. Однак, із зростанням обсягів графічних даних, виникають серйозні проблеми з їх зберіганням, обробкою та передачею, особливо в умовах обмежених ресурсів, таких як низька пропускна здатність мереж, обмежена пам'ять пристроїв та обчислювальні можливості. В результаті автоматизація процесу стиснення зображень стає необхідною для підвищення ефективності роботи КІС, особливо коли йдеться про великі обсяги даних або реальний час обробки.

Об’єктом автоматизації в даній роботі є процес стиснення цифрових зображень, які надходять або генеруються в межах комп'ютерно-інтегрованих систем. Зазвичай зображення зберігаються у різних форматах, таких як JPEG, PNG, BMP тощо, кожен з яких має свої особливості щодо ступеня стиснення, підтримки прозорості, втрати якості та інших характеристик. Ручне стиснення або попередня підготовка зображень для подальшої передачі або зберігання є малоефективними, особливо коли йдеться про великі обсяги даних або постійне надходження нових зображень, таких як з камер відеоспостереження або автоматизованих виробничих ліній.

Основними характеристиками об’єкта автоматизації є:

– Висока інтенсивність надходження зображень – великі обсяги даних, які потребують швидкої обробки та стиснення.
– Різнорідність графічних форматів – необхідність роботи з різними типами зображень, що вимагає підтримки декількох форматів та методів стиснення.

- Чутливість до якості зображення – забезпечення балансу між ступенем стиснення та збереженням візуальної інформативності зображень, оскільки в багатьох системах зображення повинні зберігати свою чіткість і точність.
- Ресурсні обмеження – обмеження на об’єм пам'яті або пропускну здатність каналів передачі даних, що потребує ефективного стиснення без значної втрати якості.

Стиснення зображень в такому контексті виконує кілька важливих функцій: зменшення навантаження на файлові системи, прискорення передачі даних, зниження витрат на зберігання інформації та оптимізація процесів виведення і візуалізації. Це особливо критично для систем реального часу або розподілених КІС, де швидкість і продуктивність обробки є важливими для ефективного функціонування.

Таким чином, автоматизоване стиснення зображень у КІС дозволяє оптимізувати обробку графічних даних без участі оператора, забезпечуючи стабільну якість та продуктивність системи. Розроблений програмний модуль для автоматизованого стиснення зображень має здатність інтегруватися в більші інформаційні системи, підтримувати фонову обробку зображень, надавати можливість налаштування параметрів стиснення та адаптуватися до змін у вхідних даних, що робить його ефективним інструментом для роботи з великими обсягами графічної інформації.

1.2 Представлення цифрових зображень

Цифрові зображення є невід’ємною частиною сучасного інформаційного простору, використовуючись у різноманітних сферах: від вебдизайну до медичної діагностики. Їхнє представлення базується на перетворенні візуальної інформації в цифровий формат, що дозволяє обробляти, зберігати та передавати

дані за допомогою комп’ютерних систем. Основою цього процесу є дискретизація та квантування, які забезпечують можливість представлення

Цифрове зображення складається з пікселів – найменших елементів, кожен із яких має певне значення кольору або яскравості. Пікселі розташовані в двовимірній сітці, де їхня кількість визначає роздільну здатність зображення. Наприклад, зображення з роздільною здатністю 1920x1080 містить 1920 пікселів по горизонталі та 1080 по вертикалі. Чим вища роздільна здатність, тим детальнішим є зображення, але це також збільшує обсяг даних, необхідних для його зберігання [1].

Кожен піксель характеризується числовим значенням, яке описує його колір або інтенсивність. У монохромних зображеннях піксель має одне значення яскравості, зазвичай у діапазоні від 0 (чорний) до 255 (білий) для 8-бітного представлення. У кольорових зображеннях використовуються моделі, такі як RGB, де кожен піксель складається з трьох компонентів – червоного, зеленого та синього. Кожен компонент також кодується, наприклад, 8 бітами, що дає 256 рівнів інтенсивності для кожного каналу, а в сумі – понад 16 мільйонів можливих кольорів.

Окрім RGB, існують інші моделі кольорів, які використовуються залежно від потреб. Наприклад, модель CMYK застосовується в поліграфії, де кольори формуються шляхом комбінації ціану, магенти, жовтого та чорного. Модель HSV (тон, насиченість, яскравість) є зручною для редагування зображень, оскільки відповідає інтуїтивному сприйняттю кольорів людиною. Вибір моделі залежить від цільового використання зображення: RGB підходить для екранів, тоді як CMYK – для друку [2].

Цифрові зображення також можуть бути представлені у векторному або растровому форматі. Растрові зображення, такі як JPEG або PNG, складаються з пікселів і втрачають якість при масштабуванні. Векторні зображення, наприклад, у форматі SVG, базуються на математичних формулах, що описують лінії та

криві, тому зберігають якість незалежно від розміру. Растрові формати домінують у фотографіях, тоді як векторні – у логотипах і графічному дизайні.

Для зберігання цифрових зображень використовуються різні формати, кожен із яких має свої переваги. Формат JPEG забезпечує стиснення з втратами, що дозволяє значно зменшити розмір файлу, але може погіршити якість. PNG підтримує стиснення без втрат і прозорість, що робить його популярним для вебграфіки. Формат GIF обмежений 256 кольорами, але підтримує анімацію, що використовується для створення простих рухомих зображень. Новіші формати, такі як WebP, пропонують кращий баланс між якістю та розміром файлу, що робить їх перспективними для веброзробки [3-5].

Стиснення зображень є важливим аспектом їхнього представлення. Стиснення з втратами, як у JPEG, видаляє частину даних, які менш помітні для людського ока, тоді як стиснення без втрат, як у PNG, зберігає всю інформацію. Вибір методу залежить від вимог до якості та обсягу пам’яті.

Обробка зображень включає такі операції, як зміна розміру, фільтрація, корекція кольорів і видалення шумів. Ці процеси виконуються за допомогою алгоритмів, які аналізують піксельні дані. Наприклад, зменшення розміру зображення може використовувати інтерполяцію для розрахунку нових піксельних значень, тоді як фільтри, такі як розмиття Гаусса, застосовуються для згладжування.

Оптимізація зображень для комп’ютерно-інтегрованих систем передбачає баланс між якістю та продуктивністю. У веброзробці, наприклад, важливо мінімізувати час завантаження сторінок, що досягається шляхом зменшення розміру файлів без значної втрати деталізації. Сучасні інструменти, такі як ImageMagick або бібліотеки на кшталт Sharp, дозволяють автоматизувати цей процес, адаптуючи зображення до різних пристроїв і умов мережі.

Одним із викликів є забезпечення сумісності зображень із різними платформами та пристроями. Наприклад, дисплеї з високою щільністю пікселів (Retina) потребують зображень із вищою роздільною здатністю, що збільшує

обсяг даних. Водночас у мобільних мережах із низькою швидкістю передачі даних потрібні максимально стиснуті файли.

Перспективи розвитку включають використання штучного інтелекту для покращення якості зображень. Алгоритми машинного навчання можуть відновлювати деталі в стиснутих зображеннях або генерувати нові зображення на основі текстових описів. Крім того, формати на основі штучного інтелекту, такі як AVIF, обіцяють ще ефективніше стиснення, зберігаючи високу якість.

Таким чином, представлення цифрових зображень є складним процесом, що поєднує принципи дискретизації, кодування кольорів і стиснення даних. Від вибору формату та моделі кольорів залежить ефективність обробки та зберігання зображень, що робить цю тему ключовою для розвитку сучасних інформаційних технологій [4].

1.3 Класифікація методів стиснення. Основні характеристики

Стиснення цифрових зображень є ключовим процесом у комп’ютерно-інтегрованих системах, спрямованим на зменшення обсягу даних для зберігання та передачі без значного погіршення якості. Методи стиснення поділяються на основні категорії залежно від принципів роботи, цілей використання та впливу на якість зображення. Класифікація методів стиснення включає два основних типи: стиснення з втратами та без втрат. Кожен із них має свої підходи, алгоритми та характеристики, які визначають їхню застосовність у різних сценаріях.

Стиснення без втрат дозволяє повністю відновити оригінальне зображення без втрати інформації. Це досягається шляхом видалення надлишкових даних, які не впливають на візуальну або інформаційну цілісність. Такі методи широко використовуються в задачах, де збереження всіх деталей є критично важливим, наприклад, у медичній візуалізації чи архівації [6].

Основні характеристики:

Повне збереження даних: Усі піксельні значення оригінального зображення залишаються незмінними після декомпресії.

Ступінь стиснення: Зазвичай нижчий, ніж у стисненні з втратами, оскільки алгоритми обмежені видаленням лише надлишкової інформації. Ступінь стиснення залежить від структури зображення (наприклад, зображення з великими однорідними областями стискаються краще).

Алгоритми: До найпоширеніших належать кодування Хаффмана, LZW (Lempel-Ziv-Welch), а також методи, що використовуються у форматах PNG і FLAC. Наприклад, PNG застосовує DEFLATE, який комбінує кодування Хаффмана з LZ77 [7].

Застосування: Медичні зображення (рентген, МРТ), архівування документів, графічні редактори, де потрібна точність даних.

Обмеження: Обмежений ступінь стиснення (зазвичай 2:1 або 3:1) робить цей метод менш ефективним для великих обсягів даних або низькошвидкісних мереж.

Прикладом є формат PNG, який використовує стиснення без втрат для збереження якості зображень із прозорістю. Алгоритми без втрат аналізують повторювані шаблони в даних і замінюють їх компактнішими представленнями, що дозволяє економити місце без втрати інформації.

Стиснення з втратами передбачає видалення частини даних, які вважаються менш важливими для людського сприйняття. Це дозволяє значно зменшити розмір файлу, але відновлене зображення не є ідентичним оригіналу. Такі методи популярні в мультимедійних додатках, де розмір файлу важливіший за ідеальну точність.

Основні характеристики:

Втрата даних: Алгоритми видаляють інформацію, яка слабо впливає на візуальну якість (наприклад, високочастотні компоненти, які людське око сприймає менш чітко).

Високий ступінь стиснення: Залежно від налаштувань, розмір файлу може зменшуватися у десятки разів (наприклад, JPEG може досягати співвідношення 10:1 або більше).

Алгоритми: Найпоширенішим є JPEG, який використовує дискретне косинусне перетворення (DCT) для аналізу частотних складових зображення. Інші приклади включають JPEG 2000 (з використанням вейвлет-перетворення) та WebP.

Застосування: Вебграфіка, потокове передавання, соціальні мережі, де швидкість завантаження важливіша за ідеальну якість.

Обмеження: При високому рівні стиснення з’являються артефакти, такі як розмиття, блоковість чи втрата деталей, що може бути неприйнятним для професійних застосувань [8].

JPEG, наприклад, розбиває зображення на блоки 8x8 пікселів, застосовує DCT для перетворення просторових даних у частотну область, а потім відкидає менш значущі частоти. Це дозволяє значно зменшити розмір файлу, але призводить до втрати дрібних деталей.

Гібридні методи поєднують елементи стиснення з втратами та без втрат для досягнення балансу між якістю та розміром файлу. Такі підходи використовуються у форматах, які дозволяють налаштовувати ступінь стиснення залежно від потреб.

Основні характеристики:

Гнучкість: Користувач може вибирати між стисненням із втратами чи без втрат у межах одного формату (наприклад, JPEG 2000 підтримує обидва режими).

Алгоритми: Поєднують методи, такі як вейвлет-перетворення, із кодуванням ентропії (наприклад, арифметичне кодування).

Застосування: Універсальні формати, такі як WebP або AVIF, які підходять як для веброзробки, так і для зберігання зображень із високою якістю.

Переваги: Забезпечують кращу якість порівняно з JPEG при тому ж розмірі файлу або менший розмір при порівнянній якості.

Обмеження: Складніші алгоритми потребують більше обчислювальних ресурсів для кодування та декодування.

Формат WebP, наприклад, використовує як стиснення з втратами (на основі VP8), так і без втрат, що робить його універсальним для різних сценаріїв. AVIF, заснований на кодеку AV1, пропонує ще ефективніше стиснення завдяки сучасним алгоритмам [5].

Кожен метод стиснення має свої сильні та слабкі сторони, які визначають його використання:

Ефективність стиснення: Стиснення з втратами забезпечує значно більший ступінь зменшення розміру файлу, але за рахунок якості. Без втрат обмежується меншим коефіцієнтом стиснення, але гарантує збереження всіх даних.

Обчислювальна складність: Алгоритми без втрат, такі як DEFLATE, зазвичай менш вимогливі до ресурсів, ніж складні методи з втратами, наприклад, вейвлет-перетворення в JPEG 2000.

Якість зображення: Без втрат ідеально підходить для професійних застосувань, тоді як стиснення з втратами оптимальне для повсякденних задач, таких як вебграфіка.

Швидкість обробки: Стиснення з втратами зазвичай швидше для кодування, але декодування може бути складнішим у гібридних форматах, таких як AVIF.

Вибір методу стиснення залежить від конкретних вимог системи. Наприклад, у мобільних додатках пріоритет надається швидкості завантаження, тому формати з втратами, такі як JPEG або WebP, є популярними. У медичних системах, де важлива точність, використовуються формати без втрат, наприклад, PNG або DICOM [6].

Серед викликів – балансування між якістю та розміром файлу, а також сумісність із різними платформами. Нові формати, такі як AVIF, хоча й

ефективні, можуть не підтримуватися старішими пристроями. Крім того, складні алгоритми, такі як вейвлет-перетворення, потребують значних обчислювальних ресурсів, що може бути проблемою для пристроїв із обмеженою продуктивністю.

Класифікація найбільш поширених методів стиснення наведена на рисунку 1.1.

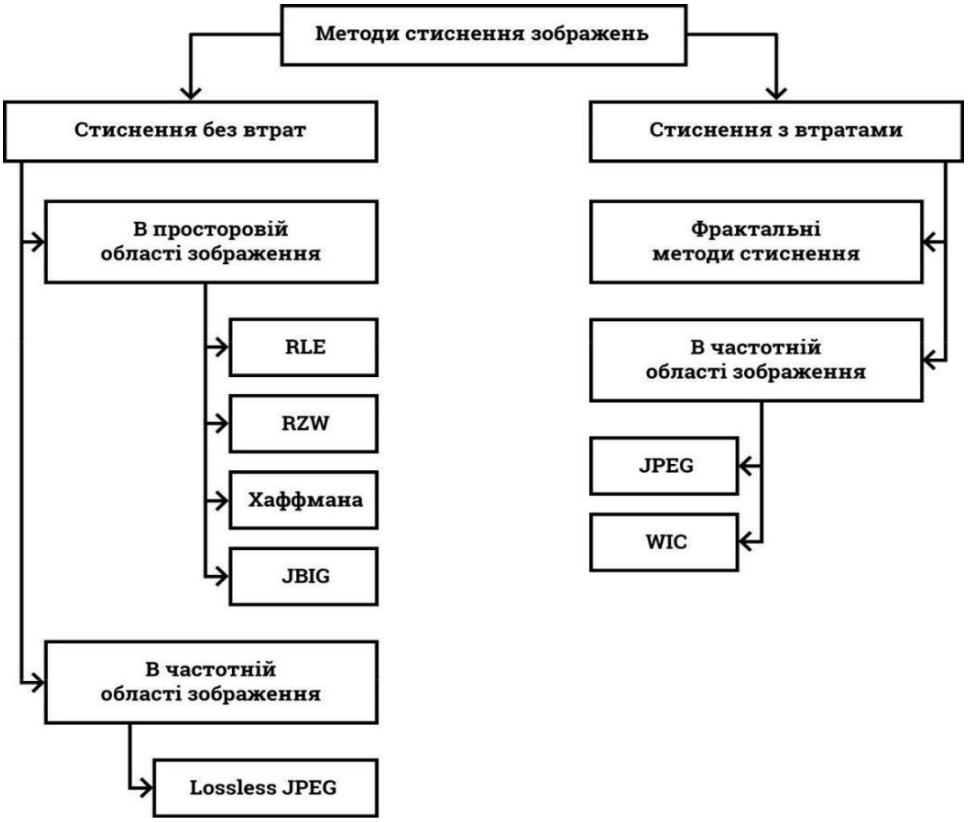

Рисунок 1.1 – Класифікація методів стиснення зображень

Сучасні тенденції у стисненні зображень пов’язані з використанням штучного інтелекту та машинного навчання. Алгоритми на основі нейронних мереж дозволяють створювати адаптивні моделі стиснення, які оптимізують видалення даних залежно від змісту зображення. Наприклад, методи, засновані на глибокому навчанні, можуть передбачати, які пікселі є менш важливими для людського сприйняття, підвищуючи ефективність стиснення.

Крім того, нові формати, такі як AVIF і JPEG XL, пропонують покращену компресію завдяки інноваційним підходам, таким як адаптивне кодування та

підтримка високодинамічного діапазону (HDR). Ці технології обіцяють революцію в обробці зображень, особливо для потокового передавання та віртуальної реальності [9].

Розглянемо етапи процедури стиснення даних в загальному вигляді. Будь-який метод стиснення реалізує три основних етапи (рисунок 1.1.):

– кодування або первинне стиснення;
– вторинне стиснення;
– декодування або відновлення зображення.

На першому етапі виконується перетворення вихідних даних з однієї форми подання в іншу. Зокрема, при стисненні зображень залежно від виду алгоритму стиснення може бути виконаний перехід від початкового зображення до наступних видів подання (таблиця 1.1).

Таблиця 1.1 – Переходи від початкового зображення

Вихідне зображення	Перетворене зображення
Матриця пікселів (значення інтенсивності)	Матриця компоненту спектру (спектральні перетворення)
	Набір коефіцієнтів перетворення (фрактальне стиснення)
	Опис об’єктів перетворення ()стиснення з розпізнаванням

На другому етапі компоненти перетворення квантуються і зводяться до виду зручного для статистичного кодування, а потім кодуються. На цьому етапі забезпечується ущільнення інформаційного потоку.

1.4 Алгоритми стиснення зображень без втрат

Стиснення зображень методом кодування серій (Run-Length Encoding, RLE) є одним із найпростіших і найефективніших підходів до зменшення обсягу даних, особливо для зображень із великими однорідними ділянками. Цей метод базується на заміні послідовностей однакових значень (серій) коротшим представленням, що складається з значення та кількості його повторень. Завдяки своїй простоті RLE широко застосовується в графіці, відеокодуванні та обробці даних, де структура зображення містить значну кількість повторюваних пікселів [10].

Принцип роботи RLE полягає в аналізі послідовності пікселів і групуванні однакових елементів. Наприклад, якщо в зображенні є ряд із 10 послідовних чорних пікселів, RLE замінить цю серію на код, який вказує значення "чорний" і кількість "10". Такий підхід значно скорочує обсяг даних, якщо зображення має великі однорідні області, як-от у бінарних картинках або графіках із простими формами.

Ефективність методу залежить від вмісту зображення. У випадках, коли пікселі різноманітні й не повторюються, RLE може навіть збільшити розмір даних через необхідність додавання кодів повторень. Тому цей метод найкраще підходить для зображень із низькою ентропією, таких як текстові документи, ігрові спрайти чи медичні зображення з чіткими контурами. Наприклад, у форматі BMP RLE використовується для стиснення палітрових зображень, де кожному кольору відповідає індекс.

Переваги RLE включають низьку обчислювальну складність і швидкість обробки, що робить його ідеальним для пристроїв із обмеженими ресурсами. Водночас метод має обмеження: він неефективний для складних фотографій із градієнтами чи деталями, де послідовності однакових пікселів короткі. У таких випадках потрібні складніші алгоритми, такі як JPEG.

Таким чином, кодування серій є цінним інструментом для стиснення зображень у специфічних умовах, забезпечуючи простоту реалізації та економію пам’яті там, де структура даних дозволяє. Його розвиток і комбінація з іншими методами, наприклад, у форматах TIFF, демонструють гнучкість цього підходу в сучасних технологіях обробки зображень.

Метод Хаффмана є одним із найпоширеніших алгоритмів стиснення без втрат, який використовується для кодування символів у компактні біти на основі їхньої частоти появи. Розроблений Девідом Хаффманом у 1952 році, цей підхід базується на побудові бінарного дерева, де кожному символу призначається код із різною довжиною залежно від його ймовірності. Чим частіше символ зустрічається в даних, тим коротший код йому надається, що дозволяє ефективно зменшувати обсяг інформації [7].

Процес починається з аналізу вхідних даних, де визначається частота кожного символу, наприклад, піксельного значення чи кольорового каналу в зображенні. На основі цих даних будується дерево Хаффмана: спочатку створюються вузли для кожного символу, а потім найменш часті елементи об’єднуються в нові вузли, формуючи ієрархію. Листові вузли представляють символи, а їхні коди формуються шляхом проходження від кореня до відповідного листа – лівий рух позначається 0, правий – 1 [11].

Перевагою методу є його адаптивність: він може застосовуватися до різних типів даних, включаючи зображення в растрових форматах. У контексті зображень метод Хаффмана часто використовується як частина стиснення без втрат, наприклад, у форматі PNG, де він комбінується з алгоритмом DEFLATE. Це дозволяє зберігати деталі зображення без спотворень, що критично для графічних редакторів чи архівування.

Однак метод має обмеження. Ефективність стиснення залежить від розподілу частот: якщо всі символи мають приблизно однакову ймовірність, виграш у розмірі буде мінімальним. Крім того, для кодування потрібно зберігати таблицю частот або саме дерево, що додає невеликий накладний витрат.

Незважаючи на це, завдяки своїй простоті та швидкості обробки метод Хаффмана залишається актуальним інструментом у комп’ютерних системах, особливо для оптимізації даних із чітко вираженими шаблонами [11].

1.5 Алгоритми стиснення з втратами

Стиснення за стандартом JPEG є одним із найпоширеніших методів обробки цифрових зображень, який забезпечує значне зменшення розміру файлів із прийнятною втратою якості. Розроблений спільною групою експертів з фотографії (Joint Photographic Experts Group), цей стандарт базується на алгоритмах, що використовують дискретне косинусне перетворення (DCT) для аналізу зображень. JPEG призначений переважно для кольорових фотографій і зображень із градієнтами, де невелика втрата деталей є менш помітною для людського ока [7-9].

Процес стиснення JPEG починається з розбиття зображення на невеликі блоки розміром 8x8 пікселів. Кожен блок піддається DCT, що перетворює просторові дані в частотну область. У результаті високі частоти, які відповідають дрібним деталям, отримують меншу вагу, тоді як низькі частоти, що визначають основні контури, зберігаються. Далі застосовується квантування – процес, під час якого менш значущі частоти відкидаються, що й забезпечує стиснення з втратами. На завершальному етапі дані кодуються за допомогою ентропійного кодування, наприклад, кодування Хаффмана, для подальшого зменшення розміру.

Однією з ключових особливостей JPEG є можливість регулювання ступеня стиснення. Користувач може вибирати рівень якості від 1 до 100, де нижчі значення дають більше стиснення, але призводять до появи артефактів, таких як блоковість чи розмиття. Ця гнучкість робить стандарт придатним для вебсайтів,

де важлива швидкість завантаження, а також для зберігання великих колекцій зображень.

Переваги JPEG включають високу ефективність стиснення (співвідношення до 10:1 або більше) та широку підтримку на різних платформах. Однак метод має обмеження: він менш ефективний для зображень із різкими границями чи текстами через появу артефактів. Крім того, повторне стиснення вже скомпресованих файлів може погіршити якість. Таким чином, JPEG залишається популярним вибором для повсякденного використання, але вимагає обережного підходу в професійних задачах, де важлива точність.

Кодування зображень за стандартом JPEG зазвичай починається з перетворення колірного простору з RGB в YUV (відоме також під назвоюYCbCr). Кольорове зображення традиційно може розглядатися як результат складання трьох компонент:

$$X_{ijC} = a_1 X_{ij(R)} + a_2 X_{ij(g)} + a_3 X_{ij(B)} \tag{1.1}$$

У типових зображеннях у форматі RGB є істотна кореляція між кольоровими компонентами і з точки зору стиснення зображення цей формат є завідомо надлишковим. Як відомо, в стандартах телевізійного мовлення використовується інше представлення зображень, при якому також використовуються 3 компоненти сигналу, але при цьому ці компоненти майже некорельовані один з одним. Компоненти *R, G і B* перетворюються в яскравісну компоненту *Y* і дві різнокольорові компоненти *U* і *V*, формату YUV.

Фрактальне стиснення зображень є одним із інноваційних підходів до зменшення обсягу даних, який базується на принципах самоподібності та фрактальної геометрії. На відміну від традиційних методів, таких як JPEG або PNG, цей метод використовує математичні властивості фракталів – структур, що повторюються на різних масштабах. Розроблений у 1980-х роках, зокрема завдяки роботам Майкла Барнслі, фрактальне стиснення стало унікальним

рішенням для обробки складних зображень, таких як природні ландшафти чи текстури [7].

Основна ідея полягає в тому, що багато зображень містять повторювані патерни, які можна описати за допомогою афінних перетворень. Алгоритм аналізує зображення, розбиваючи його на невеликі фрагменти, і шукає подібні структури в інших частинах картинки. Замість зберігання кожного пікселя окремо, система кодує ці подібності у вигляді математичних формул – ітераційних функцій системи (IFS). Під час декомпресії ці формули генерують зображення, відтворюючи його деталі на основі заданих перетворень.

Процес включає кілька етапів. Спочатку зображення розбивається на домени та діапазони – більші та менші блоки пікселів. Потім алгоритм визначає, як домени можна трансформувати (масштабувати, обертати, зміщувати), щоб вони відповідали діапазонам. Ці трансформації зберігаються як набір параметрів, що значно зменшує обсяг даних. Ключовою особливістю є те, що якість зображення може навіть покращуватися при масштабуванні, оскільки фрактали генерують деталі на основі самоподібності.

Однією з головних переваг фрактального стиснення є висока ступінь компресії, особливо для зображень із складними текстурами, таких як хмари, гори чи дерева. Співвідношення стиснення може сягати 100:1 або більше, що перевершує традиційні методи. Крім того, зображення залишаються чіткими при збільшенні, що робить цей метод привабливим для графічного дизайну чи картографії.

Проте є й значні недоліки. Процес кодування надзвичайно ресурсоємний, оскільки вимагає складних обчислень для пошуку відповідностей між доменами та діапазонами. Час обробки може тривати від хвилин до годин залежно від розміру зображення та обчислювальної потужності. Декомпресія, навпаки, є швидкою, але якість залежить від точності початкового аналізу. Крім того, метод менш ефективний для зображень із випадковими або однорідними областями, де самоподібність відсутня.

Фрактальне стиснення знайшло обмежене, але специфічне застосування. Воно використовувалося в ранніх системах обробки зображень, таких як архівування природних сцен, і тестувалося в телекомунікаціях для передачі даних. Сучасні дослідження зосереджені на інтеграції цього методу з машинним навчанням, що може пришвидшити процес кодування та покращити адаптивність до різних типів зображень [12].

Перспективи розвитку пов’язані з комбінуванням фрактального підходу з іншими техніками, наприклад, вейвлет-перетвореннями чи нейронними мережами. Такі гібридні методи можуть усунути недоліки, зберігаючи переваги високого стиснення та масштабованості. У майбутньому фрактальне стиснення може стати конкурентним рішенням для обробки великих обсягів даних у віртуальній реальності чи високоякісних відео.

Фрактальне стиснення зображень представляє унікальний підхід, який використовує самоподібність для ефективного зменшення обсягу даних. Незважаючи на складність реалізації та обмежену сферу застосування, його потенціал у створенні компактних і масштабних зображень залишається значним. З розвитком обчислювальних технологій цей метод може знайти нове життя, доповнюючи традиційні алгоритми стиснення.

В основі фрактального стиснення лежать кілька основних визначень і теорем. Розглянемо їх коротко.

Перетворення. Перетворення зіставляє точці в одному просторі точку в іншому просторі, можливо в тому ж самому просторі. Перетворення називається відображенням f і записується як $f : X_1 \to X_2$ якщо воно переводить простір X_1 в простір X_2.

Перетворення в метричному просторі (X, d) називається стискуючим відображенням, якщо існує константа s, $0 \le s < 1$ така що:

$$d(f(x_1), f(x_2)) \le s \cdot d(x_1, x_2) \tag{1.2}$$

де $d(x_1, x_2)$ - відстань від точки x_1 до точки x_2 в просторі X.

Константа *s* називається коефіцієнтом стиснення відображення *f*.

На рисунку 1.2. показаний приклад стиснення відображення (R_2, d_2), застосованого до безлічі точок в R_2. Тут дане перетворення застосовувалося більше одного разу: спочатку *f(x)* обчислювалося для точки *x*, а потім перетворення *f* застосовувалося до результату перетворення: *f(f(x))*. Таке послідовне, багаторазове застосування перетворення називають ітераціями і позначають як: f^{on}, тобто *f((... f(x) ...))*, де *f* застосовується *n* раз.

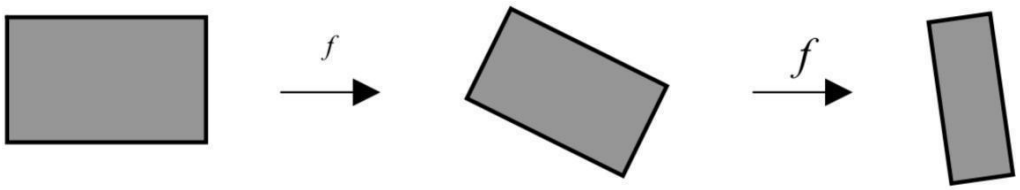

Рисунок1.2 – Стискуюче відображення для точок у просторі R^2

Система ітеруючих функцій (IFS) складається з повного метричного простору (*X*, *d*) і кінцевого безлічі стискають відображень w_n: $X_1 \to X_2$ з коефіцієнтами стиснення s_n. Таким чином, IFS можна позначити таким чином: {*X*, w_n: *n = 1,2, .., N*}, або якщо розглядати простір точок в R_2, то просто {w_n}.

У теорії фрактального стиснення доводиться теорема колажу, яку можна сформулювати наступним чином.

Нехай є двійкове зображення і нехай є стискають відображення, такі що: $U^N_n = {}_1w_n\ (L)$

Покривають *L* майже точно. Можна вважати таке $w_n\ (L)$ зменшеною копією *L*. Тоді теорема колажу стверджує, що аттрактор*A* (аттрактором IFS називається зображення, яке є єдиною нерухомою точкою IFS) системи {w_n} близький до *L*. «колаж» є набір областей $w_n\ (L)$.

Щоб на практиці застосувати теорему колажу (рисунок 1.3), необхідно вибрати перетворення, які будуть стискаючими відображеннями. Одним з таких перетворень є афінне перетворення:$T : R^2 \to R^2$:

$$T\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} \ddot{x} \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} \tag{1.3}$$

де *a, b, c, d, e, f*∈*R*. Афінні перетворення можуть здійснювати поворот, переміщення й масштабування (рисунок 1.3).

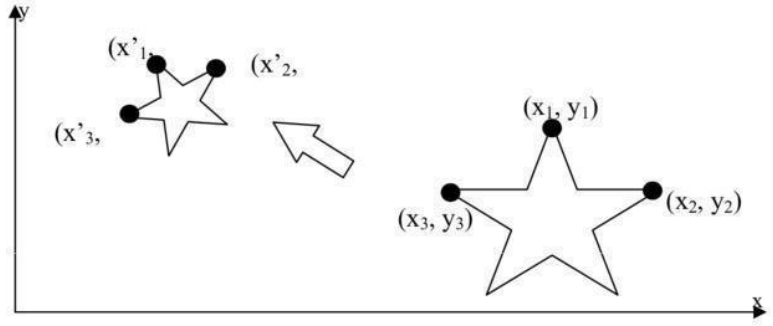

Рисунок 1.3 – Дія афінного перетворення на точки в просторі R^2

Існує два алгоритми побудови зображення-аттрактора за допомогою IFS. Один з них це пряме застосування теореми про стискають відображеннях, а інший - застосування так званої «гри хаосу». Детермінований алгоритм для побудови зображення є аттрактором IFS, до будь-якого початкового зображенню B застосовує теорему про стискають відображеннях. Алгоритм будує послідовність зображень A_n, багаторазово застосовуючи IFS відображення *W = {w1, w2, ..., wn}*:

$$A_n = W^{on}(B) \tag{1.4}$$

Детермінований алгоритм корисний з точки зору навчання, оскільки дозволяє бачити результат перетворення на кожному кроці ітерації (рисунок 1.4).

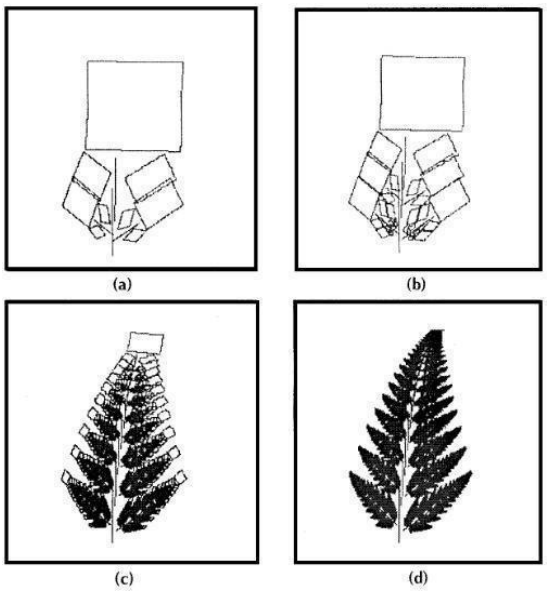

Рисунок 1.4 – Детермінований алгоритм, застосований до IFS папороті. Вид зображення A_n після: a) 2-х, b) 3-х, c) 10-і, d) 30-й ітерацій.

Імовірнісний алгоритм пов'язує з кожним афінним перетворенням w_i в IFS ймовірність p_i. Ці ймовірності визначають, наскільки щільно кожна частина зображення-аттрактора покрита точками. Імовірнісний алгоритм створює зображення-атрактори високої якості швидше, ніж детермінований алгоритм. Це відбувається не тільки через те, що імовірнісний алгоритм на кожному кроці ітерації виконує меншу роботу, а й сама виконана робота дає кращий результат.

Метод усіченого блочного кодування зображень є одним із підходів до стиснення цифрових даних, що базується на поділі зображення на окремі блоки та обробці їх із частковим збереженням інформації. Ця техніка належить до категорії методів стиснення з втратами, де пріоритетом є значне зменшення обсягу даних за рахунок видалення менш значущих деталей. Такий підхід

широко застосовується в сучасних комп’ютерно-інтегрованих системах, де важливе поєднання ефективності та прийнятної якості зображення [9].

Процес починається з розбиття зображення на невеликі прямокутні або квадратні блоки, наприклад, 8x8 або 16x16 пікселів. Кожен блок аналізується незалежно, що дозволяє обробляти зображення паралельно, підвищуючи швидкість обчислень. Наступним кроком є перетворення даних блоку в інший простір, наприклад, за допомогою дискретного косинусного перетворення (DCT), яке розкладає піксельні значення на частотні компоненти. Усічення полягає в тому, що низькочастотні компоненти, які несуть основну інформацію про зображення, зберігаються, тоді як високочастотні, що відповідають дрібним деталям і шуму, відкидаються або суттєво зменшуються.

Цей метод ефективний, оскільки людське око менш чутливе до високих частот, що дозволяє видаляти їх без значного спотворення сприйняття. Після усічення дані кодуються з використанням ентропійного кодування, такого як кодування Хаффмана або арифметичне кодування, для подальшого зменшення розміру.

Однією з головних переваг методу є висока ступінь стиснення. Завдяки усіченню високих частот розмір файлу може зменшуватися в десятки разів порівняно з оригіналом, що особливо корисно для передачі зображень у мережах із обмеженою пропускною здатністю. Крім того, алгоритм є відносно простим для реалізації, що робить його придатним для вбудованих систем або пристроїв із обмеженими ресурсами.

Метод також дозволяє гнучко налаштовувати рівень стиснення. Користувач може регулювати ступінь усічення, балансуючи між якістю зображення та його розміром. Наприклад, у професійних додатках, де важлива деталізація, рівень усічення може бути мінімальним, тоді як для вебграфіки допускаються більші втрати.

Основним недоліком методу є поява артефактів, таких як блоковість або розмиття, особливо при високому ступені стиснення. Блоковість виникає через

незалежну обробку блоків, коли межі між ними стають помітними. Щоб зменшити цей ефект, застосовуються додаткові фільтри, але це може збільшити обчислювальну складність.

Іншим обмеженням є залежність ефективності від вмісту зображення. У зображеннях із великою кількістю дрібних деталей (наприклад, текстур або гострих країв) метод може призводити до значних спотворень, тоді як однорідні області стискаються краще. Це вимагає адаптивного підходу до вибору розміру блоків або рівня усічення залежно від характеристик зображення.

Метод усіченого блочного кодування лежить в основі популярного формату JPEG, який використовується для стиснення фотографій і графіки. Його модифікації застосовуються в відео кодуванні, таких як MPEG, де схожі принципи використовуються для обробки кадрів. У сучасних технологіях цей підхід інтегрується з машинним навчанням для покращення якості стиснення, де нейронні мережі допомагають визначати, які дані можна відкинути з мінімальним впливом на сприйняття.

Перспективи розвитку включають створення адаптивних алгоритмів, які аналізуватимуть вміст зображення в реальному часі та оптимізуватимуть процес усічення. Також досліджуються гібридні методи, що комбінують усічене кодування з іншими техніками, такими як вейвлет-перетворення, для підвищення ефективності та якості.

Метод усіченого блочного кодування зображень є потужним інструментом для стиснення даних, що поєднує простоту реалізації з високою ефективністю. Незважаючи на певні недоліки, такі як артефакти та залежність від вмісту, його гнучкість і широке застосування роблять його актуальним у сучасних інформаційних системах. З розвитком технологій цей метод має шанс еволюціонувати, стаючи ще більш адаптивним і якісним.

Базовий алгоритм УБК будується наступним чином [9]. Зображення, представлене M × N – матрицею $||bij||$ яскравості пікселів, розбивається на

невеликі прямокутні блоки $m \times n$ елементів. Кожен такий блок обробляється незалежно від інших, тому опишемо алгоритм обробки одного блоку.

Обробка блоку починається з обчислення порога і двох рівнів квантування, потім проводиться квантування блоку на два рівня, після чого слідує упаковка проквантованного блоку. Для визначення рівнів квантування спочатку обчислюються два перших вибіркових моменти – середнє значення *C* і середній квадрат E:

$$C = \frac{1}{m \times n} \sum_i \sum_j b_{ij}, \; E = \frac{1}{m \times n} \sum_i \sum_j b_{ij}^2 \tag{1.4}$$

(де підсумовуються елементи зображення в межах блоку) і дисперсія

$$\sigma^2 = E - c^2 \tag{1.5}$$

Порогова величина квантування d покладається рівною середньому C. Верхній a і нижній b рівні квантування обчислюються за такими формулами:

$$a = C - \sigma\sqrt{q/(p-q)}, \; b = C + \sigma\sqrt{(p-q)/q} \tag{1.6}$$

де $p = m \times n$ – число елементів блоку, q – число елементів, що перевищують поріг d.

Квантування проводиться за правилом:

$$s_{ij} = \begin{cases} a, \text{якщо } b_{ij} < d \\ b, \text{якщо } b_{ij} \geq d \end{cases} \tag{1.7}$$

де s_{ij} – елементи зображення після квантування.

Після квантування виходить блок, що містить тільки рівні a та b. Неважко показати, що середнє значення і середній квадрат вихідного і проквантованного

блоків збігаються. Практично для зручності подальшої упаковки замість a записується нуль, замість b – одиниця. Рівні a і b записуються окремо [19].

Упаковка полягає в том у, що блок, який містить лише нулі і одиниці, інтерпретується як двійкове число, що має розрядів. Відновлення закодованого зображення також проводиться по блоках і полягає в розпакуванні і зворотной підстановці.

З алгоритму видно, що ступінь стиснення безпосередньо залежить від розмірів блоку. Найбільш задовільні результати, як за ступенем стиснення, так і за якістю відновленого зображення були отримані при використанні блоків розміром 4 × 4 [11].

1.6 Вибір і обґрунтування аналогу

Для розробки алгоритму стиснення зображень на основі дискретних ортогональних функцій як базового аналогу було обрано метод стиснення JPEG. Цей вибір обґрунтований низкою технічних, практичних та історичних аспектів, які роблять JPEG надійною відправною точкою для порівняння та вдосконалення. JPEG є одним із найпоширеніших стандартів компресії зображень, що використовується в усьому світі, і його характеристики дозволяють глибше зрозуміти принципи стиснення з втратами, які можуть бути адаптовані або вдосконалені в новому алгоритмі.

Метод JPEG базується на дискретному косинусному перетворенні (DCT), яке розбиває зображення на блоки розміром 8x8 пікселів і перетворює просторові дані в частотну область. Цей підхід дозволяє видаляти високочастотні компоненти, які менш помітні для людського ока, що забезпечує ефективне зменшення розміру файлу. Подібність із дискретними ортогональними функціями полягає в тому, що обидва методи використовують математичні перетворення для аналізу та компресії даних. Однак JPEG уже довів свою

здатність балансувати між якістю зображення та обсягом даних, що робить його зручним еталоном для порівняння.

Крім того, JPEG підтримує гнучкість у регулюванні рівня стиснення, що дозволяє адаптувати алгоритм до різних сценаріїв – від високоякісних зображень до файлів із мінімальним розміром. Ця адаптивність є важливою для тестування нового алгоритму, оскільки вона дає змогу оцінити, наскільки розроблений метод може конкурувати з уже усталеним стандартом у різних умовах.

Використання JPEG як аналогу є логічним через його широке впровадження в реальних системах. Цей формат застосовується в веброзробці, цифровій фотографії та потоковому передаванні, що забезпечує доступ до великої кількості тестових даних і практичних прикладів. Завдяки цьому можна провести порівняльний аналіз продуктивності нового алгоритму в умовах, близьких до реального використання. Наприклад, можна перевірити, як дискретні ортогональні функції впливають на швидкість обробки чи появу артефактів у порівнянні з DCT у JPEG.

Іншим важливим аспектом є сумісність. Оскільки JPEG підтримується майже всіма пристроями та програмним забезпеченням, його вибір як аналогу дозволяє оцінити, наскільки легко новий метод може інтегруватися в існуючі платформи. Це особливо актуально для комп’ютерно-інтегрованих систем, де важлива універсальність і можливість роботи з уже усталеними стандартами.

Переваги JPEG як аналогу включають його перевірену ефективність і добре задокументовані алгоритми. Стандарт JPEG був розроблений спільнотою експертів і пройшов багаторічну оптимізацію, що робить його надійним орієнтиром. Водночас обмеження JPEG, такі як поява блокових артефактів при високому стисненні чи недостатня ефективність для зображень із великою кількістю деталей, створюють простір для вдосконалення. Ці недоліки слугують мотивацією для розробки нового алгоритму на основі дискретних ортогональних функцій, який може запропонувати кращий баланс між стисненням і збереженням якості.

Порівняння з JPEG дозволить визначити сильні сторони розробленого методу, зокрема його здатність зменшувати розмір файлу без значних візуальних втрат або покращувати обробку складних зображень. Наприклад, дискретні ортогональні функції можуть забезпечити більш гнучке представлення даних у частотній області, що потенційно зменшить артефакти, властиві JPEG. У процесі дослідження планується провести експерименти, щоб кількісно оцінити різницю в коефіцієнтах стиснення, часі обробки та суб'єктивній якості зображень.

Таким чином, вибір JPEG як аналогу є виправданим завдяки його технічній основі, практичному досвіду використання та можливостям для порівняльного аналізу. Цей підхід допоможе не лише оцінити ефективність нового алгоритму, а й визначити напрямки його подальшого вдосконалення для широкого впровадження в комп'ютерно-інтегрованих системах.

Недоліками аналогу є:

- при підвищенні ступені стиснення зображення розпадається на окремі квадрати (8 х 8, 16 х 16). Це пов'язано з тим, що відбуваються великі втрати в низьких частотах при квантуванні, і відновити вихідні дані стає неможливо.

- проявляється «ефект Гіббса» – ореоли на границях різких переходів кольорів [17].

1.7 Висновки до розділу

Методи компресії зображень поділяються на дві основні категорії: компресію без втрат і компресію з втратами.

Компресія без втрат забезпечує повну ідентичність відновленого зображення з оригіналом. Такі підходи застосовуються в ситуаціях, коли точність вихідного та отриманого зображень має вирішальне значення або коли шумові компоненти є предметом аналізу, наприклад, у медичних дослідженнях

чи наукових експериментах. Проте ці методи забезпечують лише незначні показники стиснення.

Компресія з втратами використовується, коли незначні зміни в зображенні, непомітні для людського зору, вважаються допустимими, зокрема для фото- та відеокамер, особливо в веб середовищі. Такі зміни дозволяють досягти значно вищих коефіцієнтів стиснення порівняно з методами без втрат.

Було проведено аналіз сучасних методів компресії зображень та здійснено вибір відповідного аналогу.

2 ЗАСТОСУВАННЯ ОРТОГОНАЛЬНИХ ПЕРЕТВОРЕНЬ ДЛЯ СТИСНЕННЯ ЗОБРАЖЕНЬ

Ортогональні перетворення відіграють важливу роль у сучасних методах стиснення цифрових зображень, забезпечуючи ефективне представлення даних у компактній формі. Ці методи базуються на розкладі зображення на компоненти, що полегшує видалення надлишкової інформації без значного погіршення якості. Завдяки своїм математичним властивостям, ортогональні перетворення дозволяють аналізувати зображення в частотній області, що є ключовим для оптимізації процесів кодування та декодування.

Ортогональні перетворення, такі як дискретне косинусне перетворення (DCT) або вейвлет-перетворення, перетворюють просторові дані пікселів у набір коефіцієнтів, які описують частотні складові зображення. Цей процес базується на властивості ортогональності, що гарантує збереження енергії зображення та можливість зворотного перетворення без втрат при ідеальних умовах. DCT, наприклад, розбиває зображення на блоки (зазвичай 8x8 пікселів) і обчислює коефіцієнти для низько-, середньо- та високофреквентних компонентів.

Низькочастотні коефіцієнти відповідають основним контурам і кольорам зображення, тоді як високофреквентні пов’язані з дрібними деталями та шумами. У стисненні з втратами ці високофреквентні компоненти частково відкидаються, оскільки людське око менш чутливе до їхніх змін. Це дозволяє значно зменшити обсяг даних, зберігаючи візуальну схожість із оригіналом.

Найвідомішим прикладом використання ортогональних перетворень є алгоритм JPEG. У цьому форматі DCT застосовується до кожного блоку зображення, після чого виконується квантування – процес, що знижує точність коефіцієнтів. Квантування є ключовим етапом, який визначає ступінь стиснення: сильніше квантування призводить до більшого зменшення розміру файлу, але додає артефакти, такі як блоковість. Після квантування використовується

ентропійне кодування (наприклад, кодування Хаффмана), щоб додатково стиснути дані.

Іншим прикладом є JPEG 2000, який використовує вейвлет-перетворення замість DCT. Вейвлети дозволяють аналізувати зображення на різних рівнях деталізації, що забезпечує кращу якість при високих ступенях стиснення. Цей метод розкладає зображення на кілька масштабів, відокремлюючи грубі контури від дрібних текстур. Завдяки цьому JPEG 2000 ефективніший для зображень із складними градієнтами або текстурами порівняно з класичним JPEG.

Використання ортогональних перетворень має кілька переваг. По-перше, вони дозволяють ефективно виділити надлишкову інформацію, що зменшує розмір файлу без значних втрат якості. По-друге, ортогональність забезпечує зворотність процесу, що полегшує декодування. По-третє, такі методи добре адаптовані до обробки великих обсягів даних, що важливо для комп’ютерно-інтегрованих систем.

Однак є й обмеження. Алгоритми, такі як DCT, чутливі до блокових артефактів, особливо при сильному стисненні. Вейвлет-перетворення, хоча й зменшує ці ефекти, потребує більше обчислювальних ресурсів, що може бути проблемою для пристроїв із низькою продуктивністю. Крім того, ефективність залежить від змісту зображення: однорідні зображення стискаються краще, ніж ті, що мають багато дрібних деталей.

Розвиток ортогональних перетворень іде в напрямку інтеграції з штучним інтелектом. Алгоритми машинного навчання можуть оптимізувати квантування або адаптувати перетворення до конкретного типу зображення, наприклад, фотографій чи медичних сканів. Це дозволяє досягти кращого балансу між стисненням і якістю. Нові формати, такі як AVIF, також використовують вдосконалені версії вейвлет-перетворень, що обіцяє підвищення ефективності в майбутньому.

Таким чином, ортогональні перетворення є основою сучасних методів стиснення зображень, забезпечуючи ефективне кодування та декодування даних.

Їхнє вдосконалення відкриває нові можливості для оптимізації ресурсів у цифрових системах, роблячи їх незамінним інструментом у обробці зображень.

2.1 Перетворення Адамара

Перетворення Адамара є одним із математичних інструментів, що використовуються в обробці сигналів і зображень, зокрема для стиснення даних. Цей метод базується на ортогональних матрицях Адамара, які складаються з елементів +1 і -1, що забезпечує простоту обчислень і ефективність у певних задачах. Розроблене на основі робіт французького математика Жака Адамара, це перетворення знайшло широке застосування в цифровій обробці зображень, кодуванні та аналізі сигналів завдяки своїм унікальним властивостям.

Перетворення Адамара виконується шляхом множення вхідного вектора або матриці зображення на матрицю Адамара відповідного розміру. Матриці Адамара мають розмір, який є степенем двійки (наприклад, 2x2, 4x4, 8x8), і генеруються рекурсивно. Основна ідея полягає в розкладанні вхідних даних на суму базисних функцій, які представляють різні частотні компоненти. На відміну від дискретного косинусного перетворення (DCT), яке використовується в JPEG, перетворення Адамара не включає множення на дробові коефіцієнти, що робить його обчислення швидшими на апаратному рівні.

Процес починається з поділу зображення на блоки, наприклад, 8x8 пікселів. Кожен блок множиться на матрицю Адамара, що перетворює просторові дані в частотну область. Результуючі коефіцієнти відображають вклад різних частот у зображення: низькі частоти концентруються в лівому верхньому куті матриці, а високі – у правому нижньому. Це дозволяє видаляти менш значущі високі частоти для стиснення без значного впливу на візуальну якість.

Однією з головних переваг перетворення Адамара є його обчислювальна простота. Оскільки матриці містять лише +1 і -1, обчислення зводяться до додавання та віднімання, що знижує навантаження на процесор. Це робить метод особливо привабливим для вбудованих систем із обмеженими ресурсами. Крім того, перетворення Адамара є ортогональним, що забезпечує повне відновлення даних при зворотному перетворенні, якщо не застосовувати стиснення з втратами.

Проте цей метод має обмеження. Ефективність стиснення нижча порівняно з DCT або вейвлет-перетворенням, оскільки матриці Адамара не оптимізують енергію сигналу так само ефективно. Це означає, що для досягнення того ж рівня компресії можуть знадобитися більші блоки або додаткові методи обробки. Крім того, перетворення Адамара менш адаптивне до складних текстур і градієнтів, що часто зустрічаються в природних зображеннях.

Перетворення Адамара використовується в задачах, де пріоритетом є швидкість обчислень і простота реалізації. Наприклад, у ранніх системах кодування відео, таких як H.261, цей метод застосовувався для попередньої обробки даних. У сучасних комп’ютерно-інтегрованих системах його можна комбінувати з іншими техніками, такими як квантування та ентропійне кодування, для покращення результатів стиснення.

Інше застосування пов’язане з аналізом зображень у реальному часі, наприклад, у системах розпізнавання об’єктів або стеження. Завдяки швидкості обчислень перетворення Адамара підходить для пристроїв із низькою обчислювальною потужністю, таких як камери відеоспостереження або сенсори в автономних системах.

Для подолання обмежень базового перетворення Адамара були розроблені його модифікації, такі як швидке перетворення Адамара (Fast Hadamard Transform, FHT). FHT використовує рекурсивну структуру матриць для зменшення обсягу обчислень з $O(n^2)$ до $O(n \log n)$, що значно прискорює процес.

Ця модифікація широко застосовується в цифровій обробці сигналів і дозволяє ефективніше інтегрувати метод у сучасні алгоритми стиснення.

Крім того, дослідники експериментують із комбінацією перетворення Адамара з іншими техніками, такими як вейвлет-перетворення, для створення гібридних методів. Такі підходи можуть покращити якість стиснення, зберігаючи при цьому переваги швидкості обчислень.

Перетворення Адамара є цінним інструментом у обробці зображень завдяки своїй простоті та ефективності в обчисленнях. Хоча воно поступається іншим методам за ступенем стиснення, його унікальні властивості роблять його актуальним для спеціалізованих застосувань. З розвитком технологій і модифікацій, таких як FHT, цей метод може знайти нове життя в інноваційних системах, де важлива швидкість і економія ресурсів. Його вивчення та вдосконалення залишаються важливими для оптимізації обробки даних у сучасних комп'ютерно-інтегрованих системах.

Простим методом перетворення вихідного сигналу є перетворення Адамара. Матриця перетворення H для двох випадкових величин буде мати вигляд:

$$H = \begin{vmatrix} 1 & 1 \\ 1 & -1 \end{vmatrix} \tag{2.1}$$

Якщо з елементів матриці H скласти базисні вектори $e_1 = [h_{11}, h_{12}] = [1, 1]$, $e_2 = [h_{21}, h_{22}] = [1, -1]$, то вони будуть характеризувати поворот ортогональної системи координат на 45^0 відносно одиничного базису (рисунок 2.1).

Якщо випадкові величини x_1 та x_2 мають кореляційну залежність, то проекція вектора $X = [x_1, x_2]^T$ на базисний вектор e_1 буде, в середньому, більше, ніж проекція цього ж вектора на базисний вектор e_2. Завдяки цьому інформація

буде зосереджувати в першому коефіцієнті перетворення. Другий коефіцієнт слугує для уточнення представлення вектора *X* у новому базисі.

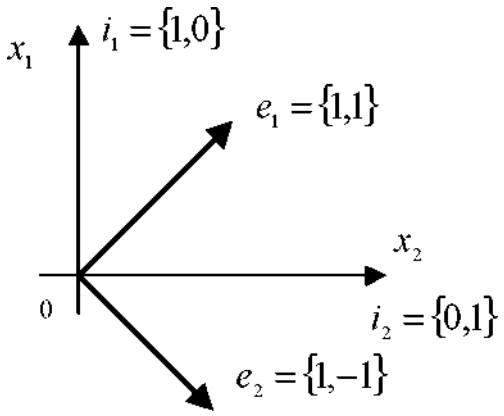

Рисунок 2.1 – Розташування базисних векторів перетворення Адамара щодо одиничного базису

При збільшенні розмірності простору більша частина інформації буде зосереджена в малому числі коефіцієнтів, які потрібно зберегти з найменшими втратами. Інші коефіцієнти можна або відкинути, або квантувати із більшим кроком.

Матрицю Адамара розмірності 4x4 елементів легко побудувати з матриці Адамара *H* розмірністю 2x2 елементи:

$$H_4 = \begin{vmatrix} H & H \\ H & -H \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{vmatrix} \quad (2.3)$$

Користуючись співвідношенням (2.3), можна побудувати матрицю Адамара будь-якої розмірності $N = 2^n$, де *n* – будь-яке додатне число.

При аналізі зображень за допомогою даного перетворення спочатку обчислюються коефіцієнти по рядках зображення, а потім по стовпцях. У результаті виходить роздільне перетворення виду:

$$Y = HAH^T, \tag{2.4}$$

де A – N x N матриця вихідного зображення.

Так як матриця H являє собою оператор ортогонального перетворення, то зворотне перетворення з Y в A запишеться у вигляді:

$$A = H^T YH \tag{2.5}$$

Однією з ключових переваг цього перетворення є легкість реалізації та незначна обчислювальна складність. Розкладання показує високу ефективність для функцій із кусочно-постійними значеннями, оскільки базисні функції Адамара акцентують увагу на сталих компонентах сигналу. У таких випадках значна частина коефіцієнтів розкладання наближається до нуля, що знижує ентропію перетворених даних і підвищує коефіцієнт стиснення. Проте в реальних умовах такі сигнали трапляються нечасто, що створює потребу в розробці перетворення, яке забезпечуватиме оптимальне або близьке до нього розкладання сигналів, краще відображаючих природні зображення. Оптимальність у цьому контексті визначається як мінімальна середня дисперсія помилки при відновленні, коли відкидаються окремі високочастотні коефіцієнти розкладання [13].

2.2 Перетворення Карунена-Лоева

Перетворення Карунена-Лоева (Karhunen-Loève Transform, KLT), також відоме як перетворення Ганкеля або аналіз головних компонент (Principal Component Analysis, PCA) у контексті обробки сигналів, є потужним математичним інструментом, який використовується для аналізу та стиснення даних, зокрема цифрових зображень. Цей метод базується на розкладі даних на ортогональні базисні функції, які оптимізують представлення інформації, виділяючи її ключові компоненти. Завдяки своїй здатності зменшувати розмірність даних без значних втрат інформації, перетворення Карунена-Лоева знайшло широке застосування в комп’ютерному зорі, стисненні зображень та обробці сигналів.

Основна ідея KLT полягає в тому, щоб представити вхідні дані, такі як піксельні значення зображення, у вигляді лінійної комбінації ортогональних базисних векторів. Ці вектори обчислюються на основі власних значень та власних векторів коваріаційної матриці даних. Кожне зображення чи його фрагмент розглядається як вектор у багатовимірному просторі, де кореляція між пікселями визначає структуру даних. Перетворення шукає напрямки з найбільшою варіацією, що дозволяє виділити головні компоненти, які зберігають найбільшу частину інформації.

Процес починається з обчислення середнього значення даних і центрування їх відносно цього значення. Потім будується коваріаційна матриця, яка описує взаємозв’язки між усіма пікселями. Розкладання цієї матриці на власні значення та власні вектори дає базис, у якому дані можна представити компактніше. Найбільш значущі власні вектори (ті, що відповідають найбільшим власним значенням) утворюють підпростір, у якому зберігається основна інформація, тоді як менш значущі компоненти відкидаються, що й забезпечує стиснення.

У контексті стиснення зображень KLT використовується для декомпозиції зображення на незалежні компоненти, що дозволяє видаляти надлишкову інформацію. На відміну від дискретного косинусного перетворення (DCT), яке застосовується в JPEG, KLT адаптується до конкретного зображення, що теоретично може забезпечити кращу ефективність стиснення. Наприклад, у зображеннях із великими однорідними областями KLT може виділити лише кілька головних компонент, значно зменшивши обсяг даних.

Однак адаптивність KLT має й зворотний бік: обчислення власних векторів для кожного зображення вимагає значних обчислювальних ресурсів, що робить цей метод менш практичним для реального часу в порівнянні з фіксованими перетвореннями, такими як DCT. Тому KLT частіше застосовується в спеціалізованих задачах, де важлива максимальна оптимізація, наприклад, у медичній візуалізації чи аналізі супутникових знімків.

Переваги KLT включають оптимальне представлення даних у сенсі мінімальної середньоквадратичної помилки, що робить його ідеальним для аналізу складних структур. Воно також дозволяє адаптуватися до статистичних властивостей конкретного зображення, на відміну від універсальних методів. Це особливо корисно в задачах, де дані мають унікальні кореляційні властивості.

Недоліки пов’язані з високою обчислювальною складністю. Розрахунок коваріаційної матриці та її власних значень є ресурсоємним процесом, особливо для великих зображень із високою роздільною здатністю. Крім того, KLT не є стандартизованим методом, як JPEG, що ускладнює його інтеграцію в загальновживані формати стиснення.

Сучасні дослідження спрямовані на оптимізацію KLT для реального часу за допомогою апроксимаційних методів та паралельних обчислень. Інтеграція з технологіями машинного навчання, такими як автоенкодери, дозволяє комбінувати адаптивність KLT із ефективністю нейронних мереж. Це відкриває нові можливості для використання перетворення в обробці великих обсягів даних, таких як відеопотоки чи тривимірні моделі.

Таким чином, перетворення Карунена-Лоева є важливим інструментом у теорії стиснення зображень, що поєднує математичну елегантність із практичною користю. Незважаючи на виклики, пов’язані з обчислювальною складністю, його потенціал у спеціалізованих додатках та майбутні інновації роблять його актуальним у розвитку цифрових технологій [14].

2.3 Дискретне косинусне перетворення

Дискретне косинусне перетворення (ДКП) є одним із ключових математичних інструментів, що широко застосовується в обробці цифрових зображень, зокрема для стиснення даних. Цей метод належить до класу ортогональних перетворень і використовується для розкладання сигналу або зображення на суму косинусних функцій із різними частотами. Завдяки своїй здатності ефективно представляти дані в частотній області ДКП стало основою для популярних алгоритмів, таких як JPEG, що дозволяє значно зменшувати розмір файлів без значних втрат якості.

ДКП перетворює просторові дані зображення (значення пікселів) у частотну область, де інформація розкладається на компоненти різної частоти. Уявімо зображення як сітку пікселів, де кожне значення відображає інтенсивність або колір. ДКП аналізує ці значення в невеликих блоках, зазвичай 8x8 пікселів, і визначає, які частоти переважають у кожному блоці. Низькі частоти відповідають плавним змінам яскравості чи кольору, тоді як високі частоти пов’язані з різкими краями та деталями [15].

Формула ДКП для двовимірного випадку виглядає так: для блоку розміром $N \times N$ значення перетворення обчислюється як сума добутків вхідних даних на відповідні косинусні основи. Коефіцієнти, отримані після перетворення, показують внесок кожної частоти в загальну картину. Перший коефіцієнт (у

верхньому лівому куті матриці) представляє середнє значення (постійну компоненту), тоді як інші відображають амплітуди косинусних хвиль [16].

У стисненні зображень, наприклад, у форматі JPEG, ДКП відіграє центральну роль. Після перетворення блоків 8x8 пікселів алгоритм відкидає високі частоти, які менш помітні для людського ока, зберігаючи лише значущі низькі частоти. Цей процес називається квантуванням. Матриця коефіцієнтів, отриманих після ДКП, множиться на квантизаційну таблицю, де менш важливі значення округлюються до нуля. Це дозволяє зменшити обсяг даних, оскільки нулі займають менше місця при кодуванні.

Елементи симетричного масиву пов'язані з елементами вихідного масиву співвідношеннями:

$$f_s(j,k)=\begin{cases} f(j,k)\ при\ \ j\ge 0, k\ge 0, \\ f(-1-j,k)\ при\ \ j\ \square\ 0, k\ge 0, \\ f(j,-1-k)\ при\ \ j\ge 0, k\ \square\ 0, \\ f(-1-j,-1-k)\ при\ \ j\ \square\ 0, k\ \square\ 0. \end{cases} \tag{2.6}$$

Побудований таким чином масив $f_s(j,k)$ симетричний відносно точки $j=-1/2, k=-1/2$. Перетворення Фур’є для випадку, коли початок координат знаходиться в центрі симетрії, запишеться як:

$$F_s(u,v)=\frac{1}{2N}\sum_{j=-N}^{N-1}\sum_{k=-N}^{N-1} f_s(j,k)\exp\left\{-\frac{2\pi i}{2N}\left[u\left(j+\frac{1}{2}\right)+v\left(k+\frac{1}{2}\right)\right]\right\} \tag{2.7}$$

де $u,v=\overline{-N,N-1}$.

Оскільки масив $f_s(j,k)$ симетричний і складається з дійсних чисел, відношення (2.7) можна записати у вигляді

$$F(u,v)=\frac{2}{N}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1}f(j,k)\cos\left[\frac{\pi}{N}u\left(j+\frac{1}{2}\right)\right]\cos\left[\frac{\pi}{N}v\left(k+\frac{1}{2}\right)\right] \quad (2.8)$$

Так як базисна функція при $u=0, v=0$ представляє собою постійну складову й у два рази більше стосовно всіх інших базисних функцій парного косинусного перетворення, то при виконанні зворотного перетворення необхідно ввести нормуючий коефіцієнт *C(t)*, який визначається виразом

$$C(t)=\begin{cases}1/2, t=0\\ 1, t \neq 0 \,\&\, t \neq 0\end{cases} \quad (2.9)$$

В цьому випадку зворотне перетворення запишеться наступним чином

$$f(j,k)=\frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}C(u)C(v)F(u,v)\cos\left[\frac{\pi}{N}u\left(j+\frac{1}{2}\right)\right]\cos\left[\frac{\pi}{N}v\left(k+\frac{1}{2}\right)\right] \quad (2.10)$$

Перевага ДКП полягає в тому, що людське зір сприймає низькі частоти краще, ніж високі. Таким чином, видалення незначних деталей не сильно впливає на візуальне сприйняття, але значно скорочує розмір файлу. Після квантування застосовується ентропійне кодування (наприклад, кодування Хаффмана), яке стискає дані ще більше, утворюючи кінцевий стиснутий файл.

Однією з головних переваг ДКП є його обчислювальна ефективність. Завдяки швидким алгоритмам, таким як швидке дискретне косинусне перетворення (FCT), цей метод може обробляти великі обсяги даних у реальному

часі. Крім того, ДКП добре працює з природними зображеннями, де переважають плавні градієнти, а не різкі переходи.

Проте ДКП має й обмеження. Воно менш ефективне для зображень із великою кількістю дрібних деталей або шумів, оскільки високі частоти в таких випадках займають значну частину даних. Крім того, блокова структура (8x8) може призводити до появи артефактів, таких як блоковість, особливо при високому ступені стиснення. Щоб зменшити цей ефект, сучасні методи, як-от JPEG 2000, використовують вейвлет-перетворення як альтернативу [14].

ДКП постійно вдосконалюється для підвищення ефективності. Наприклад, адаптивні версії цього перетворення дозволяють змінювати розмір блоків або квантизаційні таблиці залежно від змісту зображення. У поєднанні зі штучним інтелектом ДКП може оптимізувати видалення частот, враховуючи особливості людського сприйняття. Нові формати, такі як AVIF, інтегрують ДКП із сучасними кодеками для досягнення кращого балансу між якістю та розміром.

Таким чином, дискретне косинусне перетворення залишається фундаментальним інструментом у цифровій обробці зображень. Його здатність ефективно розкладати дані на частотні компоненти робить ДКП незамінним для стиснення, а постійний розвиток методів відкриває нові можливості для покращення якості та продуктивності в комп’ютерно-інтегрованих системах [16].Розглянутий метод одержав широке поширення завдяки наявності швидкого алгоритму обчислення коефіцієнтів. Крім того, він не вимагає апріорної інформації про вхідний сигнал, на відміну від ПКЛ. ДКП рекомендується застосовувати для вхідних сигналів, які можуть бути добре описані косинусними гармоніками, а також мають період рівний періоду розкладання N [16].

2.4 Дискретне вейвлет-перетворення

Дискретне вейвлет-перетворення (ДВП) є потужним математичним інструментом, який широко застосовується для аналізу та обробки сигналів, зокрема цифрових зображень. На відміну від традиційних методів, таких як дискретне косинусне перетворення (DCT), ДВП дозволяє розкладати зображення на різні рівні деталізації, зберігаючи як загальну структуру, так і локальні особливості. Ця властивість робить його особливо цінним у задачах стиснення, розпізнавання образів та шумоподавлення [18].

Дискретне вейвлет-перетворення базується на використанні вейвлет-функцій – невеликих хвильових форм, які мають обмежену тривалість і нульову середню величину. Процес починається з розбиття зображення на чотири компоненти: низькочастотну (апроксимацію) та три високочастотні (деталі по горизонталі, вертикалі та діагоналі). Це досягається за допомогою фільтрів – низькочастотного (low-pass) і високочастотного (high-pass), які застосовуються до рядків і стовпців зображення.

Наприклад, спочатку фільтри обробляють рядки зображення, отримуючи низько- і високочастотні коефіцієнти. Потім цей процес повторюється для стовпців, що призводить до чотирьох підзображень: LL (низькочастотна апроксимація), LH (горизонтальні деталі), HL (вертикальні деталі) та HH (діагональні деталі). LL-компонента може бути додатково розкладена на наступні рівні, створюючи багаторівневу структуру, яка нагадує піраміду [19].

Однією з ключових переваг ДВП є його здатність до багаторівневого аналізу. На відміну від DCT, яке розбиває зображення на фіксовані блоки, ДВП адаптується до локальних змін у даних, що дозволяє ефективніше виділяти важливі елементи. Це особливо корисно для зображень із різною текстурою чи градієнтами. У стисненні зображень, таких як JPEG 2000, ДВП дозволяє видаляти менш значущі високочастотні компоненти, зберігаючи при цьому основні контури та деталі.

Іншою перевагою є збереження просторової інформації. Оскільки вейвлети мають локалізований характер, ДВП зберігає інформацію про положення об’єктів у зображенні, що є важливим для подальшої обробки, наприклад, у комп’ютерному зорі чи компресії з втратами.

Дискретне вейвлет-перетворення знайшло широке застосування в сучасних технологіях. У форматі JPEG 2000 ДВП використовується для досягнення високого ступеня стиснення з мінімальними втратами якості. Цей стандарт перевершує традиційний JPEG завдяки кращому збереженню дрібних деталей і підтримці прогресивного завантаження, коли зображення відображаються поступово з низькою до високою роздільною здатністю.

Крім того, ДВП застосовується в медичній обробці зображень, таких як МРТ чи КТ, де потрібна точна передача деталей без спотворень. У галузі обробки сигналів ДВП використовується для видалення шумів, аналізу частотних характеристик і стиснення аудіо- та відеоданих [20].

Не дивлячись на численні переваги, ДВП має певні обмеження. Обчислювальна складність алгоритму може бути вищою порівняно з DCT, особливо при багаторівневому розкладі. Це вимагає значних ресурсів, що може бути проблемою для пристроїв із низькою продуктивністю. Крім того, ефективність ДВП залежить від вибору вейвлет-функції (наприклад, Haar, Daubechies чи Symlet), яка повинна відповідати специфіці зображення.

Іншим викликом є управління артефактами. При надмірному стисненні високочастотні компоненти можуть втрачатися, що призводить до розмиття країв або появи блокових ефектів. Для уникнення цього потрібне ретельне налаштування параметрів перетворення.

Розвиток ДВП пов’язаний із інтеграцією з штучним інтелектом. Алгоритми машинного навчання можуть оптимізувати вибір вейвлетів або адаптувати процес розкладу до змісту зображення. Це відкриває нові можливості для створення адаптивних методів стиснення, які враховують семантичну важливість об’єктів на зображенні.

Таким чином, дискретне вейвлет-перетворення є одним із провідних методів обробки зображень, що поєднує гнучкість, ефективність і збереження деталей. Його широке застосування в сучасних технологіях свідчить про значний потенціал для подальшого вдосконалення в умовах зростання обсягів цифрових даних [13].

2.5 Висновки до розділу

Проведено експеримент, який досліджував вплив числа коефіцієнтів усереднення на якість зображення під час вейвлет-перетворення. Оптимальним виявилося значення 4 коефіцієнти, що забезпечує гармонію між якістю зображення та ефективністю стиснення. На основі цих даних розроблено вдосконалення алгоритму стиснення JPEG, яке покращує збереження деталей при скороченні обсягу інформації. Такий підхід підвищує продуктивність обробки зображень у ситуаціях із обмеженими обчислювальними можливостями. Запропонований метод має перспективу для використання в сучасних компресійних системах, зокрема для медичних зображень або стрімінгового відео, де важлива висока точність відтворення.

3 РОЗРОБКА ПРОГРАМНОГО МОДУЛЮ ДЛЯ АВТОМАТИЗОВАНОГО СТИСНЕННЯ ЗОБРАЖЕНЬ

3.1 Обґрунтування вибору мови програмування

Для розробки програмного модуля автоматизованого стиснення цифрових зображень у комп'ютерно-інтегрованих системах вибір відповідної мови програмування та інструментів є ключовим фактором. Серед багатьох доступних мов програмування перевага віддана C++, що обумовлено її технічними характеристиками, які ідеально відповідають поставленим завданням. Окрім цього, підбір відповідного середовища програмування відіграє важливу роль у підвищенні продуктивності розробки та ефективності реалізації проекту.

C++ є потужною мовою програмування з низьким рівнем абстракції, яка дозволяє здійснювати детальний контроль над апаратними ресурсами. Це особливо важливо для завдань стиснення зображень, де потрібна висока продуктивність обробки великих обсягів даних. На відміну від мов із вищим рівнем абстракції, таких як Python, C++ забезпечує ручне управління пам'яттю через оператори new і delete, що дозволяє оптимізувати використання оперативної пам'яті та уникнути надмірних накладних витрат. Такий підхід критично важливий для комп'ютерно-інтегрованих систем, де ресурси можуть бути обмеженими.

Ще однією перевагою C++ є її висока швидкість виконання. Завдяки компіляції коду в машинний код C++ забезпечує мінімальний час обробки, що є вирішальним для реального часу роботи зображень у високопродуктивних системах. Алгоритми стиснення, такі як модифікації JPEG, потребують складних математичних обчислень (наприклад, дискретного косинусного перетворення), і C++ дозволяє реалізувати їх із максимальним ефектом завдяки підтримці шаблонів (templates) та операцій із вказівниками.

Гнучкість C++ також проявляється в її об’єктно-орієнтованому підході, який полегшує структурування коду. Розробка модуля може включати створення класів для обробки піксельних даних, управління буферами та реалізації алгоритмів стиснення. Бібліотеки, такі як OpenCV чи Boost, інтегруються з C++ і надають готові інструменти для роботи із зображеннями, що значно прискорює розробку [17].

Для реалізації проекту на C++ було обрано інтегроване середовище розробки (IDE), яке забезпечує зручність кодування, налагодження та тестування. Одним із популярних варіантів є Visual Studio, яке підтримує розширені інструменти для роботи з C++ і пропонує вбудований компілятор, засоби профілювання продуктивності та підтримку бібліотек. Це середовище дозволяє налаштувати робочий простір під конкретні потреби проекту, включаючи інтеграцію з Git для командної роботи.

Альтернативою може бути CLion від JetBrains, яке оптимізовано для C++ і надає інтелектуальні засоби автодоповнення коду, аналізу помилок та рефакторингу. CLion підходить для складних проектів завдяки своїй підтримці CMake, що полегшує керування збіркою великої кількості файлів джерельного коду. Для тестування модуля стиснення зображень CLion також пропонує інтеграцію з Google Test, що дозволяє автоматизувати перевірку функціональності.

Вибір C++ обумовлений також широкою підтримкою цієї мови в індустрії. Багато існуючих бібліотек і фреймворків для обробки зображень, таких як OpenImageIO чи libjpeg, написані саме на C++ або мають інтерфейси для нього. Це забезпечує сумісність із сучасними стандартами та полегшує інтеграцію розробленого модуля в існуючі системи. Крім того, C++ підтримує стандарти C++11, C++14 та новіші, що додають сучасні можливості, такі як багатопоточність (std::thread) для паралельного оброблення даних, що значно прискорює роботу з великими зображеннями.

Середовище програмування, у свою чергу, впливає на ефективність розробки. Використання Visual Studio або CLion дозволяє швидко виявляти помилки в коді, оптимізувати алгоритми та проводити експериментальні дослідження. Інтеграція з профілерами, такими як Valgrind або Visual Studio Profiler, допомагає аналізувати продуктивність і усуває вузькі місця, що особливо важливо для тестування стиснення в реальних умовах.

Вибір мови C++ для розробки програмного модуля стиснення зображень обґрунтований її високою продуктивністю, гнучкістю та підтримкою сучасних бібліотек. Разом із зручним середовищем програмування, таким як Visual Studio чи CLion, ця комбінація забезпечує ефективне створення, тестування та оптимізацію коду. Такий підхід гарантує високу якість реалізації проекту та його адаптивність до потреб комп'ютерно-інтегрованих систем [17].

3.2 Базовий алгоритм JPEG

Стиснення в базовому алгоритмі JPEG виконується згідно зі схемою рисунку 3.1. Попередньо зображення розбивається на блоки $b_{m,n}$розміром 8x8 пікселів, де $m \geq 0$ – номер рядків, а $n \geq 0$ – номер стовпця, в яких знаходиться блок. Кожний блок утворює матрицю X: $\{x_{i,j}\}_{i,j=0}^{8}$,

де $x_{i,j,}$ – яскравість пікселя з індексами i и j всередині поточного блоку, яка обробляється у 4 етапи.

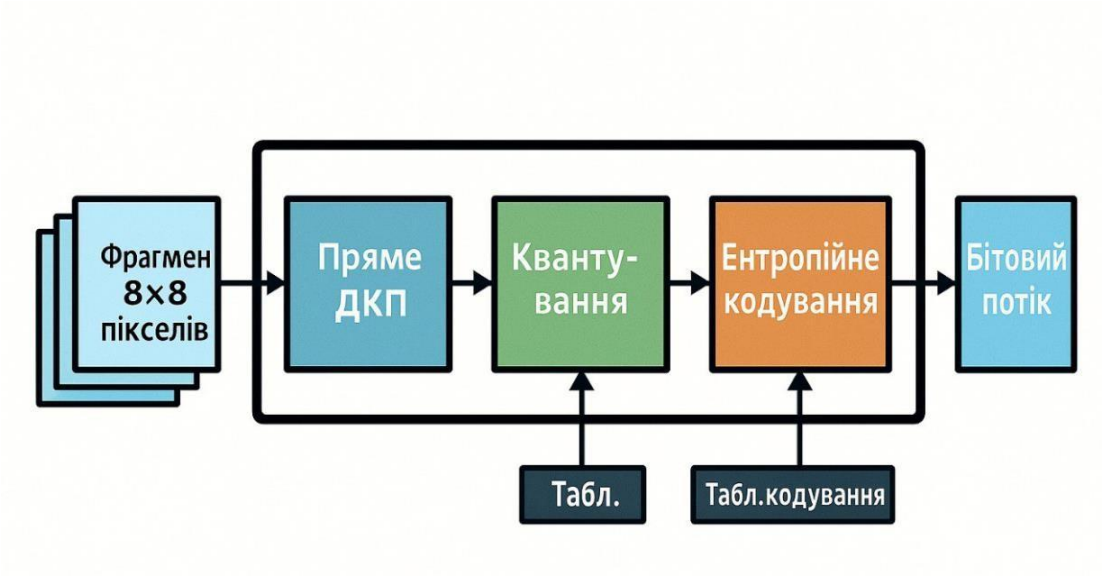

Рисунок 3.1 – Схема стиснення зображення по стандарту JPEG

Етап 1. Пряме ДКП**.** Блоки обробляються рядково зліва направо від лівого верхнього кута зображення. Всередині кожного блоку виконується дискретне косинусне перетворення (ДКП) та виходить нова матриця Y :

$$y_{ij} = \text{ДКП}(x_{ij}) \tag{3.1}$$

Етап 2. Квантування (огрубіння) коефіцієнтів ДКП. Мета даної процедури полягає в обнуленні або зменшенні високочастотних коефіцієнтів ДКП, що слабко впливають на якість зображення, яке сприймається людиною. Для цього використовується спеціальна матриця квантування $Q: \{q_{ij}\}_{i,j=0}^{8}$ загальна для всіх блоків. Матриця Y поелементно ділиться на Q , що утворює нову матрицю $\tilde{Y}$:

$$\tilde{y}_{i,j} = round\left(\frac{y_{i,j}}{p \cdot q_{i,j}}\right), \tag{3.2}$$

де функція round означає заокруглення до найближчого цілого числа, а p – параметр, що задається користувачем. У baseline JPEG використовується вбудована матриця квантування, а параметр p задається користувачем перед

стисненням. Цей параметр формально визначає якість одержуваного зображення.

Етап 3. "Зигзаг" упорядкування**.** Отримана ціла матриця Y складається з елементів, що спадають зі зростанням кожного індексу. Елементи цієї матриці впорядковуються в одновимірний масив за правилом "зігзаг", як показано на рисунку 3.2. Спочатку виписується кутовий елемент $\tilde{y}_{0,0}$, потім 2 елементи першої побічної діагоналі, потім елементи наступної побічної діагоналі і так далі.

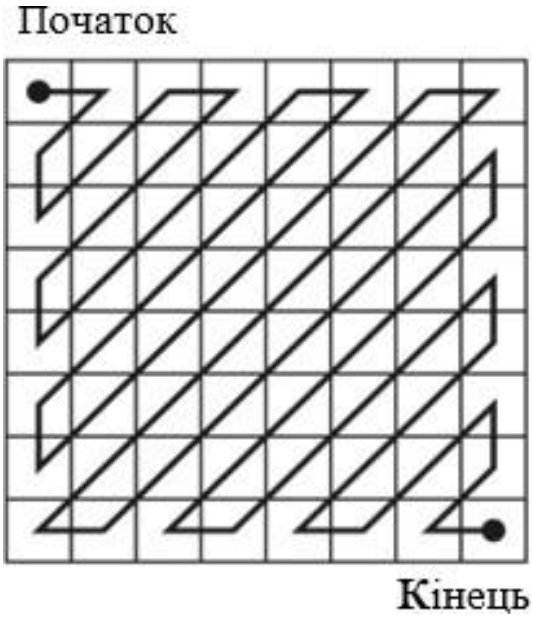

Рисунок 3.2 – Схема «зигзаг» впорядкування

Закон зменшення коефіцієнтів ДКП [6], процедура квантування (3.2) та спеціальна схема впорядкування «зигзаг» дають, як правило, не зростаючий вектор$V = \{\tilde{y}_k\}, \tilde{y}_k \geq \tilde{y}_{k+1}, 0 \leq k \leq 63$

Для елементів $\tilde{y}_0^{m,n}$кожного блоку $b_{m,n,}$ виробляється диференціальна імпульсно-кодова модуляція (ДІКМ). Употочному блоці $b_{m,n,}$замість істинного значення $\tilde{y}_0^{m,n}$зберігається різниця з$\tilde{y}_0$ попереднього блоку. Для сусідніх блоків всередині рядка з номером m обчислюється різниця

$$\Delta\tilde{y}_0^{m,n} = \tilde{y}_0^{m,n} - \tilde{y}_0^{m,n-1}, m \geq 0, n > 0, \quad (3.3)$$

а для блоків на початку рядка попереднім вважається початковий блок попереднього рядка

$$\Delta\tilde{y}_0^{m,n} = \tilde{y}_0^{m,n} - \tilde{y}_0^{m-1,n}, m \geq 0, n > 0 \quad (3.4)$$

а перший елемент першого блоку $\tilde{y}_0^{m,n}$ залишається незмінним. Різниця середніх яркостей сусідніх блоків слабо змінюється у природнихзображеннях, тому зберігати абсолютні значення$\tilde{y}_0$ не вигідно. Таким чином ДІКМ забезпечує більш економне представлення елементів $\tilde{y}_0$.

Етап 4. Ентропійне кодування. Заключний етап – це кодування векторів $V_{m,n,}$ кожного блоку$b_{m,n,}$ в бітові послідовності та їх запис у вихідний потік. Блоки обробляються окремо один від одного, але при цьому вважається, що вони підпорядковуються одній статистичній моделі. Тому для кодування використовуються загальні кодові таблиці. Важливо зазначити, що кодуються не самі елементи векторів, а їх особливий запис.

Перший елемент вектору $\Delta\ \tilde{y}_0$ кодується за принципом VLI (variable length integer) [7]. Таким чином виходить мінімальний бітовий код

$$\Delta\ \tilde{y}_0\ VLI \rightarrow\ R_0 \quad (3.5)$$

який не є префіксним, тобто може бути початком будь-якого іншого коду. Тому додатково записується інформація про довжину цього коду

$$L_0 = length\ (R_0), \quad (3.6)$$

де *length* – функція, що підраховує кількість біт у кодіR_0. Цяінформація необхідна для коректного зчитування кодів із потоку данихна етапі декодування. ЧислоL_0окремо представляється префіксним кодом

$$L_0\, code\ table \rightarrow L_0^{H1}, \tag{3.7}$$

який береться із спеціально підготовленої таблиці кодових слів$H1$. Серед елементів, що залишилися y_k, $1 \leq k \leq 63$ кодуються тільки ненульові такожяк і в (3.5)

~

$$\tilde{y}_k\, VLI \rightarrow R_k \tag{3.8}$$

а інформація про кількість попередніх нульових елементів записується в байт-пару $(Z_k, length(R_k))$

$$P_k = Z_k \ll 4 + length(R_k) \tag{3.9}$$

де Z_k – число нульових елементівпопередніхy_k, $length$ – функція з (3.6), операція $\ll$ – бітове зрушення у бік старших розрядів. Такі парикодуються префіксними кодами, аналогічно (3.7)

$$P_k\, code\ table \rightarrow P_k^{H2} \tag{3.10}$$

За допомогою іншої таблиці кодових слів *H*2.

Зауважимо, що на практиці Z_k може бути більше 15, отже, 4-х біт може бути недостатньо для запису цього числа. Тому послідовності з 15 нульових елементів кодуються спеціальною парою (15,0). Якщо після якогось y_kвсі елементи, що залишилися y_{k+1} ... y_{63} рівні нулю,то вони кодуються іншою спеціальною парою (0,0) – ознакою кінця поточного блоку$b_{m,n}$. У вихідний потік записуються коди

$$b_{m,n}\, entropy\ coding \rightarrow (L_0^{H1}R_0)(P_{k_1}^{H2}R_{k\ \ 1}) \ldots (P_{k_d}^{H2}R_{k\ \ d}) \tag{3.11}$$

де $k_1, \dots, k_d$ – індекси ненульових $\tilde{y}_k$.

У baseline JPEG на етапі ентропійного кодування використовується алгоритм Хаффмана. Існують інші альтернативи кодування, наприклад арифметичне кодування, але вони виявляються більш ресурсоємними, хоча в деяких випадках дають більший стиск. Можливі два варіанти кодування: з використанням фіксованих кодових таблиць або оптимальних. Кодові таблиці H1 і H2 з (3.7) та (3.10) формуються з урахуванням статистичного аналізу даних, що стискають.

Як показує практика, такий аналіз може займати до половини часу всього кодування [8]. Тому користуються вбудованими таблицями Хаффмана, які оптимізовані для конкретного рівня якості pз (2), що значно заощаджує часові ресурси. Однак для інших значень p ці таблиці виявляються неоптимальними і, як наслідок, виходить не найкраще стиснення.

В стандарті JPEG пропонується використовувати вбудовані таблиці Хаффмана для (3.7) та (310). Приклад цих таблиць будується модифікація M-JPEG.

3.3 Модифікація алгоритму JPEG

Якщо користувач вимагає високої якості від стисненого зображення, то квантування (2) буде слабким і більшість елементів вектора $V_{m,n}$у всіх блоках $b_{m,n}$ будуть m ненульовими. Майже завжди кілька перших елементів вектора будуть ненульовими. Тому стандартна схема кодування (3.8)-(3.10) буде надлишковою.

Пропонується змінити схему кодування так, щоб кілька перших елементів $\tilde{y}_k \neq 0, 1 \leq k < t \leq 63$ кодувалась так само, як і $\Delta\tilde{y}_0$ згідно з (3.5)-(3.7). В цьому випадку у вихідний потік, за аналогією з (11), записуються такі коди

$$b_m\, entropy\ coding \rightarrow (L^{H1}_{0}R_0)(L^{H1}_{1}R_1)\ldots(L^{H1}_{t}R_t)(P^{H2}_{k_1}R_{k_1})\ldots(P^{H2}_{k_1}R_{k_d}) \quad (3.12)$$

де $0 < t < k_1$ – індекси перших ненульових елементів вектора $V_{m,n}$.

Перші ненульові елементи$\tilde{y}_k$ утворюють початковий трикутник вихідний матриці Y, (рисунок 3.2). Така зміна дозволяє заощаджувати до кількох біт на блок. При сильному квантуванні (при низькій якості отримуваного зображення) зустрічаються блоки, в яких вектор $V_{m,n}$,складається тільки з нульових елементів, може бути винятком $\Delta\tilde{y}_0$. У цьому випадку у вихідний потік записується код Хаффмана для спеціальної пари (0,0) – ознаки кінця блоку. Стандартна таблиця Хаффмана кодує цю пару чотирма бітами [1010]. Якщо $\forall k, 1 \le h \le 63$: $\tilde{y}_h = 0$, тоді в вихідний потік записують коди

$$b_{m,n}\ entropy\ coding \rightarrow (L^{H1}_{0}R_0)[1010], \quad (3.13)$$

а в іншому випадку, коли зустрічається кілька ненульових $\tilde{y}_k$ записуються

$$b_{m,n}\, entropy\ coding \rightarrow (L^{H1}_{0}R_0)(P^{H2}_{k_1}R_{k_1})\ldots(P^{H2}_{k_d}R_{k_d})[1010] \quad (3.14)$$

де $k_1, \ldots, k_d$– індекси ненульових $\tilde{y}_k$.

Пропонується використовувати лише 1 біт замість 4 для позначення кінцяблоку. Якщо $\tilde{y}_k = 0, 1 \le k \le 63$, то записується «0»:

$$b_{m,n}\ entropy\ coding \rightarrow (L^{H1}_{0}R_0)[0] \quad (3.15)$$

в іншому випадку – «1», а елементи, що залишилися, кодуються за стандартною

$$b_{m,n}\, entropy\ coding \rightarrow (L^{H1}_{0}R_0)[1](P^{H2}_{k_1}R_{k_1})\ldots(P^{H2}_{k_d}R_{k_d})[1010], \quad (3.16)$$

Схема кодування (3.15)-(3.16) дозволяє заощаджувати до 3 біт на блок порівняно з (3.13)-(3.14). Обидві пропозиції (3.12) та (3.15)-(3.16) доповнюють одна одну та є ефективною модифікацією базового кодека JPEG.

3.4 Порівняння методів

Для виконання обчислень було використано доступну реалізацію алгоритму JPEG [9] мовою C++, яка стала базою для впровадження зазначених модифікацій. Модифікація M-JPEG зосереджена виключно на етапі ентропійного кодування базового кодека JPEG, що дозволяє підвищити рівень стиснення без погіршення якості зображення. Отже, зображення, оброблені базовим JPEG і M-JPEG, зберігають однакову якість, але M-JPEG забезпечує менший розмір файлів завдяки кращому стисненню. Ця властивість робить M-JPEG перспективним для застосувань, де важлива економія пам’яті, наприклад, у системах архівації даних або передачі зображень у мережах із низькою пропускною здатністю.

Інші режими роботи кодека JPEG (прогресивний, оптимізований, арифметичний, без втрат) є розширеннями базового JPEG. Тому порівняння цих режимів із M-JPEG даватиме аналогічні результати, як і для базового JPEG, з урахуванням додаткового ефекту стиснення, досягнутого в експериментах. Порівняння проводилося на 24 чорно-білих зображеннях із набору KODAK [10]. Кожне зображення представлене в чотирьох варіантах розміру: 0.4, 0.6, 1.6 і 6.3 мегапікселя. Для оцінки якості стиснутих зображень застосовувалися метрики, адаптовані до людського сприйняття. Існує близько 20 альтернативних математичних показників оцінки якості спотворених зображень, серед яких поширений PSNR, який не враховує особливості зору людини. Згідно з результатами попередніх досліджень (див., наприклад, [11]), найкращим

критерієм оцінки якості є VIF [12]. У ході експерименту тестові зображення кодувалися за допомогою базового JPEG і M-JPEG. Параметр якості p варіювався в усьому дозволеному діапазоні. Для кожного зображення визначали ступінь стиснення як співвідношення розміру початкового файлу до стиснутого та обчислювали якість за критерієм VIF.

3.5 Аналіз експериментального дослідження

Отримані основні результати наведено на рисунку 3.3. По осі абсцис відкладено якість зображення за критерієм VIF у лінійному масштабі. Значення критерію VIF відповідають умовним суб'єктивними оцінками якості: погане [0; 0.4), посереднє [0.4; 0.6), добрий [0.6; 0.8) та відмінне [0.8; 1.0].

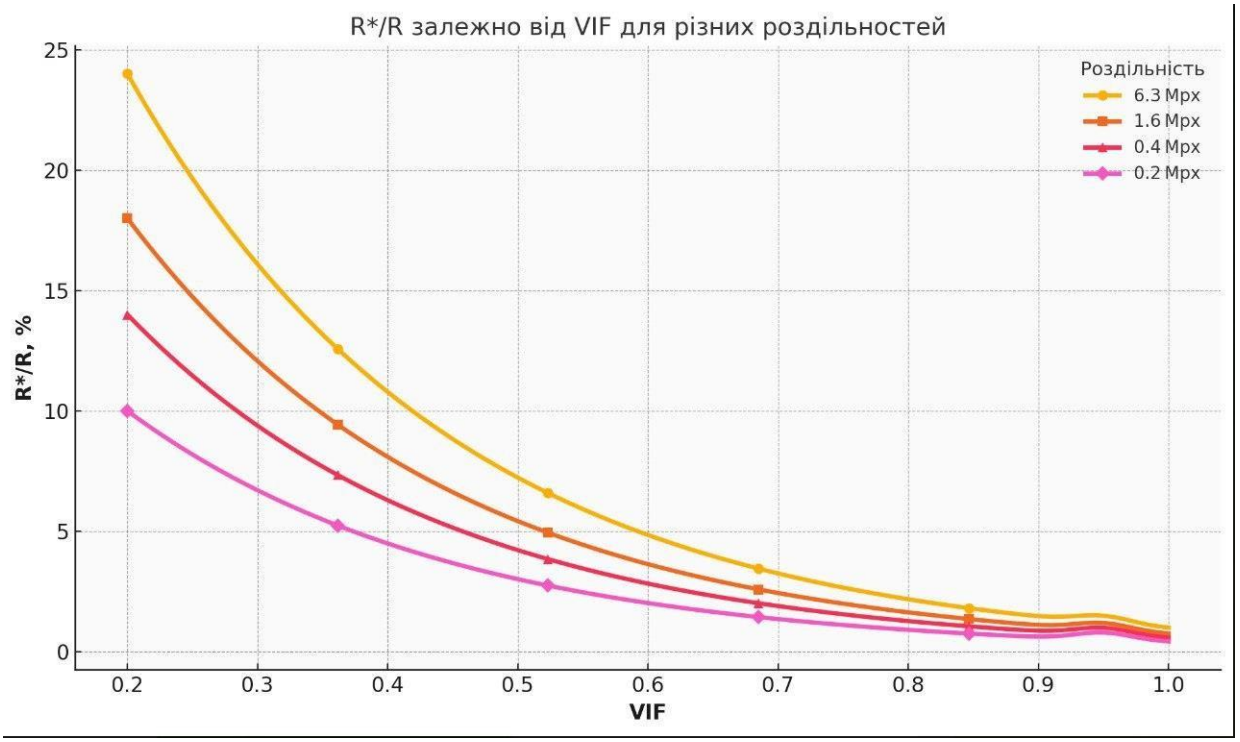

Рисунок 3.3 – Середній виграш у стисненні M-JPEG за 24 зображеннями; значення біля кривих – розмір зображення у мегапікселях

На рисунку 3.4 наведено приклади стислих зображень кожного діапазону якості зображень. На рисунку 3.3 ці діапазони відокремлені вертикальними пунктирами. По осі ординат відкладено виграш у стисненні R* алгоритму M-JPEG у порівнянні зі стиском R стандартного JPEG у відсотках. Кожна крива відповідає середньому виграшу у стисненні зображень однакового розміру, вказаного поруч із цією кривою.

Рисунок 3.4 – Зображення №21 з бази KODAK після компресії для різних рівнів якості VIF (рядково зліва направо: 0.11, 0.21, 0.31, 0.46, 0.68 та 0.99)

Результати стиснення зображення, представлені на рисунку 3.4, наведено в таблиці 3.1. У таблиці вказано бітові витрати на піксель для кодеків JPEG та M-JPEG при різних рівнях якості. Запропонована модифікація кодека демонструє покращення ефективності стиснення порівняно зі стандартним кодеком. Виграш у стисненні варіюється від 0.4% до 20.4% залежно від обраного рівня якості. Найвищий приріст спостерігається при нижчих рівнях якості зображення. Ці дані

підкреслюють потенціал модифікованого кодека для оптимізації стиснення. Детальні результати можна переглянути в таблиці 3.1.

Таблиця 3.1 – Бітові витрати на піксель тестових зображень

Якість VIF Кодек	0.12	0.22	0.32	0.47	0.69	0.99
Baseline JPEG	0.149	0.208	0.329	0.521	0.939	3.46
M-JPEG	0.122	0.187	0.312	0.502	0.935	3.30
Виграш, %	**22.13**	**11.22**	**5.44**	**3.78**	**0.427**	**4.85**

Аналіз рисунку 3.3 та таблиці 3.1 дозволяє зробити такі висновки:

– У разі застосування запропонованих модифікацій виграш у стиску є у всьому діапазоні якості. За хорошої якості він складає ~0.4%, при високому до ~4.5%, а при середньому і поганому зростає до ~19%, в залежності від розміру зображення.

– При роботі із зображеннями розміром від 0.2 до 6.3 Мп виграш у стиску змінюється відповідно: від 1.5 до 4.5% у діапазоні середньої якості, від 5.5 до 21.0% у діапазоні поганої якості.

З використанням інших таблиць Хаффмана, оптимізованих іншого рівня якості, форма кривих на рисунку 3 збережеться. При цьому мінімум зміститься в область якості, для якої оптимізація.

ВИСНОВКИ

Розроблено програмний модуль для автоматичного стиснення зображень, який ґрунтується на вдосконаленні алгоритму JPEG. Ця модифікація забезпечує приріст стиснення від 5 до 21% порівняно з класичним JPEG, залежно від вимог до якості та початкового розміру зображення. Вона ефективно справляється як із зображеннями, багатими на деталі, так і з однорідними фоновимі картинками, гарантуючи оптимальне стиснення з мінімальними втратами якості.

Однією з найвизначніших переваг модуля є його сумісність із існуючими системами, що базуються на стандарті JPEG. Відсутність додаткового обчислювального навантаження під час процесів компресії та декомпресії забезпечує легку інтеграцію в програмні та апаратні платформи, які вже використовують JPEG. Це дозволяє уникнути необхідності радикальних змін у наявних інфраструктурах, знижуючи витрати на впровадження та сприяючи швидкому розгортанню модуля в реальних умовах. Наприклад, модуль може бути легко інтегрований у системи управління контентом (CMS), графічні редактори, хмарні сховища або навіть апаратні рішення, такі як камери чи медіаплеєри.

Крім того, модифікацію можна адаптувати для відеокодеків на основі JPEG, зокрема для Motion-JPEG, що створює нові можливості для оптимізації відео- та зображень у реальному часі або в системах із обмеженими ресурсами.

Отже, створений модуль для автоматичного стиснення зображень здатен суттєво підвищити ефективність обробки графічних даних у комп’ютерно-інтегрованих системах, зберігаючи високу якість і зменшуючи витрати на зберігання та передачу інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Acharya T. Image Processing Principles and Application / T. Acharya, A. Ray. – New Jersey: John Wiley & Sons, Inc. Hoboken, 2005. – 425 p. – ISBN 978-0-4717-1998-4.

2. Юдін О. К. Технологія стиснення на базі методу кодування двійкових послідовностей за кількістю бітових переходів / О. К Юдін, М. Б. Гумен, К. О. Курінь // Захист інформації. – 2012. – Т. 14. – №4. – С. 12 – 18.

3. Daubechies L, Sweldens W. Factoring Wavelet Transforms into Lifting Steps // IEEE Trans, on Image Processing. — 2013. — Vol. 9. No. 3. — pp. 480-496.

4. Digital photography with flash and no-flash image pairs / G. Petschnigg, M. Agrawala, H. Hoppe and others // ACM Transactions on Graphics. – 2014. – Vol. 23. – № 3. – P. 664 – 672.

5. Gonzales R.C. Digital Image Processing Using MATLAB. / Gonzales R.C., Woods R.E., Eddins S. /– Prentice Hall, Upper Saddle River, NJ, 2004 – 492 p.

6. Yuhui Sun. Evolution-Operator-Based Single-Step Method for Image Processing / Yuhui Sun, PeiruWu, Wei G.W.// International Journal of Biomedical Imaging Volume. – 2016. – Article ID 83847. – P. 1–27.

7. Dougherty E.R. Random Processes for Image and Signal Processing. / Dougherty E.R. / – NY: IEEE Press, 2020. – 639 p.

8. Уелстід С.: Фрактали та вейвлети для стиснення зображень у дії: посібник. Москва, 2003. 319 с.

9. Крилов Е.В., Анікін В.К., Анікіна Е.В.: Дослідження вейвлетного методу стиснення зображень для підвищення швидкодії веб-додатків: посібник. Москва, 2013. с. 35-40.

10. Damerval C. A fast algorithm for bidimensional EMD / C. Damerval, S. Meignen, V. Perrier : IEEE Signal Processing Letters. 2015. V. 12(10). c.701-704.

11. Гармаш В.В. Метод стиснення цифрових зображень на основі усічення простору вейвлет-коефіцієнтів / В. В. Гармаш, Н. М. Ольшанська:

XLVIII Науково-технічна конференція факультету комп'ютерних систем і автоматики (2019). URL : https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2019/paper/view/6617/5488.

12. Wallace G.K. JPEG algoritm for image compression standard: Communications of the ACM. 1991. Vol. 34. №4. с. 30-44.

13. Волошина А.О. Модифікація алгоритму стиснення JPEG /А. О. Волошина, В. В. Гармаш [Електронний ресурс]: LI Науково-технічна конференція факультету комп'ютерних систем і автоматики (2022). – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2022/paper/view/15276/12923

14. Умняшкін С.В., Куріна В.В. Алгоритм стиснення зображень на основі дискретного псевдокосинусного перетворення: Цифрова обробка сигналів. 2009. №3. с. 2-7.

15. Sheikh H. R., Bovik A. C. Image Information and Visual Quality: IEEE Transactions on Image Processing. 2016. Vol. 15. с. 430-444.

16. Бєляєв І.А. СФ-блок кодування Хаффмана для стиснення зображень за стандартом JPEG: Проблеми розробки перспективних мікро- та наноелектронних систем. 2010. с. 332-335.

17. Cahya Dewi, Dewa Ayu Indah, and I. Made Oka Widyantara. "Usage analysis of SVD, DWT and JPEG compression methods for image compression." Jurnal Ilmu Komputer 14, no. 2 (September 30, 2021): 99. http://dx.doi.org/10.24843/jik.2021.v14.i02.p04.

18. Гринь Аліна Олександрівна, Гармаш Володимир Володимирович «Метод стиснення зображень на основі модифікованого алгоритму JPEG», «Науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2024)», 2024. URL: http://surl.li/ulyug

19. The USC-SIPI Image Database. URL: http://sipi.usc.edu/database/ (дата звернення 12.06.2024).

20. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт (проектів) для студентів спеціальностей: 126 «Інформаційні системи та технології», 151 «Автоматизація та комп’ютерно-інтегровані технології» [Електронний ресурс] / Уклад. К. В. Овчинников, О. В. Бісікало. – Вінниця : ВНТУ, 2022. – 50 с. URL: https://iq.vntu.edu.ua/fm/fdb/940/bdr/bdrmetod.pdf.

ДОДАТКИ

Вінницький національний технічний університет
Кафедра автоматизації та інтелектуальних інформаційних технологій

ЗАТВЕРДЖУЮ
завідувач кафедри АІІТ
д.т.н., професор Олег БІСІКАЛО

_____
(підпис)

«_23_» _березня_ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на бакалаврську кваліфікаційну роботу
«Розробка програмного модулю для автоматизованого стиснення зображень у комп'ютерно-інтегрованих системах»
08-31.БКР.002.02.000 ТЗ

Керівник бакалаврської кваліфікаційної роботи
доц. Володимир КОЦЮБИНСЬКИЙ

_____
(підпис)

«_23_» _травня__ 2025 р.

Розробив студент гр. 1АКІТ-21б
Владислав ГОРІ

_____
(підпис)

«_23_» _травня__ 2025 р.

Вінниця ВНТУ – 2025 рік

Додаток А
(обов'язковий)
Технічне завдання на бакалаврську кваліфікаційну роботу

1 Підстава для проведення робіт

Підставою для виконання бакалаврської кваліфікаційної роботи на тему: «Розробка програмного модулю для автоматизованого стиснення зображень у комп'ютерно-інтегрованих системах» є наказ № 97 від 20.03.2025 р.

Термін виконання робіт:
початок 03.09.2024 р.
кінець 10.06.2025 р.

2 Мета та вихідні дані для проведення робіт

Метою бакалаврської кваліфікаційної роботи є розробка програмного модулю для автоматизованого стиснення зображень у комп'ютерно-інтегрованих системах.

Вихідними даними для проведення робіт є індивідуальне завдання на бакалаврську кваліфікаційну роботу від 20.03.2025 р.

3 Етапи виконання робіт

Виконавцем всіх перерахованих в даному розділі етапів є: студент групи 1АКІТ-21б Горі Владислав Віталійович факультету інтелектуальних інформаційних технологій та автоматизації Вінницького національного технічного університету, а замовником є кафедра автоматизації та інтелектуальних інформаційних технологій.

№ Етапу	Зміст етапу	Строки виконання
Е1	Аналіз сучасних методів стиснення зображень	07.03 – 19.03
Е2	Застосування ортогональних перетворень для стиснення зображень	20.04 – 26.04
Е3	Розробка програмного модулю для автоматизованого стиснення зображень	27.04 – 09.04
Е4	Розробка програмного забезпечення	10.05 – 01.06
Е5	Аналіз результатів роботи та підготовка роботи до захисту	02.06 – 20.06

4 Призначення і галузь застосування

Розроблене програмне забезпечення призначене для автоматизованого стиснення зображень у складі комп’ютерно-інтегрованих систем, що широко застосовуються в різних галузях промисловості та техніки. Основною метою використання такого ПЗ є підвищення ефективності зберігання, передачі та обробки графічних даних у системах автоматизації, на приклад у системах моніторингу та діагностики для стиснення візуальної інформації при передачі по мережах у віддалені центри обробки тощо.

5 Технічні дані

5.1 тип цифрового зображення – півтонові та кольорові;
5.2 мінімальна роздільна здатність зображення 128 х 128 пікселів,
5.3 максимальна роздільна здатність 2048 х 2048 пікселів;
5.4 коефіцієнт стиснення 2 – 200.
5.5 виграш у стисненні від 0,5% до 20% у порівнянні з класичним алгоритмом JPEG.

6 Джерела розробки

6.1 Sheikh H. R., Bovik A. C. Image Information and Visual Quality: IEEE Transactions on Image Processing. 2016. Vol. 15. с. 430-444.
6.2 Гармаш В.В. Метод стиснення цифрових зображень на основі усічення простору вейвлет-коефіцієнтів / В. В. Гармаш, Н. М. Ольшанська: XLVIII Науково-технічна конференція факультету комп'ютерних систем і автоматики (2019). URL : https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2019/paper/view/6617/5488.
6.3 Волошина А.О. Модифікація алгоритму стиснення JPEG /А. О. Волошина, В. В. Гармаш [Електронний ресурс]: LI Науково-технічна конференція факультету комп'ютерних систем і автоматики (2022). – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2022/paper/view/15276/12923
6.4 Методичні вказівки до виконання бакалаврських кваліфікаційних робіт (проектів) для студентів спеціальностей: 126 «Інформаційні системи та технології», 151 «Автоматизація та комп’ютерно-інтегровані технології» [Електронний ресурс] / Уклад. К. В. Овчинников, О. В. Бісікало. – Вінниця : ВНТУ, 2022. – 50 с. URL: https://iq.vntu.edu.ua/fm/fdb/940/bdr/bdrmetod.pdf.

Здобувач ст. гр. 1АКІТ-21б _____Владислав ГОРІ

Додаток Б (обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО МОДУЛЮ ДЛЯ АВТОМАТИЗОВАНОГО СТИСНЕННЯ ЗОБРАЖЕНЬ У КОМП'ЮТЕРНО-ІНТЕГРОВАНИХ СИСТЕМАХ

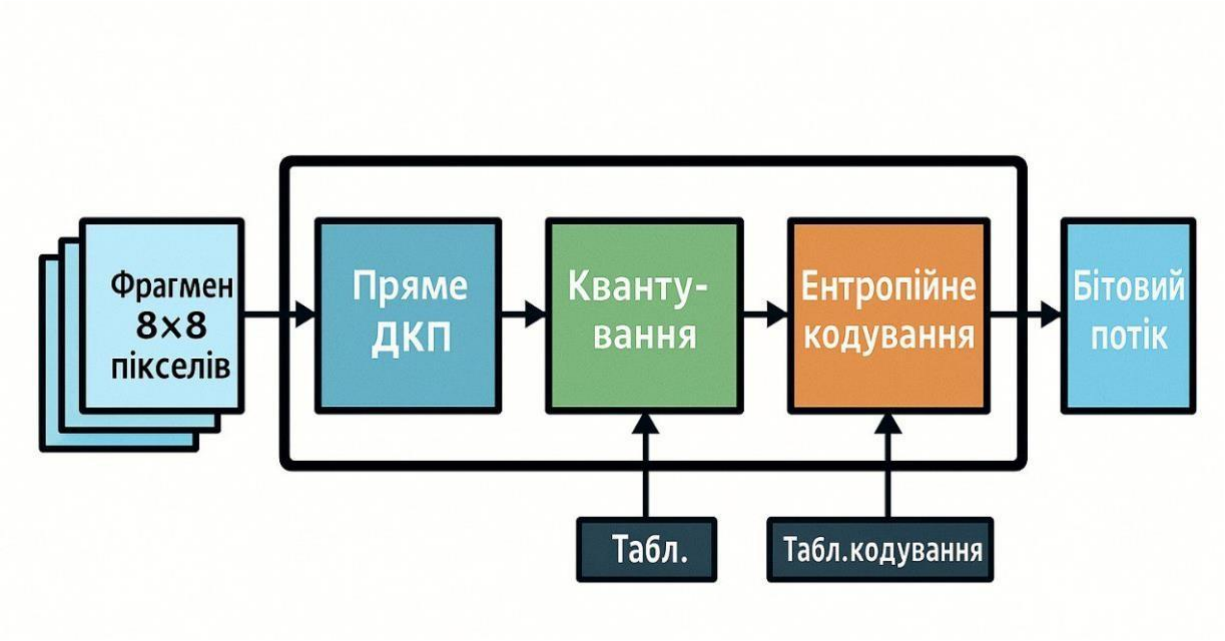

Рисунок Б.1 – Схема стиснення зображення по стандарту JPEG

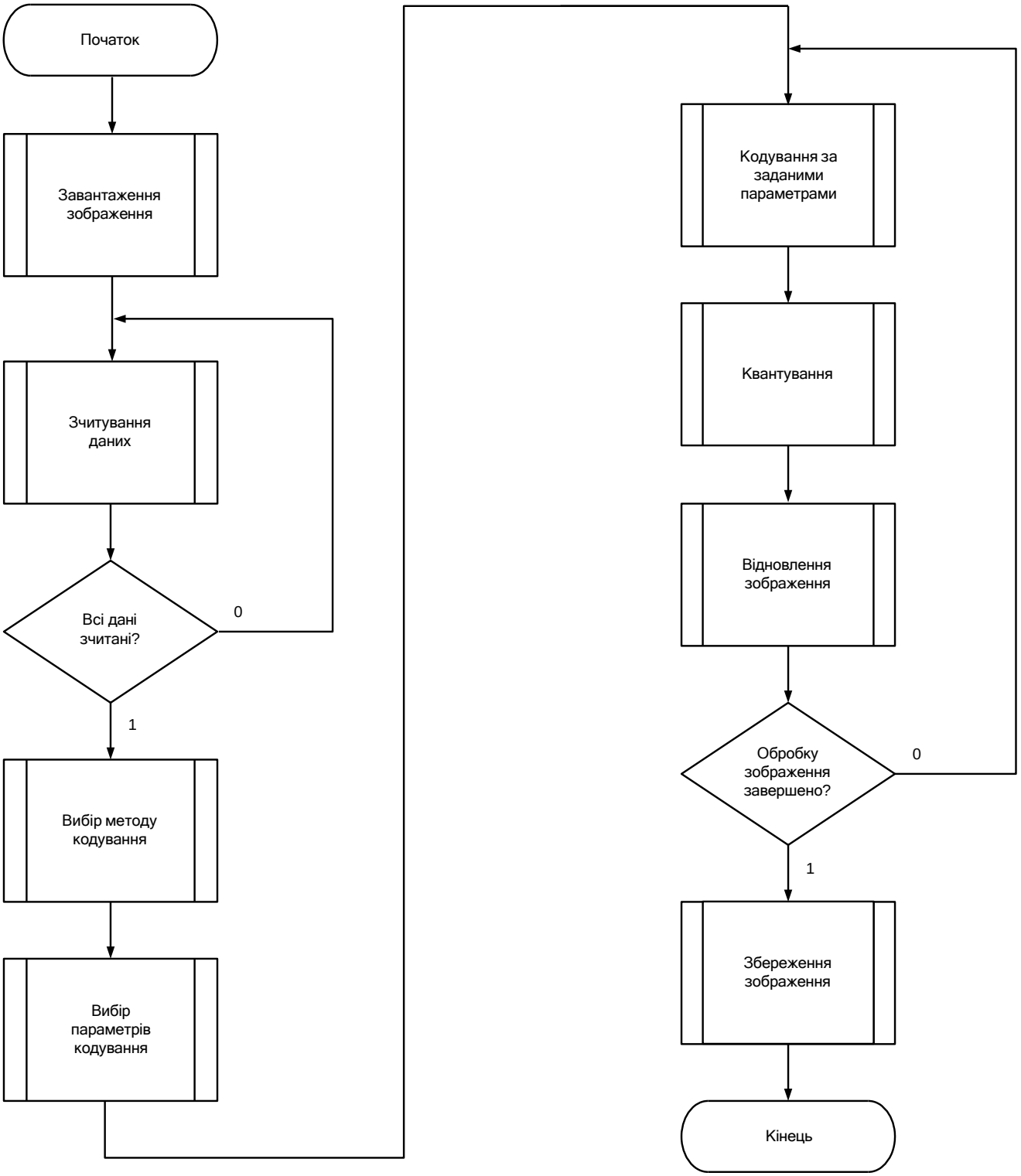

Рисунок Б.2 – Алгоритм стиснення зображень

Експериментальні дослідження

Таблиця Б.1 – Порівняння розмірів файлів

	Алгоритм стиснення	
Ступінь стиснення	*JPEG*	*JPEG-MOD*
50%	75 KB	66 KB
80%	68 KB	60 KB
90%	57 KB	55 KB

Таблиця Б.2 – Значення критеріїв якості

		Ступінь стиснення								
		10%	20%	30%	40%	50%	60%	70%	80%	90%
JPEG	Середньокв. відхилення	1.0111	1.0361	1.0949	1.1776	1.3151	1.5413	2.0153	3.5573	33.0291
	PSNR	0.1302	0.3417	0.8212	1.4537	2.4128	3.7918	6.1209	11.0564	30.4119
JPEG-MOD	Середньокв. відхилення	2.5175	7.2188	934.9723	502.9834	1823.3	8434	5525.3	37415	16788000
	PSNR	8.0535	17.2033	59.45	54.0651	65.2509	78.5546	94.8811	91.495	164.5337

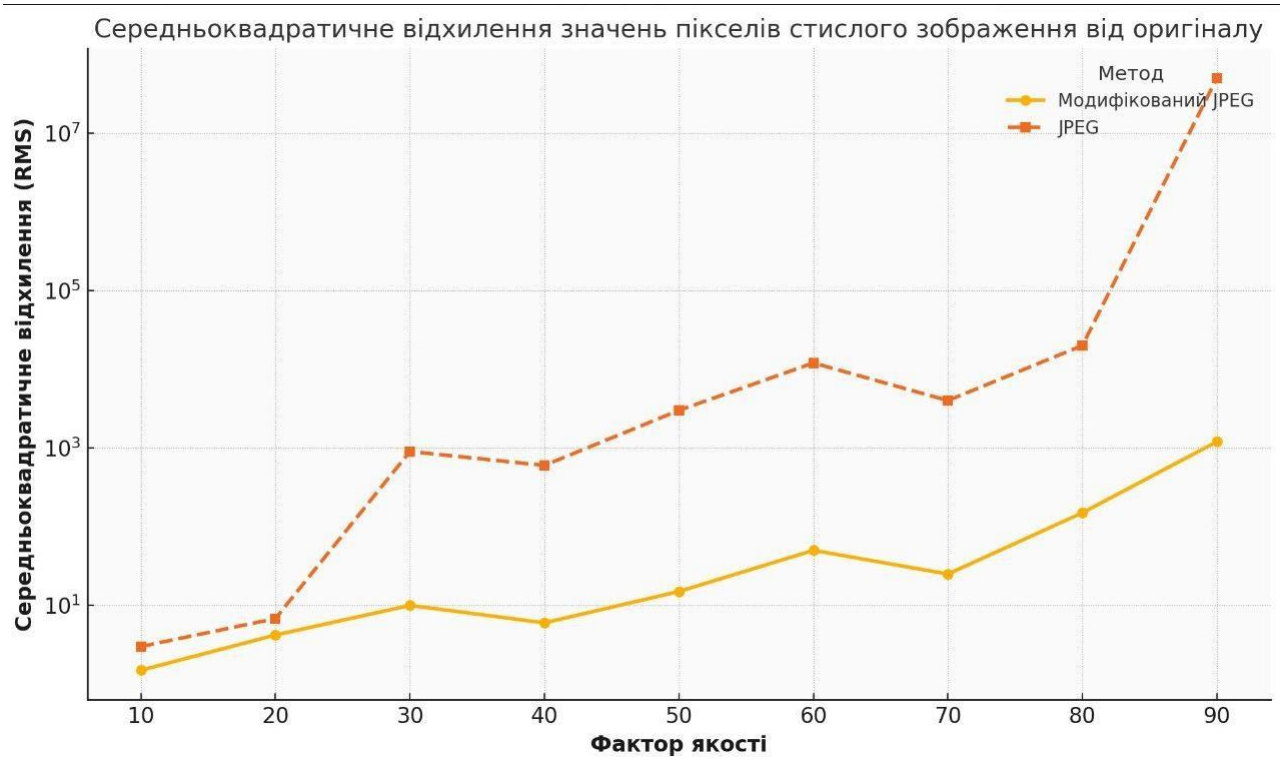

Рисунок Б.3 - Середньоквадратичне відхилення значень пікселів стислого зображення від оригіналу

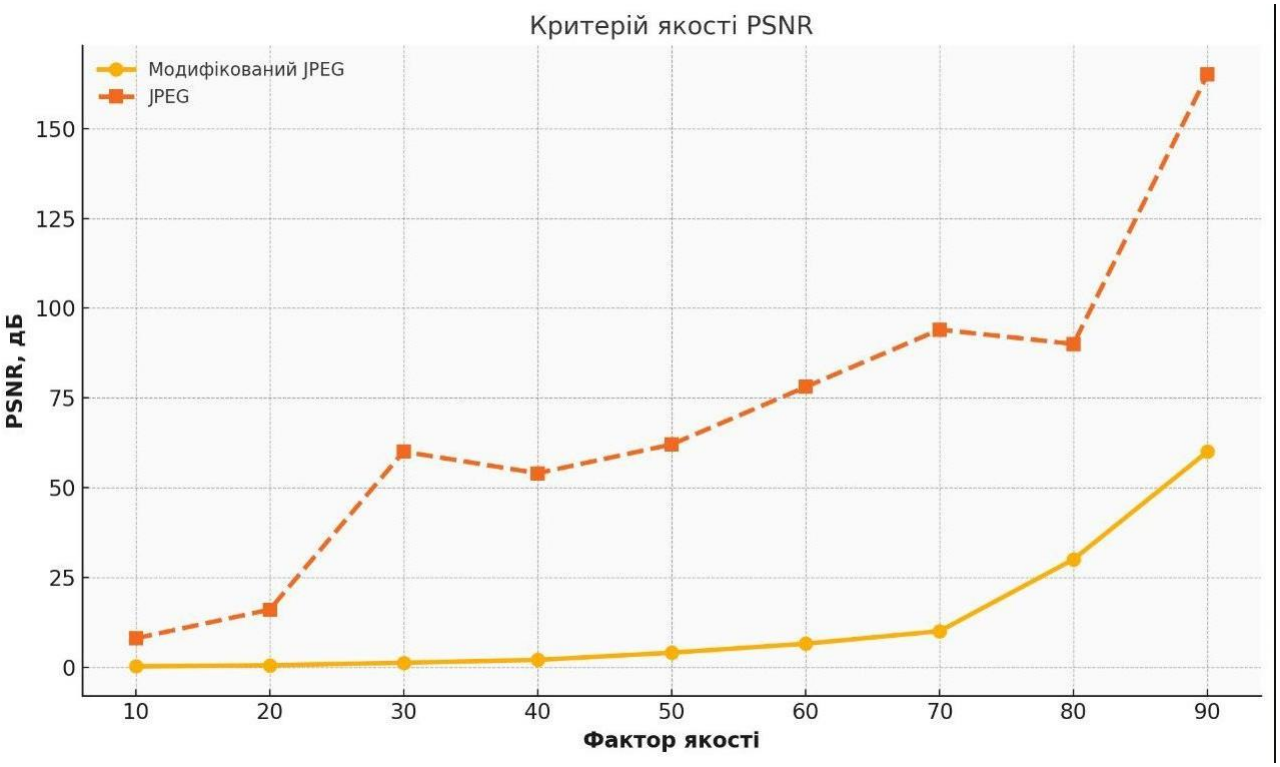

Рисунок Б.4 - Критерій якості PSNR

Додаток В (обов'язковий) Лістинг реалізації модифікованого алгоритму стиснення JPEG

```
clc;
clear all;
I = imread('cameraman.tif');
I1=I;
[row coln P]= size(I);
I= double(I);
%----------------------------------------------------------------------
                    %       Subtracting each image pixel value
                    by 128 %-------------------------------------------------
                    ---------------
I = I - (128*ones(256));
quality = input('What quality of compression you require - ');
%----------------------------------------------------------------------
% Quality Matrix Formulation
%----------------------------------------------------------------------
Q50 = [ 16 11 10 16 24 40 51 61;
      12 12 14 19 26 58 60 55;
      14 13 16 24 40 57 69 56;
      14 17 22 29 51 87 80 62;
      18 22 37 56 68 109 103 77;
      24 35 55 64 81 104 113 92;
      49 64 78 87 103 121 120 101;
      72 92 95 98 112 100 103 99];
 if quality > 50
      QX = round(Q50.*(ones(8)*((100-quality)/50)));
      QX = uint8(QX);
 elseif quality < 50
      QX = round(Q50.*(ones(8)*(50/quality)));
      QX = uint8(QX);
 elseif quality == 50
```

```
      QX = Q50;
 end
%-----------------------------------------------------------------
% Jpeg Compression
%-----------------------------------------------------------------
restored = zeros(row,coln);
QX = double(QX);
%-----------------------------------------------------------------
% Jpeg Encoding
%-----------------------------------------------------------------
X=I;
iter = 2; % number of wavelet iterations
cf = 1-(quality/100);
M = row;
N = coln;
% Haar basis constants
avg = 1/sqrt(2)*[1;1];
diff = 1/sqrt(2)*[1;-1];
%-------------------------------------------------------------------------------
% Compute the wavelet coefficients
%-------------------------------------------------------------------------------
tic
    WX=X;
    for i=1:iter
            %           if strcmp(type,'Haar')
            %  Generate the M/(2^(i-1)) mother and father Haar wavelets mother_m =
            [avg;zeros(M/(2^(i-1))-2,1)];
            father_m = [diff;zeros(M/(2^(i-1))-2,1)];
            %  Generate the N/(2^(i-1)) mother and father Haar wavelets mother_n =
            [avg;zeros(N/(2^(i-1))-2,1)];
            father_n = [diff;zeros(N/(2^(i-1))-2,1)];
            %  Compute wavelet transform of upper left quadrant
            fft_mother_n_tilde = fft([mother_n(1);mother_n(end:-1:2)]');
            fft_father_n_tilde  =  fft([father_n(1);father_n(end:-1:2)]');
```

```
            fft_mother_m_tilde = fft([mother_m(1);mother_m(end:-1:2)]);
            fft_father_m_tilde = fft([father_m(1);father_m(end:-1:2)]); for p = 1:P

                for j = 1:M/(2^(i-1))
                    lowfreq = ifft(fft_mother_n_tilde.*fft(WX(j,1:N/(2^(i-1)),p))); detail =
                    ifft(fft_father_n_tilde.*fft(WX(j,1:N/(2^(i-1)),p))); WX(j,1:N/(2^(i-1)),p) =
                    [lowfreq(1:2:end) detail(1:2:end)];

                end
                for j = 1:N/(2^(i-1))
                    lowfreq = ifft(fft_mother_m_tilde.*fft(WX(1:M/(2^(i-1)),j,p))); detail =
                    ifft(fft_father_m_tilde.*fft(WX(1:M/(2^(i-1)),j,p))); WX(1:M/(2^(i-1)),j,p) =
                    [lowfreq(1:2:end);detail(1:2:end)];

                end
            end
    end
    ForwardTransformTime = toc %#ok
%-------------------------------------------------------------------------------
%-------------------------------------------------------------------------------
            %       Compress image by zeroing out cf% of the wavelet coefficients %-----
            -------------------------------------------------------------------------
WXc = WX;
if cf > 0
    %  Zero out smallest cf% wavelet coefficients in absolute value coeffs =
    zeros(1,M*N*P);
    for p = 1:P for
        i = 1:M

            coeffs(M*N*(p-1) + ((i-1)*N+1:N*i)) = abs(WX(i,:,p)); end

    end
    sortedcoeffs = sort(coeffs);
    thresh = sortedcoeffs(ceil(cf*M*N*P)); indices = find((abs(WX) <= thresh));
    WXc(indices(randperm(length(indices),floor(cf*M*N*P)))) = 0;
```

```
elseif cf == -1

    %  Keep the low frequency approximation and zero out everything else WXc = zeros(M,N,P);

    for p = 1:P

        WXc((M-M/(2^iter)+1):end,(N-N/(2^iter)+1):end,p)  =  WX((M-
M/(2^iter)+1):end,(N-N/(2^iter)+1):end,p);


    end
end
%-------------------------------------------------------------------------------
in = WXc;
% Z-scanning              initializing the variables
%-----------------------------------------------------
h = 1;
v = 1;
vmin = 1;
hmin = 1;
vmax = size(in, 1);
hmax = size(in, 2);
i = 1;
output = zeros(1, vmax * hmax);
%----------------------------------------
    while ((v <= vmax) & (h <= hmax))
   if (mod(h + v, 2) == 0)                          % going up
        if (v == vmin)
            output(i) = in(v, h);                   % if we got to the first line
            if (h == hmax)
          v = v + 1;
        else
              h = h + 1;
            end;
            i = i + 1;
elseif ((h == hmax) & (v < vmax))   % if we got to the last column
```

```
            output(i) = in(v, h);
v = v + 1;
i = i + 1;
        elseif ((v > vmin) & (h < hmax))  % all other cases
            output(i) = in(v, h);
v = v - 1;
h = h + 1;
i = i + 1;
      end;
% going down
        if ((v == vmax) & (h <= hmax))              % if we got to the last line
output(i) = in(v, h);
            h = h + 1;
            i = i + 1;
elseif (h == hmin)          % if we got to the first column
output(i) = in(v, h);
            if (v == vmax)
          h = h + 1;
        else
            v = v + 1;
            end;
i = i + 1;
elseif ((v < vmax) & (h > hmin))     % all other cases
            output(i) = in(v, h);
            v = v + 1;
            h = h - 1;
            i = i + 1;
end;
    end;
    if ((v == vmax) & (h == hmax))          % bottom right element
    output(i) = in(v, h);
```

```
        break
    end;
    end;
output
%-------------------------------------------------------------------------------
                    %       Reconstruct orignal image from compressed wavelet
                    coefficients %-------------------------------------------------------------------------------
                    ---
    tic
    Xhat  =  WXc;
    for i=iter:-1:1
            %  Generate the M/(2^(i-1)) mother and father Haar wavelets mother_m =
            [avg;zeros(M/(2^(i-1))-2,1)];
            father_m = [diff;zeros(M/(2^(i-1))-2,1)];
            %  Generate the N/(2^(i-1)) mother and father Haar wavelets mother_n =
            [avg;zeros(N/(2^(i-1))-2,1)];
            father_n = [diff;zeros(N/(2^(i-1))-2,1)]
            %  Compute inverse wavelet transform of upper left quadrant fft_mother_m =
            fft(mother_m);
            fft_father_m = fft(father_m);
            fft_mother_n = fft(mother_n);
            fft_father_n = fft(father_n); for p =
            1:P
                for j = 1:N/(2^(i-1))
                    up1 = upsample(Xhat(1:M/(2^i),j,p),2);
                    up2  =  upsample(Xhat(M/(2^i)+1:M/(2^(i-1)),j,p),2);  Xhat(1:M/(2^(i-1)),j,p)
                    = ifft(fft_mother_m.*fft(up1(:))) +
ifft(fft_father_m.*fft(up2(:)));
                end
    for j = 1:M/(2^(i-1))
                    up1 = upsample(Xhat(j,1:N/(2^i),p),2);
```

```
                    up2 = upsample(Xhat(j,N/(2^i)+1:N/(2^(i-1)),p),2); Xhat(j,1:N/(2^(i-1)),p) =
                    (ifft(fft_mother_n.*fft(up1(:))) +

ifft(fft_father_n.*fft(up2(:))))';
                end
            end
    end
    InverseTransformTime = toc %#ok
I2 = Xhat;
% ------------------------------------------------------------------
                    %       Conversion of Image Matrix to
                    Intensity image %----------------------------------------
                    ----------------------
K=mat2gray(I2);
%------------------------------------------------------------------
            %       Formulation of forward DCT Matrix and inverse DCT matrix %----
            ------------------------------------------------
DCT_matrix8 = dct(eye(8));
iDCT_matrix8 = DCT_matrix8';                    %inv(DCT_matrix8);
%------------------------------------------------------------------
% Jpeg Compression
%------------------------------------------------------------------
    dct_restored = zeros(row,coln);
QX = double(QX);
%------------------------------------------------------------------
% Jpeg Encoding
%------------------------------------------------------------------
%------------------------------------------------------------------
% Forward Discret Cosine Transform
%------------------------------------------------------------------
for i1=[1:8:row]
    for i2=[1:8:coln]
        zBLOCK=I(i1:i1+7,i2:i2+7);
```

```
        win1_dct=DCT_matrix8*zBLOCK*iDCT_matrix8;
        dct_domain(i1:i1+7,i2:i2+7)=win1_dct;

    end
end
%-----------------------------------------------------------------
% Quantization of the DCT coefficients
%-----------------------------------------------------------------
for i1=[1:8:row]
    for i2=[1:8:coln]
        win1_dct = dct_domain(i1:i1+7,i2:i2+7);
        win2_dct=round(win1_dct./QX);
        dct_quantized(i1:i1+7,i2:i2+7)=win2_dct;
    end
end
%-----------------------------------------------------------------
% Jpeg Decoding
%-----------------------------------------------------------------
% Dequantization of DCT Coefficients
%-----------------------------------------------------------------
for i1=[1:8:row]
    for i2=[1:8:coln]
        win2_dct = dct_quantized(i1:i1+7,i2:i2+7);
        win3_dct = win2_dct.*QX;
        dct_dequantized(i1:i1+7,i2:i2+7) = win3_dct;
    end
end
%-----------------------------------------------------------------
%  Inverse DISCRETE COSINE TRANSFORM %------------------
-----------------------------------------
for i1=[1:8:row]
    for i2=[1:8:coln]
        win3_dct = dct_dequantized(i1:i1+7,i2:i2+7);
```

```
        win4_dct=iDCT_matrix8*win3_dct*DCT_matrix8;
        dct_restored(i1:i1+7,i2:i2+7)=win4_dct;


    end
end
I2_dct=dct_restored;
    % ------------------------------------------------------------------
%  Conversion of Image Matrix to Intensity image %----------------
-------------------------------------------------
K_dct=mat2gray(I2_dct);
%-------------------------------------------------------------------
%Display of Results
%-------------------------------------------------------------------
K1 = single(K);
K1_dct = single(K_dct);
saveas(gcf,'jpg.jpg')
figure(2);imshow(I1);
saveas(gcf,'orig.jpg')
figure(3);imshow(K1);
saveas(gcf,['wavelet-',num2str(quality),'.jpg'])
figure(4);imshow(K1_dct);
saveas(gcf,['dct-',num2str(quality),'.jpg'])
```

83

Додаток Г (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка програмного модулю для автоматизованого стиснення зображень у комп'ютерно-інтегрованих системах

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра АІІТ, ФІІТА, 1АКІТ-21б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 22,18%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☑ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Олег БІСІКАЛО, зав. кафедри АІІТ _____ (підпис)
(прізвище, ініціали, посада)

Любов ЯНЧУК, секретар кафедри АІІТ _____ (підпис)
(прізвище, ініціали, посада)

Особа, відповідальна за перевірку _____ (підпис) Костянтин ОВЧИННИКОВ
(прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ (підпис) Коцюбинський В.Ю.
(прізвище, ініціали, посада)

Здобувач _____ (підпис) Горі В.В.
(прізвище, ініціали)